INTRODUCTION:

MOTION DETECTION IN LIVE VIDEO STREAM

A surveillance systems should be a reflection of the real world we live in. As people become more and more security savvy, they will demand real protection for their property. The new digital video systems will have to raise that security to a new level. They should make the customers feel good. Scare off a few troublemakers. And those who do try to beat the system should face a far greater risk of getting caught. Hence, the new digital video surveillance systems should be able to provide a high sense of security. The peace of mind can only be achieved when the person is assured that he will be informed of any thefts of his property while they are in progress. He would also feel more secure if he can be guaranteed that the surveillance system that he uses will not only give him evidence against the perpetrators but also try to stop the thefts from taking place in the first place. Therefore, to achieve such kind of security Motion Detection in the live video stream is implemented. The motion detection systems will not only be monitoring the areas of interest but will also keep an active lookout for any motion being produced.

REQUIREMENT OF VIDEO SURVEILLANCE

While it is important to understand the various places video surveillance can be used it is also important to asses the risks involved in the protection of a certain item. In the recent years, as more and more items such as art are gaining importance, the prices of such things are also going through the roof. Therefore, technology has come in the forefront for protection and surveillance of such goods and items.

AIM:

In our project we have aimed to build such a surveillance system, which can not only detect motion, but will:

a) Warn the user of the intrusion through messages by using a rest API
b) Record the statistics related to the unidentified motion using Bokeh
c) Record the image of the unidentified person/face
d) Pair the entire system with an IP WEBCAM thereby making it economical and reducing the need for Hardware(Rpi etc)
e) Record the footage of the video from the moment the motion was detected.
f) A frontend app for easy deployment of the entire model

REFERENCES:

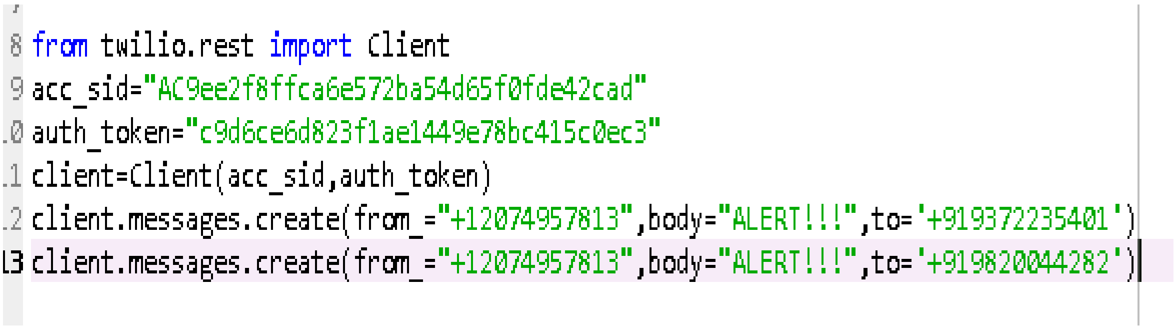
https://www.researchgate.net/publication/260714774

https://www.researchgate.net/publication/242733396

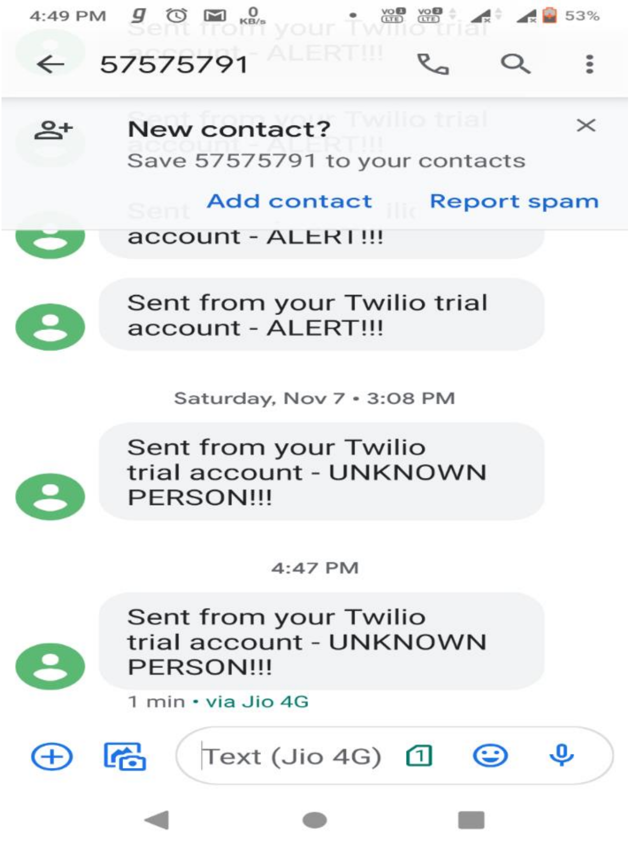
IMPLEMENTATION SPECIFICS AND CODE:

a)As soon as some motion is detected the message script portion of our code will be executed and will alert the security/watchmen/guard:

Here’s a snippet of the same:



```
from twilio.rest import Client
acc_sid="AC9ee2f8ffca6e572ba54d65f0fde42cad"
auth_token="c9d6ce6d823f1ae1449e78bc415c0ec3"
client=Client(acc_sid,auth_token)
client.messages.create(from_="+12074957813",body="ALERT!!!",to='+919372235401')
client.messages.create(from_="+12074957813",body="ALERT!!!",to='+919820044282')
```



account - ALERT!!!

Sent from your Twilio trial account - ALERT!!!

Saturday, Nov 7 • 3:08 PM

Sent from your Twilio trial account - UNKNOWN PERSON!!!

