

# FACE RECOGNITION USING VIOLA & JONES ALGORITHM AND CSRT TRACKING (OPENCV) IN HOSPITALS



Project from AI Application course  
GROUP C

## ABOUT US

Students from INHA UNIVERSITY in South Korea working in a small project from AI Application course leaded by Professor from ISE department Mehdi Pirahandeh based on deep learning and AI. Our project is based on face recognition on healthcare field in order to help both the citizens and assistants in hospital to save time and work efficiently. Let's take a look at how face recognition impacts healthcare.





## ◦ HOW DOES FACE RECOGNITION WORK?

- Facial recognition is a technology that helps identify or verify a person based on their specific facial features. Detect human faces in images, videos or live feeds and compare them to a face database
- In other words, all biometric models are mathematical representations of people's faces. In this way, biometric and personal information are strictly separated, ensuring the highest level of privacy, even when dealing with highly sensitive data.
- The facial recognition software calculates the correspondence between the entered facial descriptor and all facial descriptors previously stored in a database. The goal is to find the face or faces from the database that most closely resemble the input face. The higher the match, the more likely the face is to be identified accurately.





# PROBLEM

## SERVICE IN HOSPITALS



**NO MORE  
WAITING**

- In the waiting hall when patients take their waiting number, sometimes it takes too much time waiting for your turn, so therefore we should make an automated or checking mechanism that checks the people who left the hall.

# SOLUTION



- Our program makes managing tasks easier, and no other program on the market offers the same benefits. It checks the people faces when they are taking the waiting number and selects how many people they are, so mechanism automatically cancels the number who left the hall or waiting room





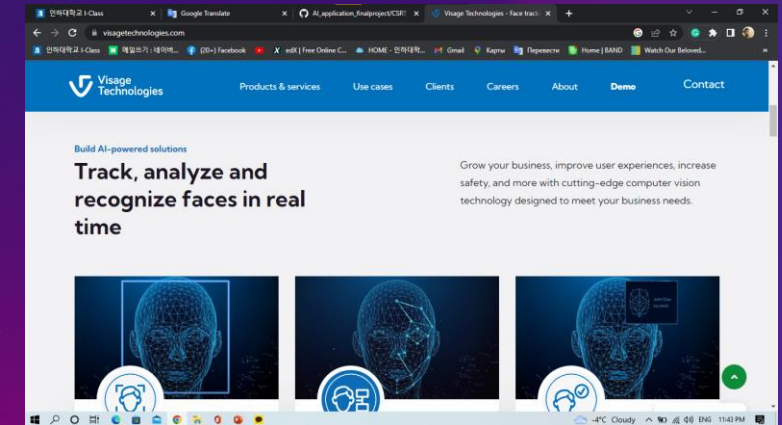
# PREVIOUS RESEARCHES

## 1. VISAGE TECHNOLOGIES

<https://visagetechologies.com/face-recognition-in-healthcare/>

They have already made a face recognition mechanism in healthcare including security

- Facility security
- Patient diagnosis and monitoring
- Patient check in and check out
- Access control



## 2. GENETIC BASED FACE RECOGNITION FOR HEALTHCARE APPLICATIONS

- <http://hdl.handle.net/10995/84113>

They have already made a face recognition research in the field of security authentication

- 3-LEVEL DWT
- GENETICALLY CLOSER
- MOBILE TERMINAL



# PROJECT OVERVIEW



## UNIQUE

Only program specifically  
dedicated to hospital  
management



## FIRST TO MARKET

First beautifully  
designed program that's  
both stylish and functional



## TESTING

Conducted testing  
with datasets we have and  
planning to test in a real  
atmosphere in future



## AUTHENTIC

Designed with the help and  
input of professors in this  
area

# FACE DATA COLLECTION(DATASETS)



Kaggle  
Dataset's

## 1. Kaggle

We are going to use Kaggle for datasets that already exists.

<https://www.kaggle.com/datasets/andrewmvd/face-mask-detection>

## 2. VIS datasets

the link, on this one there is 13,000 thousand image, the one I download was name "all images as zipped tar file"

<http://vis-www.cs.umass.edu/lfw/>

Labeled Faces in the Wild Home





## About this Data

Masks play a crucial role in protecting the health of individuals against respiratory diseases, as is one of the few precautions available for COVID-19 in the absence of immunization. With this dataset, it is possible to create a model to detect people wearing masks, not wearing them, or wearing masks improperly.

