A STUDY OF VIDEO FIRE DETECTION AND ITS APPLICATION

WONG KWOK KEUNG ARTHUR

PhD

The Hong Kong Polytechnic University

2018

The Hong Kong Polytechnic University

Department of Building Services Engineering

A Study of video fire detection and its application

Wong Kwok Keung Arthur

A thesis submitted in partial fulfilment of the

requirements for the Degree of Doctor of Philosophy

October 2016

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

______________________________ (Signed)

Wong Kwok Keung Arthur (Name of student)

Abstract

In order to save life and property from fire, early and accurate fire detection is one of the important aspects in fire safety design. For traditional spot type and line type fire detectors, information concerning the fire parameters such as flame height, fire growth rate and fire location are difficult to obtain. Such information is very useful especially to enable effective fire evacuation and firefighting. In addition, traditional fire detection technology has 8 – 10% false alarm rates. With the fast development of computer technology and image processing techniques, it is now possible to use video images to detect fire in different locations and obtain those fire parameters. It is able to supplement traditional fire detection technology so the development of video fire detection is becoming important. Besides, it is also useful in fire safety design for outdoor environments. This thesis presents the study of video fire detection and its application.

Based on the literature review, video fire detection systems have already been developed to detect forest fires. Four kinds of video fire detection analysis methods including the digital image processing method, statistical colour model method, artificial neural network method and combined different approaches are used. The primary objective for the development of different analysis methods is to enhance the accuracy of video fire detection. In addition, video fire detection system is not only connected to surveillance system but it can also be combined with the fire suppression system for better fire control and mitigate unnecessary water damages.

In this study, computer program analysis of the fire images is conducted using MATLAB toolbox, Visual C++, C, C++. Open source computer vision library (OpenCV) and software is developed to capture the fire image and conduct analysis. In addition, image processing techniques using OpenCV are able to processing the real time images for fire detection. Experimental study was conducted using thermal still images and normal still images to develop flame region segmentation codes using the Otsu's method. The traditional Otsu's method is able to segment the flame region and background but unable to segment the flame region completely with complicated background such as the shading and the objects reflection. Therefore, modified Otsu's multi – threshold analysis method is developed to segment the flame region from complicated background video images. To recognize the fire images, parameters such as colour, flame size, motion characteristics of fire are used. Flickering frequency and direction of flame spread are two important fire images characteristics used. Logistic discrimination rules are then set up to determine the probability of the true fire conditions. Once the fire condition is determined, the analyses of real time flame motion images are conducted. Optical flow analysis is then used to track the flame spread direction and flame height can then be estimated. With the information of the flame height, the fire size is estimated using the flame height empirical equation developed by other fire researchers.

Publications arising from the thesis

(1) Wong, A. K., and Fong, N. (2012). Using Thermal Image for Fire Detection. The 2nd Asian-US-European Thermophysics Conference Thermal Science for Sustainable World 2012

(2) Wong, A. K., and Fong, N. (2014). Experimental study of video fire detection and its applications. Procedia Engineering, 71, 316 - 327.

(3) Wong, A. K., and Fong, N. (2014). Study of pool fire heat release rate using video fire detection. 3th International High Performance Building Conference, Purdue.

Acknowledgements

I would like to express my deepest gratitude to my chief supervisor, Dr. N. K. Fong and co – supervisor, Professor W. K. Chow, for their guidance, encouragement and support over the years. Their great enthusiasm and devotion to research has inspired me greatly.

I also sincerely thank Ms. Anson and Professor Anson for their guidance and encouragement in finishing this thesis.

Finally, I want to thank my wife, my children, and the rest of my family who have always supported me.

Table of contents

List of Figures

List of Tables

Nomenclature

R	Red colours
G	Green colours
B	Blue colours
I	Intensity
Y	Luminance
H_{flame}	Real flame height (m)
$Dpool$	Pool diameter (m)
$\dot{Q}^*$	Heat release rate (-)
$\dot{Q}$	Total heat release rate (kW)
ρ_∞	Ambient density (kg/m^3)
c_p	Specific heat of air at constant pressure (kJ/kg K),
T_∞	Ambient temperature (k)
g	Acceleration due to gravity (m/s^2)
C_b	Blue colour Chroma components
C_r	Red colours Chroma components
Q	Heat release rate (kW)
V_f	Flame volume (m^3)
γ	Coefficient of probability (kW/m^3)
f_1	Original signal from first images
f_2	New signal from second images
θ	Predetermined threshold
S	Self-radiation of the flame, and

N	Reflected radiation of the surroundings
T_{obj}	Temperature of the flame
T_s	Temperature of the illumination source
D	Attenuation coefficient of the illumination
W	Corresponds to the total number of pixels in each line
h	Line index
n	Frame number
π_{ROI}	Active pixels of Ω_{ROI}
η_1	Threshold from 0 to 255
Ω_{ROI}	Fire candidate in the region
$Y_{ik}(t)$	Luminance component.
L	The number of possible levels of intensity
$p(z)$	The histogram of the intensity levels in a region
z_i	The variable intensity
$Npixels$	The number of pixels
$\Phi_{i,j}$	Property vector of the pixel at its location (i, j)
$B_{n+1}(i, j)$	Background intensity value
$I_n(i, j)$	Intensity value of the pixel at its location (i, j)
σ	The standard deviation of grey-level probability distribution
μ_4	The fourth central moment of the grey-level probability distribution
$H_m(x, y)$	The history matrix with the coordinate (x, y) at time point m
δ	The maximum duration

$MHI_m(x,y)$	The intensity of the pixel with the coordinate (x,y) in the $m-th$ frame
i	The frame of a video stream
p_i	The fire probability for i video frame
H_m	The height sequence set of fire-like area
Cl^m	Extracted from a video image sequence.
A_m	The set of the coefficients of the Discrete Fourier Transform
a_k^m	A coefficient of the Discrete Fourier Transform
l	The length of the Discrete Fourier Transform
f	Pagni's oscillation frequency of flame (Hz)
$Ncount$	The counting period
SUM_0	The threshold
h_{image}	The flame height in the images
$y_{bottom} - y_{top}$	The number of flame pixels
p_{size}	The pixel size of the images
H_c	Height of digital camera
L_c	Horizontal distance from digital camera to pool fire
D_c	Images distance from fire sources to the image sensors.
S_{year}	Standardisation result of each year
N_y	The collected data of unwanted alarms and false alarm
$\overline{n_{11}}$	The average value of 11 years
σ_{11}	The standard deviation value of 11 years

Chapter 1 Introduction

Building fire is a complex phenomenon. Fire and smoke spread within the building can be affected by various factors such as the geometry, dimension, layout and usage of the building. In order to provide fire protection in the building, it is very important to detect fire at its early stage. This can be achieved by detecting the fire signatures such as aerosol, Infrared (IR), Ultraviolet (UV), heat, and gases, generated from fire. The most common fire and smoke detection methods include the use of point type detectors (i.e. ionisation smoke detectors, photoelectric detectors, heat detectors), line type detectors etc. However, these detection methods have some significant drawbacks including delay in smoke and fire detection especially in large space such as atrium and large shopping mall.

For example, in heat detection, the heat released by burning materials in an atrium would not be able to heat the large amount of air to the flashover temperature of 500 °C within a short period because the different building geometry is able to affect the concentration of hot smoke layer. It has been demonstrated that the air temperatures will be very low compared to flashover temperatures due to air entrainment into the smoke plume. Experimental studies have shown that the smoke temperature in a 26.3 m atrium with a 1.3 MW methanol fire was less than 50 °C. (Yamana & Tanaka, 1985)

In some cases, the smoke may not be able to reach the ceiling of the atrium. Similar problems can be encountered in using spot type smoke detectors. Besides, important parameters such as flame height, flame spread direction cannot be obtained easily using these type of spot detectors.

As computer technology develops and becomes both more sophisticated and more widely employed, it is now possible to use computer to analyse the fire images captured from a video surveillance system and act as a fire detection device. In recent years, the function of video surveillance systems is becoming increasingly diverse. The functionality of modern video surveillance systems is not limited to monitoring environments but also for the recognition and tracking of dynamic objects over time. In closed – circuit television (CCTV) monitoring system, car camera and unmanned aerial vehicles (UAVs) are also required to use a manned or unmanned video surveillance system. (Xie , 2015) A CCTV system is able to monitor traffic situations, recognise car registrations, and recognize faces. A car camera is able to monitor the distance in between cars for prevention of collision. An UAV is able to monitor the aerial reconnaissance to detect forest fires. (Merino, Caballero, Ramiro Martinez - de - Dios, Maza, & Ollero, 2011)

Recognition and tracking methods can be used as detection algorithms. Different detection algorithms require the use of digital image processing technology. Spatial, spectral, and temporal properties of dynamic objects are important parameters for unmanned video surveillance systems. Figure 1.1 illustrates the properties of dynamic objects for unmanned video surveillance.

A dynamic object’s temporal, spectral, and spatial properties define its relation to time, colour, and images shape. (Healey, Slater, Lin, Drda, & Goedeke, 1993) When analysis takes place on two or three properties of a dynamic object, a video surveillance system can respond automatically. An unmanned video surveillance system provides this automatic response.

Today, unmanned video monitoring technology is useful for fire prevention in a variety of environments. (So & Chan, 1994) When analysis algorithms are embedded in traditional video surveillance systems, the system is able to identify fire from the images captured. This technology is known as a video fire detection system.

Video fire detection systems are not widely used in Hong Kong. In other countries, however, many fire engineers use video fire detection technologies as one of the fire safety strategies. In analysis the video fire detection, any features of fire regions in image is related to spectral, spatial, and temporal features properties. Figure 1.2 shows the relationship of features of fire regions and spectral, spatial, temporal features

1.1 Early application of video fire detection

Currently, video fire detection systems are able to protect many different environments and premises including (a) warehouses, (b) atriums, (c) tunnels, (d) forests, (e) historical buildings, (f) plant rooms and (g) aircraft hangars. Figure 1.3 illustrates some of the inhospitable environments. In addition, in these environments, Installation of traditional fire detection systems are not feasible.

1.2 Traditional fire detection

Prior to the discussion of video fire detection technologies, it is necessary to consider traditional fire detection systems. The primary objective of fire detection systems is the protection of life and property. Based on the environmental condition changes caused by fire, traditional fire detectors are able to detect fire automatically. Examples of traditional fire detection technologies are smoke detectors, heat detectors, flame detectors, and gas sensors. Figure 1.4 shows examples of traditional fire detectors including smoke detectors, heat detectors, flame detectors, and gas sensors.

Only three types of fire detection systems are commonly used in Hong Kong. All types of fire detectors used in Hong Kong must be endorsed by the Fire Services Department (FSD). Figure 1.5 shows examples of traditional fire detectors approved by FSD.

Generally, fire development is related to the temporal changing. A review of the literature reveals that three phases have been considered to the fire detection systems including (1) alarm phase, (2) response time (3) extinguishment. (Chapter 10 Fire Detection Systems, 1993). Figure 1.6 shows the time relationship to fire development and fire control.

1.2.1 Heat detectors

The function of heat detectors is to recognise the presence of the thermal energy out-the heat-of a fire in a protected area. (12 Thermal Detection Systems, 1993) The heat is dissipated from the laminar flow and convection heat so traditional heat detectors activated by the convective heat from the fire source. Although different kinds of heat detectors have been designed for the detection of convective heat, smoke detectors are more sensitive than heat detectors. The performance of heat detectors depends on the ambient conditions, the size and volume of the room, and the amount of space. Generally, heat detectors are located on ceilings or in confined spaces such as electrical and mechanical (E and M) plant rooms and storerooms. When the ambient temperature is over a certain threshold value, a heat detector generates a signal and sends it to the associated fire annunciator panels. (Chapter 10 Automatic fire detection, 1990)

Generally, the FSD have approved four kinds of heat detectors including (1) fixed temperature, (2) rate of rise temperature, (3) combination, and (4) linear cable.

1.2.2 Smoke detectors

The FSD have approved four kinds of traditional smoke detectors including (1) ionisation, (2) Photoelectric optical, (3) beam, and (4) self-aspirating. The function of traditional smoke detectors is to detect the presence of smoke particles, and aerosol in a protected area. Smoke includes both solid and liquid particles. Smoke detectors are able to detect smoke particles rapidly. (Chapter 11 Smoke detectors, 1990) Although the response of smoke detectors is faster than that of heat detectors, it depends on what is burning and the burning conditions. However, dust and small insects can also activate smoke detectors, causing nuisance alarms when smoke is not actually present.

1.2.3 Flame detectors

The function of flame detectors is to detect the presence of flame in a protected area. Generally, infrared (IR) and ultraviolet (UV) flame detectors are used for the protection of life and property and the FSD have approved their use. IR and UV flame detectors each operates in a specific spectrum of wavelength. The wavelength spectrum for IR flame detectors is approximately 0.76μm to 220μm. (Dungan, 2008) The wavelength spectrum for UV flame detectors is approximately 0.1μm to 0.35μm. (Dungan, 2008) As with smoke detectors, nuisance alarms are also generated by flame detectors, false triggers can include lighting, arc welding, X-rays, radioactive materials, gas welding, and solar radiation.

1.2.4 Gas sensors

The function of gas sensors is to recognise the presence of abnormal gas in protected area. Gas sensors monitor the detectable levels of gases and measure gas concentration. (Dungan, 2008) Examples of gases detected by gas sensors include carbon dioxide, carbon monoxide, water vapour and hydrocarbons. Generally, different gas sensors have been used to detect the gases from a fire such as Semiconductor type, Catalytic type, Infrared Absorption type. Gas sensing fire detectors are defined and classified by the National Fire Protection Association. (Gas Sensing Fire Detectors, 1993)

1.3 False alarms

To study the status of unwanted alarm and false alarm in Hong Kong, the literature from "Hong Kong Fire Services Department Review" are reviewed and data is collected for statistical analysis. Table 1.1 shows the statistical results including the total numbers of fire calls, unwanted alarms, and false alarms in Hong Kong from 2002 to 2015. From the statistic results in table 1.1, it is clear that many unwanted alarms and false alarm occur. In 2002, a total of 4,131 false alarms occurred, but in 2010, a total of 30,710 unwanted alarms occurred. False alarm and unwanted alarms are a nuisance to the fire services. Figure 1.7 illustrates the total fire calls, unwanted alarms, and false alarms. The number of unwanted alarms is greater than the number of false alarms.

Figure 1.8 illustrates the standardisation of total fire calls, unwanted alarms and false alarms. From the Hong Kong Fire Services Department Review collected 14 years total fire calls, unwanted alarms and false alarm data. There are used to calculation the standardisation.

$$S_{year} = \frac{N_{year} - \overline{n_{14}}}{\sigma_{14}} \qquad \textbf{1.1}$$

where S_{year} is standardisation result of each year. N_y is the collected data of unwanted alarms and false alarm. $\overline{n_{14}}$ is the average value of 14 years, and σ_{14} is the standard deviation value of 14 years

1.4 Video fire detection

Traditional video fire detection technologies can provide extra information such as the flame location and flame height. Video images can be used to identify fire because they record unique visual signatures. Video fire detection technologies offer advantages over traditional fire detection methods. Research of video fire detection technology has shown that the primary objective of video fire detection system is the protection of various harsh environments by capturing video images.

The two main parts of a video fire detection system are:

(1) Hardware (image sensors, central processing unit)

(2) Software (Algorithm, computer program and language)

Different hardware and algorithm can provide the distinct function and the application in distinct environment. Figure 1.9 provides conceptual diagram of video fire detection. Video detection of flame images and smoke images is commonly used in video fire detection systems. Generally, the movement of flame and smoke can be captured by video images. Figure 1.10 shows examples of a smoke image and a flame image. Based on analysis of fire signatures and the use of various algorithms, video fire detection systems are able to recognise the fire sources.

In the early development of video fire detection methods, thermal images have been considered as an important technology. In thermal images, each pixel records temperature data. (Noda & Ueda, 1994) Figure 1.11 shows an example of a thermal image. Thermal images provide not only temperature data but they also display the flame shape. Flame shape seen in still images is similar to that of thermal images. (Wong & Fong, 2014) Figure 1.12 shows an example of a flame image and a thermal image simultaneously. Typical thermal imagers can record by analysing the temperature differences between objects in any lighting condition, day or night. (Thermal imaging for Safety and Efficiency in Public Transportation, 2016)

Image processing technology is commonly used for flame images and thermal images analysis because computer program technology continuously develops. Computer programming, computer languages and computer technologies are important tools for the processing of images. Examples of these technologies are the following: Visual C++, C++, Open Source Computer Vision Library (OpenCV), and the MATLAB Image Processing Toolbox. (Wong & Fong, 2014) (Wong & Fong, 2014) Figure 1.13 illustrates the operation framework of video fire detection.

Besides that, video fire detection technologies are able to capture the fire images. The digital image technologies including the lens and image sensors are also very important evolution in comparison with traditional film cameras. (Toyota, 1972) Figure 1.14 compares the traditional film camera with the digital camera. Whereas a traditional film camera records analogue images on film, a digital camera records digital images using image sensors.

Generally, digital cameras used Charge-coupled Device (CCD) and Complementary Metal Oxide Semiconductor (CMOS) to capture the image. CCD and CMOS image sensor have different amplifiers and analogue to digital (A/D) convertor design in circuity. When the light source falls on the CCD or CMOS chip, the amplifier and A/D convertor can transform the analogue signal to digital signal and output from the sensor. (CCD and CMOS sensor technology Technical white paper, 2010)

Although CCD image sensors and CMOS image sensors are two different technologies, they are both used to capture fire images in video fire detection systems. Fire images include four unique visual signatures and these signatures display the region of interest (ROI). The four unique visual signatures are the following:

1. Colour,
2. Shape
3. Intensity level
4. Motion direction

When the CCD and CMOS image sensors capture the images, image-processing technology is used to identify those images.

During the evolution of video fire detection, computer technology and image processing analysis have played an important role. In early 1985, International Business Machines Corporation (IBM) developed an image processing experimental study on aimed at analysing digital images (Mayers & Bernstein, 1985). The personal computer system, which includes charge-coupled device (CCD) camera, has used. The processing of image pixels is1032 x 1025. The function of the system is identified the boxcar from the digital images.

In addition to the rapid developments made in digital camera technology and image processing techniques, downward trends in computer hardware and image-processing costs have made video fire detection more attractive. The webcam and the Internet protocol (IP) camera, shown in Figure 1.15, are commonly used in computer technology. Today, unmanned aircraft systems have also been used in video fire detection (Merino, Caballero, Ramiro Martinez - de - Dios, Maza, & Ollero, 2011). In this study, a webcam is used to record real-time fire images.

1.5 Images file format

The image file format must be considered for the computer program algorithm. Image file formats can be separated into two types including still images and video images. Image file formats follow standards regarding how the researchers organise and store data (Gonzalez & Woods, Image Compression, 2010). Generally, there are six international organisations that sanction standards for image file format, as follows:

1. International standards organisation (ISO)
2. International Electrotechnical Commission (IEC)
3. International Telecommunications Union (ITU-T)
4. Consultative Committee of International Telephone and Telegraph (CCITT)
5. Society of Motion Pictures and Television Engineers (SMPTE)

6. Chinese Ministry of Information Industry (MII)

An image file formats contain the image data and has different compression standards. Compression standards aim to reduce the transmission time and storage space. Table 1.2 provides the popular formats for still images and video images. In this study, two kinds of image file formats for analysing fire images are used: Windows bitmap (BMP) and audio video interleave (AVI).

1.6 Codes and Standards

Several different codes of practice and international standards have been developed to meet the specific requirement of a video fire detection system. The National Fire Protection Association’s Fire Alarm Code (NFPA72) involves video image flame detection (VIFD) and video image smoke detection (VISD). As per NFPA 72, a video fire detection system requires inspection, testing, and maintenance in accordance with the manufacturer’s recommendation (Gottuk D. , 2008).

NFPA 72 is not the only international standards, further Codes of Practice concerning video fire detection system are available in the public domain. Table 1.3 shows the different Codes and Standards related to the video fire detection system. Any video fire detection must adhere to the relevant Codes of Practice and specified product standards (Wong & Fong, 2014). Table 1.4 shows the summary of the contents from different standards, as follows: NFPA 72, ANSI/FM 3260, BS 5839, and UL 268. In fact, the requirements of video fire detection technology lacks standards and codes of practice at this moment.

1.7 Digital image processing

Digital images processing includes editing images and recovering files. In 1900s, an important advancement in digital image processing was the increasing speed of image data transfer. (Introduction, 2010) Today, digital image processing technologies focuses also on colour image analysis.

The colour spectrum approach has relating to visible flame and colour analysis is important in this current study. A review of the literature reveals that, Sir Isaac Newton discovered that sunlight could be transformed into a colour spectrum when it passes through a prism. Figure 1.16 shows this natural phenomenon of the colour spectrum. Figure 1.17 illustrates the visible spectrum and electromagnetic spectrum. Table 1.5 shows the visible band of the electromagnetic spectrum (red, green and blue colours). The maximum wavelength is about 790nm and the minimum wavelength is about 430nm. Various types of colour models have been developed in digital image processing technology to analyse colours in a flame image.

Digital image processing technologies used around the world employ different colour space models. In video fire detection, the following colour space models are used: (Red, Green, and Blue) RGB, (Hue, Saturation, and Intensity) HSI, (Hue, Saturation, and Value) HSV, (Luminance, and Chroma) YIQ, (Luminance, and Chroma) YUV, (Luminance, and Chroma) YCbCr, and (Luminance and Two colour channel) CIELAB.

Colour space model analysis is a type of spatial analysis. Generally, the RGB, HSI and YIQ colour space models are used for converting colour images to the greyscale images.

1.7.1 The Red, Green and Blue (RGB) colour space model

Different calculation approaches in colour space models generate different colours in images. One traditional approach to colour calculation is the analysis of the mixture of colour. Figure 1.18 illustrates the mixture of colour approach (additive primaries).

In digital images, different variations of red, green and blue function values generate the different colours in the images. Table 1.6 shows common colours of RGB function values.

1.7.2 The greyscale images approach

In most of the methods used for image analysis, the colour images are transformed to grey-scale images so that the intensity of the RGB values can be calculated. (Color Image Processing, 2010) Figure 1.19 depicts a schematic image of a colour cube and provides a description of the grey-scale axis. When the function values of the red, green and blue colours are placed along the grey-scale axis, grey coloured images are displayed.

1.7.3 The HSI colour space model

For the HSI colour space model, three components are used for the generation of colour images, namely: hue (H), saturation (S), and intensity (I). In order to transform the colour images into greyscale images, it is necessary to consider the intensity *I* calculation. Equation 1.2 shows the method for calculating the intensity *I*. Generally, the range of intensity is from 0 to 255.

$$\boldsymbol{I} = \frac{\mathbf{1}}{\mathbf{3}}(\boldsymbol{R} + \boldsymbol{B} + \boldsymbol{G}) \qquad 1.2$$

where I is the intensity of output images, and where R , G and B are primary colours within the input images.

1.7.4 The YIQ colour space model

In the Matrix Laboratory (MATLAB), it is possible to use the YIQ colour space model to transform colour images to greyscale images. Equation 1.3 shows the YIQ colour space model algorithm. The YIQ colour space model has three components, namely: one luminance value (Y), two chrominance value (I and Q).

$$\begin{pmatrix} \boldsymbol{Y} \\ \boldsymbol{I} \\ \boldsymbol{Q} \end{pmatrix} = \begin{pmatrix} \mathbf{0.299} & \mathbf{0.587} & \mathbf{0.114} \\ \mathbf{0.597} & \mathbf{-0.274} & \mathbf{-0.322} \\ \mathbf{0.211} & \mathbf{-0.253} & \mathbf{-0.312} \end{pmatrix} \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{G} \\ \boldsymbol{B} \end{pmatrix} \qquad 1.3$$

In greyscale image analysis, it is only necessary to consider the calculation of the luminance (Y). Colour images can be transformed into greyscale images. Equation 1.4 shows the calculation method. The range of luminance is from 0 to 255.

$$\boldsymbol{Y} = \mathbf{0.299} \times \boldsymbol{R} + \mathbf{0.587} \times \boldsymbol{G} + \mathbf{0.114} \times \boldsymbol{B} \qquad 1.4$$

where Y is the luminance of output images and where R , G and B are primary colours of the input images.

1.7.5 The YCbCr colour space model

Another colour space model used in the analysis of fire images is the YCbCr colour space. Equation 1.5 shows the YCbCr colour space model algorithm.

$$\begin{pmatrix} \boldsymbol{Y} \\ \boldsymbol{C_b} \\ \boldsymbol{C_r} \end{pmatrix} = \begin{pmatrix} \mathbf{16} \\ \mathbf{128} \\ \mathbf{128} \end{pmatrix} + \begin{pmatrix} \mathbf{65.484} & \mathbf{128.553} & \mathbf{24.966} \\ \mathbf{-37.797} & \mathbf{-74.203} & \mathbf{112.00} \\ \mathbf{112.00} & \mathbf{-93.786} & \mathbf{-18.214} \end{pmatrix} \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{G} \\ \boldsymbol{B} \end{pmatrix} \qquad 1.5$$

The YCbCr colour space model has three components. Y is the luminance; C_b and C_r are both Chroma components. C_r is the red-difference component and C_b is the blue-difference component. The YCbCr colour space model is especially useful in video fire detection technology. In chapter 2, the colour space model and its applications are studied in detail.

1.8 Statistical analysis

Statistical analysis is also an important tool in video fire detection technology. It is worth noting, however, that statistical approaches to video fire detection often suffer from an *iceberg effect*. (Suzuki & Takehara, 2012) The iceberg effect occurs when there is insufficient image data to reflect a real-world scenario and insufficient number of flame images available for analysis. Consequently, due to insufficient sample size, the results of the analysis may not be able to reflect reality. In practical application, this iceberg effect can lead to false alarms or unwanted fire alarms. However, discussion of the problem that the iceberg effect brings to video fire detection technologies, however, is not the objective of this thesis. Thus, this problem is left for further research. Hence, Chapter 2 focuses on the detail concerning the statistical approach to video fire detection.

1.9 Artificial neural networks

Artificial Neural Networks (ANNs) are commonly used to recognise fire in video images. (Song, Fan, & Wu, 1999) One type of ANN is the Back-Propagation Neural Network (BPNN). The structure of BPNN requires an input layer, a hidden layer, and an output layer. Each layer in the structure is assigned a different number of nodes. Figure 1.20 shows the typical structure of neural networks. Chapter 2 provides more detail about the ANN approach and the applications.

1.10 Fire characteristics

Another important area in the development of the video fire detection system is the understanding of fire characteristics. Generally, fire has unique visual signatures. These are seen with temporal change. The characteristics of fire include light (flame), heat (radiation), sound (combustion noise), and smoke (combustion product) (Yang, Deng, Fan, & Wang, 2001). All can be used both to identify the fire load in the video image and to recognise the flame shape and the smoke spread at the early stage of a fire.

Chapter 2 Literature review of video fire detection

A literature review of four key areas is presented in this chapter. Focus is on 1) the analysis method used in current study and the related algorithm, 2) how images are captured, 3) video fire detection functions, and application and 4) video fire detection technology since 1991.

From 1991 to 2015, the total number of video fire detection journal articles was approximately 146. By the year 2009, studies relating to video fire detection had increased with the highest number of papers being published during the years 2009 and 2011 reaching 18 in total. Figure 2.1 gives the publication rate of research papers published from 1991 through to 2015.

Various kinds of video fire detection technologies have been studied by several countries, each stimulated by that country's specific demands. Of such published research papers, China has been responsible for approximately 25.7 percent, while in Hong Kong, research on video fire detection technology makes up only 4.7 percent of the worldwide.

Figure 2.2 shows the percentage of research papers published in different countries. In total, nine countries have studied video fire detection technology quite intensively. Besides China, the second highest number of papers were published by the USA, but that number is only 9.5 percent of the total. Korea, Turkey, Taiwan, Spain, Japan, and Canada have also studied video fire detection technology.

In some studies, different countries have collaborated in the research of video fire detection technologies. Collaborating countries include Australia, UK, Canada, China, Taiwan, Japan, USA, Germany, Hong Kong, Korea, Slovenia, Spain, Portugal, Turkey, Greece, and Belgium.

2.1 Functions

The primary function of video fire detection technology is the early detection of fire with, obviously, the accuracy of video fire detection a vital issue. Reviewed literature reveals that, video fire detection systems fall into different categories. Fire, in most cases, generates smoke and flame, however, the detection of fire is actually accomplished more specifically by the analysis of the flame. Figure 2.3 shows the three different categories of video fire detection technology.

From 1991 to 2015, statistical results of the detection of the characteristics of fire were in terms of–flame only, smoke only, or a combination of the specifics of both flame and smoke. Figure 2.4 shows a percentage breakdown of these ways in which fire is detected. Statistics indicate that, many researchers study video flame detection through the analysis of flame images. This is likely because flame images are easier to analyse than those of smoke images are.

Recognition of the characteristics of fire, in terms of flame images, is normally, key to the success of a video fire detection system. Such recognisable characteristics include (1) flame colour; (2) flame height; (3) flame shape; (4) flame light intensity; and (5) flickering frequency. Such characteristics are recognisable in still images and in video images. Figure 2.5 shows statistics regarding the quantities of the five different fire characteristics that have been analysed from 1991 through to 2015.

The flame colour, flame shape, flame light intensity, and flame flickering frequency are key to video fire detection analysis. It is of note that flame height has not been commonly analysed, even though all fire characteristics, presented in images, can be used to recognise the fire state. Video fire detection makes use of various algorithms to obtain flame characteristics. In the following section, of this thesis the results of flame colour analysis, flame geometry analysis (including flame height, flame shape) flame light intensity and flame flickering frequency analyses are described

2.1.1 Flame colour

In image processing technology, the colour of fire is the main parameter for flame recognition from images. A review of the literature reveals that seven kinds of colour space models have been used for flame recognition or segmenting flame in images. These seven colour space models are as follows:

1. RGB colour space model (Noda & Ueda, 1994) (Phillips III, Shah, & Lobo, 2000) (CHen, Kao, & Chang, 2003) (Liu & Ahuja, 2004)
2. HSI colour space model (CHen, Kao, & Chang, 2003)
3. HSV colour space model (Yamagishi & Yamaguchi, 1999)
4. YUV colour space model (Celik, Demirel, & Ozkaramanli, 2006)
5. YCbCr colour space model (Çelik, Özkaramanlı, & Demirel, 2007)
6. YIQ colour space model (Shi, Liu, & Liu, 2009)
7. CIE LAB colour space model (Celik T. , 2010)

Most researchers of video fire detection, study the flame colours. Image analysis using a colour space model involves not only segmentation of the images but also recognition of fire in images. The RGB, HSI, HSV, and YUV colour space models were used in the earlier years of video fire detection research. Most recently, the YCbCr and CIE LAB colour space models have also been used for fire detection.

Generally, the colour components of digital images are red, green, and blue. The RGB colour space model is commonly used for the analysis of flame images. Flame colour only, however, is not converted to temperature distribution. In this case, the use of an infrared camera is an important auxiliary piece of equipment. When grey colour images from an infrared camera are compared with the colour ratio of colour images (Noda & Ueda, 1994), the flame region is obtained from these images.

Although the colour of the flame is an important phenomenon in the study of flame images, the probability of fire (i.e. the colorprob) (Phillips III, Shah, & Lobo, 2000) can also be analysed. To identify fire in images, the nature of flame motion and colorprob are used (Phillips III, Shah, & Lobo, 2000). The threshold (k) of the fire is found from these two parameters $Color(x, y)$ and $\sigma(x, y)$.

To reduce the detection time in video fire detection, the colour decision rule can be used. In 2002, Chen et al (CHen, Kao, & Chang, 2003) proposed the use of a colour model and decision rule for detecting real-time fire. To apply the decision rule, the fire pixels are first extracted from the images. The colour image processing uses the HSI and the RGB colour models. The colour decision rule for use with fire has three decision rules, and are as follows.

Rule 1: Red colour ≥ Green colour > Blue colour

Rule 2: Red colour > Red colour $_{\text{threshold}}$

Rule 3: IF (Saturation ≥ ((255-Red colour) x Saturation $_{\text{threshold}}$/Red colour $_{\text{threshold}}$))

Fire – pixel

ELSE

Not fire – pixel

The second decision rule considers the fire pixels in comparison to the threshold value.

$$i = \frac{1}{3}(red + green + blue) \qquad 2.1$$

$$S = 1 - \frac{3}{(r+g+b)}[min(r, g, b)] \qquad 2.2$$

$$h = \begin{cases} \theta & if\ b \leq g \\ 260 - \theta & if\ b > g \end{cases} \qquad 2.3$$

$$\theta = cos^{-1}\left\{\frac{\frac{1}{2}[(r-g)+(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{1/2}}\right\} \qquad 2.4$$

The use of the results of research on flame colour features in different environments, Table 2.1 show saturation and intensity information for different features of low-temperature fire. (Horng, Peng , & Chen, 2005)

By analysing the flame colour features, video fire detection can segment the fire region from these images, by removing false fire-like regions and estimating the burning degree of flames (Horng, Peng , & Chen, 2005).

Normally, the main colours visible in fire images are orange, yellow and red (Liu & Ahuja, 2004). To analyse the ratio of colours in fire, it is necessary to include blue along with red, orange, yellow. Table 2.2 shows the ratio distribution of blue, yellow, orange, and red. A review of the literature reveals that the colour ratio of real fire pixels is not less than 1.5%.

2.1.2 Flame height

A review of the literature reveals that, although flame height is an evident feature of fire visible in images, this fire characteristic is not commonly used in video detection of fire. Clearly, flame height can be measured in pixels and it is of note from the results of different kinds of fuel used in experiments that correspondingly, different flame heights result. (Maoult, Sentenac, Orteu, & Arcens, 2007) Table 2.3 shows test results of the flame height when gasoline, acetone and alcohol were burned.

An important technique for use in video fire detection is the analysis of the actual flame height, achieved by an estimation of flame height measured in pixels. Table 2.4 shows flame height in pixels in accordance with the actual flame height.

Only the real-flame height measurement can be used to estimate the heat release rate (HRR) indirectly. When video fire detection is able to separate real flame images, real flame height can estimate the HRR using fire dynamics.

Analysis of real flame height is a subject of study undertaken by many researchers such as Heskestad and McCaffrey (Drysdale, 1999). Using measurements of real flame height and the fuel-pool diameter, researchers are able to estimate the HRR. Equations 2.5 and 2.6 show Heskestad and McCaffrey algorithm (Drysdale, 1999).

$$\frac{H_{flame}}{D_{pool}} = -1.02 + 3.7\dot{Q}^{*2/5} \qquad 2.5$$

$$\dot{Q}^{*} = \frac{\dot{Q}}{\rho_{\infty} c_p T_{\infty} \sqrt{g D_{pool}} D_{pool}^{2}} \qquad 2.6$$

where H_{flame} is real flame height (m). $D_{pool}$is pool diameter (m). $\dot{Q}^{*}$is dimensionless heat release rate (-). $\dot{Q}$ is the total heat release rate (kW). ρ_{∞} is ambient density (kg/m^3). c_p is the specific air heat at constant pressure (kJ/kg K), T_{∞} is ambient temperature (k) and g is acceleration due to gravity (m/s^2).

Using another approach, the HRR can also be calculated using the flame volume (Beji, Merci, Verstockt, & Walle, 2012) (Stratton, 2005).

$$Q = \gamma V_f \qquad 2.7$$

where Q is the heat release rate (kW), V_f is the flame volume (m^3), and γ is the coefficient (kW/m^3) of probability.

To verify the detection of flame in images effectively, the flame height and width can be used (Verstockt, et al., 2011). Equation 2.8 shows the local maxima and minima, in the set of N consecutive BB^{width} and BB^{height}

$$BBD = \frac{\left|extrema\left(BB_{1:N}^{width}\right)\right| + \left|extrema\left(BB_{1:N}^{height}\right)\right|}{N} \qquad 2.8$$

Some researchers have used the ratio of the maximum height and maximum width to determine the existence of both the flame and its volume within the images. Generally, when the height and width ratio is greater than the threshold value, the image contains a fire region. (Nguyen - Ti, Nguyen - Phuc, & Do - Hong, 2013)

2.1.3 Flame shape

Dynamic flame shape is a stochastic motion, so flame shape analysis is a spatial-temporal analysis (Wang, Finn, Erdinc, & Vincitore, 2013). Many researchers use flame shape analysis to study video fire detection. In flame detection, three primary flame shape analysis are used. They are (1) flame area analysis; (2) flame contour feature analysis; and (3) flame texture analysis. In some research, analysis of flame shape is used to reconstruct fire images. (Stratton, 2005)

A review of the literature reveals that in the early years of video fire detection, the polar coordinate transformation (Yamagishi & Yamaguchi, 1999) approach was used to analyse the fluctuation area. The polar coordinate transformation approach employs an angle (θ) against the horizontal axis, a distance (r) between the contour, and a position (G) being the centre of gravity. The fluctuation data is $r(\theta, t)$. The polar coordinate transformation approach results require fluctuation data to be recorded over a period of time (t)

In recent years, some researchers (Liu & Ahuja, 2004) have used Fourier coefficients to represent the shape of a fire region. In general, the shape of a fire region is illustrated by a boundary line. To determine the boundary of that shape, the Discrete Fourier Transform (DFT) method can be used. Equation 2.9 shows the coefficients of the DFT.

$$a_k = \frac{1}{N}\sum_{i=1}^{N} z_i exp\left(-j\frac{2\pi}{n}ik\right) \qquad 2.9$$

where $k = -\left\lfloor\frac{N-1}{2}\right\rfloor, \dots\dots\dots\dots\dots\dots, \left\lfloor\frac{N}{2}\right\rfloor$

In general, after video fire detection has segmented the foreground and background, the flame contour information can be used for the analysis by the probability model $f\left(F(A^m)\right)$ (Hongliang, Qing, & Sun'an, 2012).

A review of the literature reveals that video fire detection can be used to discriminate fire from contour dynamic features. The spatial-temporal contour dynamics feature is also discriminated by the support vector machine (SVM) (Wang, Finn, Erdinc, & Vincitore, 2013).

2.1.4 Flame brightness

Generally, flame brightness is used for illumination, for instance in video fire detection research, flame brightness is not simply a fire characteristic, it also supplies the light needed for image capture. For flame image detection in dark environments such as inside an aircraft (Foo, 1996), flame brightness is an important consideration. Normally, a charge coupled device (CCD) camera is able to capture flame images because the flame itself generates the brightness required. (Cheng, Wu, Yuan, & Zhou, 1999) High contrast between the flame and its surrounding environments is also an important image aspect enabling recognition of flame regions therein (Liu & Ahuja, 2004).

Generally, the image processing approach is an appropriate method for analysis of flame brightness. A comparison of two images is a commonly used method in the analysis of flame brightness.

$$d_{ij}(x,y) = \begin{cases} 1 & if\ \left|f(x,y,t_i) - f\left(x,y,t_j\right)\right| > \theta \\ 0 & otherwise \end{cases} \qquad 2.10$$

where θ is a predetermined threshold.

Generally, flame images have two brightness sources: the *self-radiation of the flame* and the *reflected radiation of the surroundings* (Cheng, Wu, Yuan, & Zhou, 1999). Equations 2.11 and 1.12 show the self-radiation of the flame and the reflected radiation of the surroundings.

$$\boldsymbol{S} = \int_{\lambda_1}^{\lambda_2} \frac{\boldsymbol{C_1}}{\boldsymbol{\lambda^5\left[exp\left(\frac{C_2}{\lambda T_{obj}}\right)-1\right]}} \times \boldsymbol{CCD(\lambda)d\lambda} \qquad 2.11$$

$$\boldsymbol{N} = \int_{\lambda_1}^{\lambda_2} \boldsymbol{D} \times \frac{\boldsymbol{C_1}}{\boldsymbol{\lambda^5\left[exp\left(\frac{C_2}{\lambda T_s}\right)-1\right]}} \times \boldsymbol{CCD(\lambda)d\lambda} \qquad 2.12$$

where $S$is self-radiation of the flame, and $N$is the reflected radiation of the surroundings. T_{obj} and T_s represent the respective temperature and illumination source of the flame. D is the attenuation coefficient of the illumination. $CCD(\lambda)$ is the spectral response function of the CCD – camera. A review of the literature reveals that the radiation distribution spectrum of the flame's surroundings can be obtained by differentiation. Equation 2.13 and 2.14 illustrates the expression.

$$\frac{dS}{d\lambda} = \frac{C_1 \times CCD(\lambda)}{\lambda^5\left[exp\left(\frac{C_2}{\lambda T_{obj}}\right)-1\right]} \approx \frac{C_1 \times CCD(\lambda)}{\lambda^5 \times exp\left(\frac{C_2}{\lambda T_{obj}}\right)} \qquad 2.13$$

$$\frac{dN}{d\lambda} = D \times \frac{C_1 \times CCD(\lambda)}{\lambda^5\left[exp\left(\frac{C_2}{\lambda T_{obj}}\right)-1\right]} \approx D \times \frac{C_1 \times CCD(\lambda)}{\lambda^5 \times exp\left(\frac{C_2}{\lambda T_s}\right)} \qquad 2.14$$

To analyse wavelength, 0.4 – 0.8 μm wavelength band blocking, filter technology can be used for recognition of flame. A liquid crystal – light valve (LC – LV) is located at the front of the CCD camera.

Analysis of flame images requires not only algorithms but also the LCLV (Yang, Deng, Fan, & Wang, 2001), as the principle function of the LCLV is to filter out other nearby images. Two criteria are used for the analysis of the flame images. The first is that the grey-scale of the flame pixels must be brighter than that of the background. The second criterion is that the number of bright flame pixels must be greater than the threshold value. The threshold value can be used to identify the flame region from the images.

The mean value of the brightness can be used (Schultze, Kempka, & Willms, 2006) to analyse flame brightness. The mean brightness is between 0 and 255. (Equation 2.15 below shows the mean value calculation of brightness).

$$\boldsymbol{g(h,n) = \frac{1}{W}\sum_{i=1}^{W} s(i,h,n)} \tag{2.15}$$

where W corresponds to the total number of pixels in each line, h is the line index and n is the frame number.

To detect flame in images from any brightness, some researchers (Owrutsky, et al., 2006) use the threshold method for the analysis of images.

If $L_{alarm} > L_{background} + L_{threshold}$, then the alarm count increase. If $L_{alarm} < L_{background} + L_{threshold}$, then the alarm count decrease. In a past study, the alarm count reached 75. The response time was within five seconds (Owrutsky, et al., 2006).

Luminance of active pixels $I_{ROI}(t)$ (Marbach , Loepfe, & Brupbacher, 2006) is a primary feature in flame images. The following algorithm for luminance of active pixels can be used:

$$\boldsymbol{I_{ROI}(t) = \{mean\{Y_{ik}(t)\} | (i,k) \in \pi_{ROI}\}} \tag{2.16}$$

$$\boldsymbol{\pi_{ROI} = \{(i,k) \in \Omega_{ROI} | A_{ik}(t) \geq \eta_1\}} \tag{2.17}$$

where $\pi_{ROI}$is the set of active pixels of Ω_{ROI}, η_1 is the threshold from 0 to 255, Ω_{ROI} is the fire candidate in the region, and $Y_{ik}(t)$ is the luminance component.

In flame image analysis, the phenomenon of flame brightness can also use different spectral range values. The spectral range (Maoult, Sentenac, Orteu, & Arcens, 2007) includes (1) Near UV: 350-390 nm (low cost camera), (2) Visible 390-750 nm, (3) Near infrared (NIR) 750-1100 nm, and (4) NIR limited to Δλ = 100nm around 950nm.

To detect fire from video images by analysing luminance, the variances shown on the luminance map can be used. (Ko, Cheong, & Nam, 2009) Equation 2.18 shows the luminance map variances. Equation 2.19 shows the variance of luminance.

$$\boldsymbol{\mu_{x,yL} = \frac{1}{\sum_{u=1}^{N} H_{x,yL(u)}} \sum_{u=1}^{N} u H_{x,yL}(u)} \qquad 2.18$$

$$\boldsymbol{\sigma_{x,yL} = \frac{1}{\sum_{u=1}^{N} H_{x,yL(u)}} \sum_{u=1}^{N} \left(u - \mu_{x,yL}\right)^2 H_{x,yL}(u)} \qquad 2.19$$

Where $\sigma_{x,yL} > L_T$ fire pixel else non-fire pixel; L_T is the default threshold for variance, N is the number of consecutive images, $\boldsymbol{H_{x,yL}(u)}$ is the luminance histograms of pixel (x, y) in 10 consecutive frames and subscripts, and L is used to indicate this histogram as part of the luminance map.

Generally, flame images have a very high brightness in tunnel environments. Some researchers (Han & Lee, 2009) have used the discrepancy between brightness of objects and brightness of tunnel environments for flame image recognition. Equation 2.20, 2.21 and 2.22 show the algorithm used.

$$\boldsymbol{I'_k(x,y) = \begin{cases} 1 & if\ I_k(x,y) > th_1 \\ 0 & otherwise \end{cases} \quad 0 \le k \le N}$$ 2.20

$$\boldsymbol{T''(x,y) = \begin{cases} 1 & if\ T(x,y) > th_1 \\ 0 & otherwise \end{cases}}$$ 2.21

$$\boldsymbol{D^f_k(x,y) = I'_k(x,y) - T''(x,y) \quad 0 \le k} \le \mathrm{N}$$ 2.22

$\boldsymbol{I_k(x,y)}$ is the intensity of one image among $\boldsymbol{N}$ sequential input images. $\boldsymbol{T(x,y)}$ is the intensity of the background and $\boldsymbol{th_1}$ is the threshold value. $\boldsymbol{D^f_k(x,y)}$ is the difference in the intensity of the images between binary inputs and image backgrounds.

In addition, the intensity values of the background and the image can be used in threshold method analysis (Ko, Cheong, & Nam, 2010). If $|I_n(x) - B_n(x)| > T_n(x)$ then x is moving, otherwise x is non-moving. $I_n(x)$ is the intensity value at each spatial location x in frame n . $B_n(x)$ is the background value at the same position, and $T_n(x)$ is the threshold value of the difference between background and image.

In addition, the level of intensity can be used in pixel analysis (Bosch, Gomez, Molina, & Miralles, 2009). Equation 2.23 shows the expression for the intensity.

$$\boldsymbol{m = \sum_{i=0}^{L-1} Z_i \cdot p(z_i)}$$ 2.23

where L is the number of possible levels of intensity, $p(z)$ represents the histogram of the intensity levels in a region, z_i is the variable intensity.

Generally, video fire detection can make use of the spatial and temporal information for flame detection (Habiboglu, Gunay, & Cetin, 2011). Equations from 2.24 to 2.30 show calculations of pixel property parameters. However, these equations consider only spatial information.

$$\boldsymbol{X(i,j) = i} \qquad 2.24$$

$$\boldsymbol{Y(i,j) = j} \qquad 2.25$$

$$\boldsymbol{I(i,j) = Intensity\ (i,j)} \qquad 2.26$$

$$\boldsymbol{Ix(i,j) = \left|\frac{\partial Intensity\ (i,j)}{\partial i}\right|} \qquad 2.27$$

$$\boldsymbol{Iy(i,j) = \left|\frac{\partial Intensity\ (i,j)}{\partial j}\right|} \qquad 2.28$$

$$\boldsymbol{Ixx(i,j) = \left|\frac{\partial^2 Intensity\ (i,j)}{\partial i^2}\right|} \qquad 2.29$$

$$\boldsymbol{Iyy(i,j) = \left|\frac{\partial^2 Intensity\ (i,j)}{\partial j^2}\right|} \qquad 2.30$$

When video detection of fire in images uses spatial analysis and temporal analysis of the same coordinates, another calculation method for pixel property parameters is necessary. Equation 2.31 to 2.37 show the calculation method.

$$\boldsymbol{I(x,y,n) = Intensity\ (x,y,n)} \qquad 2.31$$

$$\boldsymbol{Ix(i,j,n)} = \left|\frac{\partial Intensity\ (i,j,n)}{\partial i}\right| \tag{2.32}$$

$$\boldsymbol{Iy(i,j,n)} = \left|\frac{\partial Intensity\ (i,j,n)}{\partial j}\right| \tag{2.33}$$

$$\boldsymbol{Ixx(i,j,n)} = \left|\frac{\partial^2 Intensity\ (i,j,n)}{\partial i^2}\right| \tag{2.34}$$

$$\boldsymbol{Iyy(i,j,n)} = \left|\frac{\partial^2 Intensity\ (i,j,n)}{\partial j^2}\right| \tag{2.35}$$

$$\boldsymbol{\partial_t I(i,j,n)} = \left|\frac{\partial Intensity\ (i,j,n)}{\partial n}\right| \tag{2.36}$$

$$\boldsymbol{\partial_t^2 I(i,j,n)} = \left|\frac{\partial^2 Intensity\ (i,j,n)}{\partial n^2}\right| \tag{2.37}$$

$\boldsymbol{\partial_t I}$ and $\boldsymbol{\partial_t^2 I}$ are the first and second derivatives of intensity with respect to time (t). From the above equations, the property vector $\Phi_{ST}(i,j,n)$ is able to use a covariance matrix to detect a flame.

$$\boldsymbol{\Phi_{ST}(i,j,n)} = \begin{bmatrix} I(i,j,n) \\ Ix(i,j,n) \\ Iy(i,j,n) \\ Ixx(i,j,n) \\ Iyy(i,j,n) \\ \partial_t I(i,j,n) \\ \partial_t^2 I(i,j,n) \end{bmatrix} \tag{2.38}$$

The following covariance matrix can be used to estimate the given region in images.

$$\Sigma = \frac{1}{Npixels-1}\sum_i \sum_j \left(\boldsymbol{\Phi_{i,j}} - \bar{\boldsymbol{\Phi}}\right)\left(\boldsymbol{\Phi_{i,j}} - \bar{\boldsymbol{\Phi}}\right)^{\boldsymbol{T}} \tag{2.39}$$

where

$$\bar{\mathbf{\Phi}} = \frac{1}{Npixels} \sum_i \sum_j \mathbf{\Phi}_{i,j} \qquad 2.40$$

where $\boldsymbol{Npixels}$ is the number of pixels, and $\mathbf{\Phi}_{i,j}$ is the property vector of the pixel at its location, and (i, j).

The intensity background and the intensity value of the pixel at its location (i, j) in the n^{th}video frame are used to detect the moving regions. A review of the literature, (Truong, Kin, & Kin, 2011) reveals that this analysis method is popular for flame detection .

$$B_{n+1}(i, j) = \begin{cases} I_n(i, j) > B_n(i, j) & B_n(i, j) + 1 \\ I_n(i, j) < B_n(i, j) & B_n(i, j) - 1 \end{cases} \qquad 2.41$$

$$|I_n(i, j) - B_n(i, j)| > Threshold \qquad 2.42$$

The threshold value can be assumed from guesswork or experience. $B_{n+1}(i, j)$ is the background intensity value, and $I_n(i, j)$ is the intensity value of the pixel at its location (i, j).

The contrast in an image, however reveals the amount of local intensity. The intensity histogram graphs can describe the range of brightness levels (Jiao, Weir, & Yan, 2011). Equation 2.43 shows he picture contrast used for analysis of the images.

$$F_{CON} = \frac{\sigma}{(\alpha_4)^n} \qquad 2.43$$

n is a positive number, σ is the standard deviation of grey-level probability distribution, and α_4 is kurtosis. Equation 2.44 shows the calculation of kurtosis.

$$\alpha_4 = \frac{\mu_4}{\sigma^4} \qquad 2.44$$

μ_4 is the fourth central moment of the grey-level probability distribution.

The absolute value calculation is performed using two images $f(A)$ and $f(B)$ (Ning & Fei, 2012). Equation 2.45 shows the calculation. Equation 2.46 shows the algorithms.

$$f(C) = |f(A) - f(B)| \qquad 2.45$$

when

$$\begin{cases} f(C) = 0, f(C) < Threshold \\ f(C) = 1, f(C) > Threshold \end{cases} \qquad 2.46$$

When the pixels are larger than the threshold, the pixels are a grey colour. Therefore the pixels are equal to 1 $f(1) = 1$.

The Motion History Image (HMI) algorithm is another method for the analysis of the motion history images (Xu, Zhu, & Xie, 2012).

$$H_m(x,y) = \begin{cases} m \quad if \quad I'_m(x,y) = 1 \\ 0 \quad if \quad I'_m(x,y) = 0 \quad and \quad H_{m-1}(x,y) \le (m - \delta) \end{cases} \qquad 2.47$$

$H_m(x,y)$ is an element in the history matrix with the coordinate (x,y) at time point m. δ is the maximum duration. A review of the literature reveals that, δ is set to 5.

$$MHI_m(x,y) = \begin{cases} \frac{H_m(x,y) - (m-\delta)}{\delta} \times 255 & if \quad H_m(x,y) \neq 0 \\ 0 & others \end{cases} \qquad 2.48$$

$MHI_m(x,y)$ is the intensity of the pixel with the coordinate (x,y) in the $m-th$ frame. The history information of the motion region is recorded in the MHI.

In recent years, intensity values have been used for the dynamic background subtraction method (Beji, Merci, Verstockt, & Walle, 2012). Equation 2.49 shows the algorithm of the background subtraction method. $T_n[x,y]$ is the threshold at position $[i,j]$. The brightness values in image frame x_n and frame x_{n-1}

$$|x_n[i,j] - x_{n-1}[i,j]| > T_n[i,j] \qquad 2.49$$

The video detection method is commonly used for the analysis of moving objects such as flame flickering. A review of literature reveals that flickering frequency is observed in many studies of video fire detection. Normally, analysis of flickering frequency is necessary to enable records of different brightness levels, over time to be made.

2.1.5 Flame flickering frequency

The recognition of flame flickering frequency is generally, a form of space-time data analysis (Yamagishi & Yamaguchi, 1999). The magnitudes of flame flickering are used to analyse the frequency. When images are recorded by a digital camera, generally the image sampling frequency is 25Hz meaning 25 frames per second in a video sequence (Zhang, Zhuang, Du, Wang, & Li, 2006). Consequently, the flame flickering cannot be directly obtained from the video sequence. Normally, the calculation of flame flickering frequency uses the Discrete Fourier Transform (DFT).

$$h_i^m = Height\left(Cl_i^m\right) \tag{2.50}$$

$$a_k^m = DFT\left(H^m\right) = \frac{1}{n}\sum_{i=1}^{n} h_i^m \exp\left(-j\frac{2\pi}{n}ik\right) \tag{2.51}$$

$$f_d\left(A^m\right) = \sum_{i=2}^{\frac{1}{2}} \frac{a_k^m \cdot \overline{a_k^m}}{l/2-1} \tag{2.52}$$

where H_m is the height sequence set of fire-like area Cl^m being extracted from a video image sequence. A_m is the set of the DFT coefficients, and a_k^m is a coefficient of the DFT, l is the length of the DFT.

Normally, the actual flame height is more difficult to analyse, hence the calculation of the suspected flame area $Zone(i,t), i = 0,1,\cdots,b$ of the fire offers an alternative method (Hongliang, Qing, & Sun'an, 2012). The relative area is expressed as $S_i(t), i = 1,2,\cdots,b,$ where b is the region number. The image resolution factor is assumed to be $M \times N$. The grey colour value is $f(x,y,t)$ at location (x,y) in the time of t. $p(k,t)$ is the probability results. Equation 2.53, 2.54 and 2.55 shows the algorithm for calculating flame area.

$$S_i(t) = MN \sum_{(x,y)\in Zone(i)} p(k,t) \qquad 2.53$$

$$p(k,t) = (1/MN) \sum_{f(x,y)=1} 1 \qquad 2.54$$

$$\overrightarrow{S_m} = \{S_m(1), S_m(2), \cdots, S_m(n)\} \qquad m = 0,1,\cdots,b \qquad 2.55$$

For extraction of the dynamic characteristic sequence of changes, Fourier coefficients can be used. Equation 2.56 shows the algorithm of Fourier coefficients.

$$f_k^m = \frac{1}{n}\sum_{i=1}^{n} S_m(i) e^{-j2\pi ik/n} \qquad 2.56$$

The Fourier power spectrum can be used for analysis of the energy spectrum. Equation 2.57 shows the algorithm of the energy spectrum. The algorithm of the energy spectrum is used to eliminate unwanted objects.

$$P(F^m) = \frac{1}{n}\sum_{k=1}^{n} f_k^m \overline{f_k^m} \qquad 2.57$$

In image processing technology, block techniques (Yu, Mei , & Zhang, 2013) is used to recognise features of flame motion. Equation 2.58 shows the block algorithm. To experiment, the predetermined threshold can be set to 50. When more than half of all pixels in a block satisfy, the block can then be considered a flame block.

$$H_{\tau}(x, y, t) > T \qquad 2.58$$

where T is a predetermined threshold.

Past experiments reveal analysis of flame flickering to be a powerful tool for differentiating between actual fire and fire-like objects. The following algorithm (Barmpoutis, Dimitropoulos, & Grammalidis, 2013) is used to calculate flame flickering frequency. Equation 2.59 shows the algorithm.

$$F(i, j) = 2^{c(i,j)} - 1 \qquad 2.59$$

where $c(i, j)$ is mathematically expression.

Normally, video fire detection uses not only one flame image feature, but several. When the method involves the use of three or four flame image features, fire detection accuracy is reinforced. Researchers have determined a threshold value for each flame feature. The results of past experiments reveal the flickering threshold is 44.

Generally, analysis of flickering frequency uses the Fast Fourier Transform (FFT) (Schröder, Krüger, & Kümmerlen, 2014). The image processing method detects a deflagration situation, but little research has been conducted on this method. The

Nyquist – Shannon sampling theorem can also be used to analyse flickering frequency. Equation 2.60 shows the theorem.

$$a = \frac{\sqrt{\sum_{f=2\ Hz}^{8\ Hz} \left|X(f)\right|^2}}{\sqrt{\sum_{f=1\ Hz}^{f_{\max}} \left|X(f)\right|^2}} \quad a \in [0,1] \qquad 2.60$$

where $f_{\max}$ is the maximum frequency, a is the power ratio of the frequency band from 2 to 8 Hz, and $X(f)$ is the amplitude value at the frequency f. In results from past experiments, typical fires or flames illustrate the ratio a in the range of 0.6 to 0.9.

From the literature (Marbach , Loepfe, & Brupbacher, 2006) it can be seen that the luminance of the active region of interest is expressed in $I_{ROI}(t)$, and the frequency of the region of interest is expressed in $f_{ROI}(t)$. Analysis of the frequency $f_{ROI}(t)$ can be used to plot the luminance curve $I_{ROI}(t)$ over time t.

Another method of flickering frequency analysis is the Power Density Spectrum (PDS) approach (Schultze, Kempka, & Willms, 2006). The spectrogram illustrates the evolution of the PDS of the pool fire's flame-flickering. The x-axis shows the time in seconds, and y-axis shows the frequency in Hertz.

Temporal wavelet analysis (Toreyin, Dedeoglu, Gudukbay, & Cetin, 2006) is another method for detecting flickering frequency. Each fire pixel $x_n[k,l]$ is fed through a high- and also a low-pass filter. The coefficients of a high-pass filter are -0.25, 0.5, and -0.25, whilethe coefficients of low-pass filter are 0.25, 0.5 and 0.25. Two wavelets expressed as $d_n[k,l]$ and $e_n[k,l]$ are sub-signals produced by the filter. In an analysis of the flame flickering frequency, flame brightness is not a unique feature. Analysis of red moving objects, however, can be created to determine the flame flickering, and the threshold value (T_R) (Duong & Tuan, 2009) of the red channel is 200. In addition, researchers can analyse flame flickering based on the variation of the flame's area (Hou, Qian, Zhao, Pan, & Zhang, 2009). Flame can also be captured by CCD cameras and near infrared cameras (NIR) although flame flickering appears in a different spectral range (Maoult, Sentenac, Orteu, & Arcens, 2007).

A review of the literature reveals that, fire researchers have used Pagni's oscillation frequency formula (Jianzhong, Jian, Jian, & Jun, 2010). Equation 2.61 shows the Pagni's oscillation frequency formula. Previous experiments have shown that the equivalent diameter ranges from 0.03m and 60m (Juan & Qifu, 2012).

$$f^2 \cong \frac{2.3}{Dpool} \qquad 2.61$$

where f is Pagni's flame oscillation frequency (Hz) and $Dpool$ is the equivalent diameter of the pool fire (m). Tables 2.5 and 2.6 show the experimental results.

In video images, when the oscillation counter (Chen, He, & Wang, 2010) exceeds a threshold SUM_0, it can be determined that the video images are showing fire. Equation 2.62 shows the algorithm of the oscillation counter.

$$(SUM(x,y,t) - SUM(x,y,t-Ncount)) > SUM_0 \qquad 2.62$$

where $Ncount$ is the counting period, SUM_0 is the threshold, and the threshold is related to the counting period. Further study is needed to analyse the optimal threshold.

In order to ensure that flame flickering analysis is able to detect real fire in images, various flame. Detection flickering frequency techniques can be used A counter $Timer(x, y)$ (He, Yang, Zeng, Ye, & Wu, 2015) can be used for counting pixel change. Equation 2.63-2.65 shows the counter algorithm. $Timer(x, y, t)$ and $Timer(x, y, t-1)$ can also be used as counter values for the pixels at time (t) and at time $(t-1)$. T_f is a predefined flickering threshold. $Y(x, y, t)$ is the Y component of the YCbCr colour model. When the counter value of a pixel is larger than the threshold at a given time, the region of interest is considered a flickering image.

$$Timer(x, y) = \begin{cases} Timer(x, y, t-1) + 1 \\ \qquad if\left(\left|\Delta Y(x, y, t)\right| \geq \Delta T_Y\right) \\ Timer(x, y, t-1) + 0 \\ \qquad if\left(\left|\Delta Y(x, y, t)\right| \leq \Delta T_Y\right) \end{cases} \qquad 2.63$$

$$\Delta Y(x, y, t) = Y(x, y, t) - Y(x, y, t-1) \qquad 2.64$$

$$\left(Timer(x, y, t) - Timer(x, y, t-n)\right) \geq T_f \qquad 2.65$$

Generally, flame flickering is the oscillation of flame in images. In flame flickering analysis, to determine the optimal threshold, different detection algorithms are necessary. Fast Fourier Transform (FFT), Wavelet Transform (WT) and Mean Crossing Rate (MCR) can also be used to calculate flame flickering frequency. In flame flickering frequency analysis, the brightness intensity variation, flame height and flame area are also used.