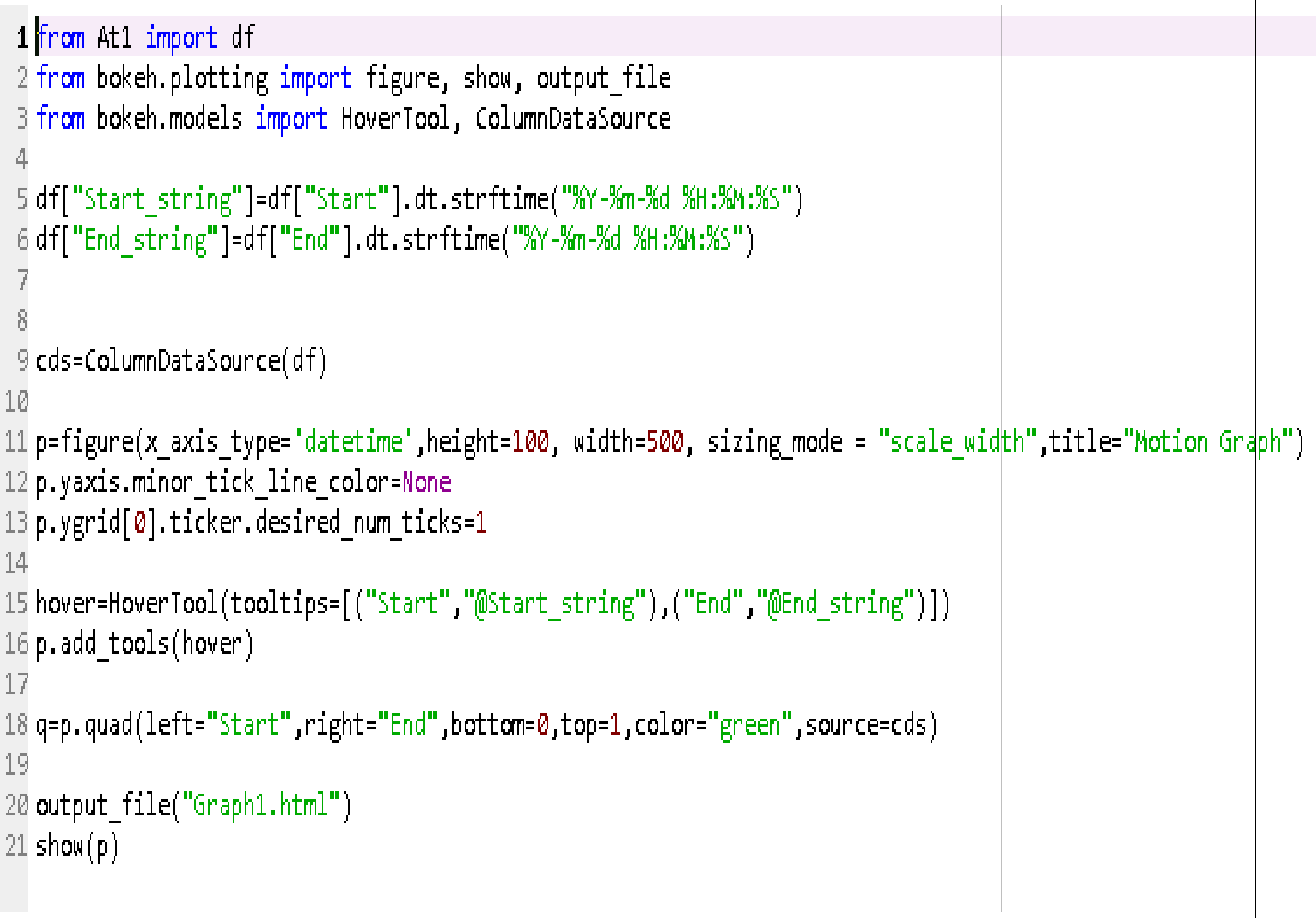
4:47 PM

Sent from your Twilio trial account - UNKNOWN PERSON!!!

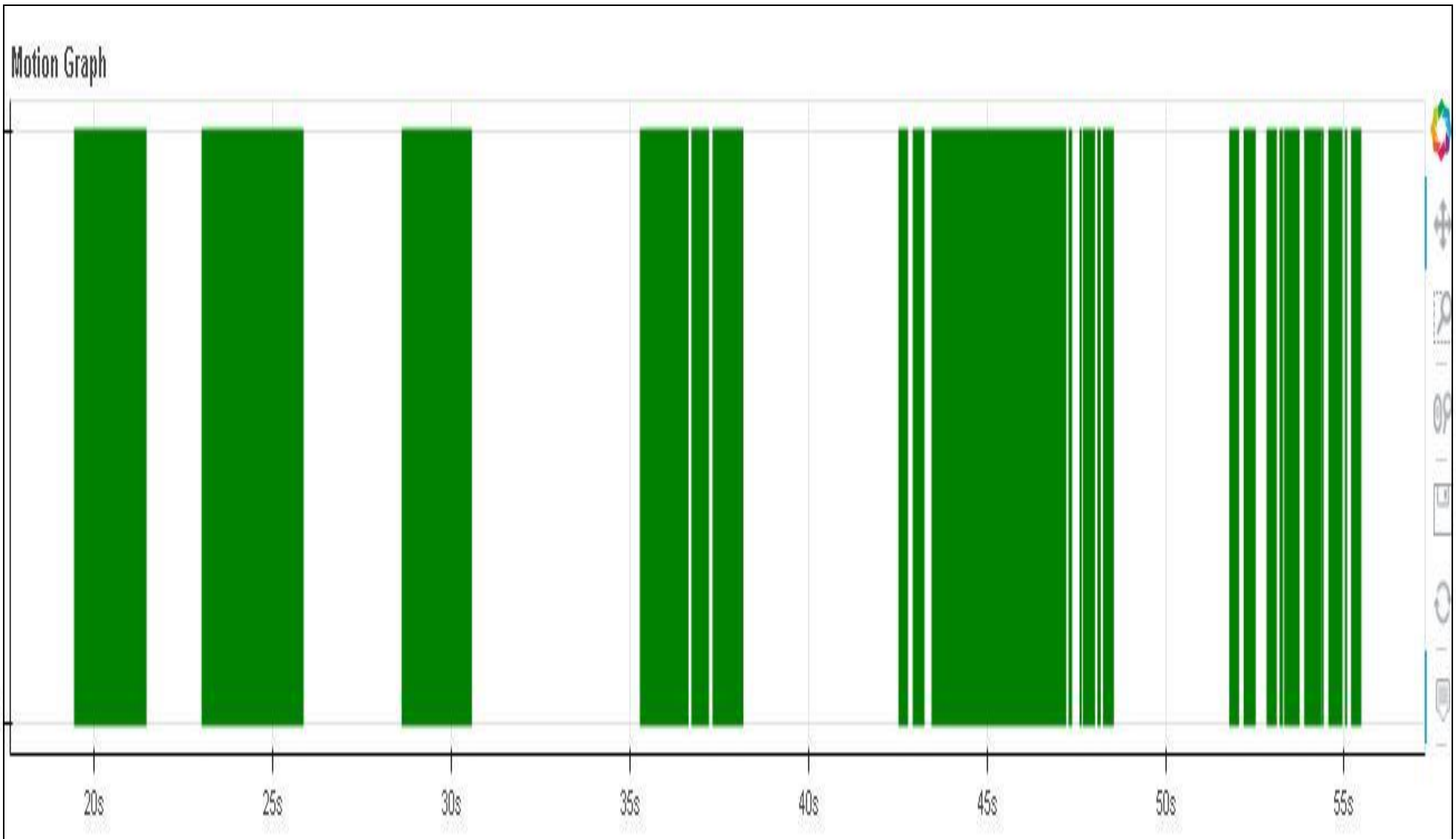
1 min • via Jio 4G

b)The next feature implemented is the motion graph related to the motion:

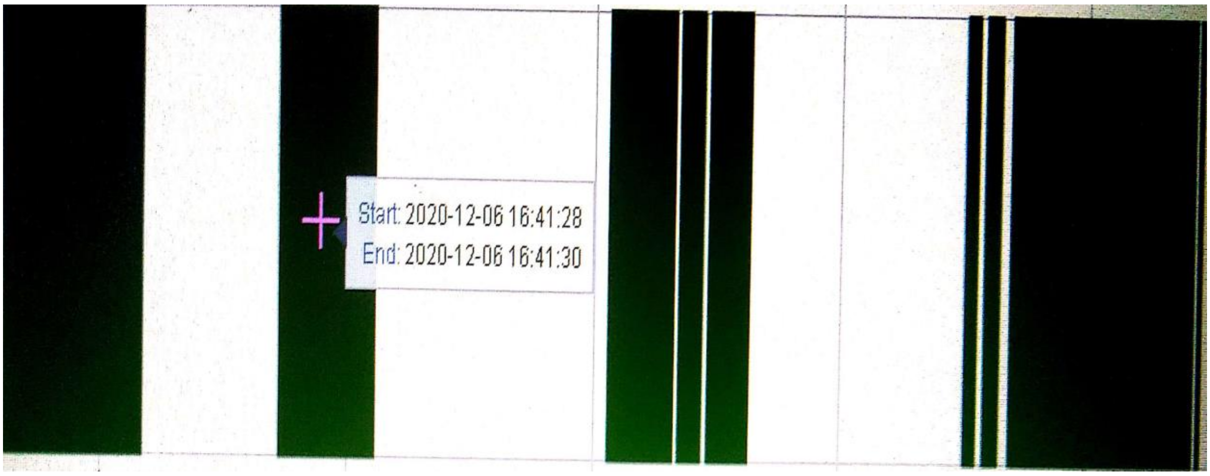
SNIPPET:



```
from At1 import df
from bokeh.plotting import figure, show, output_file
from bokeh.models import HoverTool, ColumnDataSource

df["Start_string"]=df["Start"].dt.strftime("%Y-%m-%d %H:%M:%S")
df["End_string"]=df["End"].dt.strftime("%Y-%m-%d %H:%M:%S")


cds=ColumnDataSource(df)

p=figure(x_axis_type='datetime',height=100, width=500, sizing_mode = "scale_width",title="Motion Graph")
p.yaxis.minor_tick_line_color=None
p.ygrid[0].ticker.desired_num_ticks=1

hover=HoverTool(tooltips=[("Start","@Start_string"),("End","@End_string")])
p.add_tools(hover)

q=p.quad(left="Start",right="End",bottom=0,top=1,color="green",source=cds)

output_file("Graph1.html")
show(p)
```



