Course Project Report

on

PYRO VISION

BACHELOR OF TECHNOLOGY

in

CSE – INTERNET OF THINGS

Team Members	Roll No.
B. KOUSHIK	**22E51A6906**
D. ROHITH	**22E51A6918**
M. MOHITH	**22E51A6945**
N. VISWESH	**22E51A6947**

Department of ET, HITAM

HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

Autonomous, Approved by AICTE, Accredited by NAAC, NBA.

Gowdavelly (Village), Medchal (Mandal), Medchal - Malkajgiri (Dist.), Hyderabad, TS-501401

2023-24

HYDERABAD INSTITUE OF TECHNOLOGY AND MANAGEMENT

(Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC, NBA – TS 501401)

DEPARTMENT OF COMPUTER SCIENCE [INTERNET OF THINGS]

CERTIFICATE

This is to certify that the Course Level Project work entitled bearing **" PYROVISION "** is being submitted by **N.VISWESH (22E51A6947), B.KOUSHIK (22E51A6906), M. MOHIT (22E51A6945) , D.ROHIT (22E51A6918)** in partial fulfilment of the academic requirement, at Hyderabad Institute of Technology and Management, Hyderabad is a record of bonafied work carried out by them under our guidance. The matter contained in this document has not been submitted to any other University or institute.

Internal Guide

Mr. Naveen
Associate Professor
Department of ET, HITAM

Head of Department

Dr. M.V.A.Naidu
Professor & Head
Department of ET, HITAM

DECLARATION

We here by declare that the internship project entitled **"PYROVISION"** submitted to **Hyderabad Institute of Technology and Management** affiliated to **Jawaharlal Nehru Technological University, Hyderabad (JNTUH)** as part of academic requirement in Course level project.

TEAM MEMBERS	**ROLL NO.**
B. KOUSHIK	**22E51A6906**
D. ROHITH	**22E51A6918**
M. MOHITH	**22E51A6945**
N. VISWESH	**22E51A6947**

ABSTRACT

This project presents Pyro vision, a fire detection system that utilizes HSV (Hue, Saturation, Value) colour analysis to accurately and efficiently identify the presence of fire in images or video streams. Leveraging advanced computer vision techniques, the system aims to automatically detect flames by analysing the distinct colour characteristics in the HSV colour space. By distinguishing the unique colour signatures associated with fire, the system can perform real-time detection with high precision.

Fire accidents pose a significant threat, often resulting in substantial human and property losses. Traditional fire detection methods, such as smoke detectors and temperature sensors, have limitations in terms of speed and accuracy, often leading to delayed responses. Pyro vision addresses these limitations by implementing an intelligent fire detection system that provides prompt alarm outputs and message alerts to relevant authorities. The primary objective is to prevent the escalation of fires and mitigate associated losses and suffering.

The system employs a combination of HSV colour analysis and sophisticated image processing techniques to enhance detection accuracy. The process begins with converting the input images or video frames from the RGB color space to the HSV color space. This conversion allows the system to focus on the hue, saturation, and value components of each pixel, which are critical for identifying the color signatures of flames. By applying predefined HSV thresholds, the system can isolate potential fire regions within the scene.

To further refine the detection process, the system utilizes morphological operations such as dilation and erosion to reduce noise and enhance the detected fire regions. Contour detection algorithms are then employed to identify and analyse the shapes of these regions, ensuring that only those meeting specific criteria indicative of fire are considered. This multi-step approach significantly reduces false positives and improves the overall reliability of the system.

In addition to detection, Pyro vision incorporates a novel feature of capturing images of detected fire incidents and sending them to relevant authorities. This functionality facilitates better fire management and response strategies by providing real-time visual information about the incident. The system's ability to localize and semantically understand the detected fire regions contributes to more effective and coordinated emergency responses.

LIST OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Pyrovision is an innovative fire detection system meticulously designed to identify fire incidents in real-time using advanced computer vision techniques. By leveraging the HSV (Hue, Saturation, Value) color space, Pyrovision accurately detects the unique color signatures associated with flames in images and video streams. Traditional fire detection methods, such as smoke detectors and temperature sensors, often suffer from delays and inaccuracies, which can lead to significant damage and loss. Pyrovision addresses these challenges by providing a more responsive and precise solution, revolutionizing the approach to fire safety.

