

# Early Fire Detection using Enhanced Optical Flow Analysis Technique



This thesis was submitted in partial fulfilment of the requirement for the degree of

Bachelor of Computer Science and Engineering

Under the Supervision of

Dr. Jia Uddin

By

Arnisha Khondaker (14301068)

Arman Khandaker (18141022)

**School of Engineering and Computer Science**

April 2018

BRAC University, Dhaka, Bangladesh

## Declaration

We hereby affirm that the work done for this thesis is based on results attained from our own work. All of the tools and materials used in this text have been properly acknowledged. This thesis, neither in whole nor in part, has been previously submitted to any other University or Institute for the award of any degree or diploma.

### Signature of Supervisor

### Signature of the Authors

---

Dr. Jia Uddin  
Assistant Professor, Department of  
Computer Science and Engineering, BRAC  
University, Dhaka, Bangladesh

---

Arnisha Khondaker (14301068)

---

Arman Khandaker (18141022)

## Acknowledgement

We would like to begin by expressing our deep gratitude to the Almighty Allah, the creator of everything in existence within the universe, the most benevolent and the most gracious, for giving us the strength and determination to initiate and complete our research successfully.

Next, we would like to express our very great appreciation to Dr. Jia Uddin, our thesis supervisor, for his valuable and constructive suggestions during the planning and development of this research work. His willingness to give his time so generously has been very much appreciated. We would also like to thank the BRAC University faculty members and staffs of the Computer Science and Engineering department, who have been constantly guiding us throughout our study period at BRAC University, especially in building and enhancing our base in education and knowledge.

We would also like to express our gratitude and gratefulness to our beloved family and friends. We are indebted to all of the participants who helped us directly, or indirectly in completing our research work.

Finally, and this goes without saying, we want to say a special thank you to BRAC University for enabling us to conduct this research and providing us with the right support and tools to do so.

# Table of Contents

<b>List of Figures</b> .....	<b>vi</b>
<b>List of Tables</b> .....	<b>vii</b>
<b>List of Abbreviations</b> .....	<b>viii</b>
<b>Abstract</b> .....	<b>1</b>
<b>CHAPTER 01</b> .....	<b>2</b>
<b>Introduction</b> .....	<b>2</b>
1.1 Motivation .....	2
1.2 Contribution Summary .....	3
1.3 Thesis Outline.....	4
<b>CHAPTER 02</b> .....	<b>5</b>
<b>Literature Review</b> .....	<b>5</b>
2.1 Static Image Processing.....	5
2.1.1 Color Spaces .....	5
2.1.2 Color space conversion.....	8
2.1.3 Color Segmentation .....	9
2.1.4 K-means Clustering .....	10
2.1.5 Contour extraction .....	10
2.2 Dynamic Video Processing .....	11
2.2.1 Foreground detection or Background subtraction .....	11
2.2.2 Tertiary Feature Extraction.....	13
2.2.3 Optical Flow Tracking.....	14
<b>CHAPTER 03</b> .....	<b>23</b>
<b>Proposed Model</b> .....	<b>23</b>
3.1 Block diagram of the proposed model .....	23
3.2 Chromatic segmentation in YUV Color Space .....	24

3.2 Growth Analysis.....	30
3.3 Optical Flow Analysis.....	33
3.3.1 Corner-detection.....	34
3.3.2 Motion velocity vector calculation.....	36
<b>CHAPTER 04.....</b>	<b>40</b>
<b>Experimental Analysis.....</b>	<b>40</b>
4.1 Experimental Setup.....	40
4.1.1 Sample scenes from the dataset.....	40
4.1.2 Methodology of calculating accuracy.....	42
4.2 Experimental Results.....	43
4.3 Analysis.....	48
<b>CHAPTER 05.....</b>	<b>49</b>
<b>CONCLUSION.....</b>	<b>49</b>
5.1 Concluding Remarks.....	49
5.2 Limitations and Future Works.....	49
<b>REFERENCES.....</b>	<b>50</b>

## List of Figures

Figure 1. A visual representation of the YUV color space. ....	6
Figure 2. Frame Differencing vs Median Filtering .....	12
Figure 3. Various geometric transformations.....	14
Figure 4. Optical flow tracking of cars moving across the screen. ....	16
Figure 5. Conversion from RGB to Grayscale .....	18
Figure 6. A fast implementation of generalized Lucas-Kanade optical flow tracking algorithm.....	20
Figure 7. Image pyramids.....	21
Figure 8. Block diagram of the proposed model.....	23
Figure 9. Conversion from RGB to YUV. ....	24
Figure 10. Output of split color rules. ....	26
Figure 11. Output of majority voting rule. ....	28
Figure 12. An overview of the chromatic segmentation part of the proposed model.....	29
Figure 13. Flow diagram illustrating the chromatic segmentation phase of the proposed model.....	31
Figure 14. Flow diagram illustrating the growth analysis phase of the proposed model.....	32
Figure 15. The highly chaotic behavior of flames with no definitive structure or shape.....	33
Figure 16. Shi-Tomasi vs FAST algorithm.....	35
Figure 17. Motion vectors tracked .....	36
Figure 18. Turbulent motion vectors for fire scene.....	38
Figure 19. Calm motion vectors for non-fire scene. ....	39
Figure 20. Comparative bar histogram of F1score for fire samples.....	46
Figure 21. Comparative bar histogram of F1score for non-fire samples .....	47
Figure 22. The ROC curves for the neural network trained for detecting turbulence.....	48

## List of Tables

Table 1: Performance comparison among various color spaces .....	8
Table 2: Comparison of Shi-Tomasi and FAST algorithm. ....	35
Table 3. Sample scene and description of videos in the dataset. ....	40
Table 4: The accuracy of videos in the dataset, containing both fire and non-fire videos.....	43
Table 5: The accuracy of the proposed model compared to compared to competing models. ....	44
Table 6: Precision and recall of the proposed model. ....	44

## List of Abbreviations

AVI	Audio Video Interleave
CCTV	Closed-Circuit Television
FAST	Features From Accelerated Segment Test
FN	False Negative
FP	False Positive
LKT	Lucas Kanade Tracker
RGB	Red Green Blue
ROC	Receiver Operator Characteristic
ST	Shi-Tomasi
TN	True Negative
TP	True Positive



## Abstract

This paper proposes a multi-stage fire detection model that consists of chromatic segmentation, shape analysis and differential optical flow estimation. At the initial phase, color segmentation is carried out which takes into account some of the existing state of the art color segmentation directives and employs a majority voting system among them to obtain the possible fire-like regions. The extracted sections are then passed onto a shape analyzer which verifies the authenticity of the candidate regions by inspecting the dynamics of shape. The distinctive change fire exhibits over time in its area-perimeter ratio is at the bedrock of this analyzer. Further evaluation is carried out by another analyzer that measures the turbulence of fire evaluated by an enhanced differential optical flow tracking algorithm. The Lucas-Kanade Tracking algorithm has been employed and extended to achieve this. The assessment of performance of the enhanced techniques was carried out by utilizing a versatile dataset containing videos from the MIVIA and Zenodo dataset. The dataset consists of a diverse array of different environments such as indoor, outdoor and forest fire. Some environments with no fire were also included to assess the rate of false positives. The model has successfully showed an improved accuracy of 95.62% when tested for the aforementioned dataset.

**Keywords:** Fire Detection, Color Segmentation, YUV color space, Shape Analysis, Optical Flow Analysis, Fast Features From Accelerated Segment Test, Lucas Kanade Tracker

# CHAPTER 01

## Introduction

Fire is regarded as one of the earliest human inventions and this innovation led to the inception of human civilization. Despite its many merits, fire can prove to be dangerous if it gets out of hand and claim thousands of lives and cause insurmountable damage to property. Early fire detection has therefore, become increasingly important as fire hazards can directly endanger personal security and belongings. Thus, this paper proposes a model that can help mitigate the casualties due to a fire incident by efficiently identifying fire at an earlier stage of its occurrence. Using a score-based color rule approach and an enhanced optical flow analysis technique, the proposed system allows for an efficient early detection of fire that outperforms similar models.

### 1.1 Motivation

A fact sheet published by World Health Organization (WHO) states that an estimated 180,000 deaths are caused per year due to burns and a significant portion of these injuries victims receive primarily due to fire incidents. Majority of these incidents occur in low to middle income countries such as Bangladesh. Recently, there has been an upsurge of fires erupting not only in Dhaka but all over the country which according to experts is a result of unplanned urbanization and a lack of adherence to following the National Building Code when constructing new buildings. In 2017 alone Dhaka itself saw 3,020 incidents. Ever year factory fires not only take countless lives, they also stagnate the economy of our country by slowing down the RMG sector. The Gulshan Market fire that broke out in January, 2017 engulfed the happiness of many. Hence, it is undoubtedly a matter of great importance to detect fire within a short notice in order to minimize the fatality.

Although sensor-based fire detection systems that rely on heat or smoke to detect fire are already found everywhere, these systems have multiple limitations. Firstly, these sensors are limited in terms of area coverage and thus needs to be densely distributed, which raises the installation and maintenance costs [1]. Secondly, heat and smoke do not disperse right away which makes these sensors naturally slow in detecting these signals. To mitigate these limitations, fire detection techniques based on computer vision have been gaining momentum recently [1, 2]. They also give an additional edge over conventional detection systems because they can be easily integrated as a

component of existing video surveillance systems which most industries or companies already use to monitor buildings and the surrounding environment.

State of the art computer vision algorithms utilizing multiple phases of pattern recognition can make early fire detection feasible. For high accuracy, these phases analyze various distinguishable features of fire, including its distinctive color, shape, flickering, growth etc. This paper employs YUV color segmentation, analysis of shape and an enhanced optical flow analysis technique to detect fire in a robust manner.

## **1.2 Contribution Summary**

Previously, a number of researchers have worked on different fire detection models each with its own merits and demerits. Many of these models rely heavily on the color segmentation done at the beginning of the process. A limitation of many chromatic filtering algorithms is that they perform a logical conjunction of all these rules to segment out the region of interest [3]. For distant flames we have found that this approach often fails to detect the flame properly. As later steps usually require the output from this step, a failure to extract a blob in this step means the later steps do not even get the chance to present their own evaluation, resulting in poorer overall accuracy. We have taken a more liberal approach to overcome this limitation by taking the vote of all the rules and segmenting the image based on majority decision. This increases the accuracy of detecting candidate fire like regions for further analysis wherever needed.

Another thing to take into account is that such detectors alone are not sufficient enough for detecting fire in diverse environments and also for not triggering an alarm when there is no fire. Many algorithms therefore incorporate other characteristics of fire such as shape, growth, flickering etc. in order for more accurate identification. The model proposed in this paper incorporates shape analysis and proposes an enhanced optical flow analysis of fire to eliminate the chances of producing false alarms as well as increasing the accuracy of detection. This approach has produced an accuracy of 95.62%.

### **1.3 Thesis Outline**

The rest of the paper is outlined as follows. Chapter 2 outlines the previous works done in the field of computer vision based fire detection. Next, chapter 3 describes the proposed model including the implementation specifics. Chapter 4 goes on to illustrate the experimental results based on a versatile dataset and the performance comparison of competing models. Finally, chapter 5 concludes the paper specifying its difficulties and limitations while stating possible future paradigms the system can be expanded to.

## CHAPTER 02

### Literature Review

#### 2.1 Static Image Processing

One paradigm of computer vision based fire detection focuses mainly on analyzing static digital images. Images are represented as a set of pixels which are the smallest addressable and controllable element of a picture rendered on a screen. Many researches on classification of fire have been carried out by analyzing singular frames containing fire in different scenarios [4]. Bulk of the dataset they have used contains static pictures and the inspection for fire have been done by mostly taking into account the color properties and shape of fire as discussed in this section.

##### 2.1.1 Color Spaces

Color spaces or color models are mathematical formulations that help represent the chromatic information a color image exhibits. Color information of fire is crucial for its detection since non-chemical fire exhibits a distinctive color. As a result, some form of color evaluation can be found across all models of fire detection. Chromatic information can be found out using color segmentation algorithms that rely on multiple color models such as RGB [2], YCbCr [4],  $L^*a^*b^*$  [1], YUV [5], HSI [6, 7], HSV [8] or even a combination of different color spaces [9]. There are however mainly five major color models: CIE, RGB, YUV, HSI and CMYK. Most of the other color models are simply a subdivision of these models. For the purpose of fire detection, images are typically analyzed under RGB, CIE, LAB (or  $L^*a^*b^*$ ), HSI or YUV color spaces. A quantitative analysis of these color spaces show us that some of them perform better than the others on the list.

##### 2.1.1.1 RGB

The RGB color space consists of the chromaticity of three additive primaries: Red, Green and Blue. Each of these three primaries define the intensity of color as an integer between 0 and 255 or 0 and 1. For example, if the RGB value for a pixel is (1, 0, 0) then the pixel is completely red in color. Although a very simplistic color model, it is so widely used because most surveillance cameras use this color space by default and while analyzing the images obtained from these cameras, no further conversion to other color spaces is required, leading to improved performance.

### 2.1.1.2 La\*b\*

The Lab color space has three dimensions for mathematically representing all perceivable colors. L represents lightness, whereas a\* and b\* represent green-red and blue-yellow components of color respectively. The color channel L exhibits darkest black when L=0, and brightest white when L=100. When the channels a\* and b\* are null, they represent true neutral gray. The positive a\* axis corresponds to red color components while the negative a\* axis corresponds to green. Similarly, the positive and negative b\* axis corresponds to yellow and blue color components respectively. The La\*b\* color space is an excellent choice when accuracy matters the most as shown in Table 1.

### 2.1.1.3 YUV

The breadth of the terms Y'UV, YUV, YCbCr, YPbPr, etc. can be overlapping and is a frequent source of confusion. Y' denotes the luma component (the brightness) whereas U and V are chrominance (color) components; luminance on the other hand is labeled by Y – the prime symbols (') signify gamma compression [4] with 'luminance' standing for physical linear-space brightness, while 'luma' stands for (non-linear) perceptual brightness. Y can range from 0 to 1 (or 0 to 255 in digital formats), while U and V falls in the range of -0.5 to 0.5 (or -128 to 127 in signed digital form). Standards exist to limit these ranges further so that the resulting unused bits can be utilized to indicate metadata like synchronization.

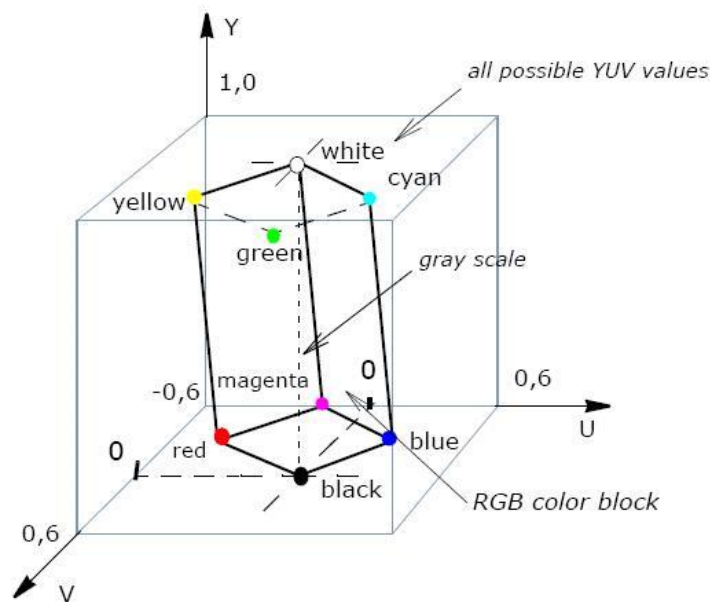


Figure 1. A visual representation of the YUV color space [10].