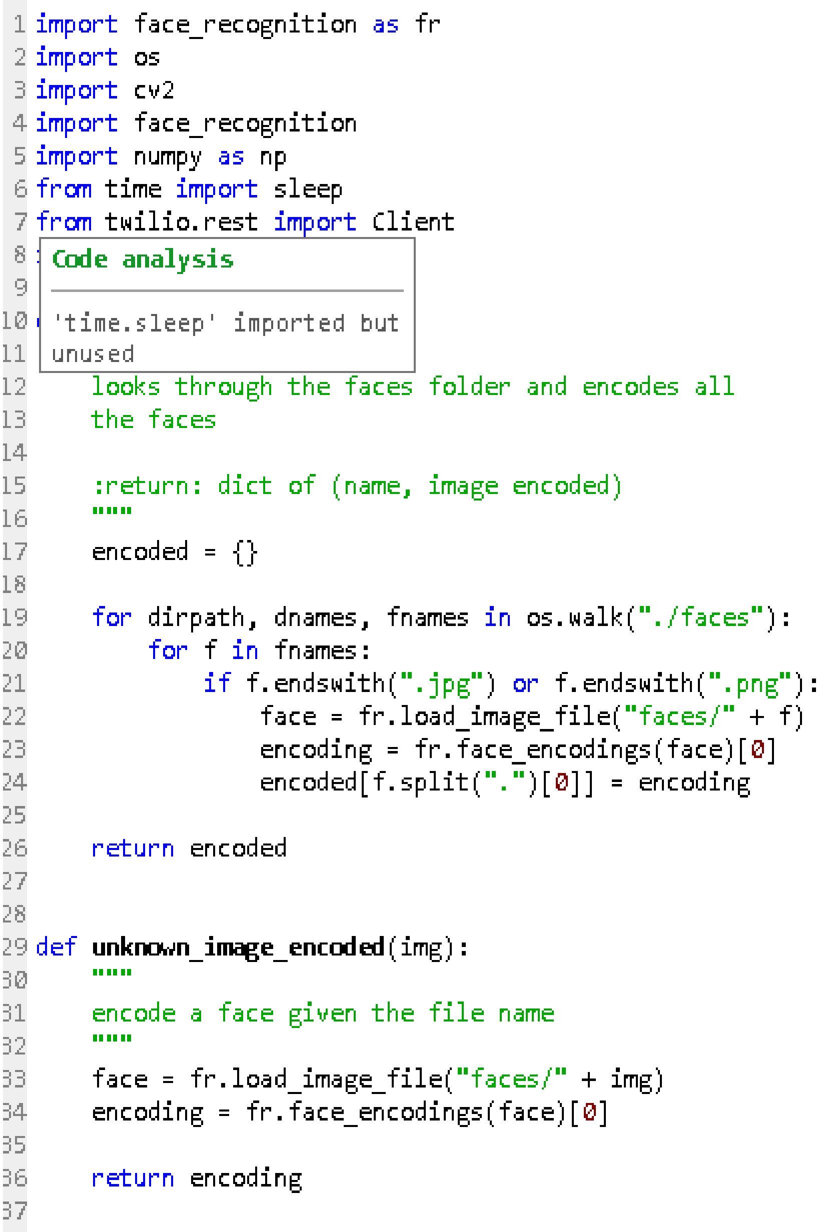
Motion Graph



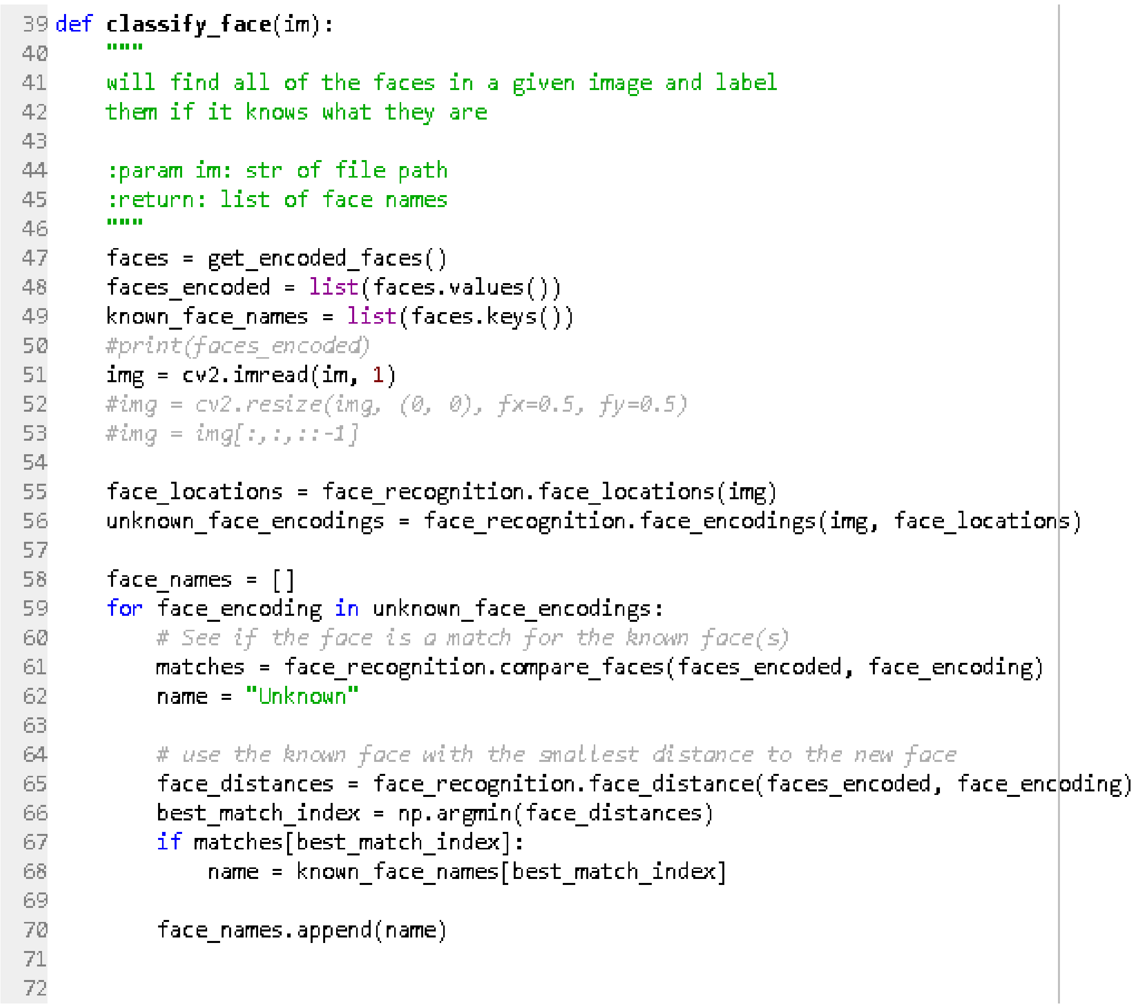
Start: 2020-12-06 16:41:28
End: 2020-12-06 16:41:30

