VISION BASED FIRE DETECTION SYSTEM

A course project in the domain of machine vision for the course Machine Vision and Robotics (IC3024)

UNDERTAKEN BY

ANAYA PAWAR

(T. Y. Q 5, G. R. 1710263)

ANEESH PODUVAL

(T. Y. Q 6, G. R. 1710045)

SARTHAK CHUDGAR

(T. Y. Q 16, G. R. 1710202)

JOHNATHAN FERNANDES

(T. Y. Q 37, G. R. 1710168)

UNDER THE GUIDANCE OF PROF. PRAMOD KANJALKAR

TABLE OF CONTENTS

PREFACE

In recent times, there have been several instances of fires all across the world. The collateral damage caused by these incidents negatively impact humans as well as nature. Due to this, the need for application for fire detection has increased in recent years.

Traditional fire detection systems utilize **light**, **heat** and **gas** sensors, each with their own strengths and drawbacks. However, there are several situations where these systems are impractical to implement, either due to cost, power or other practical reasons.

CASE STUDY

NOTRE DAME FIRE

The Notre-Dame is an 850-year old cathedral located in Paris, France. Renowned for its architectural design and stained glass windows, it is one of the most widely recognized structures of France and attracts over 12 million people annually.

Figure 1: Notre-Dame Cathedral

On 15th April 2019, during the process of renovation, the cathedral caught fire, destroying most of the structure and the iconic spire. Upon further investigation, it was revealed that the smoke detector system in the buildings alerted the authorities of the fire, but they failed to pinpoint the location of the fire until it was too late.

Figure 2: Fire ravaging the cathedral

If the fire detection system was paired with some sort of vision based system, the location of the fire could have been discerned earlier and suitable measures could have been taken.

AUSTRALIA WILDFIRES

Australia is a country which is known for regular bushfires, to the point where they have a recognized “fire season” during summers, with hot & dry weather making it easy for fire to spread. The causes for these fires are mostly natural, but there have been cases of human causes as well.

Figure 3: Ongoing forest fires in Australia

The most recent case entails a series of fires which started in June 2019 are as of March 2020, are still ongoing. This fire has resulted in approximately 18 million hectares of land, destroying over 8 thousand buildings, causing 34 known human fatalities and many more animal deaths.

Due to the nature of forests, it is impractical to implement traditional fire detection systems due to power, range and maintenance issues. Even in today's day and age, we rely on human watchtowers to locate and report forest fires. This entire system could be automated using drone mounted camera based fire detection systems.

PROPOSED SETUP (PROJECT OBJECTIVE)

Through this course research project, we aim to construct a **fire detection system** based on **image processing techniques** which can be implemented in existing surveillance devices like CCTV, wireless camera and UAVs.

SOFTWARE TECHNOLOGY

We have chosen to implement our fire detection algorithm using the **python** programming language. The **Open Computer Vision** library is utilized as it provides straightforward and convenient methods for all the procedures in our algorithm. We generate a **Custom HAAR Cascade Classifier** trained using 1600 images, obtained from **ImageNet**.

HAAR CASCADE CLASSIFIERS

Initially proposed by Paul Viola and improved upon by Rainer Lienhart, a **classifier** is a system which is trained with multiple sample images of a particular object (referred to as "**positive**" images) and random arbitrary images which do not contain the specific object ("**negative**" images).

After training, a classifier can be applied to a **region of interest** (an image or part thereof) to recognize and detect the sample object. Classifiers are designed such that they can be scaled in size to cover multiple parts of the test image, as it is more efficient than resizing the target image.

"**Haar-like features**" are simply the features extracted from images. A Haar feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. The first real-time face detector utilized Haar Wavelets and the name stuck to this day.

"Cascade" refers to the fact that the classifier consists of multiple simpler classifiers which detect haar-like features and are "stacked" or "cascaded" (**applied subsequently**) to the region of interest.