This dataset contains 853 images belonging to the 3 classes, as well as their bounding boxes in the PASCAL VOC format.

The classes are:

- With mask;
- Without mask;
- Mask worn incorrectly.

- Ready images
- Already annotated
- in a cascade classifier format **xml**
- Can be directly trained

## Data Explorer

Version 1 (417.89 MB)

### ▼ annotations

- maksssksksss0.xml
- maksssksksss1.xml
- maksssksksss10.xml
- maksssksksss100.xml
- maksssksksss101.xml
- maksssksksss102.xml
- maksssksksss103.xml
- maksssksksss104.xml
- maksssksksss105.xml
- maksssksksss106.xml
- maksssksksss107.xml

## Viola & Jones algorithm using OpenCV

The screenshot shows the Kaggle dataset page for 'Face Mask Detection'. The page header includes the Kaggle logo, a search bar, and links for 'Sign In' and 'Register'. The dataset is by 'LARXEL' and was updated 3 years ago. It has 1499 votes and a 'New Notebook' button. A 'Download (417 MB)' button is also present. The dataset title 'Face Mask Detection' is prominently displayed, followed by the description '853 images belonging to 3 classes.' and a thumbnail image of a person wearing a blue face mask. Below the title, there are tabs for 'Data', 'Code (201)', and 'Discussion (6)'. The 'About Dataset' section is visible, along with a 'Preview' section showing a grid of images. On the right side, there are metrics for 'Usability' (8.75), 'License' (CC0: Public Domain), and 'Expected update frequency' (Never).

<https://www.kaggle.com/datasets/andrewmvd/face-mask-detection?select=annotations>

GUI cascade trainer for training our datasets with giving positive and negative images

The screenshot shows the 'Train' tab of the Cascade Trainer GUI. The 'Boost Type' is set to 'GAB'. The 'Minimal Hit Rate' is 0.9950000, 'Maximal False Alarm Rate' is 0.5000000, 'Weight Trim Rate' is 0.9500000, 'Maximal Depth Weak Tree' is 1.0000000, and 'Maximal Weak Trees' is 100.

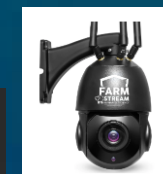
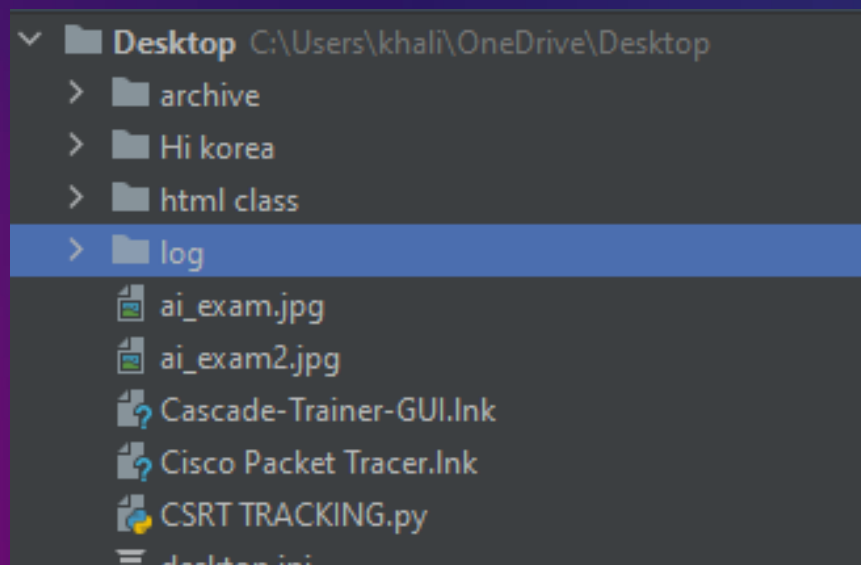
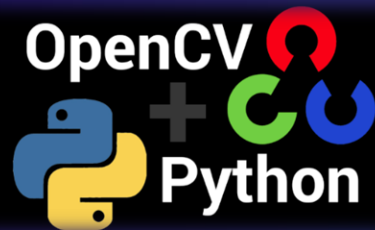
The screenshot shows the 'Train' tab of the Cascade Trainer GUI. The 'Samples Folder' is empty. The 'Positive Image Usage (percentage)' is set to 100, and the 'Force Positive Sample Count' is set to 1. The 'Negative Image Count' is 1000. A 'Start' button is visible at the bottom right.