c) Record the image of the unidentified person/face

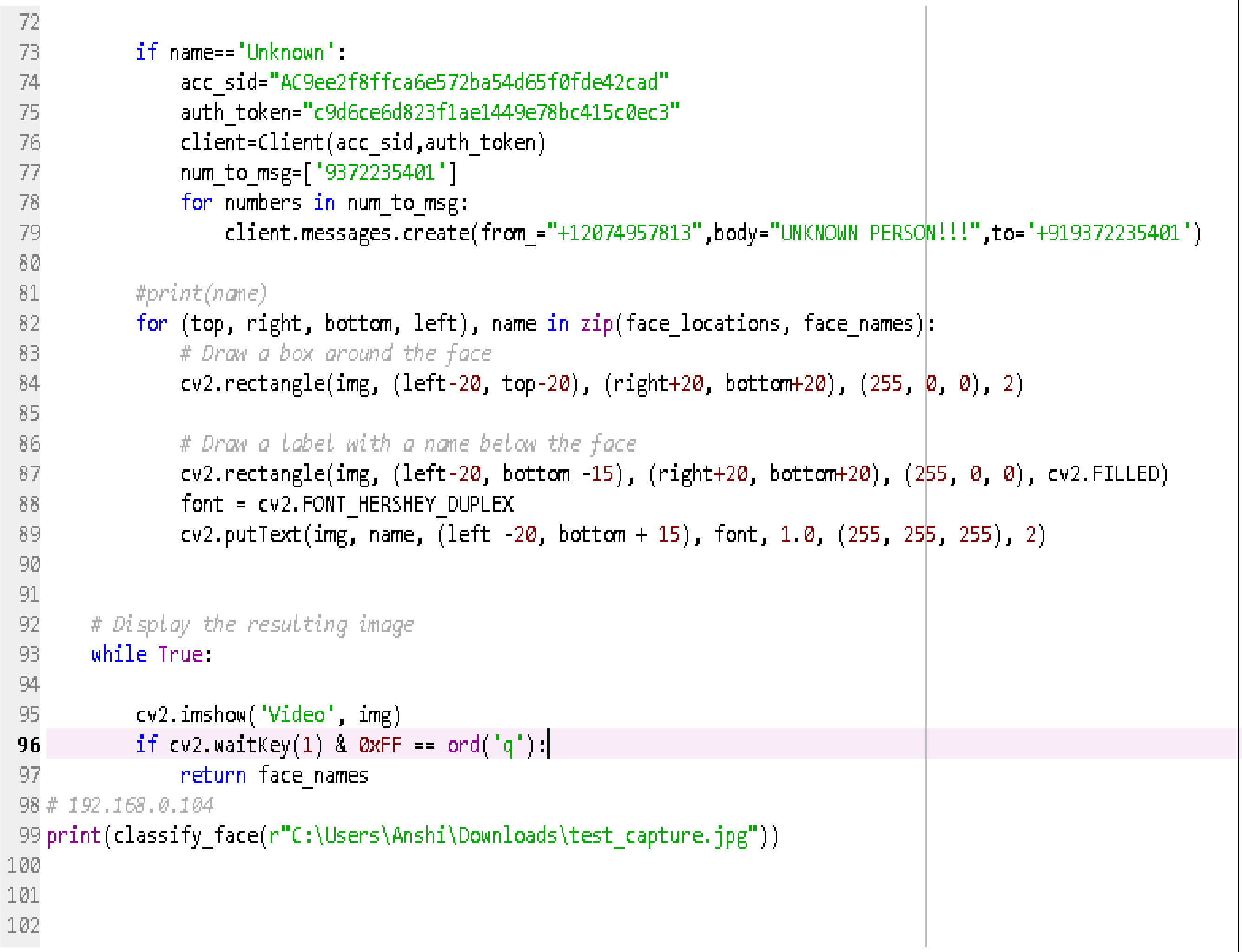
CODE:



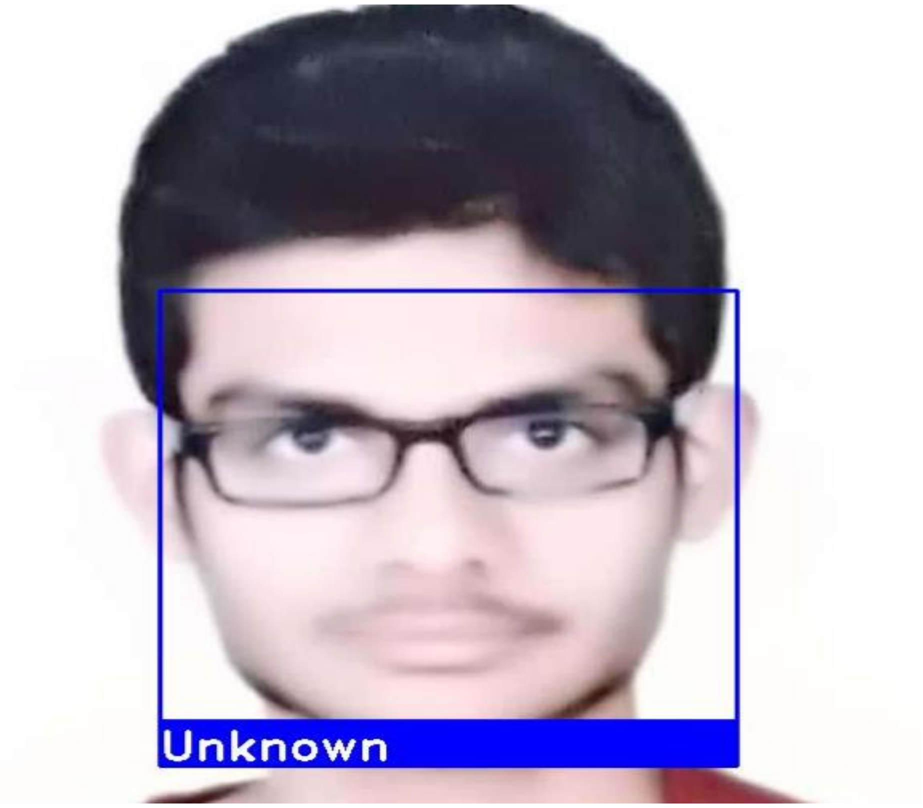
```
import face_recognition as fr
import os
import cv2
import face_recognition
import numpy as np
from time import sleep
from twilio.rest import Client
 Code analysis

 'time.sleep' imported but
 unused
    looks through the faces folder and encodes all
    the faces

    :return: dict of (name, image encoded)
    """
    encoded = {}

    for dirpath, dnames, fnames in os.walk("./faces"):
        for f in fnames:
            if f.endswith(".jpg") or f.endswith(".png"):
                face = fr.load_image_file("faces/" + f)
                encoding = fr.face_encodings(face)[0]
                encoded[f.split(".")[0]] = encoding

    return encoded


def unknown_image_encoded(img):
    """
    encode a face given the file name
    """
    face = fr.load_image_file("faces/" + img)
    encoding = fr.face_encodings(face)[0]

    return encoding