Image processing techniques are also a necessary part of the flame flickering frequency analysis. The intensity values in each pixel can be obtained from the colour space model. The colour space model used can be either the RGB colour space model or the YCbCr colour space model. In order to identify real flame in video images, the characteristics of real flame flickering frequency must be known.

2.1.6 Fire dynamics analysis for flickering frequency

Video fire detection also used the characteristics of flame. Another important characteristic is real flame's flickering frequency. Many fire science researchers have studied flickering frequency including Chitty and Cox, McCaffrey and Zukoski (Drysdale, 1999), obtaining the different flickering frequency results from observation. The range of flame flickering frequency is from 1 Hz to 10 Hz (Schröder, Krüger, & Kümmerlen, 2014). The flickering frequency effect is based on pool diameters. Equation 2.66 shows the relationship between flickering frequency and pool diameters.

$$f = (0.50 \pm 0.04)(g/Dpool)^{1/2}\ Hz \qquad 2.66$$

where $Dpool$ is pool diameter (m), f is flickering frequency, and g is a gravitational acceleration constant (9.81 m/s^2).

In video fire detection technology, the sequence of video image captured records of not only the changes of flame shape, but also of the flame flickering frequency.

In the detection of the real flame from video images, analysis of the features of those images are an important contribution in the achievement of fire control. Besides the feature, flickering frequency, mentioned above, other flame image features such as, colour, brightness intensity, flame geometry (flame shape or flame height) are important, if a fire is to be controlled. In order to enhance the accuracy of video fire detection, Researchers, such as Xiong, Cballero, Wang, Finn and Peng, all report the need to use more than one fire feature. A review of the literature by Gottuk and Dinaburg, 2010 reveals that, spectral, spatial and temporal properties of flame images can also be used to identify flame characteristics. Video fire detection can be conducted effectively both indoors and outdoors. Of special interest is the proposal that video fire detection can prevent the spread of forest fires, if caught in the early stages. This is enabled by the ability of the detection system to operate from different heights and distances. (Liu, Hadjisophocleous, Ding, & Lim, 2012), In addition, various detection algorithms can also be used to further analyse the properties of the flame images, colour strength height.

2.2 Video fire detection for forest fires

As suggested above, one primary objective of video fire detection design is to enable the prevention of forest fires. Over the years, researchers have studied a variety of video fire detection methods. Much research has been conducted in this area, especially in Spain and China, Table 2.7 shows the statistical results of the countries that have researched video fire detection with the aim of finding a means of protection against forest fire spread.

2.2.1 Satellite method

For forest fires, video fire detection offers the best chance of protection against fire spread. Beginning in 1991, many countries and researchers have analysed the performance and effectiveness of video fire detection technology for the prevention of undisciplined spread of forest fires. Satellites infrared cameras, digital cameras, unmanned aircraft systems, and IP cameras are also used in video fire detection technology. Satellites are able to capture Advanced Very High Resolution Radiometer (AVHRR) images, a special category of images.

In 1991, the Normalised Difference Vegetation Index (NDVI) method was developed to analyse AVHRR images for use in the detection of forest fires. Equation 2.67 shows the NDVI formula. (Lopez, Gonza;lez, Llop, & Cuevas, 1991)

$$NDVI = \frac{Channel\ 2 - Channel\ 1}{Channel\ 2 + Channel\ 1} \qquad 2.67$$

The normalised difference vegetation index (NDVI) calculation results are representative of the various colours. Table 2.8 shows the summary of colour relative NDVI values

Some researchers have used not only the NDVI calculation for forest fire analysis but also hot pixel analysis (PEREIRA & SETZER, 1993). The channels 1 and 2 present the reflectance of visible and near infrared. Channel 1 is 0.58-0.68μm. Channel 2 is 0.73-1.1μm (Fernández, Illera, & Casanova, 1997). In 1997, some researchers adopted a statistical method to analyse AVHRR images for the study of forest fire risk (Gonzalez-Alonso, Cuevas, Casanova, Calle, & Illera, 1997). The main aim of forest fire detection by video is the prevention of large-scale forest fires, thus flame and smoke detection can also be useful in video fire detection technology (Fang & Huang, 1998). To analyse AVHRR images in the prevention of forest fire, measurement of Fuel Moisture Content is (FMC) is another approach that can be used (Chuvieco, Aguado, Cocero, & Riaño, 2003).

Satellites are able to capture not only AVHRR images but also digital images. To analyse satellite digital images, classification fire and non-fire in images is important. In 2005, Florent Lafarge, Xavier Descombes and Josiane Zerubia proposed the use of the Support Vector Machine (SVM) classification method. SVM is a probability calculation method used in the analysis of fire texture characteristics.

2.2.2 Infrared method

Distributed Environmental Disaster Information and Control Systems (DEDICS) use infrared camera technology. DEDICS contain a number of integrated components including threshold-based detection, detection of bright oscillations, visual/infrared matching, memory-based utilisation, motion filters, size and shape detection, meteorological detection, and solar conditions (Ollero, Arrue, Martinez, & Murillo, 1999). When a certain value for each component is lower than a certain number (determined by a rule), there is zero possibility of forest fire.

Digital cameras are also popular for detecting forest fire, but the flame analysis method is different. Wavelet and Fast Fourier Transform (FFT) are two different methods of analysis used (Contour Based Forest Fire Detection Using FFT and Wavelet, 2008).

Colour space modes namely RGB, YCbCr, CIELAB, HSI and HS'I can also be used to evaluate forest fire images (Krstinić, Stipaničev, & Jakovčević, 2009). In 2012, Vipin V used the YCbCr and RGB colour space models for detection of forest fires (V, 2012) (Roberto, 2014). In some research, colour space models are used only to segment flame images. In 2009, Dengyi Zhang, Shizhong Han, Jianhui Zhao, Zhong Zhang, Chengzhang Qu, Youwang Ke and Xiang Chen not only used a colour space model for segmentation of fire images, they also worked with back propagation neural networks (BPNN) to recognize forest fire (Zhang, et al., 2009).

2.2.3 Unmanned Aircraft System and Internet Protocol

Today, the development of unmanned aircraft systems (UASs) and of (IP) cameras is important because satellites have limitations regarding forest surveillance, including, for example, the need for large economic investment, the low resolution of the images captured, and the fact that cloud layers can affect visibility (Merino, Caballero, Ramiro Martinez - de - Dios, Maza, & Ollero, 2011) (Roberto, 2014). UASs, however, offer real-time fire monitoring, and provide geographic coordinates. Analysis methods involved in the use of UASs and IP cameras are different. A review of the literature reveals that IP cameras use K-Singular Value Decomposition (K-SVD) method to analyse forest fire images. UASs adopt the probability analysis method. Probability values include fire $f_k = p(F_{k,t} = 1)$ and fuel exhaustion $q_k = p(Q_{k,t} = 1)$.

As indicated above, when discussing video recognition of the existence of flame within a forest, it is not surprising that this mode of detection can also be the first step leading to the prevention of fires taking hold and spreading out of control (Mathi & Latha, 2016). The methodology uses a computerised vision-based approach for the detection of flame in specific images. The detection approach includes spatial-temporal flame modelling and dynamic texture analysis. Researchers have used different approaches to overcoming detection errors in a variety of outdoor environments.

2.3 Video fire detection in other environments

Hence as indicated above, Video fire detection technology is popularly in use in the prevention of forest fire because the monitoring system can be controlled remotely. Remote controlled monitoring is critical in inhospitable environments to avoid loss of life.

A review of the literature reveals that video fire detection has been used to recognize fire by its physical properties. In 1993, Glenn et al. (Healey, Slater, Lin, Drda, & Goedeke, 1993) tested an automatic real-time fire detection method using colour video input. Their video fire detection algorithm analysed the spectral, spatial and temporal properties of fire images.

In 1994, Noda et al. (Noda & Ueda, 1994) researched an image-processing method to detect flame. They recorded thermal images using infrared cameras. Colour images were recorded using a CCD camera. Their analysis method included thermal values as well as red, green, and blue colour elements.

In 1996, Simon Foo (Foo, 1996) researched video fire detection for the prevention of fires occurring in aircraft dry bays and engine compartments. A rule-based machine vision approach using statistical measures was employed for real-time fire detection. The statistical measures used were the calculation of the median, the standard deviation, and the first-order moment measures of histogram data. Results from this calculation approach showed that the thresholds of the median, the standard deviation, and the first-order moments are 140.0, 700.0, and 1.3 x 10^7 respectively. When the median, the standard deviation, and the first-order moments exceed the predetermined threshold, the images are likely fire images. Foo also investigated fire spread. Foo's approach to fire spread was to analyse the change or motion observable in two images.

In 1999, Song Wei-guo, Fan Wei-cheng and Wu Long-biao used BPNN to detect fire (Song, Fan, & Wu, 1999). Their research examined six image characteristics, including area, edge, shape, flame pulsation, layer, and motion.

Prior to the year 2000, detection and recognition of flame in images was only a one-approach application. In 1999, Hideaki Yamagishi and Jun'ichi Yamaguchi combined two approaches to detect flame (Yamagishi & Yamaguchi, 1999). The two approaches were the image processing method and the neural network method. The primary objective in the image processing method is extraction of the flame region. The image processing method adopts the RGB colour space model and the HSV colour space model. (The RGB colour space model and HSI colour space model is explained above in Chapter 1.) The HSV and HSI colour space models differ in terms of their calculation of brightness (I and V). The neural network method is used for recognition of the flame region. The output layer has two output units (fire flame or non-fire flame).

A review of the literature reveals that in the early stages of video fire detection, researchers used algorithms with the image processing method, statistical measures, and the artificial neural networks (ANNs) approach. Some researchers combined two different approaches in their studies.

Before 2000, satellites, charge coupled devices (CCD) cameras, black-and-white (BW) cameras, panoramic annular lenses, and moving cameras were used by researchers. Satellites were useful in the prevention of forest fire.

A review of the literature from 1991 to 2015 reveals that many researchers studied different video fire detection technology modes, such as flame detection, smoke detection, and fire detection. Many different kinds of video fire detection used the image processing method, statistical method, Artificial Neural Networks (ANN), or combined all methods. Table 2.9 to 2.11 show the development of video fire detection method from 1991 to 2000, 2001 to 2010, and 2011 to 2015.

Chapter 3 Methodology

The video fire detection methodology used in this study involves separate processes for segmentation of images, recognition of targets, and the tracking of fire regions. Image segmentation is the first step because it is key to the recognition of the selected target. After a target area is successfully identified as a fire region, the tracking process then, makes it possible for the fire to be extinguished.

3.1 Image segmentation process

Segmentation algorithms used in image processing can divide images into two necessary regions: foreground and background. Generally, the foreground is the image’s target, which, for the purposes of video fire detection, is the fire region. The background is the surrounding environment. Image segmentation is a form of spatial analysis. The segmentation algorithm described in this chapter is the Otsu threshold method. The Otsu threshold method was proposed by Nobuyuki Otsu in 1979 (Otsu, 1979). In the early stages of its development, the Otsu method was used for segmentation of brain tumours in Magnetic Resonance Imaging (MRI) (Jeevitha & Narendrain, 2013).

The Otsu method provides a way of automatically selecting the threshold value. The concept is a cluster analysis of the flame region, followed by the segmentation of the images. Flame brightness is an important characteristic. Firstly, colour images of flames, as indicated above I the previous Chapter must be converted to greyscale images for generation of a grey-level histogram.

By calculating the histogram and the probability method of the intensity level, the Otsu method can automatically obtain the threshold value. From the results of the threshold value, binary data can be obtained from the greyscale images. In the following section, two important steps are described:

1. Conversion of colour images to grey-level images
2. The algorithm used in the Otsu threshold method

Figure 3.1 depicts the flow diagram of the Otsu threshold method. If the captured images are colour images, they have to be transformed to greyscale images. To do this, the YIQ colour space model is used. Equation 3.1 shows the conversion expression for transforming colour images to greyscale images.

$$Y(x,y,t) = 0.299 \times R(x,y,y) + 0.587 \times G(x,y,t) + 0.114 \times B(x,y,t) \qquad 3.1$$

where $Y(x,y,t)$ is the luminance value of output images in each pixel over time. $R(x,y,t)$, $G(x,y,t)$ and $B(x,y,t)$ are primary colours of input images in each pixel again, over time.

When the luminance value for all pixels from the input images has been calculated, the algorithm enables the transformation of the images to produce greyscale images as an output. The grey-level histogram is then, obtained from the grey-scale images. Figure 3.2 depicts the flow diagram of the colour to a greyscale image conversion method. The greyscale levels range from 0 to 255 (256 values). The total number of pixels depends on the image resolution. The x-axis shows the grey-levels and the y-axis shows the number of pixels. The histogram made from the greyscale fire images, the colour fire images, and the grey colours fire images are explained in Chapter 4.

3.1.1 Single threshold method

The traditional algorithm of the Otsu threshold method is able to segment the background as a histogram showing the distribution of different quantity of grey colour image. The results of single the threshold approach is explained in chapter 4. The Otsu algorithm supposes that all pixels dichotomise into background and objects, or vice versa. The algorithm of the Otsu threshold method requires four main steps:

1. Probabilities of objects and background occurrence. $(\omega_{objects}, \omega_{background})$
2. Objects and background mean levels. $(\mu_{objects}, \mu_{background})$
3. Objects and background variances. $(\sigma_{objects}, \sigma_{background})$
4. Discriminant analysis of the "maximum between class variance." (σ_B^2)

When the discriminant analysis has obtained the maximum value of between class variance $\left(\max\limits_{1 \le k \le 255} \sigma_B^2(k)\right)$, the threshold value is obtained from greyscale levels (range from 0 to 255).

Normally, the greyscale levels of fire images should be larger than the background images. Images of fire should be from threshold value (k) to the maximum grey-level (255). The background should extend from the minimum grey-level (0) to the threshold value (k). Figure 3.3 depicts a histogram analysis using the Otsu method.

3.1.2 Multi-threshold method

Some fire images, however, are unable to use a single threshold value for image segmentation. In these cases, the multi-threshold approach should be considered. In the analysis approach used in this Thesis, first, the algorithm assumes six thresholds. Normally the algorithm of the multi-threshold method has different stages. The first stage is calculation of the greyscale level, from 0 to 255. The second stage is calculation of the grey-level, from $k(1)$ to 255. Figure 3.4 depicts the example of histogram analysis of the multi-threshold method.

The multi-threshold method is able to subdivide fire images effectively because it can obtain different threshold values from calculations. Figure 3.5 depicts the flow diagram of the Otsu multi-threshold approach. For the effective segmentation of images, two significant elements must be considered: the selection of the threshold and the number of threshold values.

Normally, different images have different recorded regions of interest; consequently, the distributions of their histograms are, likewise different. For fire images, the distribution of the histogram is similar to a Rayleigh distribution when the fire occurs indoors and in dark environments. In fact, the Rayleigh distribution is not completely reflected in fire images because the fire image region has a high intensity level. Figure 3.6 depicts the distribution curve of fire images and the estimated fire region.

Although the Rayleigh distribution is not completely similar to the histogram of fire images, some researchers have studied the modified Otsu method for segmentation of images (Wan, Wang, Sun, & Hao, 2010). The development of the Rayleigh distribution algorithm is based on the traditional Otsu method. First, the images must be grey colour images otherwise colour images must be transformed to the colour, grey. (In Section 3.1, the transformation methods were introduced.) The theory of the Otsu method is based on the Normal distribution. The theory of the modified Otsu method is based on the Rayleigh distribution. The probability density function of the Rayleigh distribution is:

$$f(x;\sigma)=\frac{x}{\sigma^2}e^{-x^2/\left(2\sigma^2\right)} \quad x>0 \tag{3.2}$$

In the modified Otsu method, calculations are also required to determine the probabilities of class occurrence of object and background $(\omega_{object}, \omega_{background})$, class mean level of object and background $(\mu_{object}, \mu_{background})$, and class variance of object and background $(\sigma_{object}, \sigma_{background})$. For the traditional Otsu method, the calculation is able to obtain the threshold from the results of maximum between class variance $\lambda_B^2(i^*)=\max \lambda_B^2(i), 1\le i\le L$. For the modified Otsu method, the calculation is able to satisfy equations 3.3, 3.4, and 3.5.

$$\lambda_{object}^2(i)=\mu_{object}^2(i)-\sigma_{object}^2(i)/2 \tag{3.3}$$

where $\lambda_{object}^2(i)$ is the object parameter of the Rayleigh model in the histogram level (i).

$$\lambda_{background}^2(i)=\mu_{background}^2(i)-\sigma_{background}^2(i)/2 \tag{3.4}$$

where $\lambda^2_{background}(i)$ is the background parameter of the Rayleigh model in the histogram level (i).

$$\lambda^2_B(i) = \omega_b(i)\omega_f(i)(\lambda_f(i) - \lambda_b(i))^2 \qquad 3.5$$

From the calculation of the Otsu multi-threshold approach and the modified Otsu method (Rayleigh distribution analysis), it is possible to obtain the threshold value. Figure 3.7 depicts the flow diagram of modified Otsu method (Rayleigh distribution analysis). Two major criteria are used for deciding on the optimal threshold:

1. The multi-threshold approach $k_O(i)$ and the modified Otsu method $k_{MO}(i)$
2. The numbers of pixels $n_O(i)$ and $n_{MO}(i)$

The optimal threshold can be obtained by comparing the two major criteria and the multi-threshold calculation method. From the calculation results of the optimal threshold value, the modified threshold method more effectively segments the images. Chapter 4 provides the results in detail. Figure 3.8 depicts the flow diagram of the optimal threshold selection method.

3.2 Recognition method

When an object can be extracted from the background, generally that extracted object is the target. In the event, it is not, however, necessarily the recognition target. There is a possible segmentation of the wrong target and accuracy needs to be enhanced. To recognise fire in images, the important fire characteristics include (1) flame colour, (2) flame height, (3) flame shape, (4) flame light intensity, and (5) flickering frequency. However, most of these characteristics can also be created by false images. Analysis of accuracy is vital

For digital images, flame height analysis is an appropriate method for recognition of fire images. Flame height analysis is also important to fire engineers because the diffusion that happens during the spread of the flame can ignite other objects.

3.2.1 Natural fire flame height analysis

Digital image technology enables images to be used to quantify the height of flame diffusion. Equation 3.6 shows the calculation method for the diffusion of flame height in images.

$$h_{images} = (y_{bottom} - y_{top}) \times p_{size} \qquad 3.6$$

where

h_{image} is the flame height in the images

$y_{bottom} - y_{top}$ is the number of flame pixels

p_{size} is the pixel size of the images

The image sensors dominate the pixel size and the number of pixels can be obtained from digital images. The digital images can show the bottom level (y_{bottom}) and top level (y_{top}) of the flame region. From the bottom level and the top level of the flame region, it is possible to obtain the number of pixels. For calculating the real heights from images, photograph technology is also used.

$$\frac{1}{f} = \frac{1}{d_{images}} + \frac{1}{d_{object}} \tag{3.7}$$

Equation 3.7 can be rewritten as follow:

$$d_{images} = \frac{\left(f \times d_{object}\right)}{\left(d_{object} - f\right)} \tag{3.8}$$

$$d_{object} = \sqrt{L_c^2 + H_c^2} \tag{3.9}$$

where H_c is height of digital camera. L_c is horizontal distance from digital camera to pool fire. D_c is images distance from fire sources to the image sensors.

Equation 3.10 shows the calculation of flame height in images. The actual flame height can be calculated, based on the image height, using the relevant equation of focal length (f), object location (d_{object}), and image distance (d_{images}).

$$h_{objects} = h_{images} \times \frac{d_{objects}}{d_{images}} \tag{3.10}$$

Chapter 4 provides details of the experimental setup and the results. Figure 3.9 and 3.10 depicts a schematic of the digital camera and fire pool setup.

3.2.2 Flickering frequency analysis

Flame flickering frequency is an important parameter for the recognition of flame from images.

A commonly used method in video fire detection is the Fast Fourier Transformation (FFT). However, a simple counting method can also be used for flame flickering analysis. Using such a method saves computer processing time.

The analysis of flame from images, in this thesis, uses the simple counting method. Equation 3.11 shows the simple counting method. In Chapter 4, the counting results have been detailed.

$$f = \frac{\text{Number of flickering}}{\text{Length of time period}} \qquad 3.11$$

where f is the frequency (Hertz / Hz).

However, the simple counting method has a significant limitation in video fire detection in terms of the frame rate. This limitation is not negligible, as it affects the results.

3.2.3 Multiple Logistic regression

Logistic regression is a useful statistical method to achieve better accuracy and faster processing when identifying a flame (Kong, Jin, Li, & Kim, 2016). This method enables the calculation of the class membership probability for one of the two categories

(Dreiseitl & Ohno - Machado, 2002). Therefore, fire and non-fire can be identified in this way. Equation 3.12 shows the general equation of logistic regression.

$$f(y_i) = p_i^{y_i}(1-p_i)^{1-y_i} \qquad 3.12$$

where i is the frame of a video stream and p_i is the fire probability for i video frame.

The calculation output of y_i is the probability density function of fire classification. If the calculation output of y_i is 0, then the images captured is possibly non-fire. From the experimental data, logistic regression can recognise fire image or non-fire image. The coefficient β obtain by the calculation of training data.

$$Const + \beta_1 x_1 + \beta_2 x_2 + \cdots\cdots\cdots + \beta_n x_n > 0 \qquad 3.13$$

In appendix E shows the detail calculation of logistic regression

3.3 Tracking method (optical flow analysis)

In video fire detection technology, the diffusion phenomenon is also an important parameter. Diffusion includes flame spread, smoke spread, and fire plumes. Of interest is that optical flow analysis can effectively predict the extent of fire movement and therefore spread. Optical flow analysis is effective at tracking smoke and flame displacement. A review of the literature reveals that generally, determining optical flow can be achieved by using the Lucas-Kanade method and Hom-Schunck method. The optical flow of Lucas-Kanade method and Hom-Schunck Method are shown in the appendix D.

Figure 3.11 shows the theory of optical flow analysis. Generally, real – time images are recorded over a period of time. The coordinates of location (x and y) and time (t) change. The first image of objects (I) is located at the (x) and (y) in time (t). The second image of objects (I) is located at the (x+δx) and (y+δy) in time (t+δt). When the first and second images are analysed, the object motion can be extracted.

In the source code for this experiment, the optical flow analysis uses the Gunnar Farneback algorithm (Farneback, 2003). The primary method of the Gunnar Farneback algorithm is the estimation of two frame motions. Equation 3.15 shows the general equation of the Gunnar Farneback algorithm.

$$f(x) \sim x^T A x + b^T x + c \qquad 3.14$$

where A is a symmetric matrix, b is a vector and c is a scalar.

The polynomial expansion of the Gunnar Farneback algorithm can also be used to estimate displacement fields. Equation 3.14, 3.15 show the polynomial expansion.

$$f_1(x) = x^T A_1 x + {b_1}^T x + c_1 \qquad 3.15$$

where f_1 is an original signal from first images.

$$f_2(x) = f_1(x-d) = (x-d)^T A_1 (x-d) + {b_1}^T (x-d) + c_1$$

$$f_2(x) = f_1(x-d) = \left(x^T - 2xd + d^T\right) A_1 (x-d) + {b_1}^T x - {b_1}^T d + c_1$$

$$f_2(x) = f_1(x-d) = \left(x^T A_1 - 2A_1 xd + d^T A_1\right)(x-d) + {b_1}^T x - {b_1}^T d + c_1$$

$$f_2(x) = f_1(x-d) = x^T A_1 x + {b_1}^T x - 2b_1 A_1 dx + 2A_1 d^T x + d^T A_1 d - {b_1}^T d + c_1$$

$$f_2(x) = f_1(x-d) = x^T A_1 x + (b_1 - 2A_1 d)^T x + d^T A_1 d - {b_1}^T d + c_1$$

When $A_1 = A_2;\ b_2 = b_1 - 2A_1 d;\ c_2 = d^T A_1 d - {b_1}^T d + c_1$

$$f_2(x) = x^T A_2 x + {b_2}^T x + c_2$$

where f_2 is a new signal from second images and d is a global displacement

The Optical flow technique in video fire detection and as indicated above, is also used in the analysis of fire plumes, therefore the optical flow technique is useful in any further study of video fire detection.

The principle of motion analysis relates to the comparison of the variation in image pixels thus the threshold value is a necessary part of the software. In the software program, the presence of the sensitivity value. When two images have been compared, the results are found to be equal to the pre-set sensitivity value. The software is able to detect the flame motion while the motion of flame images is recorded on a hard disk.

Chapter 4 Experimental arrangements and results

Video fire detection technology is software based (Scheffey, 2016) requiring the processing and analysis of detected fire images by the software itself. The experiment was composed of three stages.

In the first stage, the MATLAB R2020a Image Processing Toolbox is used to separate the flame images. The toolbox provides different calculation platforms for the analysis of the images, such as image segmentation. Traditional Otsu's method is one of the tools in MATLAB.

In the second stage, Visual C++, C++, and Microsoft Foundation Class (MFC) are used to separate the still images for further study.

In the final stage, C++ and Open CV are used to develop a computer program to analyse the real-time video images. The operating system in the video fire system uses Windows XP Professional, and Windows 10. Table 4.1 summarises the three stages of the experimental study.

This chapter also describes the experimental configuration arrangements and the results. The source code is given in appendix B.

In the experimental study, propanol fuel was used to create the fire and the flame images were recorded. Figure 4.1 reveals the properties of propanol fuel while Figure 4.2 indicates the different pool diameters that can be used in future studies.

4.1 The first stage of the experimental study

4.1.1 Computer software

In the first stage of the experiment, the properties of the digital images were reviewed. The Otsu method (Otsu, 1979) was adopted to segment the colour flame images. The MATLAB source code is listed in Appendix A. Figure 4.3 depicts the original colour images alongside the binary images resulting from the MATLAB calculations.

4.1.2 Experiment

The experimental results show that the traditional Otsu method does not completely segment all flame images. Some regions of the flame lack fidelity, which indicates that the traditional method needs to be improved. Figure 4.4 illustrates the segmentation results (Distortion).

The literature review revealed that the traditional Otsu method requires greyscale images to calculate the threshold value. In the experimental study using MATLAB, the colour flame images were converted by an algorithm involving the weighted sum of the red, green, and blue components. Equation 4.1 shows the algorithm for the RGB values and the greyscale values (Mathworks.com, 1994 - 2016).

$$0.2989 \times R + 0.5870 \times G + 0.1140 \times B \qquad 4.1$$

where R is the value of the red colour, G is the value of the green colour, and B is the value of the blue colour. Figure 4.5 depicts the greyscale images and the original flame images.

4.2 The second stage of the experimental study

4.2.1 Computer software

Generally, video fire detection is based on the use of software. To further this present study of video fire detection technology, the second stage of the experiment employed computer languages and the programs C++, and C++ Microsoft Foundation Class (MFC)) to analyse the static flame images including thermal images and colour images. Figure 4.6 shows the computer analysis of colour images and thermal images.

A thermal camera is used to capture thermal images. Specific associated computer software is used to analyse the images. Further source code also can segment the flame images with the Otsu method. Firstly, thermal images have to be converted to a greyscale image. Figure 4.7 shows the thermal images and greyscale images.

From the data supplied by the greyscale images, the source code can also create a histogram. Figure 4.8 shows the histogram and the Otsu calculation result. From the Otsu calculation results he greyscale image can be converted to a binary image

4.2.2 Experiment

The results from the above experiment revealed that although the thermal images are able to separate the flame region from the background, the software cannot clearly identify the condition of the thermal images, images which we can observe for ourselves. Thus, it is better for colour images to be used, rather than thermal images in the analysis of video fire detection technology. Figure 4.9 shows the binary images.

From the source code of the Otsu method, the colour images can also be used to segment the background flame region. Figure 4.10 depicts the results, including colour flame images, greyscale images, and image histograms.

In the second stage of this experiment, a modified source code was used to create the histogram, calculate the multi-threshold result, and segment the flame images.

In addition, the modified source code can improve the lack of fidelity problem. The source code is given in appendix B. Figure 4.11 depicts the greyscale images, histogram and the segmentation results. The experimental results reveal that the first threshold value is unable to clearly segment the image. Table 4.2 shows the multi-threshold results.

In total, six threshold values were picked out, the optimal-threshold value, is that which most easily enables segmentation of the flame. The Rayleigh distribution of the Otsu method contributes to an analysis of the images, by comparing the calculated results. Table 4.3 and Figure 4.12 show the calculation results.

When the flame images can be segmented from the background using multi-threshold analysis, it is seen that the number of pixels has been reduced. If the segmentation results overlap the original images, this means successful segmentation of the flame from the image background has occurred. Figure 4.13 shows the experimental results.

After the Otsu method has segmented the flame region in the image, the next item in the video fire detection procedure is the recognition of whether the revealed image is ‘fire” or “non-fire”. However, for further clarity, the identification of the centroid coordinates, flame shape, flame height, flame colour, and flame light intensity, that is the physical characteristics of the flame images, is necessary. By these means, contributions can be made to increased accuracy of the analysis.

How the centroid coordinates of the flame shape can be used in distinguishing between "fire" and "non – fire" has been presented in the paper (Wong and Fong, 2014). Figure 4.14 shows a sample of fire and non-fire images.

In the above reported study, it was found that using flame region centroid coordinates, the Nearest Neighbour (NN) algorithm can be used to recognise whether or not an image is a fire image. Figure 4.15 depicts the centroid analysis results for fire images and non-fire images.

The experimental results above, however, reveal that the nearest neighbour algorithm is not very accurate in recognising a fire from the images if the algorithm uses only centroid coordinates. In addition, the size of the database in relation to the fire image and non-fire images is a critical factor.

In 2014, a fire size estimation method was studied based on details of the author's work. (Wong & Fong, 2014) Table 4.4 shows the numerical results. From the numerical HRR results and the different pool diameters, two heat release rate curves were provided. Figure 4.16 depicts the HRR curve. The results of the HRR curve reveal that the heat release increases when the pool diameter increases.

However, more research is required to understand how to couple HRR information with video fire detection technology.

4.3 The third stage of the experimental study

4.3.1 Computer software

In these above experiments computer software was used to record and analyse the flame images. The software used, included Microsoft Visual Studio Professional 2013 Version 12.0.21005.1 REL and Open CV Version 2.4.10. Figure 4.18 shows the screen output display. Table 4.6 shows the numerical data derived from the equipment including the pool fire diameter, the quantity of propanol, resolution of images and the distance between the webcam and the fire source.

The flame height data and the sampling counting method is used in video fire detection. The flame data is used to analyse flame flickering characteristics. Figure 4.19 and 4.20 show the flame image height (flame motion). Figure 4.21 depicts a histogram of flame flickering. The experiment showed that the flame flicker is within the range of 8 to 10.

4.3.2 Experiment

In the third experimental, real-time images were captured using a webcam. The flame height can be estimated from the video images if careful attention is paid to the configuration of the experiment and to the technical specifications of the webcam. Table 4.5 shows the webcam specification. Figure 4.17 shows the experimental configuration.

The data from the images including segmentation, recognition, and tracking performances was directly analysed and processed using the software. Appendix B shows and describes the software detailed. The software also recorded the numerical data on the hard disk. The numerical data is thus available for future analysis in relation to future experiments. These data include the timings, threshold values, maximum grey levels, areas of regions of interest, heights of the regions of interest, centroid coordinate and the red, green, and blue colour values.

In the experiment, the webcam recorded fire images at intervals between fire ignition to fire extinction and non-fire images. The regression coefficients revealed that logistic regression calculations could be used to distinguish between fire and non-fire images. The current video fire images uses logistic regression to recognise fire. Then, the fire can be tracked, by means of recorded images and spread of the flame or smoke. The optical flow method was the appropriate tool used to track the direction of smoke spread. Figure 4.23 shows the experimental result.

A video showing the above results provides an informative example, illustrating the use of video detection technology with digital image processing, motion detection, and machine learning methods. All video detection technology provides individual examples of items of information, which can be used to better the use of Video fire detection. Figure 4.24 presents the flow diagram of video fire detection method used in this research study.

The results of multiple logistic regression calculations related to fire image recognition are shown in the following equation.

$$\boldsymbol{P_{fire} = 0.014427 \times Cent(X) - 0.00273 \times Cent(Y) + 0.067545 \times Ht - 0.00017 \times ROI - 0.02166 \times R + 0.040735 \times G - 0.02869 \times B} \qquad 4.2$$

The multiple logistic regression results include the coefficient of the centroid coordinate X and Y (Cent(X), Cent(Y)), flame height (Ht), region of interest (ROI), red (R), green (G), and blue (B) colour. Equation 4.1 can be used to obtain the probability of a fire, P_{fire}. Appendix E provides the full solution of the logistic regression.

Chapter 5 Conclusions

The objective of this study is an investigation of video fire detection technology. Successful results were achieved which provide feasible suggestions regarding improvements that could be used in fire detection systems technology. From the experimental study, by means of the use of a computer program method, image processing technology, logistic regression, and fire image characteristic such as flickering frequency, recognition the fire in images was achieved. From the optical flow analysis, the computer program is able to track the flame spread direction. From the flame height, the empirical formula is able to predict the fire size. Further findings from this study are presented as follows:

Since 1991, video fire detection technology has been studied using a variety of techniques such as image processing, signal processing, statistical analysis, and ANN. Video fire detection techniques are able to detect flames, smoke, and fire plumes. Thus, forest fire surveillance and preservation of natural heritage sites are two domains that could benefit from the video fire detection techniques studied in this project, as applied to the developing early–stages of a fire.

Frequently, in this study different colour models were same to use for segmentation include the RGB colour space model, and YIQ colour space model.

The software-based methods studied in this project, offer a different imaging approach. It should be noted that the study required the application of knowledge from several distinct fields. Illustrations are provided in Figure 5.1

For segmentation of the fire images, the Otsu algorithm was used to segment the flame images. However, it was found that the traditional Otsu method cannot completely segment all flame images in all varieties of environments. A Modified Otsu method was therefore proposed for use in this study and better results were achieved.

For recognition of the fire images, logistic regression and the nearest neighbour algorithm were also proposed. The recognition analysis of the nearest neighbour algorithm aims not only enabled the analysis of the centroid coordinates of flames but also the logistic regression. This analysis, therefore takes account of more than one fire characteristic.

The logistic regression method is proposed to ensure greater accuracy in the recognition of fire images. In addition, the logistic regression methods includes regression learning, giving the power to improve the accuracy of fire recognition.

It was found that using optical flow analysis to tracking the flame spread and its motion direction is an important part of fire detection. The tracking method in video fire detection can reduce property loss, and protect the safety of firefighters.

Hardware, such as the computer and the digital camera were also found to play an important part. The digital camera is able to capture images, which are then analysed by the computer. If the image data can identify fire in a short period of time, the video fire detection system can likewise, immediately report the status of any fire.

5.1 Application

Research findings confirmed that, today, video fire detection technology is of high interest and is currently developed by different manufacturers. However, the development of video fire detection technologies has not yet been fully explored to the extent that an analysis algorithm can describe the fire characteristics sufficiently well. However it seems clear that further study by computer technology and image processing algorithms, will enable the use of video fire detection in such protected spaces as Dangerous Goods (D.G.) stores and Sub-divided flat is now practical.

A review of the literature revealed such as international video fire detection standards, system requirements, installation specifications, and analysis methodologies are being well recorded, possibly in readiness for further development in the light of further research. However, as yet the required specific and necessary data accuracy has not yet been fully determined, Hence the testing and commissioning of video fire detection is ready for further exploration and introduction to the public, and is anticipated as being the firefighting method of choice.

Chapter 6 Suggestions for future research

Today, video fire detection has been the subject of many research studies. Different approaches are currently being studied. For instance, fire naturally, can occur in a variety of areas and environments such as atriums, tunnels, in E and M plant rooms, forests, warehouses, historic buildings and aircraft hangars. However, the video fire detection algorithm currently is not widely used in Hong Kong. Thus, further experimental study is necessary and for the comfort of society, to verify different algorithms to enable the use of video fire detection, anytime and anywhere.

Better techniques that can reduce false alarms are also needed. Improved technology providing greater accuracy in video fire detection technology is a further research objective. The accuracy and compatibility of each video fire detection component identified in research studies need further verification and validation in order to further enhance and enlarge the use of this fire recognition/prevention skill features.

In Hong Kong, business people intent on the development of virgin areas or those long out of use would welcome the availability of further research into the research mechanisms already present in video fire detection systems. Currently, the design of the video fire detection product is very bulky and expensive. From the television program "Guide to navigation" reported that, the cost of developed video fire detection system is more than three hundred to four hundred thousand Hong Kong dollars. (Television Broadcasts, 2017) In addition, the system only uses thermal images to analyse only forest fires.

In the near future, low-cost digital cameras will be commonly used. Many different computer techniques are being rapidly developed. In addition, the video fire detection techniques are destined to become more commonly used in the delivery of fire services, providing people's trust in the accuracy of video fire detection systems grows.

From the study of video fire detection technology presented in this thesis, it is seen that segmentation, discrimination, and tracking are important component, with each component enabling a better than average fire detection result. The further research suggested above, together with the achievements previously described, presents a convincing recommendation for the use of fully automatic video fire detection technology and the subsequent reduction of property loss and further protection of life

References

40/40R - Single IR Flame Detector. (2010). Retrieved 10 24, 2015, from http://spectrex-inc.com/products/sharpeye/4040r

40/40U-UB - UV Flame Detector. (2010). Retrieved 10 24, 2015, from http://spectrex-inc.com/products/sharpeye/4040u-ub

Barmpoutis, P., Dimitropoulos, K., and Grammalidis, N. (2013). Real time Video Fire Detection using Spatio - Temporal Consistency Energy. *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance.*

Beer Color Laboratories. (n.d.). Retrieved 2015, from http://www.beercolor.com/color_basics1.htm

Beji, T., Merci, B., Verstockt, S., and Walle, R. V. (2012). On the Use of Real - Time Video to Forecast Fire Growth in Enclosures. *Fire Technology,* 1 - 20.

Bosch, I., Gomez, S., Molina, R., and Miralles, R. (2009). Object Discrimination by Infrared Image Processing. In *Bioinspired Applications in Artificial and Natural Computation* (pp. 30 - 40). Springer.

(2010). *CCD and CMOS sensor technology Technical white paper.* Axis Communications.

Celik, T. (2010). Fast and Efficient Method for Fire Detection Using Image Processing. *ETRI Journal, 32*, 881 - 890.

Celik, T., Demirel, H., and Ozkaramanli, H. (2006). Automatic Fire Detection in Video Sequences. *14th European Signal Processing Conference.* Florence.

Çelik, T., Özkaramanlı, H., and Demirel, H. (2007). Fire and Smoke Detection without Sensors: Image Processing based Approach. *15th European Siginal Processing Conference.* Poznan.

Chapter 10 Automatic fire detection. (1990). In *Manual of Firemanship Book 9 Fire Protection of buildings* (pp. 91-98). London: HMSO.

Chapter 10 Fire Detection Systems. (1993). In J. L. Bryan, *Fire Suppression and Detection Systems* (p. 347). USA: Prentice-Hall, Inc.

Chapter 11 Smoke detectors. (1990). In *Manual of Firemanship Book 9 Fire protection of buildings* (pp. 99-108). London: HMSO.

Chen, J., He, Y., and Wang, J. (2010). Multi-feature fusion based fast video flame detection. *Building and Environment, 45,* 1113 - 1122.

CHen, T.-H. (.-H., Kao, C.-L., and Chang, S.-M. (2003). An Intelligent Real - Time Fire - Detection Method Based on Video Processing. *IEEE,* 104 - 111.

Cheng, X., Wu, J., Yuan, X., and Zhou, H. (1999). Principles for a video fire detection system. *Fire Safety Journal, 33*, 57 - 69.

Chuvieco, E., Aguado, I., Cocero, D., and Riaño, D. (2003). Design of an empirical index to estimate fuel moisture content from NOAA-AVHRR images in forest fire danger studies. *International Journal of Remote Sensing, 24,* 1621 - 1637.

Codes of Practice Minimum Fire Service Installations and Equipment and Inspection, Testing and Maintenance of Installtions and Equipment. (1998). Hong Kong.

Color Image Processing. (2010). USA: Pearson Education Inc.

Contour Based Forest Fire Detection Using FFT and Wavelet. (2008). *International Conference on Computer Science and Software Engineering,* 760 - 763.

Dreiseitl, S., and Ohno - Machado, L. (2002). Methodological Review Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics , 35,* 352 - 359.

Drysdale, D. (1999). Diffusion Flames and Fire Plumes. In *An Introduction to Fire Dynamics* (pp. 109 - 158). John Wiley and Sons.

Dungan, K. W. (2008). Chapter 2 Automatic Fire Detectors. In A. E. Cote, *Fire Protection Handbook (Twentieth Edition) Volume II* (pp. 15 - 28). National Fire Protection Association.

Duong, H. D., and Tuan, N. A. (2009). Using Bayes method and Fuzzy C - Mean Algorithm for Fire Detection in Video. *International Conference on Advanced Technologies for Communications.*

Electrochemical Cell Toxic Gas Detector. (2015). Retrieved 10 24, 2015, from http://www.safetysys.com/wp-content/uploads/2015/03/GT814-toxic-gas-detector.pdf

Fang, M., and Huang, W. (1998). Technical note - Tracking the Indonesian forest fire using NOAA/AVHRR images. *Int. J. Remote Sensing, 19*, 387 - 390.

Farneback, G. (2003). Two-Frame Motion Estimation Based on Polynomial Expansion. 1 - 8.

Fernández, A., Illera, P., and Casanova, J. L. (1997). Automatic Mapping of Surfaces Affected by Forest Fires in Spain Using AVHRR NDVI Composite Image Data. *REMOTE SENS. ENVIRON., 60*, 153 - 162.

Foo, S. Y. (1996). A rule - based machine vision system for fire detection in aircraft dry bays and engine compartments. *Knowledge - Based Systems, 9*, 531 - 540.

Gas Sensing Fire Detectors. (1993). In J. L. Bryan, *Fire Supression and Detection Systems* (p. 511). USA: Prentice-Hall Inc.

Gonzalez, R. C., and Woods, R. E. (2010). Image Compression. In *Digital Image Processing* (pp. 547 - 648). London: Perason Edition International.

Gonzalez-Alonso, F., Cuevas, J. M., Casanova, J. L., Calle, A., and Illera, P. (1997). A forest fire risk assessment using NOAA AVHRR images in the Valencia area, eastern Spain. *International Journal of Remote Sensing,* 2201- 2207.

Gottuk, D. (2008). *Video Image Detection Systems Installation Performance Criteria Research Project.* The Fire Protection Research Foundation.

Gottuk, D. T., and Dinaburg, J. B. (2010). *Video Image Detection and Optical Flame Detection for Industrial Applications.* Orlando: Fire Suppression and Detection Research and Applicatons - A Technical Working Conference.

Habiboglu, Y. H., Gunay, O., and Cetin, A. (2011). Covariance matrix - based fire and flame detection method in video. In *Machine Vision and Applications* (pp. 1 - 11). Springer.

Han, D., and Lee, B. (2009). Flame and smoke detection method for early real - time detection of a tunnel fire. *Fire Safety Journal, 44*, 951 - 961.

He, S., Yang, X., Zeng, S., Ye, J., and Wu, H. (2015). Computer Vision Based Real - time Fire Detection Method. *Journal of Information and Computer Science , 12*(2), 533 - 545.

Healey, G., Slater, D., Lin, T., Drda, B., and Goedeke, A. D. (1993). A System for Real - Time Fire Detection. *IEEE,* 605 - 606.

(2002 - 2015). *Hong Kong Fire Services Department Review.* Hong Kong: Hong Kong Fire Services Department.

Hongliang, L., Qing, L., and Sun'an, W. (2012). A Novel Fire Recognition Algorithm Based on Flame's Multi - features Fuson. *2012 International Conference on Computer Communication and Informatices.* Coimbatore.

Horng, W.-B., Peng , J.-W., and Chen, C.-Y. (2005). A New Image - Based Real - Time Flame Detection Method Using Color Analysis. *IEEE,* 100 - 105.

Hou, J., Qian, J., Zhao, Z., Pan, P., and Zhang, W. (2009). Fire Detection Algorithms in Video Images for High and Large - span Space Structures. *IEEE,* 1 - 5.

Introduction. (2010). In R. C. Gonzalez, and R. E. Woods, *Digital Image Processing* (p. 25). USA: Pearson Education, Inc.

Jeevitha, A., and Narendrain, P. (2013). BTS (Brain Tumor Segmentation) Based on Otsu Thresholding. *Computer Science, 2*(2), 53 - 55.

Jianzhong, R., Jian, W., Jian, C., and Jun, J. (2010). An Oscillation Frequency of Flame Study based on Image processing Technology and Acoustic Measurement Technology. *IEEE*, 1 - 4.

Jiao, Y., Weir, J., and Yan, W. (2011). Flame Detection in Surveillance. *Journal of Multimedia, 6*, 22 - 32.

Jin, L., N.K., F., W.K., C., L.T., W., Puyi, L., and Dian-guo, X. (2004). The motion analysis of fire video images based on moment features and flickering frequency. *Journal of Marine Science and Application, 3,* 81 - 86.

Juan, C., and Qifu, B. (2012). Digital image processing based fire flame color and oscillation frequency analysis. *International Symposium on Safety Science and Technology.* Hangzhou.

Ko, B. C., Cheong, K.-H., and Nam, J.-Y. (2009). Fire detection based on vision sensor and support vector machines. *Fire Safety Journal, 44,* 322 - 329.

Ko, B., Cheong, K.-H., and Nam, J.-Y. (2010). Early fire detection algorithm based on irregular patterns of flames and hierarchical Bayesian Networks. *Fire Safety Journal, 45,* 262 - 270.

Kong, S., Jin, D., Li, S., and Kim, H. (2016). Fast fire flame detection in surveillance video using logistic regression and temporal smoothing. *Fire Safety Journal, 79,* 37-43.

Kopilovic, I., Vagvolgyi, B., and Sziranyi, T. (2000). Application of Panoramic Annular Lens for Motion Analysis Tasks : Surveillance and Smoke Detection. *IEEE*, 714 - 717.

Krstinić, D., Stipaničev, D., and Jakovčević, T. (2009). Histogram - Based Smoke Segmentation in Forest Fire Detection System. *INFORMATION TECHNOLOGY AND CONTROL, 38,* 237 - 244.

Litwiller, D. (2005). CMOS vs. CCD: Maturing Technologies, Maturing Markets. *Photonics Spectra*.

Liu, C.-B., and Ahuja, N. (2004). Vision Based Fire Detection. *IEEE,* 1 - 4.

Liu, Z., Hadjisophocleous, G., Ding, G., and Lim, C. S. (2012). Study of a Video Image Fire Detection System for Protection of Large Industrial Applications and Atria. *Fire Technology, 48,* 459 - 492.

Lopez, S., Gonza;lez, F., Llop, R., and Cuevas, J. M. (1991). An evaluation of the utility of NOAA AVHRR images for monitoring forest fire risk in Spain. *International Journal of Remote Sensing, 12:9,* 1841 - 1851.

Maoult, Y. L., Sentenac, T., Orteu, J. J., and Arcens, J. P. (2007). Fire Detection A New Approach Based on a Low Cost CCD Camera in the Near Infrared. *Institution of Chemical Engineers*, 193 - 206.

Marbach , G., Loepfe, M., and Brupbacher, T. (2006). An image processing technique for fire detection in video images. *Fire Safety Journal, 41,* 285 - 289.

Mathi, P., and Latha, L. (2016). Video Based Forest Fire Detection using Spatio - Temporal Flame Modeling and Dynamic Texture Analysis. *Internaltional Journal on Applications in Information and Communcation Engineering, 2*(4), 41 - 47.

Mayers, H. J., and Bernstein, R. (1985). Image Processing on the IBM Personal Computer. *Proceeding of the IEEE, 73*, 1064 - 1070.

Merino, L., Caballero, F., Ramiro Martinez - de - Dios, J., Maza, I., and Ollero, A. (2011). An Unmanned Aircraft System for Automatic Forest Fire Monitoring and Measurement. *J Intell Robot Syst,* 1 - 16.

Nguyen - Ti, T., Nguyen - Phuc, T., and Do - Hong, T. (2013). Fire Detection Based on Video Processing method. *The 2013 International Conference on Advanced technologies for Communications* .

Ning, C., and Fei, D. (2012). Flame segmentation by an improved frame difference algorithm. *Fire Safety Science, 21*, 209 - 215.

Noda, S., and Ueda, K. (1994). Fire Detection in Tunnels Using an Image Processing Method. *Vehicle Navigation and Information Systems Conference Proceeding*, (pp. 57 - 62).

Normalized Difference Vegetation Index (NDVI). (n.d.). Retrieved from http://earthobservatory.nasa.gov/Features/MeasuringVegetation/measuring_vegetation_2.php

Ollero, A., Arrue, B., Martinez, J., and Murillo, J. (1999). Techniques for reducing false alarms in infrared forest - fire automatic detection systems. *Control Engineering Practice , 7*, 123 - 131.

Otsu, N. (1979). A Threshold Selection Method from Gray - Level Histogram. *IEEE, SMC* - *9*, 62 - 66.

Owrutsky, J. C., Steinhurst, D. A., Minor, C. P., Rose - Pehrsson, S. L., Williams, F. W., and Gottuk, D. T. (2006). Long wavelength video detection of fire in ship compartments. *Fire Safety Journal, 41*, 315 - 320.

PEREIRA, M. C., and SETZER, A. W. (1993). Spectral characteristics of deforestation fires in NOAA/AVHRR images. *International Journal of Remote Sensing*, 583 - 597.

Phillips III, W., Shah, M., and Lobo, N. V. (2000). Flame Recognition in Video. *IEEE*, 224 - 229.

rgb2gray. (1994 - 2016). (The MathWorks, Inc.) Retrieved from http://www.mathworks.com/help/matlab/ref/rgb2gray.html#output_argument_i

Roberto, R.-R. (2014). Remote detection of forest fires from video signals with classifiers based on K - SVD learned dictionaries. *Engineering ApplicationsofArtificial Intelligence, 33*, 1 - 11.

Scheffey, J. (2016). New Hangar Fire Protection Design Concepts. In *SFPE Handbook of Fire Protection Engineering* (pp. 1686 - 1689). Springer.

Schröder, T., Krüger, K., and Kümmerlen, F. (2014). Image processing based deflagration detection using fuzzy logic classification. *Fire Safety Journal , 65*, 1 - 10.

Schultze, T., Kempka, T., and Willms, I. (2006). Audio - video fire - detection of open fires. *Fire Safety Journal, 41,* 311 - 314.

Shi, L., Liu, X., and Liu, P. (2009). An Analysis of Fire Frame Processing and Video Dynamic Features. Springer - Verlag Berlin Heidelberg.

Sky and Telescope. (2015). (Sky and Telescope Media, an F+W, Content + eCommerce Company) Retrieved from http://www.skyandtelescope.com/astronomy-resources/astrophotography-tips/simple-astrophotography/

So, A. T., and Chan, W. L. (1994). A Computer - Vision - Based and Fuzzy - Logic - Aided Security and Fire - Detection System. *Fire Technology Third Quarter*, 341 - 356.

Song, w. g., Fan, w. c., and Wu, l. b. (1999, 7). BP Network Based Image Fire Detection Method. *Fire Safety Science, 8,* 49 - 56.

Stratton, B. J. (2005). *Determining Flame Height and Flame Pulsation Frequency and Estimating Heat Release Rate from 3D Flame Reconstruction.* Christchurch: Department of Civil Engineering University of Canterbury.

Suzuki, K., and Takehara, A. a. (2012). *Graphic probability, statistics.* 台灣: 積木文化.

Television Broadcasts. (2017). *Guide to navigation*. Retrieved from http://programme.tvb.com/news/innovationgps/

Thermal imaging for Safety and Efficiency in Public Transportation. (2016, 9 14). Retrieved from FLIR® Systems, Inc.: http://www.flir.com/traffic/blog/details/?ID=78385

Toreyin, B., Dedeoglu, Y., Gudukbay, U., and Cetin, A. (2006). Computer vision based method for real - time fire and flame detection. *Pattern Rrecognition Letters, 27,* 49 - 58.

Toyota, K. (1972). *Graphic camera construction.* Japan: 世茂出版集團.

Truong, T. X., Kin, Y., and Kin, J. (2011). Fire Detection in Video using Genetic - based Netural Network. *IEEE,* 1 - 5.

V, V. (2012). Image Processing based Forest Fire Detection. *International Journal of Emerging Technology and Advanced Engineering, 2*(2), 87 - 95.

Verstockt, S., Hoecke, S. V., Tilley, N., Merci, B., Sette, B., Lambert, P., . . . Walle, R. V. (2011). FireCube: A multi - view localization framework for 3D fire analysis. *Fire Safety Journal*, *46*, 262 - 275.

Wallpapers You need. (2014). Retrieved 2015, from http://hdwyn.com/glacier_iceberg_under_water_hd-wallpaper-14494/

Wan, L., Wang, J., Sun, X., and Hao, M. (2010). A Modified Otsu Image Segment Method Based on the Rayleigh Distribution. *IEEE*, 281 - 285.

Wang, H., Finn, A., Erdinc, O., and Vincitore, A. (2013). Spatial - Temporal Structural and Dynamic Features for Video Fire detection. *IEEE.*

Wong, A. K., and Fong, N. (2014). Experimental study of video fire detection and its applications. *Procedia Engineering, 71,* 316 - 327.

Wong, A. K., and Fong, N. (2014). Study of pool fire heat release rate using video fire detection. *3th International Hight Performance Building Conference* . Purdue.

Xie , J. (2015). *Security personnel about to be unemployed: unmanned monitoring technology how to replace the artificial interpretation.* 台北: 佳魁資訊.

Xiong, Z., Caballero, R. E., Wang, H., Finn, A. M., and Peng, P.-y. (2009). Video Fire Detection - Techniques and Applications in the Fire Industrial. In *Multimedia Content Analysis Theory and Applications* (pp. 339 - 351). USA: Springer.

Xu, Y., Zhu, X., and Xie, B. (2012). Method Design of Small - scale Fire Detection. *Journal of Computational Information Systems, 8*, 7355 - 7365.

Y. L., D. Z., F. W., and W. Q. (2001). Experimental Study on Characteristics at Early Stage of Fire. *Journal of Fire Sciences, 19,* 190 - 203.

Yamagishi, H., and Yamaguchi, J. (1999). Fire Flame Detection Algorithm Using a Color Camera. *IEEE*, 255 - 259.

Yamana, T., and Tanaka, T. (1985). Smoke Control in Large Scale Spaces. *Fire Science and Technology, 5*, 41-54.

Yu, C., Mei , Z., and Zhang, X. (2013). A real - time video fire flame and smoke detection algorithm. *Procedia Engineering*, 891 - 898.

Zhang, D., Han, S., Zhao, J., Zhang, Z., Qu, C., Ke, Y., and Chen, X. (2009). Image Based Forest Fire Detection Using Dynamic Characteristics With Artificial Neural Networks. *INFORMATION TECHNOLOGY AND CONTROL, 38*, 237 - 244.

Zhang, J., Zhuang, J., Du, H., Wang, S., and Li, X. (2006). A Flame Detection Algorithm Based on Video Multi - feature Fusion. Springer - Verlag Berlin Heidelberg.