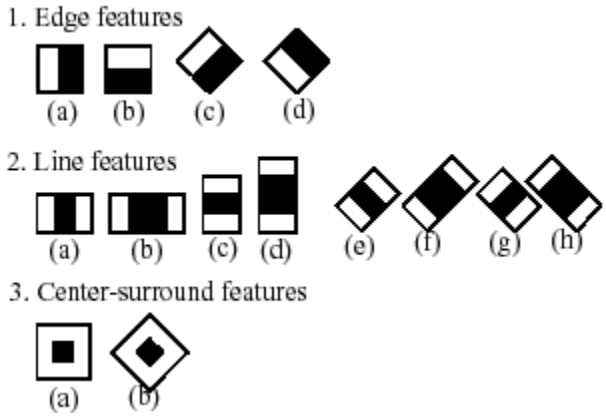

Figure 4: Simple Haar-like features cascaded in the algorithm

For our application, we specifically apply the **CascadeClassifier::detectMultiScale(image,scaleFactor, minNeighbors)** function, which detects multiple objects of varying sizes and returns their positions as a list of rectangles.

It accepts three parameters which are:

- image: Matrix containing an image where objects are detected.
- scaleFactor: Parameter specifying how much the image size is reduced at each image scale.

- minNeighbors: Parameter specifying how many neighbors each candidate rectangle should have to retain it. This parameter will affect the quality of the detected faces: higher value results in less detections but with higher quality.

ALGORITHM

The first step is to obtain our training images. We use ImageNet to download 1000 "**positive**" images (images containing fire) and 1700 "**negative**" images (images not containing fire).

The next step is to generate a **Custom Haar Cascade Classifier**. This process extracts features from positive and negative images and creates an **XML** file, which is used later in the process. Although this can be done in the command line interface with **Open Computer Vision**, ding so is extremely tedious and hence we employ a program known as **Cascade Trainer GUI** to construct the file.

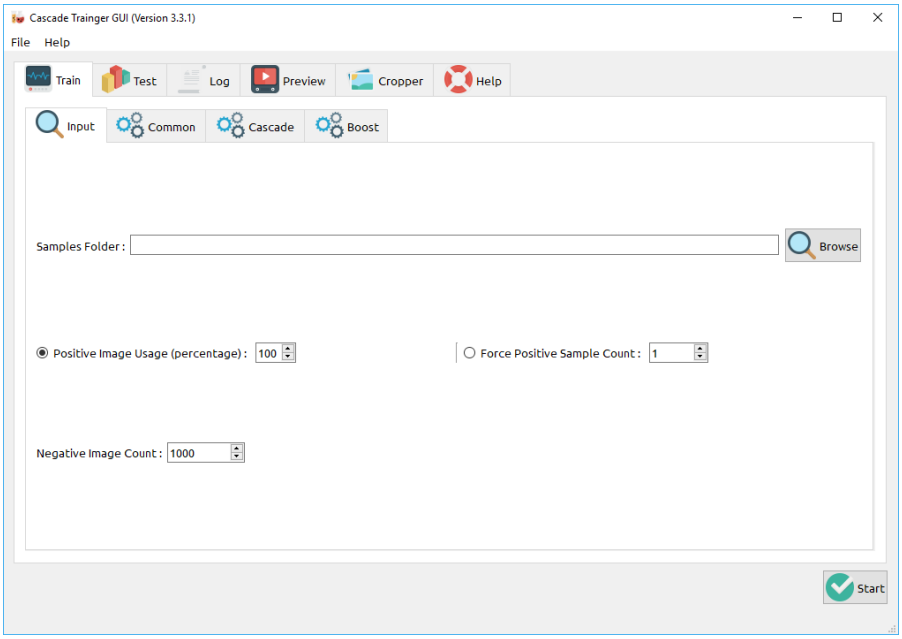

Figure 5: Cascade Trainer GUI

Meanwhile, we also create a python program to accept **live video feed** from an attached camera. Built using Open Computer Vision functions, this script will be able to execute operations on **individual frames** while still delivering **real time** results.

After obtaining the Custom Haar Cascade Classifier file, we integrate it into our python program. The module compares each frame of the video using our classifier, and when a fire is detected, a variety of **alarms** can be triggered, ranging from visual to audio.