```



```python
def classify_face(im):
    """
    will find all of the faces in a given image and label
    them if it knows what they are

    :param im: str of file path
    :return: list of face names
    """
    faces = get_encoded_faces()
    faces_encoded = list(faces.values())
    known_face_names = list(faces.keys())
    #print(faces_encoded)
    img = cv2.imread(im, 1)
    #img = cv2.resize(img, (0, 0), fx=0.5, fy=0.5)
    #img = img[:,:,::-1]

    face_locations = face_recognition.face_locations(img)
    unknown_face_encodings = face_recognition.face_encodings(img, face_locations)

    face_names = []
    for face_encoding in unknown_face_encodings:
        # See if the face is a match for the known face(s)
        matches = face_recognition.compare_faces(faces_encoded, face_encoding)
        name = "Unknown"

        # use the known face with the smallest distance to the new face
        face_distances = face_recognition.face_distance(faces_encoded, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = known_face_names[best_match_index]

        face_names.append(name)


```

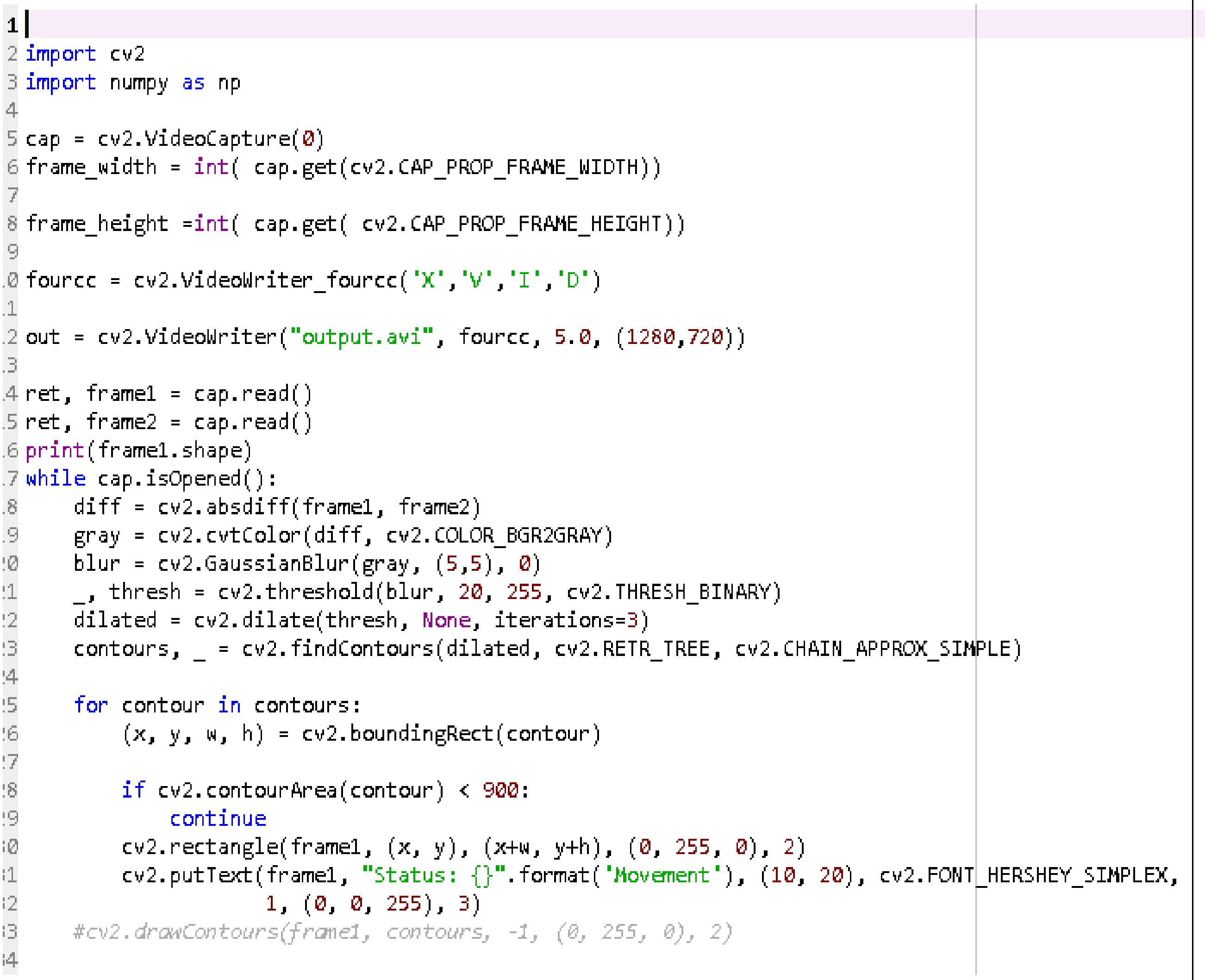


```python

        if name=='Unknown':
            acc_sid="AC9ee2f8ffca6e572ba54d65f0fde42cad"
            auth_token="c9d6ce6d823f1ae1449e78bc415c0ec3"
            client=Client(acc_sid,auth_token)
            num_to_msg=['9372235401']
            for numbers in num_to_msg:
                client.messages.create(from_="+12074957813",body="UNKNOWN PERSON!!!",to='+919372235401')

        #print(name)
        for (top, right, bottom, left), name in zip(face_locations, face_names):
            # Draw a box around the face
            cv2.rectangle(img, (left-20, top-20), (right+20, bottom+20), (255, 0, 0), 2)

            # Draw a label with a name below the face
            cv2.rectangle(img, (left-20, bottom -15), (right+20, bottom+20), (255, 0, 0), cv2.FILLED)
            font = cv2.FONT_HERSHEY_DUPLEX
            cv2.putText(img, name, (left -20, bottom + 15), font, 1.0, (255, 255, 255), 2)


    # Display the resulting image
    while True:

        cv2.imshow('Video', img)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            return face_names
# 192.168.0.104
print(classify_face(r"C:\Users\Anshi\Downloads\test_capture.jpg"))



```

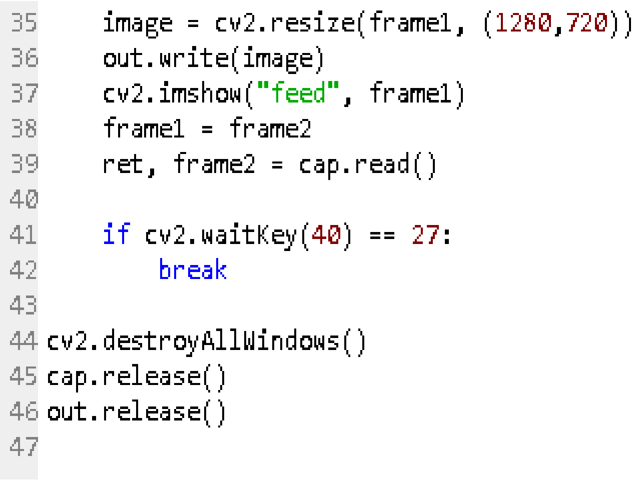


f)Record the footage of the video from the moment the motion was detected.