### Cascade classifiers

haarcascade_eye.xml	12/21/2013 11:21 ...	XML File	334 KB
haarcascade_eye_tree_eyeglasses.xml	12/21/2013 11:21 ...	XML File	588 KB
haarcascade_frontalcatface.xml	12/31/2014 9:55 AM	XML File	370 KB
haarcascade_frontalcatface_extended.xml	12/31/2014 9:55 AM	XML File	353 KB
haarcascade_frontalface_alt.xml	12/21/2013 11:21 ...	XML File	661 KB
haarcascade_frontalface_alt_tree.xml	12/21/2013 11:21 ...	XML File	2,627 KB
haarcascade_frontalface_alt2.xml	12/21/2013 11:21 ...	XML File	528 KB
haarcascade_frontalface_default.xml	12/21/2013 11:21 ...	XML File	909 KB
haarcascade_fullbody.xml	2/3/2015 9:41 AM	XML File	466 KB
haarcascade_lefteye_2splits.xml	12/21/2013 11:21 ...	XML File	191 KB
haarcascade_licence_plate_rus_16stages.xml	5/31/2014 9:41 AM	XML File	47 KB
haarcascade_lowerbody.xml	2/3/2015 9:41 AM	XML File	387 KB
haarcascade_profileface.xml	12/21/2013 11:21 ...	XML File	810 KB
haarcascade_righteye_2splits.xml	12/21/2013 11:21 ...	XML File	192 KB
haarcascade_russian_plate_number.xml	5/20/2014 10:21 AM	XML File	74 KB
haarcascade_smile.xml	2/3/2015 9:41 AM	XML File	185 KB
haarcascade_upperbody.xml	2/3/2015 9:41 AM	XML File	768 KB



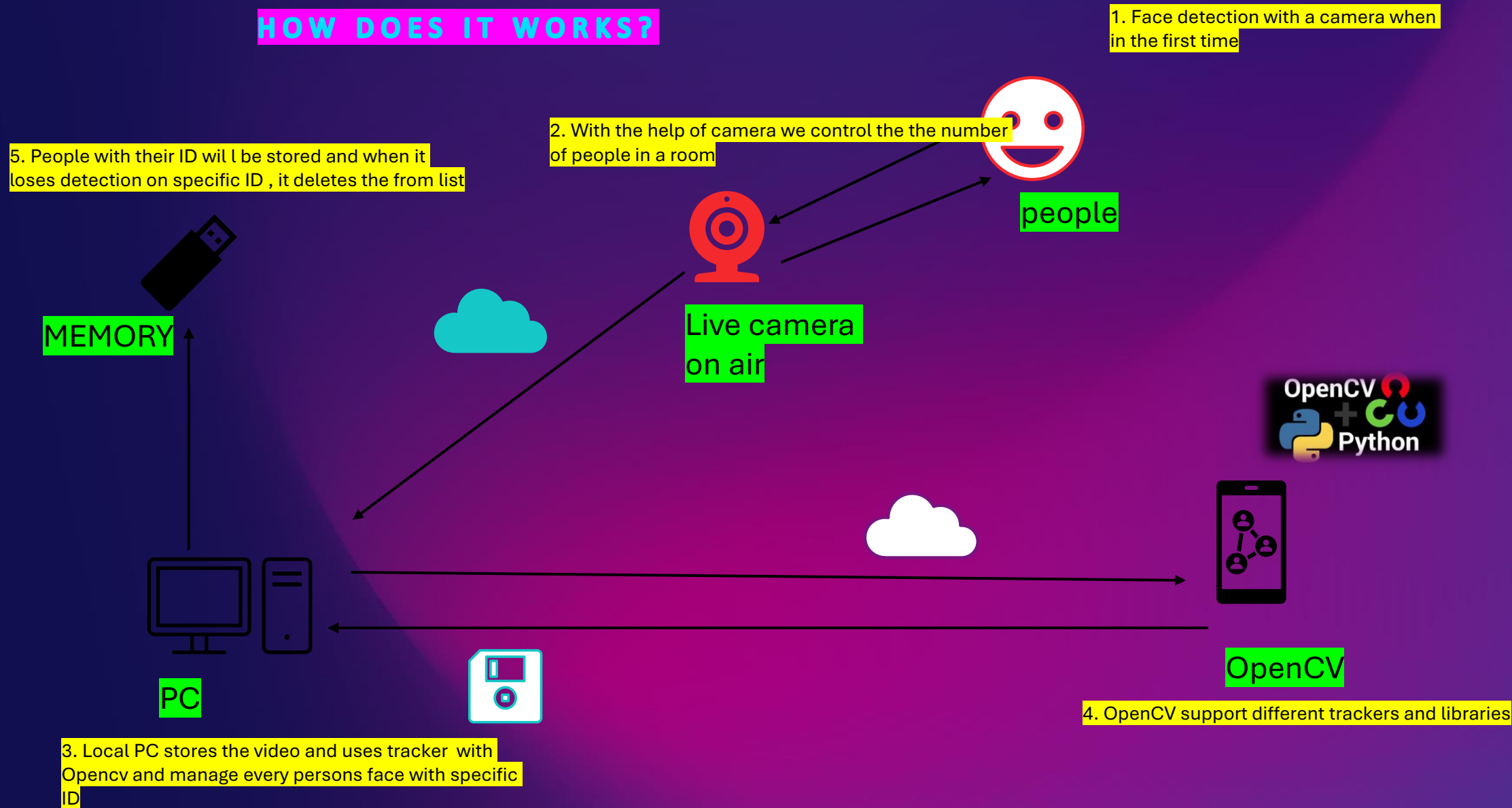
## CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability) tracker



