CHAPTER 1
INTRODUCTION

1. INTRODUCTION

1.1 Introduction

With the development of the economy, the number of large high buildings is increasing. Generally, for the complex application, high load of fire and intensive staff, major property damage and heavy casualties will be easily caused if fire happens in these places, and has a bad social impact. So difficult technical problems of fire detection and alarm are urgently solved to obtain more valuable time for extinguish and evacuation. In large rooms and high buildings, conventional fire detectors can hardly detect characteristic parameters of fire like smoke, temperature, vapor and flame in the very early time of fire, and cannot meet the demand of early fire detection in these places. Compared to Conventional fire detectors, video fire detectors which have many advantages, such as fast response, long distance of detection, large protection area et al, are particularly applicable to large rooms and high buildings. But most current methods for video fire detection have high rates of false alarms. Researchers all over the world have done a lot of work on this new technique. In this process fire detectors detect the fire based on the flame detector. Because a flame detector is the best detector for this process. Using this we can easily find out the fire at industries, hospitals, home applications, shopping malls etc…previously, we were using controllers and sensors.by using this hardware faces problems. But in this project only software using.in this process takes video streaming for the identification of fire. For the realization of real-time fire detection, block processing technique is adopted and the computation of texture features is done to every block of image. Here take flame detector and the fire alarm trigger is set according to the total smoke blocks in one frame.

Fire is very dangerous and brings great loss of life and properties. Yearly thousands of accidents related to fire happen all over the world due to power failure, accidental fire, natural lightning. So to control fire, various systems are being developed. The existing systems are the smoke sensor types and sprinkler type systems that detect fire from smoke and are designed to activate after reaching the threshold set temperature. Still, with this kind of system, there are many disadvantages like a false alarm, space coverage, signal transmission, and also the delay in a fire alarm. As smoke detectors are placed in the ceiling,

smokes take time to reach up to the ceiling, which results in time delay. And another problem of the existing system is that it is tough to implement in the open environment and large infrastructures like stadiums, aircraft hangers due to the vast area covered by these infrastructures. Using the image processing technology in fire detection opens many possibilities. The technology can be implemented in hazardous areas where the heat and temperature are very high, and there is always a chance of getting caught on fire. Those Places should be monitored continuously because of the ss high-risk zone with this technology. The system can give the right information about the site. The project is designed with the Raspberry Pi as the central processing unit to obtain cost-effectiveness, low power consumption, and portability. Raspberry Pi also offers many other advantages like the coupling of additional hardware, gives many opportunities to use software and networking, which helps this project to use in the future with other projects as well.

The importance of the proposed thesis is to make a reliable, safe, and smart system to reduce limitations and faults like false alarms, which cause panic among the people and even the loss of money with the use of new technology. And make the places safe from the hazardous fire.

1.2 Problem Statement

In more areas fire will occur easily. because of some chemical and by temperature increment. So, we want to detect each and every small and big fire. So, basically by using hardware it is only possible using heat detectors, smoke detectors and gas sensors. But, now this time we are not using that much hardware. This will be possible by only using a flame detector. So, we are using a background subtraction process for the subtract of the remaining area without a fire occurring area. So, this is very user application based we are taking. Collect and curate a dataset of images and videos containing fire and non-fire scenarios. Train Haar Cascade classifiers to effectively detect the distinct patterns and features associated with flames and fires. Implement a real-time video processing pipeline using OpenCV to capture and analyze video streams from cameras or recorded video files.

Develop a custom fire detection algorithm that utilizes the trained Haar Cascade classifiers to detect fire or flames within each frame of the video input.Thresholding and Alerting: Establish an appropriate threshold for fire detection confidence and ensure that false positives are minimized. Implement alerting mechanisms, such as alarms, notifications, or visual indicators, to signal the presence of a fire. Optimize the algorithm for real-time performance, considering factors like computational efficiency and accuracy in various lighting and environmental conditions.

Create a user-friendly interface that displays the video feed with real-time fire detection overlays and alerts. Provide options for users to configure and fine-tune the system.Evaluate the system's performance using a diverse dataset that includes different fire scenarios, varying lighting conditions, and challenging environments. Measure accuracy, precision, recall, and latency. Prepare the system for deployment in different settings, such as homes, offices, industrial facilities, or public spaces, by considering hardware requirements, scalability, and integration with existing security systems.

1.3 Objective

Detector. But in this process we are using new ones with new techniques. So, we can get good output. Because of background subtraction for the subtraction of all things except the fire area. So, in our output we can convert output and background subtraction with identifying the fire. For this process we are using real time application based. Basically, these are using at the commercial applications, industrial applications for the identification of the fire.

1.4 Related Work

Fire detection is used to detect the fire. In this process fire will be detected at the real time video streaming. In this technique background subtraction will be used to detect the fire. It shows the output only fire. Remaining will be subtracted from the video.

In large rooms and high buildings, conventional fire detectors can hardly detect characteristic parameters of fire like smoke, temperature, vapor and flame in the very early time of fire, and cannot meet the demand of early fire detection in

these places. Compared to conventional fire detectors, video fire detectors which have many advantages, such as fast response, long distance of detection, large protection area etc, are particularly applicable to large rooms and high buildings. But most current methods for video fire detection have high rates of false alarms. Researchers all over the world have done a lot of work on this new technique. Up to now, most methods make use of the visual features of fire flame. There are few studies on video fire detection that can detect flames. Establish a color model to recognize fire, flame and smoke. Generally speaking, most studies of video smoke detection focus on grayish smoke from the smoldering phase, while few on black smoke produced with flame. With the high growth of current market demand, video fire detection techniques that could detect flame will be applied to more scenes will certainly be the development trend in the future.

In the case of general fires, the flames usually display reddish colors. A color model could be built to recognize flames. Unfortunately, some fire-like regions in an image may have the same colors as fire, and these fire-similar areas are usually extracted as the real fire from an image. These fire aliases are generated by two cases: non-fire objects with the same colors as fire and background with illumination of fire-like light sources. In the first case, the object with reddish colors may cause a false extraction of fire-flames. The second reason for wrong fire-extraction is that the background with illumination of burning fires, solar reflections, and artificial lights has an important influence on extraction, making the process complex and unreliable.

Improvement of fire protection systems is aimed at increasing the efficiency of their work and reliability of fires detecting at an early stage of growth. The main element of fire protection systems are detectors (sensors) for control of various physical phenomena accompanying fires. One of the most frequently controlled phenomena is the occurrence of thermal radiation during combustion and smoldering of various materials. Flame detectors (electro optical devices) determine the presence of thermal radiation The improvement of flame detectors can be performed by using a modified spectral pyrometry method. This method allows us to determine the temperature in the absence of data on the emissivity of the object. This is due to the presence of combustible materials with unknown

emissivity . Feature of the spectral pyrometry method is the low response rate due to the need to register radiation in a wide spectral range with a small wavelength step . Small number of control points are proposed for use in the implementation of the high speed flame detector. The number of control points should be sufficient to detect fires with a given error in determining the temperature (no more than 10%) .

Therefore, it is required to determine the spectral ranges for detection of fires from the temperature with given error. Control points of flame detectors will be selected in the spectral ranges. The purpose of this work is to conduct theoretical research to determine the spectral ranges of fire radiation detection for flame detectors based on spectral pyrometry method. It is necessary to solve the following tasks: – to determine analytical dependencies and initial data for the study; – to determine the error in the calculation of temperature by the spectral pyrometry method for set of test fires that characterize the combustion of various materials; – to determine the spectral ranges of radiation detection for test fires.

Fire detection is essential for the safety of computer vision based fire detection(VFD). Image processing is the main principle. Detection can be done by extracting the image features such as color, motion, shape, size, and texture. Smoke is the good identifier of fire. Nowadays, CCTV is installed in many public places for survey purposes. Now we use CCTV for fire detection. But the CCTV is not as intelligent for detecting fire. Therefore, we need to train the system to identify the fire by advanced algorithms.

CHAPTER 2

REQUIREMENTS

2.REQUIREMENTS

2.1 Literature survey

1.Smoke and Fire Detection

Authors:

Saylee Gharge, Sumeet Birla, Sachin Pandey, Rishi Dargad, Rahul Pandita

Abstract:

This paper presents a system which can efficiently detect fire after the image of the area has been captured by a camera. Fire has destructive properties which cannot be tolerated in any work areas. Fire is the rapid oxidation of a material in the exothermic chemical process of combustion, releasing heat and light. The light parameter and color of flame helps in detecting fire. The system first detects smoke and then fire. When smoke is present in the area it displays a message on the Security terminal. When a fire breaks in the area under consideration, the corresponding fire region in the input video frame will be segmented which covers the fire. If the area of the flame increases in the subsequent frames then an alarm is sounded.

Keywords: Fire detection; Smoke detection; YCbCr model; Image separation

2.An image processing technique for fire detection in video images

Authors:

G. Marbach, M. Loepfe, and T. Brupbacher

Abstract:

This paper presents an image processing technique for automatic real time fire detection in video images. The underlying algorithm is based on the temporal variation of fire intensity captured by a visual image sensor. The full image sequences are analyzed to select a candidate flame region. Characteristic fire features are extracted from the candidate region and combined to determine the presence of fire or non-fire patterns. Fire alarm is triggered if the fire pattern persists over a period of time. r 2006 Elsevier Ltd. All rights reserved.

Keywords: Video fire detection; Real time image processing; Pattern recognition

3.Fire detection in video sequences using a generic color model.

Authors: T. Celik and H. Demirel

Abstract:

In this paper, a rule-based generic color model for flame pixel classification is proposed. The proposed algorithm uses YCbCr color space to separate the luminance from the chrominance more effectively than color spaces such as RGB or rgb. The performance of the proposed algorithm is tested on two sets of images, one of which contains fire, the other containing fire-like regions. The proposed method achieves up to 99% fire detection rate. The results are compared with two other methods in the literature and the proposed method is shown to have both a higher detection rate and a lower false alarm rate. Furthermore the proposed color model can be used for real-time fire detection in color video sequences, and we also present results for segmentation of fire in video using only the color model proposed in this paper.

4.Development of early tunnel fire detection algorithms using image processing.

Authors: D. Han and B. Lee

Abstract:

Fire disasters are man-made disasters, which cause ecological, social, and economical damages. To minimize these losses, early detection of fire and an autonomous response is important and helpful to disaster management systems. Therefore, in this article, we propose an early fire detection framework using fine-tuned convolutional neural networks for CCTV surveillance cameras, which can detect fire in varying indoor and outdoor environments. To ensure the autonomous response, we propose an adaptive prioritization mechanism for cameras in the surveillance system. Finally, we propose a dynamic channel selection algorithm for cameras based on cognitive radio networks, ensuring reliable data dissemination. Experimental results verify the higher accuracy of our fire detection scheme compared to state-of-the-art methods and validate the applicability of our framework for effective fire disaster management.

Keywords:Machine Learning, Image Classification, Learning Vision, Deep Learning, SurveillanceNetworks, Fire Detection, Disaster Management

5. Improving fire detection reliability by a combination of video analytics.

Authors:

R. Di Lascio, A. Greco, A. Saggese, and M. Vento

Abstract:

In this paper we propose a novel method for detecting fires in both indoor and outdoor environments. The videos acquired by traditional surveillance cameras are analyzed and different typologies of information, respectively based on color and movement, are combined into a multi expert system in order to increase the overall reliability of the approach, making its usage possible in real applications. The proposed algorithm has been tested on a very large dataset acquired in real environments and downloaded on the web. The obtained results confirm a consistent reduction in the number of false positives detected by the system, without paying in terms of accuracy.

Keywords: Fire detection,Multi expert system

6. Fire Detection in the Buildings Using Image Processing.

Authors:

Jareerat Seebamrungsat, Suphachai Praising, and Panomkhawn Riyamongkol.

Abstract:

To reduce loss of life and property from fire, an early warning is an imperative. A fire detection system based on light detection and analysis is proposed in the paper. This system uses HSV and YCbCr color models with given conditions to separate orange, yellow, and high brightness light from background and ambient light. Fire growth is analysed and calculated based on frame differences. The overall accuracy from the experiments has been greater than 90%.

7.Optimized flamedetection using image processing based techniques.

Authors:

Gaurav Yadav, Vikas Gupta, Vinod Gaur, Dr. Mahua Bhattacharya

Abstract:

Present work is an in depth study to detect flames in video by processing the data captured by an ordinary camera. Previous vision based methods were based on color difference, motion detection of flame pixel and flame edge detection. This paper focuses on optimizing the flame detection by identifying gray cycle

pixels nearby the flame, which is generated because of smoke and of spreading of fire pixel and the area spread of flame. These techniques can be used to reduce false alarms along with fire detection methods . The novel system simulates the existing fire detection techniques with above given new techniques of fire detection and gives an optimized way to detect the fire in terms of less false alarms by giving the accurate result of fire occurrence.

Keywords: Fire detection, Video processing, Edge detection, Color detection, Gray cycle pixel, Fire pixel,spreading.

8. Computer Vision Based Fire Alarming System

Authors:

A. E. Gunawardena, R. M. M. Ruwanthika, A. G. B. P. Jayasekara

Abstract:

Fire detection system in the surveillance system monitors the indoor environment and issues an alarm as part of the early warning mechanism with the ultimate goal to provide an alarm at an early stage before the fire becomes uncontrollable. Conventional fire detection systems suffer from the transparent delay from the fire to the sensor which is looking at a point. The reliability of the fire detection system mainly depends on the positional distribution of the sensors. This paper proposes a novel method of fire detection by processing image sequences acquired from a video.

9. Image Processing Based Forest Fire Detection using YCbCr Colour Model

Authors:

C. Emmy Premal, S. S. Vinsley

Abstract:

In this paper image processing based forest fire detection using YCbCr colour model is proposed. The proposed method adopts a rule based colour model due to its less complexity and effectiveness. YCbCr colour space effectively separates luminance from chrominance compared to other colour spaces like RGB and rgb(normalized RGB). The proposed method not only separates fire flame pixels but also separates high temperature fire centre pixels by taking into account statistical parameters of fire image in YCbCr colour space like mean and standard deviation. In this method four rules are formed to separate the true

fire region. Two rules are used for segmenting the fire region and other two rules are used for segmenting the high temperature fire centre region. The results obtained are compared with the other methods in the literature and show higher true fire detection rate and less false detection rate. The proposed method can be used for real time forest fire detection with moving camera.

