

A
MINI PROJECT REPORT
ON
FIRE DETECTION AND ALERTING SYSTEM

Submitted in partial fulfillment of the requirements

For the award of Degree of

BACHELOR OF ENGINEERING
IN

CSE (AI ML)

Submitted By

KASHETTY ANEESH 245320748020

M RAGHAVA RUTHWIK 245320748027

K PRANEETH KUMAR 245320748306

Under the guidance

Of

Mrs. S. SWAPNA

ASSISTANT PROFESSOR



Department of CSE(AIML)

NEIL GOGTE INSTITUTE OF TECHNOLOGY

Kachavanisingaram Village, Hyderabad, Telangana 500058.

JANUARY 2023



NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Osmania University,
Hyderabad

CERTIFICATE

*This is to certify that the Mini project work entitled “**FIRE DETECTION AND ALERTING SYSTEM**” is a bonafide work carried out by **KASHETTY ANEESH (245320748020)**,*

***M RAGHAVA RUTHWIK (245320748027)**, **K PRANEETH KUMAR (245320748306)** of III year V semester **Bachelor of Engineering in CSE(AIML)** by Osmania University, Hyderabad during the academic year **2022-2023** is a record of bonafide work carried out by them. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree*

Internal Guide

Mrs. S. SWAPNA
Assistant Professor

Head of Department

Dr. K. Madhuri
Associate Professor

External



NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

DECLARATION

We hereby declare that the Mini Project Report entitled, “**FIRE DETECTION AND ALERTING SYSTEM**” submitted for the B.E degree is entirely our work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

Date:

KASHETTY ANEESH 245320748020

M RAGHAVA RUTHWIK 245320747027

K PRANEETH KUMAR 245320748306



NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

ACKNOWLEDGEMENT

We are happy to express our deep sense of gratitude to the principal of the college **Dr. R. Shyam Sunder, Professor**, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank, **Dr. K. Madhuri, Head of the Department**, CSE(AIML), Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We would also like to thank my internal guide **Mrs. S. Swapna, Assistant Professor** for her technical guidance & constant encouragement.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering for their timely suggestions, healthy criticism and motivation during this work.

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

ABSTRACT

Pattern recognition (PR) is realized as a human recognition process which can be completed by computer technology. We should first enter useful information of identifying the object into the computer. For this reason, we must abstract the recognition object and establish its mathematical model to describe it and replace the recognition object for what the machine can process. The description of this object is the pattern. Simply speaking, the pattern recognition is to identify the category to which the object belongs, such as the face in face recognition, number plates in vehicles. Our project is based on PR which is to detect the fire.

Haar cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location. This algorithm is not so complex and can run in real-time. The pre-processing required in Haar Cascade compared with other classification algorithms is higher. The anticipation time for an object is less when compared to other models hence this model is used widely to decrease the detection time. When the fire is detected it plays an alarm sound which helps the victims to evacuate the place and also sends an message as an information to nearby fire station and the user. The user can be the owner of the place or the security chief of the place.

TABLE OF CONTENTS

CHAPTER	PAGE NO.
1. ACKNOWLEDGEMENT	I
ABSTRACT	II
LIST OF FIGURES	V
LIST OF TABLES	V
INTRODUCTION	
1.1 PROBLEM STATEMENT	1
1.2 MOTIVATION	1
1.3 SCOPE	1 - 2
1.4 OUTLINE	2
2. LITERATURE SURVEY	
2.1 EXISTING SYSTEM	3
2.2 PROPOSED SYSTEM	5
3. SOFTWARE REQUIREMENTS SPECIFICATION	
3.1 OVERALL DESCRIPTION	4
3.2 OPERATING ENVIRONMENT	4
3.3 FUNCTIONAL REQUIREMENTS	4 – 5
3.4 NON-FUNCTIONAL REQUIREMENTS	5 - 7
4. SYSTEM DESIGN	
4.1 USE-CASE DIAGRAM	8 - 9
4.2 CLASS DIAGRAM	10
4.3 SEQUENCE DIAGRAM	11
4.4 ACTIVITY DIAGRAM	12

5.	IMPLEMENTATION	13 - 19
	5.1 SAMPLE CODE	20 - 22
5.	TESTING	
	6.1 TEST CASES	23-25
6.	SCREENSHOTS	26-30
7.	CONCLUSION AND FUTURE SCOPE	31
8.	BIBLIOGRAPHY	32
9.	APPENDIX A: TOOLS AND TECHNOLOGY	33 - 34

List of Figures

Figure No.	Name of Figure	Page No.
1.	Use case Diagram	8 – 9
2.	Class Diagram	10
3.	Sequence Diagram	11
4.	Activity Diagram	12

List of Test Cases

Test Case No.	Name of Test Case	Page No.
1.	Python Installation verification	23
2.	Python Programs Integration Testing	23
3.	Collect Dataset and Load the Dataset	24
4.	Fire Detection	24
5.	Checks alarm message	25
6.	Checks alert message	25

CHAPTER – 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Fire plays a major role in providing light and heat but it is very dangerous as it spreads rapidly. So we need to be more cautious in monitoring it. Our project deals with monitoring of fire using camera. This mechanism gives out an alert sound and also sends an alert to the respective User or nearby fire station.

1.2 MOTIVATION

This problem is not only challenging but also its solution is applicable to other fine-grained real-world problems. This project is proposed for helping the victims to evacuate at the earliest, reduces the property loss, reduce the pollution caused by burning of the hazardous chemicals etc. This project is cost efficient as we only need the camera access present at the location.

1.3 SCOPE

This application deals with the problem of detecting the fire by using the Haar Cascade. Haar Cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location. This algorithm is not so complex and can run in real-time. We can train a Haar-Cascade detector to detect various objects like cars, bikes, buildings, fruits, etc. Haar cascade uses the cascading window, and it tries to compute features in every window and classify whether it could be an object. “Playsound” module which contains playsound function is used to play the alerting message. “Requests” module is used to send HTTP requests to the particularly mentioned Url-link.

1.4 OUTLINE

The Haar Cascade model is trained with huge amount of data which consists of both the positive, negative images such that it has a good accuracy. The input to the model is given in form of the video which is in real-time. We used “RAPIWHA” Whatsapp API to send the alerting message and playsound module to play sound when fire is detected.