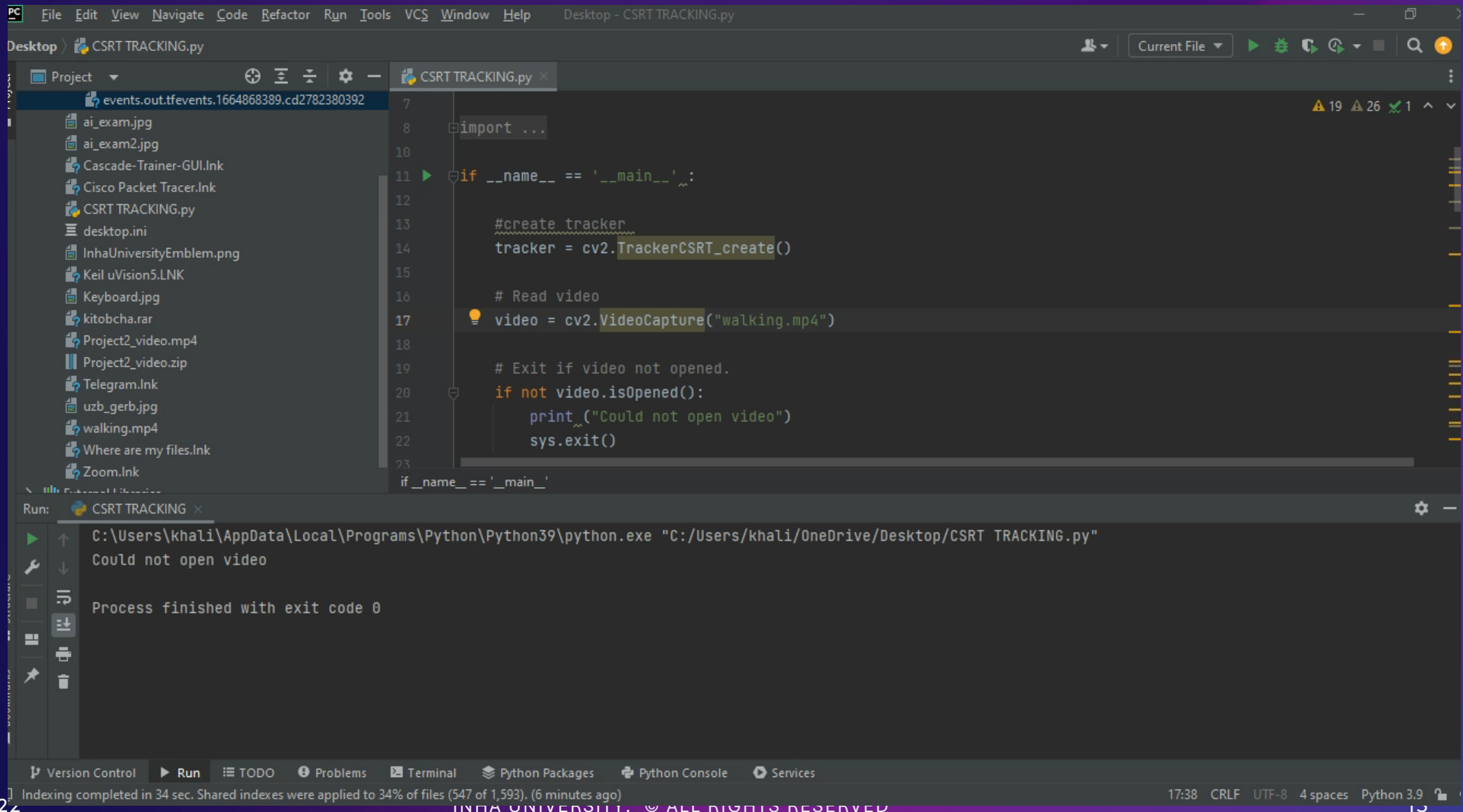
- We are using python code, with trained datasets
- Using PyCharm to run and boot
- we used a video to test it
- We can also use live camera capture to track