文昌, 鄭., and 政達, 陳. (2010). 影像式火災自動偵測系統之研究. *第九屆離島資訊技術與應用研討會論文集.*

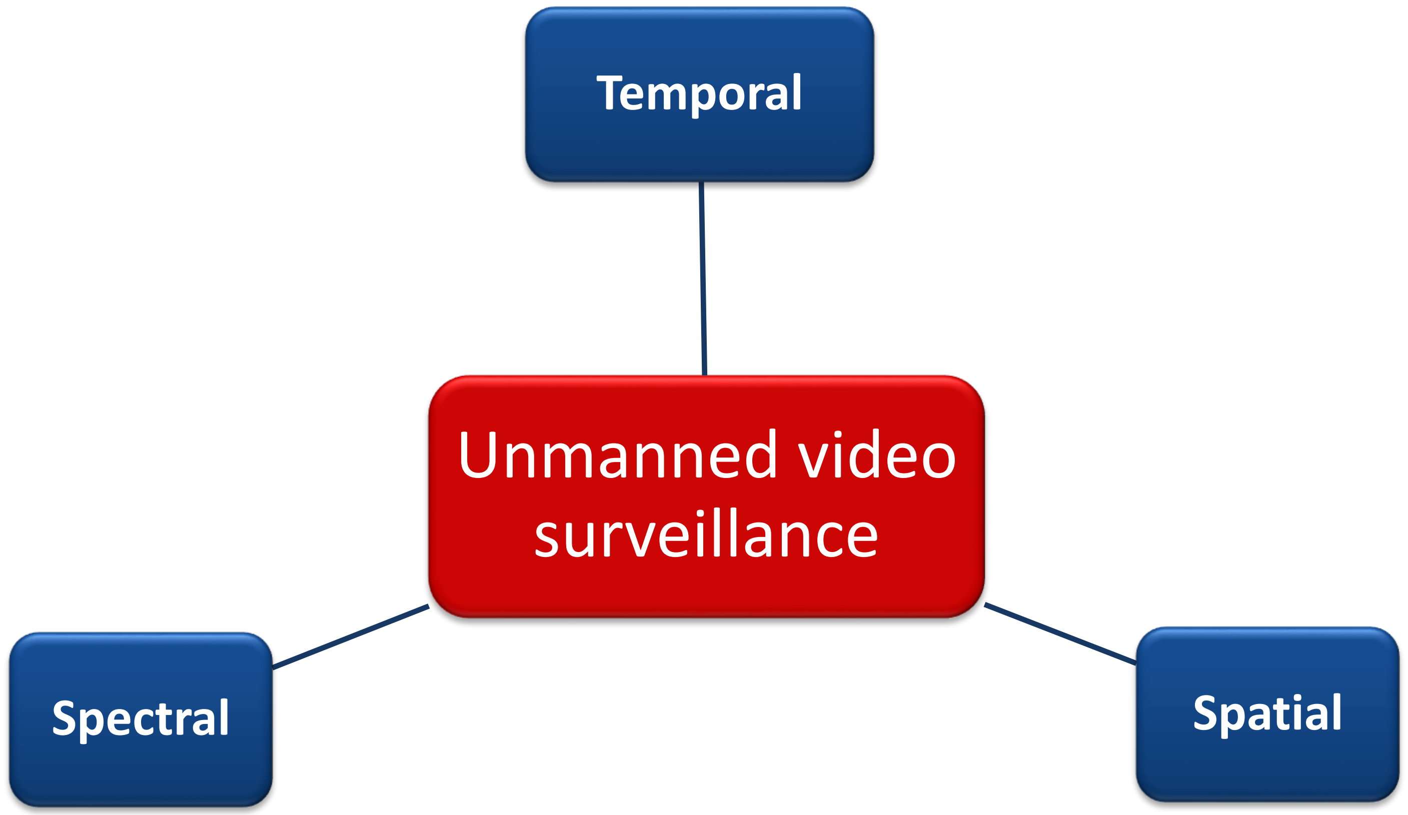


Figure 1.1 Properties of dynamic objects for unmanned video surveillance

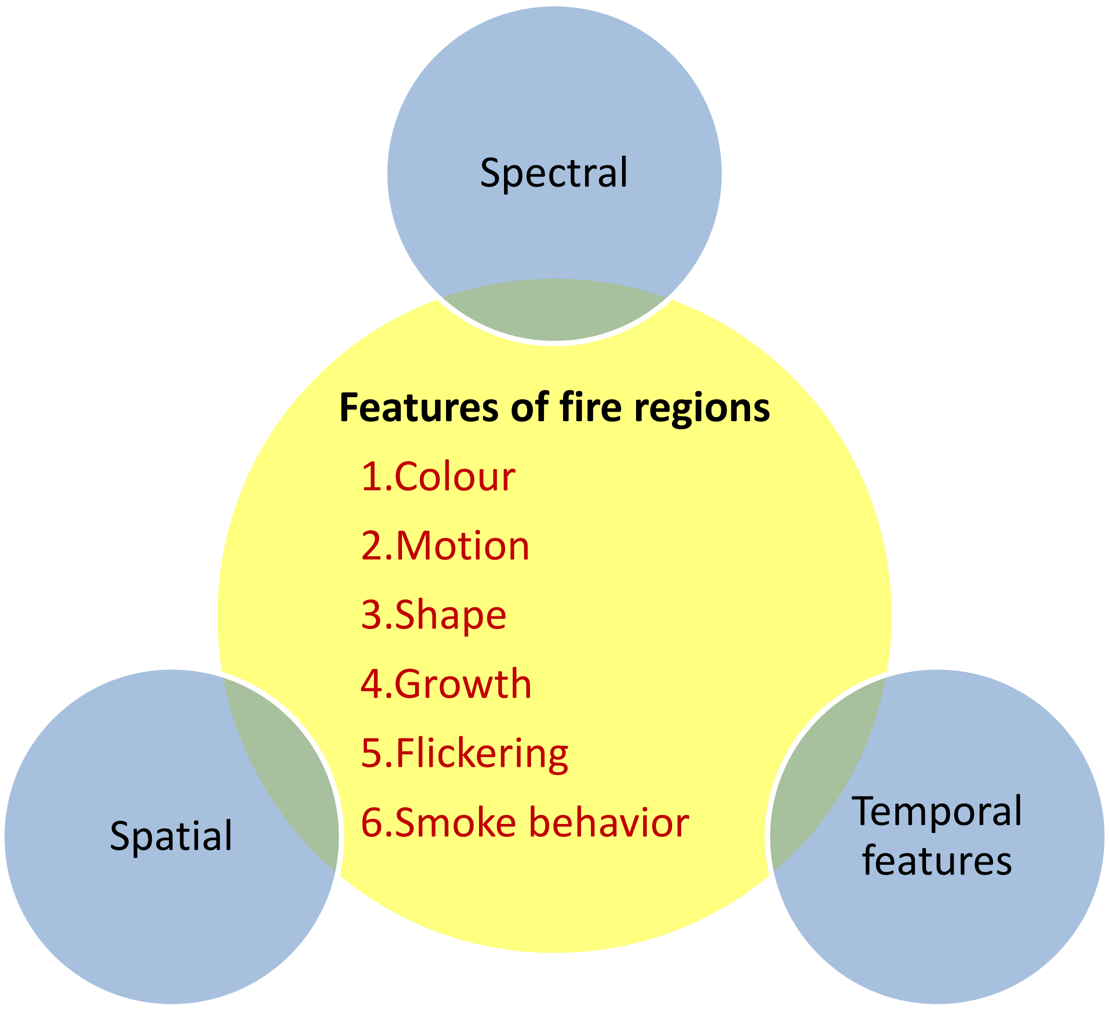


Figure 1.2 Relationship of features of fire regions and spectral, spatial, temporal features

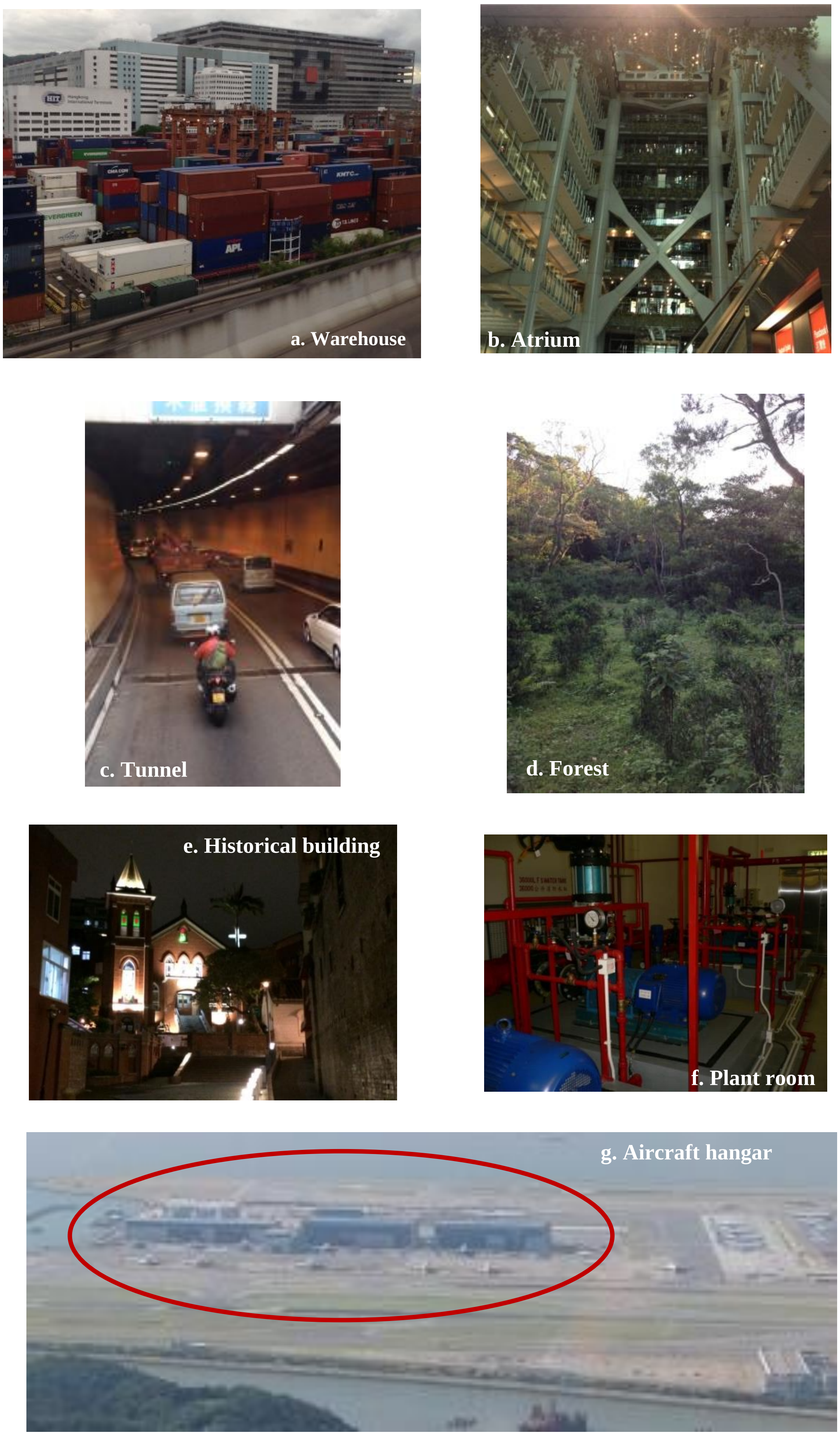


Figure 1.3 Examples of different environments and premises

a. Smoke detectors

b. Heat detectors

c. Flame detectors

d. Gas Sensors

Figure 1.4 Traditional fire detectors

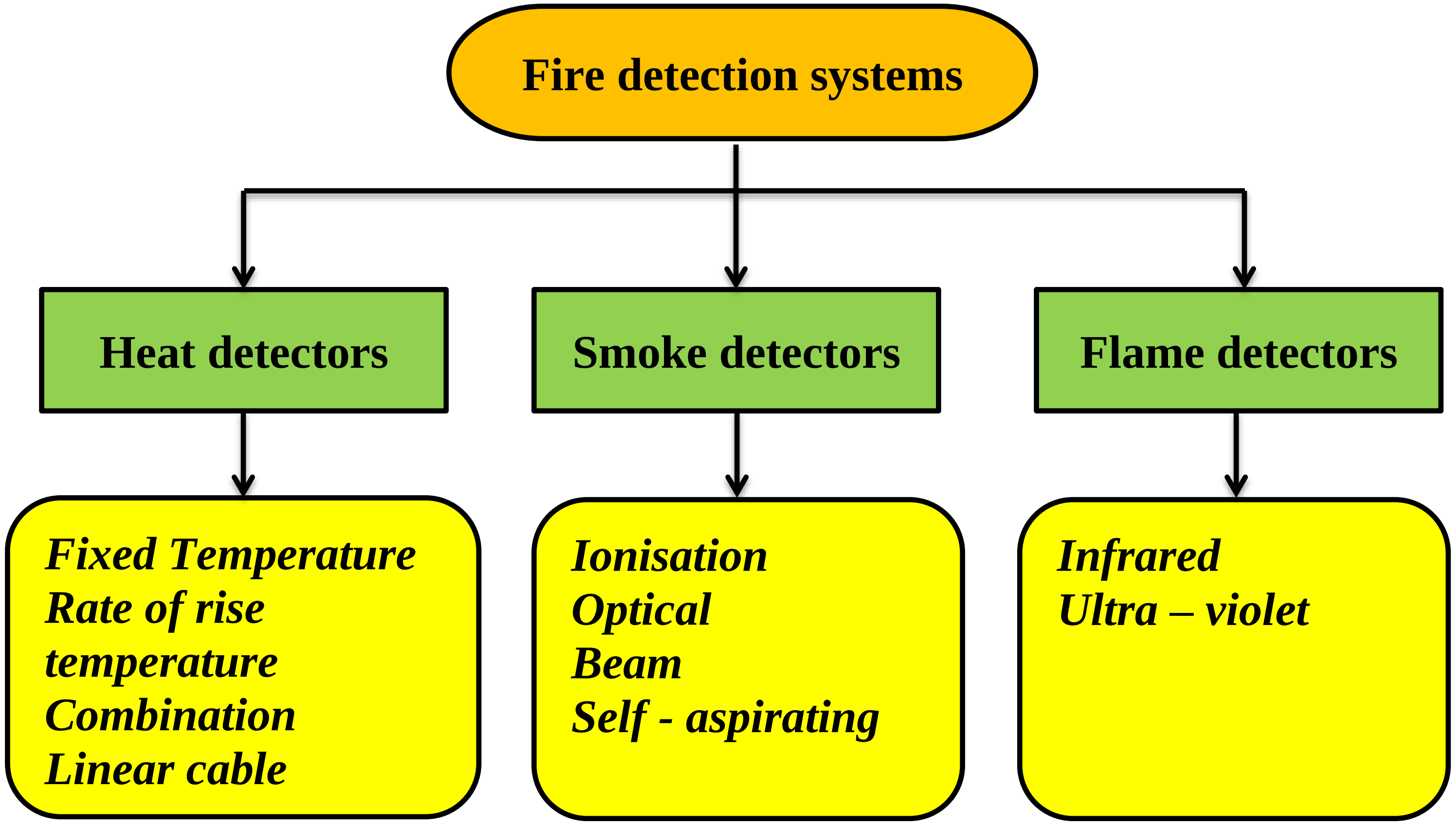


Figure 1.5 Fire detectors approved by the Fire Services Department (Codes of Practice Minimum Fire Service Installations and Equipment and Inspection, Testing and Maintenance of Installtions and Equipment, 1998)

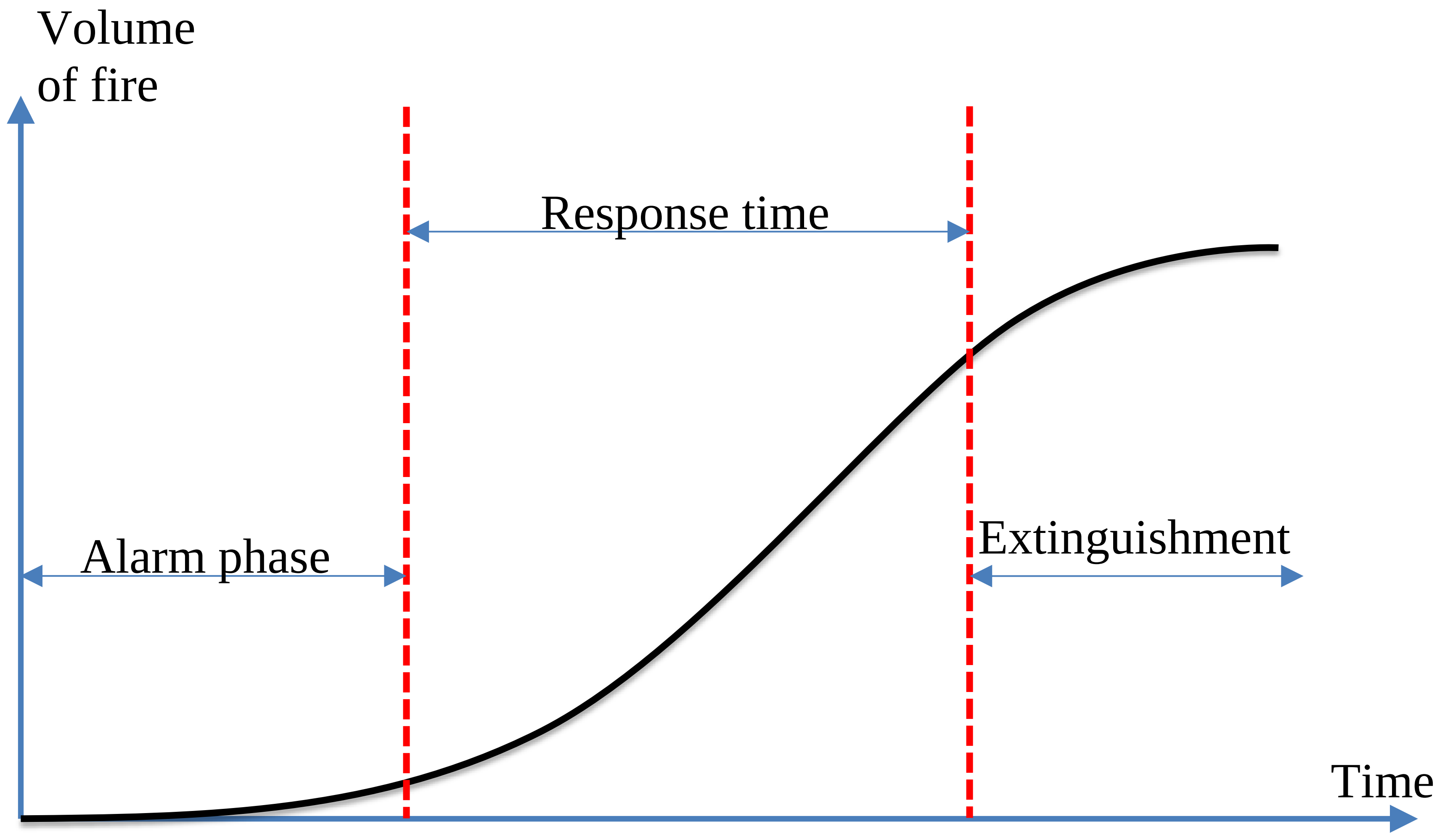


Figure 1.6 Time relationship to fire development and fire control (Chapter 10 Fire Detection Systems, 1993)

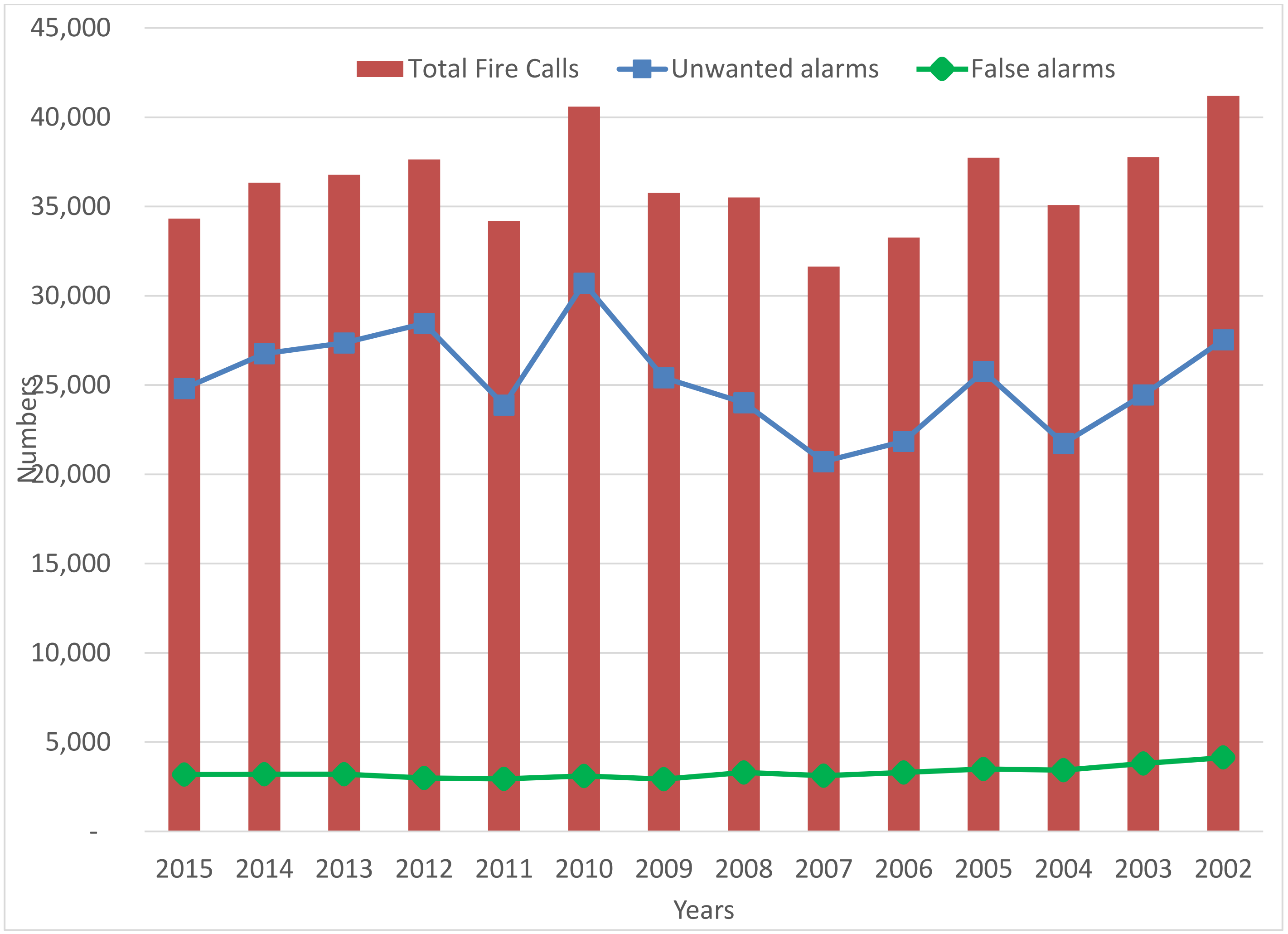


Figure 1.7 Total fire calls, unwanted alarms, and false alarms (Hong Kong Fire Services Department, 2002 - 2015)

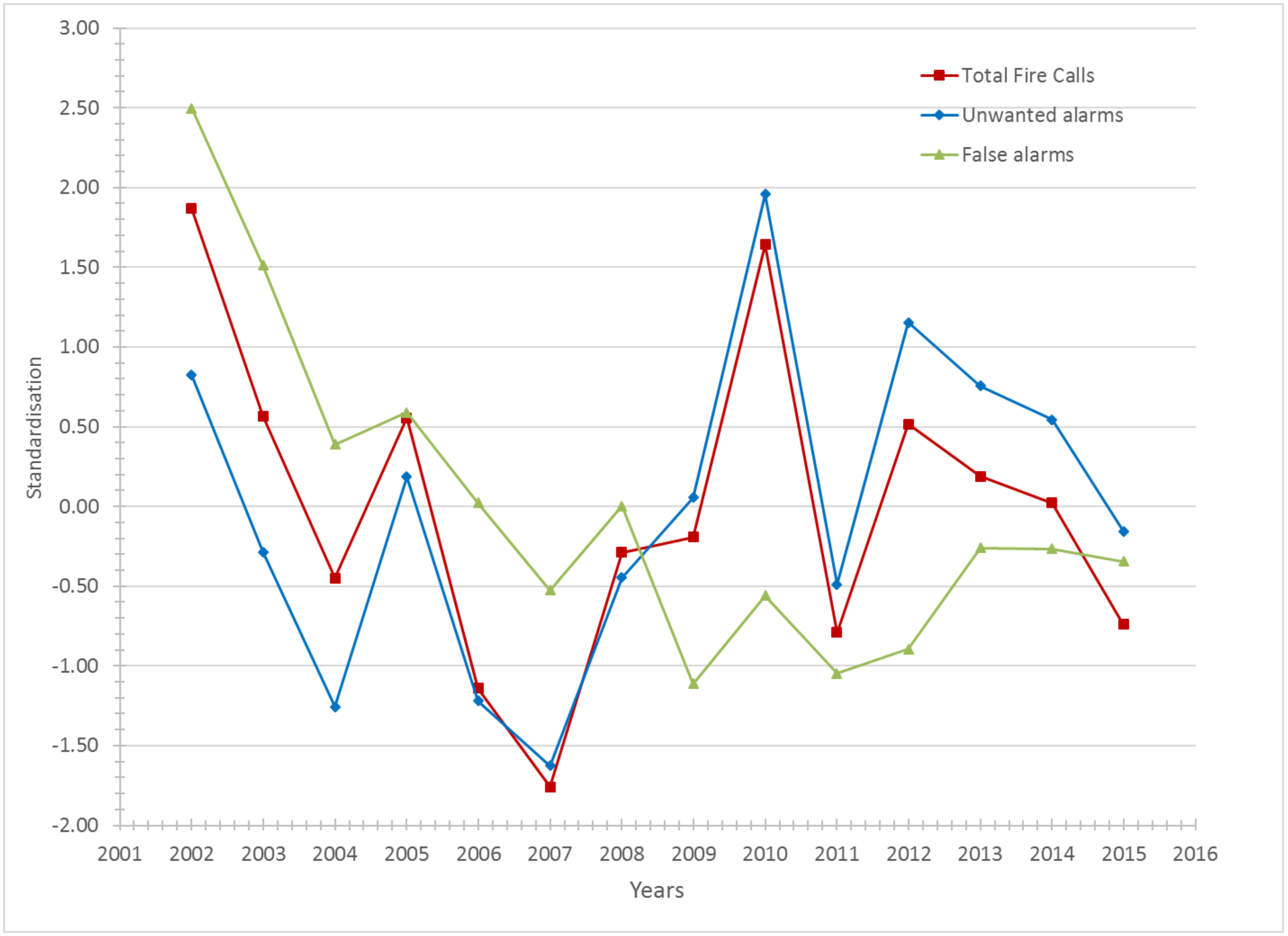


Figure 1.8 Standardisation of the numbers of total fire calls, unwanted alarms, and false alarms

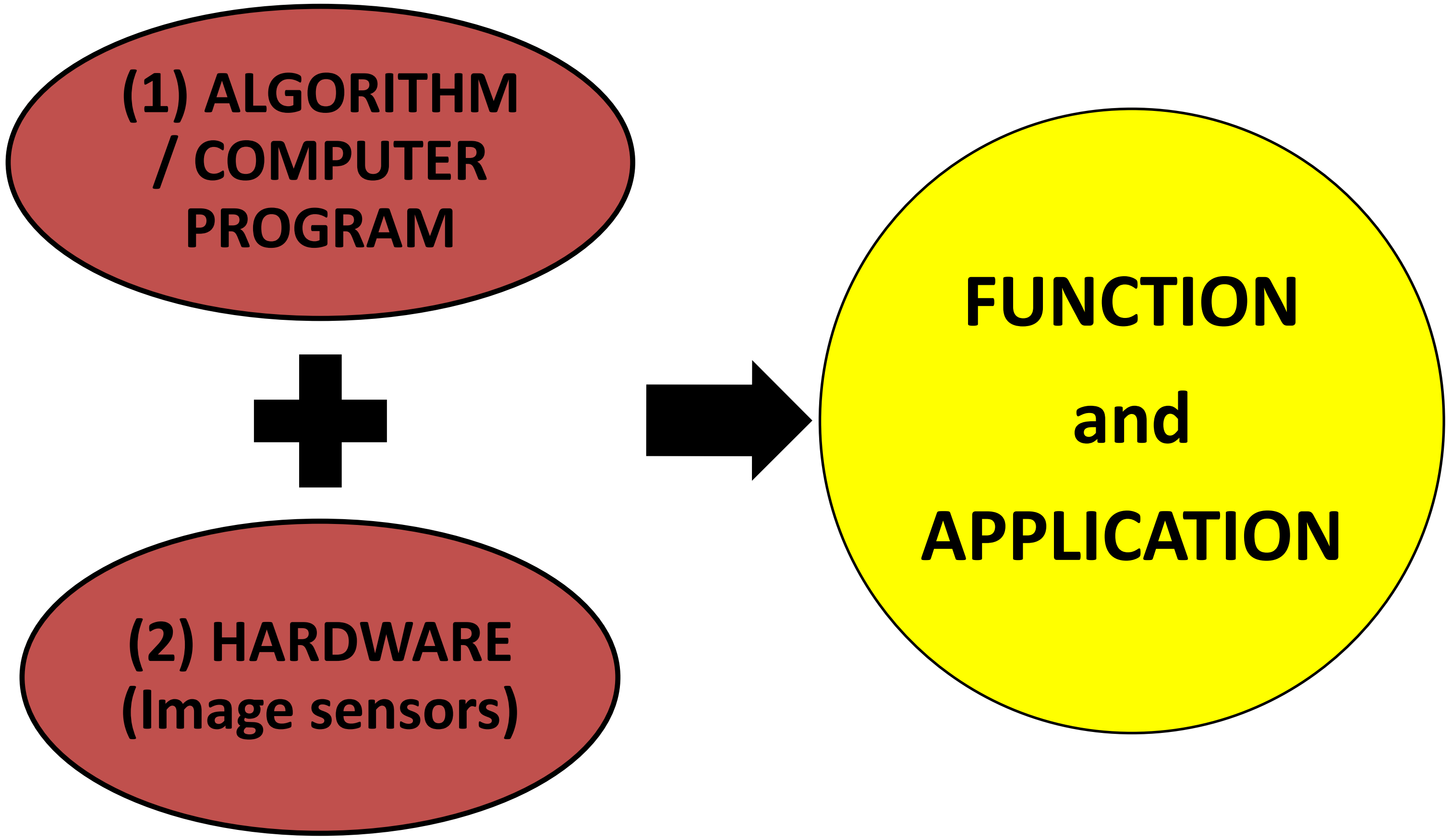
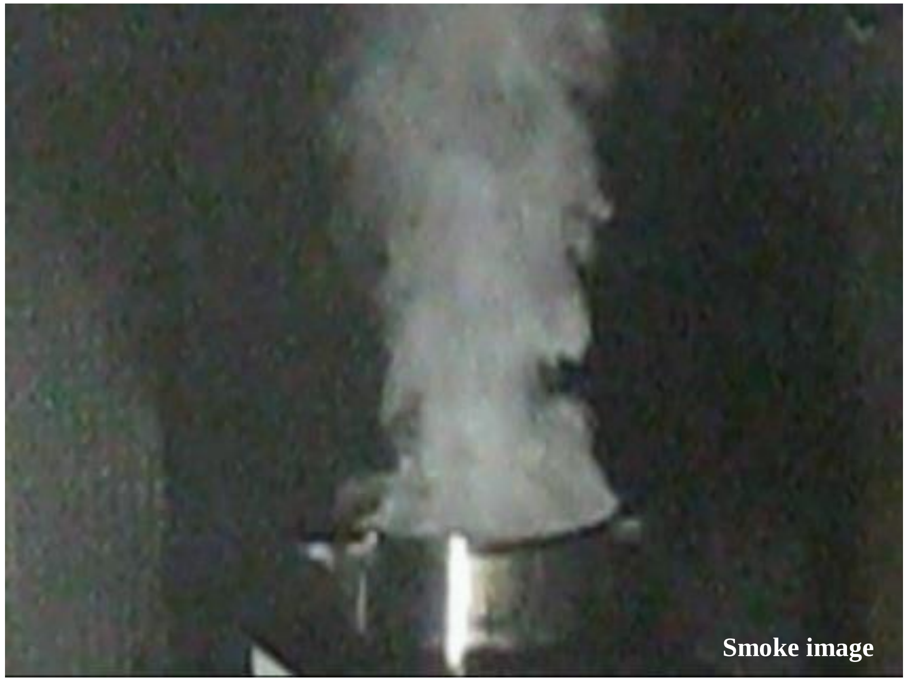

(1) ALGORITHM / COMPUTER PROGRAM
(2) HARDWARE (Image sensors)
FUNCTION and APPLICATION

Figure 1.9 Conceptual diagram of video fire detection



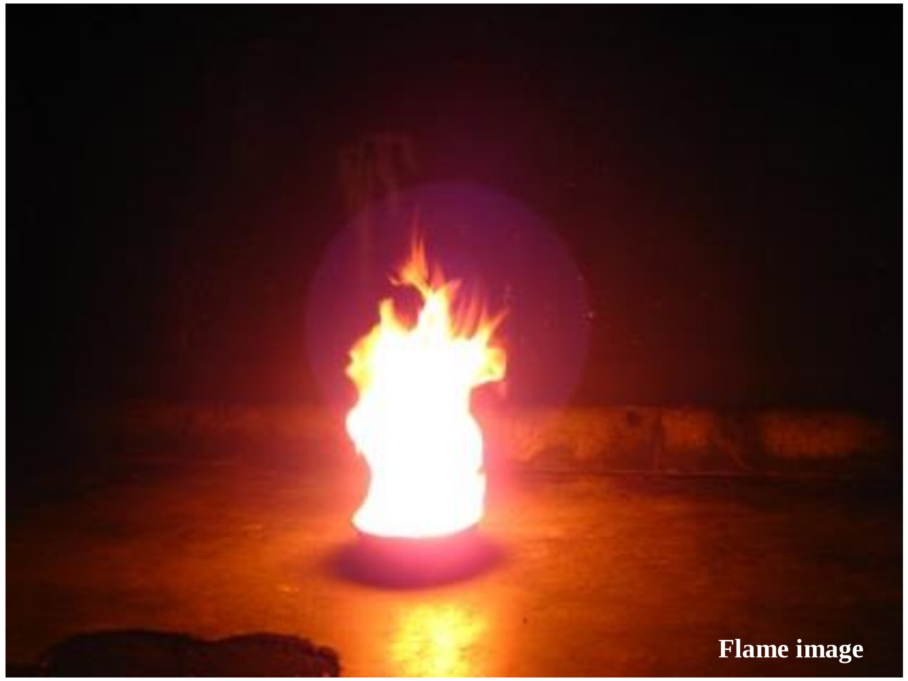


Figure 1.10 Typical fire images

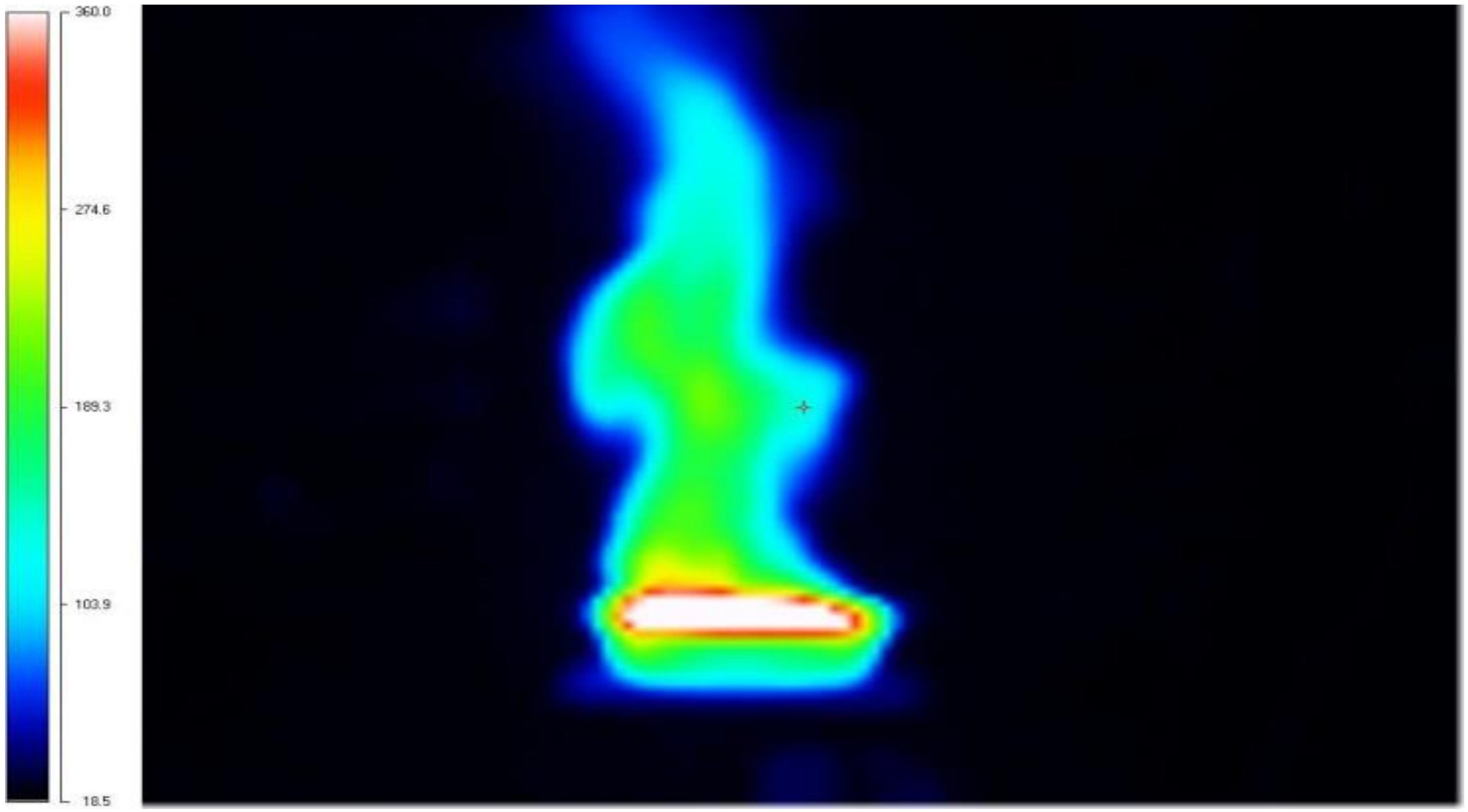


Figure 1.11 Typical thermal image

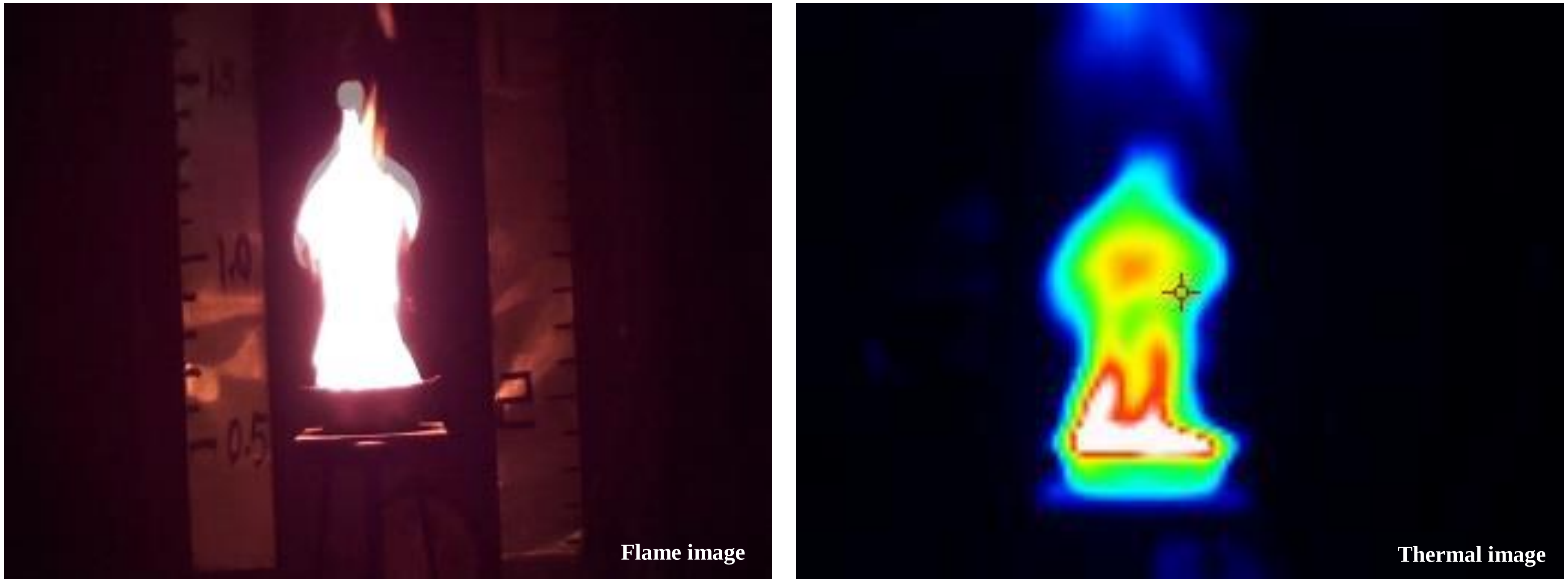


Figure 1.12 Flame shapes captured simultaneously

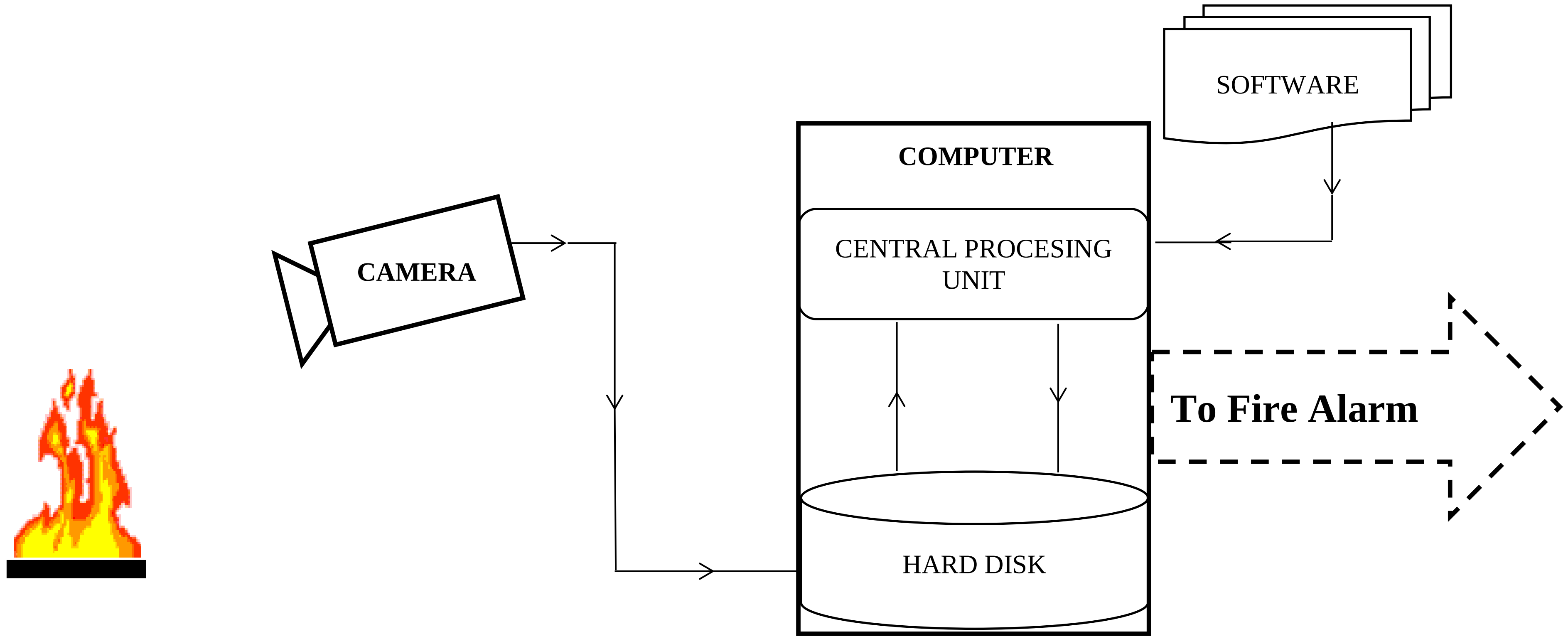
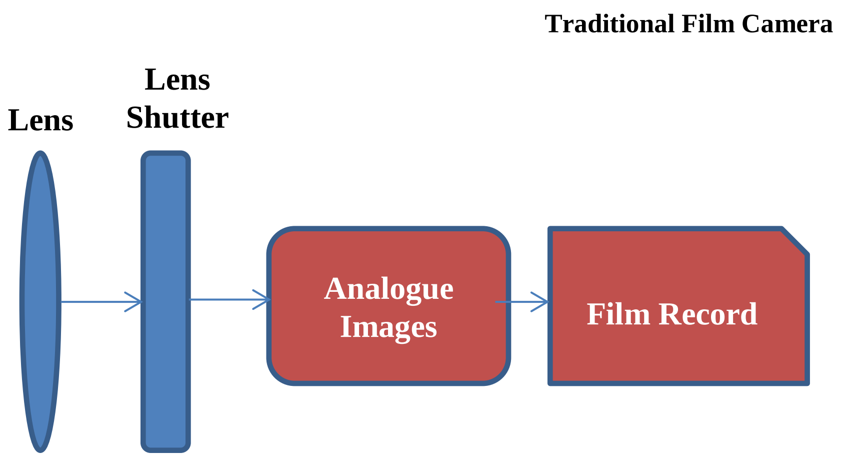


Figure 1.13 Operation framework for video fire detection



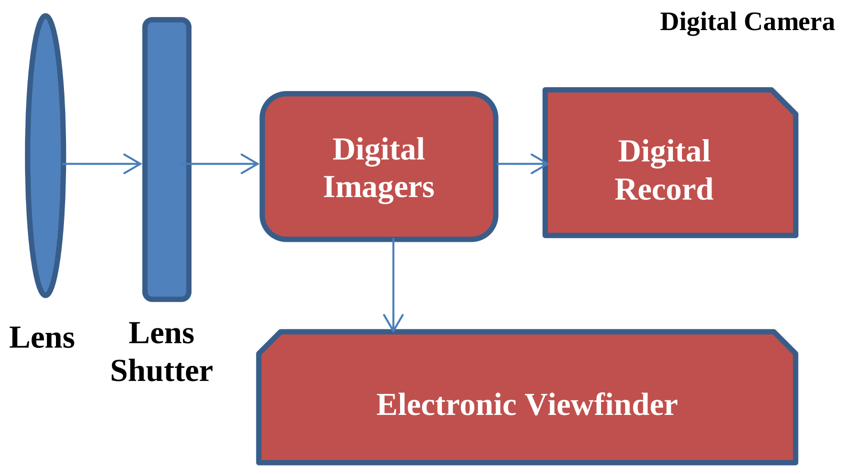
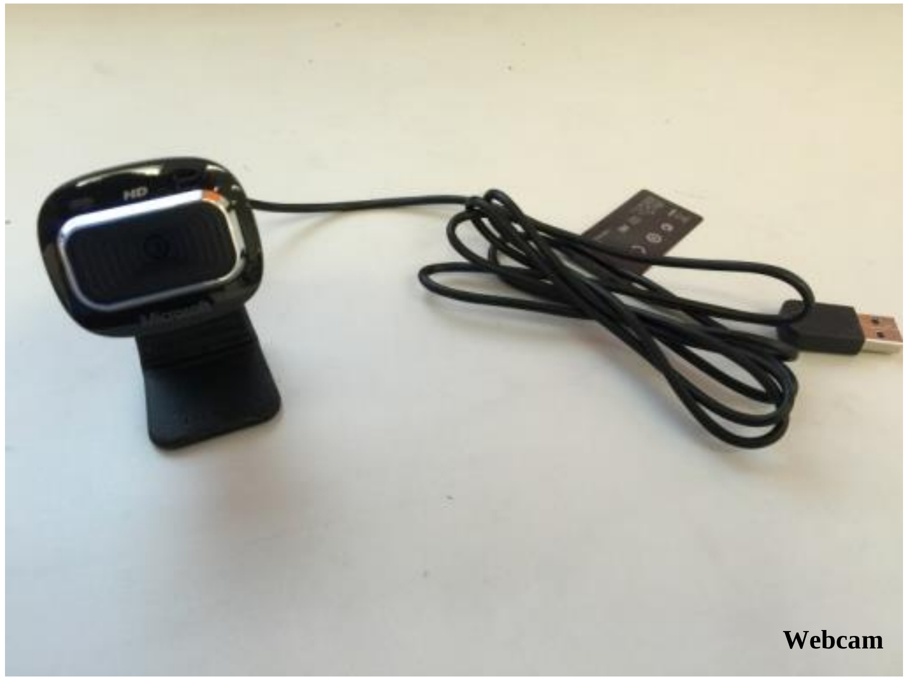


Figure 1.14 A traditional film camera compared with a digital camera



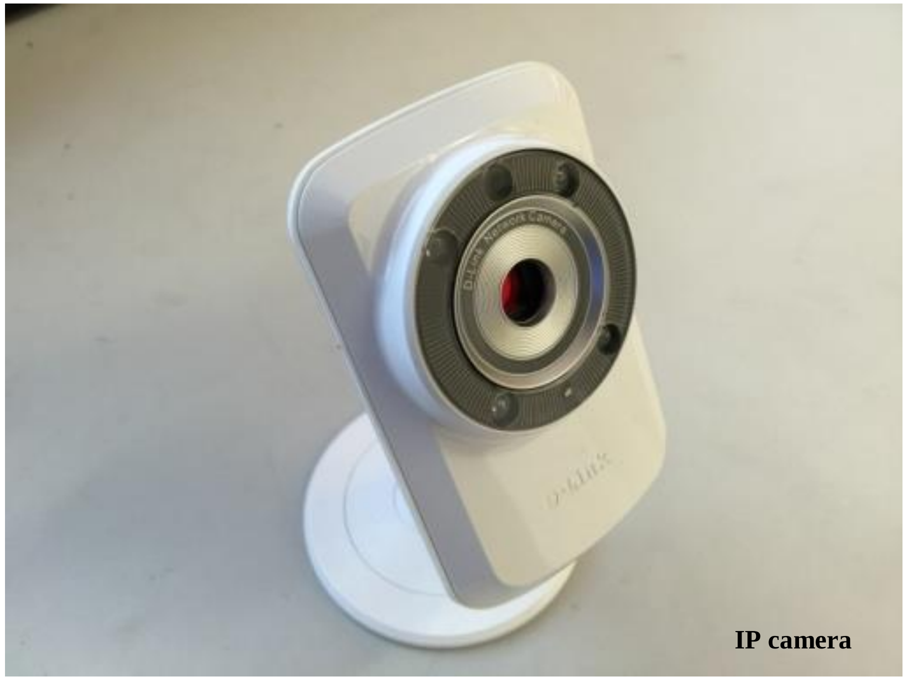


Figure 1.15 A traditional webcam compared with an IP camera

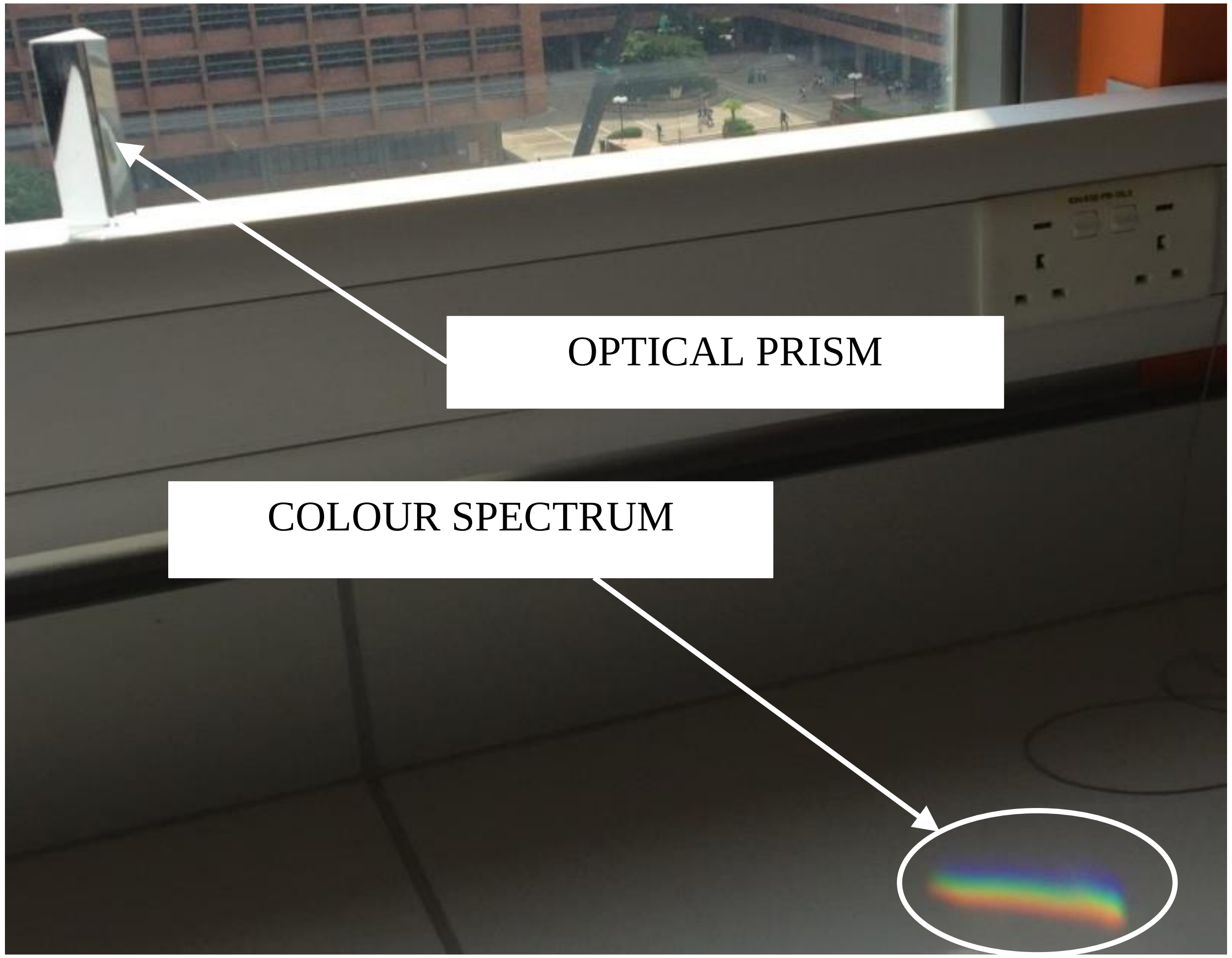


Figure 1.16 Natural phenomenon (colour spectrum)

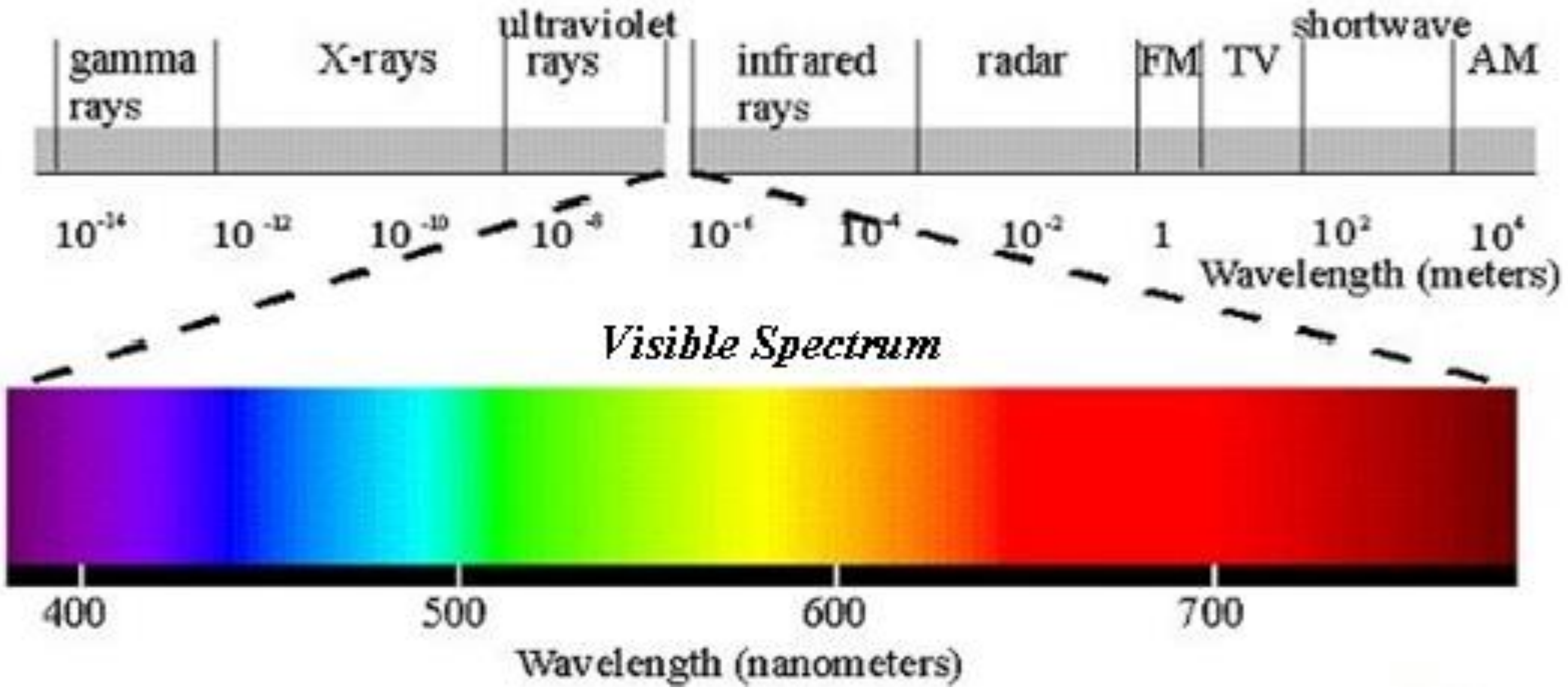


Figure 1.17 The visible spectrum and the electromagnetic spectrum (Beer Color Laboratories, n.d.)