CODE:



```python

import cv2
import numpy as np

cap = cv2.VideoCapture(0)
frame_width = int( cap.get(cv2.CAP_PROP_FRAME_WIDTH))

frame_height =int( cap.get( cv2.CAP_PROP_FRAME_HEIGHT))

fourcc = cv2.VideoWriter_fourcc('X','V','I','D')

out = cv2.VideoWriter("output.avi", fourcc, 5.0, (1280,720))

ret, frame1 = cap.read()
ret, frame2 = cap.read()
print(frame1.shape)
while cap.isOpened():
    diff = cv2.absdiff(frame1, frame2)
    gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5,5), 0)
    _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
    dilated = cv2.dilate(thresh, None, iterations=3)
    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        (x, y, w, h) = cv2.boundingRect(contour)

        if cv2.contourArea(contour) < 900:
            continue
        cv2.rectangle(frame1, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(frame1, "Status: {}".format('Movement'), (10, 20), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0, 0, 255), 3)
    #cv2.drawContours(frame1, contours, -1, (0, 255, 0), 2)

```

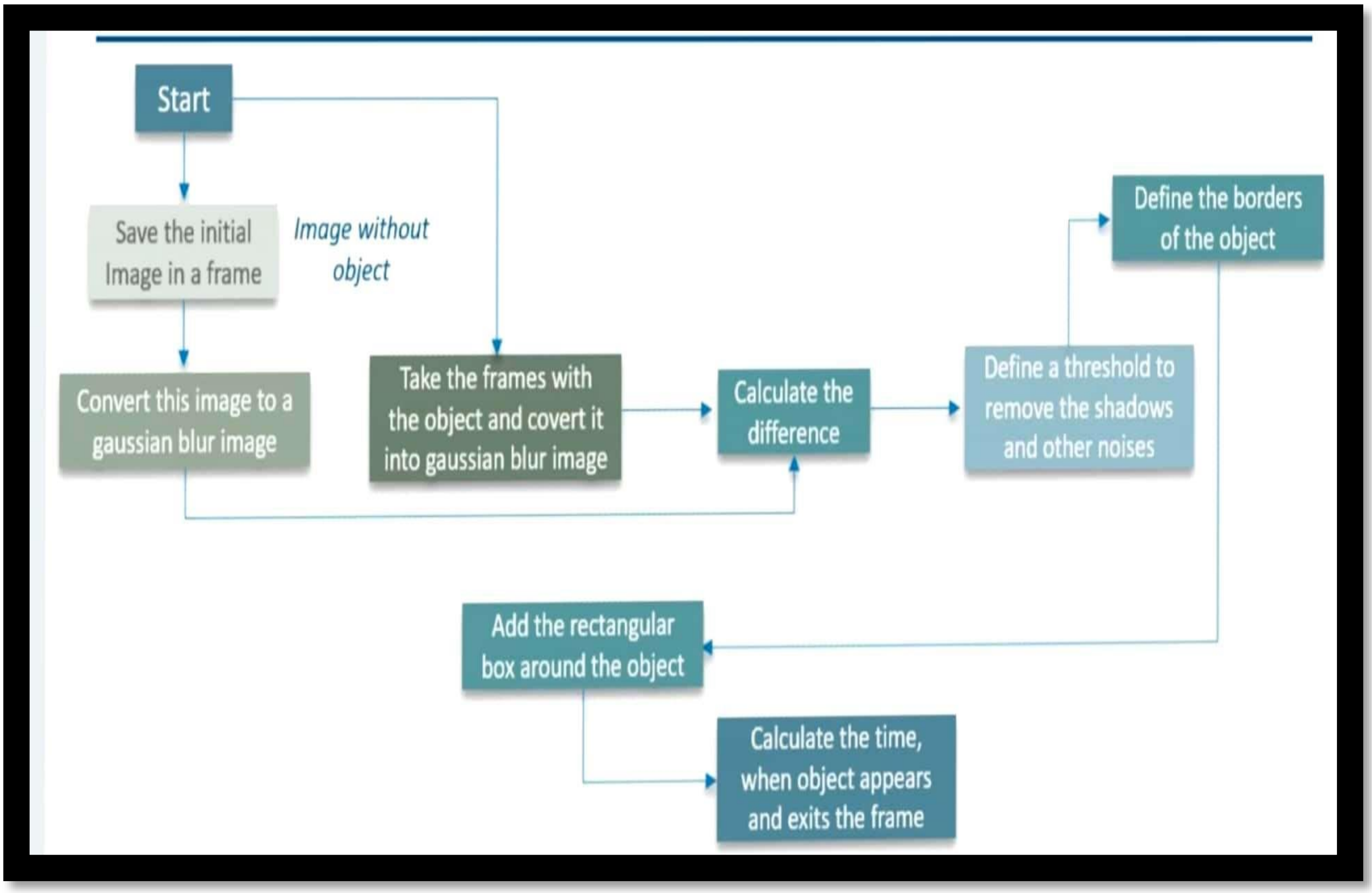


```python
    image = cv2.resize(frame1, (1280,720))
    out.write(image)
    cv2.imshow("feed", frame1)
    frame1 = frame2
    ret, frame2 = cap.read()

    if cv2.waitKey(40) == 27:
        break

cv2.destroyAllWindows()
cap.release()
out.release()

```

The following video file demonstrates how an alarm raised as soon as it detects movement

output.mp4

Now let’s look at the core methododology of how we detect motion:



Start

Save the initial Image in a frame

Image without object

Convert this image to a gaussian blur image

Take the frames with the object and covert it into gaussian blur image

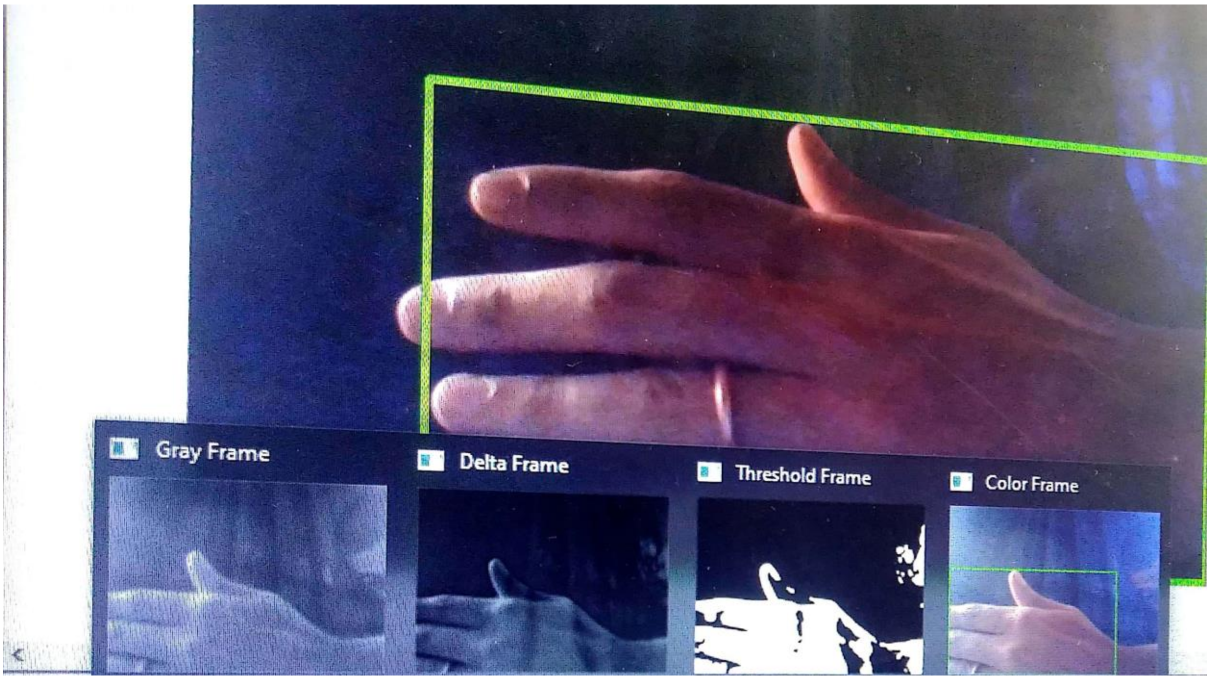
Calculate the difference

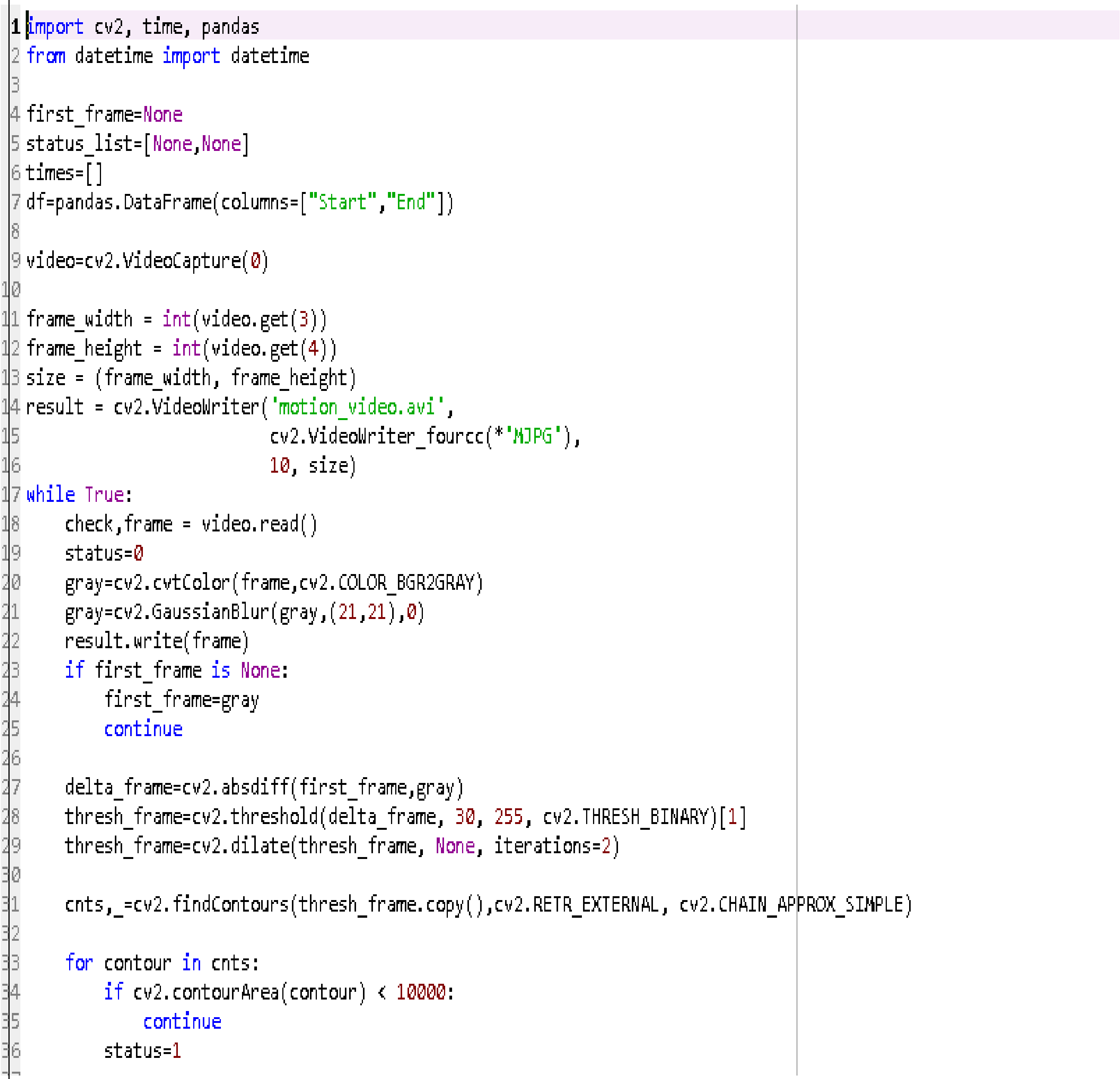
Define a threshold to remove the shadows and other noises

Define the borders of the object

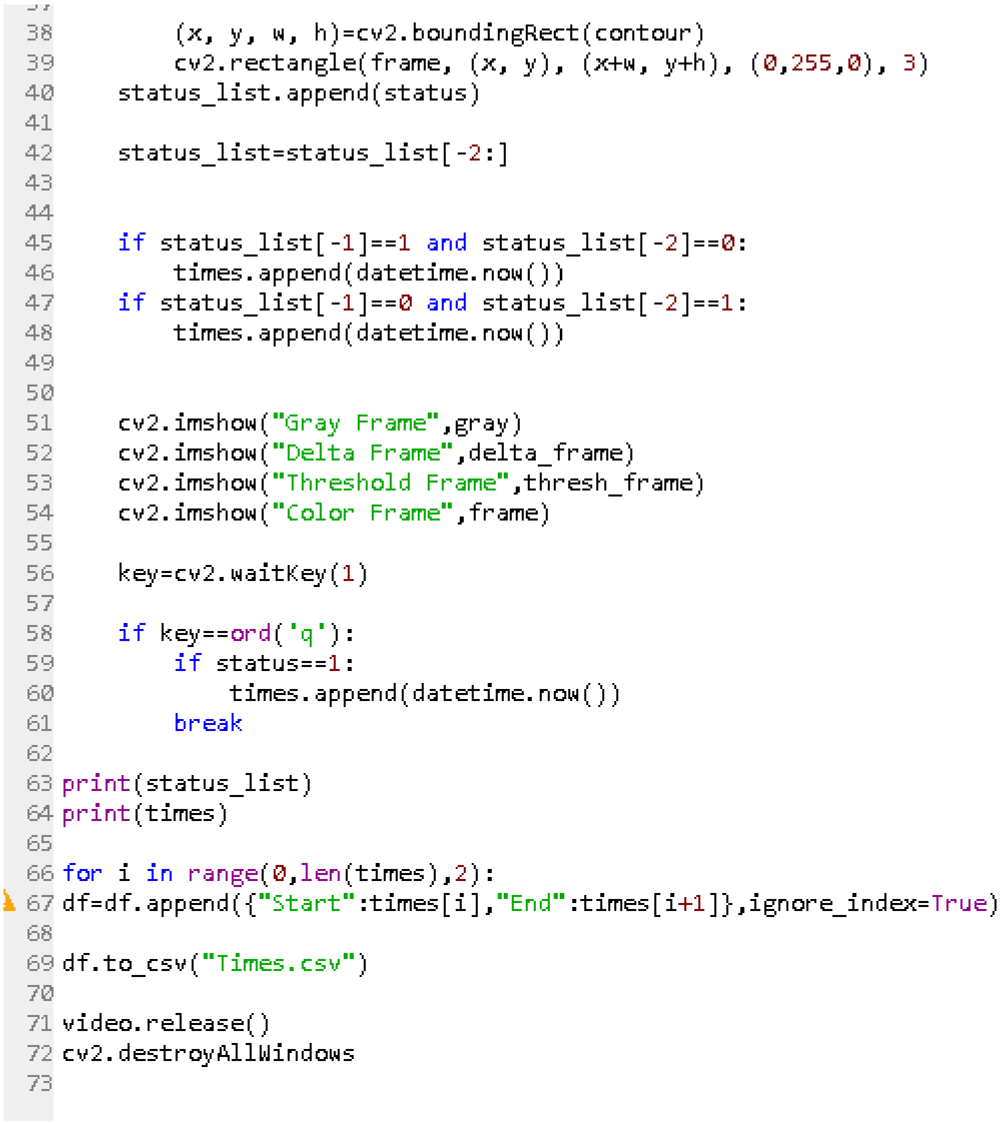
Add the rectangular box around the object