The system operates by converting video frames from the RGB color space to the HSV color space, a transformation that highlights the hue, saturation, and value components critical for detecting fire. This conversion is essential because the HSV color space separates image intensity from color information, making it easier to distinguish the bright, saturated colors of fire from the background. By applying predefined HSV thresholds, Pyrovision isolates potential fire regions within each frame, effectively distinguishing flames from other objects.

To further refine the detection process, Pyrovision employs morphological operations such as dilation and erosion. These operations help reduce noise in the image, enhancing the detected fire regions and making the system more robust against false positives. Contour detection algorithms are then used to identify and analyze the shapes of these regions, ensuring that only those meeting specific criteria indicative of fire are considered. This multi-step approach significantly improves the overall accuracy and reliability of the system, ensuring that genuine fire incidents are promptly and accurately detected.

In addition to detecting fire, Pyrovision features an alert mechanism that captures images of the detected fire incidents and sends them to relevant authorities. This functionality supports better fire management and response strategies by providing real-time visual information. When a fire is detected, the system immediately triggers alarms and sends alerts, including images and location data, to emergency response teams. This enables faster and more informed decision-making, potentially saving lives and minimizing property damage.

3.1 Objective of the Project

1. Accurate Fire Detection:

- The system aims to accurately identify fire incidents by analyzing the unique color signatures associated with flames. By leveraging the HSV color space, Pyrovision can distinguish the bright, saturated colors of fire from other elements in the scene, ensuring precise detection. Accurate fire detection is crucial to minimizing false alarms and ensuring that real fire incidents are promptly identified and addressed.

2. Real-Time Monitoring and Alerts:

- Pyrovision is designed to operate in real-time, continuously monitoring video streams for signs of fire. Upon detecting a potential fire, the system immediately triggers alarms and sends alerts to relevant authorities. Real-time monitoring and alerts enable swift responses to fire incidents, helping to prevent the escalation of fires and reduce the risk of significant damage and loss.

3. Noise Reduction and False Positive Minimization:

- To enhance detection accuracy and reliability, Pyrovision employs morphological operations such as dilation and erosion to reduce noise in the image. These operations help refine the detected fire regions, making the system more robust against false positives. Minimizing false positives is essential for ensuring that emergency response teams are only alerted to genuine fire incidents, thereby improving the efficiency of fire management efforts.

4. Image Capture and Reporting:

- In addition to detecting fire, Pyrovision captures images of detected fire incidents and sends them to relevant authorities. This functionality provides real-time visual information about the fire, facilitating better fire management and response strategies. By providing accurate and timely visual data, Pyrovision helps emergency responders assess the situation more effectively and make informed decisions.

5. Scalability and Deployment:

- Pyrovision is designed to be scalable and suitable for deployment in various environments, including residential, commercial, and industrial settings. The system's

computational efficiency and use of the HSV color model make it adaptable to different use cases and conditions. Scalability ensures that Pyrovision can be implemented widely, enhancing fire safety across diverse settings and contributing to the overall goal of reducing fire-related risks and losses.

6. Integration with Existing Systems:

- Pyrovision aims to seamlessly integrate with existing CCTV and surveillance systems, making it easy to adopt without the need for extensive infrastructure changes. This integration capability ensures that Pyrovision can be quickly and cost-effectively deployed, leveraging existing video surveillance networks to enhance fire detection capabilities.

7. Cost-Effectiveness:

- By utilizing the open-source OpenCV library and efficient image processing techniques, Pyrovision aims to provide a cost-effective solution for fire detection. The system's modest computational requirements allow it to be implemented on standard hardware, reducing overall costs while maintaining high performance and reliability. Cost-effectiveness ensures that Pyrovision can be adopted by a wide range of users, from small businesses to large industrial facilities.

8. User-Friendly Interface and Maintenance:

- Pyrovision is designed with a user-friendly interface that allows for easy configuration, monitoring, and maintenance. The system provides intuitive controls and clear visual indicators to help users manage fire detection settings and review alert history. Simplified maintenance procedures ensure that Pyrovision remains reliable and effective over time, reducing the need for specialized technical support.

CHAPTER 2

HARDWARE AND SOFTWARE REQUIREMENTS

2.1 Hardware Requirements:

1. *CCTV Cameras:*

- High-resolution CCTV cameras capable of capturing clear video footage.

- Cameras with good low-light performance for detecting fires in various lighting conditions.