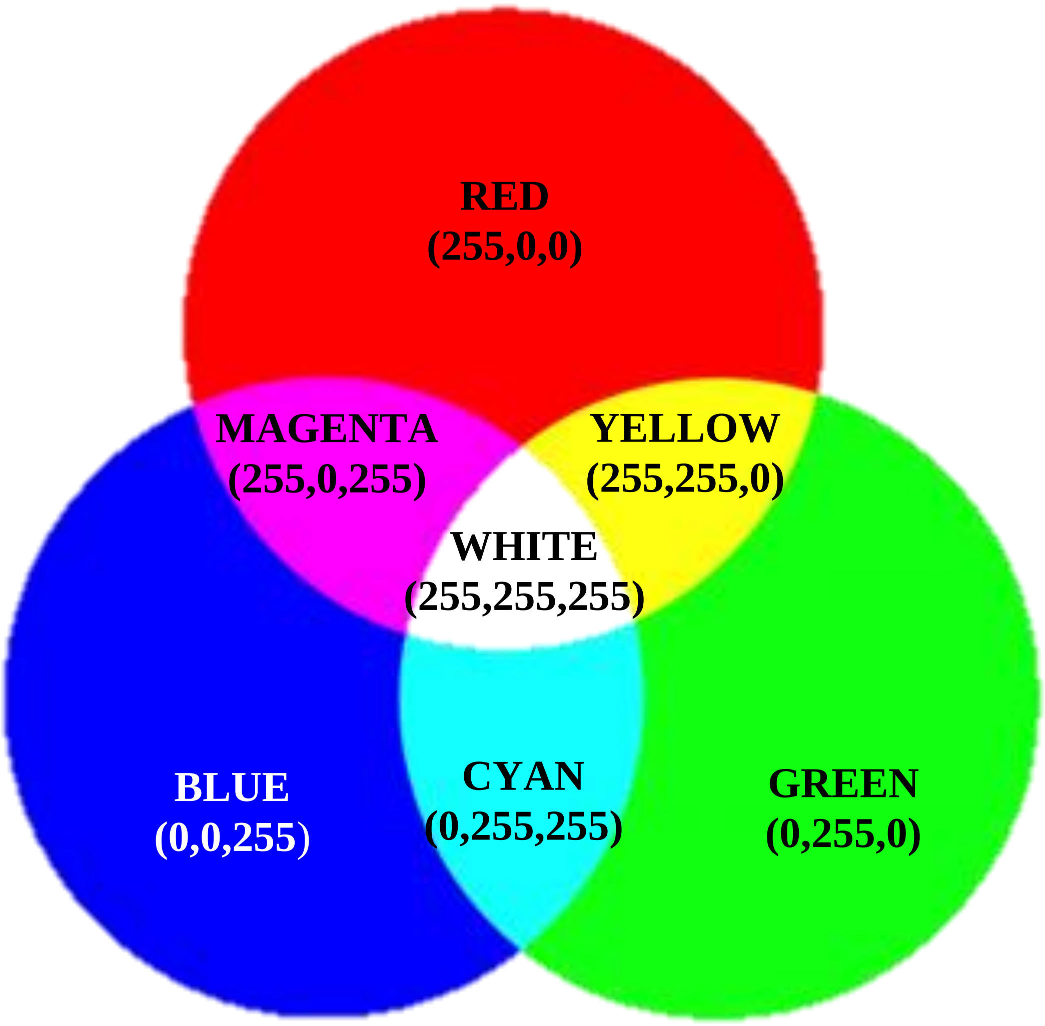
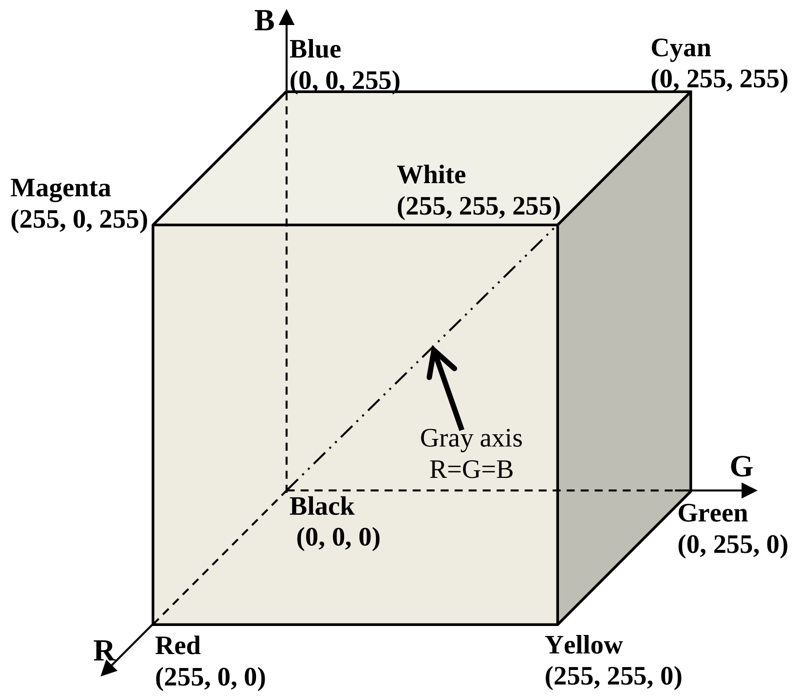
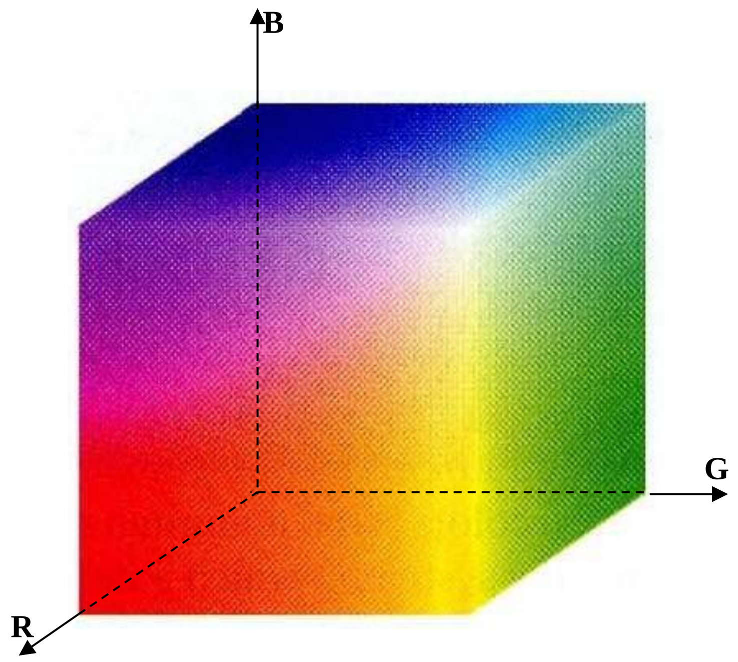


Figure 1.18 Mixture of colour (additive primaries)



(a) Schematic of colour cube



(b) Colour cube

Figure 1.19 Greyscale model: (a) schematic of colour cube, (b) colour cube

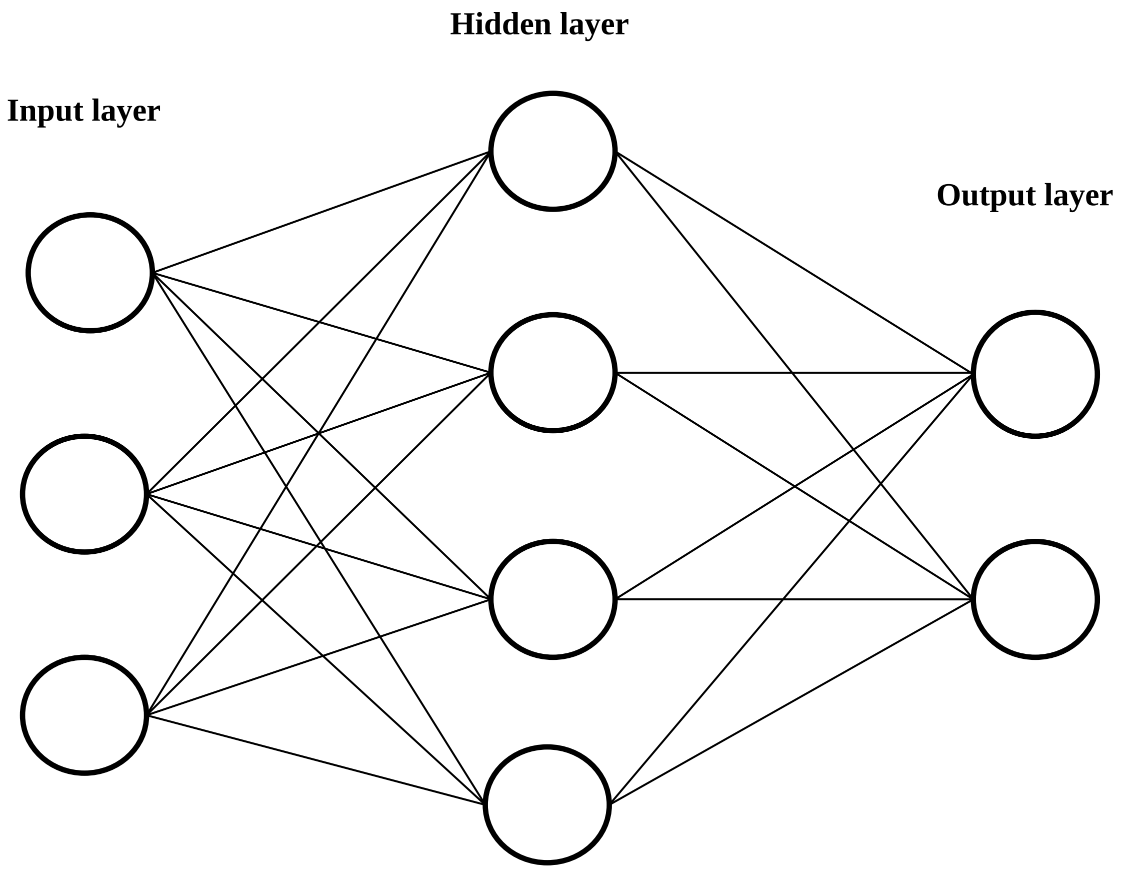


Figure 1.20 Typical structure of neural networks

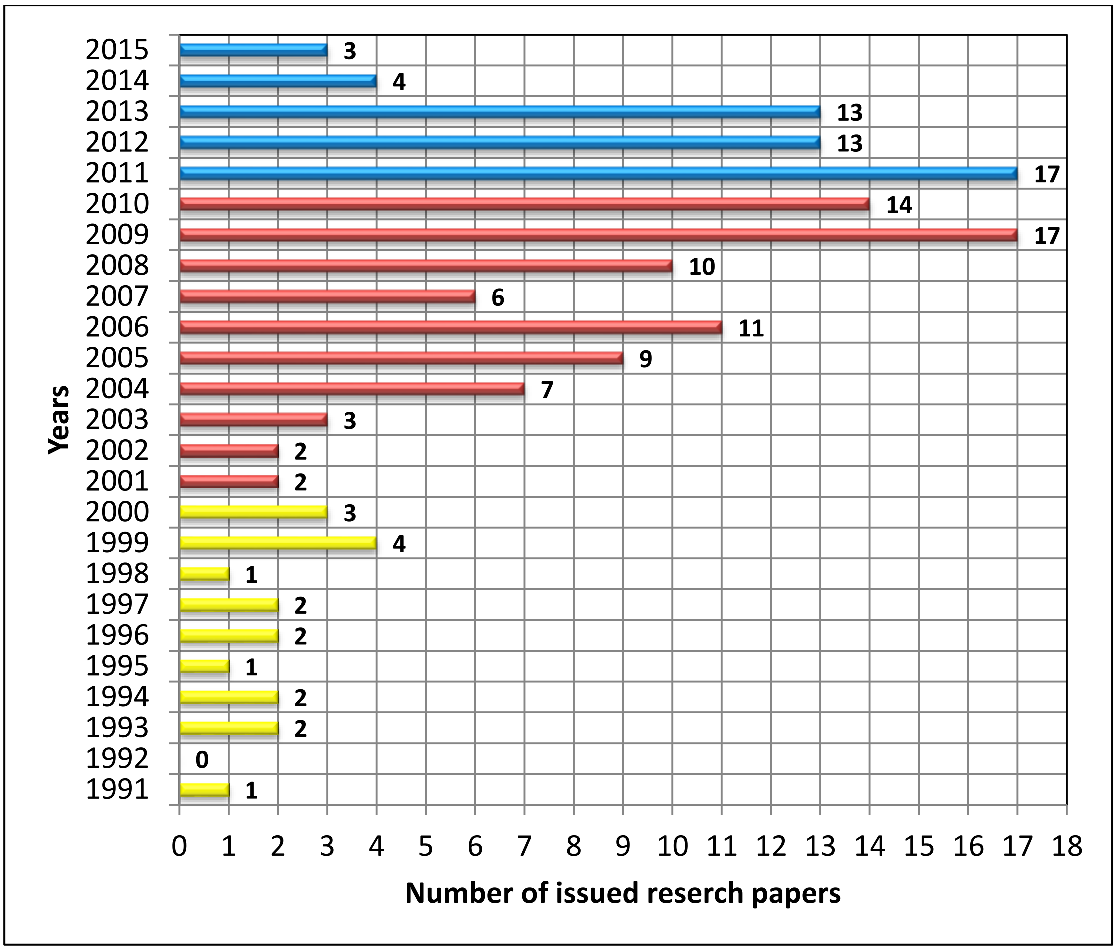


Figure 2.1 Quantity of research papers have gone through

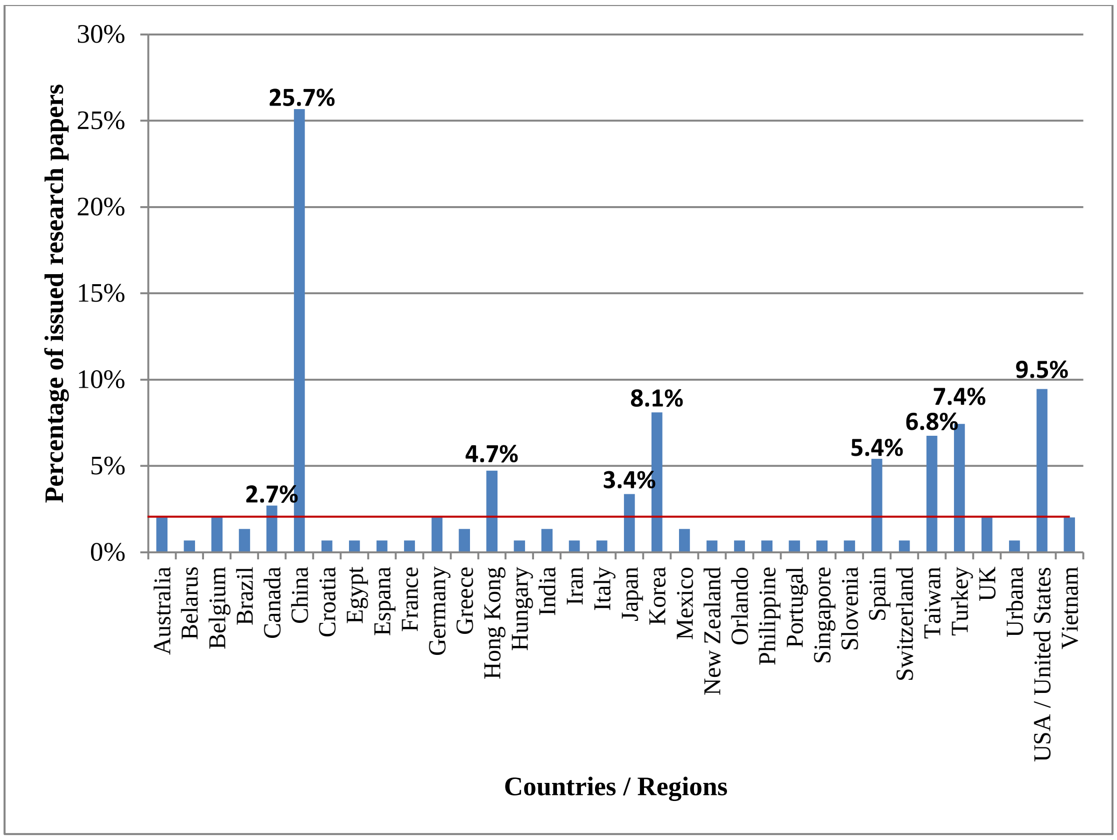


Figure 2.2 Quantity of research papers in different countries

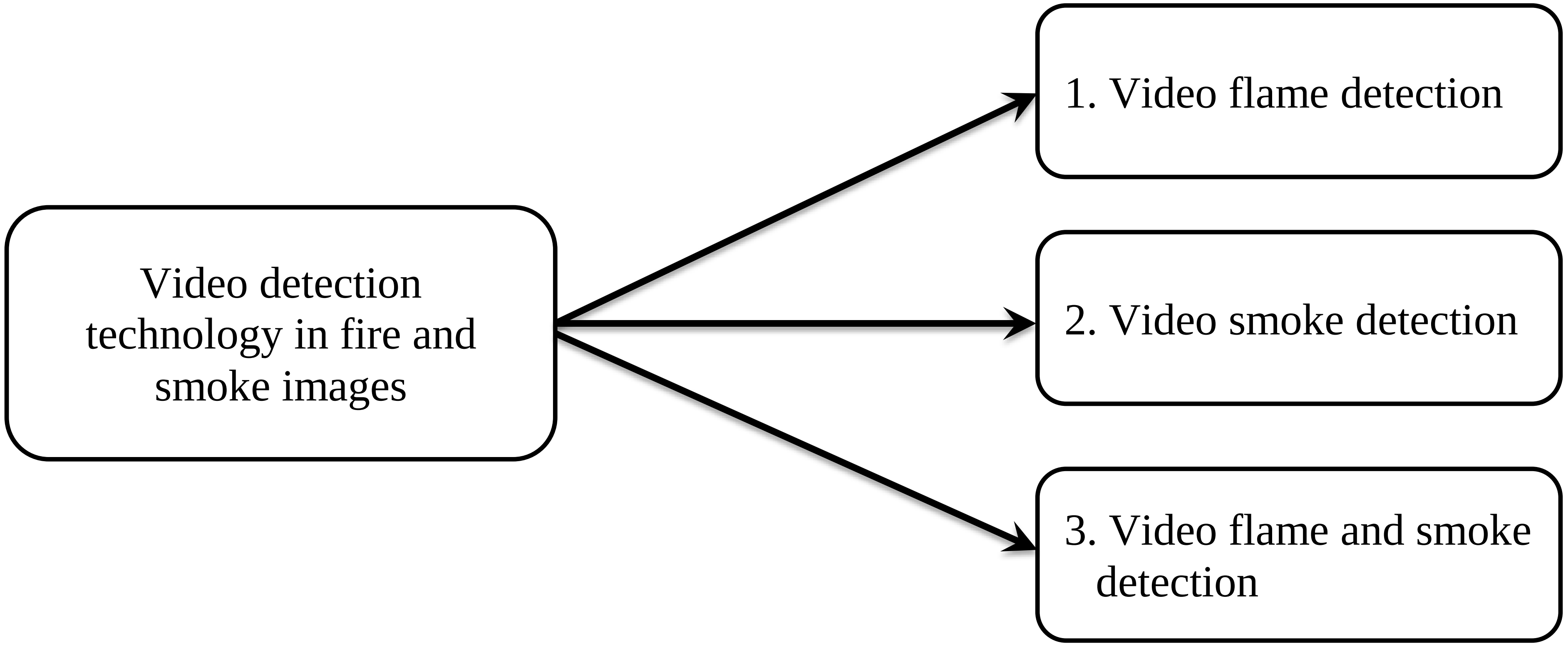

Video detection technology in fire and smoke images
1. Video flame detection
2. Video smoke detection
3. Video flame and smoke detection

Figure 2.3 Classification video fire detection technology

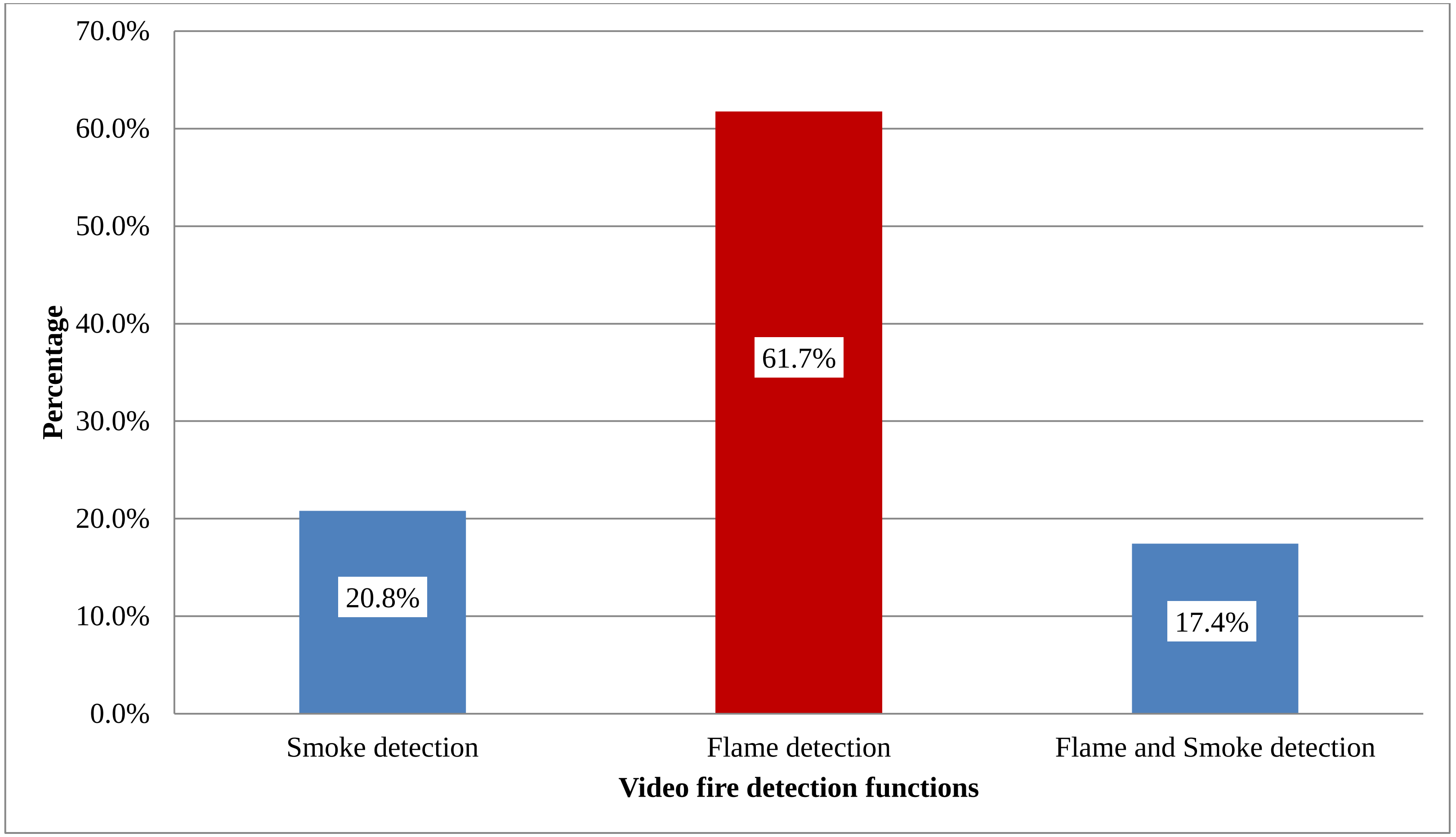


Figure 2.4 Statistical results in video fire detection functions

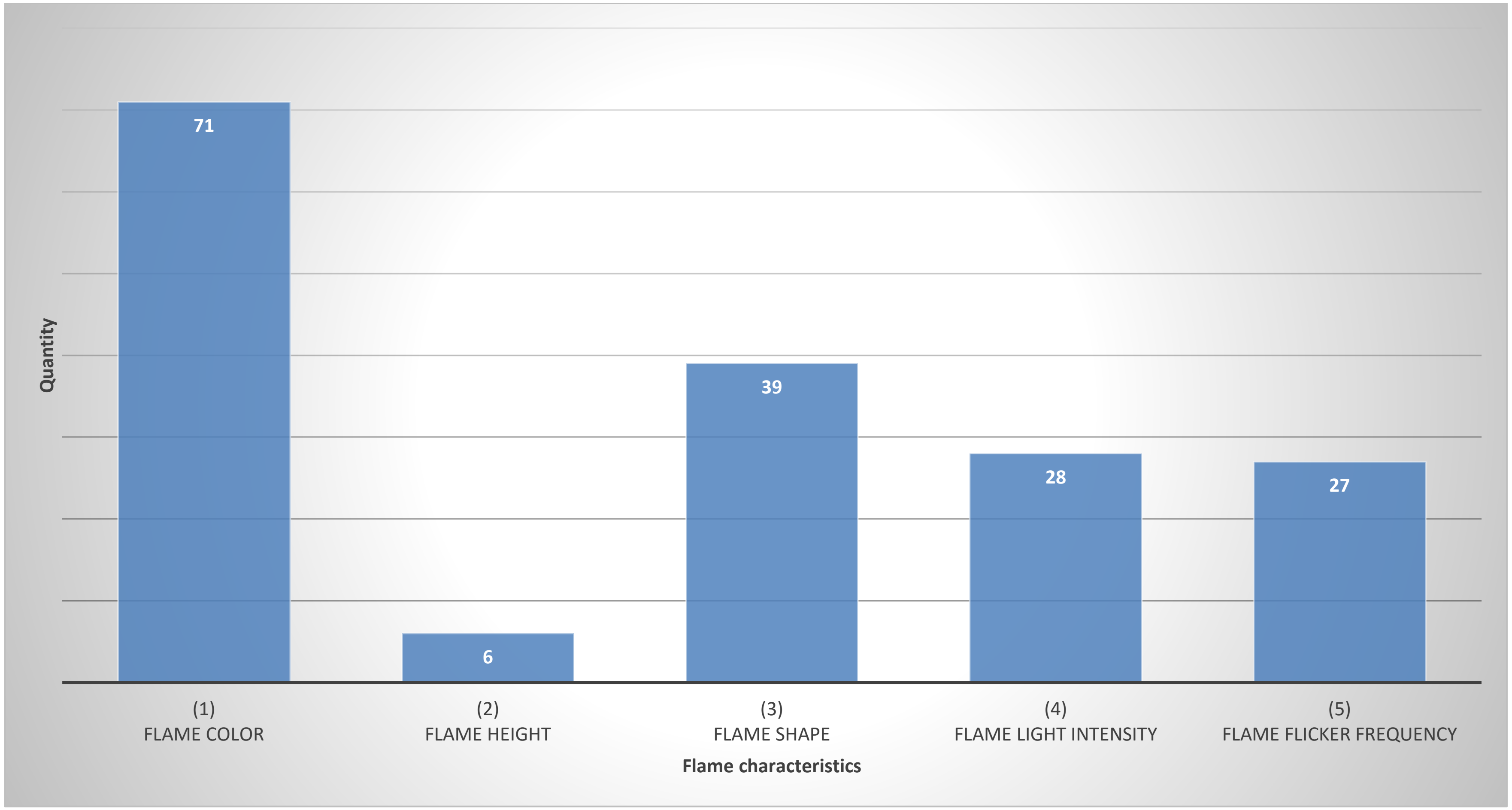
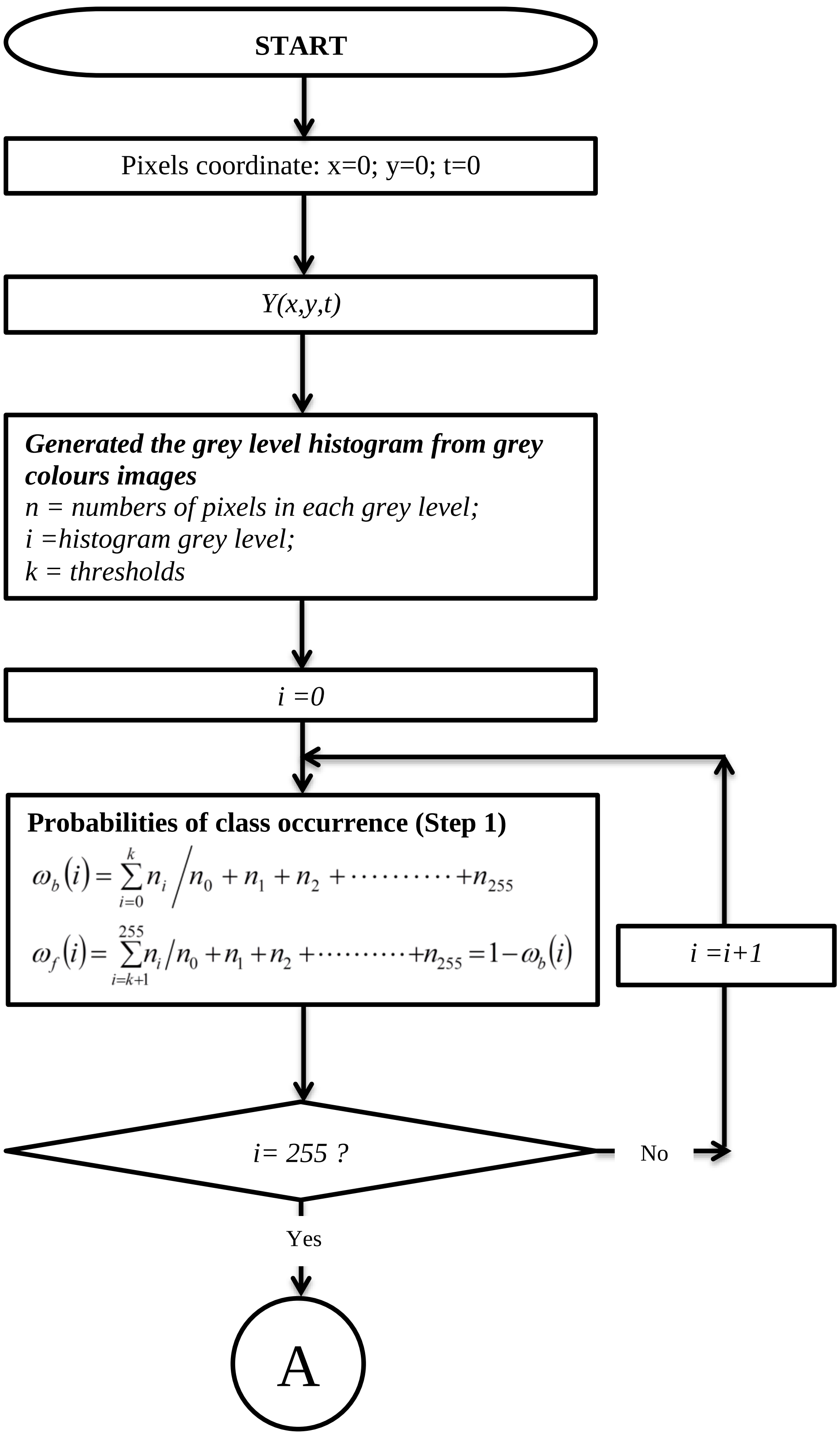
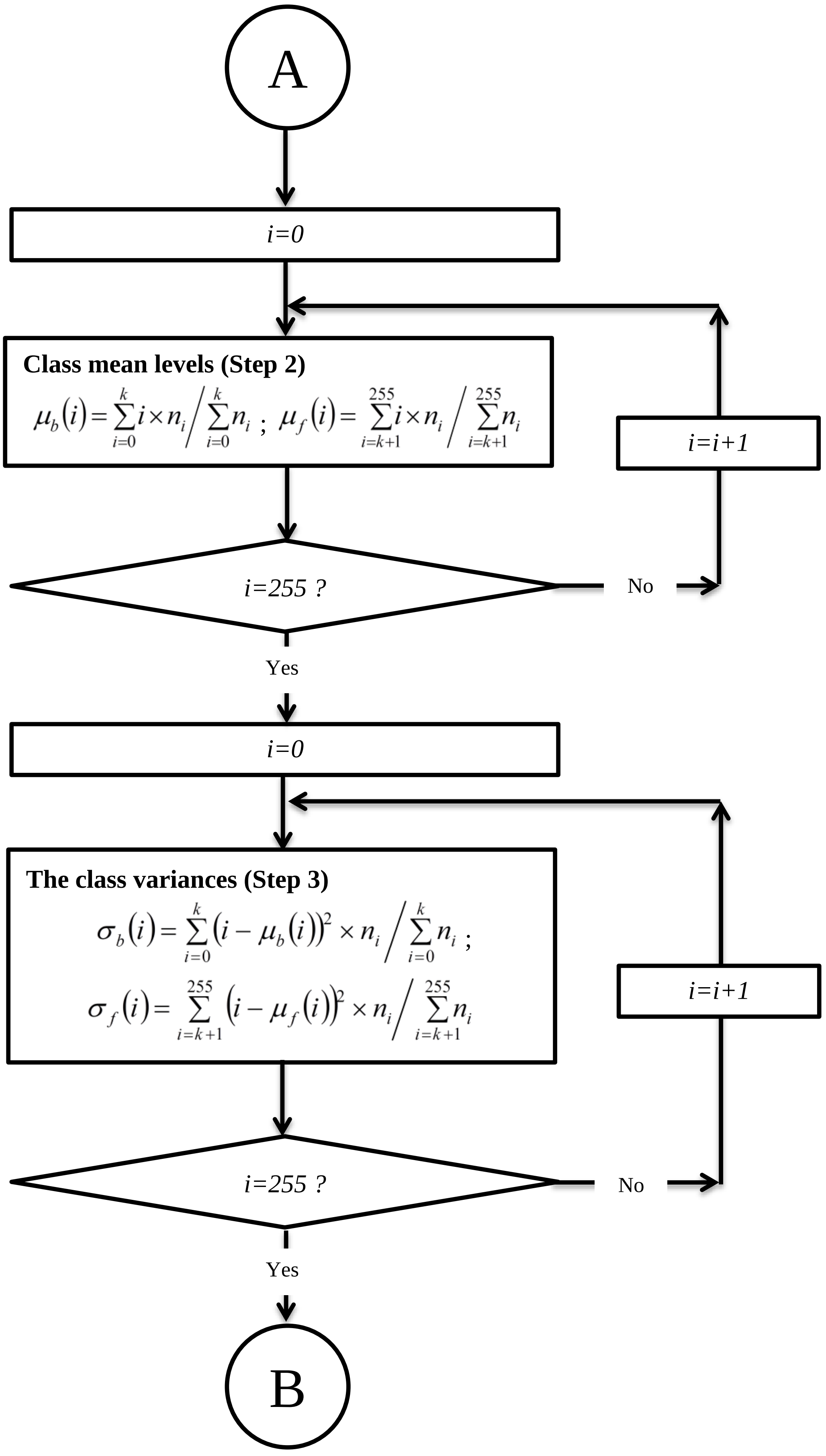


Figure 2.5 Statistical results regarding the quantity of flame characteristics


START
Pixels coordinate: x=0; y=0; t=0
Y(x,y,t)
Generated the grey level histogram from grey colours images
n = numbers of pixels in each grey level;
i =histogram grey level;
k = thresholds
i =0
Probabilities of class occurrence (Step 1)
$\omega_b(i) = \sum_{i=0}^{k} n_i \Big/ n_0 + n_1 + n_2 + \cdots\cdots\cdots\cdots + n_{255}$
$\omega_f(i) = \sum_{i=k+1}^{255} n_i / n_0 + n_1 + n_2 + \cdots\cdots\cdots\cdots + n_{255} = 1 - \omega_b(i)$
i =i+1
i= 255 ?
No
Yes
A



A
i=0
Class mean levels (Step 2)
$\mu_b(i)=\sum_{i=0}^{k} i\times n_i \Big/ \sum_{i=0}^{k} n_i$; $\mu_f(i)=\sum_{i=k+1}^{255} i\times n_i \Big/ \sum_{i=k+1}^{255} n_i$
i=i+1
i=255 ?
No
Yes
i=0
The class variances (Step 3)
$\sigma_b(i)=\sum_{i=0}^{k}(i-\mu_b(i))^2\times n_i \Big/ \sum_{i=0}^{k} n_i$;
$\sigma_f(i)=\sum_{i=k+1}^{255}(i-\mu_f(i))^2\times n_i \Big/ \sum_{i=k+1}^{255} n_i$
i=i+1
i=255 ?
No
Yes
B

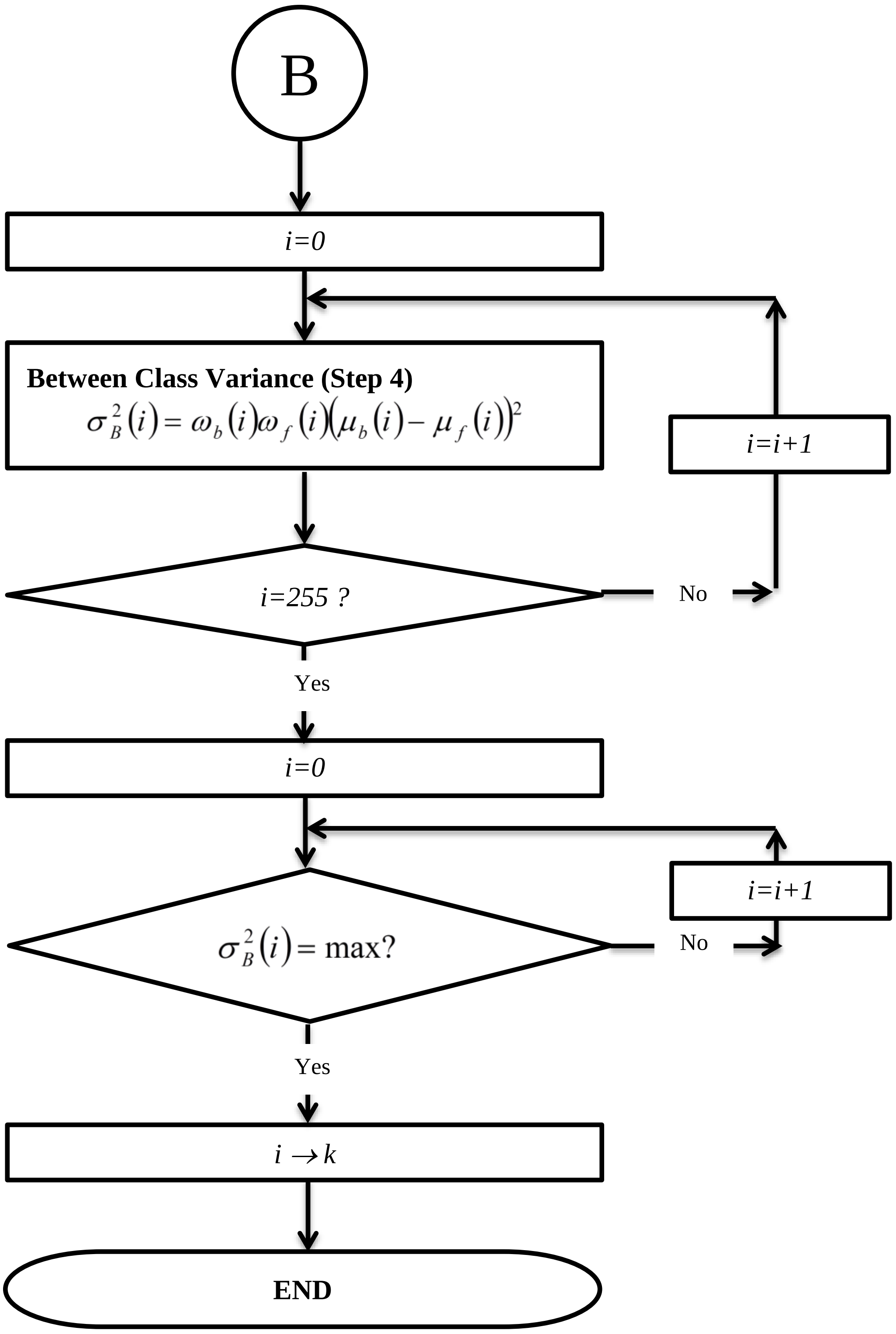

B
i=0
Between Class Variance (Step 4)
$\sigma_B^2(i) = \omega_b(i)\omega_f(i)(\mu_b(i) - \mu_f(i))^2$
i=i+1
i=255 ?
No
Yes
i=0
i=i+1
$\sigma_B^2(i) = \max?$
No
Yes
$i \rightarrow k$
END

Figure 3.1 Flow diagram of Otsu threshold method

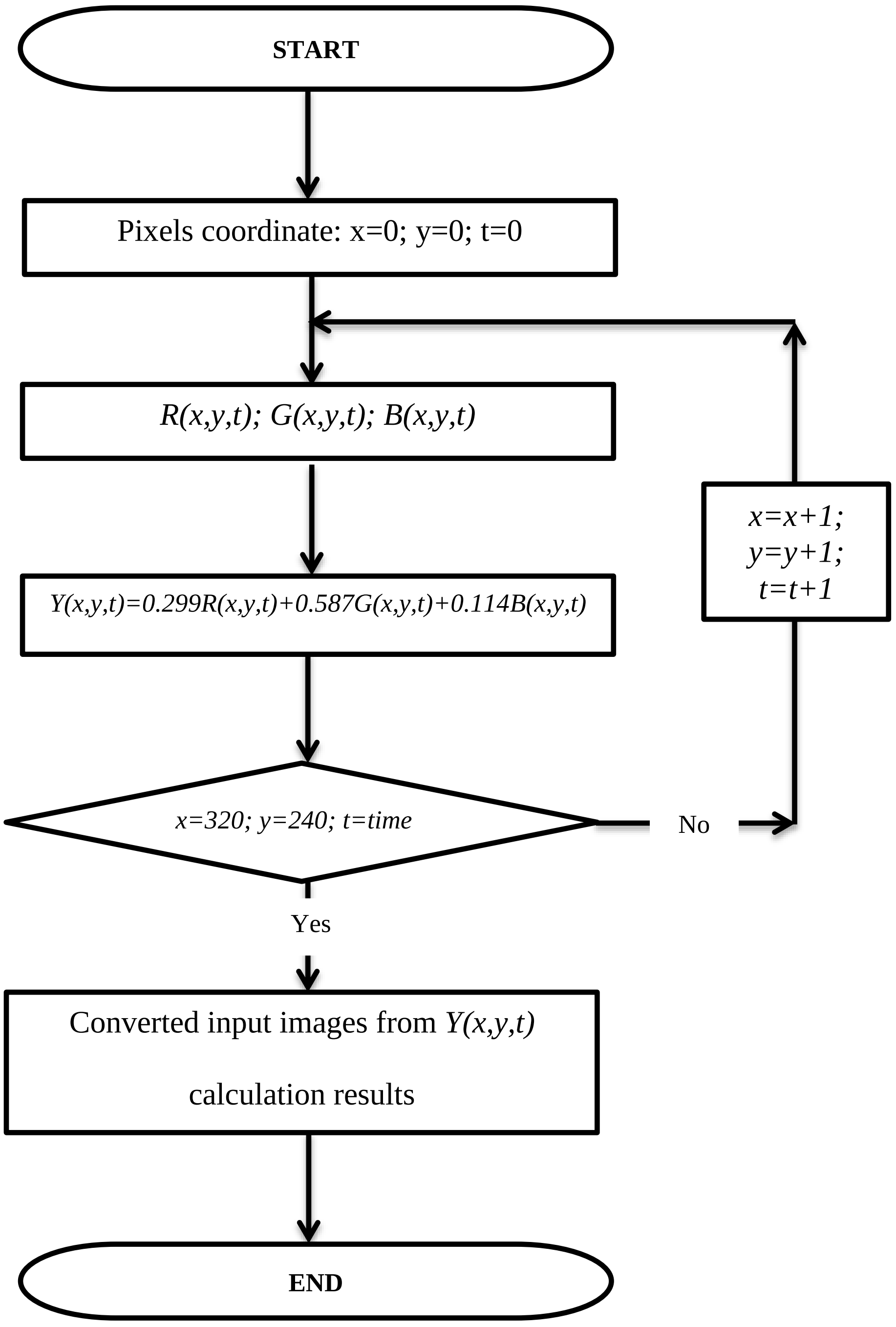

START
Pixels coordinate: x=0; y=0; t=0
R(x,y,t); G(x,y,t); B(x,y,t)
Y(x,y,t)=0.299R(x,y,t)+0.587G(x,y,t)+0.114B(x,y,t)
x=320; y=240; t=time
No
x=x+1; y=y+1; t=t+1
Yes
Converted input images from Y(x,y,t) calculation results
END

Figure 3.2 Flow diagram of the grey colours images conversion method

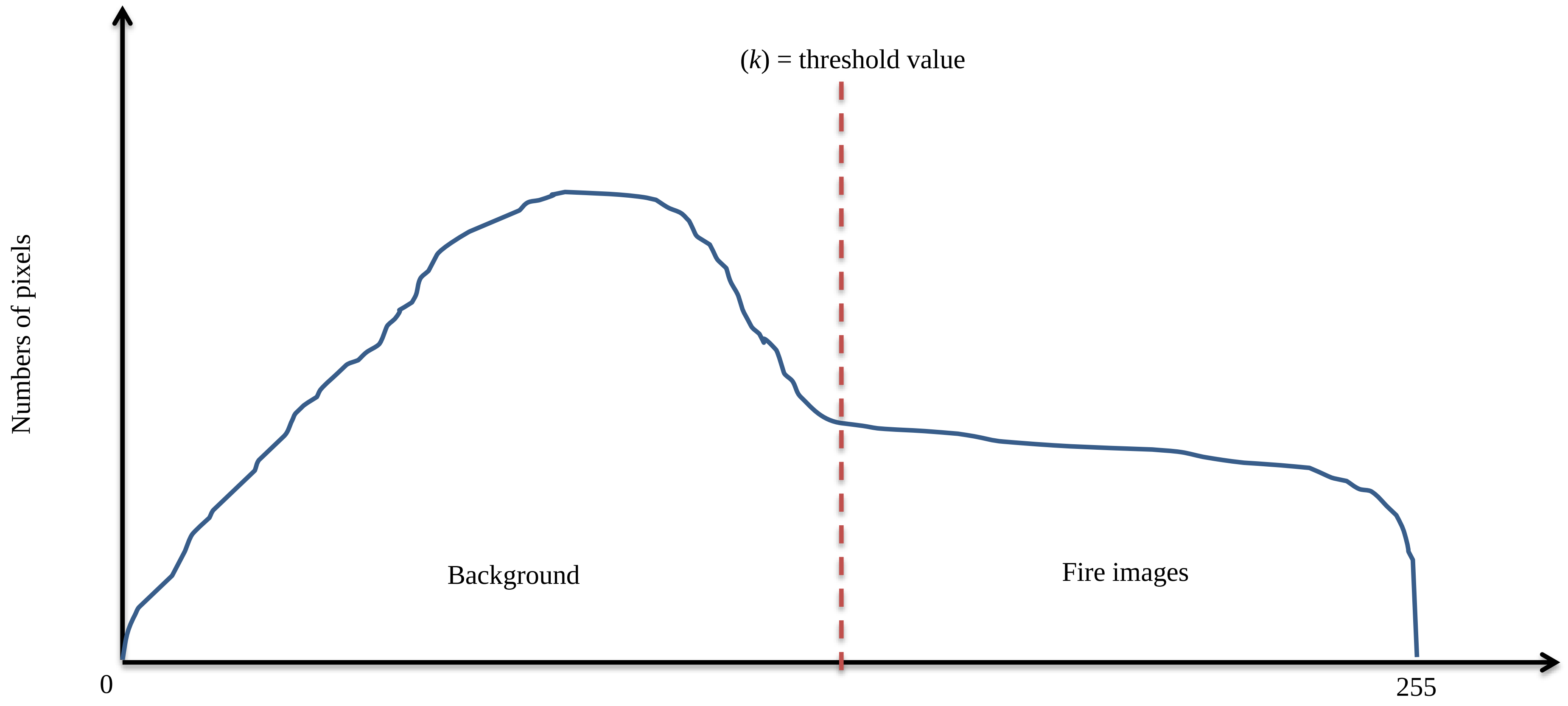


Figure 3.3 Histogram analysis of Otsu method

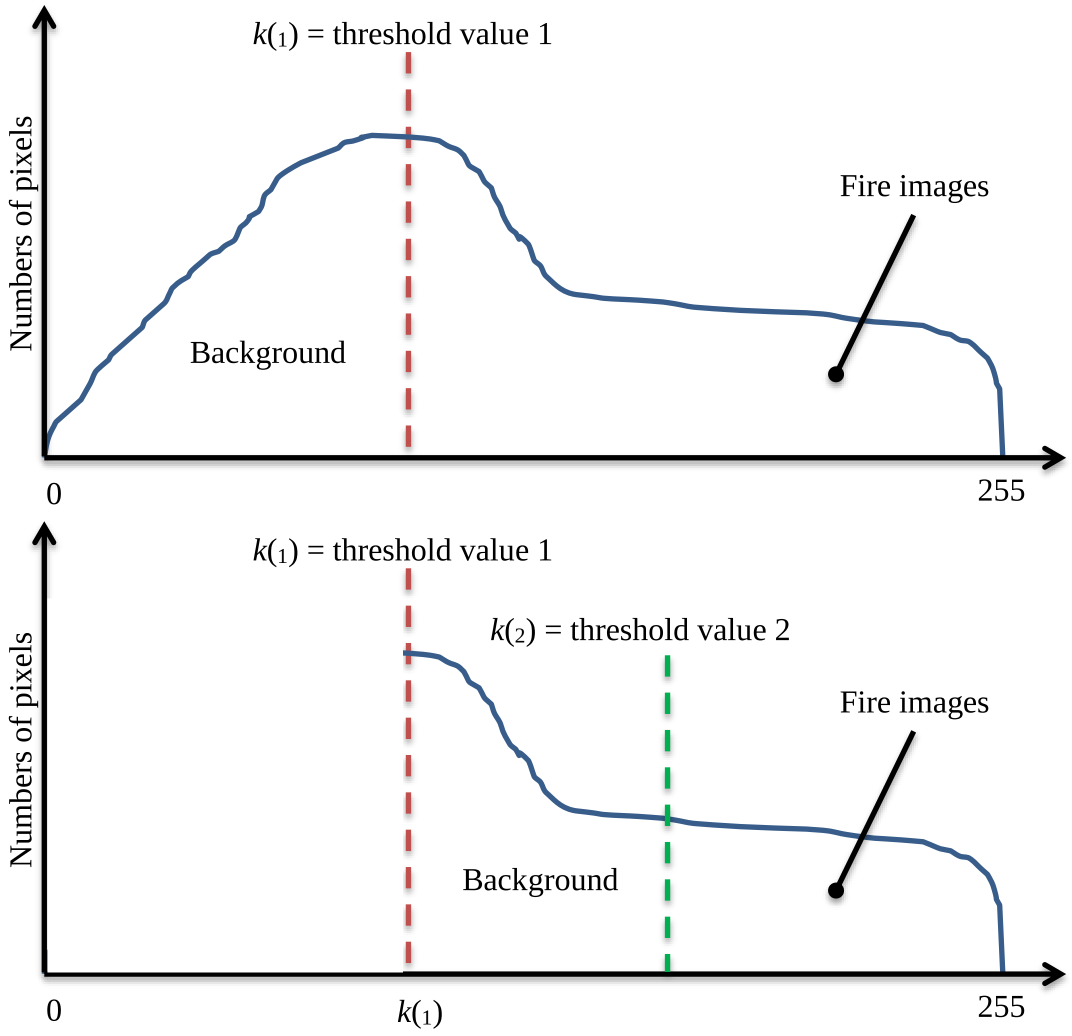
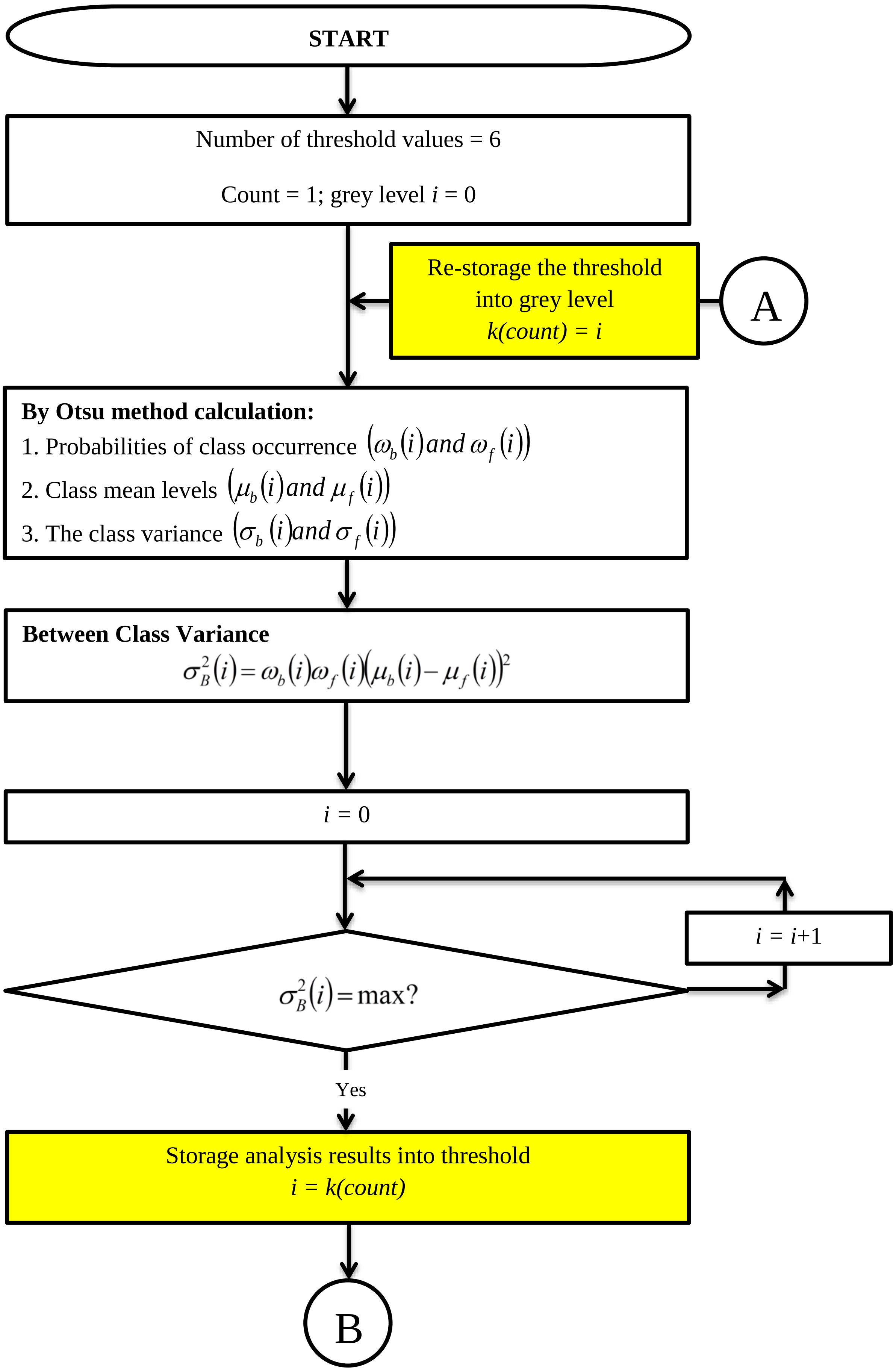


Figure 3.4 Examples of histogram analysis of Multi threshold method


START
Number of threshold values = 6
Count = 1; grey level i = 0
Re-storage the threshold into grey level $k(count) = i$
A
By Otsu method calculation:
1. Probabilities of class occurrence $(\omega_b(i)\, and\, \omega_f(i))$
2. Class mean levels $(\mu_b(i)\, and\, \mu_f(i))$
3. The class variance $(\sigma_b(i)\, and\, \sigma_f(i))$
Between Class Variance
$\sigma_B^2(i) = \omega_b(i)\omega_f(i)(\mu_b(i) - \mu_f(i))^2$
i = 0
i = i+1
$\sigma_B^2(i) = \max?$
Yes
Storage analysis results into threshold $i = k(count)$
B

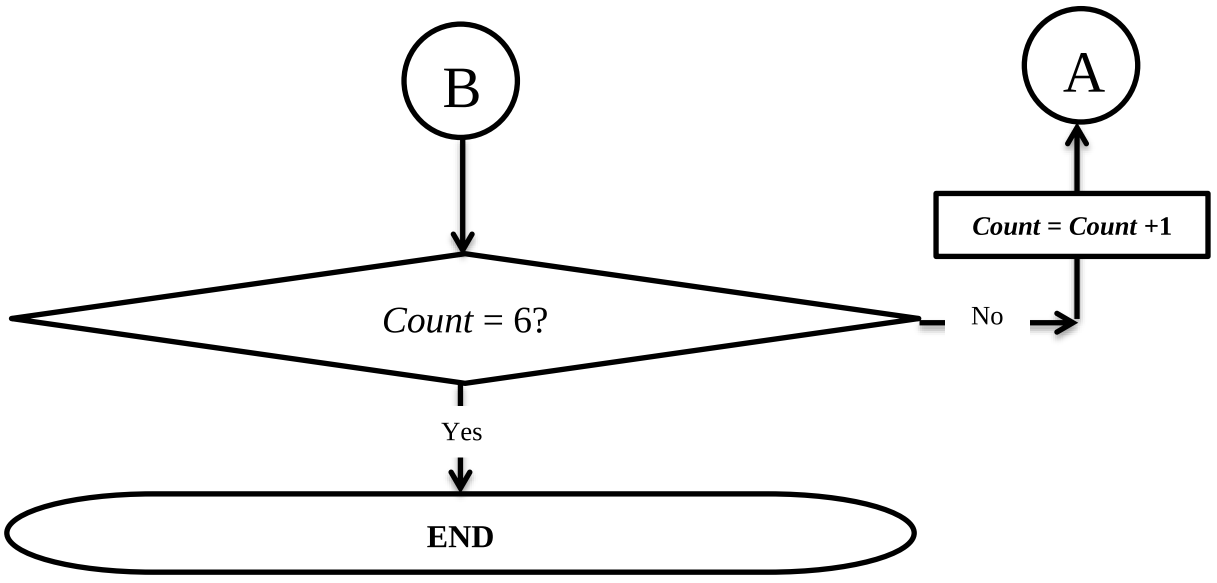


Figure 3.5 Flow diagram of Otsu multi threshold approach

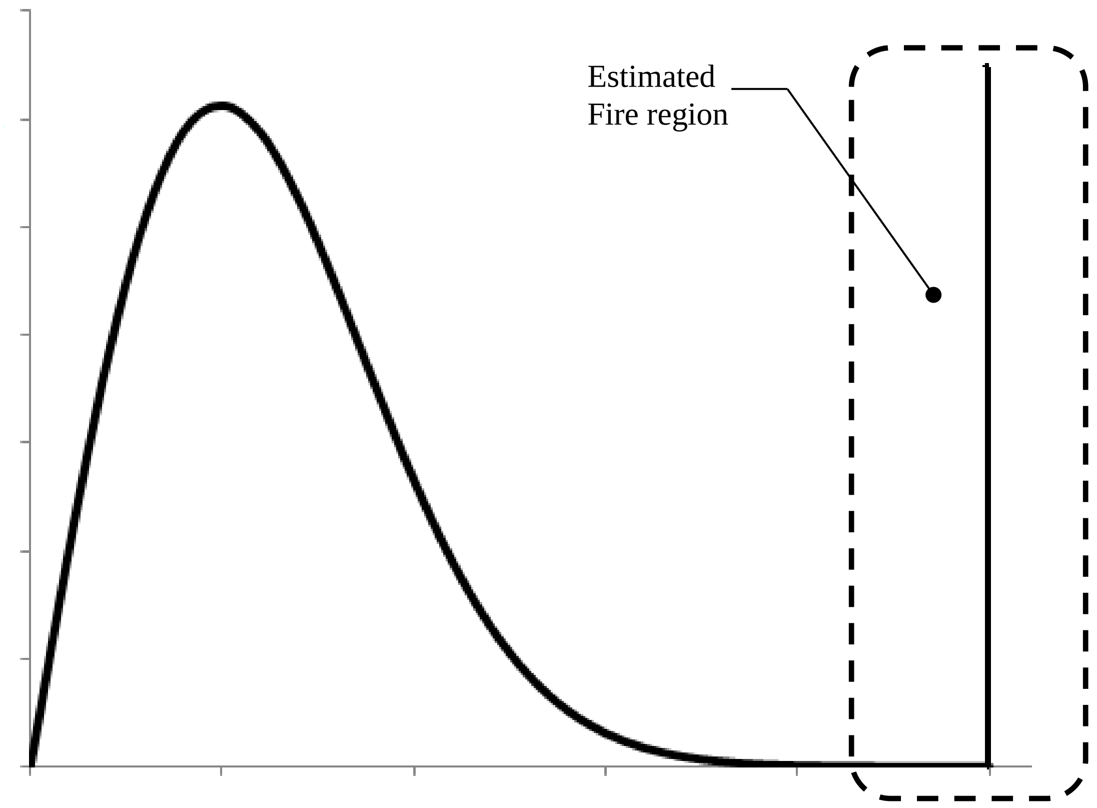
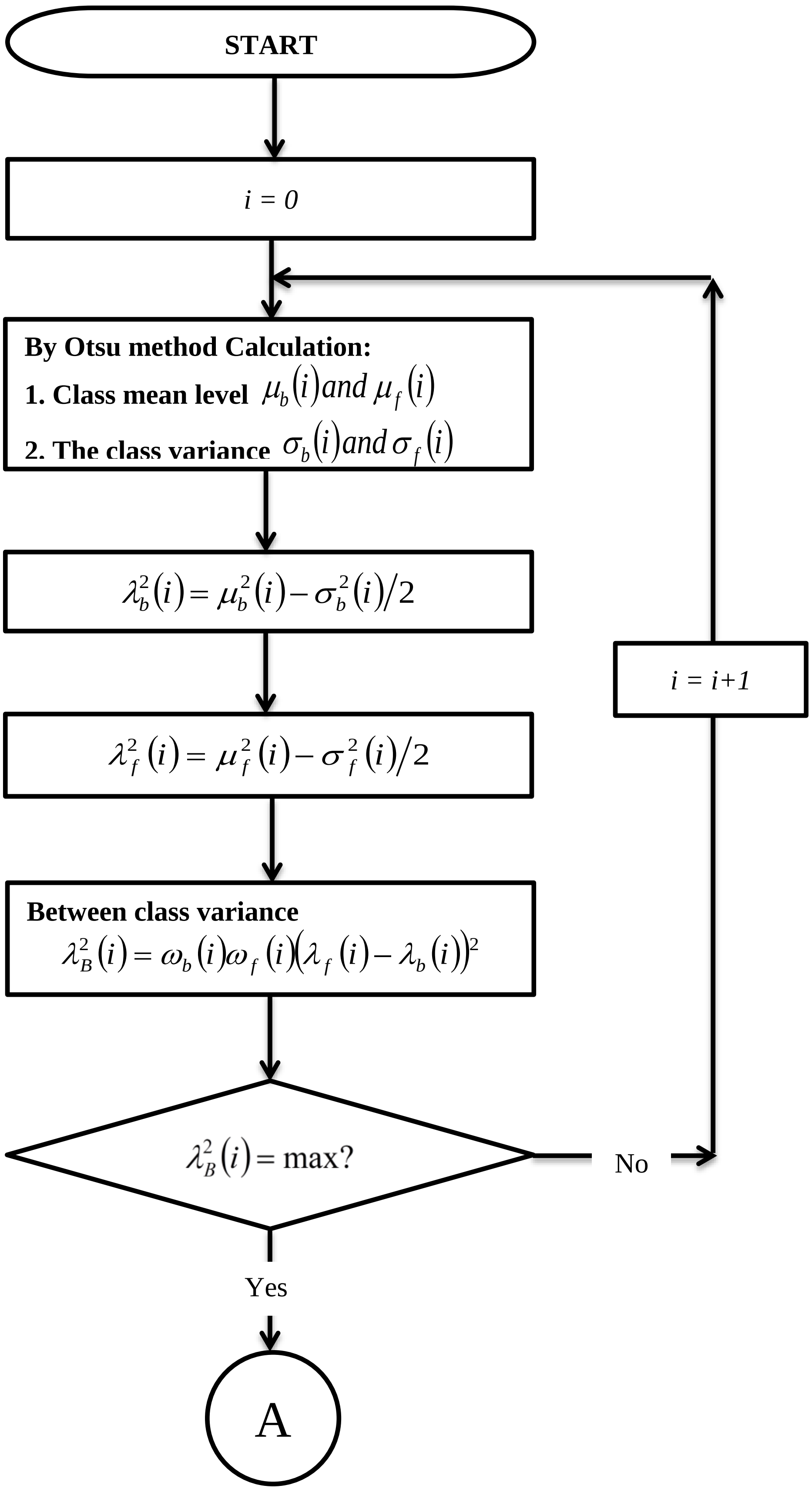


Figure 3.6 Sketch of distribution curve of fire images and estimated fire region


START
$i = 0$
By Otsu method Calculation:
1. Class mean level $\mu_b(i) and\, \mu_f(i)$
2. The class variance $\sigma_b(i) and\, \sigma_f(i)$
$\lambda_b^2(i) = \mu_b^2(i) - \sigma_b^2(i)/2$
$\lambda_f^2(i) = \mu_f^2(i) - \sigma_f^2(i)/2$
Between class variance
$\lambda_B^2(i) = \omega_b(i)\omega_f(i)(\lambda_f(i) - \lambda_b(i))^2$
$\lambda_B^2(i) = \max?$
No
$i = i+1$
Yes
A

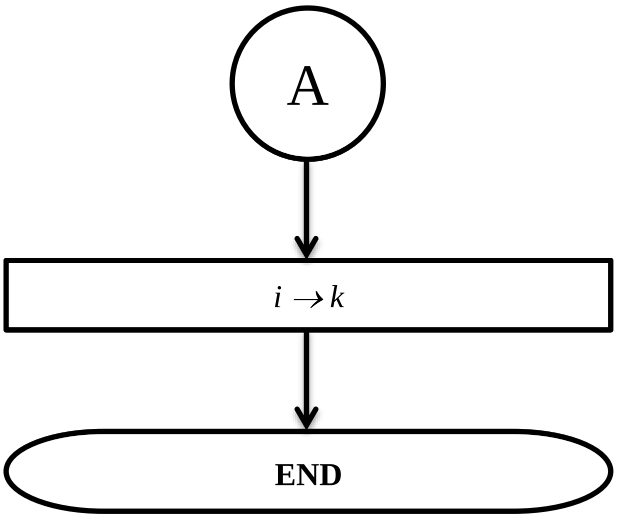
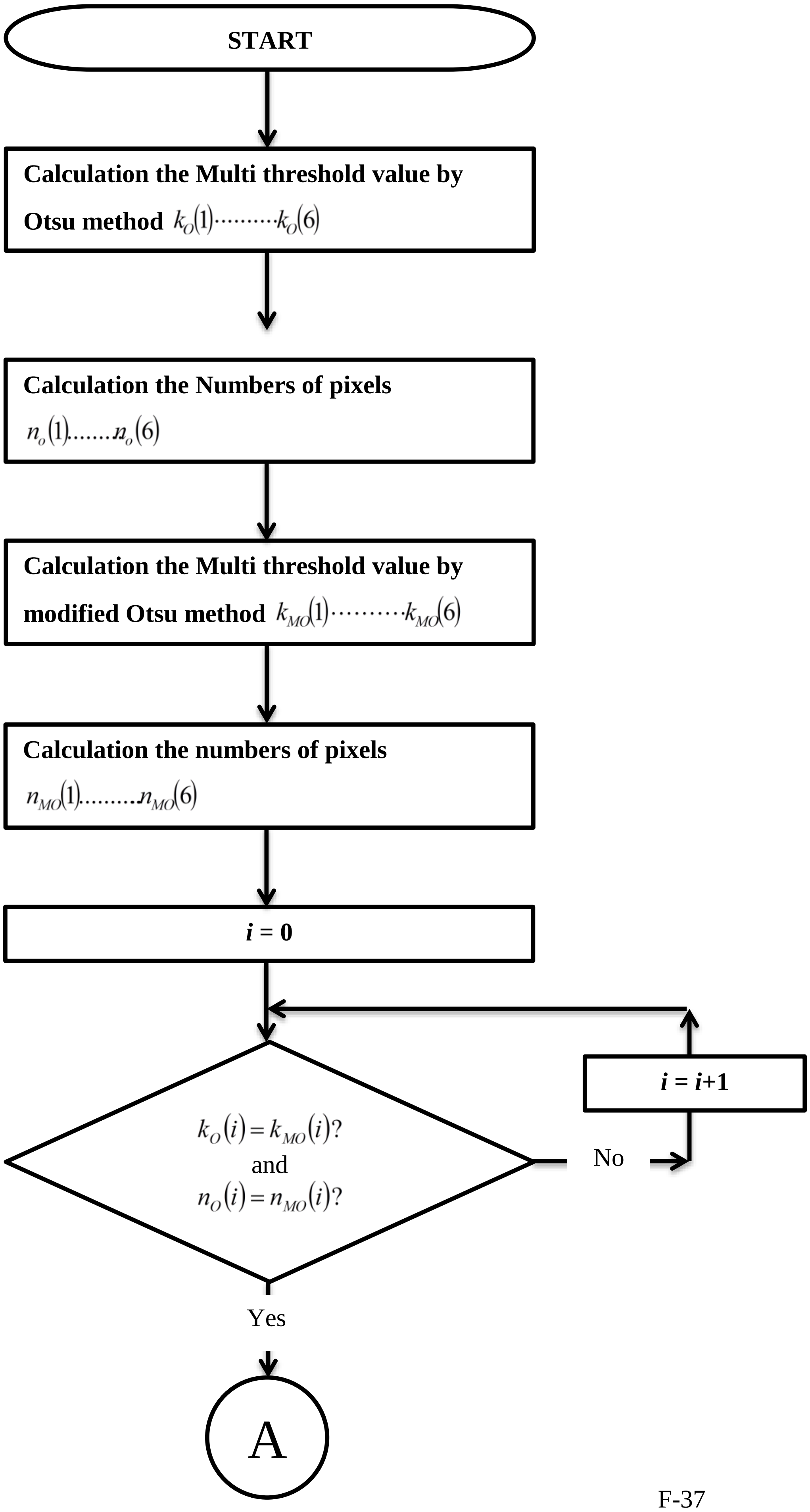


Figure 3.7 Flow diagram of modified Otsu method (Rayleigh distribution analysis)