## HOW DOES IT WORKS?



# THE OUTPUT EXAMPLE OF CSRT TRACKING USING OPENCV



The screenshot displays a code editor with the file `CSRT TRACKING.py` open. The code is as follows:

```
7
8 import ...
10
11 if __name__ == '__main__':
12
13     #create tracker
14     tracker = cv2.TrackerCSRT_create()
15
16     # Read video
17     video = cv2.VideoCapture("walking.mp4")
18
19     # Exit if video not opened.
20     if not video.isOpened():
21         print("Could not open video")
22         sys.exit()
23
24 if __name__ == '__main__':
```

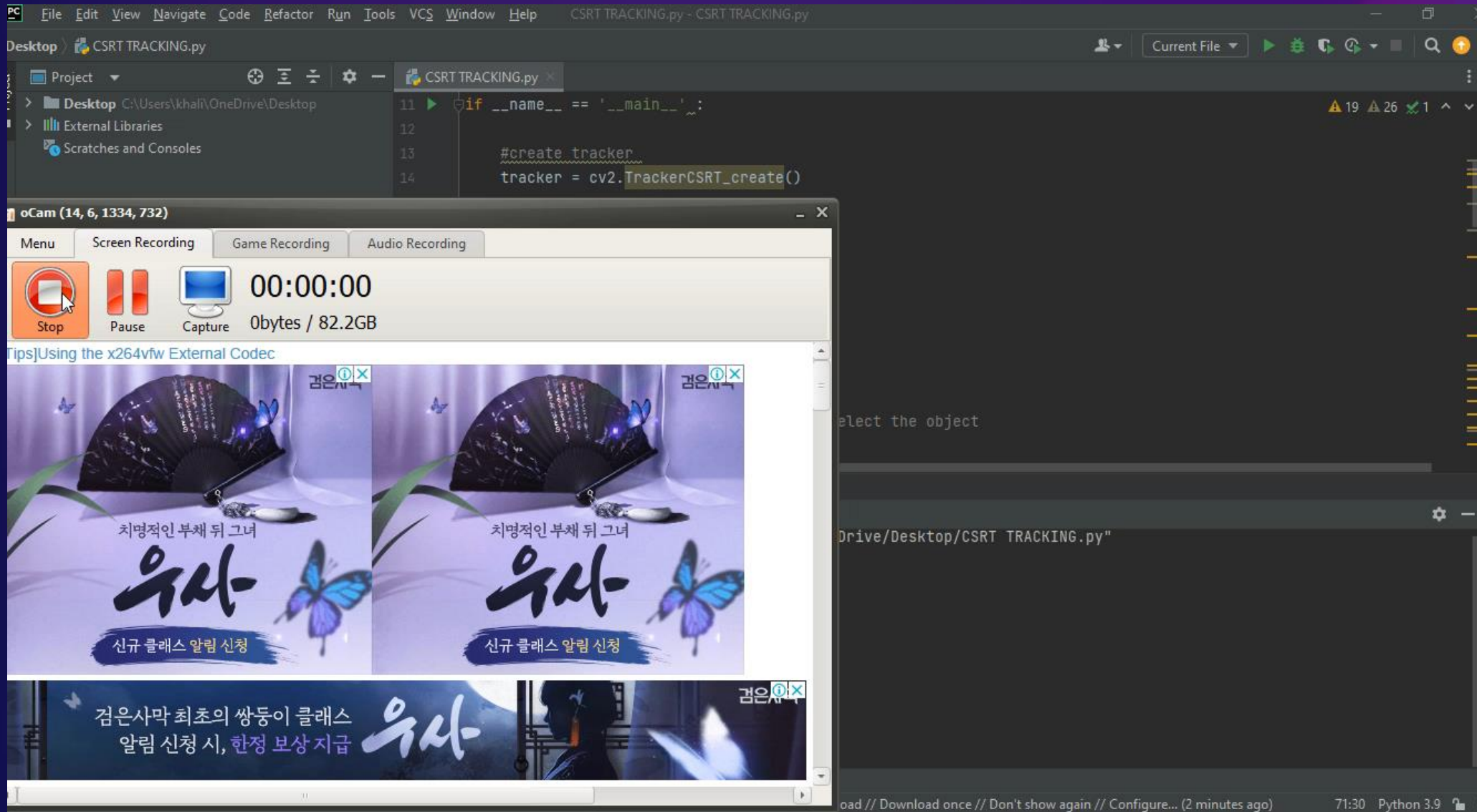
The output window shows the execution of the script:

```
Run: CSRT TRACKING x
C:\Users\khalil\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/khalil/OneDrive/Desktop/CSRT TRACKING.py"
Could not open video

Process finished with exit code 0
```