2. *Computer/Server:*

- *Processor:* Multi-core processor (e.g., Intel i5/i7 or equivalent) to handle real-time video processing.

- *RAM:* At least 8 GB of RAM to ensure smooth operation and handling of multiple video streams.

- *Storage:* Sufficient storage capacity (e.g., 1 TB HDD/SSD) to store video footage and detected incident images.

- *GPU (Optional):* A dedicated GPU (e.g., NVIDIA GTX/RTX series) can enhance image processing performance, especially for high-resolution video streams.

3. *Network Infrastructure:*

- Reliable network connectivity to ensure continuous video feed from CCTV cameras to the processing server.

- Internet connection for sending alerts and incident images to relevant authorities.

4. *Power Supply:*

- Uninterruptible Power Supply (UPS) to ensure continuous operation during power outages.

2.2 Software Requirements:

1. *Operating System:*

- A compatible operating system for running the software, such as:

  - Windows 10 or later

- Ubuntu 18.04 or later

2. *Programming Environment:*

- *Python:* The primary programming language for developing the fire detection system.

- *OpenCV:* An open-source computer vision library used for image processing and HSV color analysis.

- *NumPy:* A library for numerical computations in Python, essential for handling image data.

3. *Development Tools:*

- *IDE/Text Editor:* Integrated Development Environment (IDE) such as PyCharm, VS Code, or any preferred text editor for writing and debugging code.

- *Git:* Version control system for managing and tracking changes in the codebase.

4. *Dependencies and Libraries:*

- *OpenCV-Python:* Python bindings for OpenCV to utilize its image processing functions.

- *NumPy:* For numerical operations and array manipulations.

- *Matplotlib (Optional):* For visualizing images and data during development and testing.

- *Requests:* For sending alerts and images to external servers or APIs.

5. *Alert and Notification System:*

- *Email/SMTP Server:* For sending email alerts to relevant authorities.

- *SMS Gateway (Optional):* For sending SMS notifications if needed.

- *Push Notification Service (Optional):* For real-time push notifications to mobile devices.

6. *Database (Optional):*

- *SQLite/MySQL/PostgreSQL:* For storing incident logs, alert history, and system configurations if needed.

7. *User Interface:*

- *Web Framework (Optional):* Frameworks like Flask or Django for creating a web-based user interface for monitoring and managing the system.

CHAPTER 3

DESIGN AND METHODOLOGY

3.1 User Interface (UI) Design:

1. *Dashboard:*

- *Overview Panel:* Displays a real-time feed from CCTV cameras, with indicators for detected fire incidents.

- *Incident Log:* A chronological list of detected fire incidents with timestamps, locations, and severity levels.

- *Statistics and Reports:* Graphs and charts showing the frequency of fire incidents over time, detection accuracy, and false positive rates.

2. *Live Video Feed:*

- *Camera View:* Real-time video feed from multiple CCTV cameras.

- *Fire Detection Overlay:* Highlighted regions in the video feed where fire is detected.

- *Control Buttons:* Options to pause, play, zoom in/out, and switch between different camera feeds.

3. *Alert Management:*

- *Alert Settings:* Configuration options for setting up alert thresholds, notification preferences, and recipient contact details.

- *Alert History:* A record of all alerts sent, with details on the incident and the status of the alert (e.g., acknowledged, resolved).

4. *Configuration Panel:*

- *Camera Settings:* Options to add, remove, and configure CCTV cameras.

- *Detection Settings:* Thresholds and parameters for HSV color analysis and image processing techniques.

- *System Settings:* General settings, including user management, system logs, and maintenance options.

5. *Incident Details:*

- *Incident Summary:* Detailed information about each detected fire incident, including captured images, location, and time.

- *Response Actions:* Logs of actions taken in response to the alert (e.g., notifications sent, acknowledgments received).

6. *User Management:*

- *User Roles and Permissions:* Define different user roles (e.g., admin, operator) and assign appropriate permissions.

- *User Profiles:* Manage user information, contact details, and notification preferences.

3.2 Methodology:

1. *User-Centered Design (UCD):*

- *User Research:* Conduct interviews and surveys with potential users (e.g., security personnel, emergency responders) to understand their needs and preferences.

- *Persona Development:* Create user personas representing different types of users to guide design decisions.

- *Usability Testing:* Regularly test the interface with real users to gather feedback and make iterative improvements.

