FACIAL RECOGNITION AND TRACKING

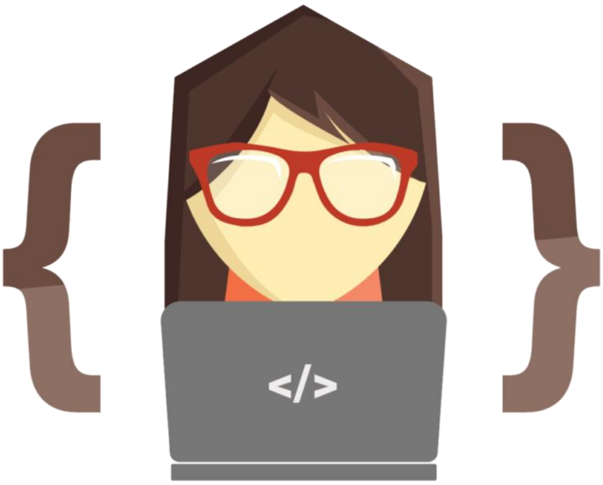
A DAY WISE PROJECT REPORT

GIRL SCRIPT DEVELOPER TECH CAMP HACK-IN PROJECT

Hack –In is a week-long coding challenge in which the participants build a small-scale project using new technology.

NAMAN BHANDARI
Manipal University Jaipur

ACKNOWLEDGEMENT



I would like to express my special thanks of gratitude to my mentor Maitree Rawat and team of GirlScript Jaipur who gave me the golden opportunity to do this wonderful project on Face Recognition, which also helped me in learning a lot of new things and technologies. Moreover, it helped me to experience essence of open source. Also, I would like to thank all the co mentees for helping me to work on project.

ABSTRACT

The growing interest in computer vision of the past decade. Fueled by the steady doubling rate of computing power every 13 months, face detection and recognition has transcended from an esoteric to a popular area of research in computer vision and one of the better and successful applications of image analysis and algorithm based understanding. Because of the intrinsic nature of the problem, computer vision is not only a computer science area of research, but also the object of neuro-scientific and psychological studies, mainly because of the general opinion that advances in computer image processing and understanding research will provide insights into how our brain work and vice versa. Because of general curiosity and interest in the matter, the author has proposed to create an application that would allow user access to a particular machine based on an in-depth analysis of a person's facial features..

TECHNOLOGY STACK

Following programming languages, models and libraries have been used as tech stack for this project:

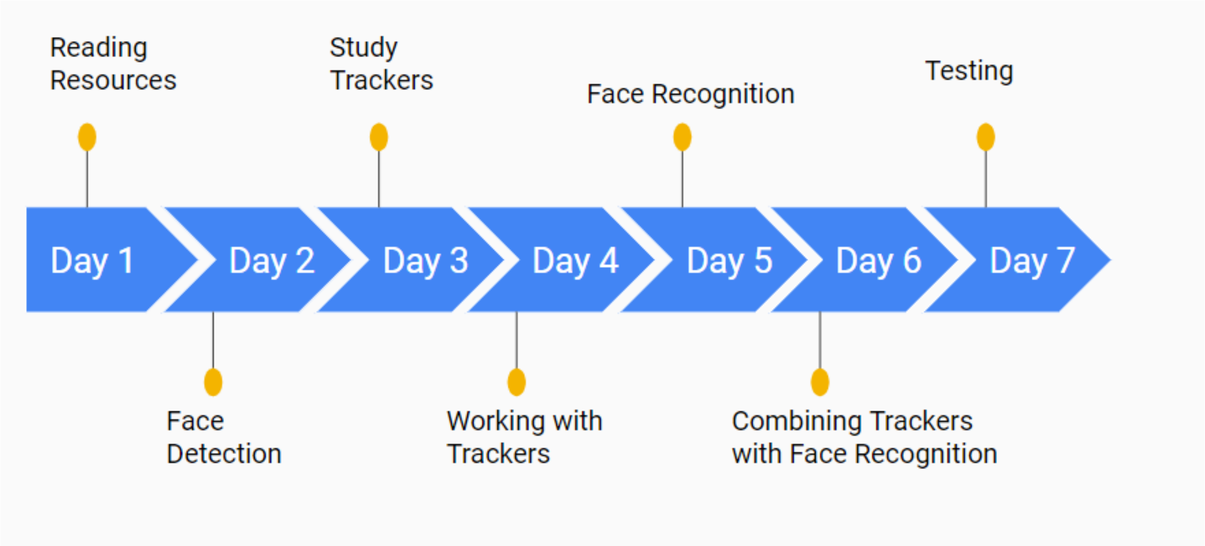
1. Python 3: Programming language used
2. OpenCV: Library used for computer vision
3. Imutils: Library used for image processing
4. Haarcascades : Pre-trained model for face recognition

