# Fire Detection With Python | Computer Vision

[[Kazi Mushfiqur Rahman](https://medium.com/@KaziMushfiq1234?source=post_page-----e55c8fc6fa54--------------------------------)](https://medium.com/@KaziMushfiq1234?source=post_page-----e55c8fc6fa54--------------------------------)

[Kazi Mushfiqur Rahman](https://medium.com/@KaziMushfiq1234?source=post_page-----e55c8fc6fa54--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F16ea6542ad64&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40KaziMushfiq1234%2Ffire-detection-with-python-computer-vision-e55c8fc6fa54&user=Kazi+Mushfiqur+Rahman&userId=16ea6542ad64&source=post_page-16ea6542ad64----e55c8fc6fa54---------------------post_header-----------)

5 min read

·

Aug 28

1



Photo by [Aishwarya Gunde](https://unsplash.com/@aishwaryagunde?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral" \t "_blank)

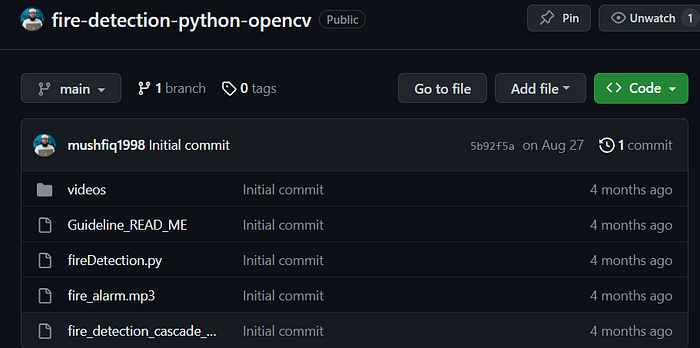
Complete project in GitHub

## [GitHub - mushfiq1998/fire-detection-python-opencv](https://github.com/mushfiq1998/fire-detection-python-opencv?source=post_page-----e55c8fc6fa54--------------------------------" \t "_blank)

### [Contribute to mushfiq1998/fire-detection-python-opencv development by creating an account on GitHub.](https://github.com/mushfiq1998/fire-detection-python-opencv?source=post_page-----e55c8fc6fa54--------------------------------" \t "_blank)

[github.com](https://github.com/mushfiq1998/fire-detection-python-opencv?source=post_page-----e55c8fc6fa54--------------------------------" \t "_blank)

**Project structure**



**fireDetection.py**

import cv2 # Library for openCV  
import threading # Library for threading -- which allows code to run in backend  
import playsound # Library for alarm sound  
import smtplib # Library for email sending  
  
# To access xml file which includes positive and negative images of fire.   
# (Trained images) File is also provided with the code.  
fire\_cascade = cv2.CascadeClassifier('fire\_detection\_cascade\_model.xml')   
  
vid = cv2.VideoCapture("videos\\fire2.mp4")  
runOnce = False # created boolean  
  
# defined function to play alarm post fire detection using threading  
def play\_alarm\_sound\_function():   
 # to play alarm # mp3 audio file is also provided with the code.  
 playsound.playsound('fire\_alarm.mp3',True)   
 print("Fire alarm end") # to print in consol  
  
# Defined function to send mail post fire detection using threading  
def send\_mail\_function():   
   
 recipientmail = "add recipients mail" # recipients mail  
 recipientmail = recipientmail.lower() # To lower case mail  
   
 try:  
 server = smtplib.SMTP('smtp.gmail.com', 587)  
 server.ehlo()  
 server.starttls()  
 # Senders mail ID and password  
 server.login("add senders mail", 'add senders password')   
 # recipients mail with mail message  
 server.sendmail('add recipients mail', recipientmail, "Warning fire accident has been reported")   
 # to print in consol to whome mail is sent  
 print("Alert mail sent sucesfully to {}".format(recipientmail))  
 server.close() ## To close server  
   
 except Exception as e:  
 print(e) # To print error if any  
   
while(True):  
 Alarm\_Status = False  
 # Value in ret is True # To read video frame  
 ret, frame = vid.read()   
 # To convert frame into gray color  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)   
 # to provide frame resolution  
 fire = fire\_cascade.detectMultiScale(frame, 1.2, 5)   
  
 ## to highlight fire with square   
 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 alarm initiated")  
 # To call alarm thread  
 threading.Thread(target=play\_alarm\_sound\_function).start()   
  
 if runOnce == False:  
 print("Mail send initiated")  
 # To call alarm thread  
 threading.Thread(target=send\_mail\_function).start()   
 runOnce = True  
 if runOnce == True:  
 print("Mail is already sent once")  
 runOnce = True  
  
 cv2.imshow('frame', frame)  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break