2. *Prototyping:*

- *Low-Fidelity Prototypes:* Start with sketches or wireframes to outline the basic layout and functionality of the interface.

- *High-Fidelity Prototypes:* Develop interactive prototypes using tools like Figma, Adobe XD, or Sketch to simulate the final user experience.

- *Feedback Iteration:* Use feedback from prototype testing to refine and enhance the design.

3. *Responsive Design:*

- *Cross-Platform Compatibility:* Ensure the interface is accessible and functional across different devices (e.g., desktops, tablets, smartphones).

- *Adaptive Layouts:* Design flexible layouts that adjust to various screen sizes and orientations.

4. *Accessibility:*

- *Inclusive Design:* Follow accessibility guidelines (e.g., WCAG) to ensure the interface is usable by people with disabilities.

- *Keyboard Navigation:* Implement keyboard shortcuts and navigation options for users who cannot use a mouse.

5. *Consistency and Standards:*

- *Design System:* Develop a design system with standardized components, styles, and guidelines to maintain consistency throughout the interface.

- *Familiar Patterns:* Use familiar UI patterns and conventions to make the interface intuitive and easy to learn.

6. *Feedback and Responsiveness:*

- *Real-Time Feedback:* Provide immediate feedback to user actions (e.g., highlighting detected fire regions, confirmation messages for settings changes).

- *Error Handling:* Clearly communicate errors and provide guidance on how to resolve them.

3.3 Sample Output:

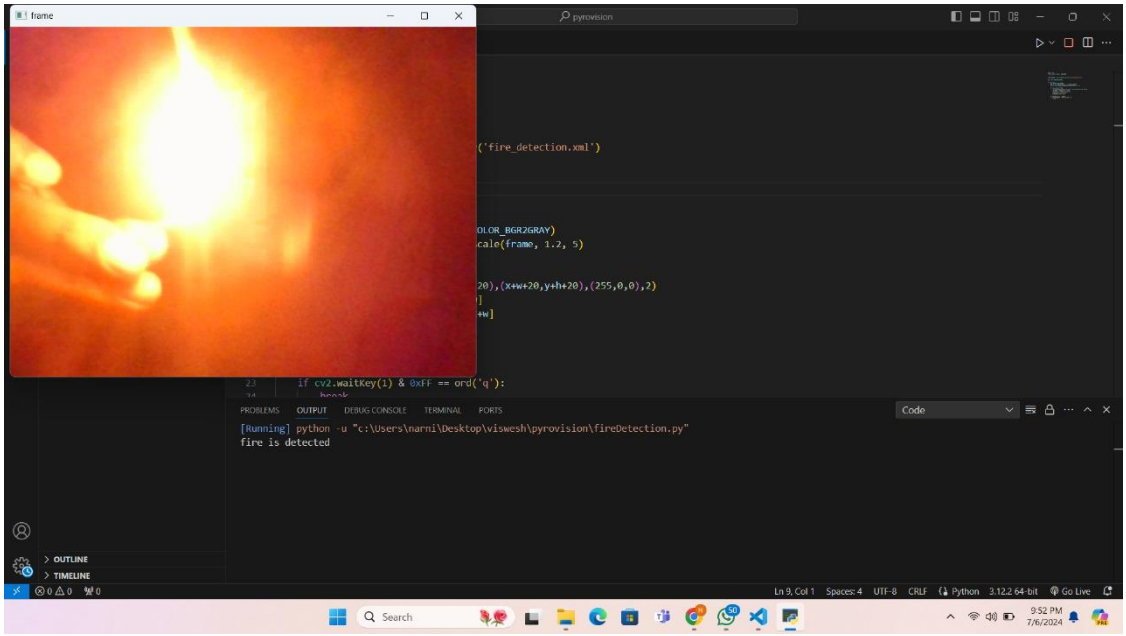

Figure 3.3(a)

3.4 Working:

The Pyrovision project functions as an advanced fire detection system by using HSV (Hue, Saturation, Value) color analysis to accurately identify fire incidents in real-time. The system's working can be broken down into several key stages:

1. Video Input and Preprocessing:

- **Video Feed Acquisition**: The system captures real-time video streams from CCTV cameras placed strategically to monitor areas prone to fire hazards.

- **Frame Extraction**: The continuous video feed is divided into individual frames for further processing.