DAY 1: CREATING TIMELINE

- ❖ Objective:

To create a time line for project

- ❖ Timeline Image:



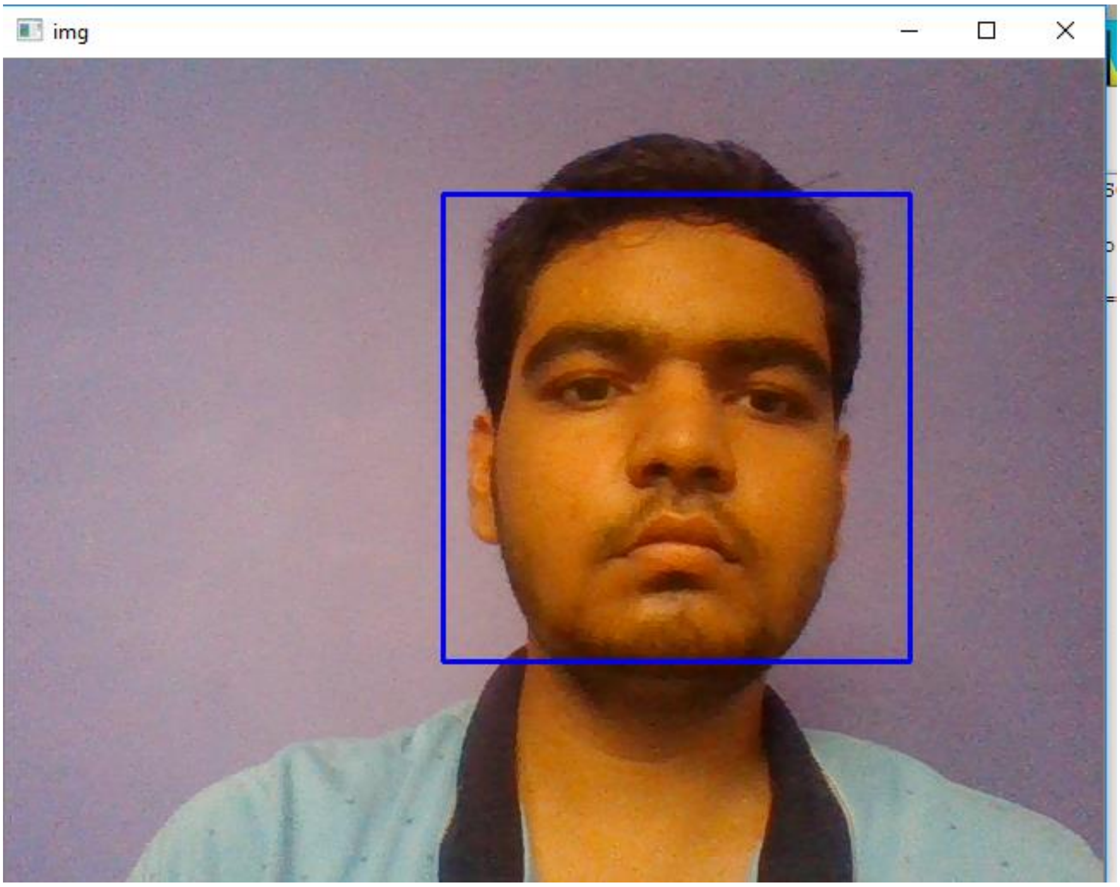
DAY 2: FACE DETECTION

- ❖ Objective: To perform face detection using haarcascades

- ❖ Code if any:

```
import
numpy
as np
         import cv2


         face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
         cap = cv2.VideoCapture(0)
         while True:
             ret, img = cap.read()
             gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
             faces = face_cascade.detectMultiScale(gray, 1.3, 5)

             for (x, y, w, h) in faces:
                 cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
                 roi_gray = gray[y:y + h, x:x + w]
                 roi_color = img[y:y + h, x:x + w]
             cv2.imshow('img', img)
             k = cv2.waitKey(30) & 0xff
             if k == 27:
                 break
         cap.release()
         cv2.destroyAllWindows()
```

Screenshot:



DAY 3: STUDY TRACKERS

- Objective: Study about Trackers in opencv

There are 8 types of trackers in opencv:
CSRT, KCF, BOOSTING, MIL, TLD, GOTURNM, MEDIANFLOW, MOUSSE

OpenCV provides a function called selectROI that pops up a GUI to select bounding boxes because we need to locate objects we want to track in the first frame and the location in simply a bounding box.