#### 2.1.1.4 Selecting an appropriate color model

One common color model to undertake in fire detection systems is RGB [2]. The reasoning behind selecting this model is the simplicity it offers and that it requires no further processing or conversion as virtually all cameras record videos directly in RGB color space. Contrary to [2], [6] and [7] for good reasons adopts HSI color space for extracting the color information from frames. RGB color model has the disadvantage of being dependent on illumination whereas HSI is less sensitive to brightness change causing a lesser number of false positives (FPs). Added to that, YUV and YCbCr color spaces also give an edge over RGB as they take into account the chrominance and luminance component which can be processed independently [11]. Celen *et al.* carried out an experiment to find out which color spaces were the most suitable for forest fire detection [12]. Possible fire like regions were identified using the values of three different channels obtained from different color models. The likelihood of each channel was obtained from a lookup table and thresholding the obtained value learned through a learning algorithm. The probability of a pixel being a fire pixel was calculated by the product of the likelihoods each channel as shown in equation (1).

$$P(C_1; C_2; C_3) = P(C_1) * P(C_2) * P(C_3) \quad (1)$$

where  $C_1$ ,  $C_2$  and  $C_3$  represent the values of channel 1, channel 2 and channel 3 in any color space of a pixel.

The comparative analysis laid out by Celen gave an idea of which color spaces work the best in the domain of fire detection which is depicted in Table 1. It is apparent from the comparison that CIE Lab and YUV surpass others in terms of performance.

Table 1: Performance comparison among various color spaces

Color space	Accuracy (%)	Recall (%)	Precision (%)	Specificity (%)
CIE Lab	92	95	94	75
CIE Luv	82	99	82	3
CIE XYZ	84	99	84	13
HLS	87	87	97	89
HSV	87	99	87	33
RGB	86	98	87	31
YUV	90	96	92	62
Other Methods	73	63	79	83

### 2.1.2 Color space conversion

Typically CCTV cameras provide output in RGB color space. Therefore, conversion of color spaces is required if processing is to be done in some other color space. As demonstrated in Table 1, CIE Lab and YUV are the most accurate color models for fire detection. For our study, we've however decided to use YUV as the conversion is computationally lesser expensive and it's almost as good as La\*b\*. Compared to YUV, HSI is thus less suitable for real-time detection. As optical flow analysis is an expensive process by itself, to increase overall performance it is important to minimize the time spent in other expensive phases like color space conversion. The conversion from RGB to YUV is performed using equation (2) as demonstrated in [11].

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2)$$

Where Y, U, and V corresponds to the Y channel, U channel and V channel of the image and domains for Y, U and V are [0,255], [-111,111] and [-157,157] respectively.



### 2.1.3 Color Segmentation

Over the last few decades, a number of rules has been developed to classify fire pixels using raw red, green and blue (RGB) information in color images [2]. Chen *et al.* adopted a statistical RGB color model and used the intensity and saturation of the red component in addition to a binary background mask with median filtering [13]. The method utilizes statistical color values and is fairly straightforward in its depth and implementation. The rules put forth by Chen *et al.* are depicted in equation (3) and (4).

$$I_R(x) - I_G(x) > 0 \quad (3)$$

$$I_G(x) - I_B(x) > 0 \quad (4)$$

where  $I_R(x)$ ,  $I_G(x)$  and  $I_B$  stand for the intensity values for red channel, green channel and blue channel of a pixel  $x$  respectively. For fire pixels, the value of red channel is greater than the green channel and that of green is greater than blue [2].

$$I_R(x) - \tau_R > 0 \quad (5)$$

$$I_S(x) - (255 - I_R(x)) \frac{\tau_S}{\tau_R} > 0 \quad (6)$$

where  $\tau_R$  and  $\tau_S$  stands for the threshold values which were not specified in [2]. However, experimentally it has been found that the best results were obtained when  $\tau_R = 135$  and  $\tau_S = 53$ .

Different versions of the formulae originally formulated by Chen are found in existing literature. Celik and Huseyn *et al.* [1] for example derived and enhanced new rules based on YCbCr which were later modified and adopted in [4]. One such rule derived in the RGB color space detailed the fact that the R component for a fire pixel must be greater than the mean of values in the R channel. Liang-Hua *et al.* implemented their work on HSV [8] and Jareerat *et al.* employed their technique in both HSV and YcbCr color spaces [9]. A statistical model based on YUV color space is also exploited in [14], where instead of a trivial heuristic, the thresholding of candidate fire pixels is reliant on a support vector machine.

Rossi and Akhloufi interestingly used both RGB and YUV color spaces [15]. Applying K-means clustering to the V channel of the YUV color space showed promising prospect while detecting fire. For the sake of their research the value of  $k$  was set to 2; one for fire and one for background.

#### 2.1.4 K-means Clustering

K-means clustering through repeated calculation allows to find centers of natural clusters within given data. It is used to find K number of partitions based on a given set of attributes. Vector space is utilized to represent the object attributes. The main objective is to mitigate the average intra-cluster variance as outlined in [15] and shown by equation (7).

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - m_i\|^2 \quad (7)$$

where  $S_i$  are the clusters in  $i \in [1, \dots, k]$ , the centroid of the points  $x_j \in S_i$  is given by  $m_i$  and  $k$  represents the number of clusters.

The first level of segmentation using K-means clustering showed promising prospect but the number of false positives it raised was too high and so a learning strategy in RGB color model using 3D Gaussian model to represent fire pixels was used to create a reference color model for fire pixel classification. In dynamic video processing, such complex analysis on each pixel is computationally expensive and therefore, in our study we have adhered mostly the simplistic rules set out by Celik *et al.* [4] as they are just as accurate but computationally cheaper.

#### 2.1.5 Contour extraction

In order to make the fire detection system more robust, geometric characterization of fire have also been taken into account in a number of different models [16]. One such method to adopt is contour extraction. Contour extraction is popularly used because only focusing on the chromatic properties of fire may give rise to a high rate of false positives. Most contour extraction algorithm uses some form of edge detector operator [16]. Canny edge detector has been utilized in [17] in order to detect large space fires. A diversion of this was seen in [18] where, Bheemul *et al.* adopted a model for contour extraction that does a horizontal line by line analysis of the change in brightness in a frame. The main disadvantage of applying these contour extraction techniques in single frames is that they fail to consider that fire is a phenomenon that changes over time. Thus, if analysis is done on a set of frames obtained from dynamic videos rather than one frame, the output obtained is more accurate.

## 2.2 Dynamic Video Processing

A video is a sequence of ordered frames. Analyzing a set of video frames is more useful when it comes to detecting fire because it is a spatiotemporal occurrence. While processing videos some form of preliminary segmentation is done either by the chromatic segmentation explained beforehand or by background substitution to extract the region of interest.

### 2.2.1 Foreground detection or Background subtraction

Fire is a phenomenon that undergoes rapid transformation over time. Considering this fact, many models attempt to detect the moving pixels between consecutive frames to figure out a region of interest. Further analysis is then performed on the blobs obtained based on some other feature of fire [5, 11]. The notable methods followed for background subtraction are frame differencing [19], median filtering [6], Gaussian average and background mixture model. Frame differencing is done by first taking a background reference frame and then taking any frame at time  $t$  to perform an image subtraction between the two frames. The background reference frame thus has to be updated iteratively as new frames arrive. The method proposed by Cho *et al.* used a binary background mask in conjunction with median filtering to minimize noise and accurately detect moving fire-like regions [6]. Shon *et al.* and Celik *et al.* use the intensity values of pixels and compares the changes to a threshold over time to detect fire like regions [1, 11]. In [11] the threshold was experimentally obtained to be 3. In the work of [20], Gaussian mixture models are used to separate the moving foreground from the stationary background. Although Gaussian mixture model is highly accurate, it is heavy on its use of computational resources. In the context of real-time fire detection, this makes it less valuable and inapplicable for most cases.

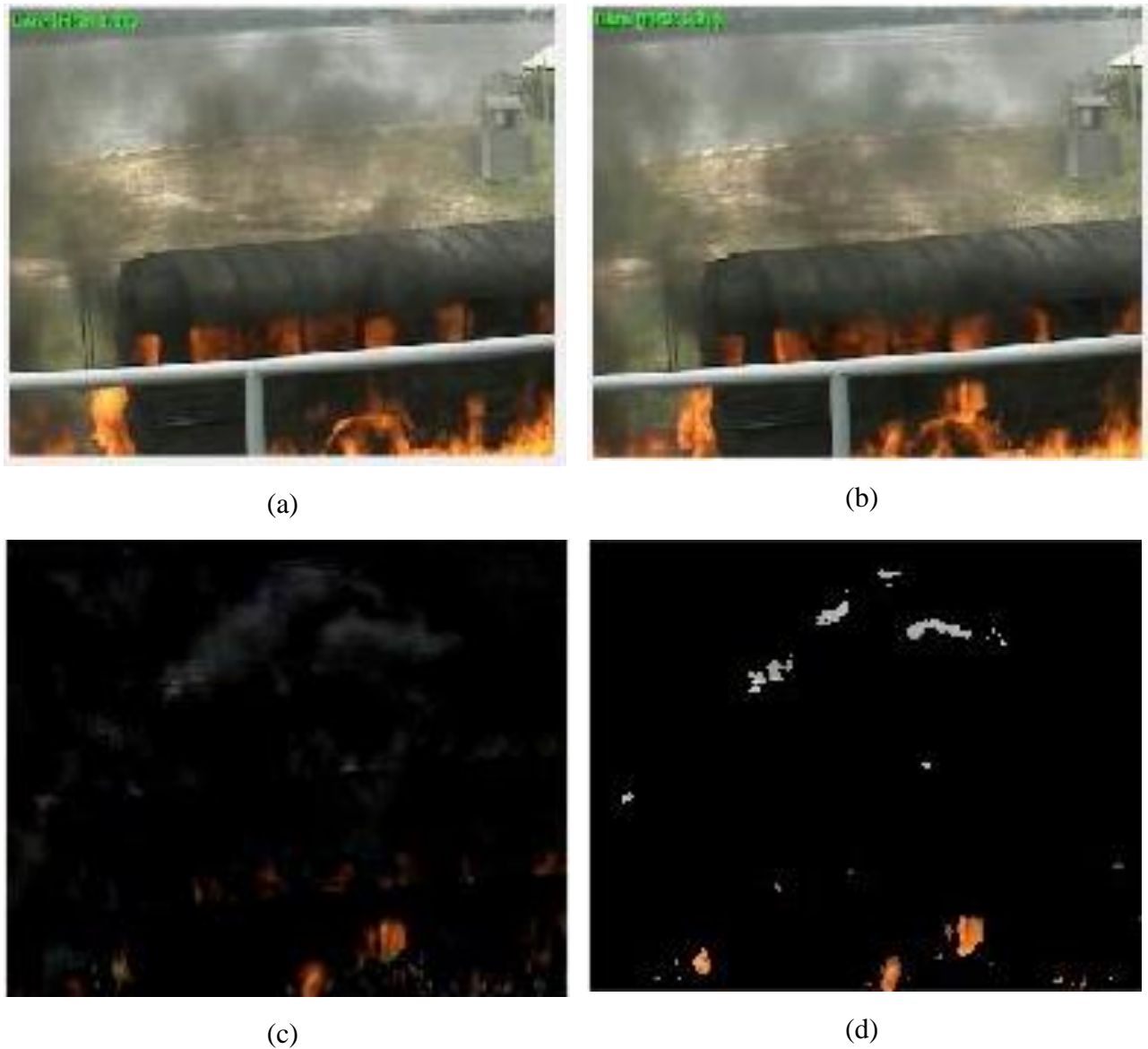


Figure 2. (a) and (b) shows two consecutive frames. (c) is the result of frame differencing while (d) is the result of median filtering.

It is however expected that foreground detection algorithms will return a lot of spurious non-fire regions. Any red fire like objects such as a moving red vehicles, a person wearing a red t-shirt etc. are also likely to be detected. That is why, further feature extraction must be done to verify the detection.

### 2.2.2 Tertiary Feature Extraction

Some features of fire such as contour, flickering, area of growth etc. and the spatiotemporal differences among them can be used to make the fire detection system more robust. Different models proposed so far uses different features of fire to enhance the accuracy of the models.

Wang *et al* carried out an experiment on different edge detection algorithms to find out which ones work the best. It was seen that Canny edge detector has many pseudo-edges and weren't very effective in terms of being continuous [17]. There also exists the popular Sobel operator that has excellent accuracy but is expensive in performance. Edge detection algorithms are good for detecting the contour of fire. Since fire has a disorderly shape, the contour is an interesting feature to analyze. This forms the basis of many shape analysis algorithms as well [16].

Flickering detection is also something that frequently appears in academia. This includes features that arise from eddies effect, and also the inherent fluctuation of colors on the surface of fire. Suchet *et al.* in [21] has employed an algorithm that analyzes the vortex shedding frequency to determine fire. Khan *et al.* in [19] has also used a novel algorithm that detect the fluctuation of the blue component in fire pixels. While flickering is a good indicator of fire, it can also result in a lot of false positives in other luminous objects like the sun.

It is common knowledge that fire grows over time. The analysis of growth is also thus an important feature that is often extracted for detection. This is often used in tandem with shape analysis. In [5] for example, Foggia *et al.* calculates the perimeter-area ratio of potential fire blobs in order and measures the variation over time to detect the change in shape and area of fire. In [19] Khan *et al.* has also taken a simple frame differencing approach to detect regions that have static area over time and eliminating them to reduce false positives. This step on its own is not usually very reliable so other steps are always incorporated to make this more robust.

Another important feature that has also been seen in research is the turbulent nature of fire. Optical flow analysis techniques are usually utilized to exploit this feature for successful detection of fire. In [22] for example the dynamic texture of fire is analyzed by forming a feature vector based on a customized point-wise optical mass transport algorithm. Although exceptionally accurate, the algorithm is too computationally heavy to be useful for real-time detection. There are more traditional Horn-Schunck [23] and Lucas-Kanade [21] tracking algorithms that are also very reliable, with Lucas-Kanade lending easier on computational resources. In the proposed model, an enhanced method making use of Lucas-Kanade Tracker is introduced. The next section will give an outline of how general optical flow analysis is applied to fire detection and overview the LKT algorithm.

### 2.2.3 Optical Flow Tracking

Optical flow is the translation, reflection, rotation, scaling, shearing or other types of transformation of objects, edges or other features in a video from one frame to another which can also be relative to the movement of the observer. Translation, reflection and rotation are examples of Euclidean transformation whereas scaling and shearing are examples of affine transformations. There are also projective transformations but in the context of optical flow analysis for fire detection, we are mostly concerned with affine transformations. Figure 3 shows the transformation of simple features or objects (like the letter ‘L’) that can be tracked by optical flow estimation algorithms.