10. Fire Detection In Different Color Models

Authors:

V. Burak Celen , M. Fatih Demirci

Abstract:

Detecting forest fire is a highly active research area in the field of pattern recognition and computer vision as a number of existing methods available in the literature. The purpose of the proposed study is to select the most suitable color space, features and classifiers for the fire classification. Our approach begins by finding the likelihood of every pixel value. The fire is then defined by multiplying pixel channel value's likelihood. By using a simple thresholding schema, the fire pixel classification process is performed. Our experimental study demonstrates which color space, features and classifiers are most suitable for fire detection.

2.2 Existing System

Existing systems include Fire and Hazard Detection systems which employ heat sensors or temperature sensors or smoke sensors or a combination of these. These are installed at heights which are usually floor level (or ceiling level). These contain individual sensors which are not lined together, which leads to unpredictability and nonsynchronous behavior of alarm. A smoke detector is a device that senses smoke, typically as an indicator of fire. Fire alarm systems known as smoke alarms, generally issue a local audible or visual alarm on detection of smoke. Generally, fire alarms consist of smoke detectors with a basic assumption that smoke will be generated by the fire. If we detect smoke, then the fire is detected. Even if there is any fire, the smoke may be generated quite later after burning the surroundings. For some fires, smoke may not be generated or it takes a long time for the smoke detectors to detect the smoke.

Disadvantages of Existing System

- Only after detection of smoke, the fire is detected.
- Even if there is any fire, the smoke may be generated quite later after burning the surroundings. For some fires, smoke may not be generated.
- It takes a long time for the smoke detectors to detect the smoke.
- Surrounding materials will be burnt till the next precautionary measure is taken.

2.3 Proposed System

We propose a system which automatically detects the presence of fire based on the algorithm described subsequently. This project proposed a fire detection algorithm which is free from sensors as the ordinary fire detection systems contain. The objective of this project was to create a system which would be able to detect fire as early as possible from a live video feed. System is expected to detect fire while it is still small and has not grown to mammoth proportions. Also, the hardware is minimal and has been already existent in places, thus saving capital. It also saves cost by getting rid of expensive temperature and heat sensors etc. Based on the results produced, the system has proven to be effective at detecting fire. This system is an amalgamation of various fire detection algorithms.

Proposed Methodology

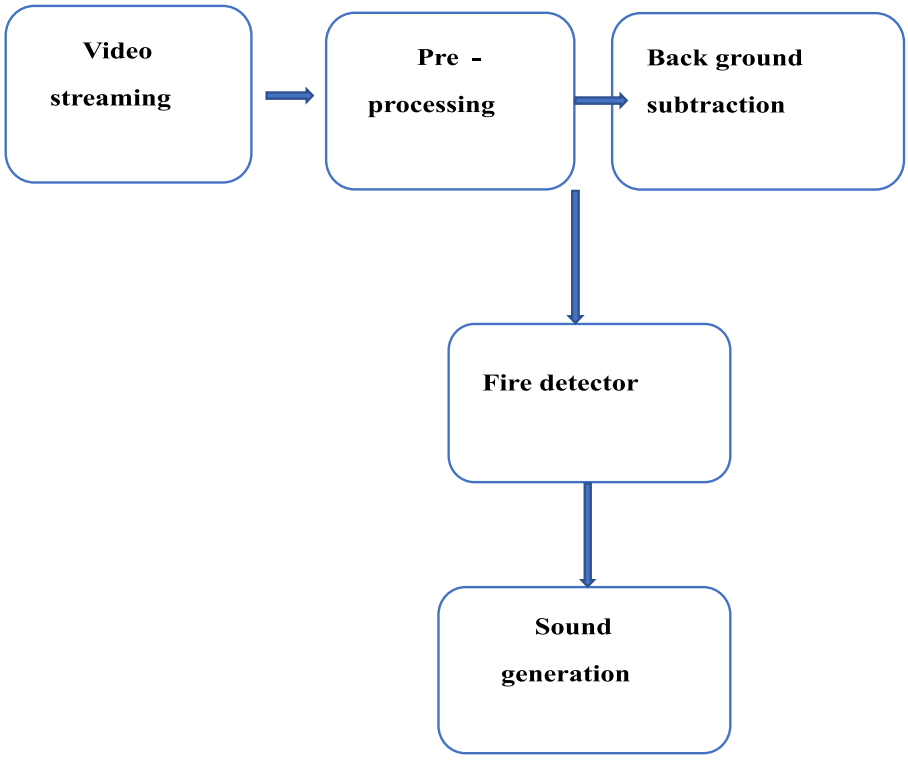

Fig 2.1 Methodology

Advantages of Proposed System

- Early detection: The system can detect fires in the early stages, allowing for quicker response and minimizing potential damage
- Cost effective:Early detection can prevent large-scale fires, which may lead to significant cost savings in terms of damage repair and recovery
- Reduced property damage: Early detection can limit the spread of fire, potentially saving property and assets

2.4 Feasibility Study

Briefly explain the project idea and its importance. Describe the goal of fire detection using Haar cascade files and OpenCV.

A.Technical feasibility:

Evaluate the availability of required hardware (computers, cameras, etc.). Assess the availability of necessary software (Python, OpenCV, libraries). Research the capabilities and limitations of Haar cascade files for fire detection. Determine if OpenCV provides the necessary tools for image processing and analysis.

Methodology: Describe the proposed approach for fire detection using Haar cascades and OpenCV. Explain how Haar cascades work for object detection. Discuss potential challenges in implementing the methodology.

Data Collection: Define the types of images or videos required for training and testing. Discuss potential sources for obtaining fire-related images or videos.

B. Economic Feasibility:

Estimate the costs associated with hardware, software, and any additional resources. Analyze the potential cost savings from using automated fire detection.

C. Operational Feasibility:

Discuss the ease of integrating the proposed solution into existing fire detection systems. Address any potential operational challenges and their mitigation.

Scheduling: Outline a timeline for development, testing, and deployment. Identify key milestones and deliverables.

Risk Assessment: Identify potential risks such as false positives/negatives, performance issues, etc.

Conclusion: Summarize the findings of the feasibility study. Recommend whether to proceed with the project based on the evaluation. Remember, this is a

high-level outline, and you'll need to go into more detail for each section. The feasibility study will serve as a guide to determine whether the project is viable and worth pursuing.

2.5 Hardware and Software Requirements

Hardware Requirements:

- RAM :2 GB min
- Hard disk :500 GB min
- Processor :1.52 Ghz speed or greater

Software Requirements:

- Operating System : Windows or Linux
- Framework :Python IDLE, OpenCV
- Language :Python

CHAPTER 3

SYSTEM ANALYSIS

3.SYSTEM ANALYSIS

3.1 Software Requirement Specification

3.1.1 Functional Requirements

The system shall detect fire using the smoke detector, gas sensor detector, heat detector, and flame detector. The flame detector shall be the primary detector due to its advantages in detecting both small and large fires. The system shall be able to distinguish between different types of fires. The system shall utilize image processing techniques for fire detection. The system shall employ a HAAR Cascade Classifier trained specifically for fire detection.

1. Video Input:

The system shall support video input from various sources, such as webcams, video files, or live streams. It shall continuously capture video frames for analysis.

2.Frame Processing:

The system shall process each frame, including resizing, converting to grayscale, and applying any necessary pre-processing steps.

3. Haar Cascade Classifier:

The system shall utilize Haar Cascade files to detect fire patterns or features within each frame. It shall be capable of training custom Haar Cascade classifiers if needed.

4. Fire Detection Logic:

The system shall determine if a fire is detected based on the results of the Haar Cascade classifier. It shall consider factors such as the size, location, and persistence of fire-like patterns.

5. Alert Generation:

The system shall trigger alerts when a fire is detected, including visual alerts, audible alarms, or notifications sent via email or other messaging channels.

3.1.1.1 Modules

1. User Module:

The User Module serves as the interface between the system and the user, allowing for interaction and control over the fire detection system. It plays a crucial role in configuring, monitoring, and managing the system's functionalities. Here's what this module typically includes:

User Interface (UI): The module provides a graphical user interface (GUI) or a command-line interface (CLI) through which users can interact with the system. Users can input commands, configure settings, and receive feedback.

System Configuration: Users can configure various parameters of the fire detection system, such as the sensitivity threshold, alert preferences, or the choice of cameras or video sources to monitor.

System Status: The module provides real-time feedback on the system's status, including whether it's active, the number of detected fires, and any alarms or notifications triggered.

Alerts and Notifications: Users can specify how they want to be alerted in case of fire detection, whether through visual indicators on the UI, audible alarms, or notifications sent via email or other messaging channels.

2. Fire Detection System Module:

The Fire Detection System Module is the core of your project, responsible for detecting fires in real-time using Haar Cascade files and OpenCV. Here's what this module typically involves:

Video Input: It interfaces with video sources, such as webcams, surveillance cameras, or video files, to continuously capture video frames for analysis.

Frame Processing: This module processes each frame from the video source. It may involve converting frames to grayscale, resizing, and other pre-processing steps to improve detection accuracy.

Haar Cascade Classifier: The heart of the system, it employs the Haar Cascade Classifier, a machine learning-based object detection technique, to identify fire patterns or features within each frame. The classifier has been trained on positive and negative image samples to recognize fires.

Fire Detection Logic: Using the Haar Cascade classifier results, this module determines if a fire is detected in the current frame. It might consider factors like the size, location, and persistence of fire-like patterns.

Alert Generation: When a fire is detected, the module triggers alerts, which can include visual alerts on the user interface, sounding alarms, and notifications to the user.

Continuous Monitoring: The system continually processes video frames in real-time, allowing for the detection of fires as soon as they appear in the camera's field of view.

Performance Optimization: The module may include optimizations to enhance detection accuracy and reduce false positives, such as adjusting detection thresholds or applying post-processing techniques.

Integration: Depending on the project's requirements, the Fire Detection System Module may be designed for integration with other systems, like fire suppression systems or security systems.

These two modules work together to create a user-friendly fire detection system using Haar Cascade files and OpenCV. The User Module allows users to interact with and configure the system, while the Fire Detection System Module performs the actual fire detection using image processing techniques.

3.1.2 Non-Functional Requirements

1. Performance:

- Response Time: The system should provide real-time or near-real-time fire detection, with minimal delay.
- Scalability: The system should be able to handle varying workloads, including processing video from multiple cameras simultaneously.

2. Reliability:

- Availability: The system should have high availability to ensure continuous monitoring and detection.
- Fault Tolerance: It should gracefully handle errors or failures, such as hardware issues or camera disconnects, without crashing.

3. Accuracy:

- Detection Accuracy: The system should achieve a high level of accuracy in fire detection, minimizing false positives and false negatives.
- Robustness: It should be able to detect fires in various lighting conditions, angles, and distances.

4. Security:

- Access Control: Ensure that only authorized personnel have access to the system and its data.
- Data Privacy: Protect sensitive data, such as video feeds, from unauthorized access or breaches.

5. Maintainability:

- Code Maintainability: The codebase should be well-documented and easy to maintain and update.
- Component Modularity: The system should be designed with modular components for easier maintenance and upgrades.

6. Compatibility:

- Platform Compatibility: Ensure the system works on different platforms and operating systems where Python and OpenCV are supported.
- Camera Compatibility: It should support various types of cameras and image sources.

7. Usability:

- User Interface: If applicable, provide a user-friendly interface for configuring and monitoring the system.
- Ease of Integration: Make it easy for users to integrate the fire detection system into existing setups

8. Resource Utilization:

- Resource Efficiency: Optimize resource usage (CPU, memory, etc.) to ensure the system runs efficiently.

9. Regulatory Compliance:

- Compliance with Laws and Regulations: Ensure that the system complies with any relevant laws and safety standards for fire detection.

10. Documentation:

- User Documentation: Provide user manuals and documentation for system installation, configuration, and troubleshooting.

11. Testing and Validation:

- Testing Procedures: Define testing procedures to validate the system's performance, accuracy, and reliability.
- Validation Criteria: Establish criteria for determining when the system is ready for deployment.

12.Performance Monitoring:

- Implement logging and monitoring mechanisms to track system performance and issues in real-time.

13.Backup and Recovery:

- Ensure a reliable data backup and recovery mechanism in case of data loss or system failures.

14.Cost Constraints:

- Budget Constraints -Adhere to budget constraints and consider the cost of hardware, software, and maintenance.

These non-functional requirements are essential for ensuring the effectiveness, reliability, and maintainability of a fire detection system using Haar Cascade files and OpenCV in Python.

CHAPTER 4
SYSTEM DESIGN

4.SYSTEM DESIGN

4.1 System Architecture

System architecture is like the blueprint or plan for a complex system. It outlines how all the different parts of the system will work together to achieve a specific goal. Just like how a house has a design that determines where rooms are located and how they connect, system architecture defines how the components of a system, whether it's a computer system or something else, will be organized and interact to function effectively and efficiently. It's the big-picture view that helps ensure everything in the system fits and operates smoothly.

The system will be built using the fire detection system is an image processing-based image processing or Raspberry Pi and IoT. This system uses a Haar Classifier Cascade Algorithm. The system will be described in Figure

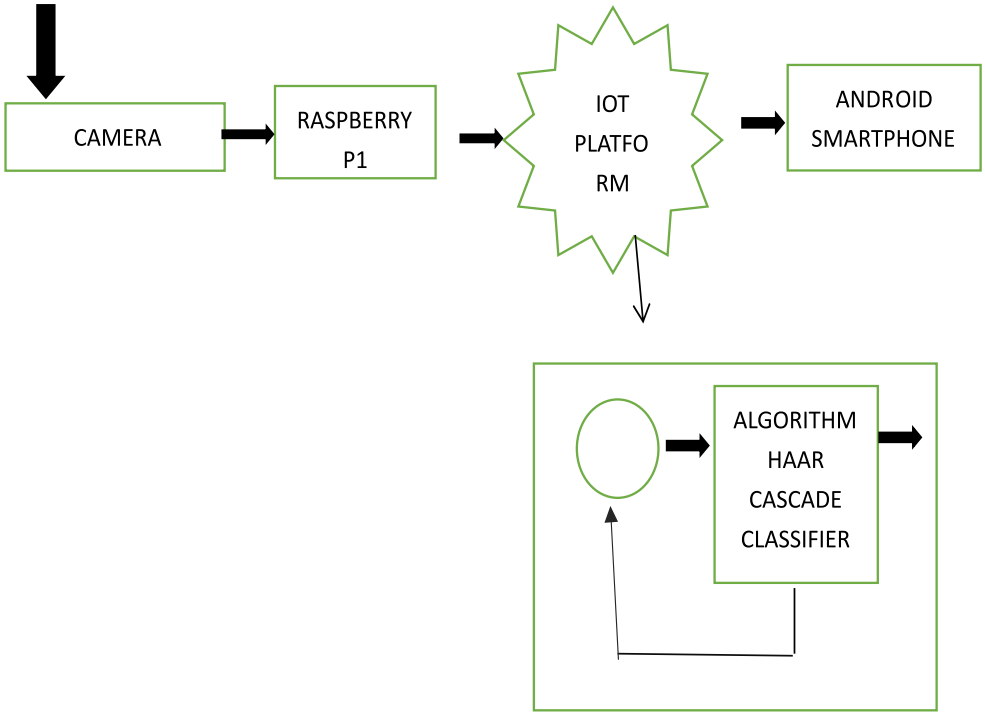

Fig 4.1 System Architecture

In this system the camera functions as a Catcher that would later become the image input (input) Raspberry Pi. Raspberry Pi used are the Raspberry Pi 3 Model B which will be the brains of the system. All the image data obtained will

be in sports in Raspberry Pi 3 Model B by using Haar Classifier Cascade algorithm. The image data that is already in the sport in Raspberry Pi 3 Model B will then be sent to the server in the form of string data in the form of warning (alert), here I use the platform server Antares is a Platform of IoT, and subsequently from the server will send data warning (alert) and display it on the user's smartphone.