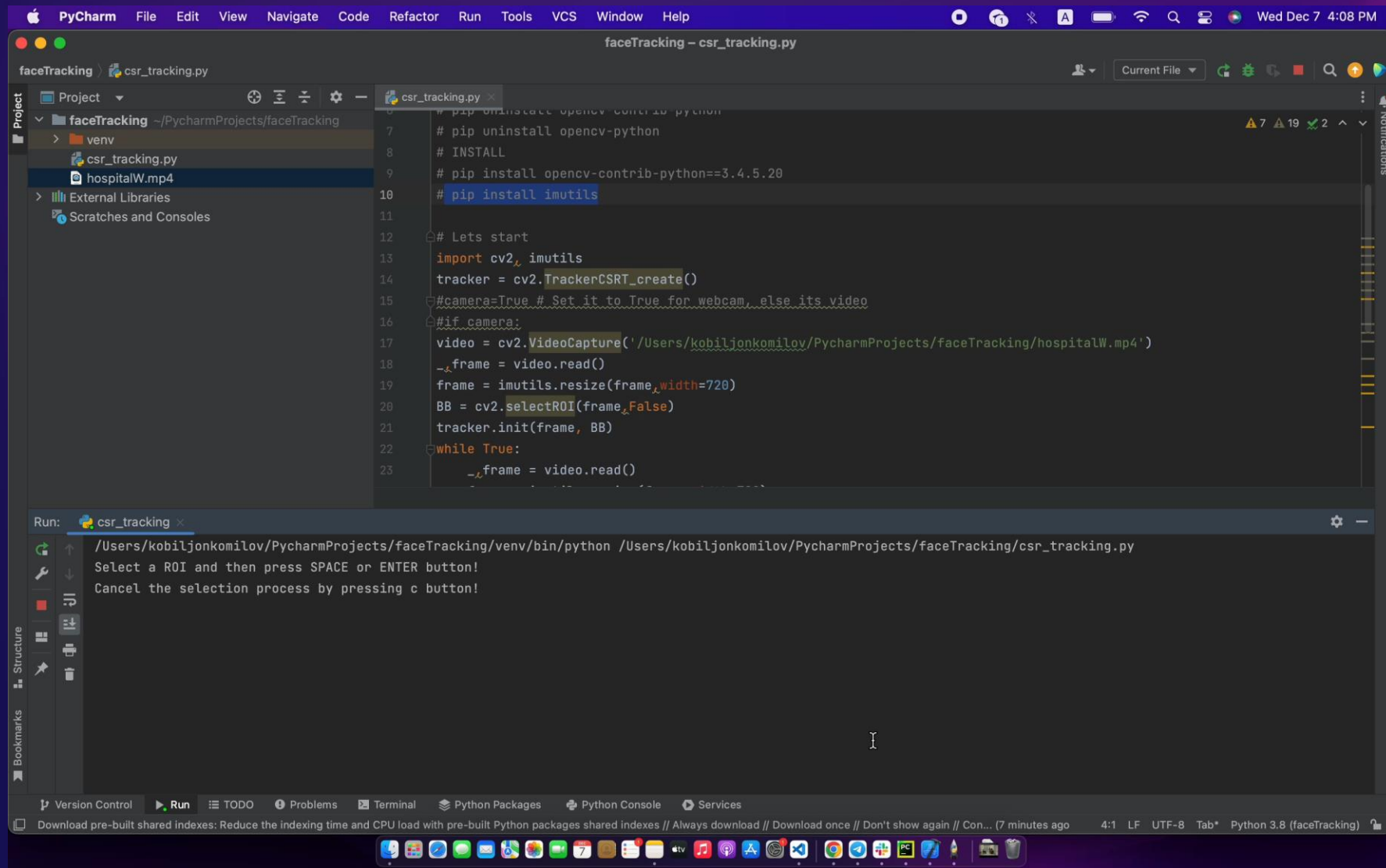
The status bar at the bottom indicates: 11/13/2022, Indexing completed in 34 sec. Shared indexes were applied to 34% of files (547 of 1,593). (6 minutes ago), 17:38 CRLF UTF-8 4 spaces Python 3.9, and INHA UNIVERSITY. © ALL RIGHTS RESERVED.