CHAPTER – 2

LITERATURE SURVEY

EXISTING SYSTEM:

Machine learning (ML) represents a set of techniques that allow systems to discover the required representations to features detection or classification from the raw data. The performance of works in the classification system depends on the quality of the features. Most of the existing “Fire Detection Systems” consists of only hardware i.e the detection sensors. They only detect fire & smoke then alert just by an alarm. They have some limitations and designed to sense fire with the smoke, which is limited to areas. And the physical sensors can be fooled easily which has become a major problem.

PROPOSED SYSTEM:

Haar cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location. This algorithm is not so complex and can run in real-time. The pre-processing required in Haar Cascade compared with other classification algorithms is higher. The anticipation time for an object is less when compared to other models hence this model is used widely to decrease the detection time. When the fire is detected it plays an alerting sound which helps the victims to evacuate the place and also sends an message as an information to nearby fire station.

- Input Real-Time Video
- Haar Cascade Model
- Output Sound, Alerting message and region of fire in Frame.

CHAPTER - 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Overall Description:

This SRS is an overview of the whole project scenario. This document is to present a detailed description of the course management system. It will explain the purpose and features of the system, the interfaces of the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both stakeholders and developers of the system.

3.2. Operating Environment:

Software Requirements:

Operating System	:	Windows 7 (Min)
Back End	:	Python

Hardware Requirements:

Processor	:	Intel Pentium® i3 Core Processor (Min)
Speed	:	2.9 GHz (Min)
RAM	:	4 GB (Min)
Hard Disk	:	8 GB (Min)

3.3 Functional Requirements:

User Functionality:

- The user will be able to upload real time video using camera .
- The user can see information regarding the fire accident (If any takes place at his place).

Admin Functionality:

- The admin manages the System.
- The admin can increase accuracy of the model.
- The admin can make changes to the system as per user requirements.
- The admin can implement a better algorithm if at all a better algorithm is created in future.

3.4 Non-Functional Requirements:

3.4.1 Performance Requirements:

Performance requirements refer to static numerical requirements placed on the interaction between the users and the software.

Response Time:

Average response time shall be less than 5 sec.

Recovery Time:

In case of system failure, the redundant system shall resume operations within 30 secs. Average repair time shall be less than 30 minutes.

Start-Up/Shutdown Time:

The system shall be operational within 1 minute of starting up.

Capacity:

-NA

Utilization of Resources:

-NA-

3.4.2 Safety Requirements:

-NA-

3.4.3 Security Requirements:

The model will be running on a secure website i.e., an HTTPS website and also on a secure browser such as Google Chrome, Brave, etc.

3.4.4 Software Quality Attributes:

Reliability:

The system shall be reliable i.e., in case the application crashes, just restart the application.

Availability:

The application will be available to limited users so as there should be a business Tie-up with the fire stations.

Security:

The model will be running 100% only on a python compiler which is much secure.

Maintainability:

The model shall be designed in such a way that it will be very easy to maintain it in future. Our model is image processing model. The system maintenance is done easily by just improving the accuracy of the model. The future versions can also be expected with more features in the future.

Usability:

The interfaces of the system will be user friendly enough that every user will be able to use it easily

Scalability:

The system will be designed in such a way that it will be extendable. If more algorithms are going to be added in the system, then it would be easily done.

CHAPTER-4

SYSTEM DESIGN

Use case Diagram:

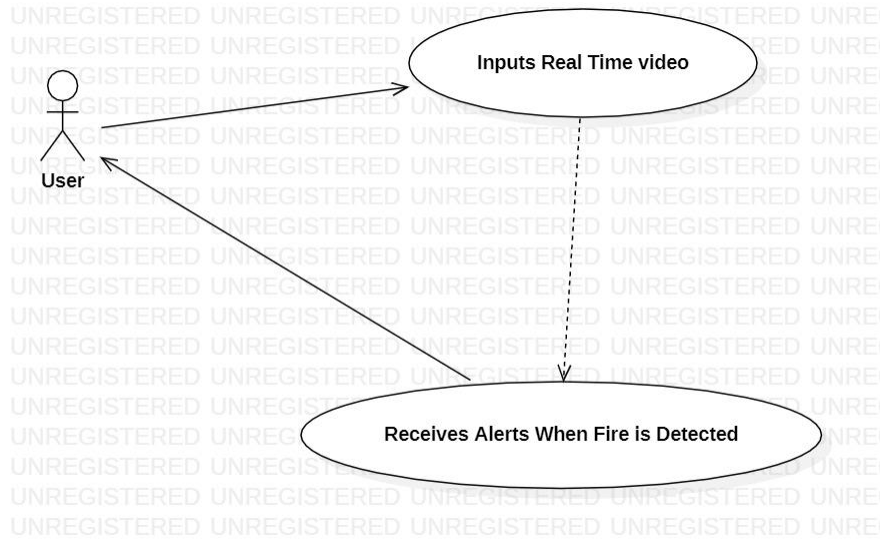


Fig 4.1: Use case diagram for User

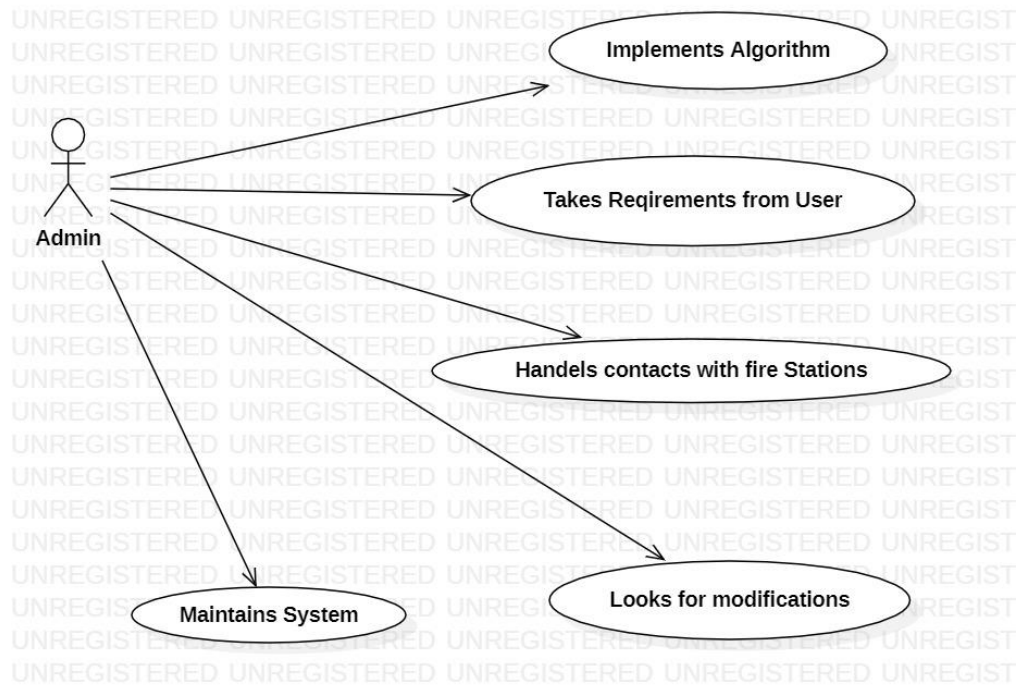


Fig 4.2: Use case diagram for Admin

Class Diagram:

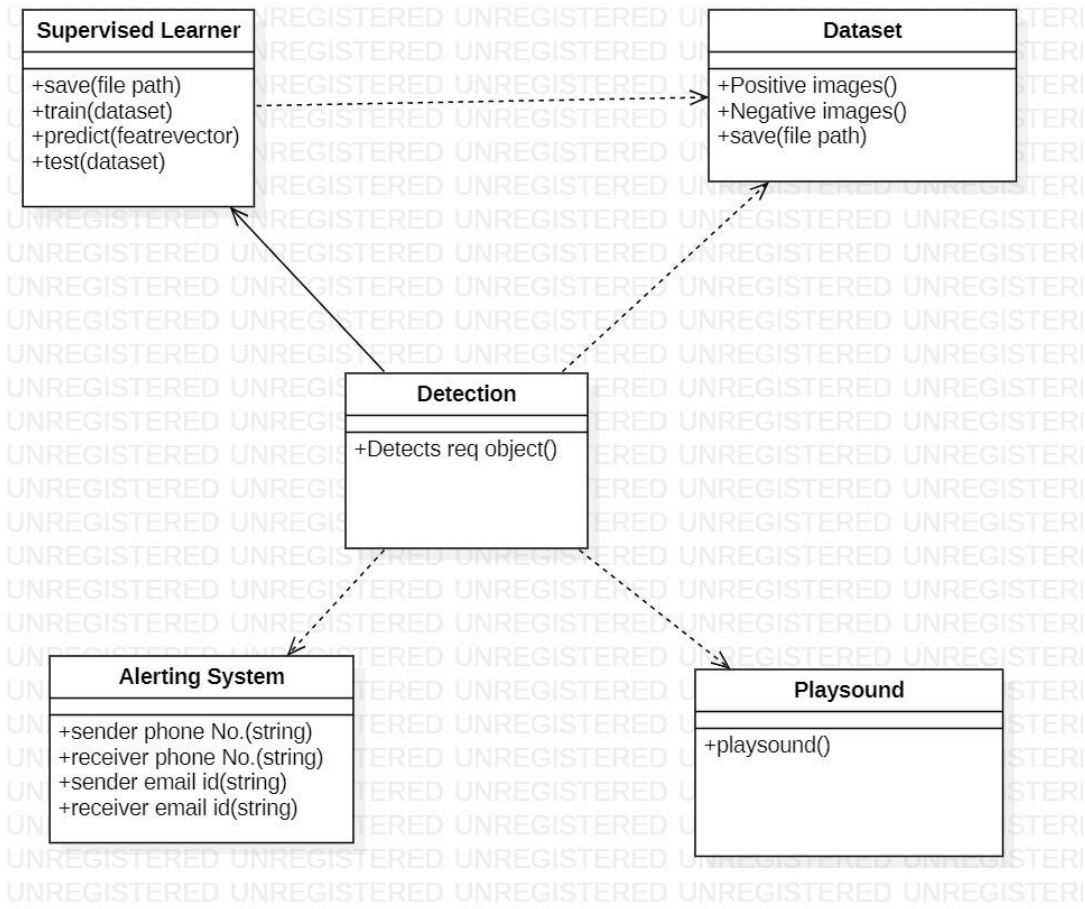


Fig.4.3 : Class diagram for Fire Detection.

Sequence Diagram:

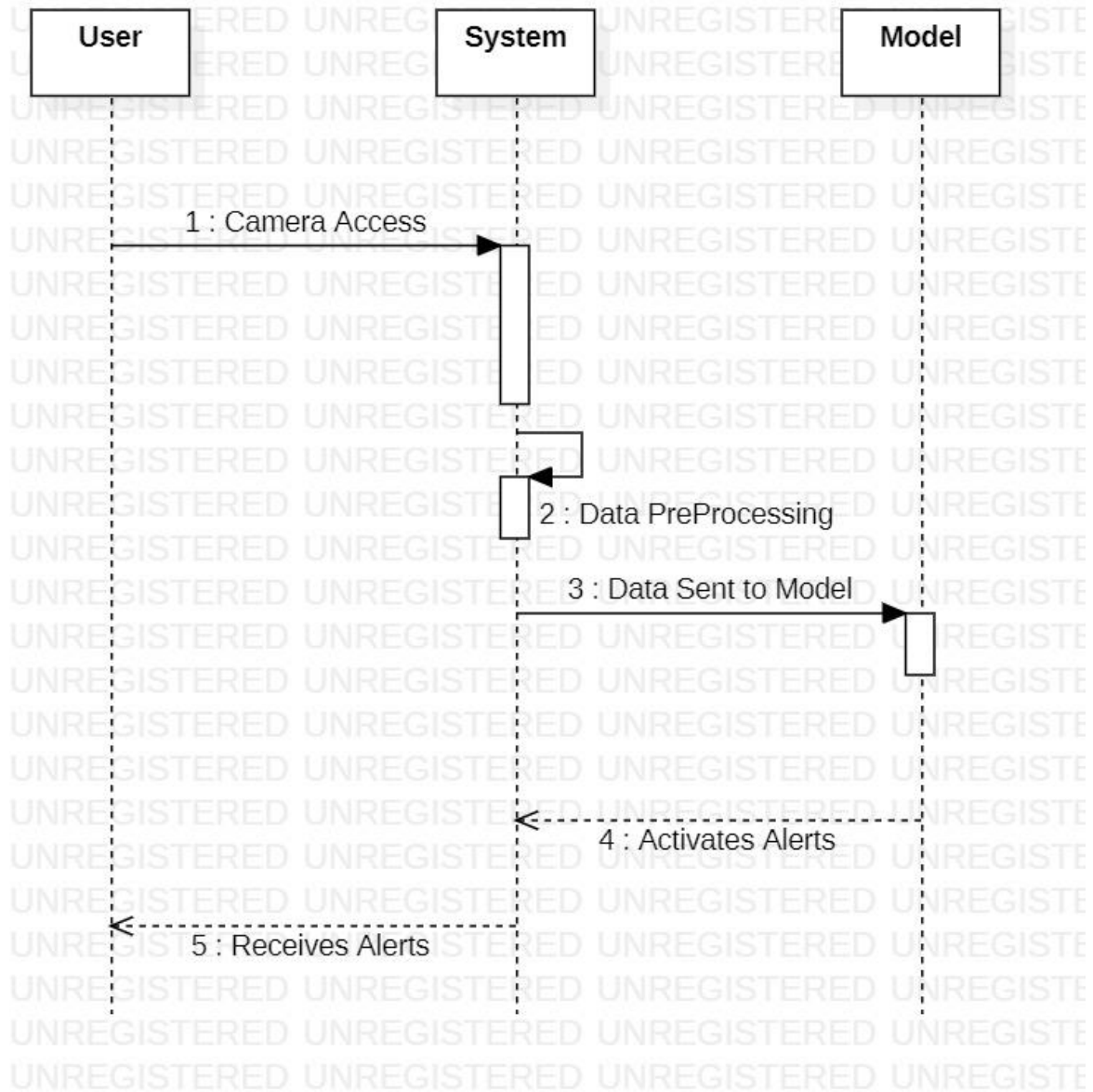


Fig4.3 : Sequence Diagram for Fire Detection and Alerting.

Activity Diagram:

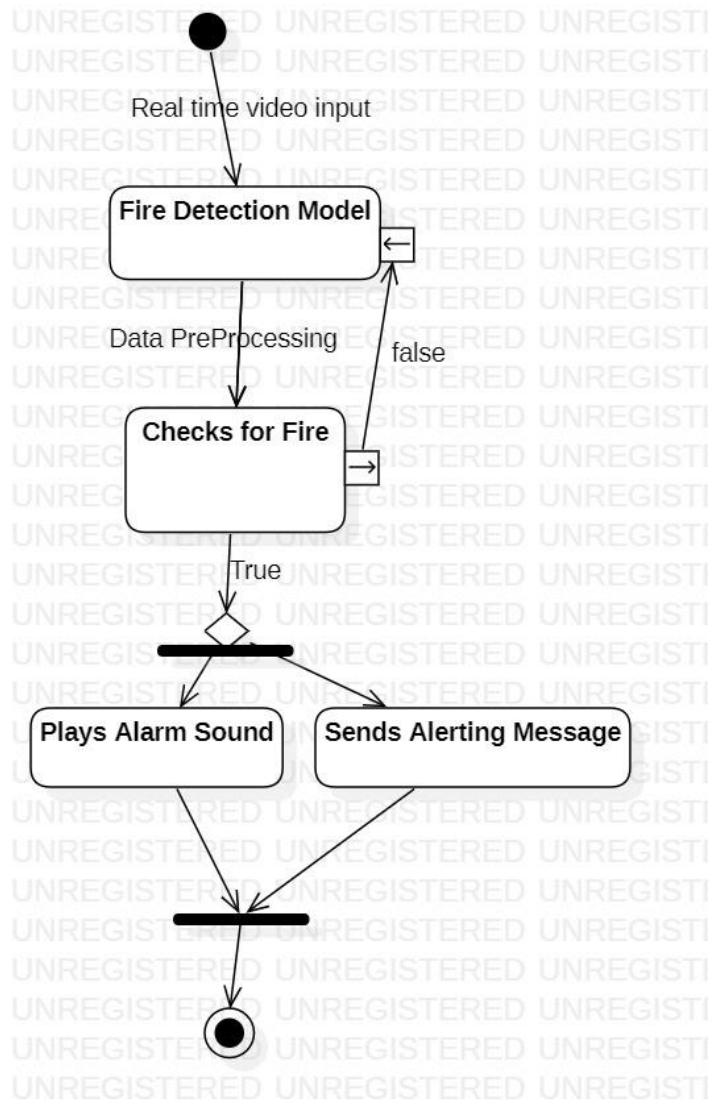


Fig4.4 : Activity Diagram for Fire Detection and Alerting.

CHAPTER-5

Implementation

This application contains code that will detect fire using live feed from a camera. The portion of the image with fire will be highlighted inside a rectangle in blue. We have used HAAR cascade classifier to classify the images as positives and negatives. Classifier has been trained using OpenCV on command line on a Windows 11 machine.

Training Steps to Create a Haar-Cascade Classifier by taking an example of “Face Detection”:

- Collection of positive and negative training images
- Marking positive images using objectmarker.exe or ImageClipper tools
- Creating a.vec (vector) file based on positive marked images using createsamples.exe
- Training the classifier using haartraining.exe
- Running the classifier using cvHaarDetectObjects()

STEP 1: Collecting Image Database

All the students will receive positive and negative sample images for training. You may like to add more positive and negative images by recording some sequences in more public images from Internet resources.

The positive images are those images that contain the object (e.g. face or eye), and negatives are those ones which do not contain the object.

Having more number of positive and negative (back ground) images will normally cause a more accurate classifier.

STEP 2: Arranging Negative Images

Put your background images in folder ...\\training\\negative and run the batch file

```
create_list.bat
```

```
dir /b *.jpg >bg.txt
```

Running this batch file, you will get a text file each line looks as below:

```
image1200.j  
pg  
  
image1201.j  
pg  
image1202.j  
pg  
  
...
```

Later, we need this negative data file for training the classifier.

STEP 3: Crop & Mark Positive Images

In this step you need to create a data file (vector file) that contains the names of positive images as well as the location of the objects in each image. You can create this file via two utilities: *Object marker* or *Image Clipper*.

The first one is simpler and faster, and the second one is a bit more versatile but more time consuming to work. We continue with *Object maker* which is straight forward; however, you may try *Image Clipper* later.

In folder `..\training\positive\rawdata` put you positive images

In folder `..\training\positive` there is a file `objectmaker.exe` that we need it for marking the objects in positive images. Note that in order to correctly run `objectmaker.exe` two files `cv.dll` and `highgui.dll` should also exist in the current directory.