4.2 Uml Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML comprises two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artefacts of software systems, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.

2. Provide extendibility and specialization mechanisms to extend the core concepts.

3. Be independent of particular programming languages and development processes.

4. Provide a formal basis for understanding the modelling language.

5. Encourage the growth of the OO tools market.

6.Support higher level development concepts such as collaborations, frameworks, patterns and components.

Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

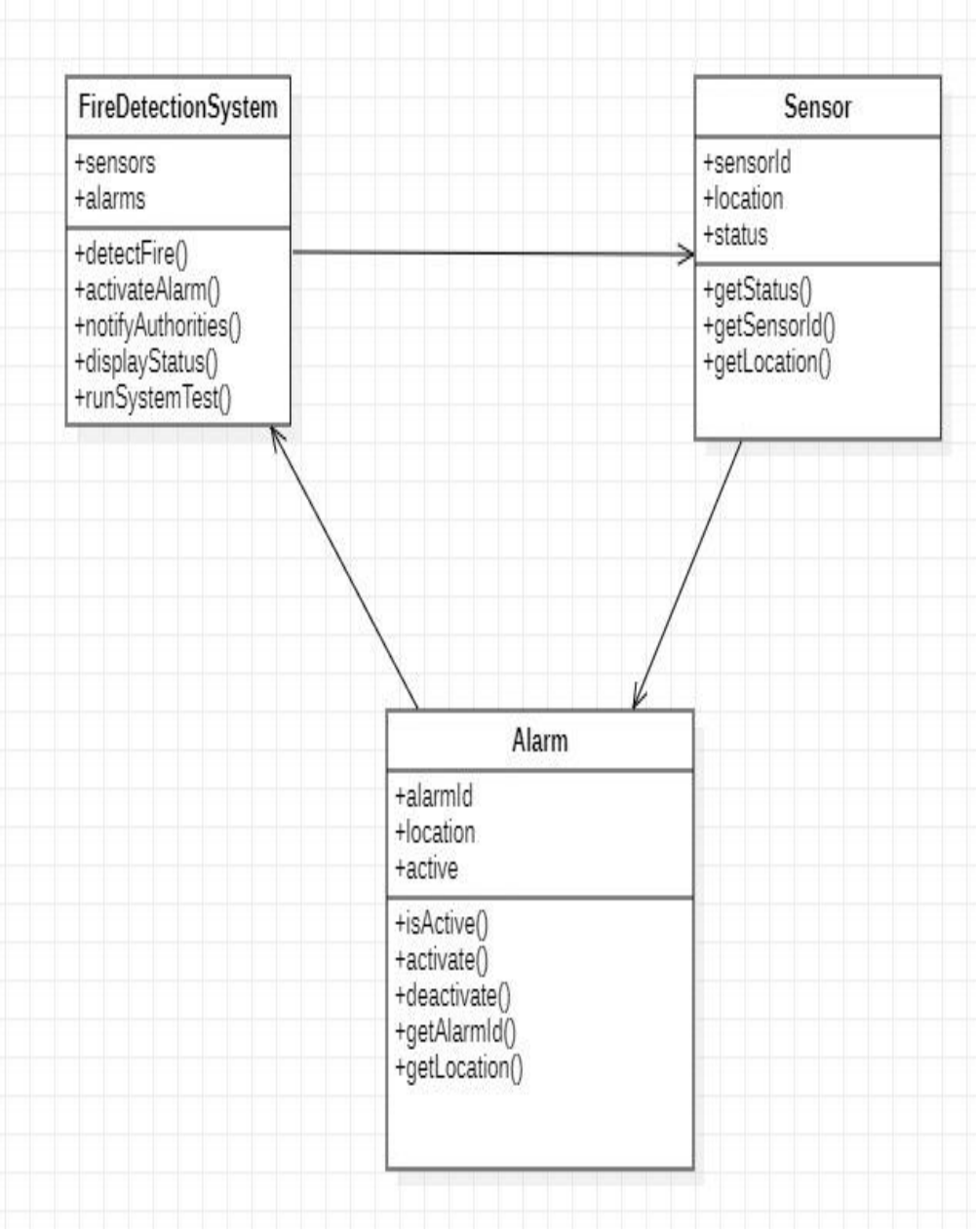

Fig 4.2 Class Diagram

Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

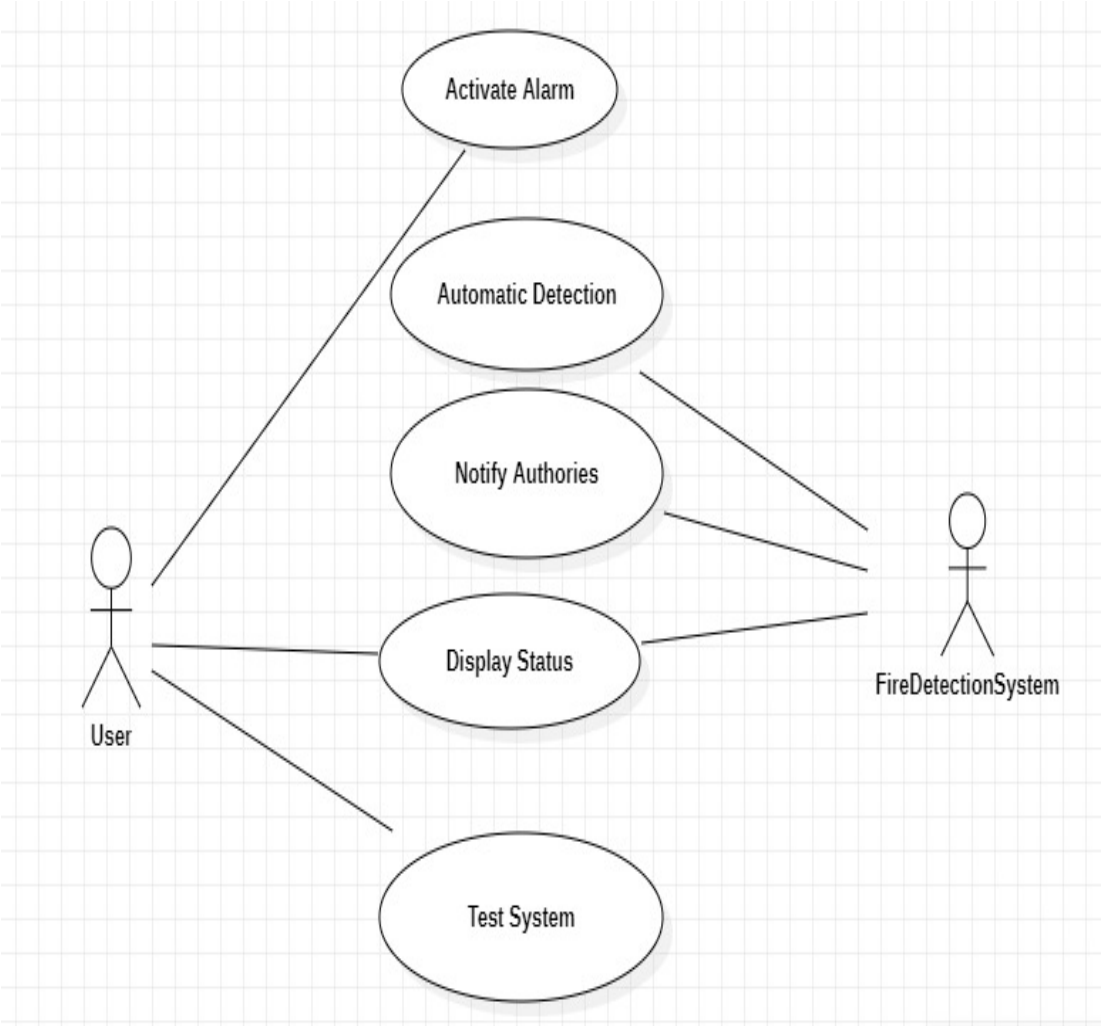

Fig 4.3 Use Case Diagram

Sequence Diagram

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

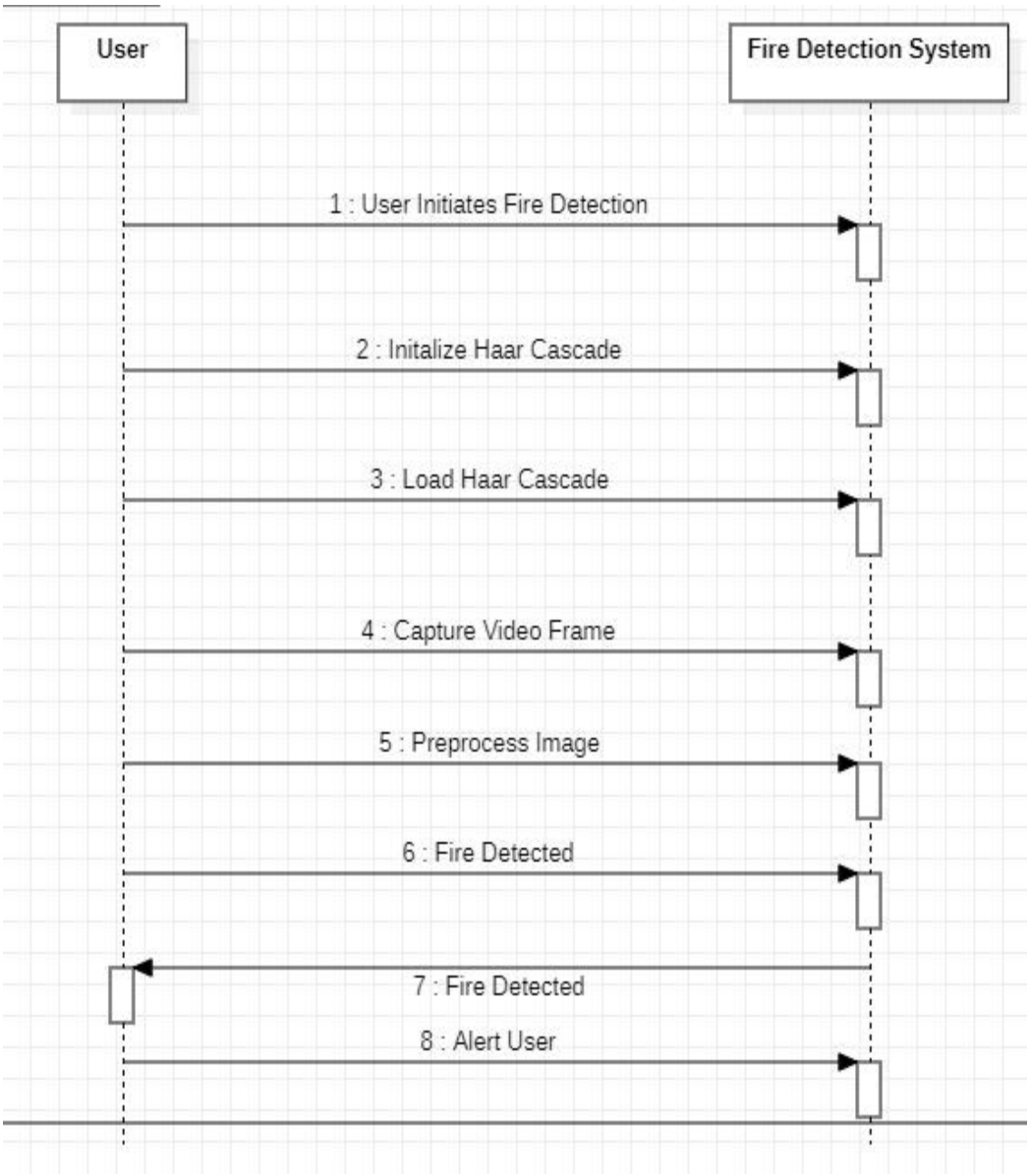

Fig 4.4 Sequence Diagram

Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

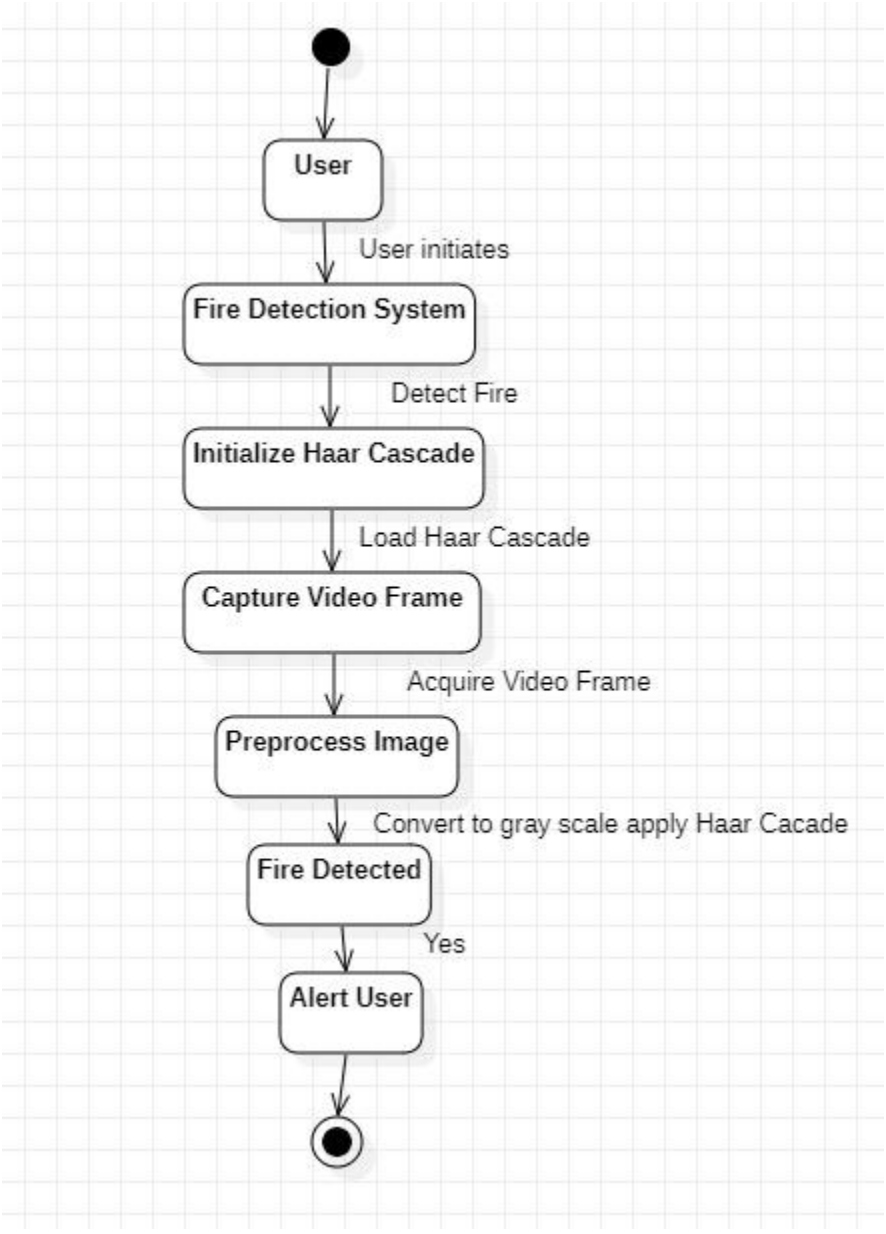

Fig 4.5 Activity Diagram

CHAPTER 5
SYSTEM IMPLEMENTATION

5.SYSTEM IMPLEMENTATION

5.1 About Technology

Python:

Python is an object-oriented, high level language, interpreted, dynamic and multipurpose programming language.Python is easy to learn yet powerful and versatile scripting language which makes it attractive for Application Development. Python's syntax and dynamic typing with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas. Python supports multiple programming patterns, including object-oriented programming, imperative and functional programming or procedural styles.

Python is not intended to work on special areas such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD etc.

We don't need to use data types to declare variables because it is dynamically typed so we can write a=10 to declare an integer value in a variable. Python makes the development and debugging fast because there is no compilation step included in python development and edit-test-debug cycle is very fast.

Python Modes

There are three different ways of working in Python:

1) Interactive Mode
2) Script Mode
3) Using IDE: (Integrated Development Environment)

1.Interactive Mode

You can enter python in the command prompt and start working with Python.

Press Enter key and the Command Prompt will appear like:

Now you can execute your Python commands.

2.Script Mode

Using Script Mode , you can write your Python code in a separate file using any editor of your Operating System.

Save it by .py extension.

Now open Command prompt and execute it by :

3.Using IDE: (Integrated Development Environment)

You can execute your Python code using a Graphical User Interface (GUI).

All you need to do is:

Click on Start button -> All Programs -> Python -> IDLE(Python GUI)

You can use both Interactive as well as Script mode in IDE.

1) Using Interactive mode:

Execute your Python code on the Python prompt and it will display the result simultaneously. 2) Using Script Mode:

i) Click on Start button -> All Programs -> Python -> IDLE(Python GUI) ii) Python Shell will be opened. Now click on File -> New Window.

A new Editor will be opened . Write your Python code here.

Run then code by clicking on Run in the Menu bar.

Run -> Run Module

OpenCV

Using software to parse the world's visual content is as big of a revolution in computing as mobile was 10 years ago, and will provide a major edge for developers and businesses to build amazing products.

Computer Vision is the process of using machines to understand and analyze imagery (both photos and videos). While these types of algorithms have been around in various forms since the 1960's, recent advances in Machine Learning, as well as leaps forward in data storage, computing capabilities, and cheap high-quality input devices, have driven major improvements in how well our software can explore this kind of content.

A.What is Computer Vision?

Computer Vision is the broad parent name for any computations involving visual content – that means images, videos, icons, and anything else with pixels involved. But within this parent idea, there are a few specific tasks that are core building blocks:

- In object classification, you train a model on a dataset of specific objects, and the model classifies new objects as belonging to one or more of your training categories.
- For object identification, your model will recognize a specific instance of an object – for example, parsing two faces in an image and tagging one as Tom Cruise and one as Katie Holmes.