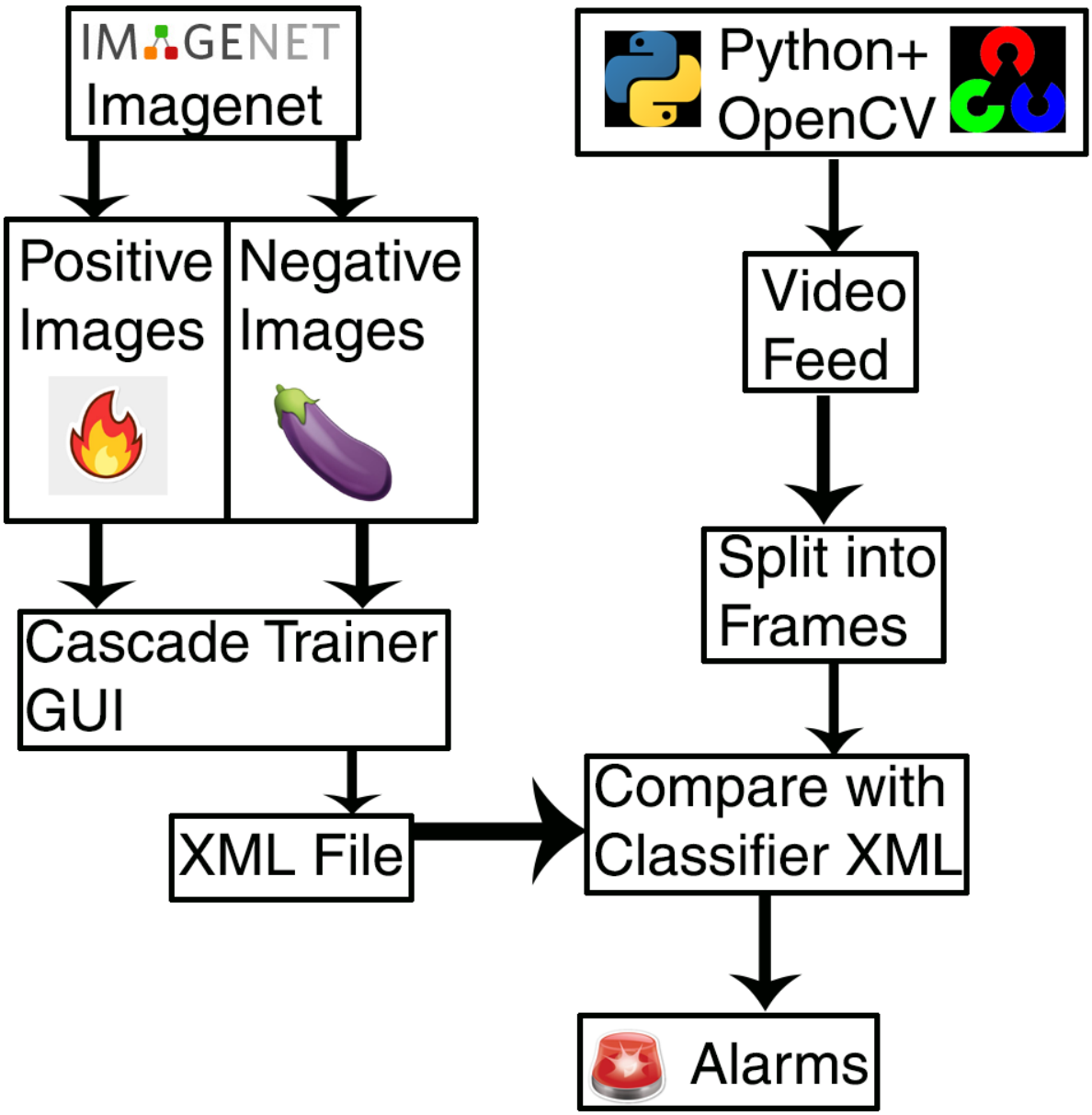

Figure 6: Block diagram of fire detection system

This results in a very robust model able to **detect fire and display its position** on the image display.