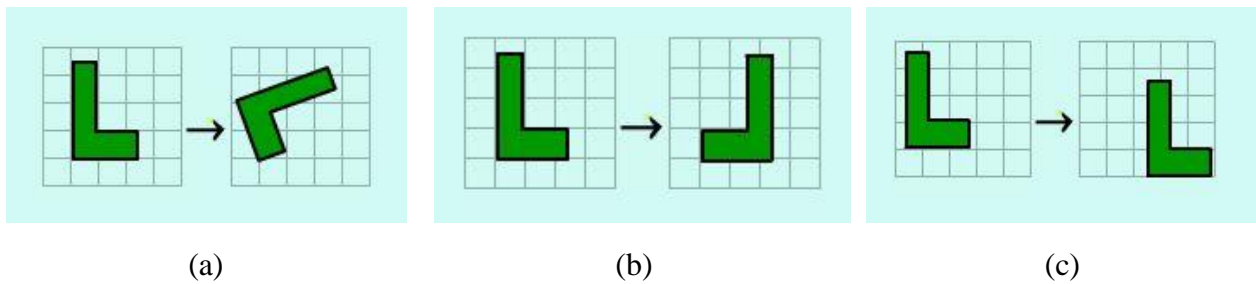


Figure 3. (a) Shows the rotational transformation of the letter ‘L’. (b) Shows the reflective transformation of the letter ‘L’ in y axis (c) Shows the translational transformation of the letter ‘L’.

$$\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ t \\ 1 \end{bmatrix} = \begin{bmatrix} x + dx \\ y + dy \\ t + dt \\ 1 \end{bmatrix} \quad (8)$$

Here,  $\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix}$  represents the translation matrix which when applied over  $x$ ,  $y$  and  $t$  yields the

translated coordinates  $x + dx$ ,  $y + dy$  and  $y + dt$ . This is a simple example of an affine transformation where length and angles are preserved.

Typically, optical flow estimation is used to evaluate the transformations that an object in a scene has undergone. This can be used to calculate the motion or instantaneous velocity vectors of

the features detected in a scene. It is frequently used in robotics vision where a robot uses optical flow estimation techniques to detect objects in its visual periphery which are then subsequently tracked according to its needs [24]. Movement detection, visual odometry, structure estimation and robot navigation in general requires the motion estimation capabilities of optical flow algorithms. The important field of video compression has also directly emerged from the techniques developed from research of optical flow analysis [25]. There are multiple techniques to determine the optical flow, including naïve approaches like phase correlation and block-based methods. More advanced methods use differential analysis that calculates partial derivatives of the image signal. Methods in this family include Lucas-Kanade, Horn-Schunck, Buxton-Buxton, Black-Jepson etc. An indicator of a good flow estimation is its ability to both quantitatively and qualitatively evaluate the motion of objects in a real-time stream. The differential methods fall in this category.

All these methods have some constraints in common that must be satisfied for successful evaluation. Optical flow estimation techniques try to calculate the motion between two consecutive frames at time  $t$  and  $t + 1$ . Differential techniques like Lucas-Kanade and Horn-Schunck require calculating partial derivatives of the two consecutive frames with respect to the spatiotemporal voxel position at  $(x, y, t)$  where  $x$  and  $y$  are spatial coordinates and  $t$  is the temporal coordinate in the image sequence. This consequently requires three fundamental assumptions or constraints that must hold at all times for a good estimation of optical flow. Firstly, the intensity must be equal in every voxel  $(x, y, t)$ , secondly, the motion must be small enough from  $t$  to  $t + 1$ , and lastly, the neighboring pixels of pixel  $(x, y)$  at given time  $t$  must have similar motion. The constraints are discussed in detail in the next sections.

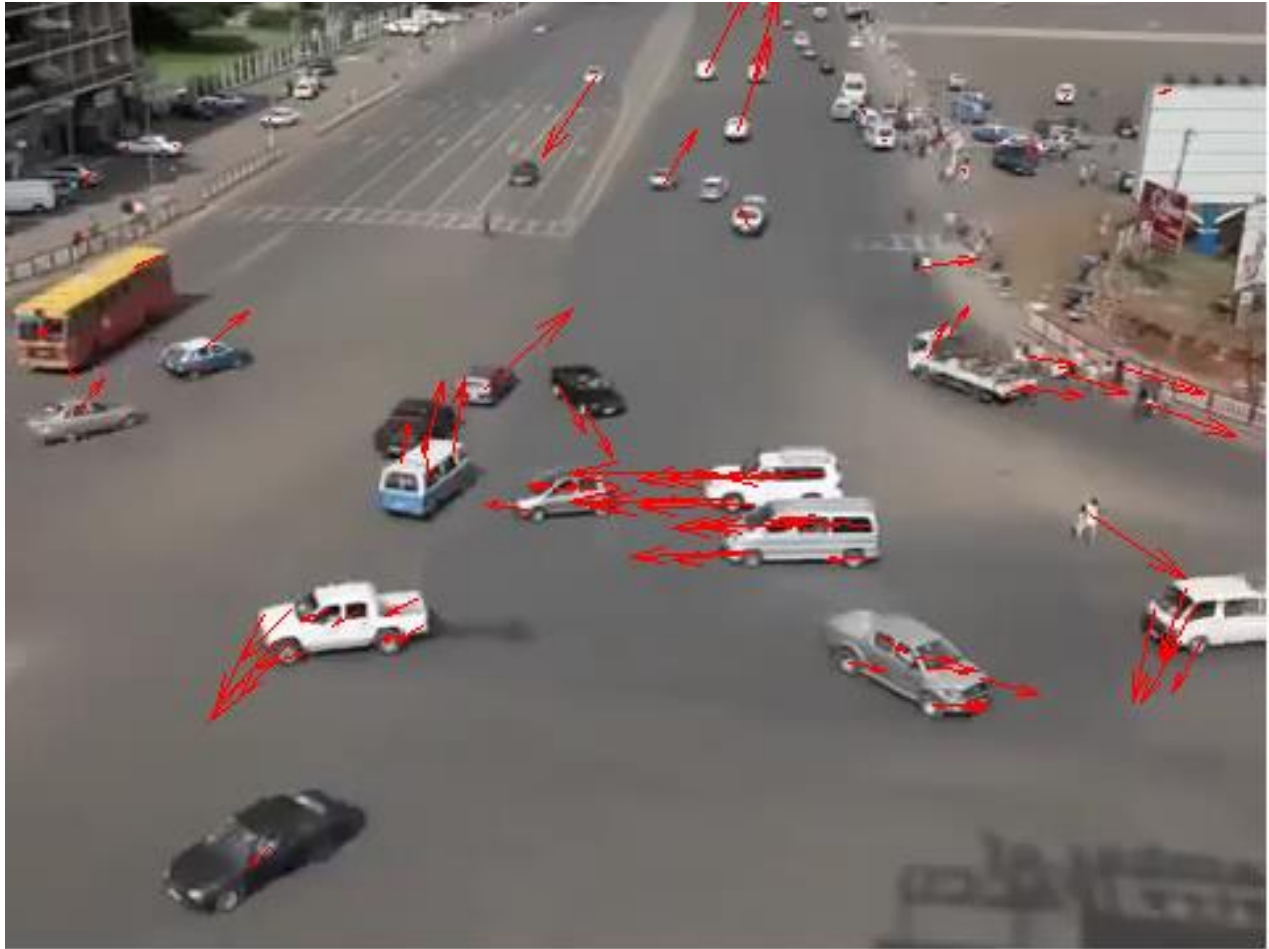


Figure 4. Optical flow tracking of cars moving across the screen [27]. It is to be noted that the velocity vectors of each object are all uniformly pointing in the same direction as their movement.

### 2.2.3.1 Constancy of brightness

For a voxel  $V$  at  $(x, y, t)$ , let us consider  $I(V)$  to be the intensity information. Given that the voxel  $V$  has moved from position  $x$  to  $x + \nabla x$ ,  $y$  to  $y + \nabla y$ , and  $t$  to  $t + \nabla t$ , the constraint can then be effectively expressed as:



$$I(V) = I(x, y, t) = I(x + \nabla x, y + \nabla y, t + \nabla t) \quad (9)$$

where  $x$ ,  $y$  and  $t$  are spatiotemporal coordinates and  $I$  is the magnitude of intensity.

The equation in (9) signifies that that the brightness of a voxel must be constant. Also, by solving the Taylor series approximation for (9), we can derive the Optical Flow equation [26]:

$$I_x u + I_y v + I_t = 0 \quad (10)$$

Here,

$$I_x = \frac{\partial I}{\partial x} \quad (11)$$

$$I_y = \frac{\partial I}{\partial y} \quad (12)$$

$$u = \frac{dx}{dt} \quad (13)$$

$$v = \frac{dy}{dt} \quad (14)$$

$I_x$  and  $I_y$  are the partial derivatives of the image brightness along the spatial dimensions  $x$  and  $y$  respectively whereas  $I_t$  is the gradient along the temporal dimension.  $u$  and  $v$  on the other hand are the horizontal and vertical optical flows respectively.

However it is not possible to solve the optical flow equation (10) because it has two unknown variables  $u$  and  $v$ . Differential optical flow techniques propose various methods to deal with this, Lucas-Kanade being one of them. The solution provided by Lucas-Kanade will be described in later in the paper.

This constraint can largely be ignored by simply converting the frames in the video stream to grayscale using any well-known conversion algorithm [28] which normalizes the brightness information. There are more accurate non-linear gamma correcting algorithms but we want this phase to be as fast as possible without forgoing the ability to satisfy the constraint. The lightness method of

conversion provides a good tradeoff between accuracy and efficiency by allowing a fast conversion with a decrement of contrast that is suitable for our purposes. The formula for the lightness method is represented by equation (15).

$$Gray(x, y) = \frac{\max(R(x, y), G(x, y), B(x, y)) + \min(R(x, y), G(x, y), B(x, y))}{2} \quad (15)$$

where  $R(x, y)$ ,  $G(x, y)$ ,  $B(x, y)$  are the color channels for the pixel  $(x, y)$  respectively and  $Gray(x, y)$  is the corresponding grayscale value.



Figure 5. Showing the original image in RGB color space and (b) showing the image in grayscale converted using the lightness method in (15)

### 2.2.3.2 Similarity of nearby motion

The constraint requires that the pixels surrounding a given pixel  $(x, y)$  must be coherent in terms of both direction and displacement over time. This implies that neighboring pixels of the tracked feature must have similar motion to the pixels of the tracked feature. This is because algorithms like the Lucas-Kanade method divides the original frame into small subsections and treats each of these subsections to have a constant velocity. For a  $3 \times 3$  subsection, we have nine points in total which are assumed to have similar motion. This results in trying to solve nine equations with two unknowns but

a better approach is to use the weighted least-square fit of the optical flow equation (21). Lucas-Kanade attempts to perform this by minimizing the equation (16).

$$\sum_{x \in k} [I_x u + I_y v + I_t]^2 W^2 \quad (16)$$

where  $k$  is a 3x3 subsection of a frame and  $W$  is a window function that accentuates the constraints at the centroid of each subsection  $k$ .

The final solution of the minimization problem in (16) which can be solved to get the optical flow estimation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x^2 W^2 & \sum I_y I_x W^2 \\ \sum I_x I_y W^2 & \sum I_y^2 W^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_t I_x W^2 \\ -\sum I_t I_y W^2 \end{bmatrix} \quad (17)$$

where  $u$  and  $v$  are the optical flow vectors.

A fast implementation of the generalized differential Lucas-Kanade algorithm at first calculates the gradient components  $I_x$  and  $I_y$  using a kernel and uses another difference filter to calculate  $I_t$  between two consecutive frames. The components are then smoothed with a isotropic 5x5 kernel and the linear equations in (28) are subsequently solved. This further requires the calculation of eigenvalues which are then compared against a noise threshold, which finally gives the values of  $u$  and  $v$ . The method is outlined visually in Fig 6.

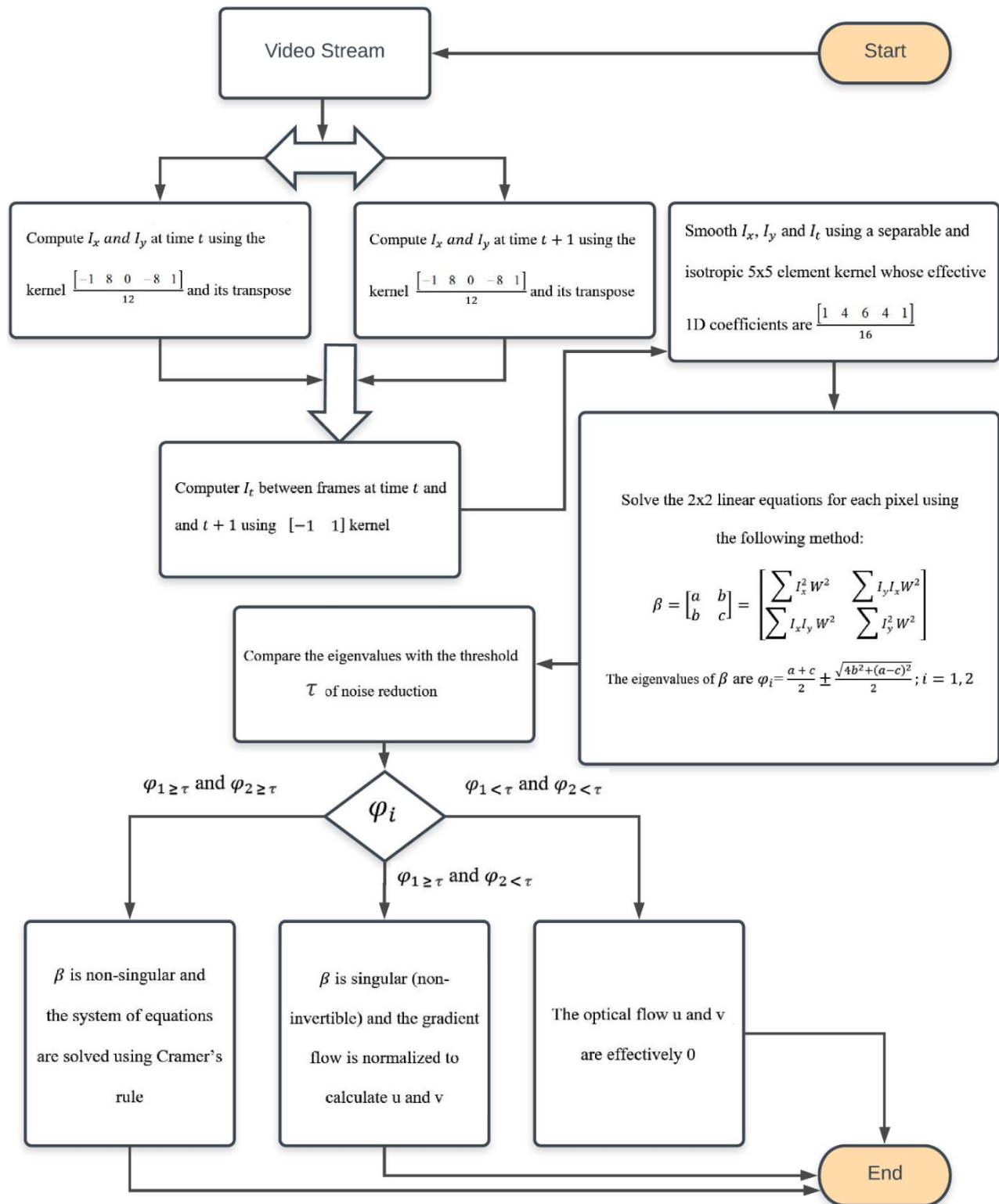


Figure 6. A fast implementation of generalized Lucas-Kanade optical flow tracking algorithm