2. Color Space Conversion:

- **RGB to HSV Conversion**: Each frame is converted from the RGB color space to the HSV color space. This conversion is crucial as the HSV color space better isolates color information (hue and saturation) from intensity (value), making it easier to detect the bright, saturated colors characteristic of fire.

3. Fire Detection:

- **HSV Thresholding**: The system applies predefined HSV thresholds to isolate potential fire regions within each frame. These thresholds are set based on the typical hue, saturation, and value ranges that characterize fire.

- **Mask Creation**: A binary mask is created where pixels falling within the HSV thresholds are marked as potential fire pixels.

4. Noise Reduction and Region Refinement:

- **Morphological Operations**: Morphological operations such as dilation and erosion are applied to the binary mask to reduce noise and refine the detected fire regions. Dilation helps to fill small gaps in detected regions, while erosion removes small, irrelevant noise.

- **Contour Detection**: Contour detection algorithms are used to identify and analyze the shapes of the detected regions. This step ensures that only regions meeting specific criteria (such as size and shape) indicative of fire are considered for further analysis.

5. Fire Localization and Alert Generation:

- **Bounding Boxes**: Bounding boxes are drawn around the detected fire regions to localize and highlight the areas where fire is present.

- **Real-Time Alerts**: Upon detecting a fire, the system generates real-time alerts. These alerts include capturing images of the detected fire incidents and sending them to relevant authorities via predefined communication channels (e.g., email, SMS, push notifications).

- **Alarm Activation**: The system can trigger local alarms to alert on-site personnel immediately.

6. **Data Logging and Reporting:**

- **Incident Logging**: Details of each detected fire incident, including timestamps, locations, and captured images, are logged in a database for future reference and analysis.

- **Reporting**: The system can generate periodic reports summarizing the frequency and details of fire incidents, helping in assessing fire risk patterns and improving safety measures.

7. User Interface and Management:

- **Dashboard**: A user-friendly dashboard provides real-time visualization of the monitored areas, detected fire incidents, and system status.

- **Configuration Panel**: Users can configure detection settings, such as HSV thresholds, alert preferences, and camera settings, through an intuitive interface.

- **User Management**: The system supports multiple user roles with different permissions, ensuring that only authorized personnel can access critical functions and data.

Workflow Summary:

1. Capture real-time video feed from CCTV cameras.

2. Convert video frames from RGB to HSV color space.

3. Apply HSV thresholds to isolate potential fire regions.

4. Use morphological operations and contour detection to refine fire regions.

5. Generate real-time alerts and trigger alarms upon fire detection.

6. Log incident details and provide periodic reports.

7. Enable users to configure and manage the system through a user-friendly interface.

3.5 Code:

fire_detection.xml:

```
<?xml version="1.0"?>
<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 -1 27 1.0459426790475845e-01</internalNodes>
          <leafValues>
            -9.3814432621002197e-01 9.0361446142196655e-
01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 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 -1 7 4.0762656927108765e-01</internalNodes>
<leafValues>
-8.6901766061782837e-01 7.5675678253173828e-
01</leafValues></_>
<_>
<internalNodes>
0 -1 6 1.3779436051845551e-01</internalNodes>
<leafValues>
-8.8750886917114258e-01 2.8855907917022705e-
01</leafValues></_>
<_>
<internalNodes>
0 -1 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 -1 2 4.1499978303909302e-01</internalNodes>
<leafValues>
-8.6945170164108276e-01 5.0000000000000000e-
01</leafValues></_>
<_>
<internalNodes>
0 -1 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 -1 13 4.9460947513580322e-01</internalNodes>
<leafValues>
-8.1265825033187866e-01 4.2105263471603394e-
01</leafValues></_>
<_>
<internalNodes>
0 -1 30 -1.0531554371118546e-01</internalNodes>