DAY 4: IMPLEMENT TRACKERS

- ❖ Objective: To implement opencv object trackers

Code:

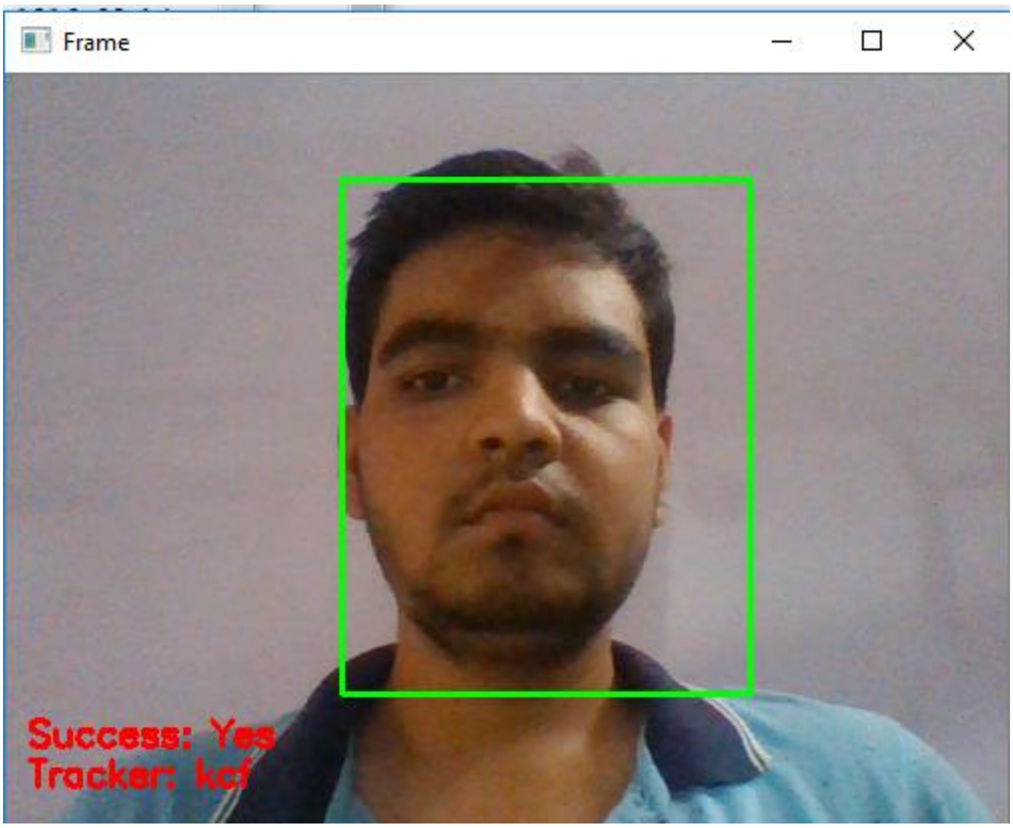
```
from imutils.video import VideoStream
from imutils.video import FPS
import argparse
import imutils
import time
import cv2
success=1



# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", type=str,
 help="path to input video file")
ap.add_argument("-t", "--tracker", type=str, default="kcf",
help="OpenCV object tracker type")
args = vars(ap.parse_args())

# extract the OpenCV version info
(major, minor) = cv2.__version__.split(".")[:2]
```

```
if int(major) == 3 and int(minor) < 3:
        tracker = cv2.Tracker_create(args["tracker"].upper())
else:
        OPENCV_OBJECT_TRACKERS = {
        "csrt": cv2.TrackerCSRT_create,
         "kcf": cv2.TrackerKCF_create,
         "boosting": cv2.TrackerBoosting_create,
        "mil": cv2.TrackerMIL_create,
         "tld": cv2.TrackerTLD_create,
         "medianflow": cv2.TrackerMedianFlow_create,
        "mosse": cv2.TrackerMOSSE_create
 }

tracker = OPENCV_OBJECT_TRACKERS[args["tracker"]]()
initBB = None
if not args.get("video", False):
        print("[INFO] starting video stream...")
        vs = VideoStream(src=0).start()
        time.sleep(1.0)
else:
        vs = cv2.VideoCapture(args["video"])
fps = None


while True:
        frame = vs.read()
        frame = frame[1] if args.get("video", False) else frame
         if frame is None:
                break
        frame = imutils.resize(frame, width=500)
        (H, W) = frame.shape[:2]
         if initBB is not None:
        # grab the new bounding box coordinates of the object
```

```
                (success, box) = tracker.update(frame)
# check to see if the tracking was a success
         if success:
                (x, y, w, h) = [int(v) for v in box]
        cv2.rectangle(frame, (x, y), (x + w, y + h),
        (0, 255, 0), 2)
# update the FPS counter
        fps.update()
        fps.stop()


# initialize the set of information we'll be displaying on the frame
info = [("Tracker", args["tracker"]),
("Success", "Yes" if success else "No")
("FPS", "{:.2f}".format(fps.fps())),
]

# loop over the info tuples and draw them on our frame
for (i, (k, v)) in enumerate(info):
         text = "{}: {}".format(k, v)
         cv2.putText(frame, text, (10, H - ((i * 20) + 20)),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF

if key == ord("s"):
         initBB = cv2.selectROI("Frame", frame, fromCenter=False,
        showCrosshair=True)
        tracker.init(frame, initBB)
         fps = FPS().start()

elif key == ord("q"):
        break
```

```
if not args.get("video", False):
	vs.stop()
else:
	vs.release()
	cv2.destroyAllWindows()
```