# THE OUTPUT EXAMPLE OF CSRT TRACKING USING OPENCV





# THE OUTPUT EXAMPLE OF TRACKING USING OPENCV



A robotic hand and a human hand are shown on the left side of the slide, reaching towards each other. The robotic hand is white with red joints, and the human hand is a realistic skin tone. They are positioned vertically, with the robotic hand at the top and the human hand at the bottom.

## **SPECIFICATION**

In our scenario, we are planning to implement face recognition using OpenCV and Python.

Here are the necessary libraries that we will use in our project:

**1.OPENCV**

**2.DLIB**

**3.FACE\_RECOGNITION**



## EXTRACT FACIAL FEATURES

First, it is necessary to have a dataset. The dataset should contain all images in folders with each folder containing images of the people.

## THE PYTHON CODE FOR FACE RECOGNITION

```
from imutils import paths #imutils includes opencv functions
import face_recognition
import pickle
import cv2
import os
```

```
#get paths of each file in folder named Images
#Images here that contains data(folders of various people)
imagePath = list(paths.list_images('Images'))
kEncodings = []
kNames = []
```

```
# loop over the image paths
for (i, ip) in enumerate(imagePath):
    # extract the person name from the image path
    name = ip.split(os.path.sep)[-2]
    # load the input image and convert it from BGR
    image = cv2.imread(ip)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```



## THE PYTHON CODE FOR FACE RECOGNITION(CONT.)

```
boxes = face_recognition.face_locations(rgb,model='hog')
# compute the facial embedding for the any face
encodings = face_recognition.face_encodings(rgb, boxes)
# loop over the encodings
for encoding in encodings:
    kEncodings.append(encoding)
    kNames.append(name)
```

```
#save emcodings along with their names in dictionary data
data = {"encodings": kEncodings, "names": kNames}
#use pickle to save data into a file for later use
f = open("face_enc", "wb")
f.write(pickle.dumps(data))#to open file in write mode
f.close()#to close file
```

## HOW TO RECOGNIZE FACE IN IMAGES

```
import face_recognition
import imutils #imutils includes opencv functions
import pickle
import time
import cv2
import os
```