- A classical application of computer vision is handwriting recognition for digitizing handwritten content (we'll explore more use cases below). Outside of just recognition, other methods of analysis include:
- Video motion analysis uses computer vision to estimate the velocity of objects in a video, or the camera itself.
- In image segmentation, algorithms partition images into multiple sets of views.
- Scene reconstruction creates a 3D model of a scene inputted through images or video (check out Selva).
- In image restoration, noise such as blurring is removed from photos using Machine Learning based filters.

Any other application that involves understanding pixels through software can safely be labeled as computer vision.

B.How Computer Vision Works ?

One of the major open questions in both Neuroscience and Machine Learning is: how exactly do our brains work, and how can we approximate that with our own algorithms? The reality is that there are very few working and comprehensive theories of brain computation; so despite the fact that Neural Nets are supposed to "mimic the way the brain works," nobody is quite sure if that's actually true. Jeff Hawkins has an entire book on this topic called On Intelligence.

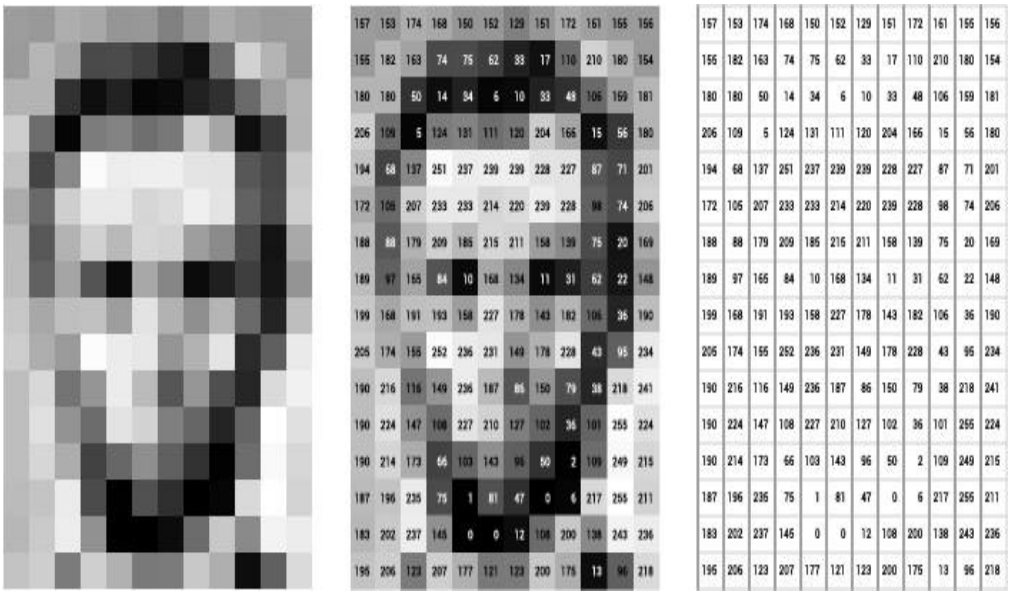

Fig 5.1 Grey Scale Value

Source: Openframeworks

Think of an image as a giant grid of different squares, or pixels (this image is a very simplified version of what looks like either Abraham Lincoln or a Dementor). Each pixel in an image can be represented by a number, usually from 0 – 255. The series of numbers on the right is what software sees when you input an image. For our image, there are 12 columns and 16 rows, which means there are 192 input values for this image.

When we start to add in color, things get more complicated. Computers usually read color as a series of 3 values – red, green, and blue (RGB) – on that same 0 – 255 scale. Now, each pixel actually has 3 values for the computer to store in addition to its position. If we were to colorize President Lincoln (or Harry Potter's worst fear), that would lead to 12 x 16 x 3 values, or 576 numbers.

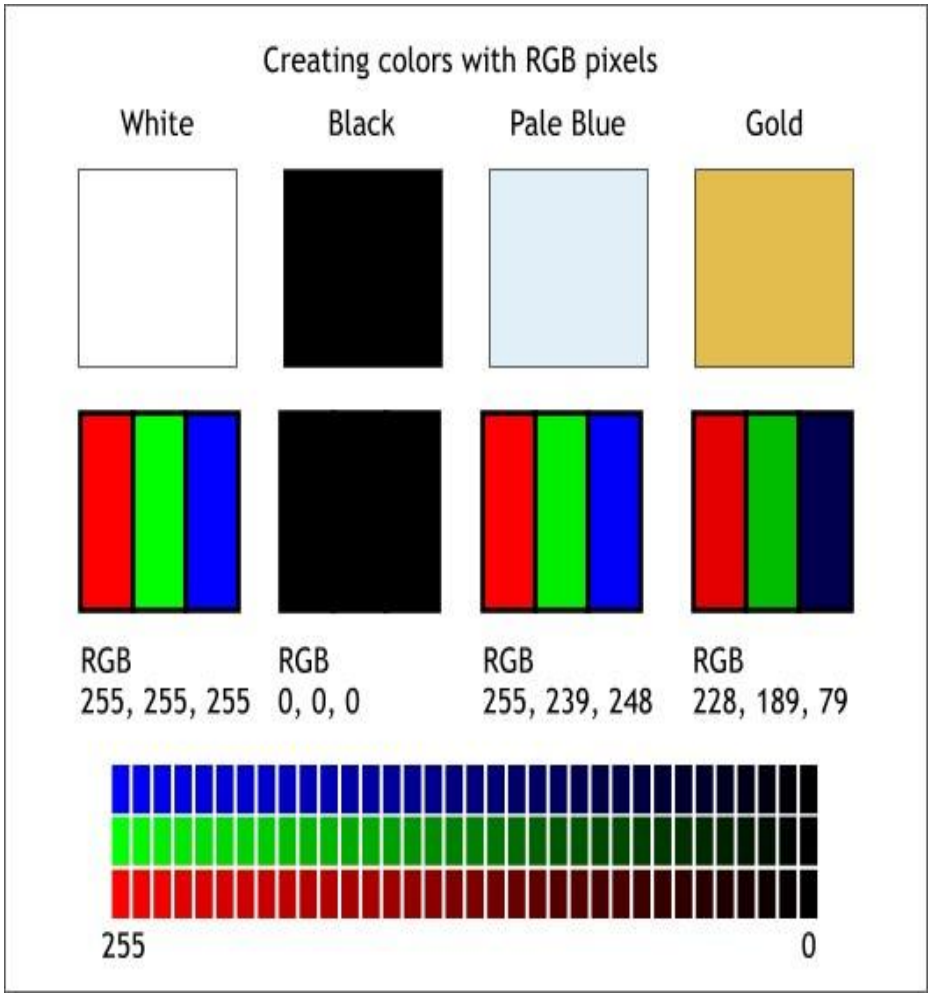

Fig 5.2 Creating colors With RGB Pixels

Digital Image Processing

Digital image processing deals with manipulation of digital images through a digital computer. It is a subfield of signals and systems but focuses particularly on images. DIP focuses on developing a computer system that is able to perform processing on an image. The input of that system is a digital image and the system processes that image using efficient algorithms and gives an image as an output. The most common example is Adobe Photoshop. It is one of the widely used applications for processing digital images. How it works In the above figure, an image has been captured by a camera and has been sent to a digital

system to remove all the other details, and just focus on the water drop by zooming it in such a way that the quality of the image remains the same.

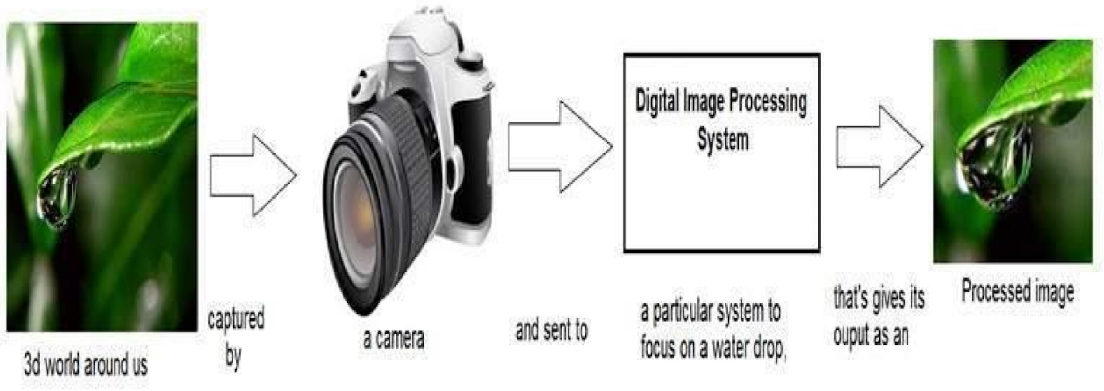

Fig 5.3 Digital Image Processing System

This tutorial gives you the knowledge of widely used methods and procedures for interpreting digital images for image enhancement and restoration and performing operations on images such as (blurring, zooming , sharpening , edge detection , etc.). It also focuses on the

understanding of how human vision works. How do the human eye visualize so many things ,

and how do the brain interpret those images? The tutorial also covers some of the important concepts of signals and systems such as (Sampling, Quantization, Convolution , Frequency domain analysis etc.).

Classification of Images

There are 3 types of images used in Digital Image Processing. They are

1. Binary Image
2. Gray Scale Image
3. Colour Image

1.Binary Image

A binary image is a digital image that has only two possible values for each pixel. Typically the two colours used for a binary image are black and white though any two colors can be used. The color used for the object(s) in the image is the foreground color while the rest of the image is the background color.

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1).This name black and white, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as grayscale images

Binary images often arise in digital image processing as masks or as the result of certain operations such as segmentation, thresholding, and dithering. Some input/output devices, such as laser printers, fax machines, and bi-level computer displays, can only handle bi-level images.