The flow of the algorithm is simple. The input is some features or points that we want to track and the output is  $[u \ v]$  or the horizontal and vertical flow vectors. However, because the algorithm requires dividing the image into small subsections with each subsection having similar motion, this assumption might not always hold true. The constraint is a bit trickier to handle for videos of low resolution or where the object being tracked is very distant in the image signal. There are no generalized workarounds but typically image pyramids are used to deal with this problem:

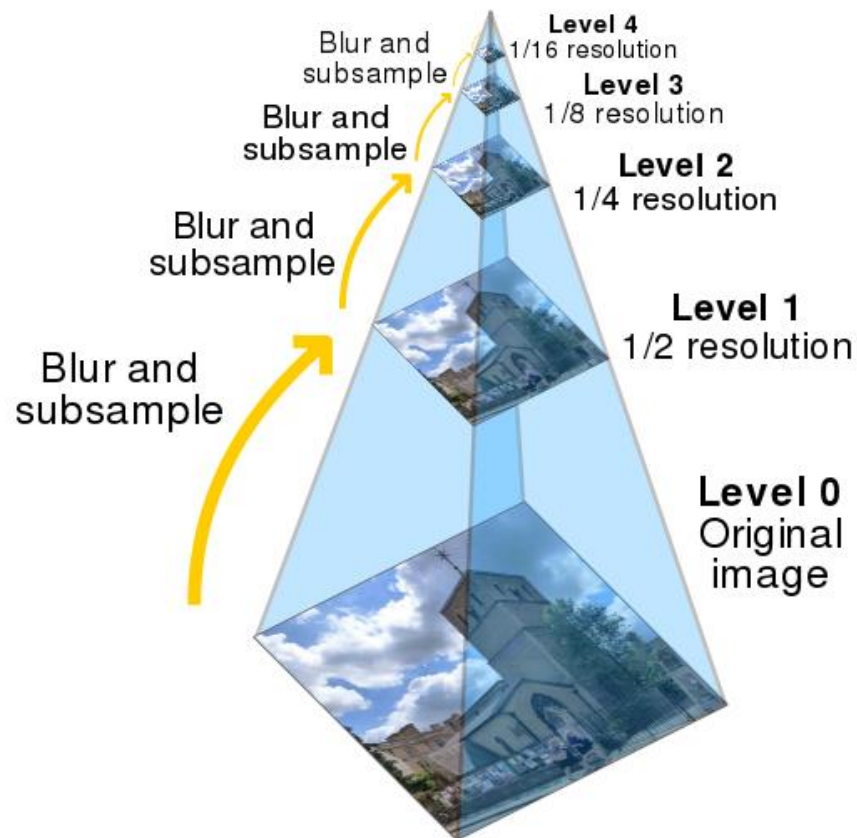


Figure 7. An image pyramid showing 5 levels of blurring and subsampling [29]. Depending on the requirements, we either go up or down the pyramid. When we go up, we effectively ‘zoom down’ the large motions into small motions and vice versa.

### 2.2.3.3 Small temporal transformations

For a feature undergoing transformation from frame at time  $t$  to  $t + \nabla t$ , either the value of  $\nabla t$  or the transformation of the features being tracked must be small enough. To put it in simple words, either the object being tracked must move really slowly over time, or the frame rate of the video stream has to be high enough for reliable tracking of the object. This ‘temporal persistence’ has been expressed in [21] as:

$$\frac{\partial I}{\partial x} \left| \left( \frac{\partial x}{\partial t} \right) + \frac{\partial I}{\partial t} \right|_{x(t)} = 0 \rightarrow I_x v + I_t = 0 \rightarrow v = -\frac{I_t}{I_x} \quad (18)$$

where,  $I_x v$  is the partial derivative of the image brightness and  $I_t$  is the gradient along time

This constraint is rather easy to satisfy by simply having a frame rate greater than 10 fps. To put that into perspective, most real-time video streams like CCTV cameras have a frame rate of at least 15 fps, so we do not have to consider anything additional. However, if the tracked object undergoes really large transformations, just having a good frame rate does not necessarily solve the problem. This is still not a significant hindrance as we can use image pyramids again to eliminate the small motions by going up the pyramid and ‘zooming’ out the large motions into small motions. By applying Lucas-Kanade on the zoomed version, we can get the optical flow vectors along with the scale [21]. Therefore having both a good frame rate and applying image pyramids alongside the Lucas-Kanade method serves to effectively satisfy the constraint. It is then necessarily true that using image pyramids with iteration in tandem with the Lucas-Kanade algorithm yields an overall better performance and that is what the proposed model uses as well.

## CHAPTER 03

### Proposed Model

#### 3.1 Block diagram of the proposed model

The proposed model consists of several different phases analyzing the input stream based on their own expertise. The basic workflow of the model is outlined in Figure 8.

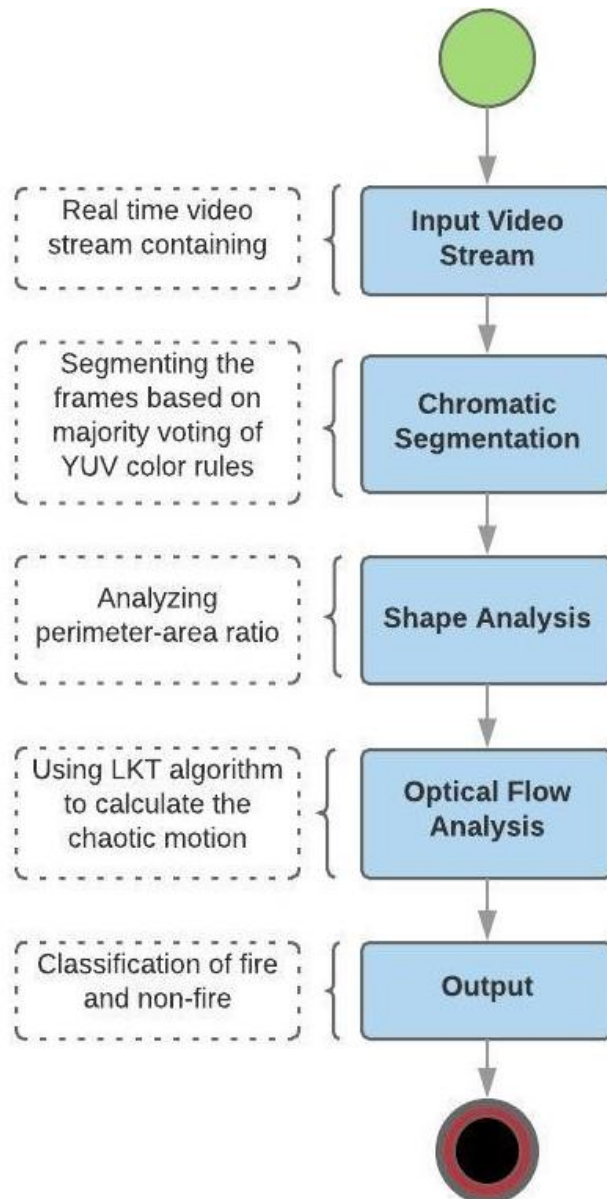


Figure 8. Block diagram of the proposed model

### 3.2 Chromatic segmentation in YUV Color Space

Before the process of chromatic segmentation is initiated, the first step that is to be carried out is the fragmentation of input video. The process is relatively simple and can be carried out merely by using library tools. The extracted frames from the input video stream are then turned in for chromatic segmentation.

At the very beginning of the chromatic segmentation phase, the appropriate color space conversion from RGB to YUV is to be made. The conversion algorithm has been previously expressed in (2). The output of the conversion is illustrated in figure 9.



Figure 9. (a) Showing the original frame from a forest fire video in RGB color space. (b) Showing the resulting image after conversion to YUV color space.

It is a commonly observed fact that non-chemical fire possesses a characteristic color that almost always exists in the red-yellow range. In most models of fire detection, chromatic segmentation exploiting this distinctive feature thus naturally acts as the primary trigger to sense the presence of fire. There are many different types of segmentation models, some of them well-devised using established techniques from the field of computer vision, others based on naïve heuristics and experimental inspection. Celik in [4] proposes a collection of six color rules that have been empirically verified. It is based on the observation that pixels in most flames exhibit a red component value that is greater than the green component value and the green component value being greater than the blue component value [4]. The hypothesis is represented by equation (19).

$$R(x, y) \geq G(x, y) > B(x, y) \quad (19)$$

Equivalent counterpart of these rules in the RGB plane are also found to exist in the YUV



plane. After using the well-known formula in [11] to convert the image from RGB to YUV, these equivalent rules can be used to classify a pixel as a fire pixel. For the generic pixel  $(x, y)$  in a frame, the rules can be expressed as:

$$r_1(x, y) = \{1 \text{ if } Y(x, y) > U(x, y) \\ 0, \text{ otherwise}\} \quad (20)$$

$$r_2(x, y) = \{1 \text{ if } V(x, y) > U(x, y) \\ 0, \text{ otherwise}\} \quad (21)$$

where  $r(x, y)$  indicates a pixel at the coordinates  $(x, y)$  which acquires the value of 1 if it manifests fire-like color and 0 otherwise.

Furthermore, it is also experimentally established that in the case of a fire pixel, its Y channel value is noticeably larger than the average Y channel value of the whole frame. These constraints can be further formulated as [5]:

$$r_3(x, y) = \{1 \text{ if } Y(x, y) > Y_{\text{mean}} \\ 0, \text{ otherwise}\} \quad (22)$$

$$r_4(x, y) = \{1 \text{ if } U(x, y) < U_{\text{mean}} \\ 0, \text{ otherwise}\} \quad (23)$$

$$r_5(x, y) = \{1 \text{ if } V(x, y) > V_{\text{mean}} \\ 0, \text{ otherwise}\} \quad (24)$$

Similarly, the average values of the Y, U and V channels in the YUV color space for an  $M \times N$  image can be delineated as follows:

$$Y_{\text{mean}} = \left(\frac{1}{M \cdot N}\right) \cdot \sum_{i=0}^M \sum_{j=0}^N Y(i, j) \quad (25)$$

$$U_{\text{mean}} = \left(\frac{1}{M*N}\right) \cdot \sum_{i=0}^M \sum_{j=0}^N U(i,j) \quad (26)$$

$$V_{\text{mean}} = \left(\frac{1}{M*N}\right) \cdot \sum_{i=0}^M \sum_{j=0}^N V(i,j) \quad (27)$$

where  $Y(i,j)$ ,  $U(i,j)$  and  $V(i,j)$  are the  $Y$ ,  $U$  and  $V$  channel values respectively of a pixel located in the position  $(i, j)$  of a frame.

Moreover, it has also been gauged that there exists a notable difference between the  $U$  and  $V$  channel values for a fire pixel [4].

$$|V(x,y) - U(x,y)| \geq t_c \quad (28)$$

However, in our experimental analysis we have observed that equation (28) does not perform equally as well when the flames in the scene are relatively brighter. We have achieved a slightly better performance in the aforementioned condition by simply splitting (28) into two equations (29) and (30) and making a minor modification to it to get rid of the threshold value:

$$r_6(x,y) = \{1 \text{ if } (0.025 * V(x,y) - 0.025 * U(x,y) - 1) > 0 \quad (29)$$

$$0, \text{ otherwise}\}$$

$$r_7(x,y) = \{1 \text{ if } (0.025 * U(x,y) - 0.025 * V(x,y) - 1) > 0 \quad (30)$$

$$0, \text{ otherwise}\}$$

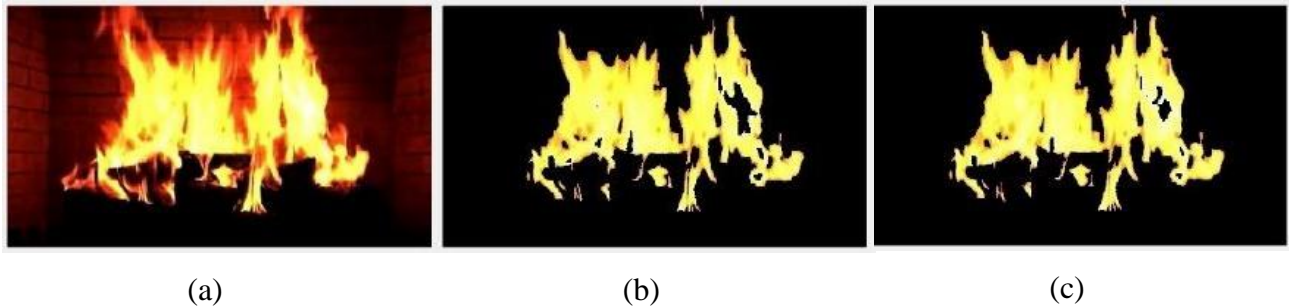


Figure 10. (a) shows the original frame of a bright flame at a close proximity. (b) shows the image segmented using (28). (c) shows the image segmented using (29) and (30). It shows a slightly better inclusion of the region of fire compared to (b).

Another disadvantage noticed in many color filtering algorithms [2, 3] is that they perform a logical conjunction of all the formulae in their model to select the region of fire. This results in occasional failure when the fire in the frame is not at close proximity. It is also to be noted that as subsequent phases rely on the output of this phase, failure to extract a blob in this phase means the subsequent phases will not even get the opportunity to offer their own assessment, resulting in a reduction of overall accuracy. The proposed model attempts to overcome this limitation by incorporating a liberal score based system where each individual rule can submit their own vote. The final selection of pixels is based on the decision of the majority as expressed in equation (31):

$$\varphi_M(x, y) = \sum_{k=1}^7 r_k(x, y) \quad (31)$$

$$F_{candidate}(x, y) = \{1 \text{ if } \varphi_M(x, y) \geq M \\ 0, \text{ otherwise}\}$$

Where  $M$  denotes the number of rules that must provide an affirmative response. The value of  $M$  has been flexibly adjusted to 4.  $\varphi_M(x, y)$  is the number of rules that has selected pixel  $(x, y)$  as a fire pixel.  $F_{candidate}(x, y)$  is a candidate fire pixel  $(x, y)$  which acquires the value of 1 in case of an actual fire pixel and 0 otherwise.

Although a smaller value for  $M$  in (31) does result in more false positives, we do not want to miss out any potential fire pixel by having a too restrictive set of rules. The smaller the value for  $M$  in (31), the greater the amount of false positives. The subsequent phases are designed to remediate this problem and eliminate any false positives arising in this phase. As is evident in Figure 11, the majority decision rule can extract fire regions more accurately.