```
          <leafValues>
            2.6704105734825134e-01 -8.5486882925033569e-
01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 41 3.4948602318763733e-02</internalNodes>
          <leafValues>
            -8.9780002832412720e-01 2.7469578385353088e-
01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 81 -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 -1 12 3.6403301358222961e-01</internalNodes>
          <leafValues>
            -8.5209006071090698e-01 -1.5000000596046448e-
01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 14 3.3683568239212036e-02</internalNodes>
          <leafValues>
            -5.1311254501342773e-01 4.6315157413482666e-
01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 69 1.3609335292130709e-04</internalNodes>
          <leafValues>
            3.1104850769042969e-01 -7.2132861614227295e-
01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 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 -1 26 6.7514598369598389e-02</internalNodes>
          <leafValues>
            -7.7188330888748169e-01 2.1276595070958138e-
02</leafValues></_>
        <_>
          <internalNodes>
            0 -1 8 3.4416305425111204e-05</internalNodes>
          <leafValues>
            -8.5286557674407959e-01 1.4937944710254669e-
            0 -1 48 -1.8346133583690971e-04</internalNodes>
          <leafValues>
            -9.0394043922424316e-01 1.9481389224529266e-
01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 60 8.8960374705493450e-04</internalNodes>
          <leafValues>
            -1. 1.9558252394199371e-
01</leafValues></></weakClassifiers></>
    <!-- stage 6 -->
    <_>
      <maxWeakCount>3</maxWeakCount>
      <stageThreshold>-7.9717081785202026e-01</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 13 4.7819304466247559e-01</internalNodes>
          <leafValues>
            -8.1215471029281616e-01 4.5871559530496597e-
02</leafValues></_>
        <_>
          <internalNodes>
            0 -1 88 -3.8815978768980131e-05</internalNodes>
          <leafValues>
            1.0888232290744781e-01 -9.4263499975204468e-
01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 77 -4.0303770219907165e-04</internalNodes>
          <leafValues>
            -9.5192468166351318e-01 1.8408368527889252e-
01</leafValues></></weakClassifiers></>
    <!-- stage 7 -->
    <_>
      <maxWeakCount>5</maxWeakCount>
```

```
      <stageThreshold>-1.6301398277282715e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 33 -1.1418235488235950e-02</internalNodes>
          <leafValues>
            7.8947371244430542e-01 -6.7256635427474976e-
01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 51 -3.8923887768760324e-04</internalNodes>
          <leafValues>
            -9.0711349248886108e-01 9.3762204051017761e-
02</leafValues></_>
        <_>
          <internalNodes>
            0 -1 69 -1.6685485024936497e-04</internalNodes>
          <leafValues>
            -6.4114141464233398e-01 2.8773099184036255e-
01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 5 -1.2296087516006082e-04</internalNodes>
          <leafValues>
            -8.7147945165634155e-01 2.3817995190620422e-
01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 0 2.7513506211107597e-05</internalNodes>
          <leafValues>
            -6.4837419986724854e-01 2.8315281867980957e-
01</leafValues></></weakClassifiers></>
    <!-- stage 8 -->
    <_>
      <maxWeakCount>5</maxWeakCount>
      <stageThreshold>-9.2982792854309082e-01</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 32 -7.2457015514373779e-02</internalNodes>
          <leafValues>
          <internalNodes>
            0 -1 31 -1.1788629926741123e-03</internalNodes>
          <leafValues>
            -5.7471264153718948e-02 -7.3958331346511841e-
01</leafValues></_>
        <_>
          <internalNodes>
```

```
            0 -1 65 -7.0812908234074712e-05</internalNodes>
          <leafValues>
            -8.5187572240829468e-01 6.6193021833896637e-
02</leafValues></_>
        <_>
          <internalNodes>
            0 -1 84 3.0261104257078841e-05</internalNodes>
          <leafValues>
            -6.9307458400726318e-01 1.9375345110893250e-
01</leafValues></_>
          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 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>
```

fireDetection.py:

```
import cv2
from playsound import playsound


fire_cascade = cv2.CascadeClassifier('fire_detection.xml')
```

```
cap = cv2.VideoCapture(0)

while(True):
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    fire = fire_cascade.detectMultiScale(frame, 1.2, 5)

    for (x,y,w,h) in fire:
        cv2.rectangle(frame,(x-20,y-20),(x+w+20,y+h+20),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]
        print("fire is detected")
        playsound('audio.mp3')