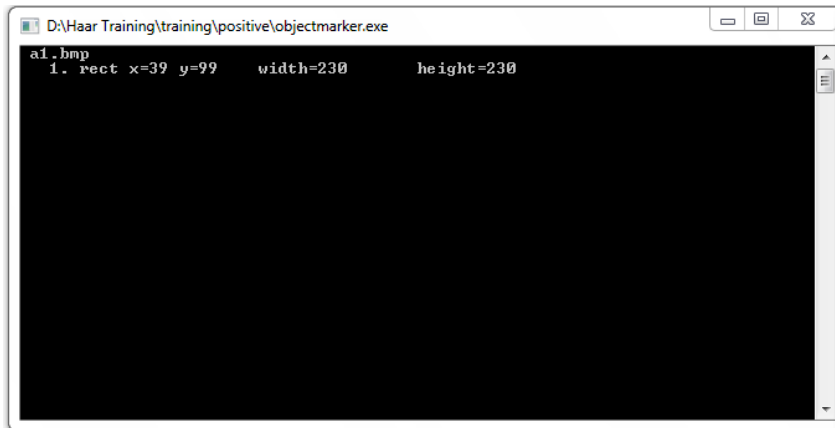
Before running the `objectmaker.exe` make sure you are relax and you have enough time to carefully mark and crop tens or hundreds of images!

How to mark objects? Running the file `objectmaker.exe` you will see two windows like below: one shows the loaded image, and the other one shows the image name.

- a- Click at the top left corner of the object area (e.g. face) and hold the mouse left-keydown.
- b- While keeping the left-key down, drag the mouse to the bottom right corner of the object. Now you could be able to see a rectangle that surrounds the object (see below). If you are not happy with your selection press any key (except Spacebar and Enter) to undo your selection, and try to draw another rectangle again.

Important note: Take care to always start the bounding box at either the top left or bottomright corner. If you use the other two corners `objectmaker.exe` will not write the coordinates of the selected object into the `info.txt` file.

- c- If you are happy with the selected rectangle, press *SPACE*. After that, the rectangle position and its size will appear on the left window (see below).



- d- Repeat steps “a” to “c” if there are multiple objects (e.g. faces) in the current .
e- When you finished with the current image, press *ENTER* to load the next image

Repeat steps “a” to “e” until the entire positive images load one by one, and finish.

If you feel tired and you want to stop it at the middle marking process, press

ESCAPE. a file named `info.txt` would be created.



: When you escaped `objectmaker` and you want to continue it later, create a backup for `info.txt` because every time you run `objectmarker.exe` it overwrites to previous `info.txt` without any notice, and it creates an empty new `info.txt`; i.e. you will lose you previous

works! Make a backup numbered file (e.g. `info1.txt`, `info2.txt` anytime you escape) and finally merge all your backups into a final `info.txt`

Within the `info.txt` there would be some information like below:

```
rawdata\image1200.bmp 1 34 12 74 24
rawdata\image1201.bmp 3 35 25 70 39 40 95 80 92 120 40 45 36
rawdata\image1202.bmp 2 10 24 90 90 45 68 99 82
```

The first number in each line defines the number of existing objects in the given image. For example, in second line, the number **3** means that you already selected three objects (e.g. face) within `image1201.bmp`. The next four numbers (shown in green) defines the location of first object in the image (top left vertex: `x=35, y=24`, `width=70` and `height=39`). The red numbers identifies the data for the second object; blues ones are for the third object, and so forth.



: Errors like this line: `rawdata/d19.bmp 3 83 119 185 183` can lead to a serious hidden issues while training your cascade. Before going further, make sure every line in `info.txt` is correct.

The error in above example is the number `3` while it should be `1`. This kind of error may happen when you merge `info.txt` files?

The next step is packing the object images into a vector-file.

STEP 4: Creating a vector of positive images

In folder `..\training\` there is a batch file named `samples_creation.bat`

The content of the bath file is:

```
createsamples.exe -info positive/info.txt -vec vector/facevector.vec -
num200 -w 24 -h 24
```

Main Parameters:

<code>-info positive/info.txt</code>	Path for positive info file
<code>-vec vector/facevector.vec</code>	Path for the output vector file
<code>-num 200</code>	Number of positive files to be packed in a <i>vector file</i>
<code>-w 24</code>	Width of objects
<code>-h 24</code>	Height of objects

The batch file loads `info.txt` and packs the object images into a vector file with the name of
e.g. `facevector.vec`

After running the batch file, you will have the file `facevector.vec` in the folder `..\training\vector`


Note: To run `creatsample.exe` you also needs the files `cv097.dll`, `cxcore097.dll`, `highgui097.dll`, and `libguide40.dll` in the folder `..\training`.

STEP 5: Haar-Training


In folder `..\training`, you can modify the `haartraining.bat` :

```
haartraining.exe -data cascades -vec vector/vector.vec -bg
negative/bg.txt
-npos 200 -nneg 200 -nstages 15 -mem 1024 -mode ALL -w 24 -h 24 -nonsym
```


-data cascades	Path and for storing the cascade of classifiers
-vec data/vector.vec	Path which points the location of vector file
-bg negative/bg.txt	Path which points to background file
-npos 200	Number of positive samples \leq no. positive bmp files
-nneg 200	Number of negative samples (patches) \geq npos
-nstages 15	Number of intended stages for training
-mem 1024	Quantity of memory assigned in MB
-mode ALL	Look literatures for more info about this parameter
-w 24 -h 24	Sample size
-nonsym	Use this if your subject is not horizontally symmetrical

 The size of -w and -h in haartraining.bat should be same as what you defined on sample-creation.bat

Haartraining.exe collects a new set of negative samples for each stage, and -nneg sets the limit for the size of the set. It uses the previous stages' information to determine which of the "candidate samples" are misclassified. Training ends when the ratio of misclassified samples to candidate samples is lower than FRstage10. So:

 Regardless of the number of stages (nstages) that you define in haartraining.bat, the program may terminate early if we reach above condition. Although this is normally a good sign of accuracy in our training process, however this also may happen when the number of positive images is not enough (e.g. less than 500).

Note: To run haartaining.exe you also needs the files cv097.dll, cxcore097.dll, and highgui097.dll in the folder ..\training.