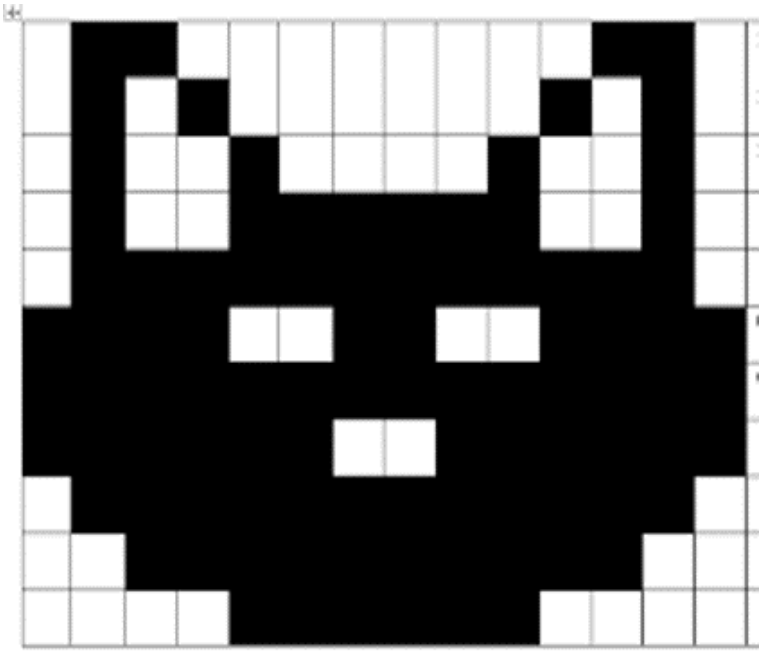

Fig 5.4 Binary Image

2.Gray Scale Image

A grayscale Image is a digital image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray(0-255), varying from black(0) at the weakest intensity to white(255) at the strongest.

Grayscale images are distinct from one-bit black-and-white images, which in the context of computer imaging are images with only the two colors, black, and white (also called bi-level or binary images). Grayscale images have many shades of gray in between. Grayscale images are also called monochromatic, denoting the absence of any chromatic variation.Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of

the electromagnetic spectrum (e.g. infrared, visible light, ultraviolet, etc.), and in such cases they are monochromatic when only a given frequency is captured. But also they can be synthesized from a full color image; see the section about converting to grayscale.

Fig 5.5 Gray Scale Image

3.Colour Image

A (digital) color image is a digital image that includes color information for each pixel. Each pixel has a particular value which determines its appearing color. This value is qualified by three numbers giving the decomposition of the color in the three primary colors Red, Green and Blue. Any color visible to the human eye can be represented this way. The decomposition of a color in the three primary colors is quantified by a number between 0 and 255. For example, white will be coded as R = 255, G = 255, B = 255; black will be known as (R,G,B) = (0,0,0); and say, bright pink will be : (255,0,255). In other words, an image is an enormous two-dimensional array of color values, pixels, each of them coded on 3 bytes, representing the three primary colors. This allows the image to contain a total of 256x256x256 = 16.8 million different colors. This technique is also known as RGB encoding, and is specifically adapted to human vision

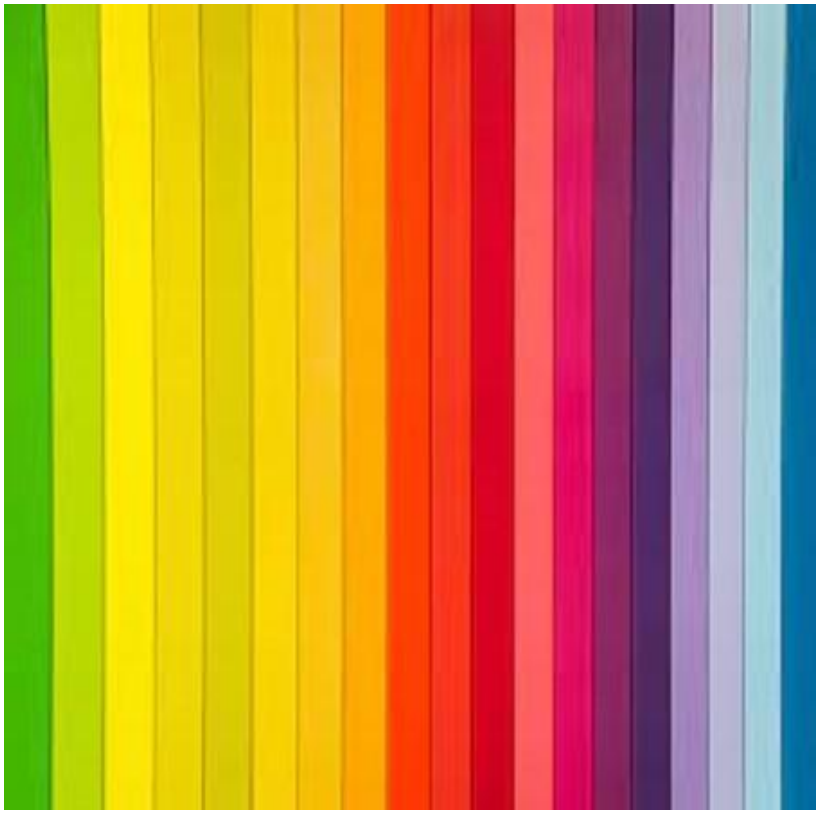

Fig 5.6 Color Image

Fire Detection

Fire detectors sense one or more of the products or phenomena resulting from fire, such as smoke, heat, infrared and/or ultraviolet light radiation, or gas.

In dwellings, smoke detectors are often stand-alone devices. In non-domestic buildings, fire detection will typically take the form of a fire alarm system, incorporating one or more of the following automatic devices:

1. Flame detector
2. Heat detector
3. Smoke detector
4. Fire gas detector

1.Flame Detector

A flame detector is a sensor designed to detect and respond to the presence of a flame or fire, allowing flame detection. Responses to a detected flame depend on the installation, but can include sounding an alarm, deactivating a fuel line (such as a propane or a natural gas line), and activating a fire suppression system. When used in applications such as industrial furnaces, their role is to provide confirmation that the furnace is working properly; in these cases they take no direct action beyond notifying the operator or control system. A flame detector can often respond faster and more accurately than a smoke or heat detector due to the mechanisms it uses to detect the flame.

2.Heat Detector

A heat detector is a fire alarm device designed to respond when the convected thermal energy of a fire increases the temperature of a heat sensitive element. The thermal mass and conductivity of the element regulate the rate flow of heat into the element. All heat detectors have this thermal lag. Heat detectors have two main classifications of operation, "rate-of-rise" and "fixed temperature". The heat detector is used to help in the reduction of property damage. It is triggered when temperature increases.

3.Smoke Detector

A smoke detector is a device that senses smoke, typically as an indicator of fire. Commercial security devices issue a signal to a fire alarm control panel as part of a fire alarm system, while household smoke detectors, also known as smoke

alarms, generally issue a local audible or visual alarm from the detector itself or from a number of detectors if there are multiple smoke detectors interlinked.

4.Fire Gas Detector

A carbon monoxide detector or CO detector is a device that detects the presence of the carbon monoxide (CO) gas to prevent carbon monoxide poisoning. In the late 1990s Underwriters Laboratories changed the definition of a single station CO detector with a sound device to carbon monoxide (CO) alarm. This applies to all CO safety alarms that meet.

Haar Cascade Classifier Algorithm

The Algorithm we use in this project is Haar Cascade Classifier, which is a method for detecting objects in an image, and the method of Haar Cascade Classifier is an object detection method developed by Viola Jones. This method is based on Haar-like features, combined with the classifier cascade. Haar-like features are features that are widely used in detection of objects, offering rapid extraction processes and are able to represent a lower resolution image. This method has been successfully applied in many object detection [8]. The classifier is usually done with training from some of the examples of drawings of simple and positive examples of negative images, which have the same size. The area is marked with 1 Classifier for rated similar to object or 0 to be assessed are not similar. After the training, the Classifier can be found all across the entire object with a region of the image. And to detect the target area more accurately, the scanning window size changed adaptively by Classifier. During the process of the International Conference on Electronics Representation and Algorithm (ICERA 2019).

1. Negative Samples: First we need to prepare the negative samples. The negative samples are taken from any images that do not contain the objects that you want to detect. These images are generated with some method and called the background samples or background images. The image that is used in the negative samples should be equal or larger than the size of the object image that you want to detect, because these negative samples are used to subsample a negative image into several image samples which have the same training window size.

2. Positive Samples: After we finished preparing the negative samples, now we need to prepare the positive samples. Positive samples are created by the opencv_createsamples command on the Raspberry Pi. The commands use the boosting process to define what the model should actually look for when trying to find the objects that you want to detect. And before that we need to prepare more than a single positive sample to make the positive samples with these commands. The bunch of images on positive samples are created to a text file that is similar to the background description file.

3. Cascade Training: The last step we need to do after we finish preparing negative and positive samples, is the training of the cascade classifiers step. This step processed the positive and negative samples that were prepared beforehand. The process of cascade training is using the command opencv_traincascade to make the file cascade.

5.2 CODING

Image Code

```
import cv2
a=cv2.imread("C:/Users/bommi/Downloads/ppp.jfif")
cv2.imshow('nani',a)
resize=cv2.resize(a,(300,500))
cv2.imshow('prabhas',resize)
cv2.imwrite('santhu.png',a)
cv2.imshow('santhu.png',a)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Video capturing code

```
import cv2
import numpy as np
cap=cv2.VideoCapture(0)
while(cap.isOpened()):
    ret,frame=cap.read()
    if ret==True:
        cv2.imshow('Frame',frame)
        if cv2.waitKey(25) & 0xFF==ord('q'):
            break
```

```
    else:
        break
cap.release()
cv2.destroyALLWindows()
```

Source code

```
#import the necessery packages
import numpy as np
import cv2
import time
import winsound
duration = 2000
freq = 440
#fire_detection.xml file & this code should be in the same
folder while running the code
fire_cascade = cv2.CascadeClassifier('fire_detection.xml')
cap = cv2.VideoCapture(0)
ret, frame1 = cap.read()
prvs = cv2.cvtColor(frame1,cv2.COLOR_BGR2GRAY)
hsv = np.zeros_like(frame1)
hsv[...,1] = 255
while 1:
    #seperating frames from the video
    ret, img = cap.read()
    #implementing optical flow algorithm
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    next = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    flow = cv2.calcOpticalFlowFarneback(prvs,next, None,
0.5, 3, 15, 3, 5, 1.2, 0)
    mag, ang = cv2.cartToPolar(flow[...,0], flow[...,1])
    hsv[...,0] = ang*180/np.pi/2
    n=hsv[...,0]
    #normalization of hsv conversion
    hsv[...,2] =
cv2.normalize(mag,None,0,255,cv2.NORM_MINMAX)
    #converting to BGR imge
    bgr = cv2.cvtColor(hsv,cv2.COLOR_HSV2BGR)
    cv2.imshow('orginial video',img)
    cv2.imshow('optical flow video',bgr)
```

```
    #implementing fire detection
    fire = fire_cascade.detectMultiScale(img, 1.2, 5)
    #background subtraction
    blur = cv2.GaussianBlur(img, (21, 21), 0)
    hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)
    lower = [18, 50, 50]
    upper = [35, 255, 255]
    lower = np.array(lower, dtype="uint8")
    upper = np.array(upper, dtype="uint8")
    mask = cv2.inRange(hsv, lower, upper)
    output = cv2.bitwise_and(img, hsv, mask=mask)
    res = cv2.bitwise_and(img,img, mask= mask)
    for (x,y,w,h) in fire:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
        print ('Fire is detected..!')
        winsound.Beep(freq, duration)
        time.sleep(0.2)
    cv2.imshow('img',img)
    cv2.imshow('back ground iamge',res)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

Cascade.xml

```
<opencv_storage>
<cascade>
<stageType>BOOST</stageType>
<featureType>HAAR</featureType>
<height>24</height>
<width>24</width>
<stageParams>
<boostType>GAB</boostType>
<minHitRate>9.9500000476837158e-01</minHitRate>
<maxFalseAlarm>5.0000000000000000e-01</maxFalseAlarm>
<weightTrimRate>9.4999998807907104e-01</weightTrimRate>
<maxDepth>1</maxDepth>
```

```
<maxWeakCount>100</maxWeakCount>
</stageParams>
<featureParams>
<maxCatCount>0</maxCatCount>
<featSize>1</featSize>
<mode>BASIC</mode>
</featureParams>
<stageNum>18</stageNum>
<stages>
<!--  stage 0  -->
<_>
<maxWeakCount>2</maxWeakCount>
<stageThreshold>-1.1354514956474304e-01</stageThreshold>
<weakClassifiers>
<_>
<internalNodes> 0 27 1.0459426790475845e-01</internalNodes>
<leafValues> -9.3814432621002197e-01
9.0361446142196655e-01</leafValues>
</_>
<_>
<internalNodes> 0 53 1.9976346194744110e-01</internalNodes>
<leafValues> -9.2785137891769409e-01
8.2459920644760132e-01</leafValues>
</_>
</weakClassifiers>
</_>
<!--  stage 1  -->
<_>
<maxWeakCount>3</maxWeakCount>
<stageThreshold>-1.3247847557067871e+00</stageThreshold>
<weakClassifiers>
<_>
<internalNodes> 0 -7 4.0762656927108765e-01</internalNodes>
<leafValues> -8.6901766061782837e-01
7.5675678253173828e-01</leafValues>
</_>
<_>
<internalNodes> 0 -6 1.3779436051845551e-01</internalNodes>
<leafValues> -8.8750886917114258e-01
2.8855907917022705e-01</leafValues>
</_>
```

<_>
<internalNodes> 0 40 2.6526537537574768e-01</internalNodes>
<leafValues> -7.4432617425918579e-01
6.6626292467117310e-01</leafValues>
</_>
</weakClassifiers>
</_>
<!-- stage 2 -->
<_>
<maxWeakCount>2</maxWeakCount>
<stageThreshold>-7.4336928129196167e-01</stageThreshold>
<weakClassifiers>
<_>
<internalNodes> 0 -2 4.1499978303909302e-01</internalNodes>
<leafValues> -8.6945170164108276e-01
5.0000000000000000e-01</leafValues>
</_>
<_>
<internalNodes> 0 11 4.5874185860157013e-02</internalNodes>
<leafValues> -1. 1.2608239054679871e-01</leafValues>
</_>
</weakClassifiers>
</_>
<!-- stage 3 -->
<_>
<maxWeakCount>4</maxWeakCount>
<stageThreshold>-1.0817888975143433e+00</stageThreshold>
<weakClassifiers>
<_>
<internalNodes> 0 13 4.9460947513580322e-01</internalNodes>
<leafValues> -8.1265825033187866e-01
4.2105263471603394e-01</leafValues>
</_>
<_>
<internalNodes> 030 -1.0531554371118546e-01</internalNodes>
<leafValues> 2.6704105734825134e-01
-8.5486882925033569e-01</leafValues>
</_>
<_>
<internalNodes> 0 41 3.4948602318763733e-02</internalNodes>