    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

opencv_script.sh :

```
#######################################
# INSTALL OPENCV ON UBUNTU OR DEBIAN #
#######################################

# |          THIS SCRIPT IS TESTED CORRECTLY ON          |
# |------------------------------------------------------|
```

```
# | OS               | OpenCV       | Test | Last test   |
# |------------------|--------------|------|-------------|
# | Ubuntu 18.04 LTS | OpenCV 3.4.2 | OK   | 18 Jul 2018 |
# | Debian 9.5       | OpenCV 3.4.2 | OK   | 18 Jul 2018 |
# |-------------------------------------------------------- |
# | Debian 9.0       | OpenCV 3.2.0 | OK   | 25 Jun 2017 |
# | Debian 8.8       | OpenCV 3.2.0 | OK   | 20 May 2017 |
# | Ubuntu 16.04 LTS | OpenCV 3.2.0 | OK   | 20 May 2017 |


# VERSION TO BE INSTALLED

OPENCV_VERSION='3.4.2'


# 1. KEEP UBUNTU OR DEBIAN UP TO DATE

sudo apt-get -y update
# sudo apt-get -y upgrade       # Uncomment this line to install the newest versions of all packages currently installed
# sudo apt-get -y dist-upgrade  # Uncomment this line to, in addition to 'upgrade', handles changing dependencies with new versions of packages
# sudo apt-get -y autoremove    # Uncomment this line to remove packages that are now no longer needed


# 2. INSTALL THE DEPENDENCIES

# Build tools:
sudo apt-get install -y build-essential cmake

# GUI (if you want to use GTK instead of Qt, replace 'qt5-default' with 'libgtkglext1-dev' and remove '-DWITH_QT=ON' option in CMake):
sudo apt-get install -y qt5-default libvtk6-dev

# Media I/O:
sudo apt-get install -y zlib1g-dev libjpeg-dev libwebp-dev libpng-dev libtiff5-dev libjasper-dev libopenexr-dev libgdal-dev

# Video I/O:
sudo apt-get install -y libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev yasm libopencore-amrnb-dev libopencore-amrwb-dev libv4l-dev libxine2-dev

# Parallelism and linear algebra libraries:
sudo apt-get install -y libtbb-dev libeigen3-dev
```

```
# Python:
sudo apt-get install -y python-dev python-tk python-numpy python3-
dev python3-tk python3-numpy

# Java:
sudo apt-get install -y ant default-jdk

# Documentation:
sudo apt-get install -y doxygen


# 3. INSTALL THE LIBRARY

sudo apt-get install -y unzip wget
wget https://github.com/opencv/opencv/archive/${OPENCV_VERSION}.zip
unzip ${OPENCV_VERSION}.zip
rm ${OPENCV_VERSION}.zip
mv opencv-${OPENCV_VERSION} OpenCV
cd OpenCV
mkdir build
cd build
cmake -DWITH_QT=ON -DWITH_OPENGL=ON -DFORCE_VTK=ON -DWITH_TBB=ON -
DWITH_GDAL=ON -DWITH_XINE=ON -DBUILD_EXAMPLES=ON -
DENABLE_PRECOMPILED_HEADERS=OFF ..
make -j4
sudo make install
sudo ldconfig


# 4. EXECUTE SOME OPENCV EXAMPLES AND COMPILE A DEMONSTRATION

# To complete this step, please visit
'http://milq.github.io/install-opencv-ubuntu-debian'.
```

CONCLUSION

Pyrovision represents a significant advancement in fire detection technology, offering a precise, efficient, and real-time solution to the critical issue of fire safety. By leveraging HSV color classification and advanced computer vision techniques, Pyrovision accurately detects the unique color signatures of flames, providing a more responsive and reliable alternative to traditional fire detection methods. The system's ability to operate in real-time, coupled with its

alert mechanism, ensures that potential fire incidents are promptly identified and communicated to relevant authorities, enabling swift and informed responses.

The project's innovative use of morphological operations and contour detection algorithms enhances the system's accuracy and reliability, significantly reducing false positives and ensuring that genuine fire incidents are detected. Additionally, the scalability and adaptability of Pyrovision make it suitable for deployment in various environments, including residential, commercial, and industrial settings. This versatility ensures that a wide range of users can benefit from improved fire safety measures.

Pyrovision's integration with existing CCTV systems and its user-friendly interface further contribute to its effectiveness, allowing for easy adoption and management. The system's ability to capture and report fire incidents in real-time provides valuable visual information to emergency responders, facilitating better fire management and response strategies.

In conclusion, Pyrovision sets a new standard in fire detection systems, demonstrating how advanced computer vision techniques and HSV color analysis can create a robust, efficient, and reliable fire safety solution. By addressing the limitations of traditional methods and providing timely alerts and critical visual information, Pyrovision helps prevent the escalation of fires, ultimately saving lives and protecting property. This project underscores the potential for future advancements in fire safety technology and highlights the importance of continued innovation in this vital area.

REFERENCES