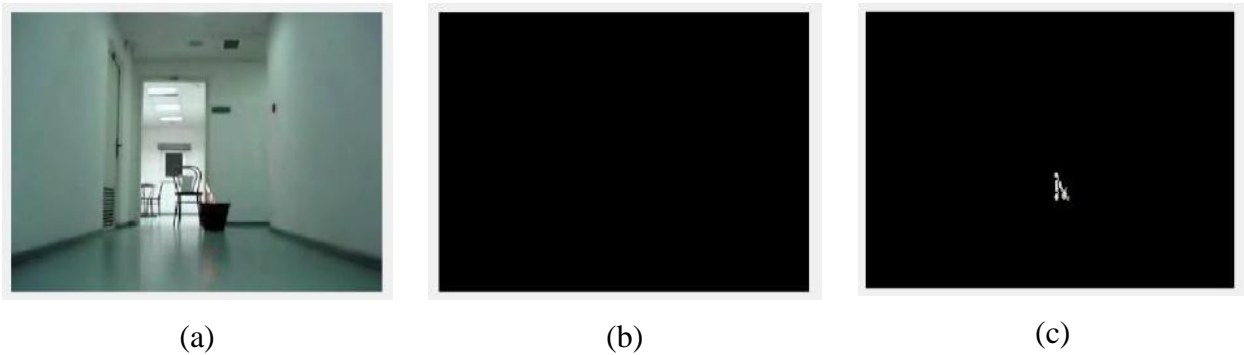


Figure 11. (a) shows a frame from a video with a fire generated in a bucket in indoor condition. (b) shows the binary mask of (a) without the majority voting rule specified in (31) applied. (c) shows the binary mask of (a) when majority voting rule is applied.

A flow diagram illustrating the entire chromatic segmentation phase with our enhanced majority voting rule is outlined in Figure 12.

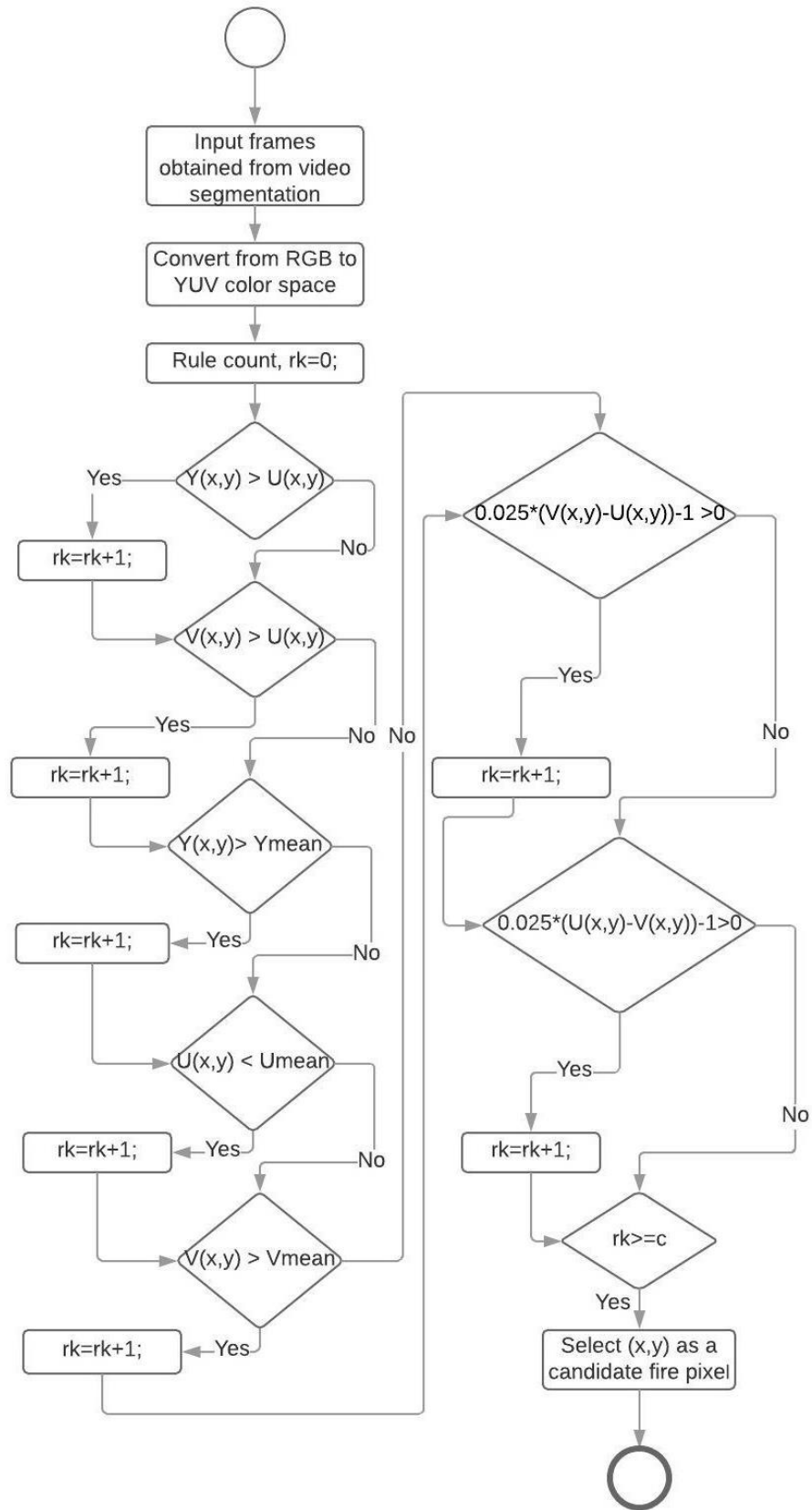


Figure 12. An overview of the chromatic segmentation part of the proposed model

## 3.2 Growth Analysis

Another typical attribute inherent to flames is its complex boundaries and its expeditious fluctuation as a function of time. In this phase of our model, extracted blobs from two successive frames from the chromatic segmentation phase is chosen for analysis. A ratio of the perimeter and area of the blobs in both the frames is calculated as suggested in [5]:

$$R_t = \frac{P_t}{A_t} \quad (32)$$

It is predicted that the ratio calculated in (32) for fire blobs should be markedly larger than the ratio calculated for ordinary rigid objects that somehow mistakenly slip through the chromatic segmentation stage. To further validate the region, the ratio for a blob extracted in frame  $t$  is compared with the blob closest to the centroid of the previous blob in frame  $t + 1$  as specified in equation (33).

$$S_{tv} = \frac{r_t - r_{t+1}}{r_t} \quad (33)$$

The growth analysis phase classifies the frame  $t$  as a fire scene if  $S_{tv}$  in (33) is larger than a threshold  $t$ . This is the type of task that is usually suitable for machine learning algorithms but as this phase of the model is less of our focus, for our dataset  $t$  has been simply adjusted to be 0.4. This analysis phase exists solely to remove spurious blobs of ordinary non-fire objects that evince the characteristic color of fire.

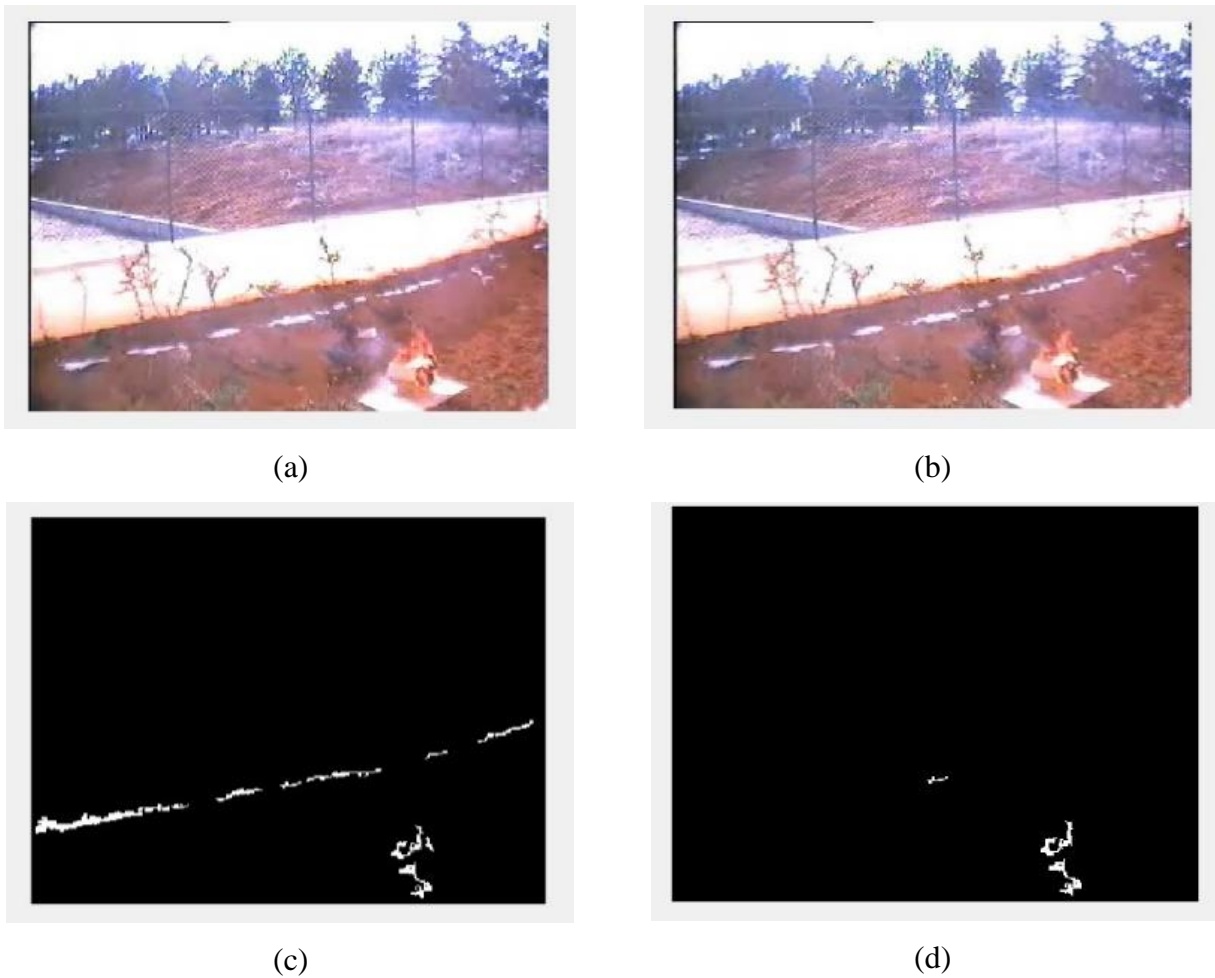


Figure 13. (a) and (b) shows two sequential frames at times  $t$  and  $t + 1$  from an outdoor video containing both fire region and non-fire region that shares the color of fire. (c) Shows the binary mask of the frame before growth analysis is undertaken (d) Shows the binary mask of the frame after the growth feature has been analyzed. It is apparent that nearly all of the non-growing regions in the scene has been successfully removed.

A flow diagram illustrating the entire process of growth analysis of fire over a sequence of two frames is outlined in Figure 14.

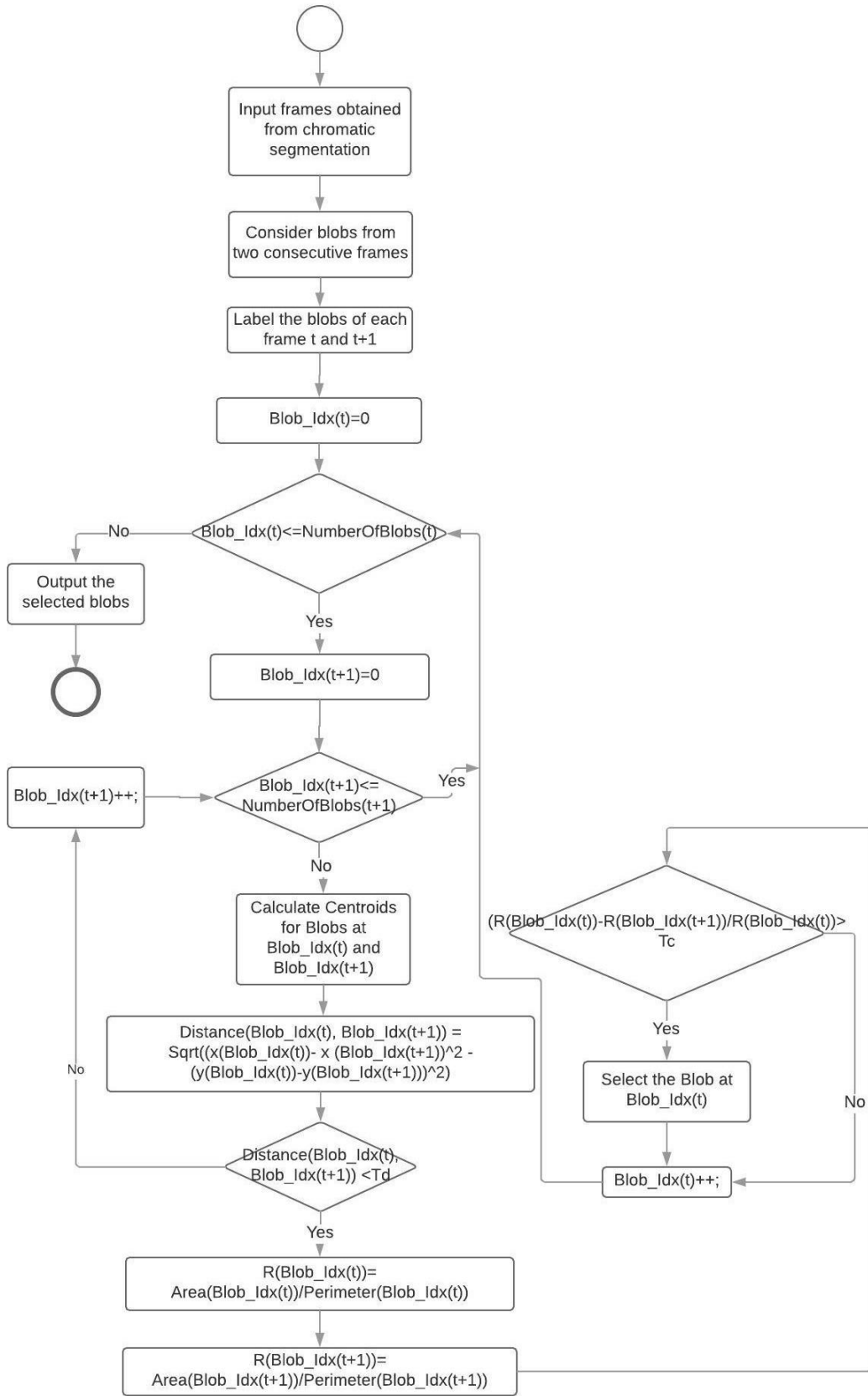


Figure 14. A flow diagram that illustrates the growth analysis phase of the proposed model



### 3.3 Optical Flow Analysis

Fire is a turbulent spatiotemporal phenomenon with its edges and inner texture exhibiting a disorderly displacement as a function of time. Fire plumes in general do not exhibit any specific pattern or shape and the velocity vectors of the feature points on the texture of fire plumes are very unstable and non-deterministic. Verily, this feature is so distinctive of fire that being able to detect it cuts down the amount of false positive by a great degree. In order to calculate the optical flow of fire, Suchet in [21] adopts the Lucas Kanade optical flow pyramid. We propose an enhanced model based on the optical flow analysis algorithm that is derived from [21] but has superior precision and is less expensive in terms of computing resources, making it better suited for early detection of fire. Figure 15 shows the turbulence of fire over three consecutive frames.