While running the haartraning.bat you will see some information similar to screen below.

```

Parent node: 0
*** 1 cluster ***
POS: 200 200 1.000000
NEG: 200 0.243605
BACKGROUND PROCESSING TIME: 0.01
Precalculation time: 8.09
+-----+
| N |%SMP|F| ST.THR | HR | FA | EXP. ERR|
+-----+
| 1|100%|-0.915344| 1.000000| 1.000000| 0.067500|
+-----+
| 2|100%|+1.761648| 1.000000| 1.000000| 0.050000|
+-----+
| 3|100%|-1.040223| 1.000000| 0.325000| 0.027500|
+-----+
Stage training time: 4.79
Number of used features: 3
Parent node: 0
Chosen number of splits: 0
Total number of splits: 0
Tree Classifier
Stage
+-----+
| 0| 1|
+-----+

```

Data provided in Image above is related for the first stage of training Parent node:

Defines the current stage under training processN: Number of used features in this stage

%SMP: Sample Percentage (Percentage of sample used for this feature)

F: “+” if flipped (when symmetry applied) and “-“ if not

ST.THR: Stage Threshold

HR: Hit Rate based on the stage threshold

FA: False Alarm based on the stage threshold

EXP.ERR: Exponential Error of strong classifierNext figure is the data for the 10th stage of classifier

- It can be seen the number of features used in higher nodes are more than the earlier nodes
- The overall false detection (false alarm) has decreased
- and, the computational time for training has increased

```

Parent node: 9
*** 1 cluster ***
POS: 200 200 1.000000
NEG: 200 0.000574246
BACKGROUND PROCESSING TIME: 2.60
Precalculation time: 7.99
+-----+
| N | %SMP | F | ST.THR | HR | FA | EXP. ERR |
+-----+
| 1 | 100% | - | -0.554502 | 1.000000 | 1.000000 | 0.207500 |
+-----+
| 2 | 100% | + | -0.883580 | 1.000000 | 1.000000 | 0.180000 |
+-----+
| 3 | 100% | - | -1.647806 | 1.000000 | 1.000000 | 0.122500 |
+-----+
| 4 | 83% | + | -1.357607 | 1.000000 | 0.785000 | 0.095000 |
+-----+
| 5 | 91% | - | -1.956339 | 1.000000 | 0.810000 | 0.100000 |
+-----+
| 6 | 76% | + | -1.634170 | 1.000000 | 0.630000 | 0.055000 |
+-----+
| 7 | 73% | - | -1.235653 | 1.000000 | 0.435000 | 0.057500 |
+-----+
Stage training time: 10.40
Number of used features: 7
Parent node: 9
Chosen number of splits: 0
Total number of splits: 0
Tree Classifier
Stage
+-----+
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
+-----+

```

STEP 6: Creating the XML File

After finishing Haar-training step, in folder ../training/cascades/ you should have catalogues named from “0” upto “N-1” in which N is the number of stages you already defined in haartraining.bat.

In each of those catalogues there should be AdaBoostCARTHaarClassifier.txt file. Copy all the folders 0..N-1 into the folder ../cascade2xml/data/

Now we should combine all created stages (classifiers) into a single XML file which will be our final file a “cascade of Haar-like classifiers”.

Run the batch file `convert.bat` at `../cascade2xml/`
Which is:

```
haarconv.exe data myfacedetector.xml 24 24
```

`myfacedetecor.xml` is the output file name and 24 24 are W and H respectively.

Now you have your own XML file. Copy it in **MyCascade** folder, point to this classifier from your project source code, and run your face detection program.

In our code we are using Open cv Library, requests, playsound, threading modules to perform few specific actions at particular time. When the input real-time video is fetched to the model. Then it preprocesses the data and checks whether it contains fire or not, if the fire is found it starts the threads, our code consists of two threads i.e 1st thread `play_sound_function()` which is activated to `play_sound()` and 2nd `alert_message_function()` which is activated to send alert message to the user and the nearby fire stations.

5.1 SAMPLE CODE

```
import cv2

import threading

import playsound

import requests

#Loading the Trained XML file

fire_cascade = cv2.CascadeClassifier(rf"C:\Users\kashe_qkoaktc\OneDrive\Desktop\Mini
Project\fire_detection.xml")

#Taking input from Camera

vid = cv2.VideoCapture(0)

runOnce = False

def play_alarm_sound_function():

#Sound Alert File

playsound.playsound(rf"C:\Users\kashe_qkoaktc\OneDrive\Desktop\Mini
Project\audio.mp3",True)

print("Fire alarm end")

def alert_message_function():

querystring = {"apikey":"","number":"","text":"FIRE DETECTED AT PLACE"}


#Whatsapp API

url = https://panel.rapiwha.com/send\_message.php

response = requests.request("GET", url, params=querystring)

querystring = {"apikey":"","number":"","text":"SEND FIRE AT place"}

querystring1 = {"apikey":"","number":"","text":"SEND FIRE ENGINES AT place"}
```

```

response1 = requests.request("GET", url, params=querystring1)

print(response1.text)

print(response.text)

while(True):

    Alarm_Status = False

    #Reads the Input Video
    ret, frame = vid.read()

    #Converts the video to gray scale

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    #Detecting Fire in Video input

    fire = fire_cascade.detectMultiScale(frame, 2.0, 8)

    for (x,y,w,h) in fire:

        #Draws a Rectangle Boundary around detected 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 alarm initiated")

        #Alarm Sound function Starts when Fire is Detected

        threading.Thread(target=play_alarm_sound_function).start()

        if runOnce == False:

            print("Alert Initiated")

        #Alert Message Function Starts when Fire is Detected

        threading.Thread(target=alert_message_function).start()

        runOnce=True

```

```
if runOnce == True:

    print("Alert message already sent")

    runOnce =True

cv2.imshow("Fire Detector", frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

    break
```

CHAPTER – 6

TESTING

6.1 TEST CASES

Test Case to check whether the required Software is installed on the systems

Test Case ID:	1
Test Case Name:	Required Software Testing
Purpose:	To check whether the required Software is installed on the systems
Input:	Enter python command
Expected Result:	Should Display the version number for the python
Actual Result:	Displays python version
Failure	If the python environment is not installed, then the Deployment fails

Table 6.1.1 python Installation verification

Test Case to check Program Integration Testing

Test Case ID:	2
Test Case Name:	Python Programs Integration Testing
Purpose:	To ensure that all the modules work together
Input:	All the modules should be accessed.
Expected Result:	All the modules should be functioning properly.
Actual Result:	All the modules should be functioning properly.
Failure	If any module fails to function properly, the implementation fails.

Table 6.1.2 python Programs Integration Testing

Test Case to Load the Trained Dataset

Test Case ID:	3
Test Case Name:	Load the Trained Dataset
Purpose:	Check Dataset is collected, and the data is stored
Input:	Provide Dataset as input
Expected Result:	View the Dataset and store the Dataset
Actual Result:	Load the Dataset and view the Dataset and store
Failure	If the dataset is not loaded, it will throw an error.