START
Calculation the Multi threshold value by Otsu method $k_O(1)\cdots\cdots\cdots k_O(6)$
Calculation the Numbers of pixels $n_o(1)\ldots\ldots n_o(6)$
Calculation the Multi threshold value by modified Otsu method $k_{MO}(1)\cdots\cdots\cdots k_{MO}(6)$
Calculation the numbers of pixels $n_{MO}(1)\ldots\ldots\ldots n_{MO}(6)$
$i = 0$
$k_O(i) = k_{MO}(i)$?
and
$n_O(i) = n_{MO}(i)$?
No
$i = i+1$
Yes
A

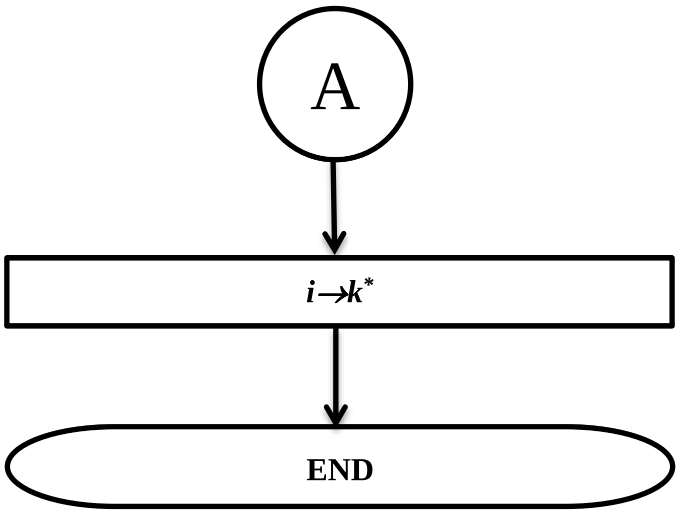


Figure 3.8 Flow diagram of optimal threshold selection method

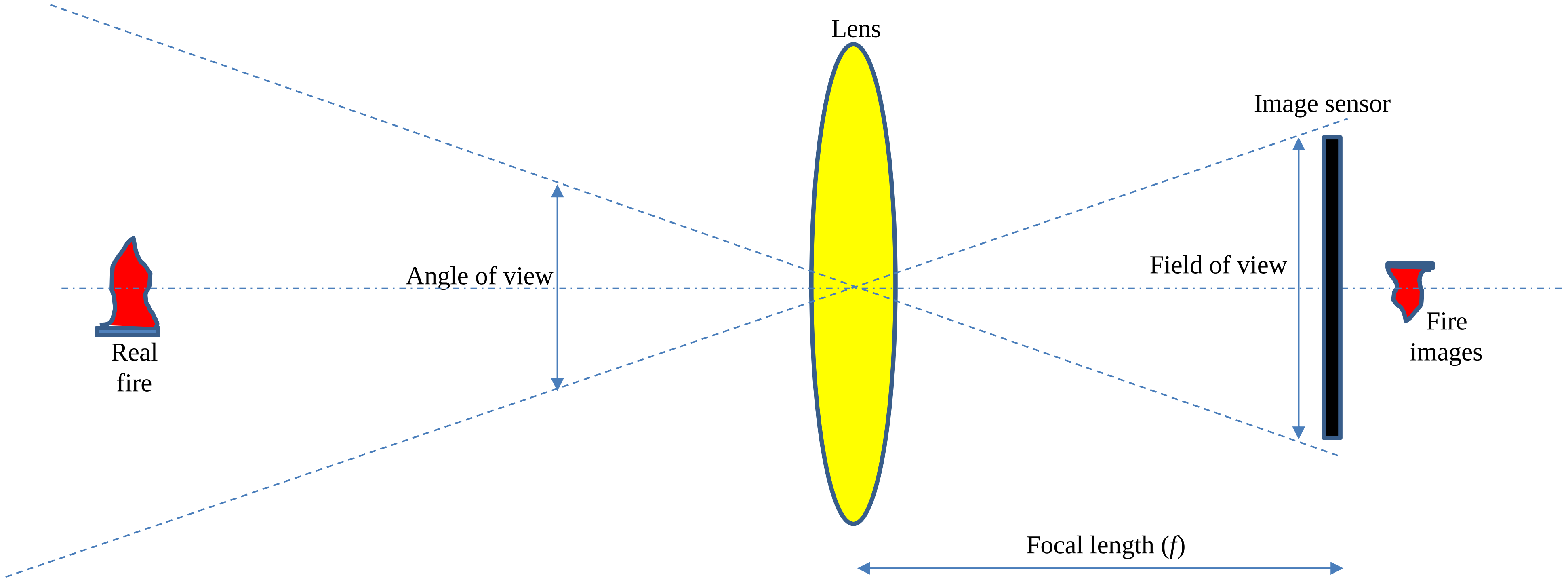


Figure 3.9 Field of View and Focal length

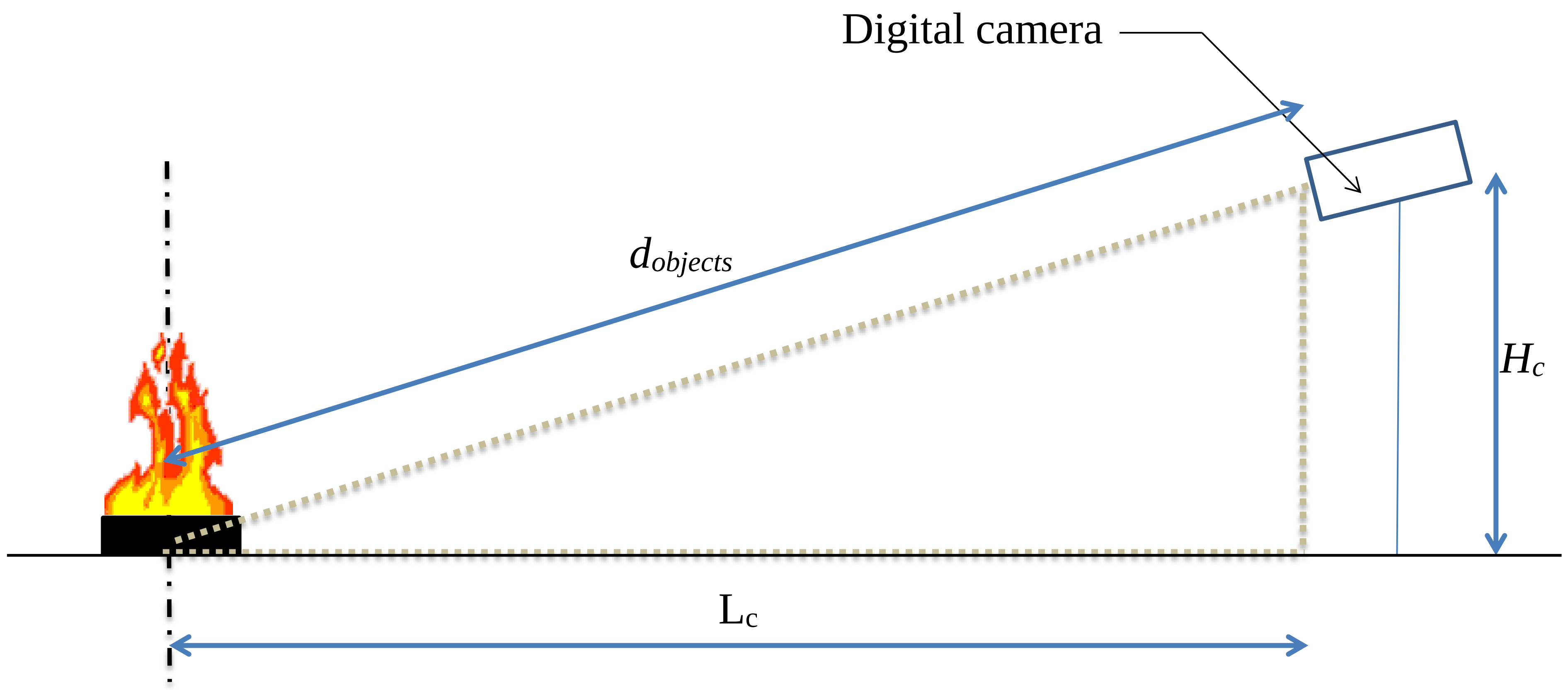


Figure 3.10 Schematic of digital camera and fire pool setup

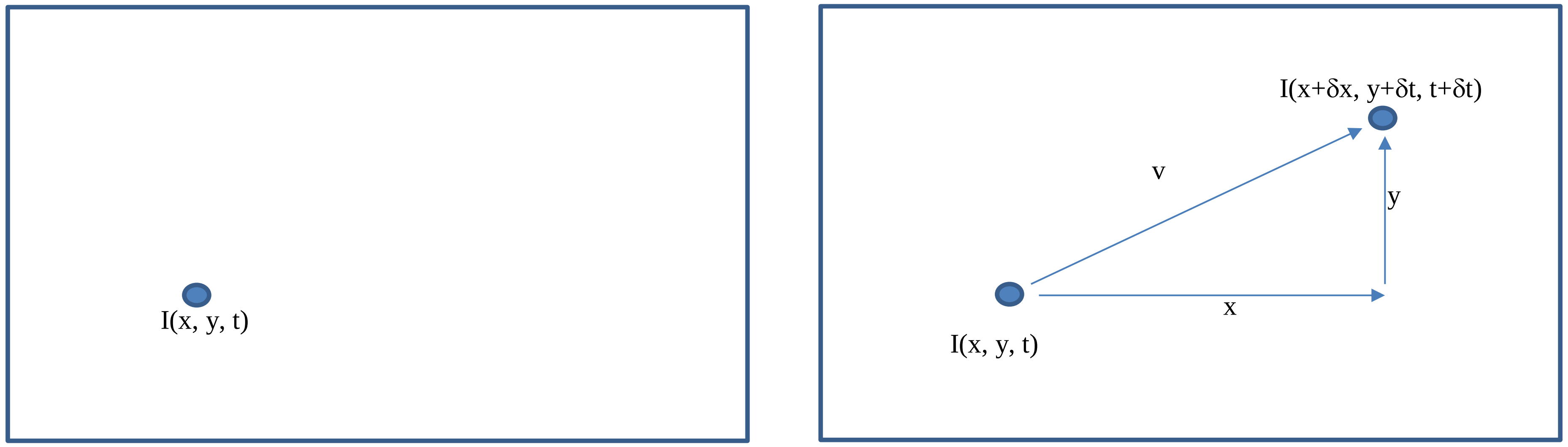


Figure 3.11 Theory of optical flow analysis

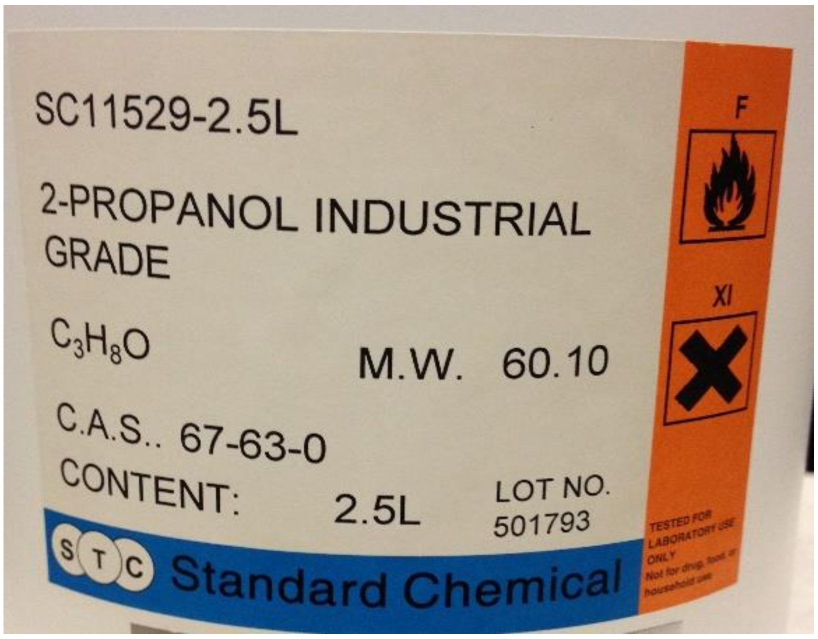


Figure 4.1 Property information of propanol fuel

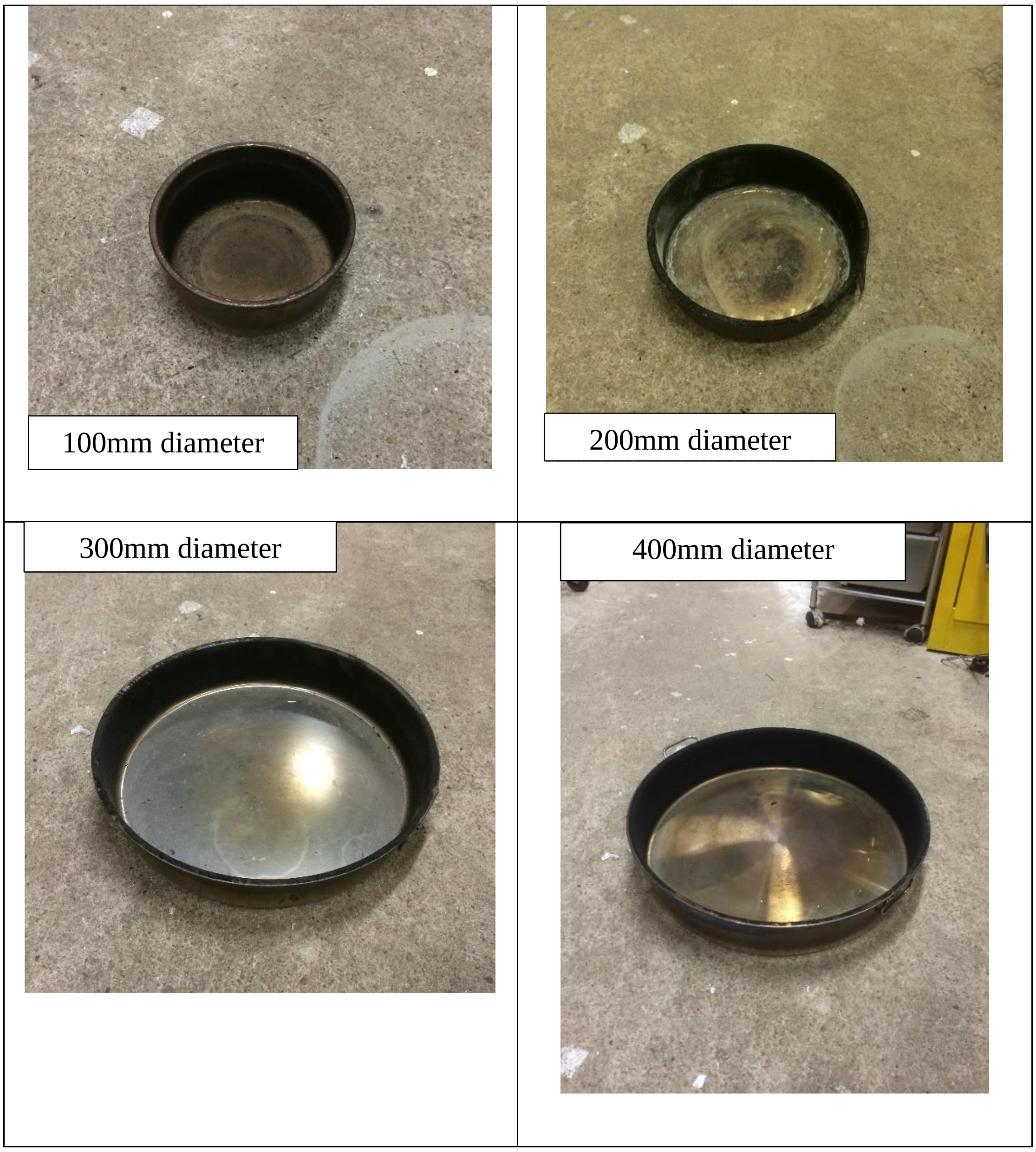


Figure 4.2 Different pool diameter for experimental study

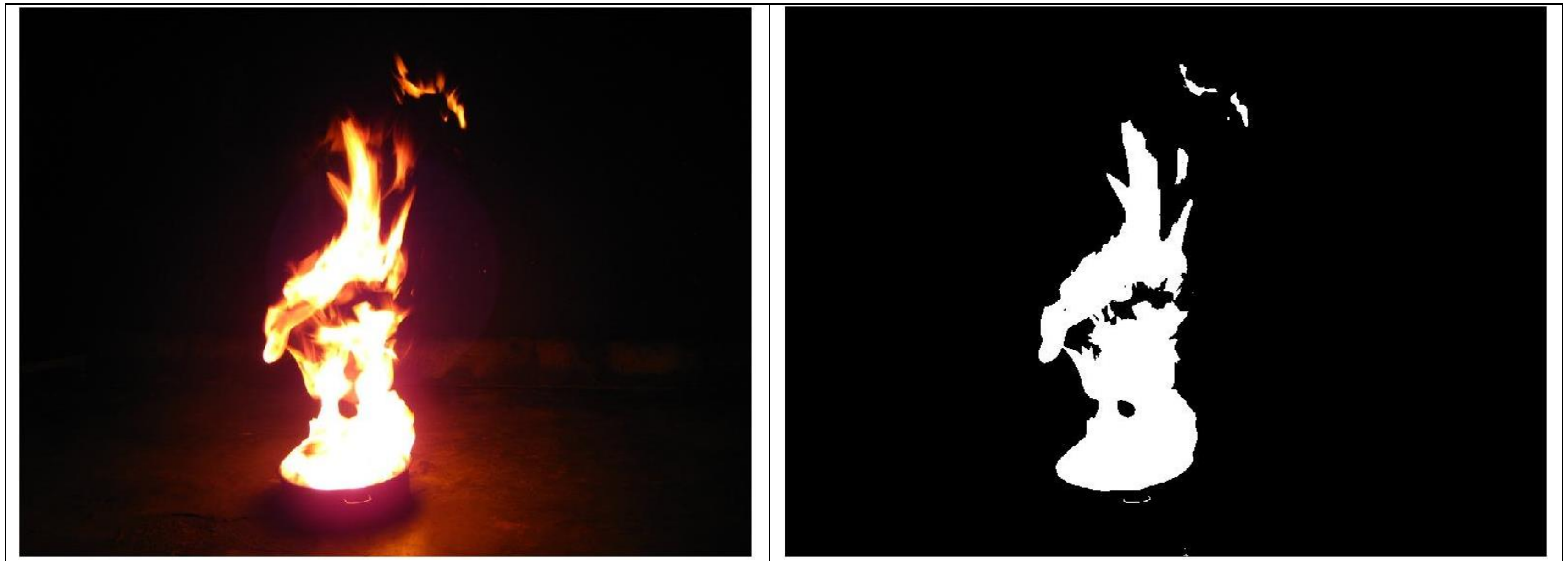


Figure 4.3 Colour images segmentation results

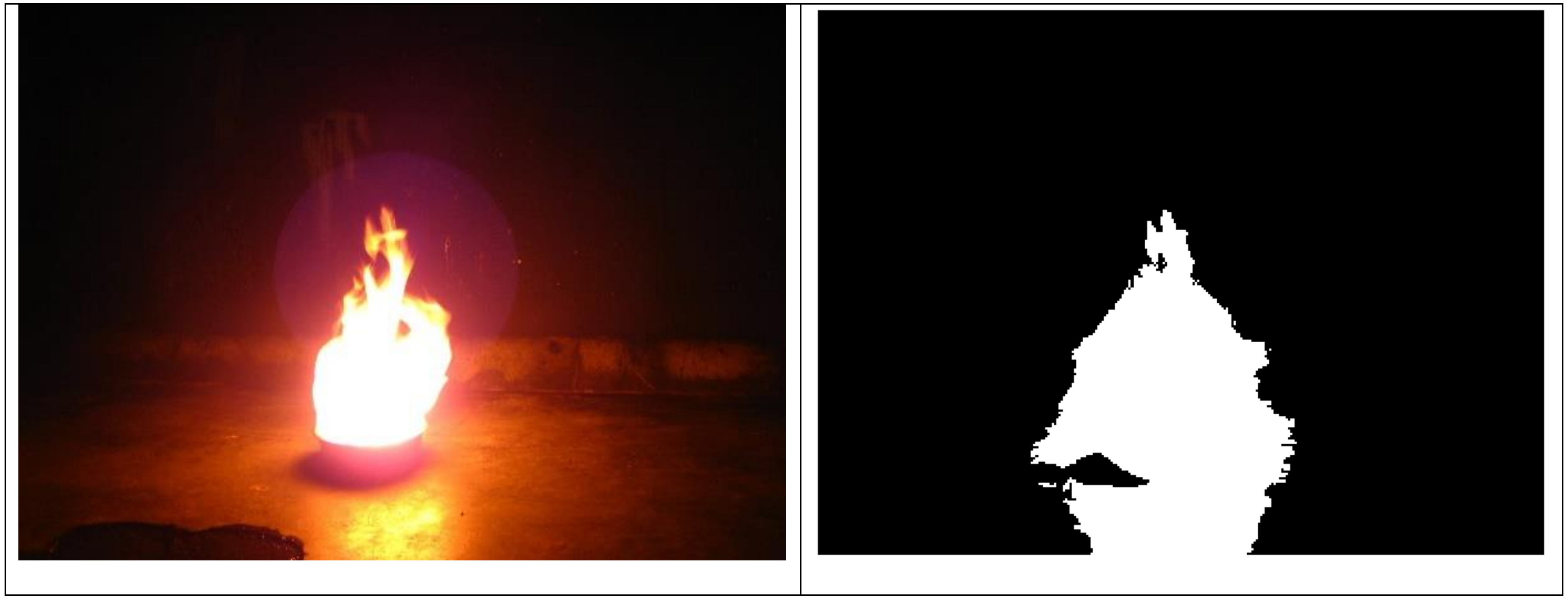


Figure 4.4 Colour images segmentation results (Distortion)

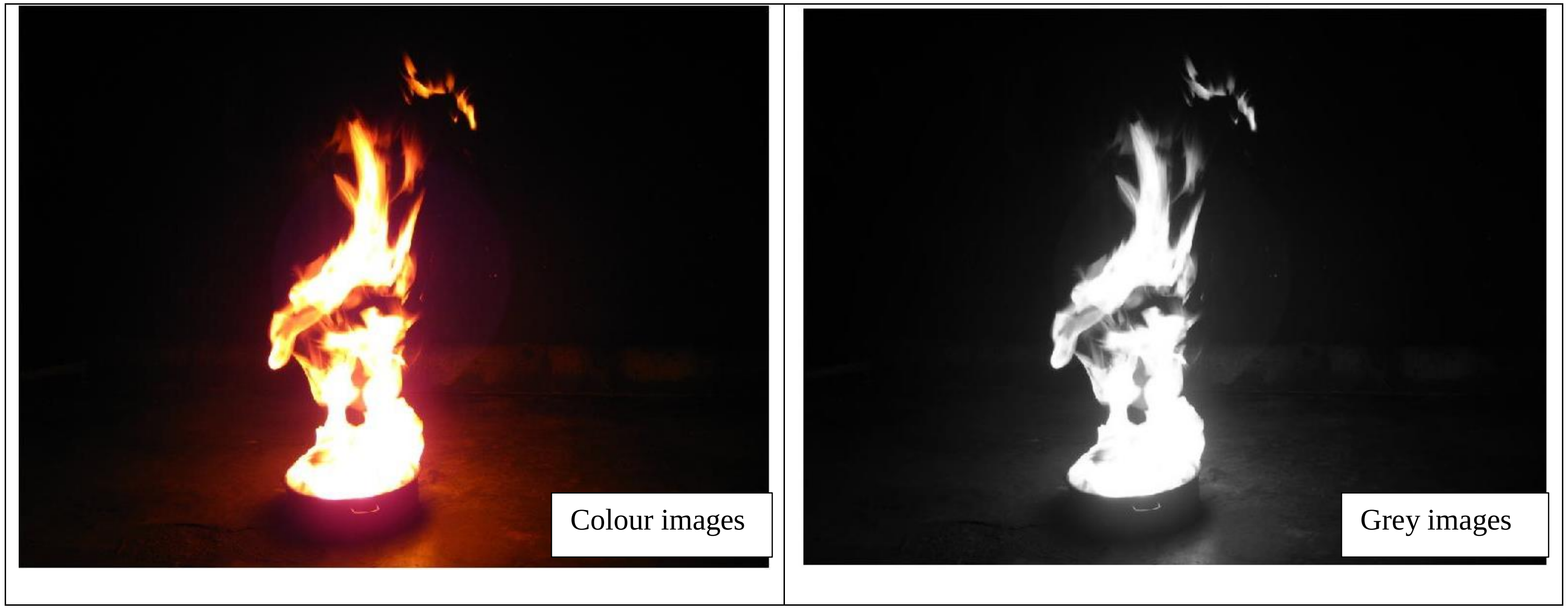
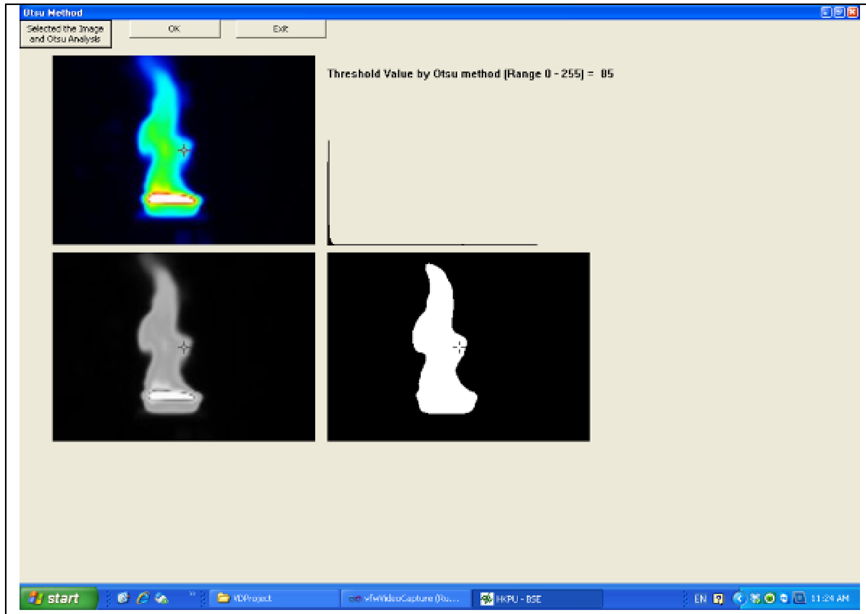
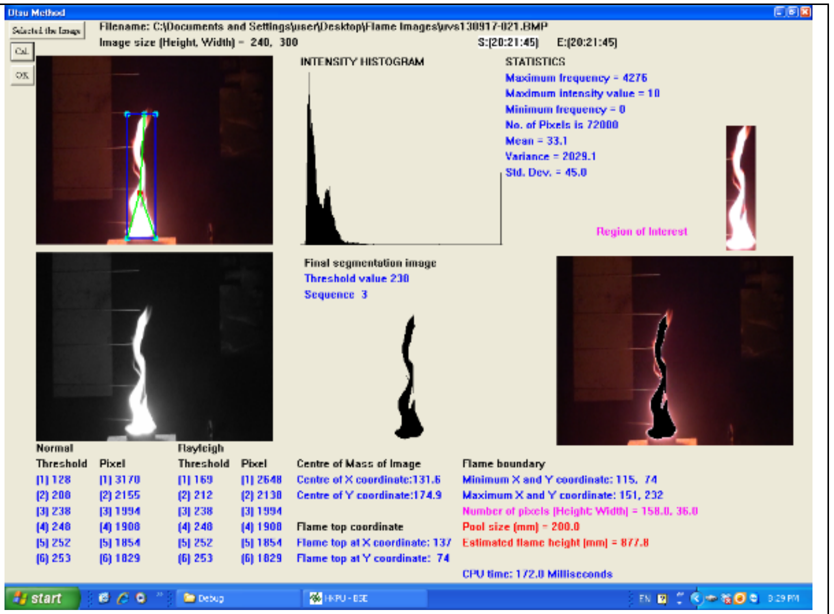


Figure 4.5 Original colour images and Greyscale images



Threshold Value by Otsu method [Range 0 - 255] = 85



Filename: C:\Documents and Settings\user\Desktop\Flame Images\vs130917-021.BMP
Image size (Height, Width) = 240, 300 S:[20:21:45] E:[20:21:45]

STATISTICS
Maximum frequency = 4276
Maximum intensity value = 10
Minimum frequency = 0
No. of Pixels is 72000
Mean = 33.1
Variance = 2029.1
Std. Dev. = 45.0

Final segmentation image
Threshold value 230
Sequence 3

```
Normal                    Rayleigh
Threshold   Pixel         Threshold   Pixel      Centre of Mass of Image               Flame boundary
[1] 128     [1] 3170      [1] 169     [1] 2648   Centre of X coordinate:131.6          Minimum X and Y coordinate: 115, 74
[2] 208     [2] 2155      [2] 212     [2] 2130   Centre of Y coordinate:174.9          Maximum X and Y coordinate: 151, 232
[3] 238     [3] 1994      [3] 238     [3] 1994                                         Number of pixels [Height: Width] = 158.0, 36.0
[4] 248     [4] 1908      [4] 248     [4] 1908   Flame top coordinate                  Pool size (mm) = 200.0
[5] 252     [5] 1854      [5] 252     [5] 1854   Flame top at X coordinate: 137        Estimated flame height (mm) = 877.8
[6] 253     [6] 1829      [6] 253     [6] 1829   Flame top at Y coordinate: 74
                                                                                       CPU time: 172.0 Milliseconds
```

Figure 4.6 Thermal images (Left) Colour still images (Right) Computer output

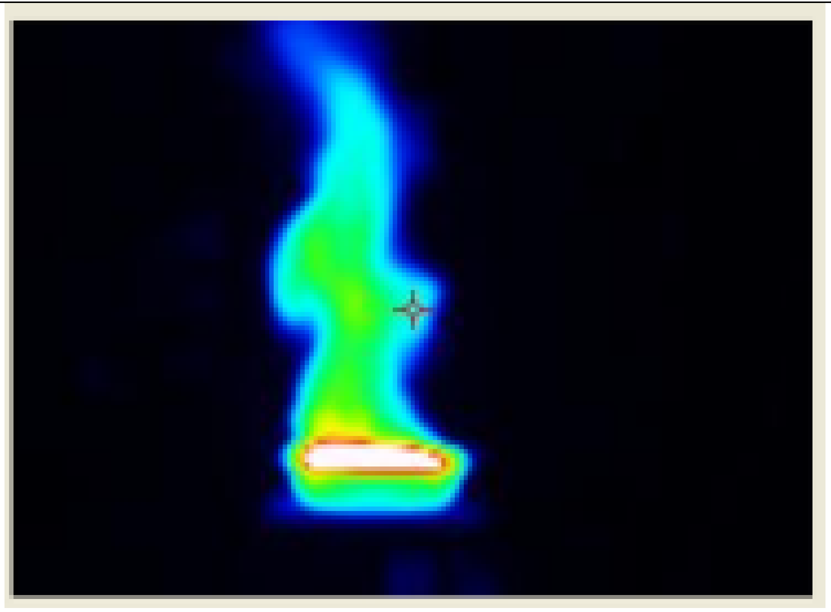
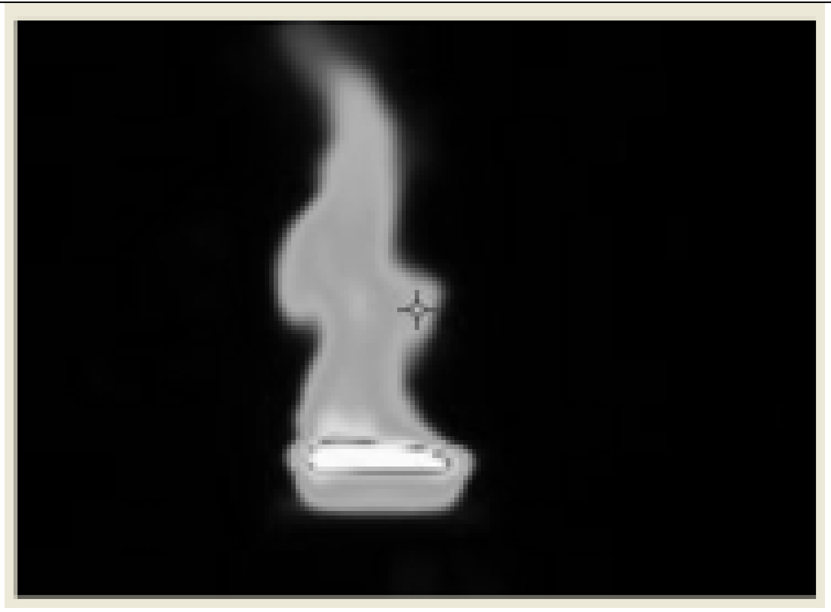



Figure 4.7 Thermal images (Left) Grey colour images (Right)

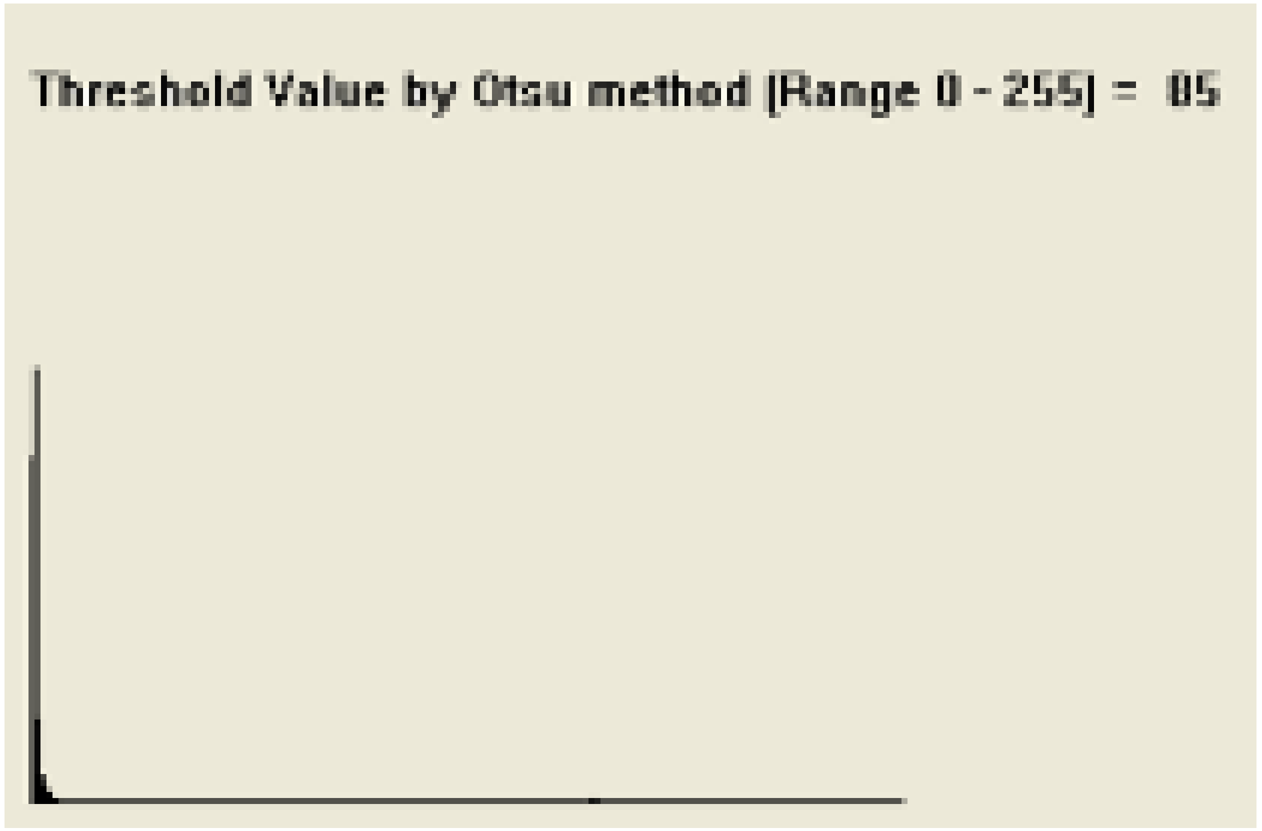

Threshold Value by Otsu method [Range 0 - 255] = 85

Figure 4.8 Histogram and Calculation result.

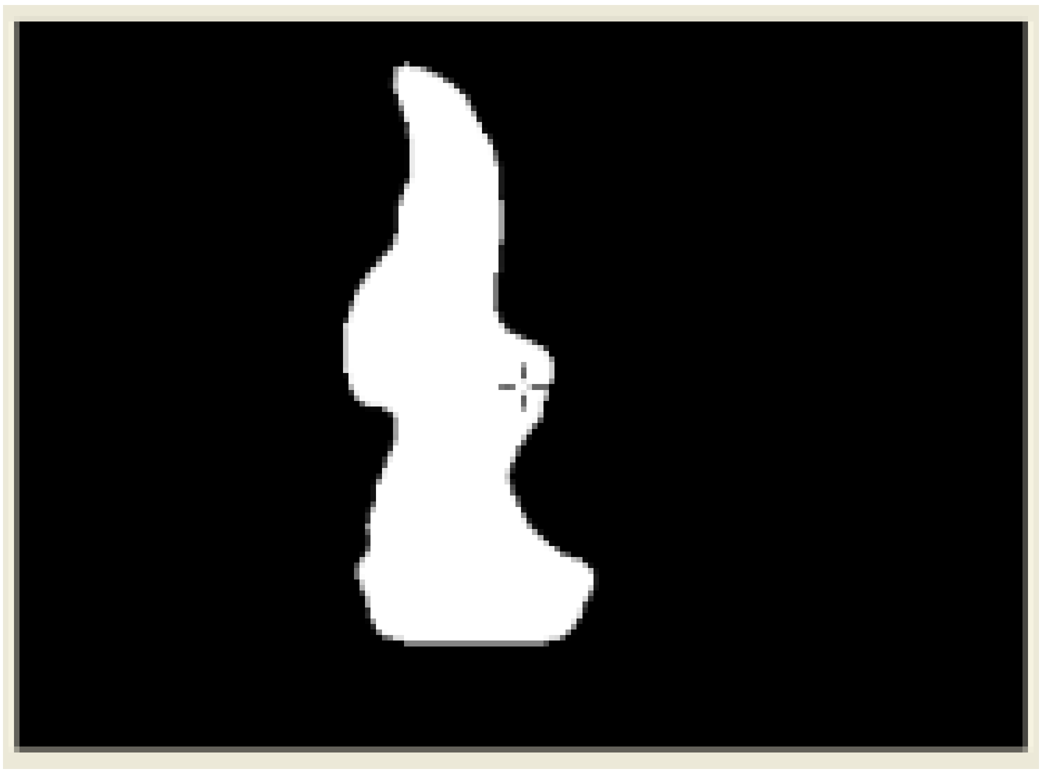


Figure 4.9 Binary image (Threshold value = 85)

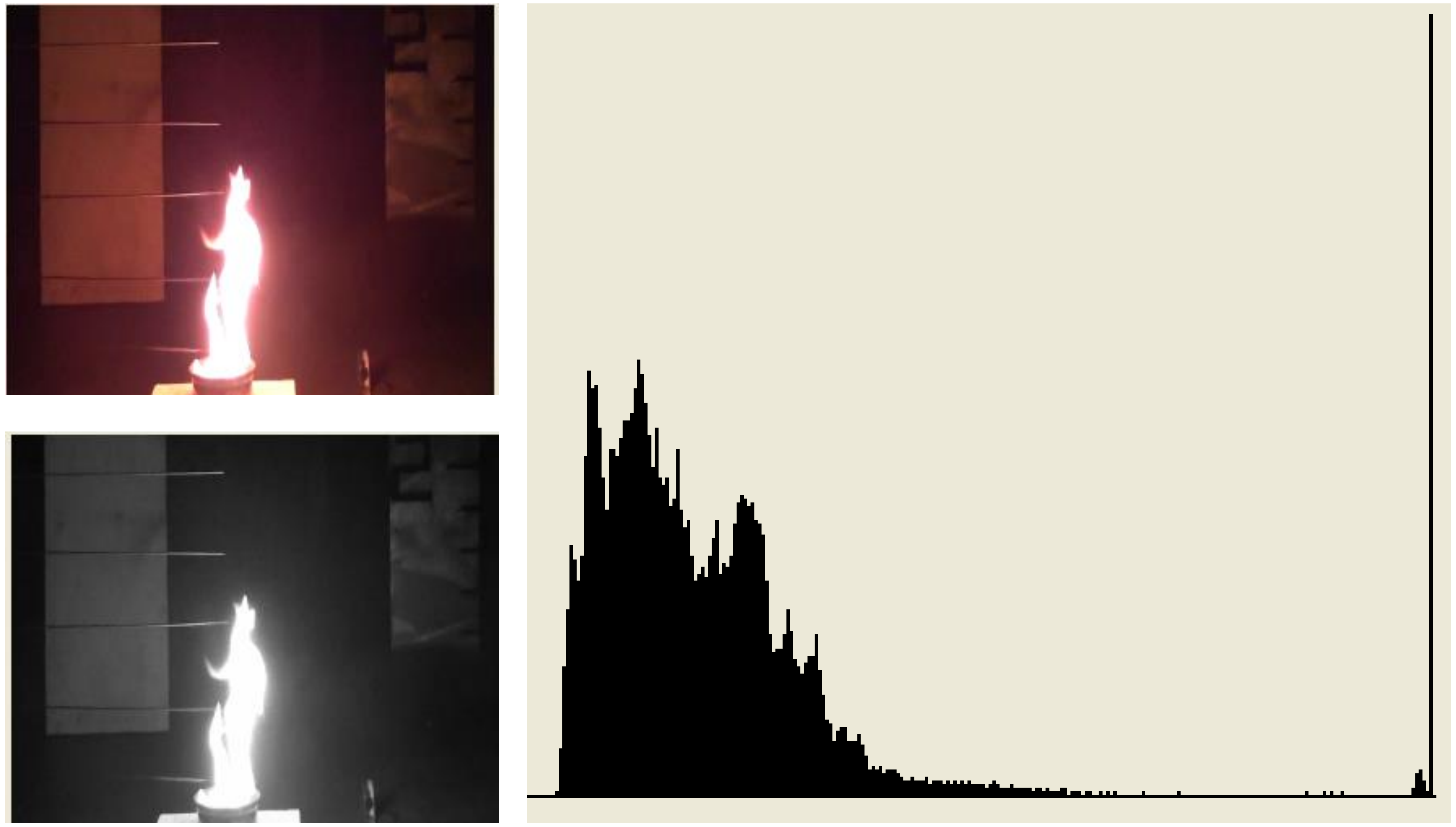
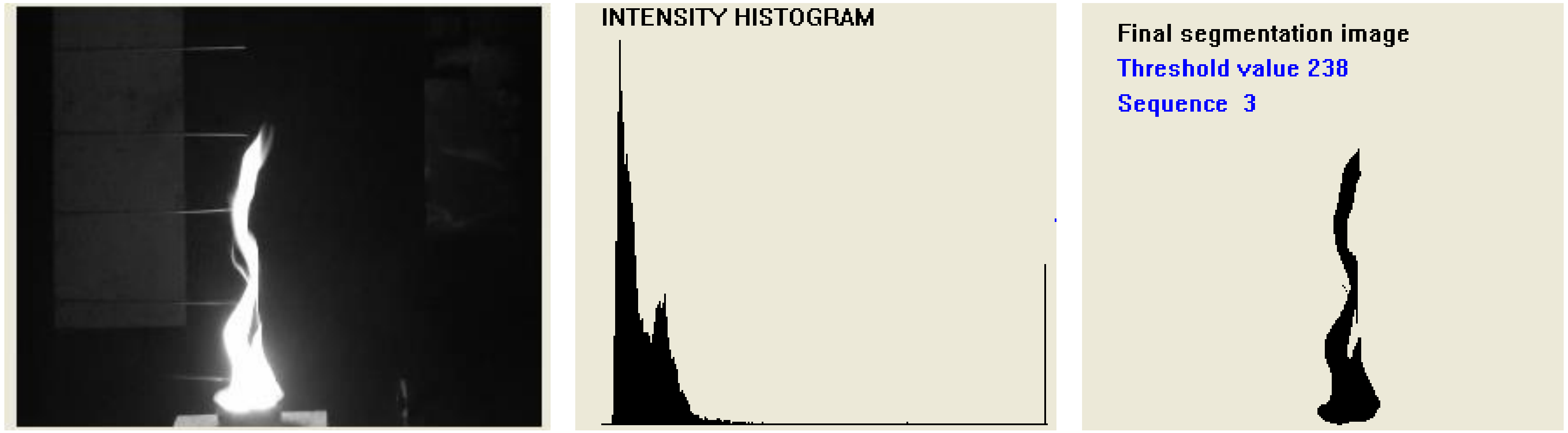


Figure 4.10 Computer output of greyscale images and histogram (still images)



(a) Grey images (b) Histogram (c) Segmentation images

Figure 4.11 Experimental results

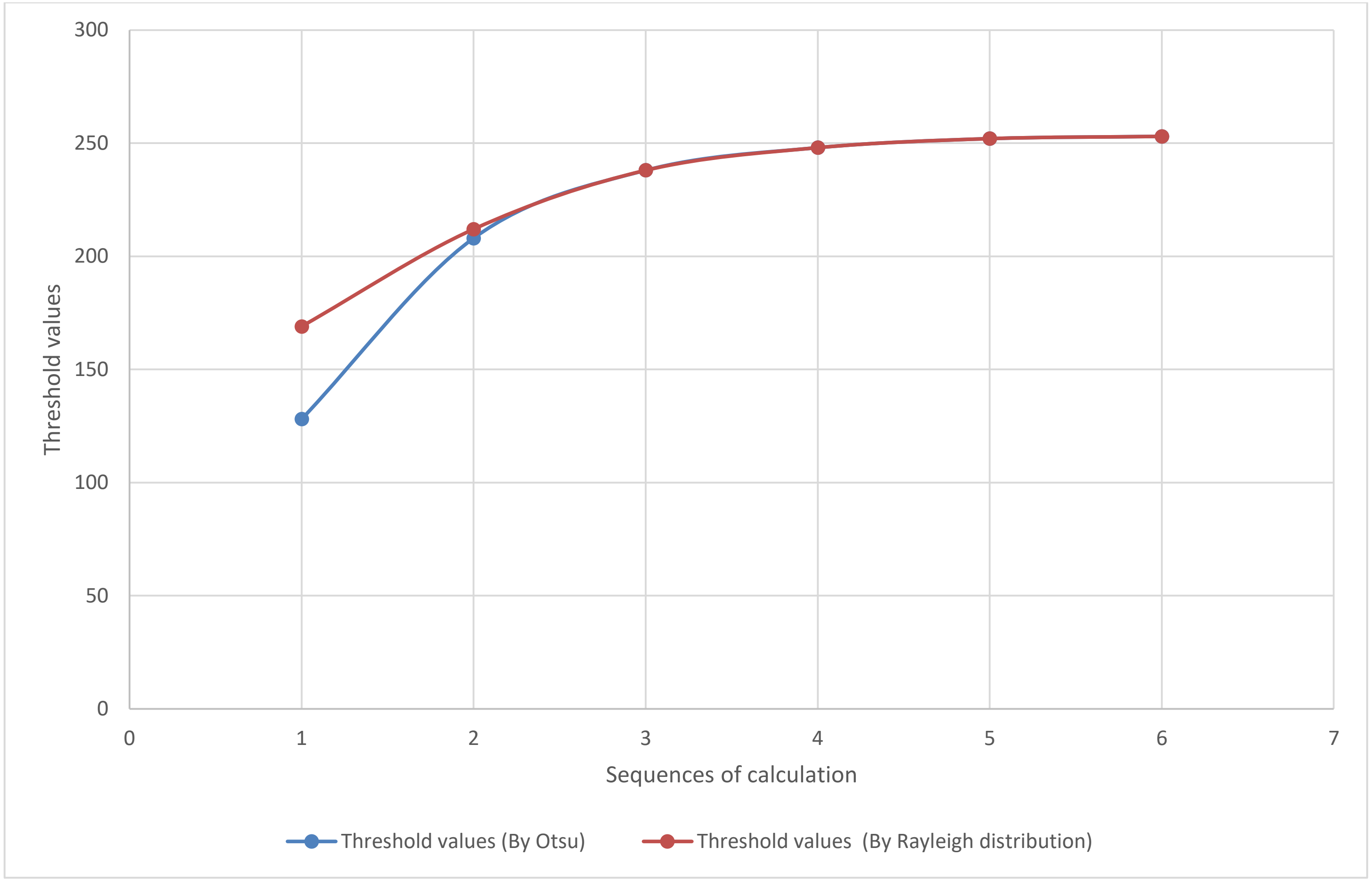


Figure 4.12 Calculation results of traditional Otsu method and Modified Otsu method

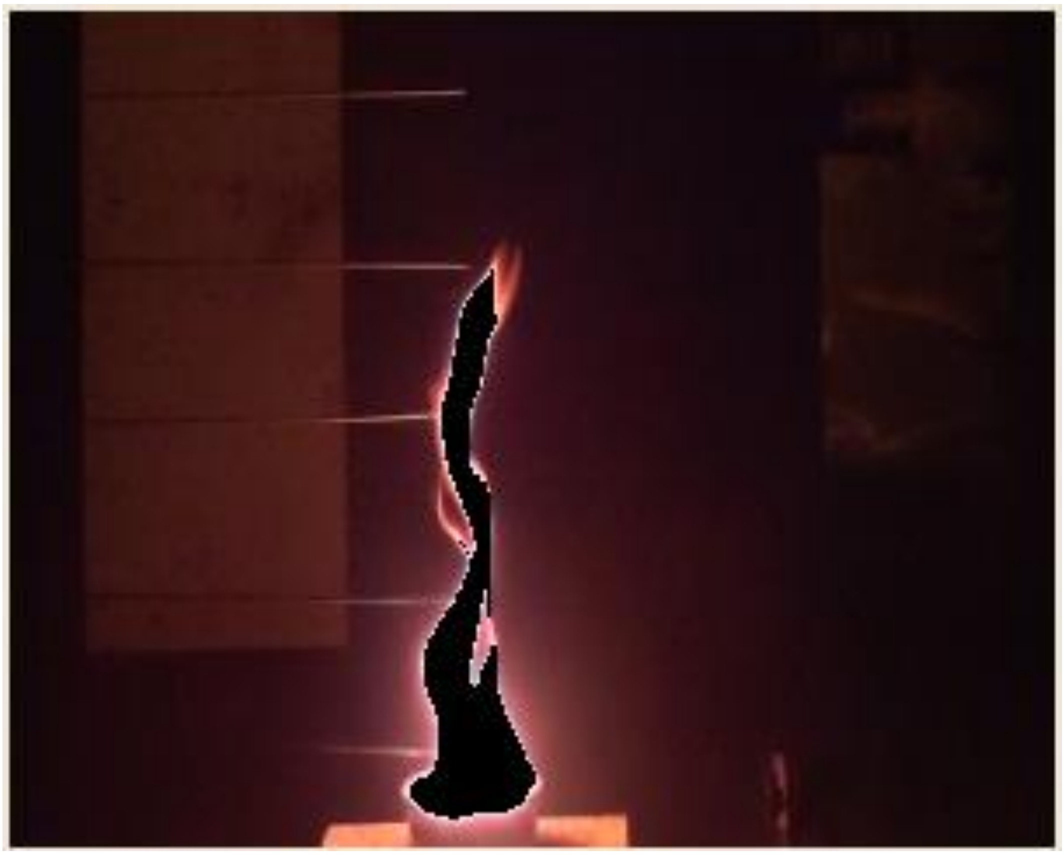


Figure 4.13 Segmentation images overlap on the original colour flame images

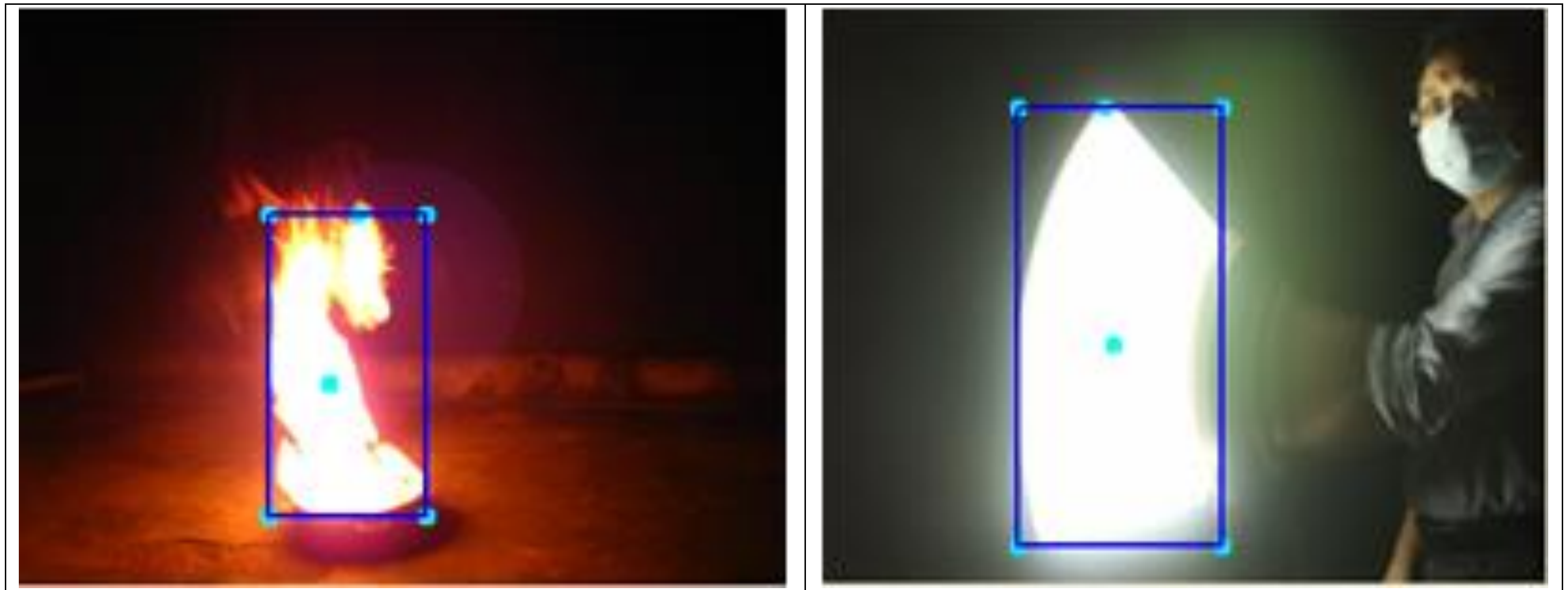


Figure 4.14 Flame images (left) non – flame images (right)

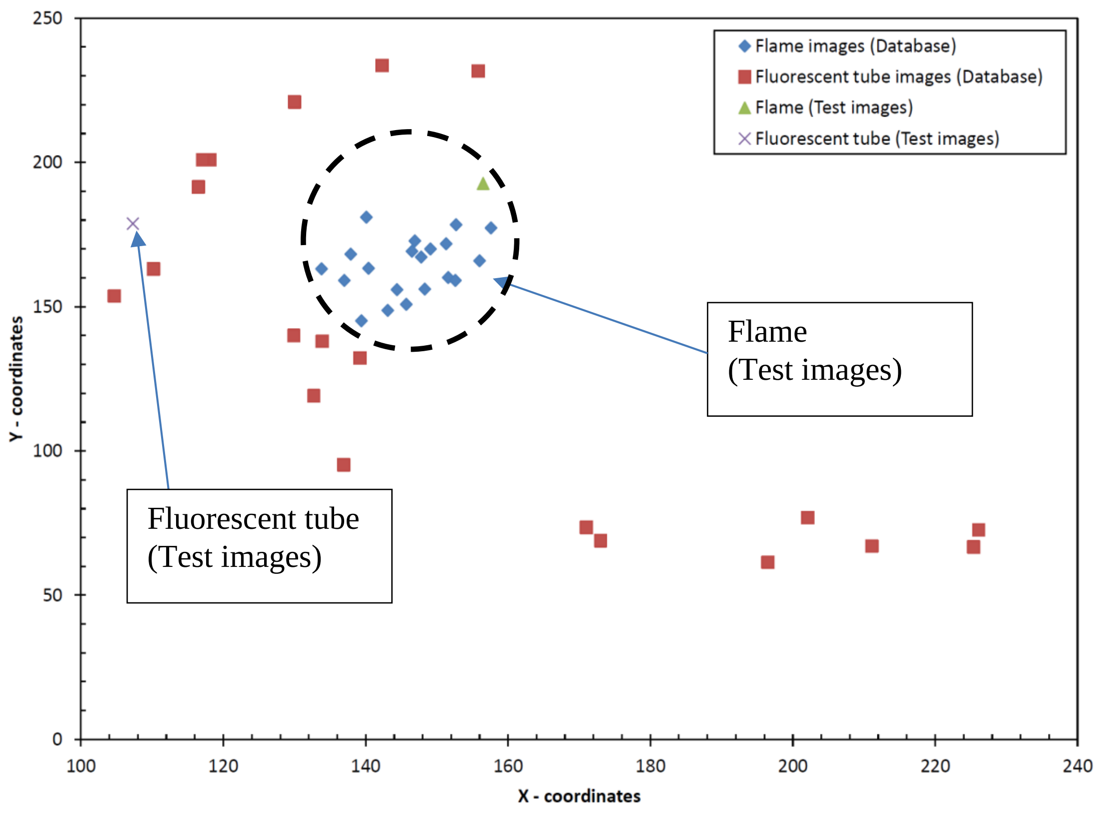


Figure 4.15 Calculation and analysis results of x and y coordinate

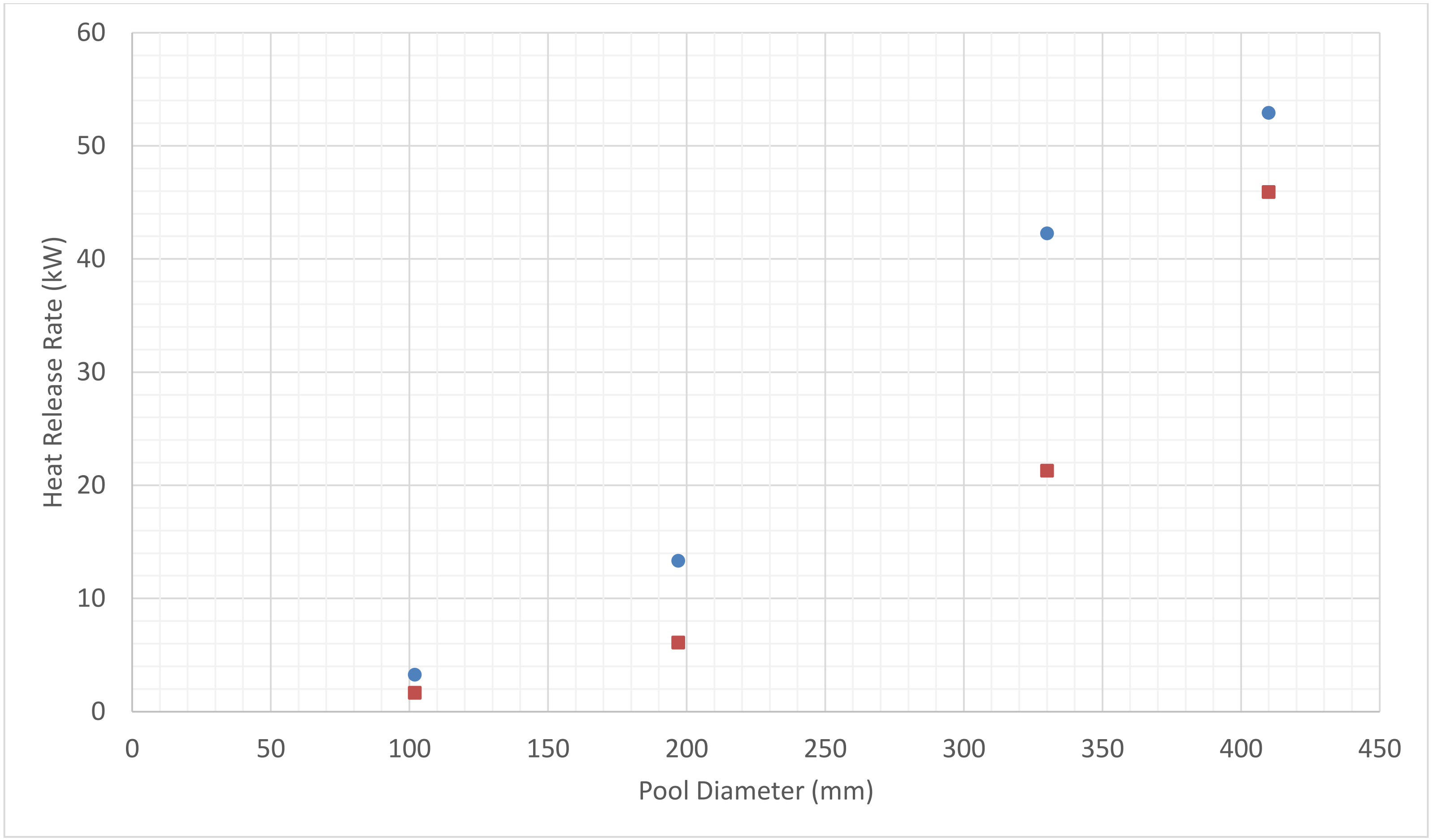


Figure 4.16 HRR curve against Pool diameter

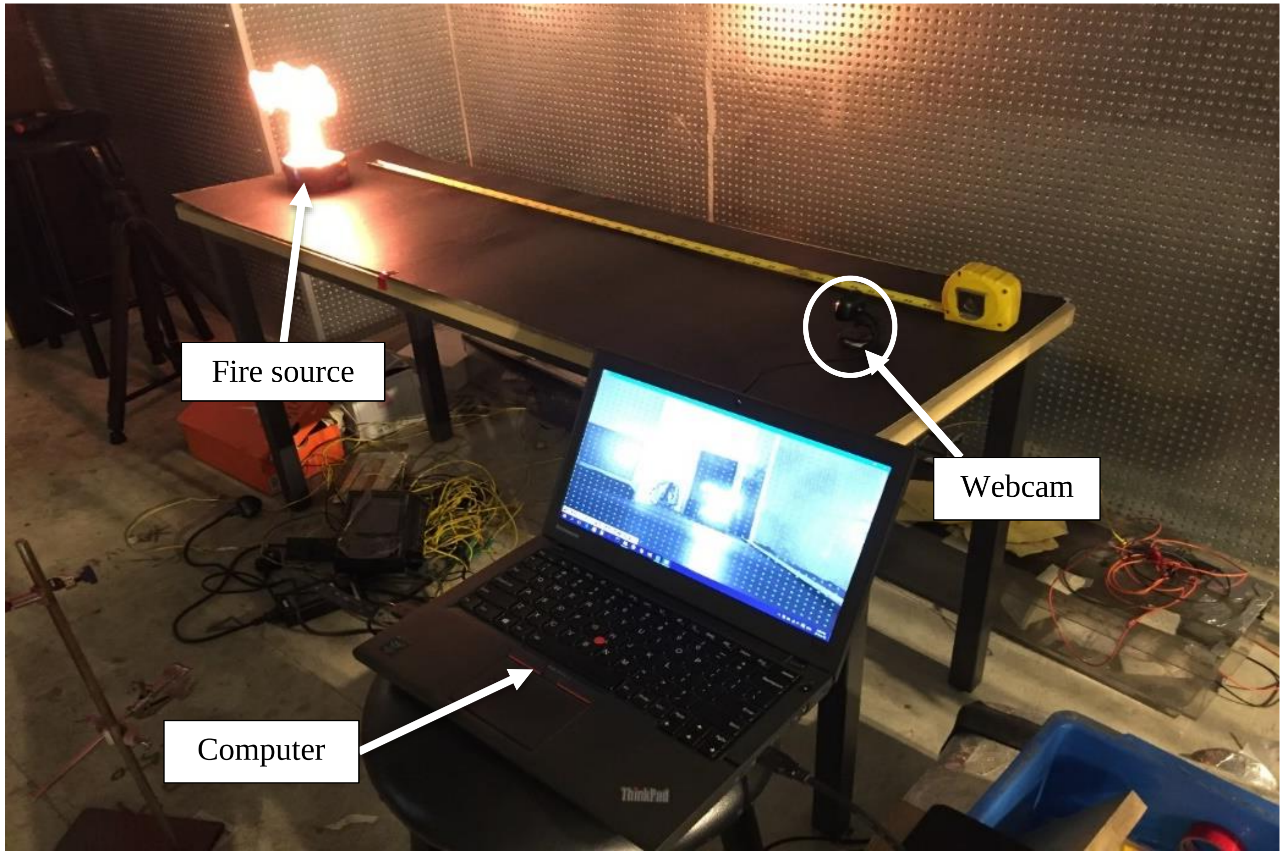


Figure 4.17 Experimental setup in fire chamber

Index	Histogram	Index	Images
A	Red colour histogram	**1**	Colour input images.
B	Green colour histogram	**2**	Grey images
C	Blue colour histogram	**3**	Segmentation images
D	Grey histogram	**4**	Optical flow images
E	Combined histogram (including RGB and grey)	**5**	Threshold images
F	Flame height flickering	**6**	Output images
		7	Image difference