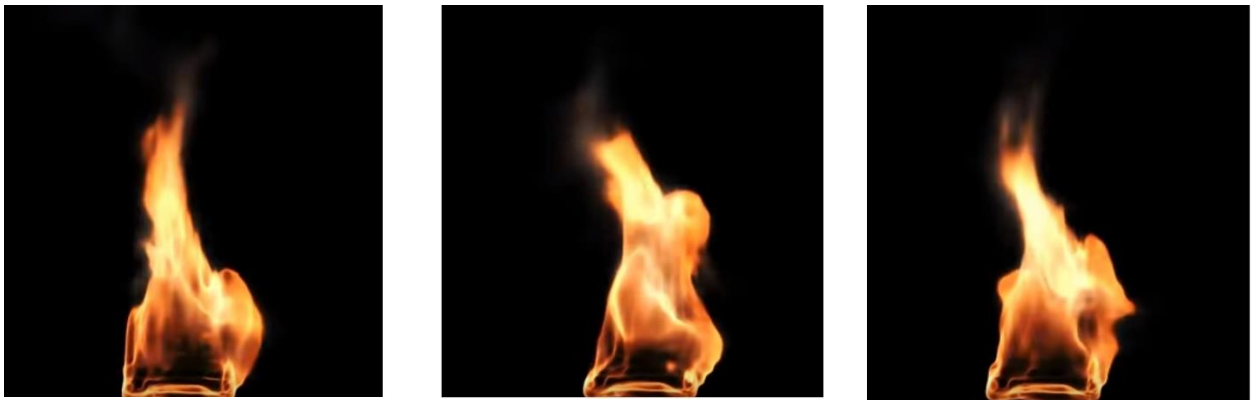


Figure 15. The highly chaotic behavior of flames with no definitive structure or shape

Optical flow analysis techniques can help us understand the turbulent nature of fire better. Utilizing the Lucas Kanade optical flow tracking algorithm, the proposed model makes an attempt to take advantage of this behavioral trait of fire. As described previously in the literature review, the Lucas Kanade tracker requires a set of corners or features to be tracked and it returns an array of flow vectors for every corner being analyzed. The implication is that features or corners of interest must be detected at first in the surface of the potential fire regions. As a consequence, this makes the quality of tracking analysis have partial dependency on the feature or corner detection algorithm used. In our investigation we have observed that the importance of this detail is often left out in models incorporating optical flow analysis including [21].

### 3.3.1 Corner-detection

There are many different corner or feature detection algorithms to choose from. The most common one is the Shi-Tomasi detector which is a slight modification of the original Harris corner detector. Harris corner selects a corner based on certain criteria. A function takes two eigenvalues for every pixel and manipulates them to calculate a score. If the score for a pixel exceeds a threshold, it is marked as a corner. Shi-Tomasi on the other hand does away with the function and uses the eigenvalues alone to calculate the score. It was experimentally verified that this criteria for scoring performs much better [30]. With just a minor modification, Shi-Tomasi can achieve much better performance than Harris.

However, for real-time optical flow analysis, we find that even Shi-Tomasi corner detection is relatively expensive. Testing several detection algorithms, we settled with the Features from Accelerated Segment Test (FAST) algorithm as it provides the best performance without a forsaking accuracy [31]. Flow analysis is a computationally intensive operation, so we want this initial feature extraction step to be as fast as possible. To choose a point  $P$  at pixel  $(x, y)$  as a corner, the FAST detector at first considers a Bresenham circle of radius 3 around  $P$ . Then it looks at the intensity of the candidate point  $P$  and compares it to its surrounding points in the Bresenham circle. The constraints for the classification can be written as [32]:

$$\forall x \in S, I_x > I_p + \alpha \quad (34)$$

$$\forall x \in S, I_x < I_p - \alpha \quad (35)$$

where  $S$  is the set of  $N$  neighboring pixels in the Bresenham circle,  $I_x$  the intensity of the candidate pixel  $x$ ,  $I_p$  the intensity of the candidate point and  $\alpha$ , a threshold value for intensity

If either of the two constraints specified in equations (34) and (35) can be satisfied, the candidate pixel  $P$  is selected as a corner. If learning algorithms are not utilized, the value of  $N$  is usually concretized to be  $3/4^{\text{th}}$  of the total number of pixels in the Bresenham circle, which is 12. This can result in over-classification but this is good enough for our purposes as it provides a good tradeoff between performance and accuracy.

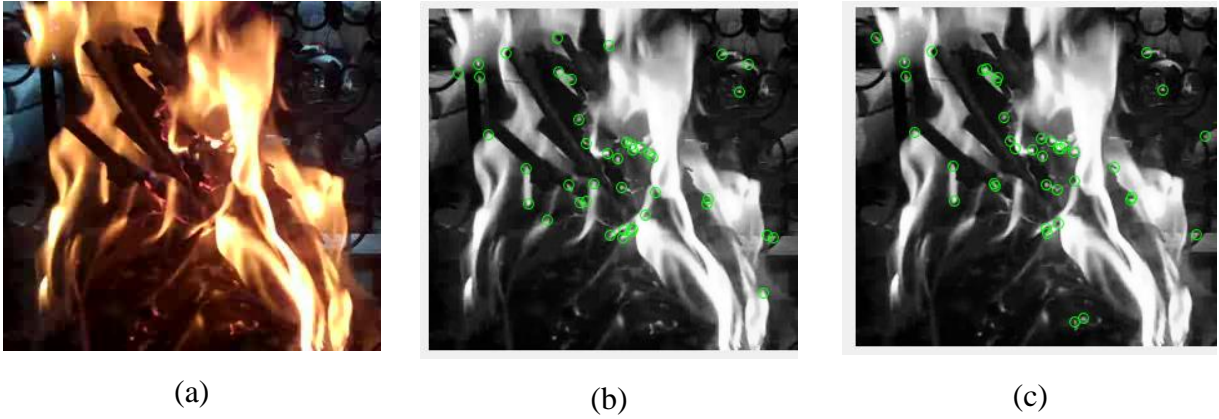


Figure 16. The original fire frame (b) Showing features detected by Shi-Tomasi and (b) showing features detected by FAST detector. As can be seen, both the detectors have a very similar quality of detection.

For the frame in Figure 16 (a), it is obvious that both detectors are good in terms of accuracy. Furthermore, FAST detector is also found to be better in performance when profiled as shown in Table 2.

Table 2: The two algorithms Shi-Tomasi and FAST compared in performance timing for a single frame. Both tried to find up to 50 corners in the scene.

Algorithm	Average Performance Timing (seconds)
Shi-Tomasi	0.076
FAST	0.031

From Table 2, it is evident that FAST detector is more than two times faster than Shi-Tomasi Detector. From Figure 16, we have also seen that both are equally accurate in terms of detection. As a whole therefore this is the corner detector of our choice. We use the FAST detector algorithm to rapidly detect up to 50 corners in the regions containing fire-like colors. These corners are then provided to the Lucas-Kanade Tracker (LKT) algorithm which subsequently tracks their positions from frame  $t$  to  $t + 1$ .

### 3.3.2 Motion velocity vector calculation

The LKT algorithm uses the steps described in Figure 6 to calculate the flow vectors  $[u \ v]$  of the corners passed to it. These represent the velocity vector for all the corners in their respective frames which can be represented by equations (36) and (37).

$$p = [p_x, p_y]_i, i = 0, 1, 2, \dots, n \quad (36)$$

$$q = [q_x, q_y]_i, i = 0, 1, 2, \dots, n \quad (37)$$

where  $p$  and  $q$  denote the starting and ending points of a corner detected in frame  $t$  to frame  $t + 1$ , and  $n$  indicating the number of corners.



Figure 17. (a) Shows the original frame from a video with fire in outdoor condition. (b) Shows the velocity vectors calculated of corners detected in the flame region in (a).

Using the vectors calculated in equations (36) and (37) we can have an approximate estimate of the complexity by measuring the average flow rate in the frame:

$$F = \frac{1}{n} \sum_{k=1}^n \sqrt{(p_{xk} - q_{xk})^2 + (p_{yk} - q_{yk})^2} \quad (38)$$

where,  $n$  is the number of corners and  $k$  is an iterator over the number of corners

This complexity of flow rate portrays the shambolic, non-deterministic movement of local features on the surface of fire. As a result, compared to ordinary objects, this value for fire will typically be larger. This is also where the proposed technique starts to take a different direction from [21]. Suchet initially calculates the flow rate and takes note of it as a reference value. The flow rate is then calculated  $n$  times for the next  $n$  frames and the calculated values are compared to the initial reference value to have a measure of the *variation* of flow rate. Afterwards, the measured variation is compared with a threshold that was determined empirically. The exact value of  $n$  is also not mentioned which is why it was hard to perfectly reproduce their expected result in our research. The equations of their model is described in (39).

$$F_v = \frac{1}{n} \sum_{j=1}^n \left( \frac{1}{m} \sum_{i=0}^m \left( \sqrt{(p_{yi} - q_{yi})^2 + (p_{xi} - q_{xi})^2} - F_a \right) \right)_j \quad (39)$$

Where  $n$  is the number of frames to calculate the flow rate in and  $F_a$  is the initial flow rate

Moreover, the threshold of 2 that  $F_v$  in (39) was claimed to exceed in case of fire [21], also proved to be unsound for our dataset. When tested against a more versatile dataset, as presented in our experimental analysis, this approach results in a startling amount of false positives, confining the practicality of this method to a more limited set of environments.

The proposed technique on the other hand directly uses the value of flow rate complexity to evaluate the possibility of fire. The value of  $n$  has been concretized to 10 as this has yielded the best result for the overall analysis. The average flow rate is thus calculated ten times for ten consecutive frames. A counter is maintained that keeps track of the number of times a threshold  $\alpha$  is exceeded. Every time there is a hit in the counter, it means the flow rate in the scene is unusually large. Finally, if the complexity happens to exceed the threshold at least half of the times in  $n$  attempts, the  $n$ -frames sequence is designated as a fire scene. To keep the computational complexity at a minimum, the corner detection and LKT algorithms are not applied on the video stream at all times. Instead it only kicks in when a blob is detected from the shape analysis step. A flow diagram of the process is illustrated in Figure 6.

However, the detector can still be misfired when one is trying to intentionally create a turbulent motion with an object. Turbulence can be better estimated by calculating the optical mass

transport of fire. The equation for this is formulated in (40).

$$T = Mean \left( \frac{I}{2} | p_{OMT} * q_{OMT} |_1^5 \right) \quad (40)$$

Where  $T$  is the turbulence of the velocity vectors, and  $p_{OMT}$  and  $q_{OMT}$  being the optical mass transport values for the vectors.

This turbulence is then trained for by using a neural network. Over 1000 frame sequences are extracted and trained using scaled conjugate gradient back-propagation. It has been able to extract a pattern that yield very good results. Overall, not only does this reduce the amount of intensive computation, there is also a significant cutback on the amount of false positives as clear in the experimental result in Table 4. A comparative analysis of the chaotic velocity vectors for fire and uniform, smoother velocity vectors for non-fire scenes is illustrated in Figure 18 and Figure 19.

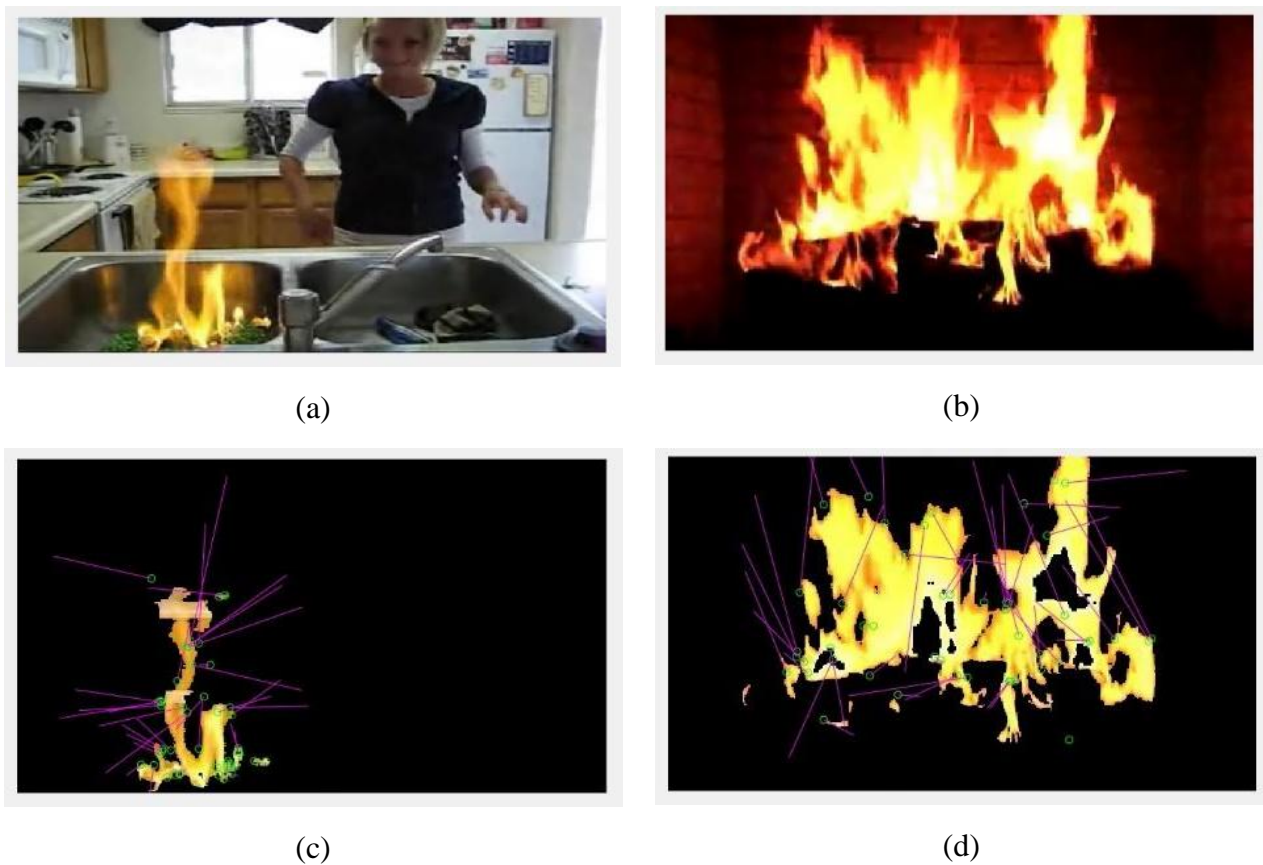
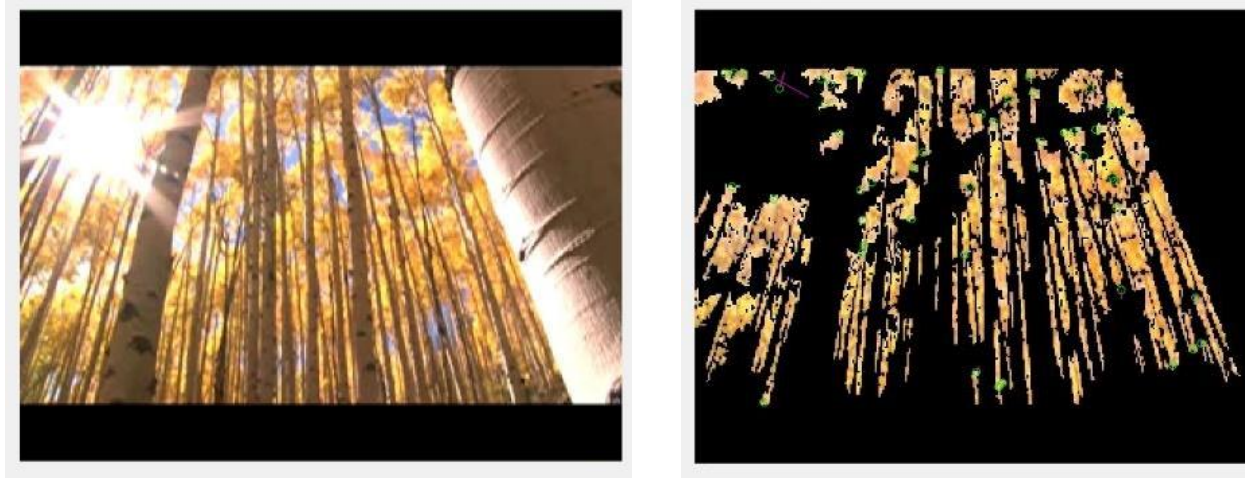


Figure 18. (a) and (b) Shows the original frames of a fire ablaze up close. (c) and (d) Shows the respective velocity vectors of (a) and (b). As noticeable, the velocity vectors are quite shambolic and disperse for fire.