```
#to find path of xml file containing haarCascade file
cfp = os.path.dirname(cv2.__file__) + "/data/haarcascade_frontalface_alt2.xml"
# load the harcaascade in the cascade classifier
fc = cv2.CascadeClassifier(cfp)
# load the known faces and embeddings saved in last file
data = pickle.loads(open('face_enc', "rb").read())
```

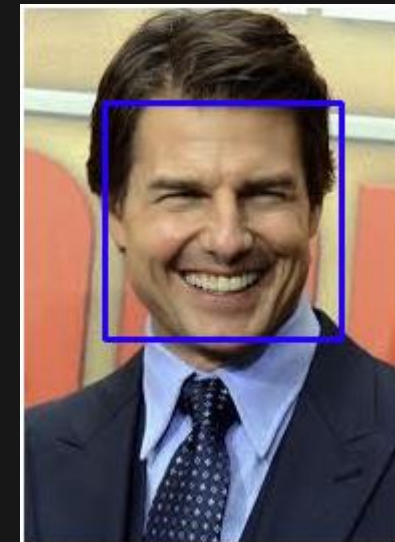
```
#Find path to the image you want to detect face and pass it here
image = cv2.imread(Path-to-img)
rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
#convert image to Greyscale for HaarCascade
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = fc.detectMultiScale(gray,
scaleFactor=1.1,
minNeighbors=6,
minSize=(60, 60),
flags=cv2.CASCADE_SCALE_IMAGE)
```

## HOW TO RECOGNIZE FACE IN IMAGES(CONT.)

```
encodings = face_recognition.face_encodings(rgb)
names = []
# loop over the facial embeddings incase
# we have multiple embeddings for multiple faces
for encoding in encodings:
    #Compare encodings with encodings in data["encodings"]
    #Matches contain array with boolean values True and False
    matches = face_recognition.compare_faces(data["encodings"],
    encoding)
    #set name =unknown if no encoding matches
    name = "Unknown"
    # check to see if we have found a match
    if True in matches:
        #Find positions at which we get True and store them
        matchedIdxs = [i for (i, b) in enumerate(matches) if b]
        count = {}
        # loop over the matched indexes and maintain a count for
        # each recognized face face
        for i in matchedIdxs:
            #Check the names at respective indexes we stored in matchedIdxs
            name = data["names"][i]
            #increase count for the name we got
            count[name] = count.get(name, 0) + 1
        #set name which has highest count
        name = max(count, key=count.get)
        # will update the list of names
        names.append(name)
```

```
# do loop over the recognized faces
for ((x, y, w, h), name) in zip(faces, names):
    # rescale the face coordinates
    # draw the predicted face name on the image
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cv2.putText(image, name, (x, y), cv2.FONT_HERSHEY_SIMPLEX,
    0.75, (0, 255, 0), 2)
    cv2.imshow("Frame", image)
    cv2.waitKey(0)
```

OUTPUT





## GITHUB LINK



[https://github.com/Khai2708/AI\\_application\\_finalproject.git](https://github.com/Khai2708/AI_application_finalproject.git)

All source files are uploaded to Github

## SOURCE FILE

PC CSRT\_FACE\_TRACKING.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """
7
8 import cv2
9 import sys
10
11 if __name__ == '__main__':
12
13     #create tracker
14     tracker = cv2.TrackerCSRT_create()
15
16     # Read video
17     video = cv2.VideoCapture("walking.mp4")
18
19     # Exit if video not opened.
20     if not video.isOpened():
21         print ("Could not open video")
22         sys.exit()
23
24     # Read first frame to be able to select the object
25     ok, frame = video.read()
26     if not ok:
27         print ('Cannot read video file')
28         sys.exit()
```

[https://github.com/Khai2708/AI\\_application\\_finalproject/blob/main/CSRT\\_FACE\\_TRACKING.py](https://github.com/Khai2708/AI_application_finalproject/blob/main/CSRT_FACE_TRACKING.py)

# SUMMARY

The implementation of facial recognition technology in healthcare has transformed many surgeries for the better. From improved patient safety and identification to better patient tracking and diagnosis, this has helped improve the patient experience and reduce the workload of healthcare workers.

Facial recognition will be even more ubiquitous in the future than it is today. Advanced health monitoring and care robots are just some of the applications being developed. The potential for facial recognition in healthcare is huge, and we haven't seen the best of it yet.







# THANK YOU

AI APPLICATION COURSE

Group C

ASADBEK 12194882

ULUGBEK 12194914

ABDULLOKH 12194900

KOBILJON 12194939

KALANDAROV BEKZODBEK