To run the project complete the following project

Create a virtual environment

python -m venv myenv



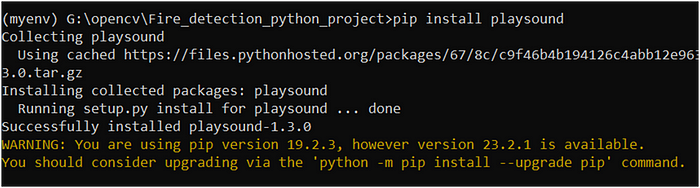
Activate the virtual environment

myenv\Scripts\activate

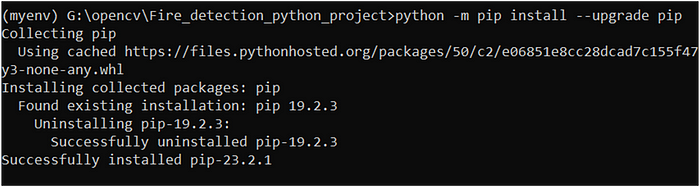


Install playsound

pip install playsound

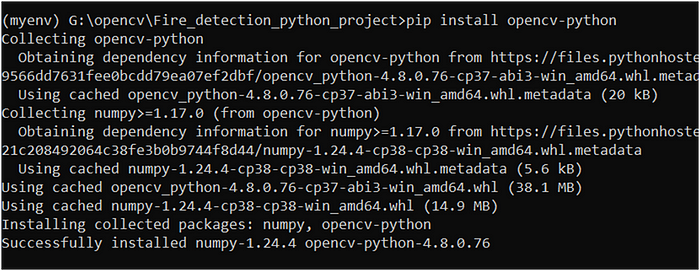


Upgrade pip

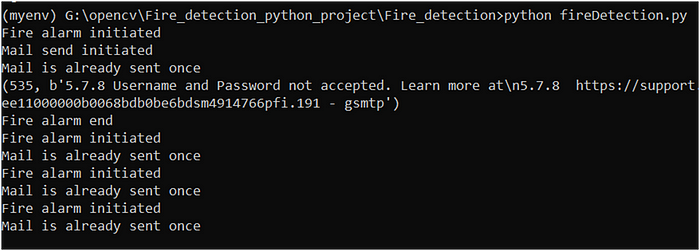


Install opencv

pip install opencv-python



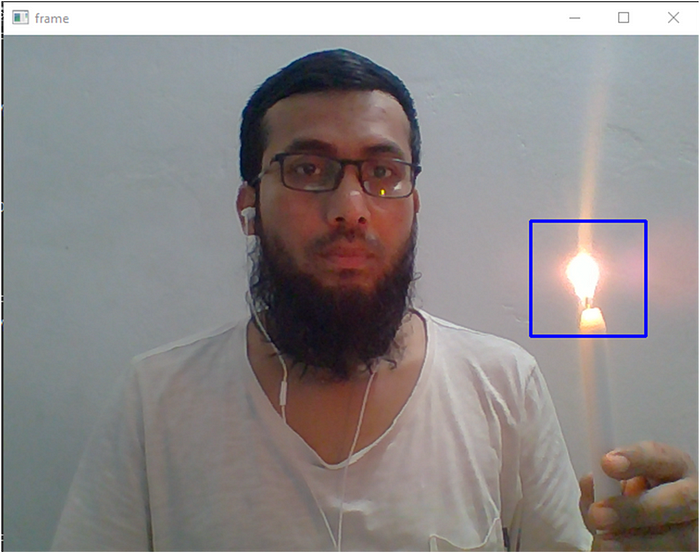
Execute the file **fireDetection.py**





Now, we will capture video from webcam

In the following screen we can see that the system is detecting fire from candle light drawing bounding box



Now, we will show fire detection from videos

Input video for fire detection

Input video for fire detection

Our system detects fire from the above videos drawing bunding box with alarm

Explanation of code in **fireDetection.py** file



**Explanation of the above code**

This Python code is a simple example of a fire detection system using OpenCV, threading, sound, and email functionalities. Here’s a breakdown of what it does in simpler terms:

1. Importing Libraries: The code starts by importing necessary libraries:

* cv2: Used for working with image and video processing, particularly for detecting the fire.
* threading: Used to run certain parts of the code concurrently (in the background).
* playsound: Used to play an alarm sound.
* smtplib: Used to send emails.

2. Loading Trained Model: The code loads a pre-trained machine learning model (XML file) that can detect fires in images.

3. Setting Up Video Source: It sets up the source of video input, which can be either the laptop’s built-in camera or an external USB camera. The code is currently configured to read video from a file named “fire2.mp4”.

4. Playing Alarm Sound: A function play\_alarm\_sound\_function() is defined to play an alarm sound. This function runs in the background (threading) and plays the alarm sound file called "fire\_alarm.mp3".

5. Sending Email: Another function send\_mail\_function() is defined to send an email. It uses Gmail's SMTP server to send a warning email about the fire detection to a specified recipient. The sender's email and password need to be provided in the code.

6. Main Loop: The main loop processes each frame of the video. It does the following:

* Converts the frame to grayscale for easier processing.
* Detects fires in the frame using the loaded model.
* If a fire is detected, it highlights the area with a blue rectangle.
* If a fire is detected for the first time (controlled by runOnce), it triggers the alarm sound and sends an email using threads. The alarm and email functions are run in the background.
* Once the alarm and email have been triggered once, the system doesn’t repeat this process for subsequent frames with fire.

7. Displaying Video: The code displays the processed frame with rectangles drawn around detected fires. The video is displayed until you press the ‘q’ key.

In simple terms, this code reads video frames, looks for fires in the frames, and if a fire is detected, it plays an alarm sound and sends an email alert. It uses separate threads for playing the alarm and sending the email, so these tasks don’t block the main video processing loop.

Please note that this is a basic example and might need adjustments and improvements for a complete and robust fire detection system. Additionally, the code contains placeholders for sender and recipient email addresses, which you would need to replace with actual email addresses.