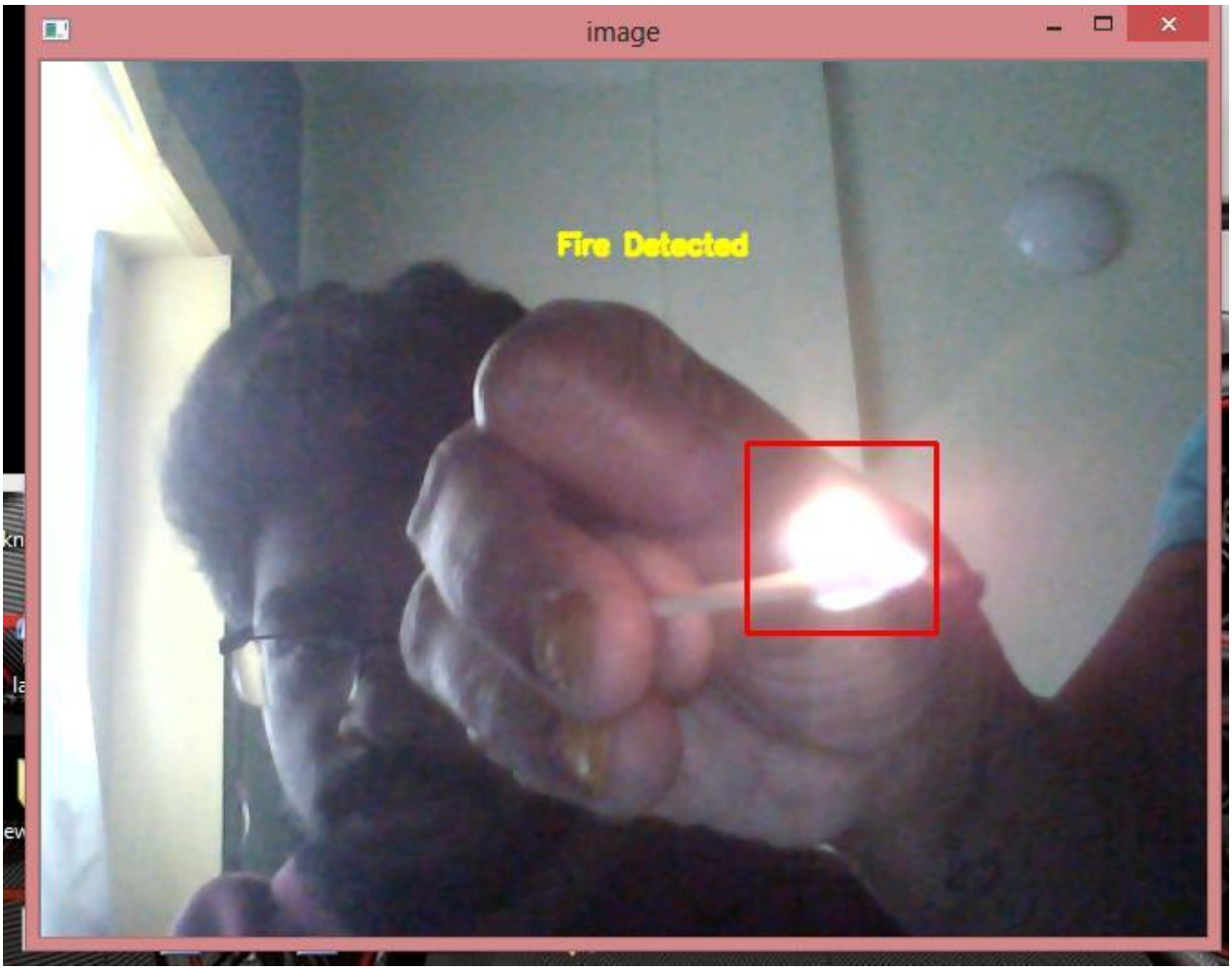

Figure 7: Visual output on system

ACCURACY CALCULATION

We applied the standard image processing testing procedure using **evaluation parameters** to determine the accuracy of our project.

$$\text{Accuracy} = \frac{TP + TN}{Ti}$$

Equation 1: Accuracy Calculation

Where:

- **TP** is number of true positive test results (known subjects correctly identified)
- **TN** is number of true negative test results (unknown subjects not recognized)
- **Ti** is total number of test samples

Using the above test, we determined the **accuracy of the system to be 85%.**

This accuracy can further be improved by including more training images.