Table 6.1.3 Collect Dataset and Load the Dataset

Test Case to check whether the fire is detected

Test Case ID:	4
Test Case Name:	Fire Detection
Purpose:	Fire Detection using Haar Cascade
Input:	Real-Time Video
Expected Result:	Plays Alarm, Sends Message to mentioned number
Actual Result:	If Fire is detected(Plays Alarm, Sends Message to mentioned number)
Failure	If the data is not Evaluated, it does not give expected result and application still runs.

Table 6.1.4 Fire Detection

Test Case to check whether the alarm message is played

Test Case ID:	5
Test Case Name:	Checks alarm message
Purpose:	Checks for whether the alarm message is played
Input:	Real-Time Video
Expected Result:	Plays Alarm Message
Actual Result:	If Fire is detected(Plays Alarm Message)
Failure	If the data is not Evaluated, it does not give expected result and application still runs.

Table 6.1.5 Checks for whether the alarm message is played**Test Case to check whether the alert message is sent**

Test Case ID:	6
Test Case Name:	Checks alert message
Purpose:	Check whether the alert message is sent
Input:	Real-Time Video
Expected Result:	sends Alert Message
Actual Result:	If Fire is detected(Sends Alert Message)
Failure	If the data is not Evaluated, it does not give expected result and application still runs.