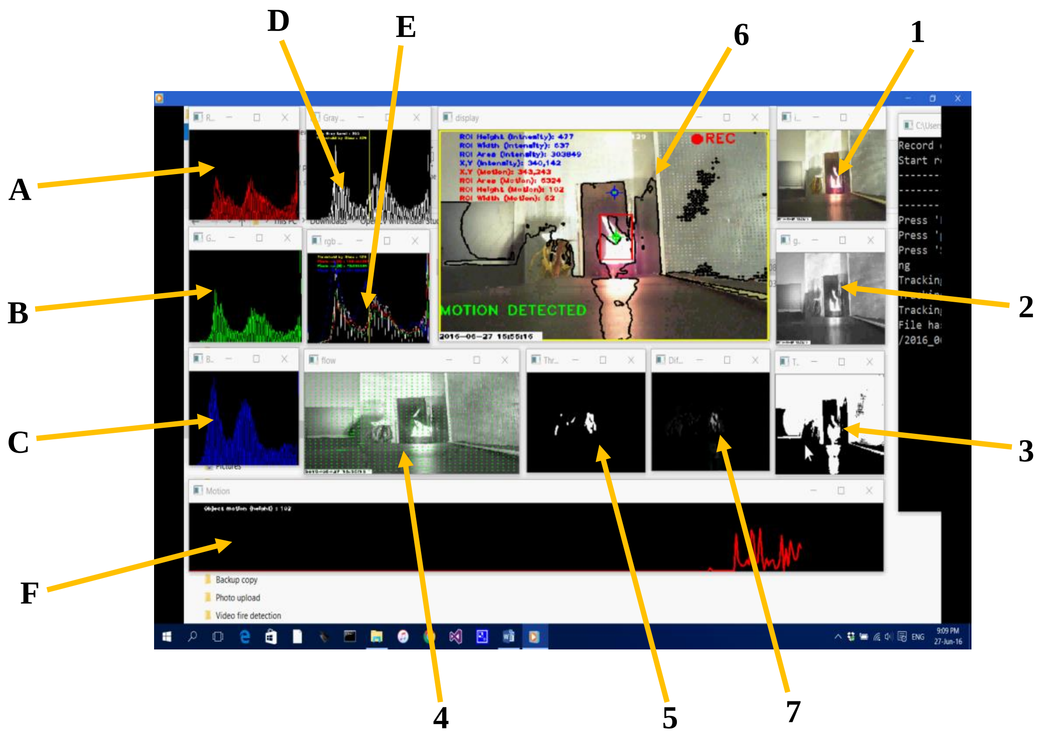


Figure 4.18 Output displayed on a computer screen

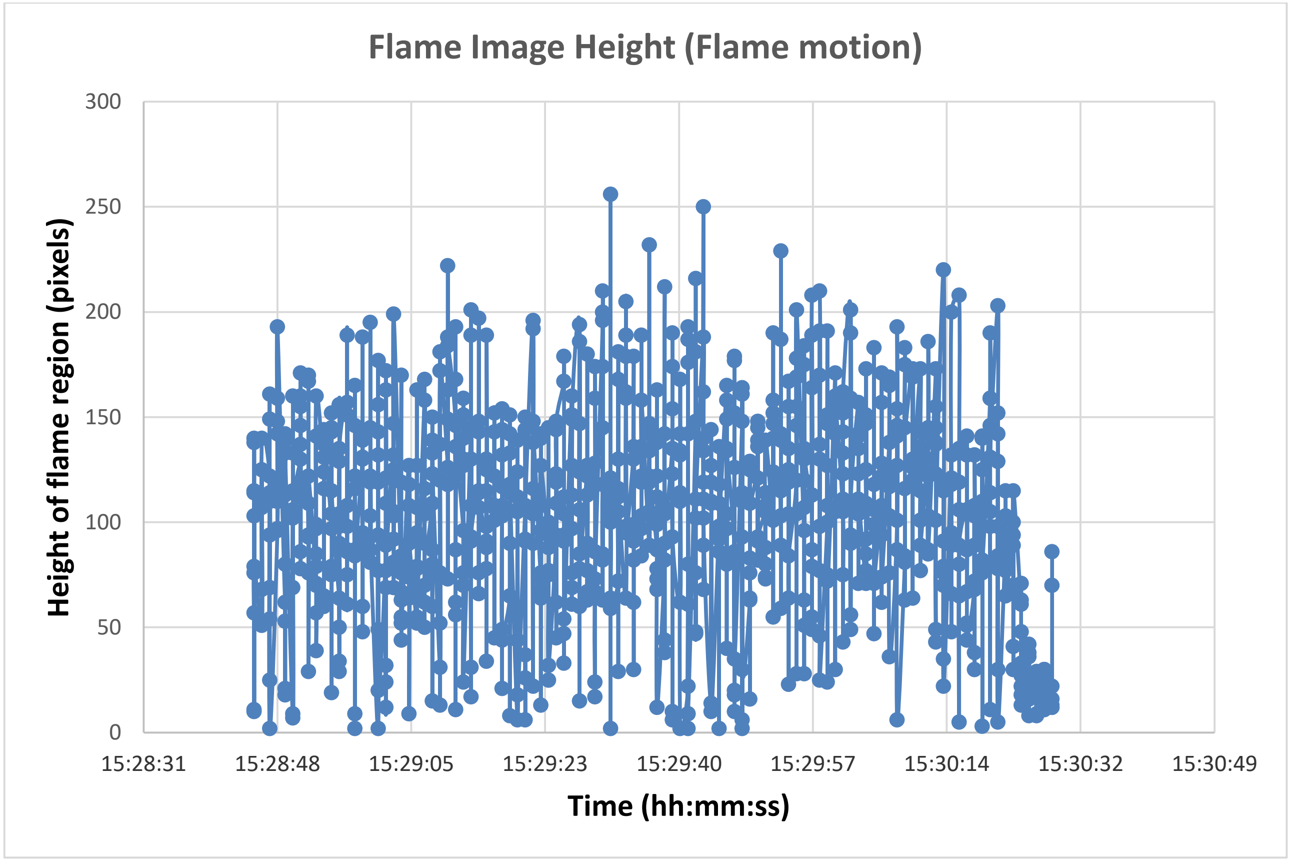


Figure 4.19 Flame Image Height (Flame motion) (10mL)

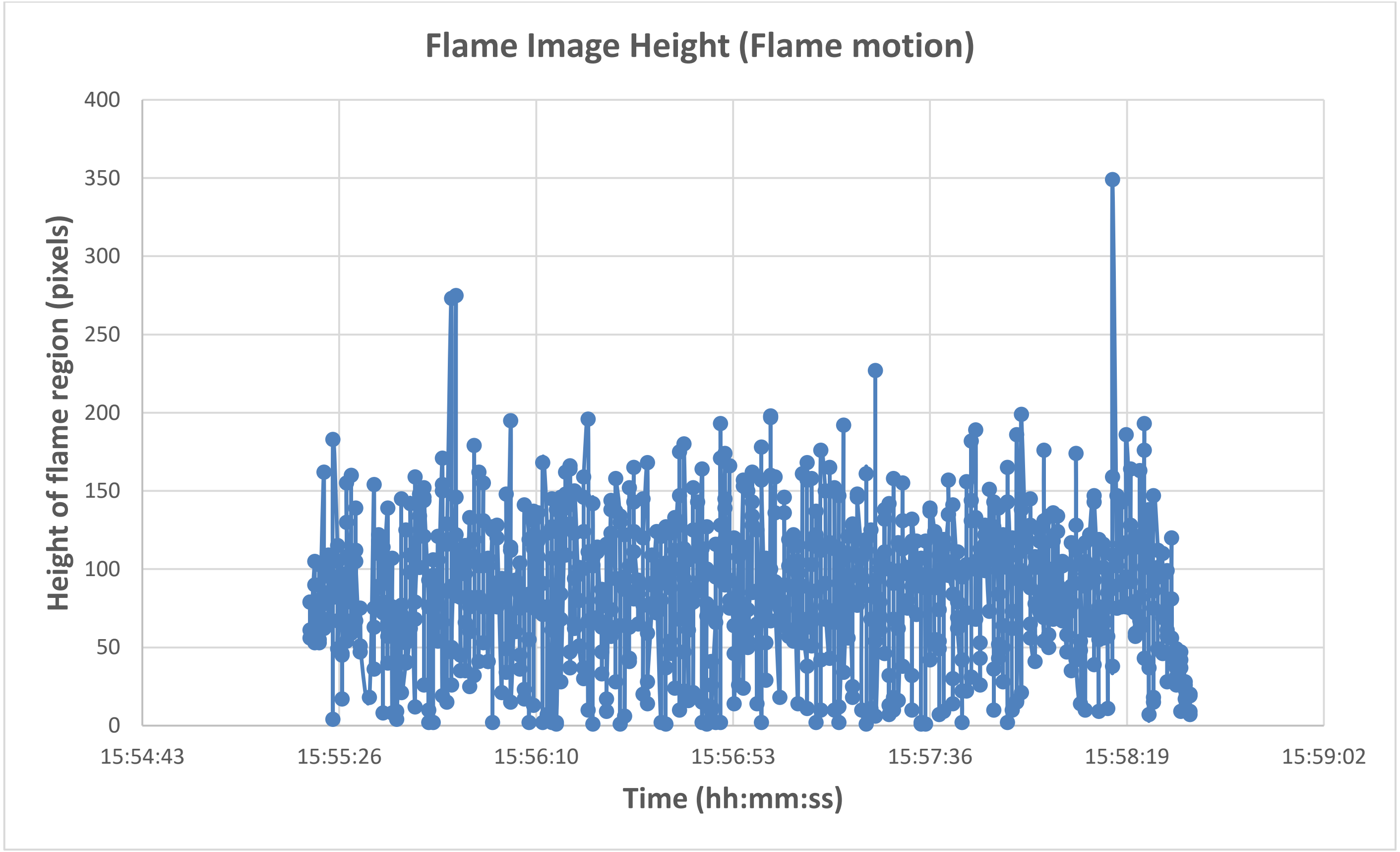


Figure 4.20 Flame Image Height (Flame motion) (20mL)

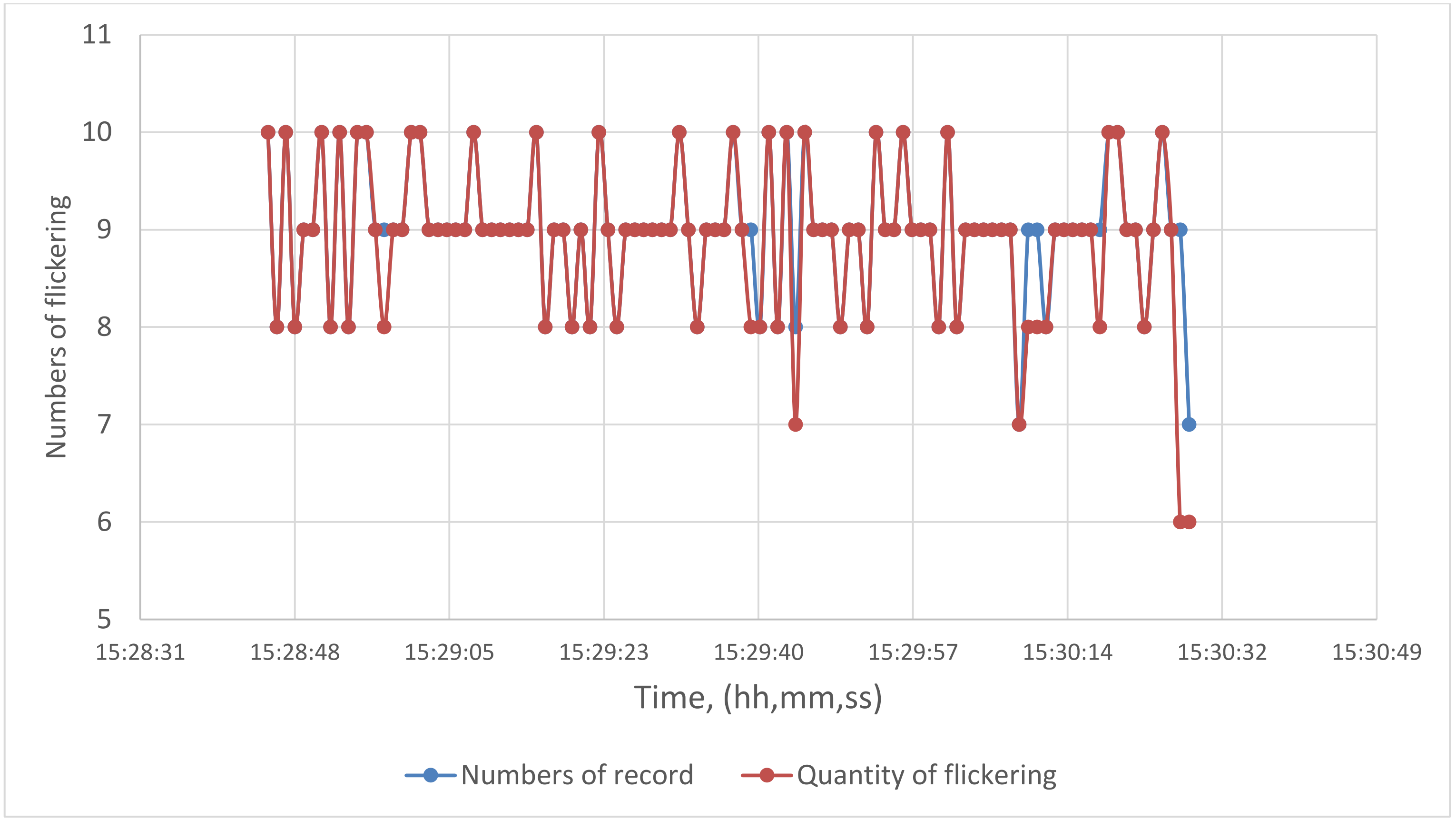


Figure 4.21 the change of flickering

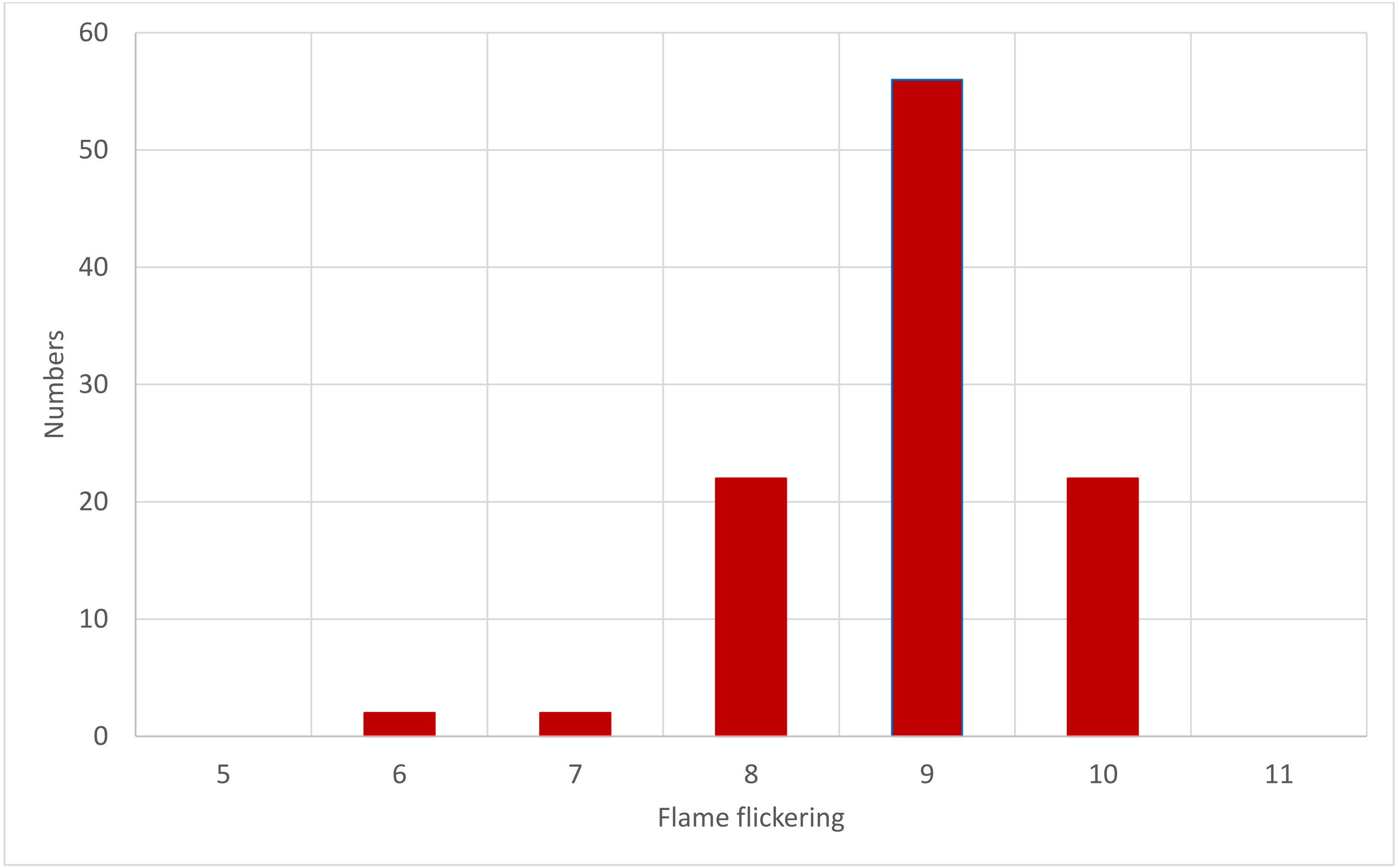
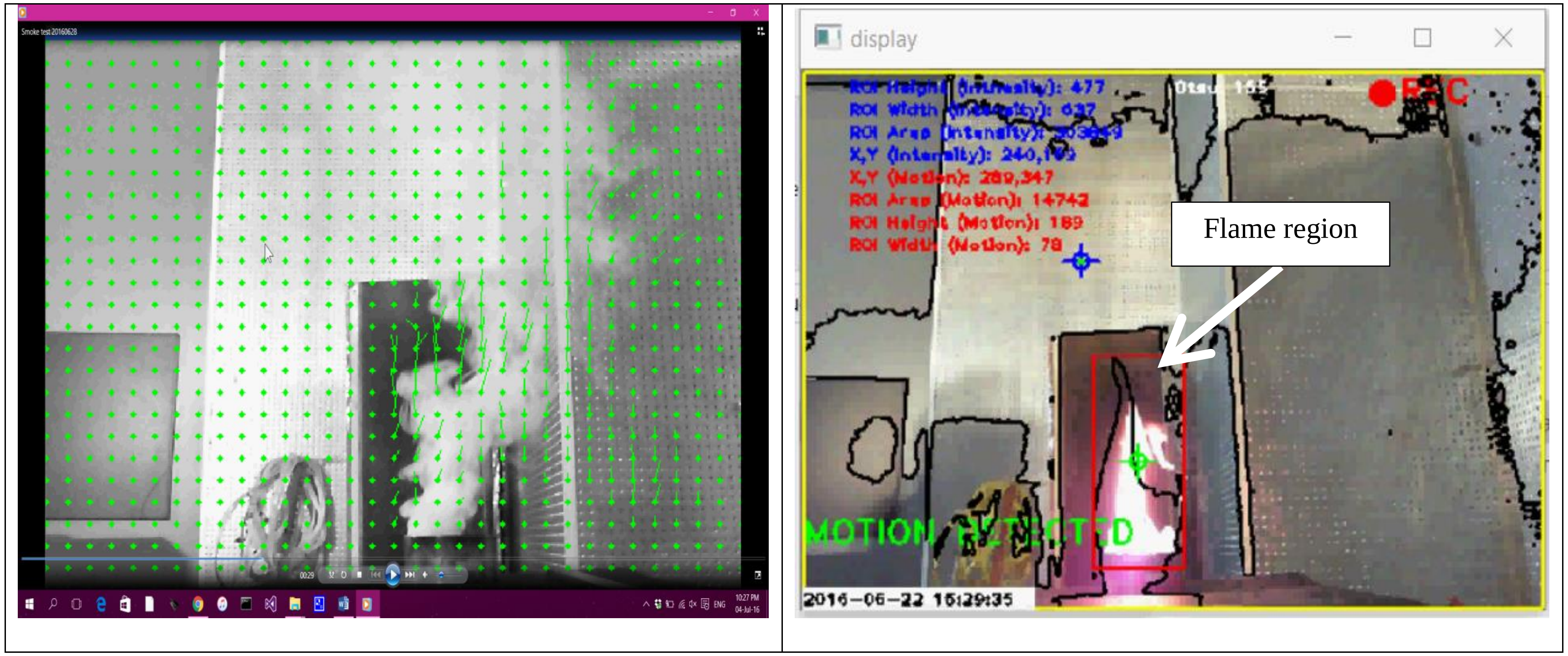


Figure 4.22 the histogram of flame flickering



```
ROI Height (Intensity): 477
ROI Width (Intensity): 637
ROI Area (Intensity): 303849
X,Y (Intensity): 240,169
X,Y (Motion): 289,347
ROI Area (Motion): 14742
ROI Height (Motion): 189
ROI Width (Motion): 78
```

Figure 4.23 Display of optical flow analysis (left) and motion detection (right)

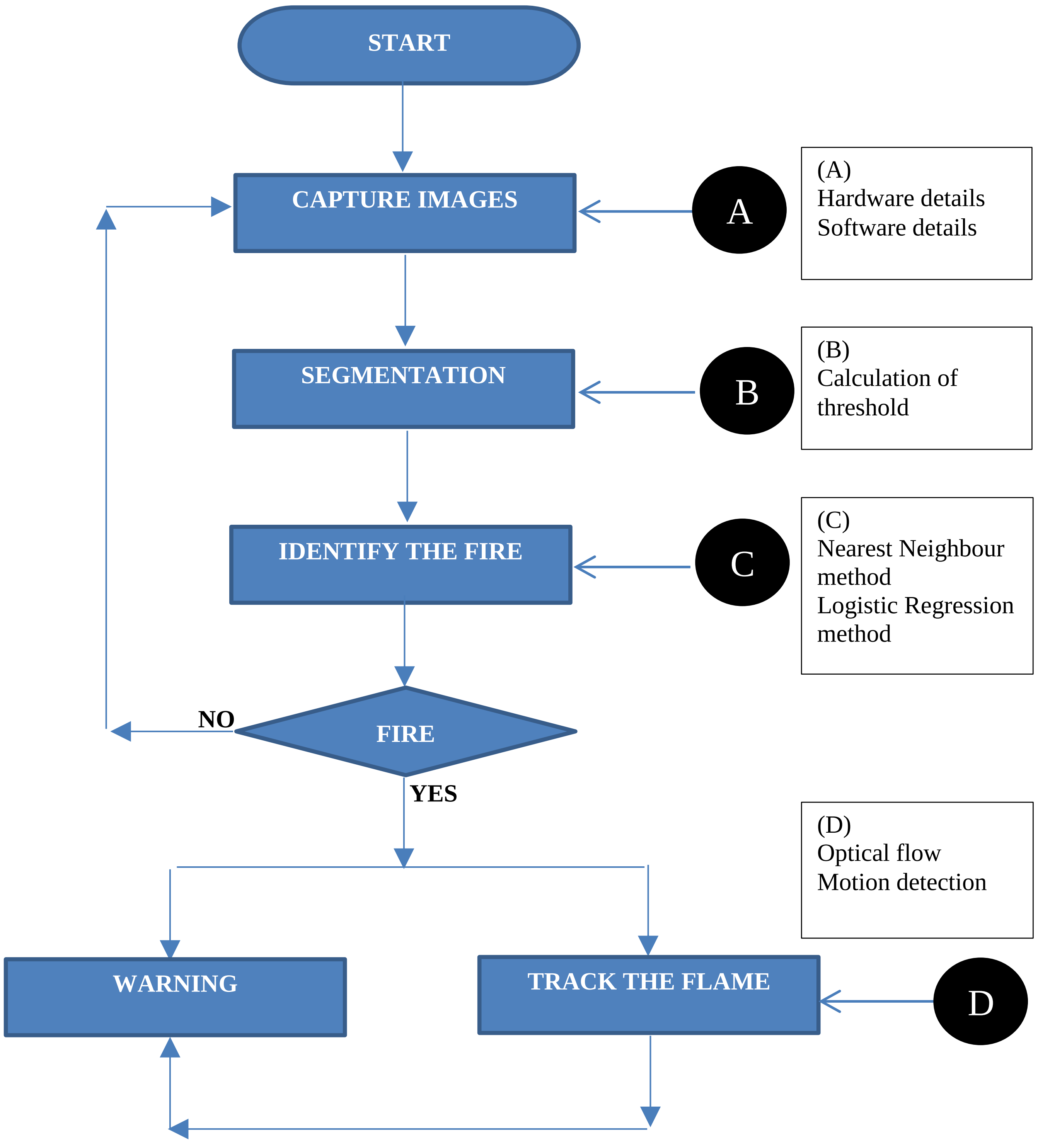


Figure 4.24 Flow diagram of video fire detection system

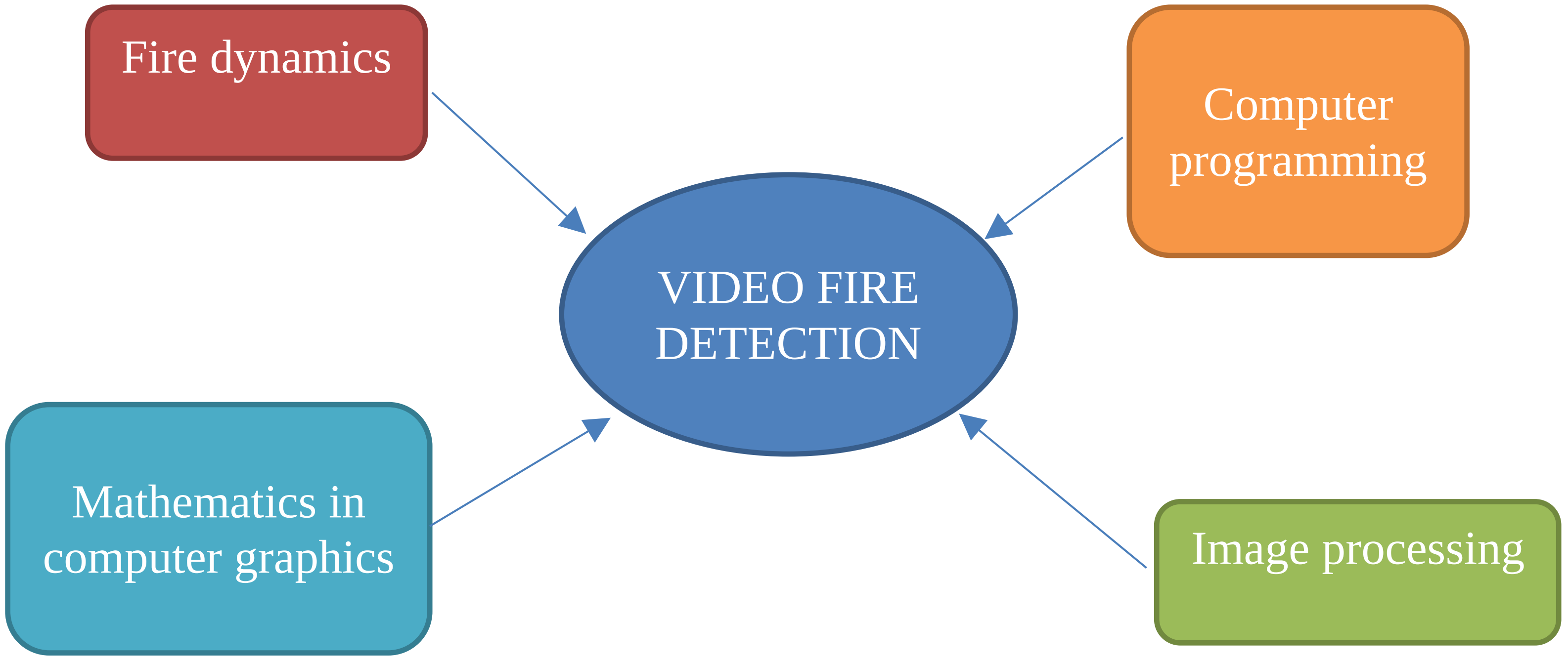


Figure 5.1 Different knowledge in video fire detection research

	Years	**Total Fire Calls**			**Unwanted alarms**			**False alarms**		
		Total	Average		Total	Average		Total	Average	
			Per month	Per day		Per month	Per day		Per month	Per day
1	**2015**	34,320	2,860	95	24,811	2,068	69	3,179	265	9
2	**2014**	36,335	3,028	101	26,765	2,230	74	3,206	267	9
3	**2013**	36,773	3,064	102	27,356	2,280	76	3,208	267	9
4	**2012**	37,638	3,137	105	28,461	2,372	79	2,995	250	8
5	**2011**	34,188	2,849	95	23,889	1,991	66	2,944	245	8
6	**2010**	40,604	3,384	113	30,710	2,559	85	3,108	259	9
7	**2009**	35,771	2,981	99	25,405	2,117	71	2,922	244	8
8	**2008**	35,513	2,959	99	24,007	2,001	67	3,296	275	9
9	**2007**	31,638	2,637	88	20,717	1,726	58	3,119	260	9
10	**2006**	33,268	2,772	92	21,846	1,821	61	3,302	275	9
11	**2005**	37,741	3,145	105	25,766	2,147	72	3,492	291	10
12	**2004**	35,092	2,924	97	21,744	1,812	60	3,425	285	10
13	**2003**	37,774	3,148	105	24,448	2,037	68	3,801	317	11
14	**2002**	41,204	3,434	114	27,548	2,296	77	4,131	344	11
Total		400,431	33,369	1,112	274,541	22,878	763	36,535	3,045	101

Table 1.1 Calculation the statistical results of total fire calls, unwanted alarms and false alarms (Hong Kong Fire Services Department, 2002 - 2015)

Video images			Still images		
Extension	**Format**	**Developed by**	**Extension**	**Format**	**Developed by**
MP4	MPEG-4 Part 14	ISO / IEC	WebP	---	Google
AVI	Audio Video Interleave	Microsoft	JPG	Joint Photographic Experts Group	ISO / IEC / ITU - T
3GP	Third Generation Partnership	3GPP	PNG	Portable Network Graphic	World Wide Web Consortium
RMVB	Real Media Variable Bitrate	Real Networks	ICO	Computer Icon	Microsoft
GIF	Graphics Interchange Format	CompuServe	BMP	Windows Bitmap	Microsoft
WMV	Windows Media Video	Microsoft	GIF	Graphics Interchange Format	CompuServe
MKV	Matroska Video	Matroska Media Container	TIF / TIFF	Tagged Image File Format	Aldus
MPG	Motion Picture Expert Group	ISO / IEC	PCX	Picture Exchange	ZSoft Corporation

Video images			Still images		
Extension	**Format**	**Developed by**	**Extension**	**Format**	**Developed by**
VOB	Video Object	DVD Forum	TGA	Truevision Graphics Adapter	Truevision
MOV	Quick Time Movie	Apple Inc.			
FLV	Flash Video	Adobe Systems			
SWF	Small Web Format	Adobe Systems			

Table 1.2 Popular still images and video file format and organisation

Standard Number	**Standard Topic**
NFPA 72	National Fire Alarm and Signaling Code
ANSI/FM 3260	American National Standard for Radiant Energy – Sensing Fire Detectors for Automatic Fire Alarm Signaling
BS 5839	Fire Detection and Fire Alarm Systems for Buildings
UL 268	Standard for Safety – Smoke Detector for Fire Alarm Systems

Table 1.3 The topics of different standards

Standard Number	**Summary**
NFPA 72	1. Video image flame detection and video image smoke detection are also described. 2. Quarterly inspection video image smoke and fire detectors are necessary. 3. Video image smoke and flame detectors is necessary to inspect, test and maintain following the manufacturer's instructions. 4. The location and spacing of video image smoke and flame detectors is following the principle of operation and engineering survey. 5. The video signal can transmit to other systems. 6. Video display require the alert and message. 7. Video image flame / smoke detection can analyse the images from images features including brightness, contrast, edge content, loss of detail and motion. The analysis method uses software – based method. 8. The protection of control and software requires the passwords and software keys or means of limiting access to authorized/qualified personnel. 9. Trouble signal requires when any change of component settings or ambient conditions affect the design performance of the video detector.

	10. Mass notification system requires including in fire alarm notification.
ANSI/FM 3260	Test methods and practices are also referenced in NFPA 72
BS 5839	1. New video smoke detection is taken into account. 2. Video techniques are used to detect the smoke. 3. Closed circuit television cameras monitor the protected space. 4. Detection can use in normal lighting environments and also use in infra – red light sources environments.
UL 268	1. The outline of video smoke detection describes. 2. Normal operation and fire test are also analysis. 3. Normal operation includes operation test, electrical supervision test, component failure, stability test, circuit measure test, overvoltage and under voltage tests, temperature test, vibration test, Jarring test, Variable ambient temperature test, etc. 4. Fire test includes paper fire test, wood fire test, smouldering smoke test, smouldering smoke test – analysis maximum obscuration without alarm,

Table 1.4 Summary of different standard relative video fire detection

	Visible name	**Wavelength λ (nm)**
1	Visible Red	≈700
2	Visible Green	≈546.1
3	Visible Blue	≈435.8

Table 1.5 Visible band of electromagnetic spectrum

Colour	**Function values**		
	Red	**Green**	**Blue**
Red	255	0	0
Green	0	255	0
Blue	0	0	255
White	255	255	255
Black	0	0	0
Yellow	255	255	0
Cyan	0	255	255
Magenta	255	0	255

Table 1.6 Common colours in RGB function

Environment	**Hue**	**Saturation**	**Intensity**
Brighter	0^o-60^o	40-100	127-255
Darker	0^o-60^o	20-100	100-255

Table 2.1 colour feature of low temperature fire flames (Horng, Peng , & Chen, 2005)

Red level	Ratio > 2.0%
Orange level	1.0% < Ratio ≤ 2.0%
Yellow level	0.5% < Ratio ≤ 1.0%
Blue level	0% < Ratio ≤ 0.5%

Table 2.2 Ratio distribution of Blue, Yellow, Orange and Red

Fire	Height in pixels
A gasoline	58
Acetone	51
Alcohol	24

Table 2.3 Measured flame heights in pixels (Maoult, Sentenac, Orteu, & Arcens, 2007)

Flame numbers	#1	#2	#3	#4	#5	#6
Height in pixels	112	159	148	164	218	256
Real flame height (m)	0.2489	0.3533	0.3289	0.3644	0.4844	0.5689
Flame numbers	#7	#8	#9	#10	#11	#12
Height in pixels	188	222	243	198	206	210
Real flame height (m)	0.4178	0.4933	0.5400	0.4400	0.4578	0.4667

Table 2.4 Number of flame pixels and real flame height (Jianzhong, Jian, Jian, & Jun, 2010)

Pool size	**Pagni's method**	f_d **Frequency of Experimental study**	f_z **Frequency of Zukoski calculation**
Bigger	3.5746	(Flue is heptane) Average 3.1875 Average 3.0950	(Flue is heptane) Average 3.1562 Average 3.1070
Smaller	4.7958	(Flue is gasoline) Average 4.5937 Average 4.5065	(Flue is gasoline) Average 4.8425 Average 4.6270

Table 2.5 Experimental results of oscillation frequency (Jianzhong, Jian, Jian, & Jun, 2010)

Pool diameter (m)	**Frequency (Hz)**		
	Hefei	Lhasa	Pagni’s method
W x L = 0.27 x 0.27	2.48, 2.24,	2.44, 2.63	2.75
D = 0.18	3.53, 3.43,	3.34, 3.38	3.57

Table 2.6 Flame oscillation in two different pool size (Juan & Qifu, 2012)

	Research countries	Qty.	Years
1	Spain	7	1991, 1995, 1997(2), 1999, 2003, 2011
2	Brazil	1	1993
3	China	4	1998, 2008, 2009, 2011
4	Canada	1	2001
5	France	1	2005
6	Croatia	1	2009
7	India	2	2012, 2016
8	Mexico	1	2014

Table 2.7 The countries of research of video fire detection for protection forest environment

Class no.	Colour	NDVI value
Class 1	Blue	-1.0 – 0.0
Class 2	Red	0.0 – 0.05
Class 3	Black	0.05 – 0.10
Class 4	Dark green	0.10 – 0.15
Class 5	Green	0.15 – 0.20
Class 6	Light green	0.20 – 0.25
Class 7	Yellow	0.25 – 0.30
Class 8	White	0.30 – 1.0

Table 2.8 Summary of colour relative HDVI value

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
1994	S. Noda, K. Ueda	✓		RGB colour model			
1996	Simon Y. Foo	✓			Median, Standard Deviation and First – order moment statistical measures of the histogram data		Aircraft dry bays and engine compartments
1999	Song Weiguo, Fan Weicheng, Wu Longbiao	✓				Back Propagation	
1999	Hideaki Yamagishi, Jun'ichi Yamaguchi	✓		HSV colour model		Back propagation	
1999	Xiaofang Cheng, Jianhua Wu, Xin Yuan, Hao Zhou	✓		Sensitivity of the CCD camera The relationship between lower illuminance and Planck's Law			
2000	Ivan Kopilovic, Balazs Vagvolgyi and Tamas Sziranyi		✓		Tracking information History of motion Detection the special motion of smoke		Panoramic Annular Lens
2000	Walter Phillips III, Mubarak Shah, Niels da Vitoria Lobo	✓			Gaussian – smoothed colour histogram define the fire colour pixels Using the temporal variation of pixels define the actual fire pixels.		Recognition the fire (High, low temporal variation)

Table 2.9 Development of video fire detection method from 1994-2000

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
2001	Yang Lizhong, Deng Zhihua, Fan Weicheng and Wang Qing'an	✓		Pixels analysis in Greyscale images			Liquid Crystal - Light Valve
2003	Thou - Ho (Chao - Ho) Chen, Cheng - Kiang Kao and Sju - Mo Chang	✓		RGB and HSI colour model	Statistical parameter (Average and Variance)		
2004	Tao Chen, Hongyong Yuan, Guofeng Su, Weicheng Fan	✓		Computer vision theory			Controlled the fire suppression system
2004	Lawrence S.M. Chiu		✓	Grey level method	Histogram method (Standard deviation, Mean and Maximum)		
2004	Wolfgang Krüll, Ingolf Willms and Jeff Shirer	✓			IMAGE STATISTICS to Cargo fire verification control unit		Cargo fire verification system
2004	Li Jin, Fong N. K., Chow W. K., Wong L. T., Lu Puyi and Xu Dian - guo	✓			Nyquist sampling theorem		Moment features / Flickering frequency
2004	Che-Bin Liu and Narendra Ahuja	✓			(1) SEED REGION - A mixture of Gaussian distributions in HSV space (2) Using Fourier Descriptors (FD) discrimination the flame shape (3) Autoregressive model (analysis the temporal variation of shape)		Colour probability density (1) Spectral model (2) Spatial model (3) Temporal model

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
2005	Wen-Bing Homg, Jim-Wen Peng, and Chih-Yuan Chen	✓		RGB and HSI colour model			Building a flame feature model
2005	B. Ugur Toreyin, Yigithan Dedeoglu, A. Enis Cetin	✓		**Flame Chrominance Model -** Defining the flame pixels	**"Hidden Markov Model" –** Both temporal and spatial analysis of flame and non – flame pixels		
2005	CHENG Xin, WANG Da - chuan, YIN Dong - liang	✓			Colour analysis		Used MATLAB
2005	Yigithan Dedeoglu, B. Ugur Toreyin, Ugur Gudukbay, A. Enis Cetin	✓		Wavelet analysis			Combined all of the clue reach a final decision.
2005	Florent Lafarge, Xavier Descombes and Josiane Zerubia	✓				Support Vector Machine (SVM) Classification	Kernel based on a flame textural information analysis Forest fire detection
2005	Thorsten Schultze and Ingolf Willms	✓		Spatial resolution of light scattering measurements			Used microscope
2005	Feiniu Yuan, Guangxuan Liao, Weicheng Fan, and Heqin Zhou	✓			Mixture Gaussian Model		Colour and temporal features
2006	Jinhua Zhang, Jian Zhuang, Haifeng Du, Sun'an Wang, and Xiaohu Li	✓			Flame detection algorithm: (Multi - feature fusion) Static flame feature: Probability model Diffusion flame feature:		Analysis the features of static flame and diffusion flame

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
2006	Giuseppe Marbach, Markus Loepfe, Thomas Brupbacher	✓		YUV colour model			Feature extraction (1) Flickering (2) Reaching maximal luminance
2006	T. Ono, H. Ishii, K. Kawamura, H. Miura, E. Momma, T. Fujisawa, J. Hozumi	✓		RGB colour model Analysis the RG colour obtain the threshold		Hierarchical type Neural Network	Car fire analysis
2006	Thorsten Schultze, Thorsten Kempka, Ingolf Willms	✓		Mean value of the brightness in each video frame			Flickering and flow movement
2006	T. Celik, H. Demirel, H. Ozkaramanli	✓			Probability method		
2006	Jeffrey C. Owrutsky, Daniel A. Steinhurst, Christian P. Minor, Susan L. Rose - Pehrsson, Frederick W. Williams, Daniel T. Gottuk	✓		Long wavelength video image - based detection			
2006	Thou-Ho (Chao-Ho) Chen, Yen-Hui Yin, Shi-Feng Huang and Yan-Ting Ye		✓	1. Chromaticity - based statistic decision rule 2. Diffusion - based dynamic characteristic decision rule			
2006	D.T. Gottuk, J.A. Lynch, S.L. Rose-Pehrsson, J.C.	✓					Evaluate the effectiveness of commercial video

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
	Owrutsky, F.W. Williams						image fire detection in shipboard
2007	C. L. Lai, J. C. Yang, and Y. H. Chen	✓	✓	Spatial - temporal, spectra variation Colour / Greyscale histogram concentration			
2007	Turgay Çelik, Hüseyin Özkaramanlı, and Hasan Demirel	✓	✓	YCbCr Colour model for fire detection RGB Colour model for smoke detection			
2007	Y. Le Maoult, T. Sentenac, J. J. Orteu and J. P. Arcens	✓					based on a low-cost CCD camera to detect a fire in the near infrared spectral band
2007	Byoungmoo Lee and Dongil Han	✓	✓	Determined threshold of RGB colour			Tunnel fire analysis
2008	Feiniu Yuan		✓	Determined the threshold based on the chrominance detection			
2008	Chin-Lun Lai, Jie-Ci Yang	✓	✓	Colour information	Otsu method		
2008	Yu Cui, Hua Dong, Enze Zhou		✓	Wavelet analysis Grey level Co-occurrence Matrices (GLCM)		For discrimination	Smoke texture analysis
2008	Zhong Zhang, Jianhui Zhao, Dengyi Zhang, Chengzhang Qu,	✓		Combine both wavelet and Fourier analysis			Forest fire detection

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
	Youwang Ke, Bo Cai						
2008	Paulo Vinicius Koerich Borges, Joceli Mayer, Ebroul Izquierdo	✓			Statistical characteristic of fire features		Skewness, texture, boundary roughness, colour, randomness of area size
2008	Bo-Ho Cho , Jong-Wook Bae , and Sung-Hwan Jung	✓		HSI colour model and RGB colour model	Statistical colour model and Binary background mask		
2009	Shuenn-Jyi Wang, Dah-Lih Jeng, Meng-Tsai Tsai	✓	✓	Discriminated the region characteristics - the shape of the changed region	Fuzzy c - mean clustering algorithm (Dominant flame colour lookup table) – By comparing the pixels of changed regions		New generation vessels
2009	Jie Hou, Jiaru Qian, Zuozhou Zhao, Peng Pan, Weijing Zhang	✓			Probability density (PD) algorithm	Fuzzy Neural Network (FNN), Combined PD, FNN, Dempster - Shafer rule and Historical data fusion	High and Large - Span Space Structures
2009	Byoung Chul Ko , Kwang - Ho Cheong, Jae - Yeal Nam	✓		1. Using the frame difference detection moving pixels 2. Flickering frequency	RGB colour probability models		1. Proposed robust fire detection algorithm that is installed in home network server 2. Detection of moving regions 3. Fire - coloured pixels
2009	Turgay Celik, Hasan Demirel	✓		YCbCr Colour model			Forest fire detection

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
2009	Dongil Han, Byoungmoo Lee	✓	✓	**Flame detection algorithm** 1. Colour information - Intensity of images 2. Erosion and Dilation remove the nose **Smoke detection algorithm** Numerical equation for extract the images Used Motion History Images (MHI) method			Experimental study for tunnel fire
2009	Damir Krstinić, Darko Stipaničev, Toni Jakovčević		✓	Segmentation the smoke region from forest fire images Evaluated several colour spaces including RGB, HIS HS'I, YCrCb, and CIELab	1. Simple Lookup Table Method 2. Probabilistic model for classification to classify the pixels into the smoke class or non - smoke class		Forest fire detection
2009	Dengyi Zhan, Shizhong Han, Jianhui Zhao, Zhong Zhang, Chengzhang Qu, Youwang Ke, Xiang Chen	✓		Segmentation based on HSV colour model Extraction the fire features		BP Neural Networks	Real time forest fire detection
2009	Ignacio Bosch, Soledad Gomez, Raquel Molina, and Ramon Miralles	✓		Segmentation - thresholding technique Feature extraction - Intensity, signature and orientation			Infrared images processing Forest fire surveillance and preservation of natural heritage

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
2009	Dong Keun Kim and Yuan-Fang Wang		✓	Block - based approach YUV colour model	**Classification of Smoke (k - temporal information)** Area, boundary rectangle, The average and standard deviation of Y - value and UV - value		Proposed three steps for smoke detection in outdoor video sequences.
2009	Yu Chunyu, Zhang Yongming, Fang Jun, Wang Jinjun		✓	Real - time detection and divided the image in block	Gaussian Mixture Model (GMM) segment the background and foreground Grey level co - occurrence matrices analyse the smoke texture	Neural Network Classifier Back Propagation Neural Network discriminate the smoke feature	Texture analysis Feature extraction and Feature Classification
2009	Ha Dai Duong, Nguyen Anh Tuan	✓			Real - time detection the fire by Bayes method Fuzzy C - Means (FCM) algorithm is group pixels into clusters and retrieve Dominant flame colour look up table (DFCLT)		
2010	Ishita Chakraborty, Tanoy Kr. Paul	✓		RGB and HSI Histogram Obtained the thresholding by HSI	Hierarchical Clustering Partition Clustering		
2010	Paulo Vinicius Koerich Borges, Ebroul Izquierdo	✓			Probabilistic Approach **Statistical characteristics of fire** Colour, Randomness of Area Size, Boundary Roughness, Surface Coarseness, Skewness, Spatial Distribution of Fire		
2010	Feiniu Yuan	✓	✓	Motion and colour detection for smoke detection	Probability of the incoming RGB colour pixel values for flame detection		An integrated fire detection and suppression system

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
2010	Rong Jianzhong, Wang Jian, Chen Jian, Jiang Jun	✓		Acoustic measurement technology calculate the oscillation frequency of flame			
2010	Byoung Chul Ko, Kwang - Ho Cheong, Jae - Yeal Nam	✓			Separate foreground from background - Threshold analysis Detection of fire coloured pixels (Colour probability model - Gaussian probability distribution) Three layers hierarchical Bayesian Networks		
2010	Turgay Celik	✓		RGB colour model convert to CIE L*a*b* colour space Using CIE L*a*b* colour space identify the fire pixels.			
2010	Magy Kandil, May Salama, Samia Rashad	✓				Back Propagation feed forward neural networks	
2010	Li Ma, Kaihua Wu, L. Zhu		✓	Extract continuous motion regions (Combined Kalman filtering and Moving History Image) Colour variation describing (colour blending coefficients) Dynamic feature verification (wavelet analysis)	Delete some moving objects dissimilar to the smoke colour model (Offline trained Gaussian mixture model in RGB colour space)		

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
2010	Jing Huang, Jianhui Zhao, Weiwei Gao, Chengjiang Long, Lu Xiong, Zhiyong Yuan, Shizhong Han	✓			Local Binary Pattern Based (LBP) operator filtering the object The texture analysis is Mahalanobis distance classifier		Texture analysis
2010	Juan Chen, Yaping He, Jian Wang	✓		Detected moving objects categorized flame region (flame colour)	Extract moving foreground objects with improved Gaussian mixture model method Statistical frequency counting distinguish true flame		Multi feature fusion
2010	Yu Chunyu, Fang Jun, Wang Jinjun and Zhang Yongming		✓	Extracted moving pixels and regions Colour - Based Decision Rule for smoke recognition	Optical flow method	Back - propagation Neural Networks discriminate the smoke features	
2010	鄭文昌, 陳政達	✓		RGB colour space transform to YCbCr YCbCr colour space analysis Flame flickering detection			

Table 2.10 Development of video fire detection method from 2001-2010

Years	**Researcher**	**Fire / Flame detection**	**Smoke detection**	**Image processing**	**Statistical analysis**	**Artificial Neural Network / Neural Network**	**Remarks**
2011	Audrey Chenebert, Toby P. Breckon and Anna Gaszczak	✓			Grey Level Co – occurrence Grey Matrix (GLCM)		Flame texture and colour feature analysis for classification
2011	Truong Xuan Tung, Jong-Myon Kim		✓		Fuzzy C – Mean method (FCM) and Support Vector Machine (SVM) distinguish the smoke features.		Smoke features : Colour: low temperature (Blush – white to white) ; temperature rise until fire ignites (grey black to black) Movement : Drifting upward and diffuse Area, size and number of smoke region are varied and change from frame to frame Surface and boundary are rough and coarse.
2011	Luis Merino, Fernando Caballero, J. Ramiro Martínez-de-Dios, Iván Maza, Aníbal Ollero	✓		Image processing for extraction by fire features	Perception system and decision system Analysis the probability values		Application the Unmanned Aircraft System (UAS) monitoring and measurement the forest fire
2011	Yusuf Hakan Habiboˇglu, Osman Günay and A, Enis Çetin	✓		Detection Algorithm: 1. Chromatic colour model – Analysis the RGB colour	Support Vector Machine (SVM) extracted the features including the RGB colour and Intensity		Using the colour, spatial and temporal information in video fire detection

Years	**Researcher**	**Fire / Flame detection**	**Smoke detection**	**Image processing**	**Statistical analysis**	**Artificial Neural Network / Neural Network**	**Remarks**
				2. Covariance matric method for object detection and texture classification			
2011	Li Jinghong, Lv Riqing, Zou Xiaohui	✓		Forest transform operation Sobel module – Edge detection	Otsu method		Field Programmable Gate Array (FPGA): The module of image collection The module of fire detection The module of image display
2011	Yang Zhao, Jianhui Zhao*, Erqian Dong, Bingyu Chen, Jun Chen, Zhiyong Yuan, and Dengyi Zhang	✓		Dynamic texture model: Multi – Resolution Analysis Linear Dynamic System (Threshold)			Fire Recognition by texture analysis
2011	Tung Xuan Truong, Yongmin Kim, Jongmyon Kim	✓		Discrete wavelet transform	Approximate median method Fuzzy c – mean algorithm	Genetic – Based Back – Propagation Neural Networks (BPNN)	FIRE FEATURES Appropriate conditions Reacting oxygen from air Generating combustion products Emitting light Release heat Colour of fire range from red to yellow and white

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
							The size of area in fire regions changed The surface and boundary of flame is rough and coarse
2011	Yongqiang Jiao, Jonathan Weir, WeiQi Yan	✓		Intensity histogram RGB and HSI colour model **Texture analysis –** Coarseness Contrast Directionality Line – likeness Regularity Roughness			Video content analysis technology monitoring Analysis the flame region by texture
2011	Yusuf Hakan Habiboglu, Osman Gunay, A. Enis Cetin	✓		Chromatic Colour model - RGB and HSI colour model	Covariance matrix computation method - RGB colour values and First and second derivatives of intensity with respect to time		
2011	CHENG Caixia (程彩霞), SUN Fuchun (孙富春), ZHOU Xinquan (周心权)	✓				Neural network: Radial basis function (RBF) network Temperature CO concentration Smoke density	
2011	Jianhui Zhao, Zhong Zhang, Shizhong Han, Chengzhang Qu, Zhiyong Yuan, and Dengyi Zhang	✓			Segmentation the possible flame regions with: 3D colour model Gaussian mixture model		Forest fire detection Statistic features including: Colour distribution (11 features); Texture

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
					Support Vector Machine (SVM) classifier trains and filters the segmented results		parameter (5 features); Shape roundness (1 feature) Dynamic features including: (variation of colour distribution, texture parameter, shape roundness, area, contour, Flickering frequency
2011	Feiniu Yuan		✓		Local binary pattern (LBP) is Grey scale texture operator. It can capture the spatial characteristic of images. Local binary patterns variance (LBPV) Histogram of LPB and LPBV pyramids enhance the feature vector	BP neural network classifier is used for discrimination the smoke and non - smoke objects	Texture analysis method is effetely detection the smoke in images
2011	Simone Calderara, Paolo Piccinini, Rita Cucchiara		✓	By means of wavelet transform coefficient analyse the image energy	Bayesian approach classify the candidate objects		Detecting the moving objects
2011	Changwoo Ha, Gwanggil Jeon, and Jechang Jeong		✓	RGB colour space – Chromatic analysis	Motion information – Moving Region Decision (MRD) is based on calculation the Motion Vector (MV) by upward characteristic of smoke spread		Smoke detection algorithm – Block – based smoke detection algorithm – three basic steps

Years	**Researcher**	**Fire / Flame detection**	**Smoke detection**	**Image processing**	**Statistical analysis**	**Artificial Neural Network / Neural Network**	**Remarks**
2012	Ha Dai Duong and Dao Thanh Tinh	✓			Combined algorithm detection the fire by average of RGB channel, coarseness and skewness of red channel distribution Bayes classifier, evaluation and classification the features		Features : colour, geometry and motion
2012	Li Hongliang, Liu Qing,Wang Sun'an	✓			Probability model of fire recognition algorithm based on multi – features fusion		Dynamic feature: Intensity Sequence Area variation Circularity
2012	Leonardo Millan-Garcia, Gabriel Sanchez-Perez, Mariko Nakano, Karina Toscano-Medina, Hector Perez-Meana, Luis Rojas-Cardenas		✓	Morphological operators reduce the noise Detection algorithm is Discrete Cosine Transform (DCT) Smoke motion and smoke colour analysis is proposed.			Video detection, IP cameras platform is used RGB and YCbCr colour model are used MPEG domain is used
2012	CHEN Juan, BAO Qifu	✓		Fire feature is Flame colour and oscillation frequency (flickering) Discrete time Fourier transform (DTFC) analyse the fire features.			Rounded oil pool fire and squared oil pool fire
2012	CHEN Ning, DING Fei	✓		**Flame segmentation** Pixel analysis is proposed for evaluation the threshold value			Fisheye camera are used for detection fire

Years	**Researcher**	**Fire / Flame detection**	**Smoke detection**	**Image processing**	**Statistical analysis**	**Artificial Neural Network / Neural Network**	**Remarks**
2012	Vipin V	✓		RGB and YCbCr colour space pixels analysis Calculation the mean value of RGB and YCbCr Histogram analysis			The processing of segmentation requires to satisfy the Rule 1 to 7.
2012	Yong XU, Xingjie ZHU, Binglei XIE	✓		First step: Pixel analysis (RGB colour space analysis), fire processing is able to detect the genuine fire region	Using adjustable KNN classifier		
2012	Tarek Beji and Bart Merci, Steven Verstockt and Rik Van de Walle	✓	✓	Smoke layer height estimation exploits the energy – related characteristics of smoke. Flame detection algorithm exploits the brightness value and threshold detection the flame. Estimation the fire size exploits the two – zone fire model (CFAST)			Forecast fire growth in enclosures
2012	Chen – Yu Lee, Chin – Teng Lin, Chao – Ting Hong, Miin – Tsair Su		✓	Feature extraction includes the wavelet analysis, energy analysis and chromatic analysis. Verification is evaluation the images in a predefined time interval. If over 50% is required an alarm then sends out a real fire alarm signal.	Support Vector Machine classifier proposes to use for classification the candidate region.		