The chaotic motion vectors in the fire scenes in Figure 18 is also easily discernible from the relatively smooth motion vectors in the non-fire scene in Figure 19.



(a)

(b)

Figure 19. (a) Shows the original frame of sunlight beaming through a forest consisting of trees and leaves exhibiting the distinctive color of fire. (b) Shows the respective motion vectors.

The model as a whole provides satisfactory results when tested on a diverse dataset. Chapter 4 will discuss in details the experimental analysis phase and how it excels over some of the competing models.

# CHAPTER 04





## Experimental Analysis

### 4.1 Experimental Setup

To assess the efficacy of the proposed model, the system has been tested using a diverse dataset containing videos of a wide range of scenarios. The dataset has been predominantly populated using videos from the Mivia [5] dataset and some using Zenodo and YouTube. The dataset includes both videos containing fire of varying illumination and non-fire videos that contains ordinary objects exhibiting fire-like color. They are in all sorts of environments including indoor, outdoor and forest fires. All of the videos have been adjusted to 320x240 resolution. Some of the scenarios in the dataset are depicted in Table 3.

#### 4.1.1 Sample scenes from the dataset

Table 3. Sample scene and description of videos in the dataset.

Video	Fire	Outdoor	Description	Thumbnail
Fire1	Yes	Yes	A fire generated in a bucket with a backdrop exhibiting fire-like color	
Fire2	Yes	Yes	A fire set in a very distant region, hard to notice	
Fire6	Yes	Yes	A fire generated in a red ground	
Fire8	Yes	Yes	A forest fire with black and white smoke in the background	



Fire13	Yes	No	A fire generated in a bucket	
Fire33	Yes	No	A close shot of a fireplace	
Fire35	Yes	No	A fire set by a woman in a kitchen sink	
Fire15	No	Yes	Smoke plumes from a bucket with sunlight shimmering in the background	
Fire27	No	Yes	Distant smoke in a city with red buildings	
Fire32	No	Yes	A forest with trees and leaves having characteristic color of fire with sunlight shimmering in the background	
Fire24	No	No	Smoke in a room with light in the background	
Fire30	No	No	A man moving with a red ball with red lockers in background	

#### 4.1.2 Methodology of calculating accuracy

The proposed model performs three steps to classify a scene as fire. First it does chromatic segmentation based on majority voting rules, then it eliminates the spurious regions via growth analysis and then finally passes the remainder blobs to the optical flow analysis phase for flow vector detection. The flow analysis technique takes 10 frames at a time to analyze the motion vectors and based on the complexity over these 10 frames, it evaluates the turbulence and classifies the 10-frame sequence as fire. If the scene has fire, it gives 1 as output and 0 otherwise. Thus to calculate the overall accuracy of the detection, the output of the last step is compared with the actual scenario in the video. If it matches, we consider it as a correct detection, if it doesn't we consider it as an incorrect detection. If the phenomenon is one such where the ground truth is that fire exists and if fire is detected, then it is considered a true positive (TP), otherwise if it is labelled as non-fire, then it is deemed as a false negative (FN). Conversely, if the situation is one such where the ground truth is that fire doesn't exist and if fire is not detected, then it is considered a true negative (TN), otherwise if it is labelled as fire, then it is deemed as a false positive (FP). Table 4 shows the percentages of true positives and false positives by the models presented by [19] and [20] as well as the proposed model. The performance of the detector is further analyzed using the precision and recall values. Precision is a ration that represents the relevant information among the retrieved information. It is used in conjunction with recall that represents the amount of relevant information gathered from the total amount of relevant information. Precision and recall can be calculated using (40).

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \quad (40)$$

where  $TP$  and  $FP$  stands for the true positive and the false positive values respectively.

The two quantities are often used in conjunction to find out the F1 score to give a single quantification of the model. Equation 41 shows how the F1-score is calculated.

$$F1score = 2 * \frac{(Precision * Recall)}{Precision + Recall} \quad (41)$$

## 4.2 Experimental Results

The whole proposed model has been simulated and furthermore other models have been chosen for comparative analysis. The models were tested against the entire dataset and different criteria were recorded and tabulated in Table 4.

Table 4: The accuracy of different videos in the dataset, containing both fire and non-fire videos.

Video	Khan		Suchet		Proposed Model	
	True Positive (%)	False Positive (%)	True Positive (%)	False Positive (%)	True Positive (%)	False Positive (%)
1 (Fire)	95.6	7.8	90.3	8.2	92.3	4.3
2 (Fire)	76.5	0	81	0	77.1	0
3 (Fire)	98.2	0	97.7	0	98.5	0
4 (Fire)	100	0	100	0	100	0
5 (Fire)	98.5	0	98.6	0	92.4	0
6 (Fire)	93.5	15.6	92.5	12.4	94.9	21
7 (Fire)	100	0	100	0	100	0
8 (Fire)	100	0	100	0	100	0
9 (Fire)	95.8	4.2	94	22.2	97.5	12
10 (Fire)	100	0	100	0	100	0
11 (Fire)	100	0	97	0	98.8	0
12 (Fire)	99.2	0	99.5	0	98.1	0
13 (Fire)	85	6.72	81	17	91	12.5
14 (Fire)	96	5.1	95	2.5	96.5	3.2
15 (Not)	90.3	3.2	91	5.6	91	2.3
16 (Not)	100	0	100	0	100	0
17 (Not)	95	5.6	92.2	2.9	92.4	2.2
18 (Not)	95.5	12.6	94.5	10.1	95	7.5
19 (Not)	100	0	100	0	100	0
20 (Not)	72	0	91	5.6	91	0

Table 5: The accuracy of the proposed model compared to Khan [19] and Suchet [21].

Model	Average Fire Detection Accuracy	Average False Fire Detection Accuracy
Khan	94.56%	4.04%
Suchet	94.77%	4.33%
Proposed	95.62%	3.25%

As shown in the comparative analysis in Table 5, the proposed model has produced an overall accuracy of 95.62% in its ability to detect the true fire scenes correctly, which is better than the accuracy of both Khan and Suchet’s techniques. Additionally, it has also obtained a lower false positive rate of 3.25%. The precision and recall of the proposed model was also recorded and tabulated in Table 6. The precision and recall of the competing models are also tabulated in Table 7.

Table 6: Precision and recall of the proposed model in identifying fire among the different videos in the dataset, containing both fire and non-fire videos.

Video	True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)	Precision	Recall
1 (Fire)	92.3	4.3	7.7	95.7	0.9554865	0.923
2 (Fire)	77.1	0	22.9	100	1	0.771
3 (Fire)	98.5	0	1.5	100	1	0.985
4 (Fire)	100	0	0	100	1	1
5 (Fire)	92.4	0	7.6	100	1	0.924
6 (Fire)	94.9	21	5.1	79	0.8188093	0.949
7 (Fire)	100	0	0	100	1	1
8 (Fire)	100	0	0	100	1	1
9 (Fire)	97.5	12	2.5	88	0.8904109	0.975
10 (Fire)	100	0	0	100	1	1
11 (Fire)	98.8	0	1.2	100	1	0.988
12 (Fire)	98.1	0	1.9	100	1	0.981

13 (Fire)	91	12.5	9	87.5	0.8792270	0.91
14 (Fire)	96.5	3.2	3.5	96.8	0.9679037	0.965
15 (Not)	91	2.3	9	97.7	0.9753483	0.91
16 (Not)	100	0	0	100	1	1
17 (Not)	92.4	2.2	7.6	97.8	0.9767441	0.924
18 (Not)	95	7.5	5	92.5	0.9268292	0.95
19 (Not)	100	0	0	100	1	1
20 (Not)	91	0	9	100	1	0.91

Table 7: Precision and recall of the models proposed by Khan and Suchet for the given dataset.

Video	Khan		Suchet	
	Precision	Recall	Precision	Recall
1 (Fire)	1	0.91	0.916751269	0.903
2 (Fire)	0.924564797	0.956	1	0.81
3 (Fire)	1	0.765	1	0.977
4 (Fire)	1	0.982	1	1
5 (Fire)	1	1	1	0.986
6 (Fire)	1	0.985	0.881792183	0.925
7 (Fire)	0.857011916	0.935	1	1
8 (Fire)	1	1	1	1
9 (Fire)	1	1	0.808950086	0.94
10 (Fire)	0.958	0.958	1	1
11 (Fire)	1	1	1	0.97
12 (Fire)	1	1	1	0.995
13 (Fire)	1	0.992	0.826530612	0.81
14 (Fire)	0.926733537	0.85	0.974358974	0.95
15 (Not)	0.949554896	0.96	0.942028986	0.91
16 (Not)	0.965775401	0.903	1	1
17 (Not)	1	1	0.969505783	0.922
18 (Not)	0.944333996	0.95	0.903441683	0.945
19 (Not)	0.883441258	0.955	1	1
20 (Not)	1	1	0.942028986	0.91

Figure 20 shows the graphical representation of F1-scores of different models in comparison to the proposed model.