Table 6.1.6 check whether the alert message is sent

CHAPTER - 7
SCREENSHOTS



Figure 7.1: Small Scaler Fire Detection

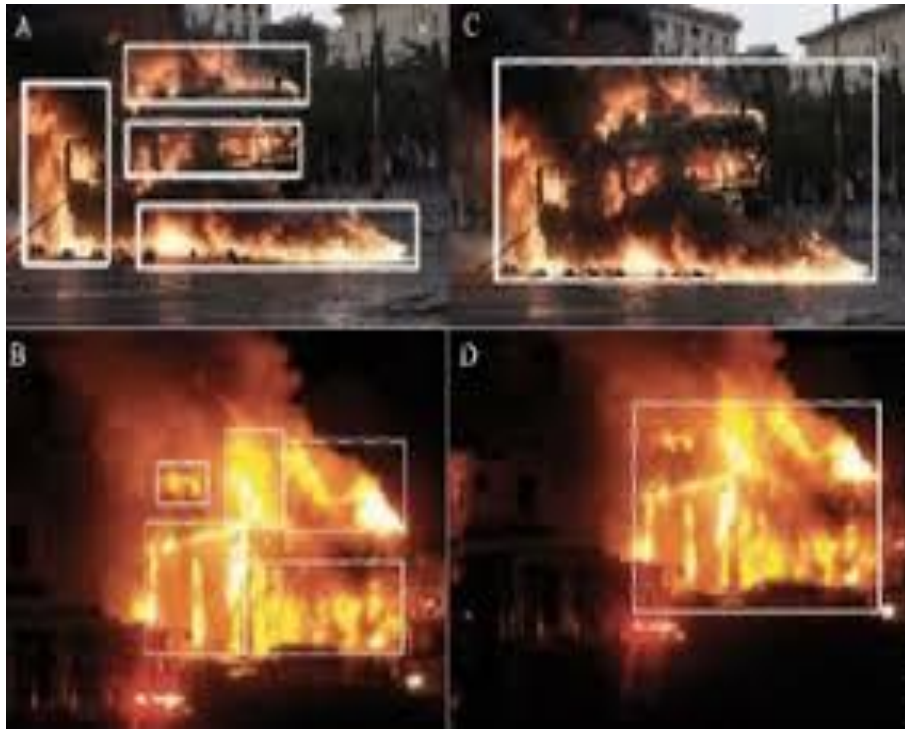


Figure 7.2: Fire Detection in Large Scale



Figure 7.3: Fire Detection in Daily Life

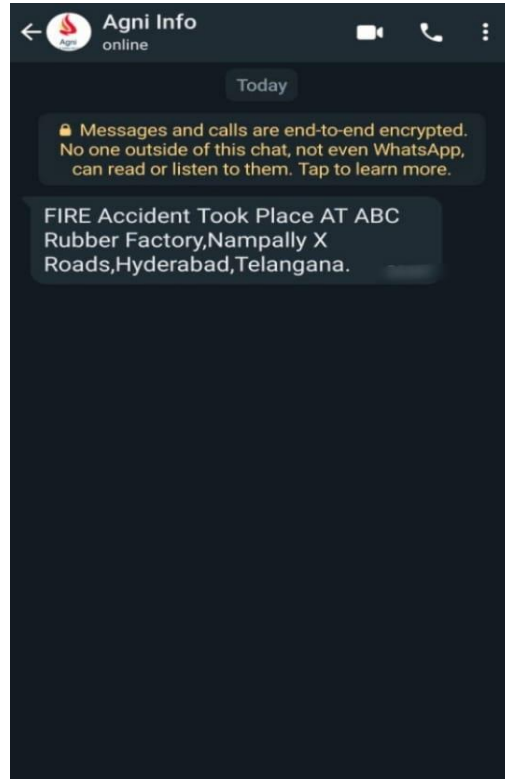


Figure 7.4: Fire Accident Alert Message received by Customer

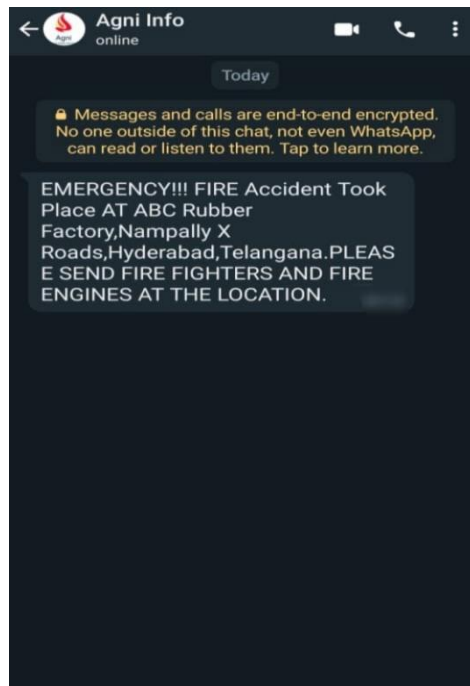


Figure 7.5: Fire Accident Alert Message received by Nearby Fire Station

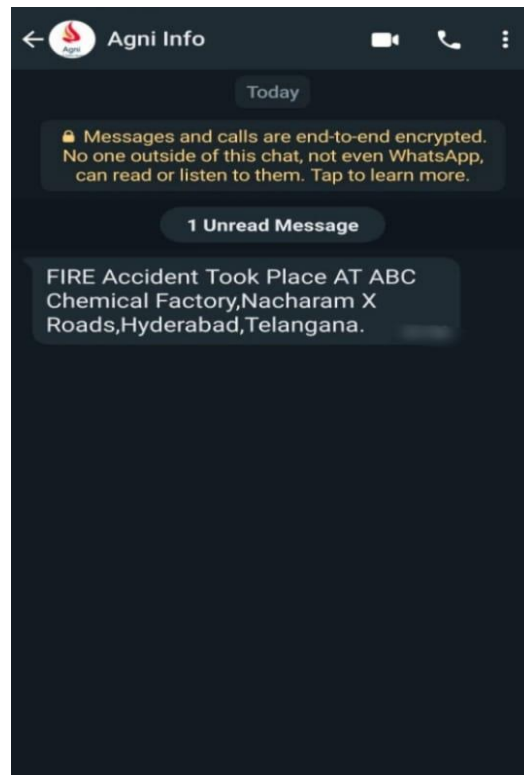


Figure 7.6: Fire Accident Alert Message received by Customer

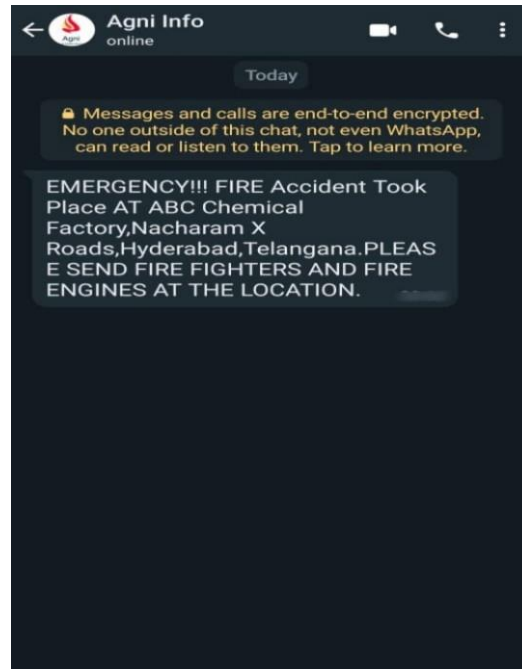


Figure 7.7: Fire Accident Alert Message received by Nearby Fire Station

CHAPTER - 8

CONCLUSION AND FUTURE SCOPE

The main idea behind the fire detection and alerting system is to reduce the damage and Causalities cause by the fire accidents, this project is proposed in such a way that the fire is detected at the earliest and alerts the victims to evacuate the place, also sends message to the security chief or the Property Owner and also sends alert messages to the near by fire stations.

This projects model is trained by using the HAAR CASCADE CLASSIFIER algorithm even though there are new object detection algorithms as this algorithm has faster reaction time when compared to other algorithms. This model gets the input from the camera in real-time and if a fire accident occurs it detects it. The playsound module is used to play the alerting sound and requests module is used to interact with the messaging API "RAPIWHA".

In future the model can be modified if a new emerging algorithm is emerged then that can be used to train the model only if, its reaction time is less when compared to the HAAR CASCADE algorithm. This model can be combined with other models to develop the organization.

BIBLIOGRAPHY

- [1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2274–2282, nov 2012.
- [2] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *Computer Vision–ECCV 2008*, pages 102–115. Springer, 2008.
- [3] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. Face recognition with local binary patterns. In *Computer vision-eccv 2004*, pages 469–481. Springer, 2004.
- [4] Cuneyt Akinlar and Cihan Topal. Edlines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 32(13):1633–1642, 2011.
- [5] Cuneyt Akinlar and Cihan Topal. Edpf: a real-time parameter-free edge segment detector with a false detection control. *International Journal of Pattern Recognition and Artificial Intelligence*, 26(01):1255002, 2012.
- [6] Cuneyt Akinlar and Cihan Topal. Edcircles: A real-time circle detector with a false detection control. *Pattern Recognition*, 46(3):725–740, 2013.
- [7] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. IEEE, 2012.

APPENDIX A: TOOLS AND TECHNOLOGIES

- **PYTHON V3** : Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.
- **JUPYTER NOTEBOOK** : Jupyter Notebook can colloquially refer to two different concepts, either the user facing application to edit code and text, or the underlying file format which is interoperable across many implementations. Jupyter Notebook (formerly IPython Notebook) is a web-based interactive computational environment for creating notebook documents. Jupyter Notebook is built using several open-source libraries, including IPython, ZeroMQ, Tornado, jQuery, Bootstrap, and MathJax.
- **REQUEST MODULE** : The requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc). Requests is one of the most downloaded Python packages today, pulling in around 30M downloads / week— according to GitHub, Requests is currently depended upon by 1,000,000+ repositories. You may certainly put your trust in this code.
- **NUMPY** : NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project.

- **WINDOWS 11** : Windows 11 is the latest major release of Microsoft's Windows NT operating system, released in October 2021. It is a free upgrade to its predecessor, Windows 10 (2015), and is available for any Windows 10 devices that meet the new Windows 11 system requirements. Windows 11 features major changes to the Windows shell influenced by the canceled Windows 10X, including a redesigned Start menu, the replacement of its "live tiles" with a separate "Widgets" panel on the taskbar, the ability to create tiled sets of windows that can be minimized and restored from the taskbar as a group, and new gaming technologies inherited from Xbox Series X and Series S such as Auto HDR and DirectStorage on compatible hardware.
- **THREADING MODULE** : The newer threading module included with Python 2.4 provides much more powerful, high-level support for threads than the thread module discussed in the previous section. The threading module exposes all the methods of the thread module and provides some additional methods – `threading.activeCount()` – Returns the number of thread objects that are active. `threading.currentThread()` – Returns the number of thread objects in the caller's thread control. `threading.enumerate()` – Returns a list of all thread objects that are currently active.
- **HAAR CASCADE** : Haar-like features are digital image features used in object recognition. They owe their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector. Historically, working with only image intensities (i.e., the RGB pixel values at each and every pixel of image) made the task of feature calculation computationally expensive. A publication by Papageorgiou et al. discussed working with an alternate feature set based on Haar wavelets instead of the usual image intensities. Paul Viola and Michael Jones adapted the idea of using Haar wavelets and developed the so-called Haar-like features.
- **OPEN CV** : OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations. OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though older C interface.