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
2012	Zhigang Liu, George Hadjisophocleous, Guofeng Ding and Choon Siong Lim	✓	✓		The second step: Smoke and flame probability	The first step: Computing the features of smoke and flame with Back propagation neural networks (BPNN)	Captured the Colour and Black and White Images and also use the Infrared Image
2013	Chunyu Yu, Zhibin Mei, Xi Zhang	✓	✓	HIS colour model analysis for smoke feature. RGB colour model analysis for flame feature. Find the coordinates of the centre points calculate with Pyramidal Lucas – Kanade feature optical flow vector		Back – propagation neural network for smoke recognition	Feature analysis of fire flame and smoke Reddish colour The frequency of flame flickering
2013	De-chang Wang,XuenanCui,EunsooPark,ChanglongJin,HakilKim				Determined the probabilities (Colour and motion features) extract the candidate flame region Wald – Wolfowitz randomness test determine the flame probability		Feature extraction by colour and motion
2013	Wanzhong Lei and Jingbo Liu	✓		Protection fire region detection is analysis the pixel in frame differencing. YCbCr and RGB colour space extract the flame region	Median filtering algorithm is able to remove the noise Bayer classifier recognize the fire features		Fire detection is able to use in Coalmine
2013	Divina A. Chua Carla Louie H. Leandicho Leo Angelo C. Magtibay				Shape and colour analysis in statistic colour model are used		Application in Mobile Application (app.) in Android Operating System and Internet

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
	Jerome T. Ortiz						Protocol (IP) camera is required
2013	LI Xiao-bai, HUA Ying, XIA Ning	✓		Optical flow method is able to use in RGB colour space analysis			Analysis of dynamic textures
2013	Ti Nguyen-Ti, Thuan Nguyen-Phuc, Tuan Do-Hong	✓		RGB colour space model and YCbCr colour space model are used for analysis fire pixels Motion filter, Colour filter and Position filter use for analysis the region growth To increase reliability, Ratio height / width and correction coefficient are used for classification the fire from images.			Total 9 rules detect the fire Colour and motion model is proposed to use.
2013	Jianzhong Rong, Dechuang Zhou, Wei Yao, Wei Gao, Juan Chen, Jian Wang	✓		Generic colour model	Geometrical Independent Component Analysis Model (GICA model) and Cumulative Geometric Independent Component Analysis Model are used for detection the moving region.	BP neural network fire recognition model	Filter the fire object

Years	Researcher	Fire / Flame detection	Smoke detection	Image processing	Statistical analysis	Artificial Neural Network / Neural Network	Remarks
2013	Panagiotis Barmpoutis, Kosmas Dimitropoulos, Nikos Grammalidis	✓		Temporal processing use for flickering analysis Analysis the Spatial and temporal consistency include the smoothness consistence and data consistency	Adaptive median algorithm uses for subtraction the background. Fire colour probability is used for filtration the non – fire moving pixels		
2013	N. Brovko, R. Bogush, S. Ablameyko		✓	Three steps of frame processing : Greyscale transformation, Histogram equalization, Discrete wavelet of current input frame	Contrast calculated with Weber formula.		Motion and contrast two key features for smoke detection
2013	Hongcheng Wang, Alan Finn, Ozgur Erdinc, Antonio Vincitore	✓		Pixel – level processing: Identify the potential fire pixels with motion and intensity and grouped into blobs. Spatial – Temporal structure features Spatial – Temporal contour dynamic features	Classifier: SVM classifier		Spatial – temporal structural Dynamics features
2014	Roberto Rosas-Romero		✓		Feature extraction include the intensity colour values, mean of the intensity value and variance of the intensities on each colour plane. Classification the forest fire K – Singular Value Decomposition (K – SVD) method		Spatial – temporal interaction Wireless camera networks Forest fire monitoring system

Years	**Researcher**	**Fire / Flame detection**	**Smoke detection**	**Image processing**	**Statistical analysis**	**Artificial Neural Network / Neural Network**	**Remarks**
2014	Yoon-Ho Kim, Alla Kim, and Hwa-Young Jeong	✓		**RGB colour model is used** 2.Colour detection of moving pixels 3.Blob analysis	1.Detection of moving pixels or regions in the current frame of a video		Wireless sensor networks
2015	Yang Jia, Jie Yuan, Jinjun Wang, Jun Fang and Yongming Zhang, Qixing Zhang		✓	Optical flow analysis with Horn–Schunck algorithm based on the image brightness so-called Motion Map Calculation	Detection the Saliency Smoke Region with human vision system based on probability		Saliency – based method Forest fire detection
2015	Sumei He, Xiaoning Yang, Sitong Zeng, Jinhua Ye, Haibin Wu	✓		Calculation the average of luminance information with fire colour pixel based on Colour clues - YCbCr colour space	Fire detection employ the Statistical analysis based on the flame motion feature (flame flickering)		Real – time fire detection Computer vision based
2015	Kosmas Dimitropoulos, Panagiotis Barmpoutis, and Nikos Grammalidis	✓			First step1: filter out the non-fire coloured moving regions by probability based on RGB colour space model **Discriminating the fire and non - fire moving object:** Spatio – temporal consistency energy Flickering energy Spatio – temporal energy		Spatio – Temporal Flame Model Dynamic Texture Analysis

Table 2.11 Development of video fire detection method from 2011-2015

	Software Tools	Image types
Stage 1	1. MATLAB image processing toolbox	*Still Images* Colour Images
Stage 2	1. Visual C++ 2. Microsoft Foundation Class (MFC)	*Still Images* Thermal Images Colour Images
Stage 3	1. C++ 2. Open CV	Real time video images

Table 4.1 Summary of three stage of experimental study

Sequences of calculation	Threshold values
1	128
2	208
3	238
4	248
5	252
6	253

Table 4.2 Multi threshold calculation results

Sequences of calculation	**Threshold values** **(By Otsu)**	**Threshold values** **(By Rayleigh distribution)**
1	128	169
2	208	212
3	238	238
4	248	248
5	252	252
6	253	253

Table 4.3 Numerical data of two different procedures in threshold values

Fuel		**2 – Propanol**			
Pool diameter	***mm***	102	197	330	410
Fuel	***mL***	100	800	800	800
Mass loss rate	***g/s***	0.1264919	0.5157317	1.6350179	2.0463882
Molar coefficient	***kJ/mol***	2220	2220	2220	2220
Molar weight	***g/mol***	60.1	60.1	60.1	60.1
Heat of combustion	***kJ/g***	36.938436	36.938436	36.938436	36.938436
Combustion efficiency		0.7	0.7	0.7	0.7
Heat Release Rate (Based on Mass Loss Rate)	***kW***	3.2706883	13.335226	42.276503	52.913266
Number of images	***nos.***	30	30	30	30
Mean flame height	***mm***	183.63	283.80	462.16	667.80
Pool diameter	***m***	0.102	0.197	0.33	0.41
Revised Heat Release Rate (Based on images results)	***kW***	1.6579322	6.1108622	21.299584	45.909748

Table 4.4 Numerical data of calculation the heat release rate

Product Name	Microsoft LifeCam HD 3000
Interface	High – speed USB compatible with the USB 2.0 specification
Operating System	1. Microsoft Windows 2. Macintosh OS 3. Android
Image Features	
Sensor	CMOS sensor technology
Resolution	Motion Video: 1280 pixel x 720 pixel resolution Still Image: 1280 x 800
Imaging Rate	Up to 30 frames per second
Field of View	68.5^o diagonal field of view
Aspect ratio	16:9 widescreen
Fixed focus	From 0.3m to 1.5m

Table 4.5 Specification of the webcam.

Pool diameter	10 cm
Propanol	10 mL – 20mL
Dimensions (spatial resolution)	640 x 480
Distance (from webcam to fire source)	110 cm

Table 4.6 Information data of pool fire setup and images

Appendix A: MATLAB

A.1 MATLAB’s Otsu method source code

6/8/16 9:10 PM C:\Documents and Settings\user\My Documents\MATLAB\otsu.m 1 of 5

```
function [IDX,sep] = otsu(I,n)

%OTSU Global image thresholding/segmentation using Otsu's method.
% IDX = OTSU(I,N) segments the image I into N classes by means of Otsu's
% N-thresholding method. OTSU returns an array IDX containing the cluster
% indices (from 1 to N) of each point. Zero values are assigned to
% non-finite (NaN or Inf) pixels.
%
% IDX = OTSU(I) uses two classes (N=2, default value).
%
% [IDX,sep] = OTSU(...) also returns the value (sep) of the separability
% criterion within the range [0 1]. Zero is obtained only with data
% having less than N values, whereas one (optimal value) is obtained only
% with N-valued arrays.
%
% Notes:
% -----
% It should be noticed that the thresholds generally become less credible
% as the number of classes (N) to be separated increases (see Otsu's
% paper for more details).
%
% If I is an RGB image, a Karhunen-Loeve transform is first performed on
% the three R,G,B channels. The segmentation is then carried out on the
% image component that contains most of the energy.
%
% Example:
% -------
% load clown
% subplot(221)
% X = ind2rgb(X,map);
% imshow(X)
% title('Original','FontWeight','bold')
% for n = 2:4
% IDX = otsu(X,n);
% subplot(2,2,n)
% imagesc(IDX), axis image off
% title(['n = ' int2str(n)],'FontWeight','bold')
% end
% colormap(gray)
%
% Reference:
% ---------
% Otsu N, <a
href="matlab:web('http://dx.doi.org/doi:10.1109/TSMC.1979.4310076')">A
Threshold
Selection Method from Gray-Level Histograms</a>,
% IEEE Trans. Syst. Man Cybern. 9:62-66;1979
%
% See also GRAYTHRESH, IM2BW
%
% -- Damien Garcia -- 2007/08, revised 2010/03
% Visit my <a
%
href="matlab:web('http://www.biomecardio.com/matlab/otsu.html')">website</a> for
more details
about OTSU
```

```

error(nargchk(1,2,nargin))

% Check if is the input is an RGB image
isRGB = isrgb(I);

assert(isRGB | ndims(I)==2,...
'The input must be a 2-D array or an RGB image.')

%% Checking n (number of classes)
if nargin==1
n = 2;
elseif n==1;
IDX = NaN(size(I));
sep = 0;
return
elseif n~=abs(round(n)) || n==0
error('MATLAB:otsu:WrongNValue',...
'n must be a strictly positive integer!')
elseif n>255
n = 255;
warning('MATLAB:otsu:TooHighN',...
'n is too high. n value has been changed to 255.')
end

I = single(I);

%% Perform a KLT if isRGB, and keep the component of highest energy
if isRGB
sizI = size(I);
I = reshape(I,[],3);
[V,D] = eig(cov(I));
[~,c] = max(diag(D));
I = reshape(I*V(:,c),sizI(1:2)); % component with the highest energy
end
```

```

%% Convert to 256 levels
I = I-min(I(:));
I = round(I/max(I(:))*255);

%% Probability distribution
unI = sort(unique(I));
nbins = min(length(unI),256);
if nbins==n
IDX = ones(size(I));
for i = 1:n, IDX(I==unI(i)) = i; end
sep = 1;
return
elseif nbins<n
IDX = NaN(size(I));
sep = 0;
return
```

```
elseif nbins<256
[histo,pixval] = hist(I(:),unI);
else
[histo,pixval] = hist(I(:),256);
end
P = histo/sum(histo);
clear unI

%% Zeroth- and first-order cumulative moments
w = cumsum(P);
mu = cumsum((1:nbins).*P);

%% Maximal sigmaB^2 and Segmented image
if n==2
sigma2B =...
(mu(end)*w(2:end-1)-mu(2:end-1)).^2./w(2:end-1)./(1-w(2:end-1));
[maxsig,k] = max(sigma2B);

% segmented image
IDX = ones(size(I));
IDX(I>pixval(k+1)) = 2;

% separability criterion
sep = maxsig/sum(((1:nbins)-mu(end)).^2.*P);

elseif n==3
w0 = w;
w2 = fliplr(cumsum(fliplr(P)));
[w0,w2] = ndgrid(w0,w2);

mu0 = mu./w;
mu2 = fliplr(cumsum(fliplr((1:nbins).*P))./cumsum(fliplr(P)));
```

```
[mu0,mu2] = ndgrid(mu0,mu2);

w1 = 1-w0-w2;
w1(w1<=0) = NaN;

sigma2B =...
w0.*(mu0-mu(end)).^2 + w2.*(mu2-mu(end)).^2 +...
(w0.*(mu0-mu(end)) + w2.*(mu2-mu(end))).^2./w1;
sigma2B(isnan(sigma2B)) = 0; % zeroing if k1 >= k2

[maxsig,k] = max(sigma2B(:));
[k1,k2] = ind2sub([nbins nbins],k);

% segmented image
IDX = ones(size(I))*3;
IDX(I<=pixval(k1)) = 1;
IDX(I>pixval(k1) and I<=pixval(k2)) = 2;

% separability criterion
sep = maxsig/sum(((1:nbins)-mu(end)).^2.*P);
```

```

else
k0 = linspace(0,1,n+1); k0 = k0(2:n);
[k,y] = fminsearch(@sig_func,k0,optimset('TolX',1));
k = round(k*(nbins-1)+1);

% segmented image
IDX = ones(size(I))*n;
IDX(I<=pixval(k(1))) = 1;
for i = 1:n-2
IDX(I>pixval(k(i)) and I<=pixval(k(i+1))) = i+1;
end

% separability criterion
sep = 1-y;

end

IDX(~isfinite(I)) = 0;

%% Function to be minimized if n>=4
function y = sig_func(k)

muT = sum((1:nbins).*P);
sigma2T = sum(((1:nbins)-muT).^2.*P);

k = round(k*(nbins-1)+1);
k = sort(k);
if any(k<1 | k>nbins), y = 1; return, end
```

```

k = [0 k nbins];
sigma2B = 0;
for j = 1:n
wj = sum(P(k(j)+1:k(j+1)));
if wj==0, y = 1; return, end
muj = sum((k(j)+1:k(j+1)).*P(k(j)+1:k(j+1)))/wj;
sigma2B = sigma2B + wj*(muj-muT)^2;
end
y = 1-sigma2B/sigma2T; % within the range [0 1]

end

end

function isRGB = isrgb(A)
% --- Do we have an RGB image?
% RGB images can be only uint8, uint16, single, or double
isRGB = ndims(A)==3 and (isfloat(A) || isa(A,'uint8') || isa(A,'uint16'));
% ---- Adapted from the obsolete function ISRGB ----
if isRGB and isfloat(A)
% At first, just test a small chunk to get a possible quick negative
mm = size(A,1);
```

```
nn = size(A,2);
chunk = A(1:min(mm,10),1:min(nn,10),:);
isRGB = (min(chunk(:))>=0 and max(chunk(:))<=1);
% If the chunk is an RGB image, test the whole image
if isRGB, isRGB = (min(A(:))>=0 and max(A(:))<=1); end
end
end



```

A.2 MATLAB’s RGB to grayscale source code

6/8/16 9:11 PM C:\Program Files\MATLAB\R2010a\toolbox\images\images\rgb2gray.m 1 of 3

```
function I = rgb2gray(varargin)
%RGB2GRAY Convert RGB image or colormap to grayscale.
% RGB2GRAY converts RGB images to grayscale by eliminating the
% hue and saturation information while retaining the
% luminance.
%
% I = RGB2GRAY(RGB) converts the truecolor image RGB to the
% grayscale intensity image I.
%
% NEWMAP = RGB2GRAY(MAP) returns a grayscale colormap
% equivalent to MAP.
%
% Class Support
% -------------
% If the input is an RGB image, it can be uint8, uint16, double, or
% single. The output image I has the same class as the input image. If the
% input is a colormap, the input and output colormaps are both of class
% double.
%
% Example
% -------
% I = imread('board.tif');
% J = rgb2gray(I);
% figure, imshow(I), figure, imshow(J);
%
% [X,map] = imread('trees.tif');
% gmap = rgb2gray(map);
% figure, imshow(X,map), figure, imshow(X,gmap);
%
% See also IND2GRAY, NTSC2RGB, RGB2IND, RGB2NTSC, MAT2GRAY.

% Copyright 1992-2007 The MathWorks, Inc.
% $Revision: 5.20.4.6 $ $Date: 2007/12/10 21:37:27 $

X = parse_inputs(varargin{:});
origSize = size(X);

% Determine if input includes a 3-D array
threeD = (ndims(X)==3);

% Calculate transformation matrix
T = inv([1.0 0.956 0.621; 1.0 -0.272 -0.647; 1.0 -1.106 1.703]);
coef = T(1,:)';

if threeD
%RGB
```

```
% Shape input matrix so that it is a n x 3 array and initialize output
% matrix
X = reshape(X(:),origSize(1)*origSize(2),3);
sizeOutput = [origSize(1), origSize(2)];

% Do transformation
```

```
if isa(X, 'double') || isa(X, 'single')
I = X*coef;
I = min(max(I,0),1);
else
%uint8 or uint16
I = imlincomb(coef(1),X(:,1),coef(2),X(:,2),coef(3),X(:,3), ...
class(X));
end
%Make sure that the output matrix has the right size
I = reshape(I,sizeOutput);

else
I = X * coef;
I = min(max(I,0),1);
I = [I,I,I];
end


%%%
%Parse Inputs
%%%
function X = parse_inputs(varargin)

iptchecknargin(1,1,nargin,mfilename);

if ndims(varargin{1})==2
if (size(varargin{1},2) ~=3 || size(varargin{1},1) < 1)
eid = sprintf('Images:%s:invalidSizeForColormap',mfilename);
msg = 'MAP must be a m x 3 array.';
error(eid,'%s',msg);
end
if ~isa(varargin{1},'double')
eid = sprintf('Images:%s:notAValidColormap',mfilename);
msg1 = 'MAP should be a double m x 3 array with values in the';
msg2 = ' range [0,1].Convert your map to double using IM2DOUBLE.';
error(eid,'%s %s',msg1,msg2);
end
elseif (ndims(varargin{1})==3)
if ((size(varargin{1},3) ~= 3))
eid = sprintf('Images:%s:invalidInputSize',mfilename);
msg = 'RGB must be a m x n x 3 array.';
error(eid,'%s',msg);
end
```

```
else
eid = sprintf('Images:%s:invalidInputSize',mfilename);
msg1 = 'RGB2GRAY only accepts a Mx3 matrix for MAP or a MxNx3 input for ';
msg2 = 'RGB.';
error(eid,'%s %s',msg1,msg2);
end
X = varargin{1};


```

```
%no logical arrays
if islogical(X)
eid = sprintf('Images:%s:invalidType',mfilename);
msg = 'RGB2GRAY does not accept logical arrays as inputs.';
error(eid,'%s',msg);
end

```

A.3 MATLAB ‘s image processing source code

6/10/16 3:06 PM C:\Documents and Settings\user\My Documents\...\Otsu_method_18_10_2011.m 1 of 1

```
%% Fire Image Processing
clear, clc;

%% Load fire image (1)
image1 = imread('IMG_0110.JPG');

figure(1), imshow(image1), title('Fig 1.1: Original Image (Fire Image)');

%% Threshold fire image (1)
I1 = rgb2gray (image1);
threshold1 = graythresh (I1);
BW1 = im2bw (I1, threshold1);

figure(2), imshow(I1), title('Fig 1.2: Convert the image to grey');
figure(3), imshow(BW1), title('Fig 1.3: Convert the image to black and white by MATLAB Otsu method');

%end of program
```

Appendix B: C++ / OpenCV

```
// opencvApplication.cpp : Defines the entry point for the console application.
//

#define _CRT_SECURE_NO_DEPRECATE


#include <cctype>
#include <iostream> // Basic input and output library
#include <iomanip>
#include <iterator>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <windows.h>
#include <fstream> // For file stream

#include "opencv2/video/tracking.hpp" // For optical flow analysis
#include "opencv2/highgui/highgui.hpp" // For histogram
#include "opencv2/imgproc/imgproc.hpp" // For histogram

#include "C:\Users\arthur\Downloads\OpenCV with Visual Studio\Code\opencvApplication_fire4.1\sgVision\SgGeneral.h"
#include "C:\Users\arthur\Downloads\OpenCV with Visual Studio\Code\opencvApplication_fire4.1\sgVision\SgSignal.h"
#include "C:\Users\arthur\Downloads\OpenCV with Visual Studio\Code\opencvApplication_fire4.1\sgVision\SgTimeControl.h"

using namespace std;
using namespace cv;

bool pause = false;

//our sensitivity value to be used in the absdiff() function
//for higher sensitivity, use a lower value
const static int SENSITIVITY_VALUE = 40;

//size of blur used to smooth the intensity image output from absdiff() function
const static int BLUR_SIZE = 10; // SMOOTH THE INTENSITY (EXISING, FOR MOTION DETECTION IS 10)
const static int BLUR_SIZE_INTENSITY = 7; // SMOOTH THE INTENSITY

//these two can be toggled by pressing 'd' or 't' (Motion detection code) (Debug mode is disabled)
//bool debugMode;
```

```
bool trackingEnabled;

//int thresholdParameter = 255; // This thresholdParameter control the intensity
int thresholdParameter; // This thresholdParameter control the intensity

int theObject[2] = { 0, 0 };

//bounding rectangle of the object, we will use the center of this as its position
Rect objectBoundingRectangle = Rect(0, 0, 0, 0);

//float fps = 25.f;

vector<vector<Point>> masks;
vector<Point> mask;
deque<Point> centroids;

//Mat frame;

// get time information //

string intToString(int number)
{

//this function has a number input and string output
std::stringstream ss;
ss << number;
return ss.str();
}

/* Display time and date on hardcopy */
string getDateTime()
{
//get the system time
SYSTEMTIME theTime;
GetLocalTime(andtheTime);
//create string to store the date and time
string dateTime;

//convert year to string
string year = intToString(theTime.wYear);

//use stringstream to add a leading '0' to the month (ie. 3 -> 03)
//we use 'setw(2)' so that we force the string 2 characters wide with a zero in front of it.
//if the month is '10' then it will remain '10'
std::stringstream m;
m << std::setfill('0') << std::setw(2) << theTime.wMonth;
string month = m.str();
//day
std::stringstream d;
```

```
d << std::setfill('0') << std::setw(2) << theTime.wDay;
string day = d.str();
//hour
std::stringstream hr;
hr << setfill('0') << std::setw(2) << theTime.wHour;
string hour = hr.str();
//minute
std::stringstream min;
min << setfill('0') << std::setw(2) << theTime.wMinute;
string minute = min.str();
//second
std::stringstream sec;
sec << setfill('0') << std::setw(2) << theTime.wSecond;
string second = sec.str();

//here we use the year, month, day, hour, minute info to create a custom string
//this will be displayed in the bottom left corner of our video feed.
dateTime = year + "-" + month + "-" + day + " " + hour + ":" + minute + ":" + second;

return dateTime;
}

/* Display time only on hardcopy */
string gettimeonly()
{
//get the system time
SYSTEMTIME theTime;
GetLocalTime(andtheTime);
//create string to store the date and time
string timeonly;

//convert year to string

//hour
std::stringstream hr;
hr << setfill('0') << std::setw(2) << theTime.wHour;
string hour = hr.str();
//minute
std::stringstream min;
min << setfill('0') << std::setw(2) << theTime.wMinute;
string minute = min.str();
//second
std::stringstream sec;
sec << setfill('0') << std::setw(2) << theTime.wSecond;
string second = sec.str();

//here we use the year, month, day, hour, minute info to create a custom string
//this will be displayed in the bottom left corner of our video feed.
```

```
//dateTime = year + "-" + month + "-" + day + " " + hour + ":" + minute + ":" + second;
timeonly = hour + ":" + minute + ":" + second;

return timeonly;
}

/* Display time and date on hardcopy */
string getdateonly()
{
//get the system time
SYSTEMTIME theTime;
GetLocalTime(andtheTime);
//create string to store the date and time
string dateonly;

//convert year to string
string year = intToString(theTime.wYear);

//use stringstream to add a leading '0' to the month (ie. 3 -> 03)
//we use 'setw(2)' so that we force the string 2 characters wide with a zero in front of it.
//if the month is '10' then it will remain '10'
std::stringstream m;
m << std::setfill('0') << std::setw(2) << theTime.wMonth;
string month = m.str();
//day
std::stringstream d;
d << std::setfill('0') << std::setw(2) << theTime.wDay;
string day = d.str();

//here we use the year, month, day, hour, minute info to create a custom string
//this will be displayed in the bottom left corner of our video feed.
dateonly = year + "-" + month + "-" + day;

return dateonly;
}

string getDateTimeForFile()
{
//this function does the exact same as "getDateTime()"
//however it returns a string that can be used as a filename
SYSTEMTIME theTime;
GetLocalTime(andtheTime);
string dateTime;

string year = intToString(theTime.wYear);

std::stringstream m;
m << std::setfill('0') << std::setw(2) << theTime.wMonth;
```

```
string month = m.str();

std::stringstream d;
d << std::setfill('0') << std::setw(2) << theTime.wDay;
string day = d.str();

std::stringstream hr;
hr << setfill('0') << std::setw(2) << theTime.wHour;
string hour = hr.str();

std::stringstream min;
min << setfill('0') << std::setw(2) << theTime.wMinute;
string minute = min.str();

std::stringstream sec;
sec << setfill('0') << std::setw(2) << theTime.wSecond;
string second = sec.str();

//here we use "_" instead of "-" and ":"
//if we try to save a filename with "-" or ":" in it we will get an error.
dateTime = year + "_" + month + "_" + day + "_" + hour + "h" + minute + "m" + second + "s";

return dateTime;
}

/* Dispaly time and date above */

/* Motion detection below */
bool detectMotion(Mat thresholdImage, Mat andcameraFeed){
//create motionDetected variable.
bool motionDetected = false;

//create temp Mat for threshold image
Mat temp;
thresholdImage.copyTo(temp);

//these two vectors needed for output of findContours
vector< vector<Point> > contours;
vector<Vec4i> hierarchy;

//find contours of filtered image using openCV findContours function
//findContours(temp,contours,hierarchy,CV_RETR_CCOMP,CV_CHAIN_APPROX_SIMPLE );// retrieves all contours
findContours(temp, contours, hierarchy, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE);// retrieves external contours

//if contours vector is not empty, we have found some objects
```

```
//we can simply say that if the vector is not empty, motion in the video feed has been detected
.
if (contours.size()>0)motionDetected = true;
else motionDetected = false;

//find the motion object
if (motionDetected){
//the large contour is found at the end of the contours vector
//we will simply assume that the biggest contour is the object
vector< vector<Point> > largestContourVec;
largestContourVec.push_back(contours.at(contours.size() - 1));

//make a bounding rectangle around the largest contour then find its centroid
//this will be the objects final estimated position
objectBoundingRectangle = boundingRect(largestContourVec.at(0));
int xpos = objectBoundingRectangle.x + objectBoundingRectangle.width / 2;
int ypos = objectBoundingRectangle.y + objectBoundingRectangle.height / 2;


//update the objects position by changing the 'theObject' array values
theObject[0] = xpos, theObject[1] = ypos;

}

return motionDetected;

}
/* Motion detection code above */

////////////////////////////////// CALCULATION OPTICAL FLOW ///////////////////////////////////////////
///////
void drawOptFlowMap(const Matand flow, Matand cflowmap, int step, double, const Scalarand color)
{
for (int y = 0; y < cflowmap.rows; y += step)
for (int x = 0; x < cflowmap.cols; x += step)
{
const Point2fand fxy = flow.at<Point2f>(y, x);

// Displacement direction
line(cflowmap, Point(x, y), Point(cvRound(x + fxy.x), cvRound(y + fxy.y)), color);

// Green dot
circle(cflowmap, Point(x, y), 2, color, -1);
}
```

```
}
//////////////////////////////////////////////////////////////////////////////////////////////////

void CallBackFunc(int event, int x, int y, int flags, void* userdata)
{
if ( event == EVENT_LBUTTONDOWN )
{
Point p = Point(x,y);
mask.push_back(p);
}
else if (event == EVENT_RBUTTONDOWN)
{
masks.push_back(mask);
mask.clear();
}
else if (event == EVENT_MBUTTONDOWN)
{
mask.clear();
masks.clear();
}

}

#define FILEPATH "D:/" + getDateTimeForFile() + ".csv" //define file stream object

// open the video image //
int main( int argc, const char** argv ) //program entry points
{

/* Declaraction the motion detection code start */

//set recording and startNewRecording initially to false.
bool recording = false;
bool startNewRecording = false;
int inc = 0;
bool firstRun = true;
//if motion is detected in the video feed, we will know to start recording.
bool motionDetected = false;

//pause and resume code (if needed)
bool pause = false;
//set debug mode and trackingenabled initially to false
//these can be toggled using 'd' and 't'
//debugMode = false;
trackingEnabled = false;

//set up the matrices that we will need
//the two frames we will be comparing
Mat frame1, frame2;
```

```
//their grayscale images (needed for absdiff() function)
Mat grayImage1, grayImage2;
//resulting difference image
Mat differenceImage;
//thresholded difference image (for use in findContours() function)
Mat thresholdImage;

Mat prevgray, gray; //OPTICAL FLOW
Mat flow, cflow; //OPTICAL FLOW

/* Declaraction the motion detection code end */

/* Trackbar control threshold value below */
SgSignal signal;
//cvNamedWindow("display", 0);
//createTrackbar( "threshold", "display", andthresholdParameter, 255, NULL );
//setMouseCallback("display", CallBackFunc, NULL);

/* Trackbar control threshold value above*/

/* Start access camera below */
VideoCapture video;

video.open(0); //Access the webcam.

video >> frame1; //Input the first real time image to frame1

VideoWriter oVideoWriter; //create videoWriter object, not initialized yet (Motion detection code)

if (!video.isOpened())
{
std::cout << "ERROR!!! could not access the webcam !!!" << std::endl;
system("PAUSE");
return EXIT_SUCCESS;
//exit(1);
}

// Record the data to file

fstream file;

file.open(FILEPATH, ios::out | ios::trunc);

if (!file)// Check file open or not
{
cerr << "Sorry!!! Can't open file in Hard drive !!!" << endl;
exit(1);
}
```

```

double dWidth = video.get(CV_CAP_PROP_FRAME_WIDTH); //get the width of frames1 of the video
double dHeight = video.get(CV_CAP_PROP_FRAME_HEIGHT); //get the height of frames1 of the video

//set framesize for use with videoWriter
Size frameSize(static_cast<int>(dWidth), static_cast<int>(dHeight));

/* Display on command prompt */
cout << "----------------------------------------------------------------------------" << endl;
cout << "VIDEO FIRE DETECTION" << endl;
cout << "Frame Size = " << frameSize << endl;
cout << "Record date " << getdateonly () << endl;
cout << "Start record time " << gettimeonly() << endl;
cout << "OpenCV verson " << CV_VERSION << endl;
cout << "----------------------------------------------------------------------------" << endl;
cout << "Press 'Esc' exit the Code" << endl;
cout << "Press 'p' paused or resume the code" << endl;
cout << "Press 'Space bar' will toggle tracking" << endl;

/* Write on hard disk */
file << "VIDEO FIRE DETECTION DATA RECORD" << endl;
file << "Frame Size " << "," << frameSize << endl;
file << "Record date " << "," << getdateonly() << endl;
file << "OpenCV verson " << CV_VERSION << endl;
file << " " << endl;

file << setw(0) << "," << setiosflags(ios::right) << "FROM FLAME-INTENSITY" << "," << "," << ",
"
<< setw(0) << "," << "," << "," << setiosflags(ios::right) << "FROM FLAME-MOTION"
<< endl;

file << setw(0) << setiosflags(ios::right) << "Time" << ","

<< setw(0) << setiosflags(ios::right) << "Otsu Threshold value" << ","

<< setw(0) << setiosflags(ios::right) << "Max. Gray level" << ","

<< setw(0) << setiosflags(ios::right) << "ROI" << ","
<< setw(0) << setiosflags(ios::right) << "Height" << ","
<< setw(0) << setiosflags(ios::right) << "Centroid X" << ","
<< setw(0) << setiosflags(ios::right) << "Centroid Y" << ","

<< setw(0) << setiosflags(ios::right) << "ROI" << ","
<< setw(0) << setiosflags(ios::right) << "Height" << ","
<< setw(0) << setiosflags(ios::right) << "Centroid X" << ","
<< setw(0) << setiosflags(ios::right) << "Centroid Y" << ","

<< setw(0) << setiosflags(ios::right) << "Pixel no. (Red)" << ","
<< setw(0) << setiosflags(ios::right) << "Pixel no. (Green)" << ","
<< setw(0) << setiosflags(ios::right) << "Pixel no. (Blue)"
<< endl;

deque<float> heights,widths,areas,mheight;
double numberFrame = video.get(CV_CAP_PROP_FRAME_COUNT);
int countFrame = 0;
float lengthTrajectory = 10.0f;
float sizeWindow = 1920.0f;

while (true) // Infinite Loop
{
if (!pause) video >> frame1;
countFrame++;
if (countFrame == numberFrame - 1)
{
video.set(CV_CAP_PROP_POS_FRAMES, 1);
countFrame = 1;
continue;
}

//localize the fire
Mat image, grayimage, grayimage2, fireMask, display, graph;

frame1.copyTo(display);

/*Motion detection code start*/

//read first frame
video.read(frame1);

////////////////////////////////////////////////////////////////////////////////
////////

//convert frame1 to gray scale for frame differencing
cv::cvtColor(frame1, grayImage1, COLOR_BGR2GRAY);

//copy second frame
video.read(frame2);

//convert frame2 to gray scale for frame differencing
cv::cvtColor(frame2, grayImage2, COLOR_BGR2GRAY);

////////////////////////////////////////////////////////////////////////////////
/////////

//perform frame differencing with the sequential images. This will output an
"intensity
```

```
image"
//do not confuse this with a threshold image, we will need to perform thresholding
afterwards.
cv::absdiff(grayImage1, grayImage2, differenceImage);
// COMPARE THE DIFF. for motion detection

//threshold intensity image at a given sensitivity value
cv::threshold(differenceImage, thresholdImage, SENSITIVITY_VALUE, 255, THRESH_BINARY);
// threshold --> SENSITIVITY VALUE = 40

/*if (debugMode == true){
//show the difference image and threshold image
cv::imshow("Difference Image", differenceImage);
cv::imshow("Threshold Image", thresholdImage);
}
else{
//if not in debug mode, destroy the windows so we don't see them anymore
cv::destroyWindow("Difference Image");
cv::destroyWindow("Threshold Image");
}*/

//blur the image to get rid of the noise. This will output an intensity image
cv::blur(thresholdImage, thresholdImage, cv::Size(BLUR_SIZE, BLUR_SIZE)); // Dilate and
Erode

//threshold again to obtain binary image from blur output
cv::threshold(thresholdImage, thresholdImage, SENSITIVITY_VALUE, 255, THRESH_BINARY);
//Sensitivity value is 40 (threshold)


/*if (debugMode == true){
//show the threshold image after it's been "blurred"

imshow("Final Threshold Image", thresholdImage);

}
else {
//if not in debug mode, destroy the windows so we don't see them anymore
cv::destroyWindow("Final Threshold Image");
}*/

//if tracking enabled, search for Motion
if (trackingEnabled){
//detectMotion function will return true if motion is detected, else it will return false.
//set motionDetected boolean to the returned value.
motionDetected = detectMotion(thresholdImage, frame1);
```

```
}
else{
//reset our variables if tracking is disabled
recording = false;
motionDetected = false;
}
/*Motion detection code end*/

/*Motion detection code start*/
//if we're in recording mode, write to file
if (recording){

oVideoWriter.write(frame1);
//show "REC" in red
//be sure to do this AFTER you write to the file so that "REC" doesn't show up
//on the recorded video.
circle(display, Point(500, 20), 12.0, Scalar(0, 0, 255), -1, 8);
putText(display, "REC", Point(515, 28), 2, 1, Scalar(0, 0, 255), 2);

}
if (motionDetected){
//show "MOTION DETECTED" in bottom left corner in green
//once again, be sure to do this AFTER you write to the file so that "MOTION DETECTED" doesn't show up
//on the recorded video. Place this code above if(recording) to see what I'm talking about.
putText(display, "MOTION DETECTED", cv::Point(0, 420), 2, 1, cv::Scalar(0, 255, 0), 2);

//set recording to true since there is motion in the video feed.
recording = true;

if (firstRun == true){

string videoFileName = "D:/" + getDateTimeForFile() + ".avi";
cout << "File has been opened for writing: " << videoFileName << endl;
oVideoWriter = VideoWriter(videoFileName, CV_FOURCC('D', 'I', 'V', '3'), 20, frameSize, true);

if (!oVideoWriter.isOpened())
{
cout << "ERROR!!! Failed to initialize video writing in Hard drive !!!" << endl;
getchar();
return -1;
}
firstRun = false;
543
544 }
545
546 }
547 else recording = false;
548 /*Motion detection code end*/
549
550 cvtColor(frame2, grayimage, CV_BGR2GRAY); // Convert to gratscale and
store in
"grayimage" ORIGINAL DATA FROM WEBCAM
551
552 // Calculation the gray values (IN PROGRESS)
553 double alpha = 0.2989, Beta = 0.5870, Gamma = 0.1140; // For calculation !!
554 vector<Mat> bgr_planesforgray;
555 split(frame2, bgr_planesforgray);
556 /*
557 float hist_val[256];
558 for (int i = 0; i < 256; i++)
559 {
560 hist_val[i] = 0.0;
561 //cout << i << " " << hist_val[i] << endl;
562
563 }
564 */
565 /* Analysis for captured the Region of Interest */
566 cvtColor(frame2,image,CV_BGR2GRAY); // Convert to grayscale and
store in "image
" for analysis
567
568 // Smooths an image using the Gaussian filter (Dilate and Erode)
569
GaussianBlur(image,image,Size(BLUR_SIZE_INTENSITY,BLUR_SIZE_INTE
NSITY),0,0);
570 //blur(image, image, Size(BLUR_SIZE, BLUR_SIZE)); // BLUR_SIZE = 10
reference: line no.
32
571
572
573 // Otsu calculation (IN PROGRESS)
574 /*
575 float wB; //Weight Background
576 float wF; //Weight Foreground
577 float mB; //Mean Background
578 float mF; //Mean Foreground
579 float varBetween; //Between Class Variance
580 float varMax; //Maximum Between Class Variance
581 */
582 // Total number of pixels
583
584
585 // Calculate Weight background
```

```


//threshold(image, fireMask, thresholdParameter, 255, cv::THRESH_BINARY);


// Optimal threshold value obtained by Otsu algorithm
thresholdParameter = threshold(image, fireMask, 0, 255, cv::THRESH_BINARY | cv::THRESH_OTSU); // WARNING double to int problem ??

fillPoly(fireMask,masks,cvScalarAll(0));
bitwise_and(image, fireMask, image);

/*
/// Apply Histogram Equalization
equalizeHist(image,image);

Mat dst;
equalizeHist(image, dst);

/// Display results
namedWindow("equalized_window", CV_WINDOW_AUTOSIZE);
imshow("equalized_window", dst);
*/

//resize the user interface
/*
resizeWindow("input", 450, 350);
resizeWindow("display", 450, 350);
cvResizeWindow("rgb and gray_Hist", 450, 350);
resizeWindow("data", 450, 350);

//resizeWindow("gray", 240, 120);
//cvResizeWindow("Threshold by Otsu", 240, 120);
//cvResizeWindow("Diff. by motion", 240, 120);
//cvResizeWindow("Threshold by motion", 240, 120);

//cvResizeWindow("Gray Hist", 320, 240);
//cvResizeWindow("Red Hist", 320, 200);
//cvResizeWindow("Green Hist", 320, 200);
//cvResizeWindow("Blue Hist", 320, 200);

//resizeWindow("Motion", 800, 100);

//resizeWindow("flow", 320, 240);
*/

//FIXED LOCATION OF WINDOWS
/*
cvMoveWindow("input", 10, 0);
cvMoveWindow("display", 10, 400);
cvMoveWindow("rgb and gray_Hist", 500, 0);
cvMoveWindow("data", 500, 400);

cvMoveWindow("Gray Hist", 640, 0);
cvMoveWindow("flow", 960, 0);
cvMoveWindow("Threshold by Otsu", 1280, 160);
cvMoveWindow("gray", 1280, 0);
cvMoveWindow("Threshold by Otsu", 1280, 160);
cvMoveWindow("Diff. by motion", 1280, 320);
cvMoveWindow("Threshold by motion", 1280, 480);

cvMoveWindow("Red Hist", 640, 280);
cvMoveWindow("Green Hist", 960, 280);
cvMoveWindow("Blue Hist", 960, 560);

cvMoveWindow("Motion", 0, 560);
*/

cvNamedWindow("Threshold by Otsu", WINDOW_NORMAL);
cvNamedWindow("Diff. by motion", WINDOW_NORMAL);
cvNamedWindow("Threshold by motion", WINDOW_NORMAL);

cv::imshow("Threshold by Otsu", fireMask);


cv::imshow("Diff. by motion",differenceImage);
cv::imshow("Threshold by motion",thresholdImage);

//get contours
vector<vector<Point>> contourFires;
vector<Vec4i> hierarchy;
        findContours(fireMask,contourFires,hierarchy,CV_RETR_CCOMP,
CV_CHAIN_APPROX_SIMPLE);
vector<Point> contourMax;
SgGeneral::sgGetMaxContour(contourFires,20,contourMax);
if (!contourMax.empty())
{
Point centroid = SgGeneral::sgGetCentroid(contourMax);

centroids.push_back(centroid);

while (centroids.size() > lengthTrajectory) centroids.pop_front();
}

float area = SgGeneral::sgGetArea(contourFires);
areas.push_back(area);
if (areas.size() > sizeWindow) areas.pop_front();

//calculate bounding box of multiple contours
int yMin = 999, yMax = 0, xMax = 0, xMin = 999;
```

```

Point peak;
for( int i = 0; i < contourFires.size(); i++ )
{
if (contourFires[i].size() > 20)
{
for( int j = 0; j < contourFires[i].size(); j++ )
{
if (contourFires[i][j].y > yMax) yMax = contourFires[i][j].y;
else if(contourFires[i][j].y < yMin)
{
yMin = contourFires[i][j].y;
peak.x = contourFires[i][j].x;
peak.y = contourFires[i][j].y;
}
if (contourFires[i][j].x > xMax) xMax = contourFires[i][j].x;
else if(contourFires[i][j].x < xMin) xMin = contourFires[i][j].x;
}
}
}

// Input the informaton data
heights.push_back(yMax-yMin); // WARNING int to float problem ??

widths.push_back(xMax-xMin); // WARNING int to float problem ??
mheight.push_back(objectBoundingRectangle.height); // NEW Pushback motion height to "mheight"

if (heights.size() > sizeWindow)
{
heights.pop_front();
widths.pop_front();
mheight.pop_front();
}

//visualize ??
polylines(display,masks,true,CV_RGB(0,0,128),1);
for (int j = 0; j < mask.size(); j++) circle(display,mask[j],2,CV_RGB(255,0,0),-1);

//mask the segmentation region
/*Mat rgb[3];
split(display,rgb);
rgb[0] += image; // Blue
rgb[1] += image; // Green
rgb[2] += image; // Red
merge(rgb,3,display);
*/

if (!contourFires.empty())
for( int i = 0; i < contourFires.size(); i++ )
```

```
drawContours(display,contourFires,i,CV_RGB(0,0,0), 2); // draw contout line in display

//float scale = 255. / lengthTrajectory;
double scale = 255. / lengthTrajectory;
if(centroids.size() > 2)
{
for (int i = 0; i < centroids.size() - 1; i++)
{
circle(display, Point(centroids.back().x, centroids.back().y), 7, Scalar(255, 0, 0), 2); //draw centroid

line(display, Point(centroids.back().x, centroids.back().y), Point(centroids.back().x, centroids.back().y - 15),
Scalar(255, 0, 0), 2);
line(display, Point(centroids.back().x, centroids.back().y), Point(centroids.back().x, centroids.back().y + 15),
Scalar(255, 0, 0), 2);
line(display, Point(centroids.back().x, centroids.back().y), Point(centroids.back().x - 15, centroids.back().y),
Scalar(255, 0, 0), 2);
line(display, Point(centroids.back().x, centroids.back().y), Point(centroids.back().x + 15, centroids.back().y),
Scalar(255, 0, 0), 2);

//line(display, centroids[i], centroids[i+1], CV_RGB(0, scale*i, 0), 2);
}
}

circle(display,centroids.back(),3,CV_RGB(0,255,0),-1); // ????

//make some temp x and y variables
int x = theObject[0];
int y = theObject[1];

//draw some crosshairs (motion detection)

circle(display, Point(x, y), 7, Scalar(0, 255, 0), 2);

line(display, Point(x, y), Point(x, y - 15), Scalar(0, 255, 0), 2);
line(display, Point(x, y), Point(x, y + 15), Scalar(0, 255, 0), 2);
line(display, Point(x, y), Point(x - 15, y), Scalar(0, 255, 0), 2);
line(display, Point(x, y), Point(x + 15, y), Scalar(0, 255, 0), 2);

// drawing region of interest (ROI) based on xMin,yMin,xMax,yMax (Threshold analysis)
```

```
rectangle(display, Point(xMin, yMin), Point(xMax, yMax), CV_RGB(255, 255, 0), 2); // threshold analysis

rectangle(display, Point(objectBoundingRectangle.x, objectBoundingRectangle.y),
Point(objectBoundingRectangle.x + objectBoundingRectangle.width, objectBoundingRectangle.y + objectBoundingRectangle.height),
CV_RGB(255, 0, 0), 2);

//draw time stamp to video in bottom left corner. We draw it before we write so that it is written on the video file.
rectangle(display, Point(0, 460), Point(200, 480), Scalar(255, 255, 255), -1);
putText(display, getDateTime(), Point(0, 475), 1, 1, Scalar(0, 0, 0), 2); // show date time on 'display'
rectangle(frame2, Point(0, 460), Point(200, 480), Scalar(255, 255, 255), -1);
putText(frame2, getDateTime(), Point(0, 475), 1, 1, Scalar(0, 0, 0), 2); // show date time on 'frame'
rectangle(grayimage, Point(0, 460), Point(200, 480), Scalar(255, 255, 255), -1);
putText(grayimage, getDateTime(), Point(0, 475), 1, 1, Scalar(0, 0, 0), 2); // show date time on 'grayimage'

//draw the height, width and area region of interest
string heightFire = "ROI Height (Intnesity) : " + to_string(yMax - yMin);
//putText(display, heightFire, Point(40, 20), 1, 1, CV_RGB(0, 0, 255), 2);

string widthFire = "ROI Width (Intensity) : " + to_string(xMax - xMin);
//putText(display, widthFire, Point(40, 40), 1, 1, CV_RGB(0, 0, 255), 2);

string ROIFire = "ROI Area (Intensity) : " + to_string((yMax - yMin)*(xMax - xMin));
//putText(display, ROIFire, Point(40, 60), 1, 1, CV_RGB(0, 0, 255), 2);

string fireThreshold = "Otsu : " + to_string(thresholdParameter); // Intensity threshold
//putText(display, fireThreshold, Point(320, 20), 1, 1, CV_RGB(255, 255, 255), 2);

//draw the centroid coordinate X and Y
string cenFire = "X,Y (Intensity) : ";
if (!centroids.empty())
cenFire = "X,Y (Intensity) : " + to_string(centroids.back().x) + "," + to_string (centroids.back().y);
//putText(display, cenFire, Point(40, 80), 1, 1, CV_RGB(0, 0, 255), 2);

string movcenFire = "X,Y (Motion) : " + to_string(x)+","+to_string(y);
```

```
//putText(display, movcenFire, Point(40, 100), 1, 1, CV_RGB(255, 0, 0), 2);

string ROIMovFire = "ROI Area (Motion) : " + to_string(objectBoundingRectangle.width * objectBoundingRectangle.height);
//putText(display, ROIMovFire, Point(40, 120), 1, 1, CV_RGB(255, 0, 0), 2);

string ROIMovheightFire = "ROI Height (Motion) : " + to_string(objectBoundingRectangle.height);
//putText(display, ROIMovheightFire, Point(40, 140), 1, 1, CV_RGB(255, 0, 0), 2);

string ROIMovwidthFire = "ROI Width (Motion) : " + to_string(objectBoundingRectangle.width);
//putText(display, ROIMovwidthFire, Point(40, 160), 1, 1, CV_RGB(255, 0, 0), 2);


/*Histogram analysis*/

/// Separate the image in 3 places ( B, G and R )
vector<Mat> bgr_planes;
split(frame2, bgr_planes);

// Separate the image in gray places
vector<Mat> gray_planes;
split(image, gray_planes);

vector<Mat> grayplanes;
split(grayimage, grayplanes);

/// Establish the number of bins
int histSize = 256; //From 0 to 255

/// Set the ranges (for B,G,R and gray)
float range[] = { 0, 256 }; //the upper boundary is exclusive
const float* histRange = { range };

bool uniform = true; bool accumulate = false;

Mat b_hist, g_hist, r_hist, gray_hist;

/// Compute the histograms:
calcHist(andbgr_planes[0], 1, 0, Mat(), b_hist, 1, andhistSize, andhistRange, uniform, accumulate);
calcHist(andbgr_planes[1], 1, 0, Mat(), g_hist, 1, andhistSize, andhistRange, uniform, accumulate);
calcHist(andbgr_planes[2], 1, 0, Mat(), r_hist, 1, andhistSize,
andhistRange, uniform,
accumulate);

calcHist(andgrayplanes[0], 1, 0, Mat(), gray_hist, 1, andhistSize,
andhistRange, uniform,
accumulate);

// Draw the histograms for B, G and R
int hist_w = 512; int hist_h = 400;
int bin_w = cvRound((double)hist_w / histSize); // int histSize = 256

Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));

Mat R_histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
Mat G_histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
Mat B_histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));

Mat grayhistImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));

/// Normalize the result to [ 0, histImage.rows ]
normalize(b_hist, b_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
normalize(g_hist, g_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
normalize(r_hist, r_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());

normalize(b_hist, b_hist, 0, B_histImage.rows, NORM_MINMAX, -1, Mat());
normalize(g_hist, g_hist, 0, G_histImage.rows, NORM_MINMAX, -1, Mat());
normalize(r_hist, r_hist, 0, R_histImage.rows, NORM_MINMAX, -1, Mat());

normalize(gray_hist, gray_hist, 0, grayhistImage.rows, NORM_MINMAX, -
1, Mat()); //Normalize
the gray result

/// Draw for each channel
int gray_maxtemp = 0;
int red_maxtemp = 0;
int green_maxtemp = 0;

for (int i = 0; i < histSize; i++)
{
line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(b_hist.at<float>(i - 1))),
Point(bin_w*(i), hist_h - cvRound(b_hist.at<float>(i))),
Scalar(255, 0, 0), 0, 8, 0);
// Blue colour

line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(g_hist.at<float>(i - 1))),
Point(bin_w*(i), hist_h - cvRound(g_hist.at<float>(i))),
Scalar(0, 255, 0), 0, 8, 0);
// Green colour

line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(r_hist.at<float>(i - 1))),
Point(bin_w*(i), hist_h - cvRound(r_hist.at<float>(i))),
Scalar(0, 0, 255), 0, 8, 0);
// Red colour

line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(gray_hist.at<float>(i - 1))),
// Gray color
Point(bin_w*(i), hist_h - cvRound(gray_hist.at<float>(i))),
Scalar(255, 255, 255), 0, 8, 0);

/* Individual histogram */
line(B_histImage, Point(bin_w*(i - 1), hist_h),
Point(bin_w*(i), hist_h - cvRound(b_hist.at<float>(i))),
Scalar(255, 0, 0), 0, 8, 0);
// Blue colour

line(G_histImage, Point(bin_w*(i - 1), hist_h),
Point(bin_w*(i), hist_h - cvRound(g_hist.at<float>(i))),
Scalar(0, 255, 0), 0, 8, 0);
// Green colour

line(R_histImage, Point(bin_w*(i - 1), hist_h),
Point(bin_w*(i), hist_h - cvRound(r_hist.at<float>(i))),
Scalar(0, 0, 255), 0, 8, 0);
// Red colour

line(grayhistImage, Point(bin_w*(i - 1), hist_h),
// Gray color
Point(bin_w*(i), hist_h - cvRound(gray_hist.at<float>(i))),
Scalar(255, 255, 255), 0, 8, 0);

// Indicate the threshold
line(grayhistImage, Point(bin_w*(thresholdParameter), hist_h),
Point(bin_w*(thresholdParameter), hist_h - cvRound(gray_hist.at<float>(i))),
Scalar(0, 255, 255), 0, 4, 0);

line(histImage, Point(bin_w*(thresholdParameter), hist_h),
Point(bin_w*(thresholdParameter), hist_h - cvRound(gray_hist.at<float>(i))),
Scalar(0, 255, 255), 0, 8, 0);

/* Analysis the maximim gray level */
float gray_binVal = gray_hist.at<float>(i);
if (gray_hist.at<float>(i) > gray_maxtemp)
{
gray_maxtemp = i;
}

}

```

```
//display on grayhistImage
string maxgray = "Max Gray Level : " + to_string(gray_maxtemp);
putText(grayhistImage, maxgray, Point(40, 20), 1, 1, CV_RGB(255, 255, 255), 2);

string threshold_value = "Threshold by Otsu : " + to_string(thresholdParameter);
//putText(grayhistImage, threshold_value, Point(40, 40), 1, 1, CV_RGB(255, 255, 0), 2);

//diaply on histImage
string threshold_value_hist = "Threshold by Otsu : " + to_string(thresholdParameter);
putText(histImage, threshold_value_hist, Point(40, 20), 1, 1, CV_RGB(255, 255, 0), 2);

string nosofpixels_red = "Pixels no.(R) : " + to_string(r_hist.at<float>(thresholdParameter));
putText(histImage, nosofpixels_red, Point(40, 40), 1, 1, CV_RGB(255, 0, 0), 2);
string pixelsno_red = to_string((int)r_hist.at<float>(thresholdParameter));

string nosofpixels_green = "Pixels no.(G) : " + to_string(g_hist.at<float>(thresholdParameter));
putText(histImage, nosofpixels_green, Point(40, 60), 1, 1, CV_RGB(0, 255, 0), 2);
string pixelsno_green = to_string((int)g_hist.at<float>(thresholdParameter));

string nosofpixels_blue = "Pixels no.(B) : " + to_string(b_hist.at<float>(thresholdParameter));
putText(histImage, nosofpixels_blue, Point(40, 80), 1, 1, CV_RGB(0, 0, 255), 2);
string pixelsno_blue = to_string((int)b_hist.at<float>(thresholdParameter));

file << setw(1) << setiosflags(ios::right) << gettimeonly() << "," //date and time
<< setw(4) << setiosflags(ios::right) << thresholdParameter << "," //threhsold Otsu
<< setw(5) << setiosflags(ios::right) << gray_maxtemp << "," //Max. gray level
<< setw(8) << setiosflags(ios::right) << (yMax - yMin)*(xMax - xMin) << "," //ROI
<< setw(4) << setiosflags(ios::right) << (yMax - yMin) << "," //Region Height
<< setw(4) << setiosflags(ios::right) << centroids.back().x <<"," //Centroid (Intensity)
<< setw(3) << setiosflags(ios::right) << centroids.back().y << "," //Centroid (Intensity)
<< setw(5) << setiosflags(ios::right) << x <<"," //Centroid X (Motion)
<< setw(3) << setiosflags(ios::right) << y << ","
//Centroid Y (Motion)
<< setw(5) << setiosflags(ios::right) << objectBoundingRectangle.height
<< ","
//motion object height
<< setw(7) << setiosflags(ios::right) << objectBoundingRectangle.width *
objectBoundingRectangle.height << "," //ROI (Motion)
<< setw(7) << setiosflags(ios::right) << pixelsno_red << ","
//red color pixels in threshold Otsu
<< setw(7) << setiosflags(ios::right) << pixelsno_green << ","
//green color pixels in threshold Otsu
<< setw(7) << setiosflags(ios::right) << pixelsno_blue << ","
//blue color pixels in threshold Otsu
<< endl;


//display the output on screen

string text1 = "Developed by : Arthur Wong" ;

int fontFace = FONT_HERSHEY_SIMPLEX;
double fontScale = 0.8;
int thickness = 2.0;

Mat img(300, 900, CV_8UC3, Scalar::all(0));

putText(img, string("VIDEO FIRE DETECTION"), Point(20, 20), fontFace,
fontScale, Scalar::
all(255), thickness, 8);

putText(img, threshold_value, Point (20, 45), fontFace, fontScale,
Scalar::all(255),
thickness, 8);
putText(img, nosofpixels_red, Point(20, 70), fontFace, fontScale,
Scalar::all(255),
thickness, 8);
putText(img, nosofpixels_green, Point(20, 95), fontFace, fontScale,
Scalar::all(255),
thickness, 8);
putText(img, nosofpixels_blue, Point(20, 120), fontFace, fontScale,
Scalar::all(255),
thickness, 8);
putText(img, heightFire, Point(20, 145), fontFace, fontScale,
Scalar::all(255), thickness,
8);
putText(img, widthFire, Point(20, 170), fontFace, fontScale,
Scalar::all(255), thickness,
8);
putText(img, cenFire, Point(20, 195), fontFace, fontScale, Scalar::all(255),
thickness, 8);
```

```
putText(img, ROIFire, Point(20, 220), fontFace, fontScale, Scalar::all(255), thickness, 8);

putText(img, ROIMovheightFire, Point(500, 45), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, ROIMovwidthFire, Point(500, 70), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, movcenFire, Point(500, 95), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, ROIMovFire, Point(500, 120), fontFace, fontScale, Scalar::all(255), thickness, 8);

putText(img, text1, Point (500, 280), fontFace, fontScale, Scalar::all(255), thickness, 8);

namedWindow("data", WINDOW_NORMAL);
imshow("data", img);



/*****************************************************************************************
**
*********************************************************/
cvNamedWindow("input", WINDOW_NORMAL);
cvNamedWindow("gray", WINDOW_NORMAL);
cvNamedWindow("display",WINDOW_NORMAL);

cv::imshow("input",frame2); // display input video image
cv::imshow("gray",grayimage); // display gray video image
cv::imshow("display",display); // display final video image

/// Display histogram
cvNamedWindow("Gray Hist", WINDOW_NORMAL);

cvNamedWindow("Red Hist", WINDOW_NORMAL);
cvNamedWindow("Green Hist", WINDOW_NORMAL);
cvNamedWindow("Blue Hist", WINDOW_NORMAL);

cvNamedWindow("rgb and gray_Hist", WINDOW_NORMAL);

cv::imshow("Red Hist", R_histImage);
cv::imshow("Green Hist", G_histImage);
cv::imshow("Blue Hist", B_histImage);
cv::imshow("Gray Hist", grayhistImage);
cv::imshow("rgb and gray_Hist", histImage);

if (widths.size() > 1) // SgSignal.cpp line no 332
```

```
{

//vector<float> tempH;
vector<float> motionheight;
//vector<float> tempW;
//vector<float> tempA;

//signal.sgDequeToVector(widths,tempW);
//signal.sgDequeToVector(heights, tempH);
signal.sgDequeToVector(mheight,motionheight);
//signal.sgDequeToVector(areas,tempA);

//signal.sgNormalizeByMinMax(tempW,tempW,0,1000); //original 300 red
//signal.sgNormalizeByMinMax(tempH, tempH, 0, 1000); //original 300
green
signal.sgNormalizeByMinMax(motionheight, motionheight, 0, 300);
//signal.sgNormalizeByMinMax(tempA,tempA,0,30000);

Mat graph = Mat::zeros(200, 1920, CV_8UC3);
//signal.sgDraw01(graph, CV_RGB(255, 255, 255), tempH, 1);
signal.sgDraw01(graph, CV_RGB(255, 255, 255), motionheight, 1);
//signal.sgDraw01(graph,CV_RGB(255,0,0),tempW,1);


//calculation and display on chart
int intensityflameheight = (yMax - yMin);
int motionflameheight = (objectBoundingRectangle.height);


//string intflicker = "flame height (intensity) : " +
to_string(intensityflameheight);
//putText(graph, intflicker, Point(40, 60), 1, 1, CV_RGB(255, 255, 255), 2);

string motflicker = "Object motion (height) : " +
to_string(motionflameheight);
putText(graph, motflicker, Point(40, 20), 1, 1, CV_RGB(255, 255, 255), 2);

cvNamedWindow("Motion", WINDOW_NORMAL);
cv::imshow("Motion", graph);


}

//////////////////////////////// CALCULATION OPTICAL FLOW
////////////////////////////

namedWindow("flow", WINDOW_NORMAL);
//namedWindow("realtime input", 1);

video.read(frame2);
cvtColor(frame2, gray, COLOR_BGR2GRAY); //
```

```

//imshow("realtime input", frame2);
imshow("flow", gray);

if (prevgray.data)
{
calcOpticalFlowFarneback(prevgray, gray, flow, 0.5, 3, 15, 3, 5, 1.2, 0);

cvtColor(prevgray, cflow, COLOR_GRAY2BGR);
drawOptFlowMap(flow, cflow, 16, 1.5, Scalar(0, 255, 0));

imshow("flow", cflow);
}

//////////////////////////////////////////////////////////////////////////////////////

/*
int c = cvWaitKey(30);
if(c == 27) break;
else if (c == 32) pause = !pause;
else if (char(c) == '+') for (int i = 0; i < 30*5; i++) video.grab();
else if (char(c) == 'n') video >> frame;
*/

switch (waitKey(30))
{
case 27: //'Esc' has been pressed. this wil exit the code.
file << "END" << endl;
file.close();
return 0;

case 32: //'Space bar' has been pressed. this will toggle tracking
trackingEnabled = !trackingEnabled;
if (trackingEnabled == false) cout << "Tracking disabled." << endl;
else cout << "Tracking enabled." << endl;
break;

case 112: //'p' has been pressed. this will pause/resume the code.
pause = !pause;
if (pause == true)
{
cout << "Code paused, press 'p' again to resume" << endl;
while (pause == true)
{
//stay in this loop until
switch (waitKey())
{
//a switch statement inside a switch statement? Mind blown.
case 112:
//change pause back to false
pause = false;
cout << "Code Resumed" << endl;
break;


}
}

}

}
swap(prevgray, gray); // OPTICAL FLOW
}
//cvDestroyWindow("fire");

return 0; // Exit the program

}
```