<leafValues> -8.9780002832412720e-01
2.7469578385353088e-01</leafValues>
</_>
<_>
<internalNodes> 081 -1.5509229342569597e-05</internalNodes>
<leafValues> 3.6162829399108887e-01
-7.1705079078674316e-01</leafValues>
</_>
</weakClassifiers>
</_>
<!-- stage 4 -->
<_>
<maxWeakCount>4</maxWeakCount>
<stageThreshold>-1.1866767406463623e+00</stageThreshold>
<weakClassifiers>
<_>
<internalNodes> 0 12 3.6403301358222961e-01</internalNodes>
<leafValues> -8.5209006071090698e-01
-1.5000000596046448e-01</leafValues>
</_>
<_>
<internalNodes> 0 14 3.3683568239212036e-02</internalNodes>
<leafValues> -5.1311254501342773e-01
4.6315157413482666e-01</leafValues>
</_>
<_>
<internalNodes> 0 69 1.3609335292130709e-04</internalNodes>
<leafValues> 3.1104850769042969e-01
-7.2132861614227295e-01</leafValues>
</_>
<_>
<internalNodes> 0 -4 1.2148362293373793e-04</internalNodes>
<leafValues> -8.3461266756057739e-01
2.4358172714710236e-01</leafValues>
</_>
</weakClassifiers>
</_>
<!-- stage 5 -->
<_>
<maxWeakCount>4</maxWeakCount>
<stageThreshold>-1.3308618068695068e+00</stageThreshold>

```
<weakClassifiers>
<_>
<internalNodes> 0   6.7514598369598389e-02</internalNodes>
<leafValues> -7.7188330888748169e-01
2.1276595070958138e-02</leafValues>
</_>
<_>
<internalNodes> 0  8 3.4416305425111204e-05</internalNodes>
<leafValues> -8.5286557674407959e-01
1.4937944710254669e-01</leafValues>
</_>
<_>
<internalNodes> 0 -1.8346133583690971e-04</internalNodes>
<leafValues> -9.0394043922424316e-01
1.9481389224529266e-01</leafValues>
</_>
<_>
<internalNodes> 0   8.9603747054934500e-04</internalNodes>
<leafValues> -1. 1.9558252394199371e-01</leafValues>
</_>
</_>
</weakClassifiers>
</_>
</stages>
<features>
<_>
<rects>
<_> 0 0 1 2 -1.</_>
<_> 0 1 1 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 0 0 2 2 -1.</_>
<_> 0 1 2 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
```

<_> 0 0 24 18 -1.</_>
<_> 8 0 8 18 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 0 0 23 6 -1.</_>
<_> 0 2 23 2 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 0 1 2 4 -1.</_>
<_> 0 3 2 2 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 0 1 8 2 -1.</_>
<_> 0 1 4 1 2.</_>
<_> 4 2 4 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 0 1 21 17 -1.</_>
<_> 7 1 7 17 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 0 1 24 17 -1.</_>
<_> 8 1 8 17 3.</_>
</rects>
<tilted>0</tilted>
</_>

<_>
<rects>
<_> 0 3 4 1 -1.</_>
<_> 2 3 2 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 0 3 12 2 -1.</_>
<_> 0 4 12 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 0 4 2 6 -1.</_>
<_> 0 4 1 3 2.</_>
<_> 1 7 1 3 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
rects>
<_> 10 21 10 2 -1.</_>
<_> 10 21 5 1 2.</_>
<_> 15 22 5 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 10 22 4 2 -1.</_>
<_> 10 22 2 1 2.</_>
<_> 12 23 2 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 11 3 3 7 -1.</_>

<_> 12 3 1 7 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 12 1 3 8 -1.</_>
<_> 13 1 1 8 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 12 5 3 4 -1.</_>
<_> 13 5 1 4 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 12 12 3 2 -1.</_>
<_> 12 13 3 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 13 7 1 4 -1.</_>
<_> 13 9 1 2 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 13 7 2 4 -1.</_>
<_> 14 7 1 4 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>

<_> 13 8 2 5 -1.</_>
<_> 14 8 1 5 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 13 9 1 12 -1.</_>
<_> 13 15 1 6 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 13 11 2 2 -1.</_>
<_> 13 12 2 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 13 14 6 8 -1.</_>
<_> 13 14 3 4 2.</_>
<_> 16 18 3 4 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 13 15 4 8 -1.</_>
<_> 13 15 2 4 2.</_>
<_> 15 19 2 4 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 14 0 3 3 -1.</_>
<_> 15 0 1 3 3.</_>
</rects>
<tilted>0</tilted>

</_>
<_>
<rects>
<_> 15 0 1 3 -1.</_>
<_> 15 1 1 1 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 15 9 2 2 -1.</_>
<_> 16 9 1 2 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 16 0 3 24 -1.</_>
<_> 16 8 3 8 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 17 4 6 3 -1.</_>
<_> 19 4 2 3 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 18 5 2 1 -1.</_>
<_> 19 5 1 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 18 19 3 2 -1.</_>
<_> 18 20 3 1 2.</_>
</rects>

<tilted>0</tilted>
</_>
<_>
<rects>
<_> 19 2 3 3 -1.</_>
<_> 19 3 3 1 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 19 4 1 2 -1.</_>
<_> 19 5 1 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 19 4 1 3 -1.</_>
<_> 19 5 1 1 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 19 7 3 1 -1.</_>
<_> 20 7 1 1 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 20 4 2 2 -1.</_>
<_> 21 4 1 2 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 20 14 4 3 -1.</_>
<_> 22 14 2 3 2.</_>

```
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 21 0 3 1 -1.</_>
<_> 22 0 1 1 3.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 21 3 2 2 -1.</_>
<_> 21 3 1 1 2.</_>
<_> 22 4 1 1 2.</_>
</rects>
<tilted>0</tilted>
</_>
<_>
<rects>
<_> 23 0 1 18 -1.</_>
<_> 23 9 1 9 2.</_>
</rects>
<tilted>0</tilted>
</_>
</features>
</cascade>
</opencv_storage>
```