Calculate the time, when object appears and exits the frame


Gray Frame
Delta Frame
Threshold Frame
Color Frame

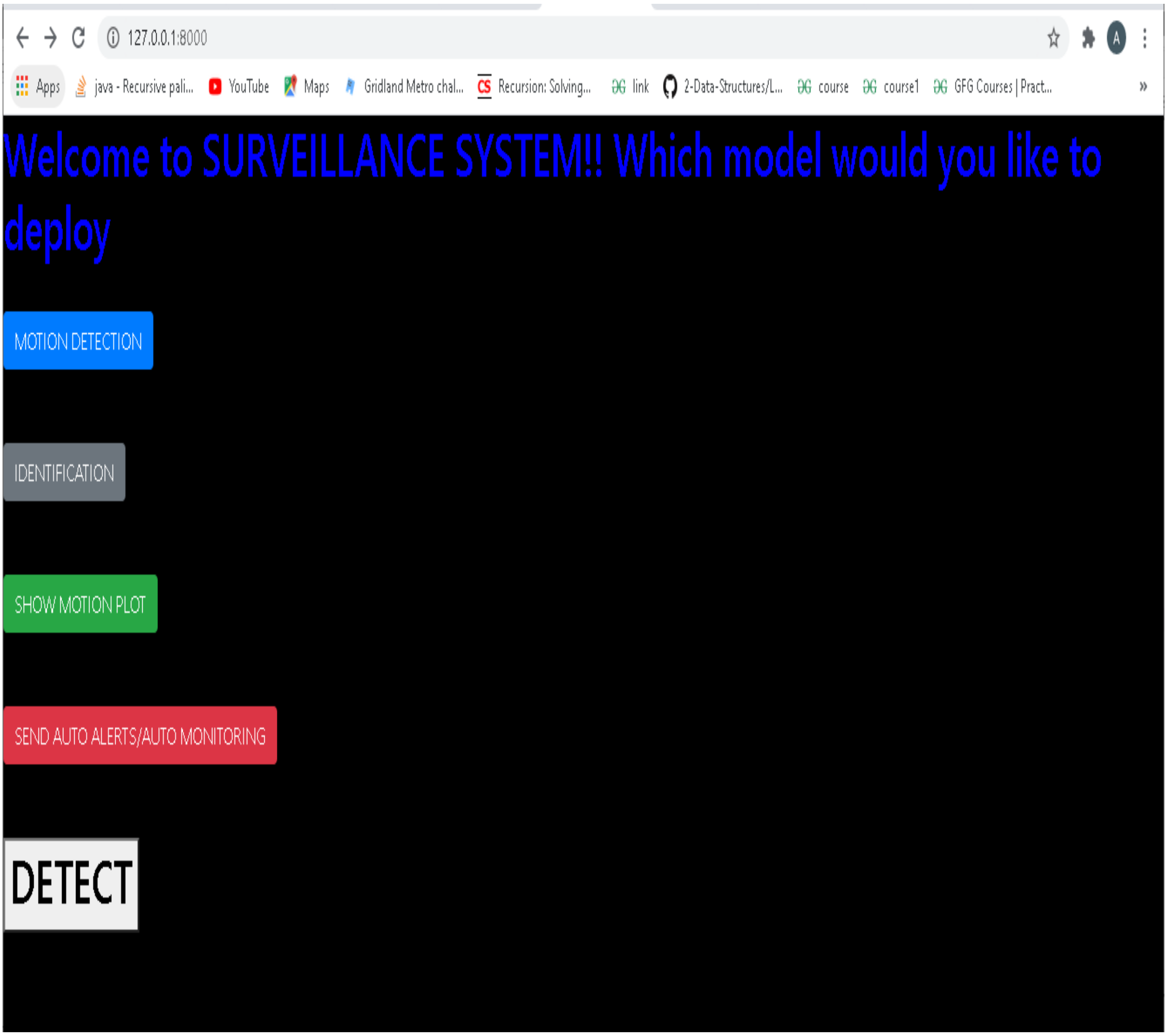


```
import cv2, time, pandas
from datetime import datetime

first_frame=None
status_list=[None,None]
times=[]
df=pandas.DataFrame(columns=["Start","End"])

video=cv2.VideoCapture(0)

frame_width = int(video.get(3))
frame_height = int(video.get(4))
size = (frame_width, frame_height)
result = cv2.VideoWriter('motion_video.avi',
                         cv2.VideoWriter_fourcc(*'MJPG'),
                         10, size)
while True:
    check,frame = video.read()
    status=0
    gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    gray=cv2.GaussianBlur(gray,(21,21),0)
    result.write(frame)
    if first_frame is None:
        first_frame=gray
        continue

    delta_frame=cv2.absdiff(first_frame,gray)
    thresh_frame=cv2.threshold(delta_frame, 30, 255, cv2.THRESH_BINARY)[1]
    thresh_frame=cv2.dilate(thresh_frame, None, iterations=2)

    cnts,_=cv2.findContours(thresh_frame.copy(),cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    for contour in cnts:
        if cv2.contourArea(contour) < 10000:
            continue
        status=1
```



```
        (x, y, w, h)=cv2.boundingRect(contour)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,0), 3)
    status_list.append(status)

    status_list=status_list[-2:]


    if status_list[-1]==1 and status_list[-2]==0:
        times.append(datetime.now())
    if status_list[-1]==0 and status_list[-2]==1:
        times.append(datetime.now())


    cv2.imshow("Gray Frame",gray)
    cv2.imshow("Delta Frame",delta_frame)
    cv2.imshow("Threshold Frame",thresh_frame)
    cv2.imshow("Color Frame",frame)

    key=cv2.waitKey(1)

    if key==ord('q'):
        if status==1:
            times.append(datetime.now())
        break

print(status_list)
print(times)

for i in range(0,len(times),2):
df=df.append({"Start":times[i],"End":times[i+1]},ignore_index=True)

df.to_csv("Times.csv")

video.release()
cv2.destroyAllWindows
```

f)FRONTEND DJANGO APP:



Welcome to SURVEILLANCE SYSTEM!! Which model would you like to deploy

SEMESTER	MONTHS	WORK DONE-7TH SEM WORK PLANNED-8TH SEM
7TH	AUGUST	PROJECT AREA/TOPIC DECISION
	SEPTEMBER	IMPLEMENT TWO PROJECTS SIMULTANEOUSLY- PH DETECTION,SURVEILLANCE SYSTEM
	OCTOBER	COME UP WITH FINAL PROPOSAL,SEARCHING RELATED ARTICLES TO GAIN SUBJECT KNOWLEDGE
	NOVEMBER	SUMMARISE THE INFORMATION,CHART OUT THE DEVELOPMENT PLAN,FINISH THE PRELIMINARY DEVELOPMENT
	DECEMBER	EXPAND THE APLICATION AREAS OF THE PROJECT
8TH	JAN-FEB	ALGORITHMIC ENHANCEMENTS FOR INCREASING EFFICIENCY OF THE BACKEND CODE
	MARCH	PREPARE FINAL REPORT