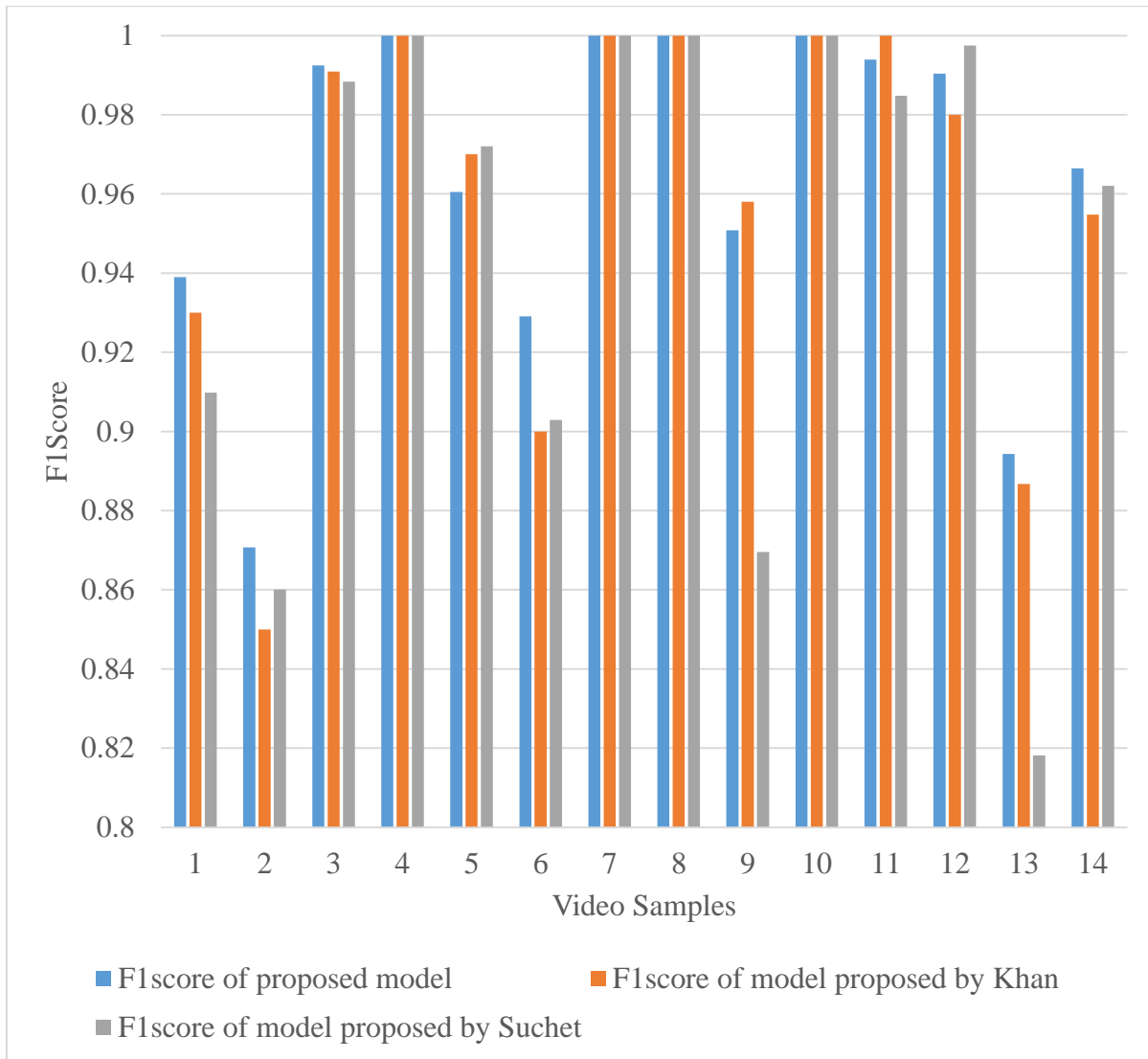


Figure 20. Comparative bar histogram of F1score for fire samples

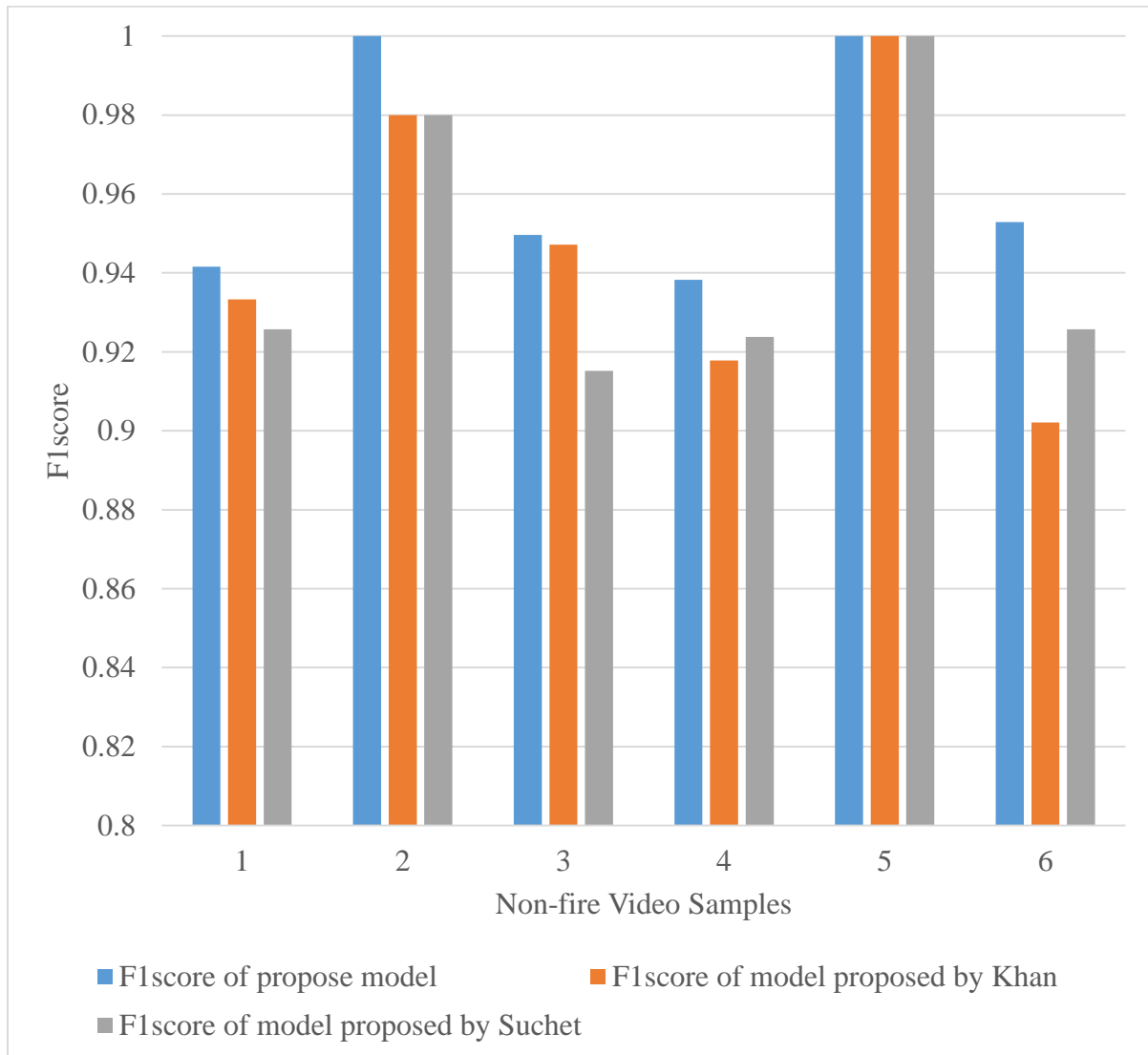


Figure 21. Comparative bar histogram of F1score for non-fire samples

The ROC curves for the neural network trained is displayed in Figure 22. As it can be seen the ROC curves are pretty good for the dataset used, which means it has a high true positive rate and still maintains a low false positive rate.

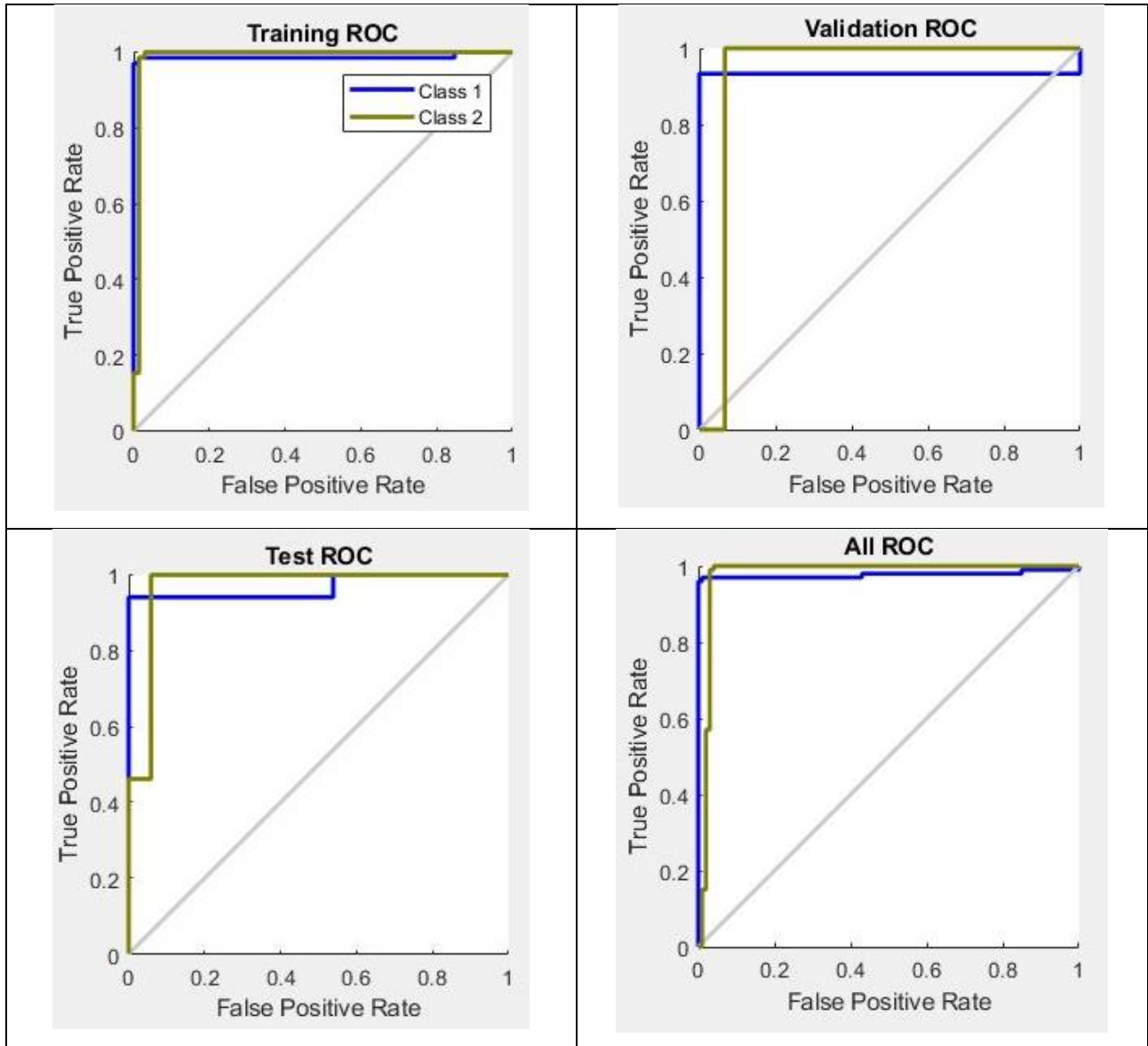


Figure 22. The ROC curves for the neural network trained for detecting turbulence

### 4.3 Analysis

The proposed model has successfully been able to detect fire in diverse environments and have given a satisfactory performance. It is apparent from the experimental results that the accuracy of the proposed model surpasses that of [19] and [21]. The primary reasons behind this being the fact that this model incorporates a majority voting system of color rules that most other models do not. It also uses an effective optical flow algorithm that decreases the number of false positives. Contrary to [21], this model uses a faster algorithm for edge detection and thus the performance time for the proposed model is faster.



# CHAPTER 05

## CONCLUSION

### 5.1 Concluding Remarks

To sum up, we have put forward a robust fire detection system in this paper that attempts to explore various properties of fire such as color, shape and shambolic motion. Two of the most rudimentary optical traits of fire were detected using a collective chromatic segmentation algorithm and contour analysis. At the tertiary step of the procedure, we have employed enhanced Lucas-Kanade optical flow algorithm to distinguish the turbulent movement of fire plumes. A miscellaneous dataset have been used to test the proposed model and the model has been further compared with some existing models and by doing so, the efficiency of the proposed model have been established. Although the model showed promising prospect, an average accuracy of 95.62% to be precise, it is not without limitations. Future work can incorporate fixing these limitations to make the system more robust.

### 5.2 Limitations and Future Works

Although our system shows an improved accuracy compared to existing models, there still remains room for improvement. The proposed model does not work well for chemical fire as they exhibit other colors besides the typical red-yellow range, but this should not pose much of a problem since in most cases hazardous fires are non-chemical fire. However, this is something that a proper fire detection system should incorporate. Another limitation of the system is that, it does not take into account that the incoming videos from the CCTV cameras could change in orientation. Meaning the model makes the assumption that the video is obtained from a static camera. This also should not pose a big threat since the surveillance cameras that our model is targeting does not really have unstable motion. An interesting approach to the one we have already adhered would be to incorporate a hybrid neural network and training it with a collection of flow rate complexities as a feature vector. This could potentially help reduce false positive rates by an even greater degree. Also, instead of just calculating the average flow vector, there could be other interesting patterns regarding the flow rate which could be exploited but we have not explored.

## REFERENCES

1. T. Çelik, "Fast and Efficient Method for Fire Detection Using Image Processing," *ETRI Journal*, vol. 32, no. 6, pp. 881–890, Jun. 2010.
2. T.-H. Chen, P.-H. Wu, and Y.-C. Chiou, "An early fire-detection method based on image processing," in *ICIP '04. 2004 International Conference on*, 2004, vol. 3, pp. 1707–1710.
3. T. Toulouse, L. Rossi, T. Celik, and M. Akhloufi, "Automatic fire pixel detection using image processing: a comparative analysis of rule-based and machine learning-based methods," *Signal, Image and Video Processing*, vol. 10, no. 4, pp. 647–654, 2015.
4. T. Çelik and H. Demirel, "Fire detection in video sequences using a generic color model," *Fire Safety Journal*, vol. 44, no. 2, pp. 147–158, 2009.
5. P. Foggia, A. Saggese, and M. Vento, "Real-Time Fire Detection for Video-Surveillance Applications Using a Combination of Experts Based on Color, Shape, and Motion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 9, pp. 1545–1556, 2015.
6. B. H. Cho, J. W. Bae and S. H. Jung, "Image Processing-Based Fire Detection System Using Statistic Color Model," *2008 International Conference on Advanced Language Processing and Web Information Technology*, Dalian Liaoning, 2008, pp. 245-250.
7. W.-B. Horng, J.-W. Peng, and C.-Y. Chen, "A new image-based real-time flame detection method using color analysis," *Proceedings. 2005 IEEE Networking, Sensing and Control*, pp. 100–105, 2005.
8. L.-H. Chen and W.-C. Huang, "Fire Detection Using Spatial-temporal Analysis," in *Proceedings of the World Congress on Engineering 2013*, 2013, vol. 3.
9. J. Seebamrungsat, S. Praising and P. Riyamongkol, "Fire detection in the buildings using image processing," *2014 Third ICT International Student Project Conference (ICT-ISPC)*, Nakhon Pathom, 2014, pp. 95-98.
10. "Color space, covering greyscale, RGB, YUV and HSV," *Graphic Content*, 07-Jan-2016. [Online]. Available: <https://slohia247.wordpress.com/2015/07/14/color-space-covering-greyscale-rgb-yuv-and-hsv/>. [Accessed: 24-Mar-2018].
11. D. Shon, M. Kang, J. Seo, and J.-M. Kim, "Early Fire Detection Using Multi-Stage Pattern Recognition Techniques in Video Sequences," in *Frontier and Innovation in Future Computing and Communications*, pp. 161–168.

12. V. B. Celen and M. F. Demirci, "Fire Detection In Different Color Models," in *WorldComp Proceedings*, 2012.
13. J. Chen, Y. He, and J. Wang, "Multi-feature fusion based fast video flame detection," *Building and Environment*, vol. 45, no. 5, pp. 1113–1122, 2010.
14. Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Comparative study of background subtraction algorithms," *Journal of Electronic Imaging*, vol. 19, no. 3, 2010.
15. L. Rossi and M. Akhloufi, "Dynamic Fire 3D Modeling Using a Real-Time Stereovision System," *Technological Developments in Education and Automation*, pp. 33–38, 2009.
16. T. Qiu, Y. Yan, and G. Lu, "An Autoadaptive Edge-Detection Algorithm for Flame and Fire Image Processing," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 5, pp. 1486–1493, May 2012.
17. Q. Jiang and Q. Wang, "Large Space Fire Image Processing of Improving Canny Edge Detector Based on Adaptive Smoothing," *2010 International Conference on Innovative Computing and Communication and 2010 Asia-Pacific Conference on Information Technology and Ocean Engineering*, Macao, 2010, pp. 264-267.
18. H. C. Bheemul, G. Lu, and Y. Yan, "Three-dimensional visualization and quantitative characterization of gaseous flames," *Measurement Science and Technology*, vol. 13, no. 10, pp. 1643–1650, Sep. 2002.
19. R. A. Khan, J. Uddin and S. Corraya, "Real-time fire detection using enhanced color segmentation and novel foreground extraction," *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, Dhaka, 2017, pp. 488-493.
20. X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, L. Gao, and L.-P. Xiao, "Video fire detection based on Gaussian Mixture Model and multi-color features," *Signal, Image and Video Processing*, vol. 11, no. 8, pp. 1419–1425, Feb. 2017.
21. S. Rinsurongkawong, M. Ekpanyapong, and M. N. Dailey, "Fire detection for early fire alarm based on optical flow video processing," *2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, 2012.
22. M. Mueller, P. Karasev, I. Kolesov and A. Tannenbaum, "Optical Flow Estimation for Flame Detection in Videos," in *IEEE Transactions on Image Processing*, vol. 22, no. 7, pp. 2786-2797, July 2013.

23. S. Harlapur and D. K. R. Nataraj, “Fire Detection using Optical Flow Method in Videos,” *International Journal of Engineering Research and Technology*, vol. V4, no. 05, May 2015.
24. H. Chao, Y. Gu, and M. Napolitano, “A Survey of Optical Flow Techniques for Robotics Navigation Applications,” *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1-4, pp. 361–372, Dec. 2013.
25. R. Krishnamurthy, P. Moulin and J. Woods, "Optical flow techniques applied to video coding," *Proceedings., International Conference on Image Processing*, Washington, DC, 1995, pp. 570-573 vol.1.
26. D. Zhang and Y. Wang, "Real-time fire detection using video sequence data," *2016 Chinese Control and Decision Conference (CCDC)*, Yinchuan, 2016, pp. 3620-3623.
27. “Optical Flow,” *MATLAB & Simulink*. [Online]. Available: <https://www.mathworks.com/discovery/optical-flow.html>. [Accessed: 25-Mar-2018].
28. W. H. Lim, “A novel adaptive color to grayscale conversion algorithm for digital images,” *Scientific Research and Essays*, vol. 7, no. 30, pp. 2718–2730, Aug. 2012.
29. “Image pyramid,” *Wikimedia Commons*. [Online]. Available: [https://commons.wikimedia.org/wiki/File:Image\\_pyramid.svg](https://commons.wikimedia.org/wiki/File:Image_pyramid.svg). [Accessed: 25-Mar-2018].
30. C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” *Proceedings of the Alvey Vision Conference 1988*, 1988.
31. E. Rosten and T. Drummond, “Machine Learning for High-Speed Corner Detection,” *Computer Vision – ECCV 2006 Lecture Notes in Computer Science*, pp. 430–443, 2006.
32. J. Bresenham, “A linear algorithm for incremental digital display of circular arcs,” *Communications of the ACM*, vol. 20, no. 2, pp. 100–106, Jan. 1977.