CHAPTER 6

OUTPUT SCREEN

6.OUTPUT SCREENS

RUN PAGE

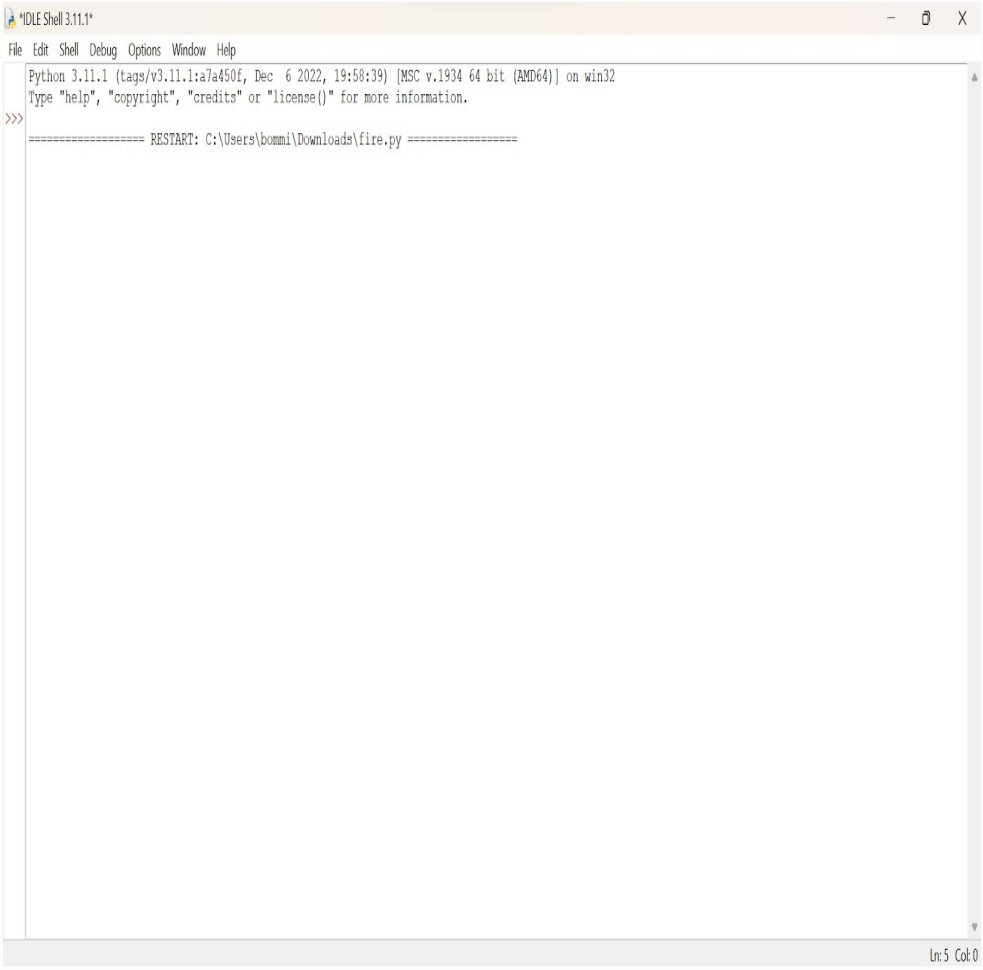

Fig 6.1 Run Page

ORIGINAL VIDEO

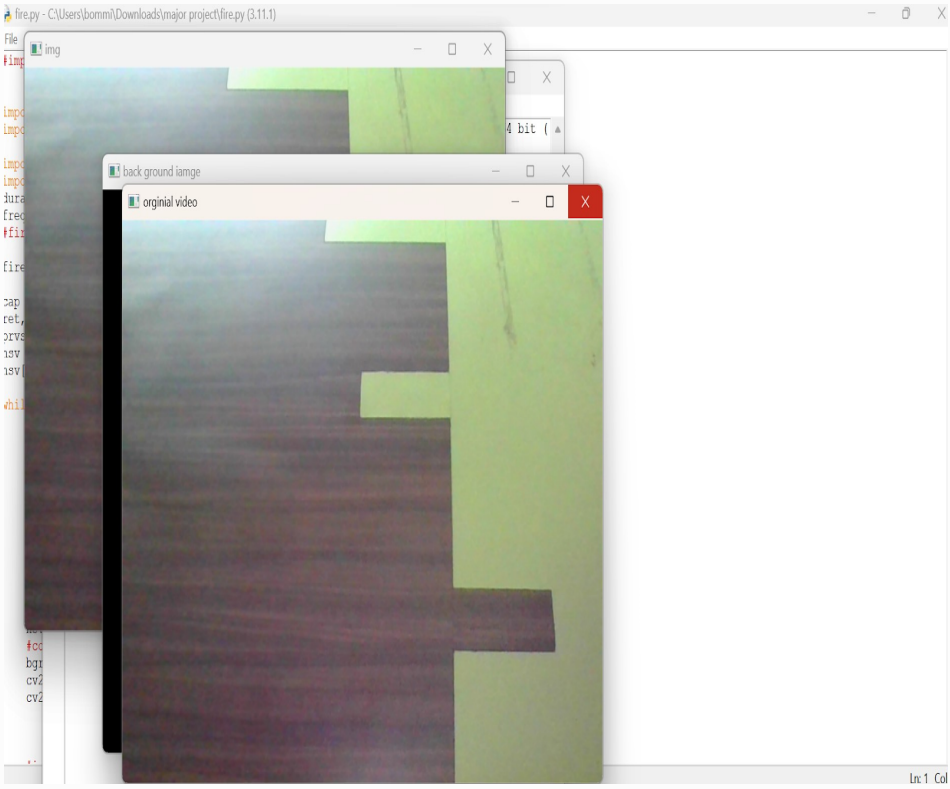

Fig 6.2 Original video

OPTICAL FLOW VIDEO

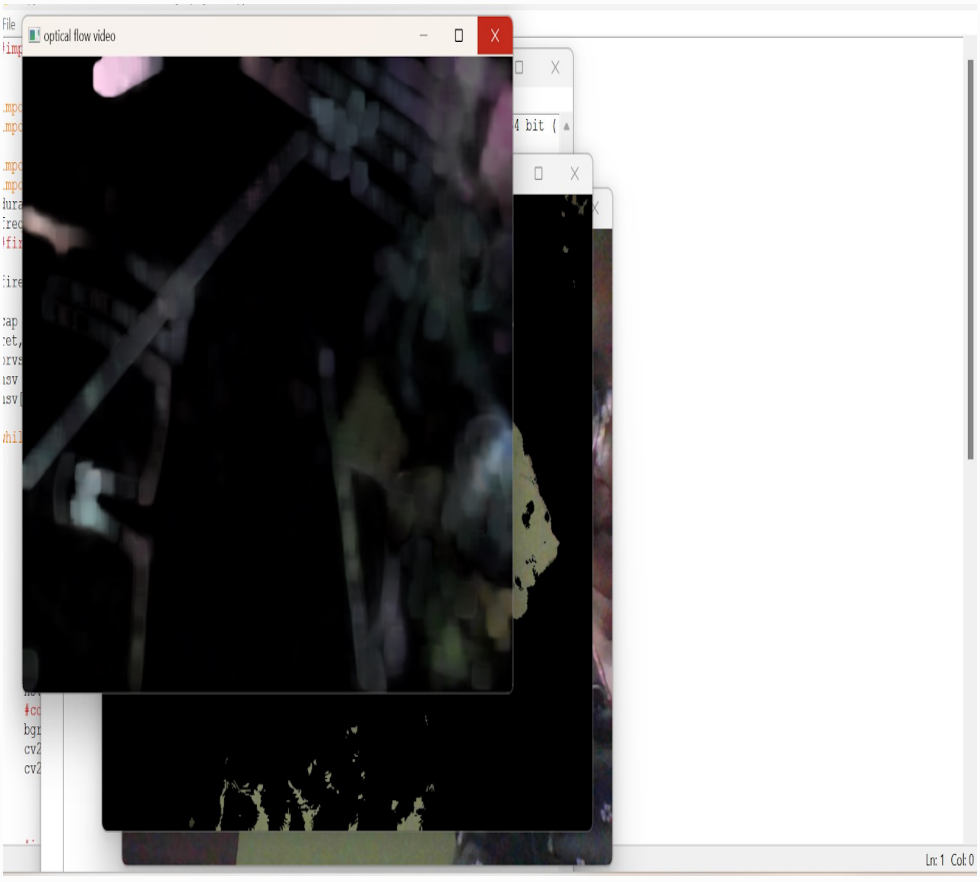

Fig 6.3 Optical Flow Video

IMAGE

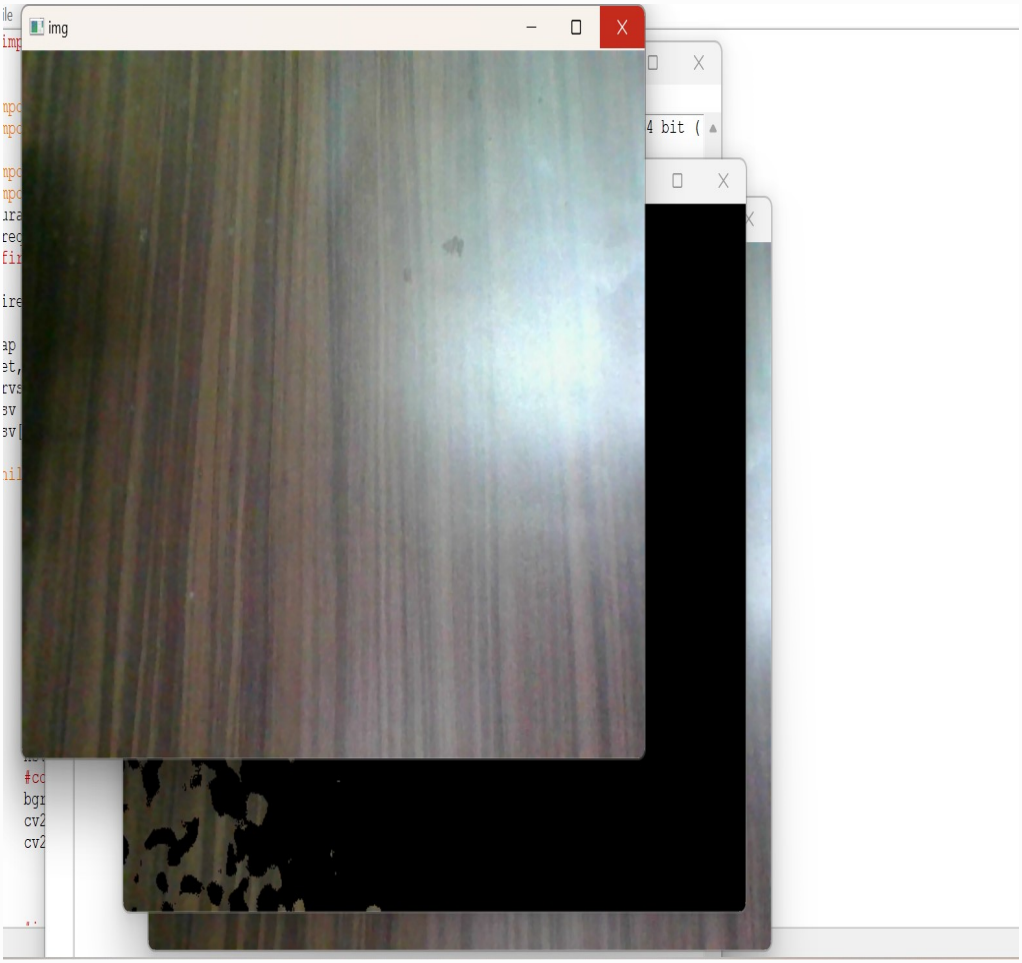

Fig 6.4 Img

BACKGROUND IMAGE

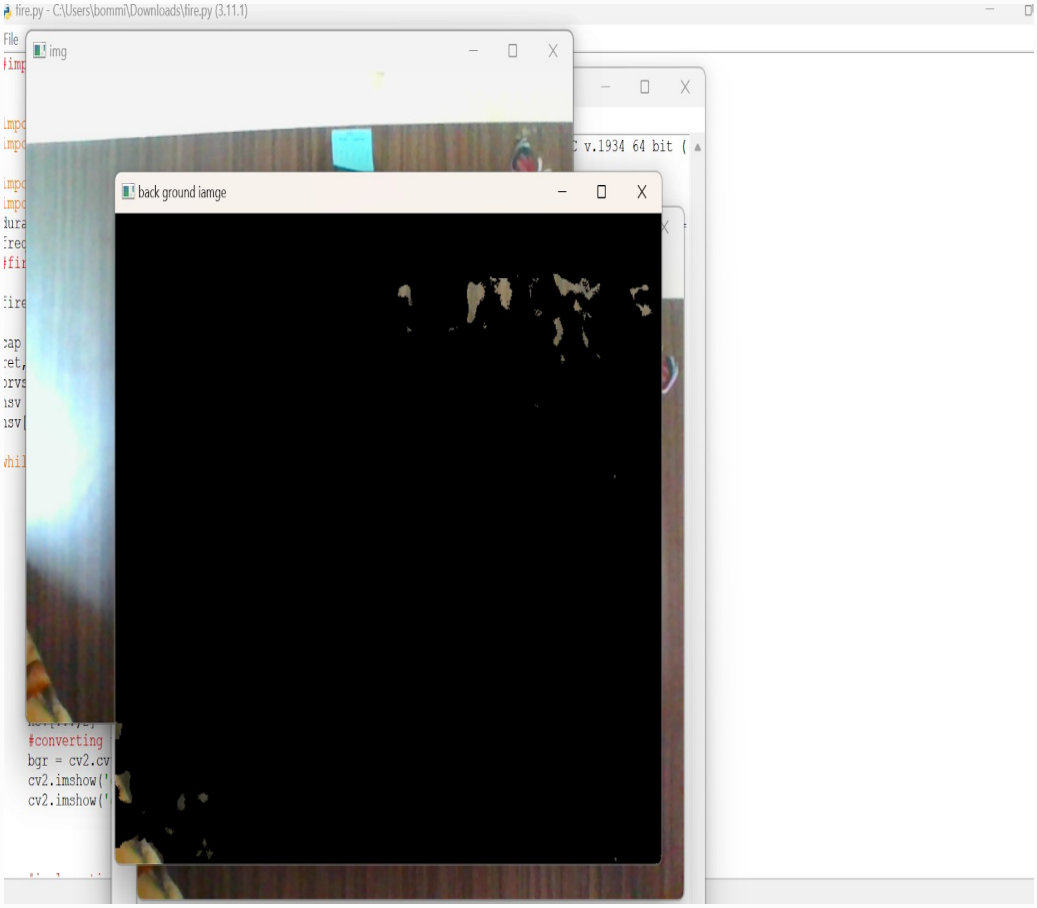

Fig 6.5 Background Image

FIRE DETECTION

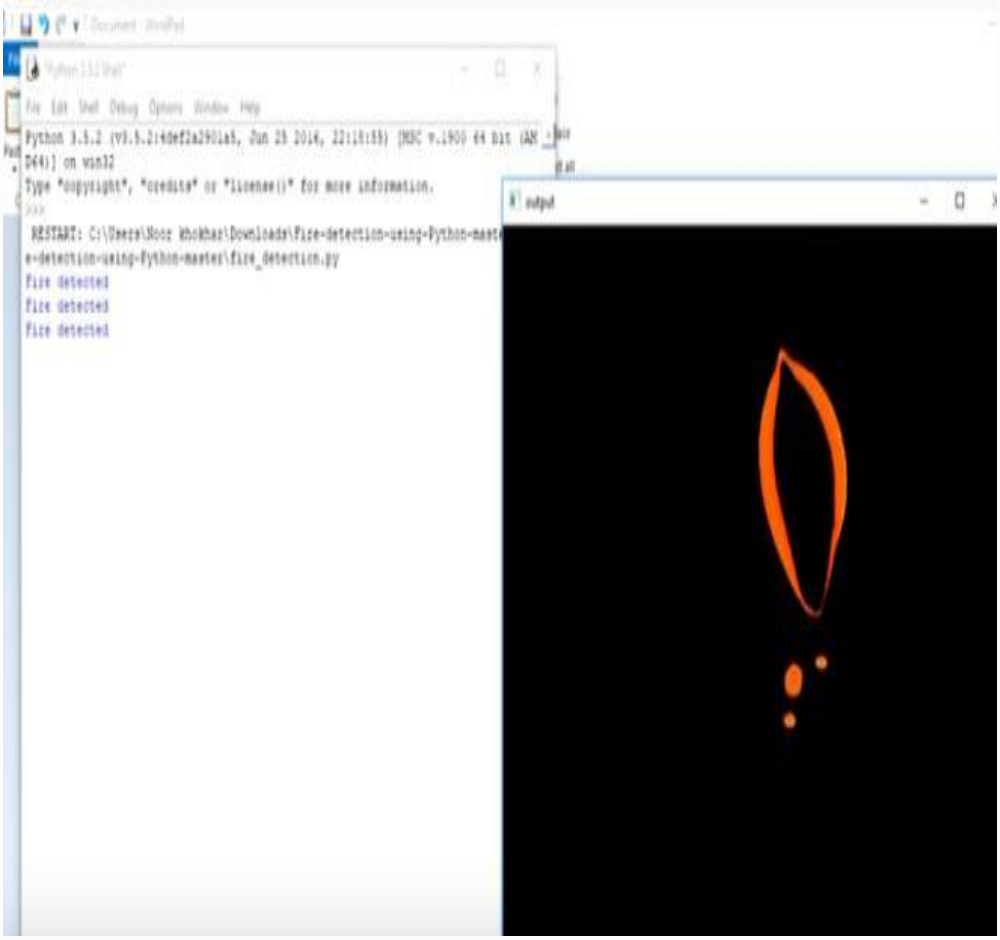

Fig 6.6 Fire Detection Image

FINAL OUTPUT

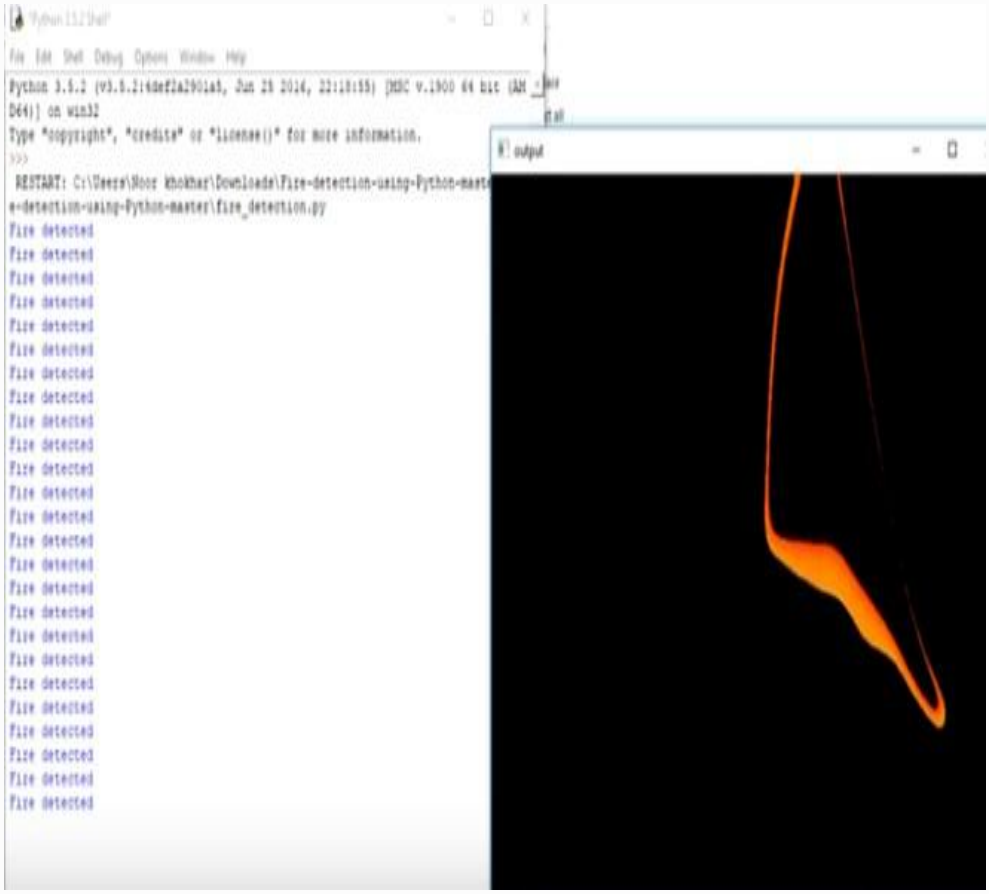

Fig 6.7 Final Output Image

CHAPTER 7
SYSTEM TESTING

7.SYSTEM TESTING

7.1Testing

Types of testing

1. Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested:

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

2. Integration Testing: Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

3.Acceptance Testing: User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

4. Performance Testing: Performance testing is executed to determine how fast a system or subsystem performs under a particular workload. It can also serve to validate and verify other quality attributes of the system such as scalability, reliability and resource usage.

5. Load Testing: Load testing is primarily concerned with testing that can continue to operate under specific load, whether that is large quantities of data or a large number of users.

6. Manual Testing: Manual Testing is the process of manually testing software for defects. Functionality of this application is manually tested to ensure the correctness. Few examples of test cases for Manual Testing are discussed later in this chapter.

7. Boundary Testing: Check how the system performs when pushed to its limits, such as processing high-resolution videos or handling a large number of concurrent fire detection tasks.

8. User Interface Testing: If the system has a user interface, test its functionality and user-friendliness.

9. CompatibilityTesting: Verify that the system works across different Python and OpenCV versions, as well as on various operating systems.

10. Security Testing: Assess the system for potential security vulnerabilities, especially if it's connected to a network or the internet.

11. Regression Testing: After any updates or modifications to the system, conduct regression testing to ensure that existing functionality remains intact.

12. Usability Testing: If the system is intended for non-technical users, involve them in usability testing to gather feedback on its ease of use.

13. End-to-End Testing: Finally, conduct end-to-end testing to ensure that the entire fire detection system functions as expected from data input to fire detection to alarm/notification.