FUTURE SCOPE

The current system can be integrated into existing camera setups, leading to a **virtually free** fire detection system in setups already utilizing camera systems

We can also mobilize the system by interfacing it with mobile cameras such as cars and drones.

This system can also be improved upon by integrating various other existing fire detection methods such as smoke, heat and light sensors along with a suitable alarm system to notify the authorities.

CONCLUSION

Working on this course project has given us the opportunity to apply knowledge gained from our course into something tangible which can be utilized further in our careers.

ACKNOWLEDGEMENT

This project would not have been possible without our professor, who constantly offered us plenty of suggestions and never hesitated to point out when there was room for expansion and improvement.

BIBLIOGRAPHY

B.C. Ko, K.H. Chong, J.Y. Nam (2009): Fire Detection based on vision Sensor and Support vector services Fire Safety Journal, vol. 44, pp. 322–329.

Dong Keun Kim," Smoke Detection using Boundary Growing and Moments", International Conference on Convergence and Hybrid Information Technology 2009.

Vipin V," Image processing based fire detection," International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 2, February 2012).

W. Phillips III, M. Shah, and N. V. Lobo, "Flame Recognition in Video," Pattern Recognition Letters, v.23 (1-3),

APPENDIX

FIGURES

EQUATIONS

CODE

```
#Machine Vision and Robotics Course Project

#Fire detection using Image Processing

#import modules

import cv2#Computer vision module for video capture and image processing

import numpy as np#Python numbers module for matrix manipulation

import time#time module to measure time for pauses

import winsound# windows sound module for alarm

v=cv2.VideoCapture(0)#Capture live video using camera 0 (default webcam) and store values in v

time.sleep(2)#pause for 2 seconds

fdc=cv2.CascadeClassifier(r'C:\python\python36\fire_detection.xml')#Load generated cascade classifier into fdc

frequency = 2500#Define alarm frequency as 2500 Hz
```

```
duration = 500#Define alarm duration as 500 ms

while(1):#Loop code continously

    d,i=v.read()#Get frame from video capture. If frame is obtained, store TRUE in d and store image matrix in i

    print(i.shape)#Print image dimensions in console

    fire=fdc.detectMultiScale(i,1.3,9)#Use detectMultiScale function to detect objects, passing image,scale factor and minNeighbors as parameters

    print(fire)#Print dimensions of detected object (fire) matrix in console


    if(len(fire)>=1):#If 1 or more objects (fires) are detected

     for x,y,w,h in fire:#Store coordinates of upper-left corner in x and y, and widht and height in w and h

        font=cv2.FONT_HERSHEY_SIMPLEX#Define font

        cv2.putText(i,'FIRE DETECTED',(x-w,y-h),font,0.5,(0,255,255),2,cv2.LINE_AA)#Display "FIRE DETECTED" test

        cv2.rectangle(i,(x,y),(x+w,y+h),(0,0,255),2)#Display rectangle around fire

        time.sleep(0.2)#Pause of  for 0.2 seconds

        winsound.Beep(frequency, duration)#Sound alarm

    cv2.imshow('image',i)#Display processed image on screen


    if(len(fire)>=1):#If 1 or more objects (fires) are detected


      print('FIRE DETECTED')#Print "FIRE DETECTED" to console

    k=cv2.waitKey(5)#If keypress is detected, wait 5 seconds

    if(k==ord('q')):#If detected key is q

        cv2.destroyAllWindows()#Stop all processing
```

```
break#Exit loop
```