Appendix B1: Revised C++ / OpenCV source code

```
// opencvApplication.cpp : Defines the entry point for the console application.
//

#define _CRT_SECURE_NO_DEPRECATE


#include <cctype>
#include <iostream> // Basic input and output library
#include <iomanip>
#include <iterator>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <windows.h>
#include <fstream> // For file stream

#include "opencv2/video/tracking.hpp" // For optical flow analysis
#include "opencv2/highgui/highgui.hpp" // For histogram
#include "opencv2/imgproc/imgproc.hpp" // For histogram

#include "C:\Users\arthur\Downloads\OpenCV with Visual Studio\Code\opencvApplication_fire4.1\sgVision\SgGeneral.h"
#include "C:\Users\arthur\Downloads\OpenCV with Visual Studio\Code\opencvApplication_fire4.1\sgVision\SgSignal.h"
#include "C:\Users\arthur\Downloads\OpenCV with Visual Studio\Code\opencvApplication_fire4.1\sgVision\SgTimeControl.h"

using namespace std;
using namespace cv;

bool pause = false;

//our sensitivity value to be used in the absdiff() function
//for higher sensitivity, use a lower value
const static int SENSITIVITY_VALUE = 40;

//size of blur used to smooth the intensity image output from absdiff() function
const static int BLUR_SIZE = 10; // SMOOTH THE INTENSITY (EXISING, FOR MOTION DETECTION IS 10)
const static int BLUR_SIZE_INTENSITY = 7; // SMOOTH THE INTENSITY

//these two can be toggled by pressing 'd' or 't' (Motion detection code) (Debug mode is disabled)
```

```
//bool debugMode;
bool trackingEnabled;

//int thresholdParameter = 255; // This thresholdParameter control the intensity
int thresholdParameter; // This thresholdParameter control the intensity

int theObject[2] = { 0, 0 };

//bounding rectangle of the object, we will use the center of this as its position
Rect objectBoundingRectangle = Rect(0, 0, 0, 0);

//float fps = 25.f;

vector<vector<Point>> masks;
vector<Point> mask;
deque<Point> centroids;

//Mat frame;

// get time information //

string intToString(int number)
{

//this function has a number input and string output
std::stringstream ss;
ss << number;
return ss.str();
}

/* Display time and date on hardcopy */
string getDateTime()
{
//get the system time
SYSTEMTIME theTime;
GetLocalTime(andtheTime);
//create string to store the date and time
string dateTime;

//convert year to string
string year = intToString(theTime.wYear);

//use stringstream to add a leading '0' to the month (ie. 3 -> 03)
//we use 'setw(2)' so that we force the string 2 characters wide with a zero in front of it.
//if the month is '10' then it will remain '10'
std::stringstream m;
m << std::setfill('0') << std::setw(2) << theTime.wMonth;
string month = m.str();
```

```
//day
std::stringstream d;
d << std::setfill('0') << std::setw(2) << theTime.wDay;
string day = d.str();
//hour
std::stringstream hr;
hr << setfill('0') << std::setw(2) << theTime.wHour;
string hour = hr.str();
//minute
std::stringstream min;
min << setfill('0') << std::setw(2) << theTime.wMinute;
string minute = min.str();
//second
std::stringstream sec;
sec << setfill('0') << std::setw(2) << theTime.wSecond;
string second = sec.str();

//here we use the year, month, day, hour, minute info to create a custom string
//this will be displayed in the bottom left corner of our video feed.
dateTime = year + "-" + month + "-" + day + " " + hour + ":" + minute + ":" + second;

return dateTime;
}

/* Display time only on hardcopy */
string gettimeonly()
{
//get the system time
SYSTEMTIME theTime;
GetLocalTime(andtheTime);
//create string to store the date and time
string timeonly;

//convert year to string

//hour
std::stringstream hr;
hr << setfill('0') << std::setw(2) << theTime.wHour;
string hour = hr.str();
//minute
std::stringstream min;
min << setfill('0') << std::setw(2) << theTime.wMinute;
string minute = min.str();
//second
std::stringstream sec;
sec << setfill('0') << std::setw(2) << theTime.wSecond;
string second = sec.str();

//here we use the year, month, day, hour, minute info to create a custom string
```

```
//this will be displayed in the bottom left corner of our video feed.
//dateTime = year + "-" + month + "-" + day + " " + hour + ":" + minute + ":" + second;
timeonly = hour + ":" + minute + ":" + second;

return timeonly;
}

/* Display time and date on hardcopy */
string getdateonly()
{
//get the system time
SYSTEMTIME theTime;
GetLocalTime(andtheTime);
//create string to store the date and time
string dateonly;

//convert year to string
string year = intToString(theTime.wYear);

//use stringstream to add a leading '0' to the month (ie. 3 -> 03)
//we use 'setw(2)' so that we force the string 2 characters wide with a zero in front of it.
//if the month is '10' then it will remain '10'
std::stringstream m;
m << std::setfill('0') << std::setw(2) << theTime.wMonth;
string month = m.str();
//day
std::stringstream d;
d << std::setfill('0') << std::setw(2) << theTime.wDay;
string day = d.str();

//here we use the year, month, day, hour, minute info to create a custom string
//this will be displayed in the bottom left corner of our video feed.
dateonly = year + "-" + month + "-" + day;

return dateonly;
}

string getDateTimeForFile()
{
//this function does the exact same as "getDateTime()"
//however it returns a string that can be used as a filename
SYSTEMTIME theTime;
GetLocalTime(andtheTime);
string dateTime;

string year = intToString(theTime.wYear);

```

```
std::stringstream m;
m << std::setfill('0') << std::setw(2) << theTime.wMonth;
string month = m.str();

std::stringstream d;
d << std::setfill('0') << std::setw(2) << theTime.wDay;
string day = d.str();

std::stringstream hr;
hr << setfill('0') << std::setw(2) << theTime.wHour;
string hour = hr.str();

std::stringstream min;
min << setfill('0') << std::setw(2) << theTime.wMinute;
string minute = min.str();

std::stringstream sec;
sec << setfill('0') << std::setw(2) << theTime.wSecond;
string second = sec.str();

//here we use "_" instead of "-" and ":"
//if we try to save a filename with "-" or ":" in it we will get an error.
dateTime = year + "_" + month + "_" + day + "_" + hour + "h" + minute + "m" + second + "s";

return dateTime;
}

/* Dispaly time and date above */

/* Motion detection below */
bool detectMotion(Mat thresholdImage, Mat andcameraFeed){
//create motionDetected variable.
bool motionDetected = false;

//create temp Mat for threshold image
Mat temp;
thresholdImage.copyTo(temp);

//these two vectors needed for output of findContours
vector< vector<Point> > contours;
vector<Vec4i> hierarchy;

//find contours of filtered image using openCV findContours function
//findContours(temp,contours,hierarchy,CV_RETR_CCOMP,CV_CHAIN_APPROX_SIMPLE );// retrieves all contours
```

```
findContours(temp, contours, hierarchy, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE);// retrieves external contours

//if contours vector is not empty, we have found some objects
//we can simply say that if the vector is not empty, motion in the video feed has been detected.
if (contours.size()>0)motionDetected = true;
else motionDetected = false;

//find the motion object
if (motionDetected){
//the large contour is found at the end of the contours vector
//we will simply assume that the biggest contour is the object
vector< vector<Point> > largestContourVec;
largestContourVec.push_back(contours.at(contours.size() - 1));

//make a bounding rectangle around the largest contour then find its centroid
//this will be the objects final estimated position
objectBoundingRectangle = boundingRect(largestContourVec.at(0));
int xpos = objectBoundingRectangle.x + objectBoundingRectangle.width / 2;
int ypos = objectBoundingRectangle.y + objectBoundingRectangle.height / 2;


//update the objects position by changing the 'theObject' array values
theObject[0] = xpos, theObject[1] = ypos;

}

return motionDetected;

}
/* Motion detection code above */

//////////////////////////////// CALCULATION OPTICAL FLOW ////////////////////////////////////////////
void drawOptFlowMap(const Matand flow, Matand cflowmap, int step, double, const Scalarand color)
{
for (int y = 0; y < cflowmap.rows; y += step)
for (int x = 0; x < cflowmap.cols; x += step)
{
const Point2fand fxy = flow.at<Point2f>(y, x);

// Displacement direction
line(cflowmap, Point(x, y), Point(cvRound(x + fxy.x), cvRound(y + fxy.y)), color);

// Green dot
circle(cflowmap, Point(x, y), 2, color, -1);
}
}
////////////////////////////////////////////////////////////////////////////////////////////////////
///

void CallBackFunc(int event, int x, int y, int flags, void* userdata)
{
if ( event == EVENT_LBUTTONDOWN )
{
Point p = Point(x,y);
mask.push_back(p);
}
else if (event == EVENT_RBUTTONDOWN)
{
masks.push_back(mask);
mask.clear();
}
else if (event == EVENT_MBUTTONDOWN)
{
mask.clear();
masks.clear();
}

}

#define FILEPATH "D:/" + getDateTimeForFile() + ".csv" //define file stream object

// open the video image //
int main( int argc, const char** argv ) //program entry points
{

/* Declaraction the motion detection code start */

//set recording and startNewRecording initially to false.
bool recording = false;
bool startNewRecording = false;
int inc = 0;
bool firstRun = true;
//if motion is detected in the video feed, we will know to start recording.
bool motionDetected = false;

//pause and resume code (if needed)
bool pause = false;
//set debug mode and trackingenabled initially to false
//these can be toggled using 'd' and 't'
//debugMode = false;
trackingEnabled = false;

```

```
//set up the matrices that we will need
//the two frames we will be comparing
Mat frame1, frame2;
//their grayscale images (needed for absdiff() function)
Mat grayImage1, grayImage2;
//resulting difference image
Mat differenceImage;
//thresholded difference image (for use in findContours() function)
Mat thresholdImage;

Mat prevgray, gray; //OPTICAL FLOW
Mat flow, cflow; //OPTICAL FLOW

Mat src; // Add for image inpuut

/* Declaraction the motion detection code end */

/* Trackbar control threshold value below */
SgSignal signal;
//cvNamedWindow("display", 0);
//createTrackbar( "threshold", "display", andthresholdParameter, 255, NULL );
//setMouseCallback("display", CallBackFunc, NULL);

/* Trackbar control threshold value above*/

VideoCapture cap("C:\\Users\arthur\Desktop\Video Clips\Fire and smoke video clips\fire1.avi");

//VideoCapture cap("C:\\Users\arthur\Documents\My Documents\Desptop folder\Fire video\fire1.avi");
if (!cap.isOpened())
{
printf("Fail to open");
return -1;
}
Mat frame;

while (1)
{
cap >> frame;
namedWindow("video", CV_WINDOW_NORMAL);
imshow("video", frame);
}




/* Start access camera below */
VideoCapture video;
```

```
video.open(0); //Access the webcam.
video >> frame1; //Input the first real time image to frame1

VideoWriter oVideoWriter; //create videoWriter object, not initialized yet (Motion detection code)

/// for realtime video
if (!video.isOpened())
{
std::cout << "ERROR!!! could not access the webcam !!!" << std::endl;
system("PAUSE");
return EXIT_SUCCESS;
//exit(1);
}

// Record the data to file

fstream file;

file.open(FILEPATH, ios::out | ios::trunc);

if (!file)// Check file open or not
{
cerr << "Sorry!!! Can't open file in Hard drive !!!" << endl;
exit(1);
}

double dWidth = video.get(CV_CAP_PROP_FRAME_WIDTH); //get the width of frames1 of the video
double dHeight = video.get(CV_CAP_PROP_FRAME_HEIGHT); //get the height of frames1 of the video

//set framesize for use with videoWriter
Size frameSize(static_cast<int>(dWidth), static_cast<int>(dHeight));

/* Display on command prompt */
cout << "--------------------------------------------------------------------------" << endl;
cout << "VIDEO FIRE DETECTION" << endl;
cout << "Frame Size = " << frameSize << endl;
cout << "Record date " << getdateonly () << endl;
cout << "Start record time " << gettimeonly() << endl;
cout << "OpenCV verson " << CV_VERSION << endl;
cout << "--------------------------------------------------------------------------" << endl;
cout << "Press 'Esc' exit the Code" << endl;
cout << "Press 'p' paused or resume the code" << endl;
cout << "Press 'Space bar' will toggle tracking" << endl;

/* Write on hard disk */
```

```
file << "VIDEO FIRE DETECTION DATA RECORD" << endl;
file << "Frame Size " << "," << frameSize << endl;
file << "Record date " << "," << getdateonly() << endl;
file << "OpenCV verson " << CV_VERSION << endl;
file << " " << endl;

file << setw(0) << "," << setiosflags(ios::right) << "FROM FLAME-INTENSITY" << "," << "," << ","
<< setw(0) << "," << "," << "," << setiosflags(ios::right) << "FROM FLAME-MOTION"
<< endl;

file << setw(0) << setiosflags(ios::right) << "Time" << ","

<< setw(0) << setiosflags(ios::right) << "Otsu Threshold value" << ","

<< setw(0) << setiosflags(ios::right) << "Max. Gray level" << ","

<< setw(0) << setiosflags(ios::right) << "ROI" << ","
<< setw(0) << setiosflags(ios::right) << "Height" << ","
<< setw(0) << setiosflags(ios::right) << "Centroid X" << ","
<< setw(0) << setiosflags(ios::right) << "Centroid Y" << ","

<< setw(0) << setiosflags(ios::right) << "ROI" << ","
<< setw(0) << setiosflags(ios::right) << "Height" << ","
<< setw(0) << setiosflags(ios::right) << "Centroid X" << ","
<< setw(0) << setiosflags(ios::right) << "Centroid Y" << ","

<< setw(0) << setiosflags(ios::right) << "Pixel no. (Red)" << ","
<< setw(0) << setiosflags(ios::right) << "Pixel no. (Green)" << ","
<< setw(0) << setiosflags(ios::right) << "Pixel no. (Blue)"
<< endl;

deque<float> heights,widths,areas,mheight;
double numberFrame = video.get(CV_CAP_PROP_FRAME_COUNT);
int countFrame = 0;
float lengthTrajectory = 10.0f;
float sizeWindow = 1920.0f;

while (true) // Infinite Loop
{
if (!pause) video >> frame1;
countFrame++;
if (countFrame == numberFrame - 1)
{
video.set(CV_CAP_PROP_POS_FRAMES, 1);
countFrame = 1;
continue;
}
```

```

//localize the fire
Mat image, grayimage, grayimage2, fireMask, display, graph;

frame1.copyTo(display);

/*Motion detection code start*/

//read first frame
video.read(frame1);

/////////////////////////////////////////////////////////////////////////////////////////////////////

//convert frame1 to gray scale for frame differencing
cv::cvtColor(frame1, grayImage1, COLOR_BGR2GRAY);

//copy second frame
video.read(frame2);

//convert frame2 to gray scale for frame differencing
cv::cvtColor(frame2, grayImage2, COLOR_BGR2GRAY);

//////////////////////////////////////////////////////////////////////////////////////////////////////

//perform frame differencing with the sequential images. This will output an "intensity image"
//do not confuse this with a threshold image, we will need to perform thresholding afterwards.
cv::absdiff(grayImage1, grayImage2, differenceImage); // COMPARE THE DIFF. for motion detection

//threshold intensity image at a given sensitivity value
cv::threshold(differenceImage, thresholdImage, SENSITIVITY_VALUE, 255, THRESH_BINARY); // threshold --> SENSITIVITY VALUE = 40

/*if (debugMode == true){
//show the difference image and threshold image
cv::imshow("Difference Image", differenceImage);
cv::imshow("Threshold Image", thresholdImage);
}
else{
//if not in debug mode, destroy the windows so we don't see them anymore
cv::destroyWindow("Difference Image");
cv::destroyWindow("Threshold Image");
}*/

```

```
//blur the image to get rid of the noise. This will output an intensity image
cv::blur(thresholdImage, thresholdImage, cv::Size(BLUR_SIZE, BLUR_SIZE)); // Dilate and Erode

//threshold again to obtain binary image from blur output
cv::threshold(thresholdImage, thresholdImage, SENSITIVITY_VALUE, 255, THRESH_BINARY);
//Sensitivity value is 40 (threshold)


/*if (debugMode == true){
//show the threshold image after it's been "blurred"

imshow("Final Threshold Image", thresholdImage);

}
else {
//if not in debug mode, destroy the windows so we don't see them anymore
cv::destroyWindow("Final Threshold Image");
}*/

//if tracking enabled, search for Motion
if (trackingEnabled){
//detectMotion function will return true if motion is detected, else it will return false.
//set motionDetected boolean to the returned value.
motionDetected = detectMotion(thresholdImage, frame1);
}
else{
//reset our variables if tracking is disabled
recording = false;
motionDetected = false;
}
/*Motion detection code end*/

/*Motion detection code start*/
//if we're in recording mode, write to file
if (recording){

oVideoWriter.write(frame1);
//show "REC" in red
//be sure to do this AFTER you write to the file so that "REC" doesn't show up
//on the recorded video.
circle(display, Point(500, 20), 12.0, Scalar(0, 0, 255), -1, 8);
putText(display, "REC", Point(515, 28), 2, 1, Scalar(0, 0, 255), 2);

}
if (motionDetected){
//show "MOTION DETECTED" in bottom left corner in green
```

```
//once again, be sure to do this AFTER you write to the file so that "MOTION DETECTED" doesn't show up
//on the recorded video. Place this code above if(recording) to see what I'm talking about.
putText(display, "MOTION DETECTED", cv::Point(0, 420), 2, 1, cv::Scalar(0, 255, 0), 2);

//set recording to true since there is motion in the video feed.
recording = true;

if (firstRun == true){

string videoFileName = "D:/" + getDateTimeForFile() + ".avi";
cout << "File has been opened for writing: " << videoFileName << endl;
oVideoWriter = VideoWriter(videoFileName, CV_FOURCC('D', 'I', 'V', '3'), 20, frameSize, true);

if (!oVideoWriter.isOpened())
{
cout << "ERROR!!! Failed to initialize video writing in Hard drive !!!" << endl;
getchar();
return -1;
}
firstRun = false;

}

}
else recording = false;
/*Motion detection code end*/

cvtColor(frame2, grayimage, CV_BGR2GRAY); // Convert to gratscale and store in "grayimage" ORIGINAL DATA FROM WEBCAM

// Calculation the gray values (IN PROGRESS)
double alpha = 0.2989, Beta = 0.5870, Gamma = 0.1140; // For calculation !!
vector<Mat> bgr_planesforgray;
split(frame2, bgr_planesforgray);
/*
float hist_val[256];
for (int i = 0; i < 256; i++)
{
hist_val[i] = 0.0;
//cout << i << " " << hist_val[i] << endl;

```

```
584 }
585 */
586 /* Analysis for captured the Region of Interest */
587 cvtColor(frame2,image,CV_BGR2GRAY); // Convert to grayscale and store in "image" for analysis
588
589 // Smooths an image using the Gaussian filter (Dilate and Erode)
590 GaussianBlur(image,image,Size(BLUR_SIZE_INTENSITY,BLUR_SIZE_INTENSITY),0,0);
591 //blur(image, image, Size(BLUR_SIZE, BLUR_SIZE)); // BLUR_SIZE = 10 reference: line no. 32
592
593
594 // Otsu calculation (IN PROGRESS)
595 /*
596 float wB; //Weight Background
597 float wF; //Weight Foreground
598 float mB; //Mean Background
599 float mF; //Mean Foreground
600 float varBetween; //Between Class Variance
601 float varMax; //Maximum Between Class Variance
602 */
603 // Total number of pixels
604
605
606 // Calculate Weight background
607
608
609 //threshold(image, fireMask, thresholdParameter, 255, cv::THRESH_BINARY);
610
611
612 // Optimal threshold value obtained by Otsu algorithm
613 thresholdParameter = threshold(image, fireMask, 0, 255, cv::THRESH_BINARY | cv::THRESH_OTSU);
// WARNING double to int problem ??
614
615 fillPoly(fireMask,masks,cvScalarAll(0));
616 bitwise_and(image, fireMask, image);
617
618 /*
619 /// Apply Histogram Equalization
620 equalizeHist(image,image);
621
622 Mat dst;
623 equalizeHist(image, dst);
624
625 /// Display results
```

```
namedWindow("equalized_window", CV_WINDOW_AUTOSIZE);
imshow("equalized_window", dst);
*/

//resize the user interface
/*
resizeWindow("input", 450, 350);
resizeWindow("display", 450, 350);
cvResizeWindow("rgb and gray_Hist", 450, 350);
resizeWindow("data", 450, 350);

//resizeWindow("gray", 240, 120);
//cvResizeWindow("Threshold by Otsu", 240, 120);
//cvResizeWindow("Diff. by motion", 240, 120);
//cvResizeWindow("Threshold by motion", 240, 120);

//cvResizeWindow("Gray Hist", 320, 240);
//cvResizeWindow("Red Hist", 320, 200);
//cvResizeWindow("Green Hist", 320, 200);
//cvResizeWindow("Blue Hist", 320, 200);

//resizeWindow("Motion", 800, 100);

//resizeWindow("flow", 320, 240);
*/

//FIXED LOCATION OF WINDOWS
/*
cvMoveWindow("input", 10, 0);
cvMoveWindow("display", 10, 400);
cvMoveWindow("rgb and gray_Hist", 500, 0);
cvMoveWindow("data", 500, 400);

cvMoveWindow("Gray Hist", 640, 0);
cvMoveWindow("flow", 960, 0);
cvMoveWindow("Threshold by Otsu", 1280, 160);
cvMoveWindow("gray", 1280, 0);
cvMoveWindow("Threshold by Otsu", 1280, 160);
cvMoveWindow("Diff. by motion", 1280, 320);
cvMoveWindow("Threshold by motion", 1280, 480);

cvMoveWindow("Red Hist", 640, 280);
cvMoveWindow("Green Hist", 960, 280);
cvMoveWindow("Blue Hist", 960, 560);

cvMoveWindow("Motion", 0, 560);
*/

cvNamedWindow("Threshold by Otsu", WINDOW_NORMAL);
cvNamedWindow("Diff. by motion", WINDOW_NORMAL);
cvNamedWindow("Threshold by motion", WINDOW_NORMAL);

cv::imshow("Threshold by Otsu", fireMask);


cv::imshow("Diff. by motion",differenceImage);
cv::imshow("Threshold by motion",thresholdImage);

//get contours
vector<vector<Point>> contourFires;
vector<Vec4i> hierarchy;
findContours(fireMask,contourFires,hierarchy,CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE);
vector<Point> contourMax;
SgGeneral::sgGetMaxContour(contourFires,20,contourMax);
if (!contourMax.empty())
{
Point centroid = SgGeneral::sgGetCentroid(contourMax);

centroids.push_back(centroid);

while (centroids.size() > lengthTrajectory) centroids.pop_front();
}

float area = SgGeneral::sgGetArea(contourFires);
areas.push_back(area);
if (areas.size() > sizeWindow) areas.pop_front();

//calculate bounding box of multiple contours
int yMin = 999, yMax = 0, xMax = 0, xMin = 999;

Point peak;
for( int i = 0; i < contourFires.size(); i++ )
{
if (contourFires[i].size() > 20)
{
for( int j = 0; j < contourFires[i].size(); j++ )
{
if (contourFires[i][j].y > yMax) yMax = contourFires[i][j].y;
else if(contourFires[i][j].y < yMin)
{
yMin = contourFires[i][j].y;
peak.x = contourFires[i][j].x;
peak.y = contourFires[i][j].y;
}
if (contourFires[i][j].x > xMax) xMax = contourFires[i][j].x;
else if(contourFires[i][j].x < xMin) xMin = contourFires[i][j].x;
}
```

```
}
}

// Input the informaton data
heights.push_back(yMax-yMin); // WARNING int to float problem ??

widths.push_back(xMax-xMin); // WARNING int to float problem ??
mheight.push_back(objectBoundingRectangle.height); // NEW Pushback motion height to "mheight"

if (heights.size() > sizeWindow)
{
heights.pop_front();
widths.pop_front();
mheight.pop_front();
}

//visualize ??
polylines(display,masks,true,CV_RGB(0,0,128),1);
for (int j = 0; j < mask.size(); j++) circle(display,mask[j],2,CV_RGB(255,0,0),-1);

//mask the segmentation region
/*Mat rgb[3];
split(display,rgb);
rgb[0] += image; // Blue
rgb[1] += image; // Green
rgb[2] += image; // Red
merge(rgb,3,display);
*/

if (!contourFires.empty())
for( int i = 0; i < contourFires.size(); i++ )
drawContours(display,contourFires,i,CV_RGB(0,0,0), 2); // draw contout line in display

//float scale = 255. / lengthTrajectory;
double scale = 255. / lengthTrajectory;
if(centroids.size() > 2)
{
for (int i = 0; i < centroids.size() - 1; i++)
{
circle(display, Point(centroids.back().x, centroids.back().y), 7, Scalar(255, 0, 0), 2); //draw centroid

line(display, Point(centroids.back().x, centroids.back().y), Point(centroids.back().x, centroids.back().y - 15),
Scalar(255, 0, 0), 2);
line(display, Point(centroids.back().x, centroids.back().y), Point(centroids.back().x, centroids.back().y + 15),
Scalar(255, 0, 0), 2);
line(display, Point(centroids.back().x, centroids.back().y), Point(centroids.back().x
- 15, centroids.back().y),
Scalar(255, 0, 0), 2);
line(display, Point(centroids.back().x, centroids.back().y), Point(centroids.back().x
+ 15, centroids.back().y),
Scalar(255, 0, 0), 2);

//line(display, centroids[i], centroids[i+1], CV_RGB(0, scale*i, 0), 2);
}
}

circle(display,centroids.back(),3,CV_RGB(0,255,0),-1); // ????

//make some temp x and y variables
int x = theObject[0];
int y = theObject[1];

//draw some crosshairs (motion detection)

circle(display, Point(x, y), 7, Scalar(0, 255, 0), 2);

line(display, Point(x, y), Point(x, y - 15), Scalar(0, 255, 0), 2);
line(display, Point(x, y), Point(x, y + 15), Scalar(0, 255, 0), 2);
line(display, Point(x, y), Point(x - 15, y), Scalar(0, 255, 0), 2);
line(display, Point(x, y), Point(x + 15, y), Scalar(0, 255, 0), 2);

// drawing region of interest (ROI) based on xMin,yMin,xMax,yMax (Threshold
analysis)
rectangle(display, Point(xMin, yMin), Point(xMax, yMax), CV_RGB(255, 255, 0),
2); //
threshold analysis

rectangle(display, Point(objectBoundingRectangle.x,
objectBoundingRectangle.y),
Point(objectBoundingRectangle.x + objectBoundingRectangle.width,
objectBoundingRectangle.y
+ objectBoundingRectangle.height),
CV_RGB(255, 0, 0), 2);

//draw time stamp to video in bottom left corner. We draw it before we write so
that it is
written on the video file.
rectangle(display, Point(0, 460), Point(200, 480), Scalar(255, 255, 255), -1);
putText(display, getDateTime(), Point(0, 475), 1, 1, Scalar(0, 0, 0), 2); //show
date time on 'display'
rectangle(frame2, Point(0, 460), Point(200, 480), Scalar(255, 255, 255), -1);
putText(frame2, getDateTime(), Point(0, 475), 1, 1, Scalar(0, 0, 0), 2); //show
date time on 'frame'
```

```
rectangle(grayimage, Point(0, 460), Point(200, 480), Scalar(255, 255, 255), -1);
putText(grayimage, getDateTime(), Point(0, 475), 1, 1, Scalar(0, 0, 0), 2); //show date time on 'grayimage'

//draw the height, width and area region of interest
string heightFire = "ROI Height (Intnesity) : " + to_string(yMax - yMin);
//putText(display, heightFire, Point(40, 20), 1, 1, CV_RGB(0, 0, 255), 2);

string widthFire = "ROI Width (Intensity) : " + to_string(xMax - xMin);
//putText(display, widthFire, Point(40, 40), 1, 1, CV_RGB(0, 0, 255), 2);

string ROIFire = "ROI Area (Intensity) : " + to_string((yMax - yMin)*(xMax - xMin));
//putText(display, ROIFire, Point(40, 60), 1, 1, CV_RGB(0, 0, 255), 2);

string fireThreshold = "Otsu : " + to_string(thresholdParameter); // Intensity threshold
//putText(display, fireThreshold, Point(320, 20), 1, 1, CV_RGB(255, 255, 255), 2);

//draw the centroid coordinate X and Y
string cenFire = "X,Y (Intensity) : ";
if (!centroids.empty())
cenFire = "X,Y (Intensity) : " + to_string(centroids.back().x) + "," + to_string(centroids.back().y);
//putText(display, cenFire, Point(40, 80), 1, 1, CV_RGB(0, 0, 255), 2);

string movcenFire = "X,Y (Motion) : " + to_string(x)+","+to_string(y);
//putText(display, movcenFire, Point(40, 100), 1, 1, CV_RGB(255, 0, 0), 2);

string ROIMovFire = "ROI Area (Motion) : " + to_string(objectBoundingRectangle.width * objectBoundingRectangle.height);
//putText(display, ROIMovFire, Point(40, 120), 1, 1, CV_RGB(255, 0, 0), 2);

string ROIMovheightFire = "ROI Height (Motion) : " + to_string(objectBoundingRectangle.height);
//putText(display, ROIMovheightFire, Point(40, 140), 1, 1, CV_RGB(255, 0, 0), 2);

string ROIMovwidthFire = "ROI Width (Motion) : " + to_string(objectBoundingRectangle.width);
//putText(display, ROIMovwidthFire, Point(40, 160), 1, 1, CV_RGB(255, 0, 0), 2);


/*Histogram analysis*/

/// Separate the image in 3 places ( B, G and R )
vector<Mat> bgr_planes;
split(frame2, bgr_planes);
```

```

// Separate the image in gray places
vector<Mat> gray_planes;
split(image, gray_planes);

vector<Mat> grayplanes;
split(grayimage, grayplanes);

/// Establish the number of bins
int histSize = 256; //From 0 to 255

/// Set the ranges (for B,G,R and gray)
float range[] = { 0, 256 }; //the upper boundary is exclusive
const float* histRange = { range };

bool uniform = true; bool accumulate = false;

Mat b_hist, g_hist, r_hist, gray_hist;

/// Compute the histograms:
calcHist(andbgr_planes[0], 1, 0, Mat(), b_hist, 1, andhistSize, andhistRange,
uniform, accumulate);
calcHist(andbgr_planes[1], 1, 0, Mat(), g_hist, 1, andhistSize, andhistRange,
uniform, accumulate);
calcHist(andbgr_planes[2], 1, 0, Mat(), r_hist, 1, andhistSize, andhistRange,
uniform, accumulate);

calcHist(andgrayplanes[0], 1, 0, Mat(), gray_hist, 1, andhistSize, andhistRange,
uniform,
accumulate);

// Draw the histograms for B, G and R
int hist_w = 512; int hist_h = 400;
int bin_w = cvRound((double)hist_w / histSize); // int histSize = 256

Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));

Mat R_histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
Mat G_histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
Mat B_histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));

Mat grayhistImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));

/// Normalize the result to [ 0, histImage.rows ]
normalize(b_hist, b_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
normalize(g_hist, g_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
normalize(r_hist, r_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());

normalize(b_hist, b_hist, 0, B_histImage.rows, NORM_MINMAX, -1, Mat());
normalize(g_hist, g_hist, 0, G_histImage.rows, NORM_MINMAX, -1, Mat());
normalize(r_hist, r_hist, 0, R_histImage.rows, NORM_MINMAX, -1, Mat());

normalize(gray_hist, gray_hist, 0, grayhistImage.rows, NORM_MINMAX, -1, Mat()); //Normalize
the gray result

/// Draw for each channel
int gray_maxtemp = 0;
int red_maxtemp = 0;
int green_maxtemp = 0;

for (int i = 0; i < histSize; i++)
{
line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(b_hist.at<float>(i - 1))),
Point(bin_w*(i), hist_h - cvRound(b_hist.at<float>(i))),
Scalar(255, 0, 0), 0, 8, 0);
// Blue colour

line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(g_hist.at<float>(i - 1))),
Point(bin_w*(i), hist_h - cvRound(g_hist.at<float>(i))),
Scalar(0, 255, 0), 0, 8, 0);
// Green colour

line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(r_hist.at<float>(i - 1))),
Point(bin_w*(i), hist_h - cvRound(r_hist.at<float>(i))),
Scalar(0, 0, 255), 0, 8, 0);
// Red colour

line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(gray_hist.at<float>(i - 1))),
// Gray color
Point(bin_w*(i), hist_h - cvRound(gray_hist.at<float>(i))),
Scalar(255, 255, 255), 0, 8, 0);

/* Individual histogram */
line(B_histImage, Point(bin_w*(i - 1), hist_h),
Point(bin_w*(i), hist_h - cvRound(b_hist.at<float>(i))),
Scalar(255, 0, 0), 0, 8, 0); //
Blue colour

line(G_histImage, Point(bin_w*(i - 1), hist_h),
Point(bin_w*(i), hist_h - cvRound(g_hist.at<float>(i))),
Scalar(0, 255, 0), 0, 8, 0); //
Green colour

line(R_histImage, Point(bin_w*(i - 1), hist_h),
Point(bin_w*(i), hist_h - cvRound(r_hist.at<float>(i))),
Scalar(0, 0, 255), 0, 8, 0); //
Red colour

line(grayhistImage, Point(bin_w*(i - 1), hist_h), // Gray color
Point(bin_w*(i), hist_h - cvRound(gray_hist.at<float>(i))),
Scalar(255, 255, 255), 0, 8, 0);

// Indicate the threshold
line(grayhistImage, Point(bin_w*(thresholdParameter), hist_h),
Point(bin_w*(thresholdParameter), hist_h - cvRound(gray_hist.at<float>(i))),
Scalar(0, 255, 255), 0, 4, 0);

line(histImage, Point(bin_w*(thresholdParameter), hist_h),
Point(bin_w*(thresholdParameter), hist_h - cvRound(gray_hist.at<float>(i))),
Scalar(0, 255, 255), 0, 8, 0);

/* Analysis the maximim gray level */
float gray_binVal = gray_hist.at<float>(i);
if (gray_hist.at<float>(i) > gray_maxtemp)
{
gray_maxtemp = i;
}

}

//display on grayhistImage
string maxgray = "Max Gray Level : " + to_string(gray_maxtemp);
putText(grayhistImage, maxgray, Point(40, 20), 1, 1, CV_RGB(255, 255, 255), 2);

string threshold_value = "Threshold by Otsu : " + to_string(thresholdParameter);
//putText(grayhistImage, threshold_value, Point(40, 40), 1, 1, CV_RGB(255, 255, 0), 2);

//diaply on histImage
string threshold_value_hist = "Threshold by Otsu : " + to_string(thresholdParameter);
putText(histImage, threshold_value_hist, Point(40, 20), 1, 1, CV_RGB(255, 255, 0), 2);

string nosofpixels_red = "Pixels no.(R) : " + to_string(r_hist.at<float>(thresholdParameter));
putText(histImage, nosofpixels_red, Point(40, 40), 1, 1, CV_RGB(255, 0, 0), 2);
string pixelsno_red = to_string((int)r_hist.at<float>(thresholdParameter));

string nosofpixels_green = "Pixels no.(G) : " + to_string(g_hist.at<float>(thresholdParameter));
putText(histImage, nosofpixels_green, Point(40, 60), 1, 1, CV_RGB(0, 255, 0), 2);
string pixelsno_green = to_string((int)g_hist.at<float>(thresholdParameter));

string nosofpixels_blue = "Pixels no.(B) : " + to_string(b_hist.at<float>(thresholdParameter));
putText(histImage, nosofpixels_blue, Point(40, 80), 1, 1, CV_RGB(0, 0, 255), 2);
string pixelsno_blue = to_string((int)b_hist.at<float>(thresholdParameter));

file << setw(1) << setiosflags(ios::right) << gettimeonly() << "," //date and time
<< setw(4) << setiosflags(ios::right) << thresholdParameter << "," //threhsold Otsu
<< setw(5) << setiosflags(ios::right) << gray_maxtemp << "," //Max. gray level
<< setw(8) << setiosflags(ios::right) << (yMax - yMin)*(xMax - xMin) << "," //ROI
<< setw(4) << setiosflags(ios::right) << (yMax - yMin) << "," //Region Height
<< setw(4) << setiosflags(ios::right) << centroids.back().x <<"," //Centroid (Intensity)
<< setw(3) << setiosflags(ios::right) << centroids.back().y << "," //Centroid (Intensity)
<< setw(5) << setiosflags(ios::right) << x <<"," //Centroid X (Motion)
<< setw(3) << setiosflags(ios::right) << y << "," //Centroid Y (Motion)
<< setw(5) << setiosflags(ios::right) << objectBoundingRectangle.height << "," //motion object height
<< setw(7) << setiosflags(ios::right) << objectBoundingRectangle.width * objectBoundingRectangle.height << "," //ROI (Motion)
<< setw(7) << setiosflags(ios::right) << pixelsno_red << "," //red color pixels in threshold Otsu
<< setw(7) << setiosflags(ios::right) << pixelsno_green << "," //green color pixels in threshold Otsu
<< setw(7) << setiosflags(ios::right) << pixelsno_blue << "," //blue color pixels in threshold Otsu
<< endl;


//display the output on screen

string text1 = "Developed by : Arthur Wong" ;

int fontFace = FONT_HERSHEY_SIMPLEX;
double fontScale = 0.8;
int thickness = 2.0;

Mat img(300, 900, CV_8UC3, Scalar::all(0));

putText(img, string("VIDEO FIRE DETECTION"), Point(20, 20), fontFace, fontScale, Scalar::all
(255), thickness, 8);

putText(img, threshold_value, Point (20, 45), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, nosofpixels_red, Point(20, 70), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, nosofpixels_green, Point(20, 95), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, nosofpixels_blue, Point(20, 120), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, heightFire, Point(20, 145), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, widthFire, Point(20, 170), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, cenFire, Point(20, 195), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, ROIFire, Point(20, 220), fontFace, fontScale, Scalar::all(255), thickness, 8);

putText(img, ROIMovheightFire, Point(500, 45), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, ROIMovwidthFire, Point(500, 70), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, movcenFire, Point(500, 95), fontFace, fontScale, Scalar::all(255), thickness, 8);
putText(img, ROIMovFire, Point(500, 120), fontFace, fontScale, Scalar::all(255), thickness, 8);

putText(img, text1, Point (500, 280), fontFace, fontScale, Scalar::all(255), thickness, 8);

namedWindow("data", WINDOW_NORMAL);
imshow("data", img);



/***************************************************************************************************************************************************/
cvNamedWindow("input", WINDOW_NORMAL);
cvNamedWindow("gray", WINDOW_NORMAL);
cvNamedWindow("display",WINDOW_NORMAL);

```

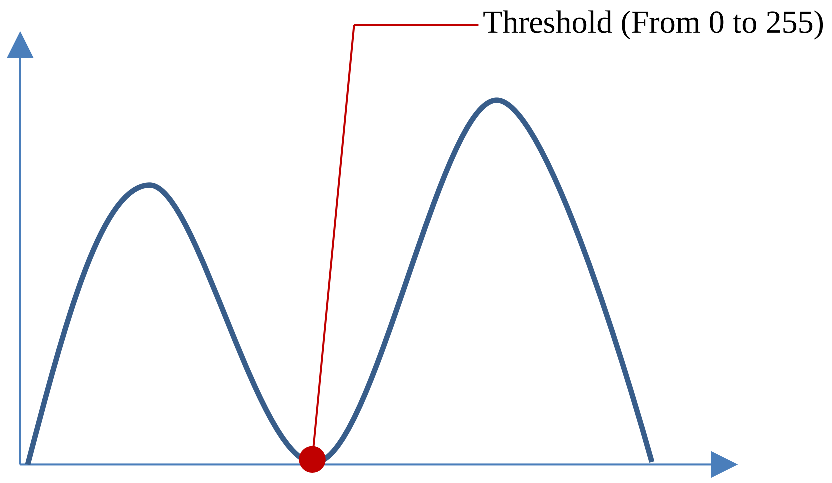
```
cv::imshow("input",frame2); // display input video image
cv::imshow("gray",grayimage); // display gray video image
cv::imshow("display",display); // display final video image

/// Display histogram
cvNamedWindow("Gray Hist", WINDOW_NORMAL);

cvNamedWindow("Red Hist", WINDOW_NORMAL);
cvNamedWindow("Green Hist", WINDOW_NORMAL);
cvNamedWindow("Blue Hist", WINDOW_NORMAL);

cvNamedWindow("rgb and gray_Hist", WINDOW_NORMAL);

cv::imshow("Red Hist", R_histImage);
cv::imshow("Green Hist", G_histImage);
cv::imshow("Blue Hist", B_histImage);
cv::imshow("Gray Hist", grayhistImage);
cv::imshow("rgb and gray_Hist", histImage);

if (widths.size() > 1) // SgSignal.cpp line no 332
{

//vector<float> tempH;
vector<float> motionheight;
//vector<float> tempW;
//vector<float> tempA;

//signal.sgDequeToVector(widths,tempW);
//signal.sgDequeToVector(heights, tempH);
signal.sgDequeToVector(mheight,motionheight);
//signal.sgDequeToVector(areas,tempA);

//signal.sgNormalizeByMinMax(tempW,tempW,0,1000); //original 300 red
//signal.sgNormalizeByMinMax(tempH, tempH, 0, 1000); //original 300 green
signal.sgNormalizeByMinMax(motionheight, motionheight, 0, 300);
//signal.sgNormalizeByMinMax(tempA,tempA,0,30000);

Mat graph = Mat::zeros(200, 1920, CV_8UC3);
//signal.sgDraw01(graph, CV_RGB(255, 255, 255), tempH, 1);
signal.sgDraw01(graph, CV_RGB(255, 255, 255), motionheight, 1);
//signal.sgDraw01(graph,CV_RGB(255,0,0),tempW,1);


//calculation and display on chart
int intensityflameheight = (yMax - yMin);
int motionflameheight = (objectBoundingRectangle.height);


```

```
//string intflicker = "flame height (intensity) : " + to_string(intensityflameheight);
//putText(graph, intflicker, Point(40, 60), 1, 1, CV_RGB(255, 255, 255), 2);

string motflicker = "Object motion (height) : " + to_string(motionflameheight);
putText(graph, motflicker, Point(40, 20), 1, 1, CV_RGB(255, 255, 255), 2);

cvNamedWindow("Motion", WINDOW_NORMAL);
cv::imshow("Motion", graph);


}

//////////////////////////////// CALCULATION OPTICAL FLOW /////////////////////////////

namedWindow("flow", WINDOW_NORMAL);
//namedWindow("realtime input", 1);

video.read(frame2);
cvtColor(frame2, gray, COLOR_BGR2GRAY); //

//imshow("realtime input", frame2);
imshow("flow", gray);

if (prevgray.data)
{
calcOpticalFlowFarneback(prevgray, gray, flow, 0.5, 3, 15, 3, 5, 1.2, 0);

cvtColor(prevgray, cflow, COLOR_GRAY2BGR);
drawOptFlowMap(flow, cflow, 16, 1.5, Scalar(0, 255, 0));

imshow("flow", cflow);
}

////////////////////////////////////////////////////////////////////////////////////

/*
int c = cvWaitKey(30);
if(c == 27) break;
else if (c == 32) pause = !pause;
else if (char(c) == '+') for (int i = 0; i < 30*5; i++) video.grab();
else if (char(c) == 'n') video >> frame;
*/

switch (waitKey(30))
{
case 27: //'Esc' has been pressed. this wil exit the code.
file << "END" << endl;
file.close();
return 0;
```

```

case 32: //'Space bar' has been pressed. this will toggle tracking
trackingEnabled = !trackingEnabled;
if (trackingEnabled == false) cout << "Tracking disabled." << endl;
else cout << "Tracking enabled." << endl;
break;

case 112: //'p' has been pressed. this will pause/resume the code.
pause = !pause;
if (pause == true)
{
cout << "Code paused, press 'p' again to resume" << endl;
while (pause == true)
{
//stay in this loop until
switch (waitKey())
{
//a switch statement inside a switch statement? Mind blown.
case 112:
//change pause back to false
pause = false;
cout << "Code Resumed" << endl;
break;


}
}

}

}
swap(prevgray, gray); // OPTICAL FLOW
}
//cvDestroyWindow("fire");

return 0; // Exit the program

}
```

Appendix C: Otsu method formulation (Otsu, 1979)

In an ideal case, if the histogram has two peaks and one deep valley, the threshold can be selected from the deep valley. Generally, two peaks represent the object and background. The deep valley represents the threshold.



However, for most of real situation, the histogram is difficult to select the valley bottom precisely so the threshold is also difficult to elevate from the grey level histogram. The Otsu method can be used to overcome these difficulties when the histogram is not in an ideal case.

Otsu method is aimed in selection the threshold value k^* from the calculation result of maximum between class variance $\left(\max_{1 \le k \le L} \sigma_B^2(k)\right)$.

Letter N is the total number of pixel and Letter L is represented the grey levels, the range is from 0-255.

$$N = n_1 + n_2 + n_3 + \cdots\cdots\cdots\cdots\cdots\cdots + n_{L-1} + n_L$$

$$P_i = \frac{n_i}{N} \qquad\qquad P_i \geq 0, \sum_{i=1}^{L} P_i = 1$$

Classes: C_0 , C_1

C_0: Background: level $[1 \cdots\cdots\cdots\cdots k]$

C_1: Objects: level $[k+1 \ \cdots\cdots\cdots L]$

Threshold level: k

Probability

$$\omega_0 = p(C_0) = \sum_{i=1}^{k} p_i = \omega(k)$$

$$\omega_1 = p(C_1) = \sum_{i=k+1}^{L} p_i = \left(1 - \omega(k)\right)$$

Mean

$$\mu_0 = \sum_{i=1}^{k} i\, P(i|C_0) = \sum_{i=1}^{k} \frac{ip_i}{\omega_0} = \frac{\mu(k)}{\omega(k)}$$

$$\mu_1 = \sum_{i=k+1}^{L} iP(i|C_1) = \sum_{i=k+1}^{L} \frac{ip_i}{\omega_1} = \frac{\mu_T - \mu(k)}{1 - \omega(k)}$$

$$\Rightarrow \omega_0\omega_1 = \mu(k)$$

$$\Rightarrow \omega_1\mu_1 = \mu_T - \omega_0\mu_o$$

$$\Rightarrow \mu_T = \omega_1\mu_1 + \omega_0\mu_0$$

Since $\omega_0 = \omega(k)$ $\qquad$ $\omega_1 = 1 - \omega(k)$

$$\Rightarrow \omega_0 + \omega_1 = 1$$

Variance

$\sigma_B^2 = \omega_1(\mu_1 - \mu_T)^2 + \omega_0(\mu_0 - \mu_T)^2$

$= \omega_1(\mu_1^2 - 2\mu_1\mu_T + \mu_T^2) + \omega_0(\mu_0^2 - 2\mu_0\mu_T + \mu_T^2)$

$= \omega_1\mu_1^2 - 2\omega_1\mu_1\mu_T + \omega_1\mu_T^2 + \omega_0\mu_0^2 - 2\omega_0\mu_0\mu_T + \omega_o\mu_T^2$

$= \omega_1\mu_1^2 + \omega_0\mu_0^2 - (2\omega_1\mu_1\mu_T + 2\omega_0\mu_0\mu_T) + \omega_1\mu_T^2 + \omega_o\mu_T^2$

$= \omega_1\mu_1^2 + \omega_0\mu_0^2 - (2\omega_1\mu_1\mu_T + 2\omega_0\mu_0\mu_T) + (\omega_1 + \omega_0)\mu_T^2$

$\omega_1 + \omega_0 = 1$

$= \omega_1\mu_1^2 + \omega_0\mu_0^2 - 2\mu_T(\omega_1\mu_1 + \omega_0\mu_0) + \mu_T^2$

$\omega_1\mu_1 + \omega_0\mu_0 = \mu_T$

$= \omega_1\mu_1^2 + \omega_0\mu_0^2 - 2\mu_T\mu_T + \mu_T^2$

$= \omega_1\mu_1^2 + \omega_0\mu_0^2 - \mu_T^2$

$\sigma_B^2 = \left(\sum_{i=1}^{L} \omega_i\,\mu_i^2\right) - \mu_T^2$

$\mu_T^2 = (\omega_0\mu_0 + \omega_1\mu_1)^2$

$= (\omega_0\mu_0 + \omega_1\mu_1)\,(\omega_0\mu_0 + \omega_1\mu_1)$

$= \omega_0^2\mu_0^2 + \omega_1\omega_0\mu_1\mu_0 + \omega_1\omega_0\mu_1\mu_0 + \omega_1^2\mu_1^2$

$= \omega_0^2\mu_0^2 + 2\omega_1\omega_0\mu_1\mu_0 + \omega_1^2\mu_1^2$

$\sigma_B^2 = \omega_1\mu_1^2 + \omega_0\mu_0^2 - \mu_T^2$

$\sigma_B^2 = \omega_1\mu_1^2 + \omega_0\mu_0^2 - (\omega_0^2\mu_0^2 + 2\omega_1\omega_0\mu_1\mu_0 + \omega_1^2\mu_1^2)$

$\sigma_B^2 = \omega_0\mu_0^2 + \omega_1\mu_1^2 - \omega_0^2\mu_0^2 - 2\omega_1\omega_0\mu_1\mu_0 - \omega_1^2\mu_1^2$

$\sigma_B^2 = (\omega_0 - \omega_0^2)\mu_0^2 - 2\omega_1\omega_0\mu_1\mu_0 + (\omega_1 - \omega_1^2)\mu_1^2$

$\sigma_B^2 = \omega_0(1-\omega_0)\mu_0^2 - 2\omega_1\omega_0\mu_1\mu_0 + \omega_1(1-\omega_1)\mu_1^2$

$\omega_0 = 1-\omega_1\ ;\ \omega_1 = 1-\omega_0$

$\sigma_B^2 = \omega_0\omega_1\mu_0^2 - 2\omega_1\omega_0\mu_1\mu_0 + \omega_1\omega_0\mu_1^2$

$\sigma_B^2 = \omega_0\omega_1(\mu_0^2 - 2\mu_1\mu_0 + \mu_1^2)$

$\sigma_B^2 = \omega_0\omega_1(\mu_0-\mu_1)^2$

$\sigma_B = \omega_0\omega_1(\mu_1-\mu_0)^2$

The optimal threshold value k* is obtained from maximum between class variance σ_B^2.

The range is from 0 to 255

$$\sigma_B^2(k^*) = \max_{0\le k\le L} \sigma_B^2(k)$$

Since

$\sigma^2 = \sigma_w^2 + \sigma_B^2$

$\sigma^2 = \sigma_w^2 + (\omega_1)(1-\omega_1)(\mu_0-\mu_1)^2$

Or

$\sigma^2 = \sigma_w^2 + (\omega_0)(1-\omega_0)(\mu_0-\mu_1)^2$

Multithreshold analysis

For the selected multithreshold$(k, k_1, k_2 \ldots \ldots \ldots, k_n)$, the method to multithresholding is straightforward to calculate the maximum between class variance. The multithreshold can be segmented the fire image succefully, depending on the quantity of threshold.

1st threshold is k (Single threshold value)

2nd threshold is k_1

3rd thrrehsold is k_2

.

.

.

n th threshold is k_n

where n is the quantity of threshold from 0 to 255

Probabillity of Multithreshold

$$\omega_0 = \mathrm{p}(\mathrm{C}_0) = \sum_{\mathrm{i=k}}^{k_n} \mathrm{p_i} = \omega\,(k_n)$$

$$\omega_1 = \mathrm{p}(\mathrm{C}_1) = \sum_{\mathrm{i}=k_n+1}^{\mathrm{L}} \mathrm{p_i} = \left(1 - \omega\,(k_n)\right)$$

Mean of Multithreshold

$$\mu_0 = \sum_{\mathrm{i=k}}^{k_n} \mathrm{i}\,\mathrm{P}(\mathrm{i}|\mathrm{C}_0) = \sum_{\mathrm{i=k}}^{k_n} \frac{\mathrm{ip_i}}{\omega_0} = \frac{\mu\,(k_n)}{\omega\,(k_n)}$$

$$\mu_1 = \sum_{\mathrm{i}=k_n+1}^{\mathrm{L}} \mathrm{iP}(\mathrm{i}|\mathrm{C}_1) = \sum_{\mathrm{i}=k_n+1}^{\mathrm{L}} \frac{\mathrm{ip_i}}{\omega_1} = \frac{\mu_T - \mu\,(k_n)}{1 - \omega\,(k_n)}$$

$\Rightarrow \omega_0\omega_1 = \mu(k_n)$

$\Rightarrow \omega_1\mu_1 = \mu_T - \omega_0\mu_o$

$\Rightarrow \mu_T = \omega_1\mu_1 + \omega_0\mu_0$

Since $\omega_0 = \omega(k_n)$ $\qquad$ $\omega_1 = 1 - \omega(k_n)$

$\Rightarrow \omega_0 + \omega_1 = 1$

Variance of Multithreshold

$\sigma_B^2 = \omega_1(\mu_1 - \mu_T)^2 + \omega_0(\mu_0 - \mu_T)^2$

$= \omega_1(\mu_1^2 - 2\mu_1\mu_T + \mu_T^2) + \omega_0(\mu_0^2 - 2\mu_0\mu_T + \mu_T^2)$

$= \omega_1\mu_1^2 - 2\omega_1\mu_1\mu_T + \omega_1\mu_T^2 + \omega_0\mu_0^2 - 2\omega_0\mu_0\mu_T + \omega_o\mu_T^2$

$= \omega_1\mu_1^2 + \omega_0\mu_0^2 - (2\omega_1\mu_1\mu_T + 2\omega_0\mu_0\mu_T) + \omega_1\mu_T^2 + \omega_o\mu_T^2$

$= \omega_1\mu_1^2 + \omega_0\mu_0^2 - (2\omega_1\mu_1\mu_T + 2\omega_0\mu_0\mu_T) + (\omega_1 + \omega_0)\mu_T^2$

$\omega_1 + \omega_0 = 1$

$= \omega_1\mu_1^2 + \omega_0\mu_0^2 - 2\mu_T(\omega_1\mu_1 + \omega_0\mu_0) + \mu_T^2$

$\omega_1\mu_1 + \omega_0\mu_0 = \mu_T$

$= \omega_1\mu_1^2 + \omega_0\mu_0^2 - 2\mu_T\mu_T + \mu_T^2$

$= \omega_1\mu_1^2 + \omega_0\mu_0^2 - \mu_T^2$

$$\sigma_B^2 = \left(\sum_{i=k_n}^{L} \omega_i\,\mu_i^2\right) - \mu_T^2$$

$\mu_T^2 = (\omega_0\mu_0 + \omega_1\mu_1)^2$

$= (\omega_0\mu_0 + \omega_1\mu_1)\,(\omega_0\mu_0 + \omega_1\mu_1)$

$= \omega_0^2\mu_0^2 + \omega_1\omega_0\mu_1\mu_0 + \omega_1\omega_0\mu_1\mu_0 + \omega_1^2\mu_1^2$

$= \omega_0^2\mu_0^2 + 2\omega_1\omega_0\mu_1\mu_0 + \omega_1^2\mu_1^2$

$\sigma_B^2 = \omega_1\mu_1^2 + \omega_0\mu_0^2 - \mu_T^2$

$\sigma_B^2 = \omega_1\mu_1^2 + \omega_0\mu_0^2 - (\omega_0^2\mu_0^2 + 2\omega_1\omega_0\mu_1\mu_0 + \omega_1^2\mu_1^2)$

$\sigma_B^2 = \omega_0\mu_0^2 + \omega_1\mu_1^2 - \omega_0^2\mu_0^2 - 2\omega_1\omega_0\mu_1\mu_0 - \omega_1^2\mu_1^2$

$\sigma_B^2 = (\omega_0 - \omega_0^2)\mu_0^2 - 2\omega_1\omega_0\mu_1\mu_0 + (\omega_1 - \omega_1^2)\mu_1^2$

$\sigma_B^2 = \omega_0(1 - \omega_0)\mu_0^2 - 2\omega_1\omega_0\mu_1\mu_0 + \omega_1(1 - \omega_1)\mu_1^2$

$\omega_0 = 1 - \omega_1\,;\ \omega_1 = 1 - \omega_0$

$\sigma_B^2 = \omega_0\omega_1\mu_0^2 - 2\omega_1\omega_0\mu_1\mu_0 + \omega_1\omega_0\mu_1^2$

$$\sigma_B^2 = \omega_0\omega_1(\mu_0^2 - 2\mu_1\mu_0 + \mu_1^2)$$

$$\sigma_B^2 = \omega_0\omega_1(\mu_0 - \mu_1)^2$$

$$\sigma_B = \omega_0\omega_1(\mu_1 - \mu_0)^2$$

The optimal threshold value k* is obtained from maximum between class variance σ_B^2.

The range is from k to 255

$$\sigma_B^2(k_n^*) = \max_{k \le k_n \le L} \sigma_B^2(k_n)$$

Appendix D: Optical flow approach

Several approaches can use to determine the optical flow field. Lucas-Kanade method is a sparse/local method. Hon schunck Method is a dense/global method. Existing state of the art algorithm are used to calculate the optical flow field or velocity flow field. In our experimental study, application of Gunnar Farneback method are used.

(Lucas-Kanade Method)

The general equation of Lucas-Kanade algorithm.

$$I_x V_x + I_y V_y + I_t = 0$$

The objective is analysis the flow velocity vector (V_x, V_y). To re-arranged the general equation as follow.

$$I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1)$$

$$I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2)$$

$$\vdots$$

$$I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n)$$

Lucas-Kanade method can write as the matrix equation. $Av = b$

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ & \vdots & \\ I_x(q_n) & I_y(q_n) \end{bmatrix}$$

$$v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}$$

$$b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

Solve the matrix equation, $v = A^{-1}b$ can be obtained the new flow velocity vector. (V_x, V_y)

(Hon schunck Method)

The intensity of image pixel in the pattern is constant, so that

$$\frac{dI}{dt} = 0$$

Rate of change of Imgae brightness describe mathematically as follows.

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)$$

$$I(x, y, t) = I(x, y, t) + \delta x \frac{\partial I}{\partial x} + \delta y \frac{\partial I}{\partial y} + \delta t \frac{\partial I}{\partial t} + \epsilon$$

ϵ contains second and higher order terms in $\delta x, \delta y$ and δt.

$$\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t} + \mathcal{O}(\delta t) = 0$$

$$\delta t \to 0$$

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} + 0 = 0$$

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

$$u = \frac{dx}{dt};\ v = \frac{dy}{dt}$$

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$$

$$\frac{\partial I}{\partial x} = I_x;\ \frac{\partial I}{\partial y} = I_y;\ \frac{\partial I}{\partial t} = I_t$$

Linear equation have been obtained and have two unknown u and v.

$$I_x u + I_y v + I_t = 0$$

Where I_x, I_y and I_t are the derivatives of the image brightness. u and v are two components of the optical flow vectorat the pixel position.

$$I_x u + I_y v = -I_t$$

The equation can illustrate in another way.

$$(I_x I_y)(uv) = -I_t$$

$$-\frac{I_t}{\sqrt{{I_x}^2 + {I_y}^2}}$$

The Horn and Schunck global optimisation algorithm, two optical flow analysis requires to consideration.

Optical flow is smooth:

$$F_{smooth}(u,v) = \iint ({u_x}^2 + {u_y}^2)\,({v_x}^2 + {v_y}^2)dxdy$$

Optical flow constraint equation:

$$F_{optical}(u,v) = \iint (I_x u + I_y v + I_t)^2\, dxdy$$

Minimisation the sum of the errors in the equation for calculation the rate of change of brightness.

$$F_{HS}(u,v) = \iint \left[(I_x u + I_y v + I_t)^2 + \alpha^2({u_x}^2 + {u_y}^2 + {v_x}^2 + {v_y}^2)\right] dxdy$$

Using the calculus of variation, the equation can see that

$$I_x^2 u + I_x I_y v = \alpha^2 \nabla^2 u - I_x I_t$$

$$\nabla^2 u = \bar{u} - u$$

$$I_x^2 u + I_x I_y v = \alpha^2(\bar{u} - u) - I_x I_t$$

$$I_x^2 u + I_x I_y v = \alpha^2 \bar{u} - \alpha^2 u - I_x I_t$$

$$I_x^2 u + \alpha^2 u + I_x I_y v = \alpha^2 \bar{u} - I_x I_y$$

$$(I_x^2 + \alpha^2)u + I_x I_y v = \alpha^2 \bar{u} - I_x I_y$$

$I_x I_y u + I_y^2 v = \alpha^2 \nabla^2 v - I_y I_t$

$\nabla^2 v = \bar{v} - v$

$I_x I_y u + I_y^2 v = \alpha^2 (\bar{v} - v) - I_y I_t$

$I_x I_y u + I_y^2 v = \alpha^2 \bar{v} - \alpha^2 v - I_y I_t$

$I_x I_y u + I_y^2 v + \alpha^2 v = \alpha^2 \bar{v} - I_y I_t$

$I_x I_y u + \left(I_y^2 + \alpha^2\right) v = \alpha^2 \bar{v} - I_y I_t$

To slove the velocity field u

$\left(\alpha^2 + I_x^2 + I_y^2\right) u = \left(\alpha^2 + I_y^2\right) \bar{u} - I_x I_y \bar{v} - I_x I_t$

Rrewrite to alternate form

$\left(\alpha^2 + I_x^2 + I_y^2\right)(u - \bar{u}) = -I_x\left(I_x \bar{u} + I_y \bar{v} + I_t\right)$

From the iterative solution can estimate the new velocity.

$$u^{n+1} = \bar{u}^n - I_x \frac{I_x \bar{u}^n + I_y \bar{v}^n + I_t}{\alpha^2 + I_x^2 + I_y^2}$$

To slove the velocity field v

$\left(\alpha^2 + I_x^2 + I_y^2\right) v = (\alpha^2 + I_x^2) \bar{v} - I_x I_y \bar{u} - I_y I_t$

$\left(\alpha^2 + I_x^2 + I_y^2\right)(v - \bar{v}) = -I_y\left(I_x \bar{u} + I_y \bar{v} + I_t\right)$

$$v^{n+1} = \bar{v}^n - I_y \frac{I_x \bar{u}^n + I_y \bar{v}^n + I_t}{\alpha^2 + I_x^2 + I_y^2}$$

u^n and v^n is the previous estimate velocity

u^{n+1} and v^{n+1} is the new estimate velocity

Appendix E: Multiple logistic regression

Multiple logistic regression can be used to calculate the multivariable case which is more than one independent variable.

$$x' = (x_1, x_2, \dots, x_p)$$

The general equation of multiple logistic regression:

$$log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

To solve above equation for probability p apply the exponential function into both side.

$$exp\left(log\left(\frac{p}{1-p}\right)\right) = exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)$$

In right hand side:

If $exp(x) = e^x$, then

$$exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p) = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p}$$

In right hand side:

$$exp\left(log\left(\frac{p}{1-p}\right)\right) = \frac{p}{1-p}$$

$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p}$$

$$p = (1-p)e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p}$$

$$p = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p} - pe^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p}$$

$$p + pe^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p} = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p}$$

$$p\left(1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p}\right) = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p}$$

$$p = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p}}$$

$$g(\mathrm{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

$$p = \frac{e^{g(\mathrm{x})}}{1 + e^{g(\mathrm{x})}}$$

The probability p can be shown by $P(Y = 1|\mathrm{x}) = \pi(\mathrm{x})$.

$$\pi(\mathrm{x}) = \frac{e^{g(\mathrm{x})}}{1 + e^{g(\mathrm{x})}}$$