14. Scalability Testing: Test the system's ability to scale when additional resources or cameras are added.

7.2 Test Cases

Test Case 1

Test Cases	
Description	Verify Image Input
Steps	Input an image without fire
Expected Results	Check if the system correctly identifies no fire
Status	Pass

Fig 7.1 Test Case for Input an image without fire

Test Case 2

Test Cases	
Description	Verify Image Input
Steps	Input an image with fire
Expected Result	Check if the system correctly identifies fire
Status	Pass

Fig 7.2 Test Case for Input an image with fire

Test Case 3

Test Cases	
Description	Verify Video Input
Steps	Input a video without fire
Expected Result	Check if the system correctly identifies no fire throughout the video
Status	Pass

Fig 7.3 Test Case for Input a video without fire

Test Case 4

Test Cases	
Description	Verify Video Input
Steps	Input a video with fire
Expected Result	Check if the system correctly identifies no fire throughout the video
Status	Fail

Fig 7.4 Test Case for Input a video with fire

Test Case 5

Test Cases	
Description	Verify Real-time Detection
Steps	Test real-time fire detection using a webcame
Expected Result	Ensure the system can detect fire in real-time
Status	Fail

Fig 7.5 Test Case for Test real-time fire detection using a webcam

Test Case 6

Test Cases	
Description	Check False Positives
Steps	Input an image of a non-fire object
Expected Result	Verify that the system does not report false positives
Status	Fail

Fig 7.6 Test Case for Input an image of a non-fire object

Test Case 7

Test Cases	
Description	Check Robustness
Steps	Introduce noise or low lighting conditions in the image
Expected Result	Verify if the system can still detect fire
Status	Fail

Fig 7.7 Test Case for Check Robustness

Test Case 8

Test Cases	
Description	Check Multiple Fires
Steps	Input an image or video with multiple fire instances
Expected Result	Ensure the system can detect all instances
Status	Pass

Fig 7.8 Test Case for Check Multiple Fires

Test Case 9

Test Cases	
Description	Verify Haar Cascade Files
Steps	Remove or corrupt Haar Cascade files
Expected Result	Check if the system can handle missing/corrupted files gracefully
Status	Pass

Fig 7.9 Test Case for Verify Haar Cascade Files

Test Case 10

Test Cases	
Description	Verify Scalability
Steps	Increase the size of the image or video
Expected Result	Check if the system can handle larger inputs
Status	Pass

Fig 7.10 Test Case for Verify Scalability

CHAPTER 8

CONCLUSION

8.CONCLUSION

8.1 Conclusion

The vision based fire detection system captures the images through a digital camera and analyses each frame for the presence of fire pixels based on flame colour, area, and motion. and texture analysis.by using this we can easily detect small or bog fires in our areas. Here for that purpose a flame detector will be used. And background subtraction will be used to subtract all areas without a fire occurring area. Then generate the alarm. Using this we can easily find out the fire. Before these we used gas sensors and other these are mainly used at home applications, industrial applications, medical applications and other.

8.2 Future Scope

Fire detection is possible in the terms of heat detector smoke detector flame detector and fire gas detector. Heats smoke fire gas sensors have disagvantages.by reducing disadvantages maximum we are using flame detection.so, we can get clear output. We have to develop a new detector for the purpose of fire detection before it occurs at any place because fire detection will be used in industries, medical applications, agriculture applications ,aeroplanes. before fire will occur in industries and other places we can detect by taking alarms for the future purpose.and we have to arrange other devices. Suppose fire will occur ,we want to send messages through the internet or without internet by using some other things.This will develop for the future purpose.to stop fire sometimes we are using fire extinguishers, chemical powders and other. That are taken based on our requirements

1.Enhanced Accuracy with Deep Learning: Integrate deep learning techniques, such as convolutional neural networks (CNNs), to further improve the accuracy of fire detection. Training models with large datasets can enhance the system's ability to distinguish between different fire scenarios, reducing false alarms.

2. Multi-Object Detection: Extend the system's capabilities to detect multiple fire sources and various fire-related objects simultaneously. This could be valuable in scenarios where multiple fires or hazards need to be identified in a single frame.

3. Environmental Sensing: Combine image processing with environmental sensors (e.g., temperature, humidity, gas sensors) to create a comprehensive fire detection system. This could provide early warning by detecting changes in environmental conditions that precede fires.

4. Edge Computing: Optimize the algorithm to run efficiently on edge computing devices, enabling real-time fire detection in resource-constrained environments, such as IoT devices and drones. This can facilitate rapid deployment in remote locations.

5. Integration with IoT: Integrate the fire detection system with the Internet of Things (IoT) platforms to enable remote monitoring, alerting, and control. IoT-enabled devices can transmit data and alerts to central monitoring systems or mobile applications.

6. Cloud Integration: Develop cloud-based solutions that allow centralized monitoring and management of fire detection systems across multiple locations. This can be especially useful for large-scale deployments in industrial settings.

7. Mobile Applications: Create mobile applications that connect to the fire detection system, providing users with real-time alerts and the ability to view video feeds from cameras equipped with fire detection capabilities.

8. Human Detection: Expand the capabilities of the system to not only detect fires but also identify human presence within the monitored area. This can improve safety by ensuring timely evacuation in case of a fire.

9. Machine Learning for False Alarm Reduction: Implement machine learning algorithms to analyze historical data and reduce false alarms by distinguishing between real fire incidents and non-fire events, such as cooking smoke or moving light sources.

10. Cross-Domain Applications: Explore the use of fire detection technology in new domains, such as wildlife conservation (early forest fire detection), smart cities (fire prevention and urban planning), and disaster management (rapid response to fire-related emergencies)

11. Regulatory Compliance: Ensure that the system meets industry-specific safety standards and regulations, such as those applicable to industrial facilities, healthcare settings, and residential buildings.

12. International Expansion: Consider adapting the system for international markets, where fire detection needs and environmental conditions may vary, and where different languages and regulations may apply.

13. Energy Efficiency: Focus on optimizing the system's power consumption, especially for battery-operated devices, to extend operational life and reduce environmental impact.

14. Collaboration and Research: Collaborate with research institutions and industry partners to continually advance the field of fire detection technology, share knowledge, and stay updated with the latest developments.

BIBLIOGRAPHY

[1] D. Han and B. Lee, "Development of early tunnel fire detection algorithm using image processing," in International Symposium on Visual Computing, 2006, pp. 39-48.

[2] T.-H. Chen, P.-H. Wu, and Y.-C. Chiou, "An early fire-detection method based on image processing," in Image Processing, 2004. ICIP'04. 2004 International Conference on, 2004, pp. 1707-1710.

[3] G. Marbach, M. Loepfe, and T. Brupbacher, "An image processing technique for fire detection in video images," Fire safety journal, vol. 41, pp. 285-289, 2006.

[4] T. Celik and H. Demirel, "Fire detection in video sequences using a generic color model," Fire Safety Journal, vol. 44, pp. 147-158, 2009.

[5] A. Rafiee, R. Dianat, M. Jamshidi, R. Tavakoli, and S. Abbaspour, "Fire and smoke detection using wavelet analysis and disorder characteristics," in Computer Research and Development (ICCRD), 2011 3rd International Conference on, 2011, pp. 262-265.

[6] Y. H. Habiboğlu, O. Günay, and A. E. Çetin, "Covariance matrix based fire and flame detection method in video," Machine Vision and Applications, vol. 23, pp. 1103-1113, 2012.

[7] R. Di Lascio, A. Greco, A. Saggese, and M. Vento, "Improving fire detection reliability by a combination of video analytics," in International Conference Image Analysis and Recognition, 2014, pp. 477-484

[8] Kumarguru Poobalan and Siau-Chuin Liew, “Fire detection algorithm using image processing techniques”, (AICS2015) (ISBN 978-967- 0792-06-4).

[9] Jareerat Seebamrungsat, Suphachai Praising, and Panomkhawn Suriyamongkol,“ Fire Detection in the Buildings Using Image Processing”, 2014 Third ICT International Student Project Conference.

[10] Gaurav Yadav, Vikas Gupta, Vinod Gaur, Dr. Mahua Bhattacharya, “Optimized flame detection using image processing based techniques”, IJCSE , ISSN : 0976-5166.

[11] A. E. Gunawardena, R. M. M. Ruwanthika, A. G. B. P. Jayasekara, Computer Vision Based Fire Alarming System, Moratuwa Engineering Research Conference (MERCon), 2016.

[12] C. Emmy Premal, S. S. Vinsley, Image Processing Based Forest Fire Detection using YCbCr Colour Model, 2014 International Conference on Circuits, Power and Computing Technologies.

[13] Jiaqiu Chen, Yaowei Wang_, Yonghong Tian, Tiejun Huang, “Wavelet based smoke detection method with rgb contrast-image and shape constraints”, Visual Communications and Image Processing (VCIP), 2013.

[14] Wenhao Wang, Hong Zhou, “Fire Detection Based on Flame Color and Area”, 2012 IEEE International Conference on Computer Science and Automation.

[15] Pedro Santana, Pedro Gomes, and Jos´e Barata, “A vision-based system for early fire detection”, 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC).

[16] V. Burak Celen , M. Fatih Demirci, “Fire Detection In Different Color Models”, Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition,2012

[17] Saylee Gharge, Sumeet Birla, Sachin Pandey, Rishi Dargad, Rahul Pandita, “Smoke and Fire Detection”, IJARCCE, Vol. 2, Issue 6, June 2013.

[18] Sudeep D. Thepade, Jaya H. Dewan, Akash A. Kulkarni, “Effect of color spaces on image compression using hybrid wavelet transform generated with varying proportions of constituent transforms ”, Third International Conference on Computational Intelligence and Information Technology, 2013.

[19] S. Y. Foo, "A fuzzy logic approach to fire detection in aircraft dry bays and engine compartments", IEEE Trans.

Industrial Electronics, vol. 47, no. 5, pp. 1161-1171, 2000

[20] B.U. Töreyin, Y. Dedeoglu, U. Gudukbay and A. E. Cetin, "Computer vision based method for real time fire and flame detection", Patt. Recog. L., vol. 27, pp. 49-58, 2006.

[21] C-C Ho, "Machine vision-based real-time early flame and smoke detection", Meas. Sci. Technol., vol. 20, pp.

[22] B.C. Ko, K-H Cheong and J-Y Nam, "Fire detection based on vision sensor and support vector machines", Fire Safety J., vol. 44, pp. 322-329, 2009.

[23] M. Doost Fatemeh and S. C. Kremer, "New directions in fuzzy automata ", Int. J. of Approximate Reasoning, vol. 38, pp. 175-214, 2005.

[24] F. Lin and H. Ying, "Modeling and control of fuzzy discrete event systems", IEEE Trans. On SMC-B, vol. 32, pp. 408-415, 2002

[25] C. M. Bishop, "Pattern recognition and machine learning, " Springer Publisher, 2006.

[26] A. Patcha and J. M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends, " Comput. Netw., vol. 51, no. 12, pp. 3448-3470, 2007. (Pub Itemid 46921030)

[27] W. Phillips III, M. Shah and N. V. Lobo, "Flame recognition in video, " Pattern Recognition Letters, vol. 23(1-3), pp. 319-327, 2002. (Pub Itemid 33119526)

[28] B. U. Toreyin, Y. Dedeoglu, and A. E. Petin, "Flame detection in video using hidden markov models, " ICIP '05, vol. 2, no. 2, pp. 1230-1233, 2005. (Pub Itemid 44539168)

[29] T. Chen, P. Wu and Y. Chiou, "An early fire-Detection method based on image processing, " ICIP '04, vol. 3, no. 24-27, pp. 1707-1710, 2004. (Pub Itemid 40820559)

[30] X. L. Zhou, F. X. Yu, Y. C. Wen, Z. M. Lu and G. H. Song, "Early fire detection based on flame contours in video, " Information Technology Journal, 2010.

GLOSSARY

- **Algorithm**: A set of rules or instructions for solving a specific problem or performing a task, such as detecting fire in this project.
- **Capital Savings**: Refers to the cost-saving aspect of the project, where existing hardware can be utilized, and expensive sensors can be avoided.
- **Crop Firing:** The detection of fires in agricultural fields, which can help prevent crop damage.
- **Detector:** The component of the HAAR Cascade Classifier system responsible for identifying objects based on the trained classifier.
- **Early Detection:** The capability to identify fires in their initial stages when they are still small and easier to control.
- **Fire Hazard Detection System:** Systems designed to identify and mitigate fire hazards, often involving a combination of sensors and detection methods.
- **Fire Detection Algorithm:** A set of computational steps used to analyze video data and detect the presence of fire, as described in your project.
- **Flame Detector:** A specialized sensor designed to detect the presence of flames or combustion.
- **Gas Sensor Detector:** A sensor that detects the presence of specific gases, which can be a sign of a fire or other hazards.
- **Heat Detector:** A device or sensor that monitors temperature changes and can indicate the presence of a fire.
- **Home Applications:** Fire detection systems installed in residential settings for early warning and protection.
- **HAAR Cascade Classifier:** A machine learning object detection method used to identify objects within images. In this project, it's used to detect fires.
- **Image Processing:** The manipulation and analysis of images to extract information or detect patterns. In this project, it's used for fire detection from live video feed.
- **Industrial Applications:** Environments such as factories, warehouses, and manufacturing facilities where fire detection is crucial for safety and asset protection.
- **Live Video Feed:** Real-time streaming of video data from a camera or other source.

- **Medical Applications:** Utilizing fire detection for medical facilities to ensure patient safety and prevent fire-related accidents.
- **Negative Image Samples:** Images that do not contain the object of interest (no fire in this case) and are used to train the HAAR Cascade Classifier.
- **Object Detection:** The process of identifying and locating specific objects within an image or video stream, in this case, the detection of flames or fires.
- **Positive Image Samples:** Images that contain the object you want to detect (fire in this case) and are used to train the HAAR Cascade Classifier.
- **Smoke Detector:** A device that senses the presence of smoke, often used as an indicator of fire.
- **Smoke Alarm:** A device that emits a loud warning sound when it detects smoke, typically used in homes and buildings.
- **Sensor-Free Fire Detection:** A method of detecting fire that does not rely on traditional sensors like smoke or heat detectors.
- **Temperature and Heat Sensors:** Sensors that monitor temperature variations and heat levels, commonly used in traditional fire detection systems.
- **Trainer:** The component of the HAAR Cascade Classifier system responsible for training the classifier using positive and negative image samples.