Screenshot:



Success: Yes
Tracker: kcf

DAY 5: FACE RECOGNITION

- ❖ Objective: Face Recognition

Code:

1.fr.py:

```
import cv2
import os
import numpy as np

def faceDetection(test_img):
           gray_img=cv2.cvtColor(test_img,cv2.COLOR_BGR2GRAY)
           face_haar_cascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
           faces=face_haar_cascade.detectMultiScale(gray_img,scaleFactor=1.32,minNeighbors=5)
          return faces,gray_img

def labels_for_training_data(directory):
  faces=[]
  faceID=[]

  for path,subdirnames,filenames in os.walk(directory):
    for filename in filenames:
               if filename.startswith("."):
                          print("Skipping system file")
                          continue

               id=os.path.basename(path)
               img_path=os.path.join(path,filename)
               print("img_path:",img_path)
               print("id:",id)
               test_img=cv2.imread(img_path)
               if test_img is None:
                          print("Image not loaded properly")
                          continue
```

```
                faces_rect,gray_img=faceDetection(test_img)
                if len(faces_rect)!=1:
                        continue
                (x,y,w,h)=faces_rect[0]
                roi_gray=gray_img[y:y+w,x:x+h]
                faces.append(roi_gray)
                faceID.append(int(id))
    return faces,faceID



def train_classifier(faces,faceID):
            face_recognizer= cv2.face.LBPHFaceRecognizer_create()
            face_recognizer.train(faces,np.array(faceID))
            return face_recognizer


def draw_rect(test_img,face):
            (x,y,w,h)=face
             cv2.rectangle(test_img,(x,y),(x+w,y+h),(255,0,0),thickness=1)


def put_text(test_img,text,x,y):
        cv2.putText(test_img,text,(x,y),cv2.FONT_HERSHEY_DUPLEX,2,(255,255,255),4)
```

2. resize_images.py:

```
import cv2

import os

import numpy as np


count=0


for path, subdirnames, filenames in os.walk("trainingImages"):


    for filename in filenames:

            if filename.startswith("."):

                    print("Skipping File:",filename)
```

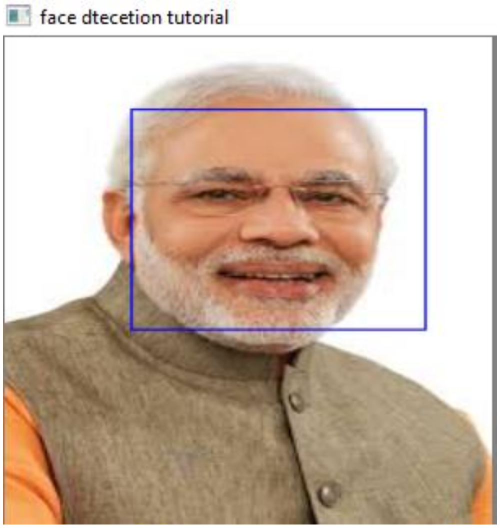
```
                continue
        img_path=os.path.join(path, filename)
        print("img_path",img_path)
        id=os.path.basename(path)
        img = cv2.imread(img_path)
        if img is None:
                print("Image not loaded properly")
                continue
        resized_image = cv2.resize(img, (100, 100))
        new_path="resizedTrainingImages"+"/"+str(id)
        print("desired path is",os.path.join(new_path, "frame%d.jpg" % count))
        cv2.imwrite(os.path.join(new_path, "frame%d.jpg" % count),resized_image)
        count += 1
```

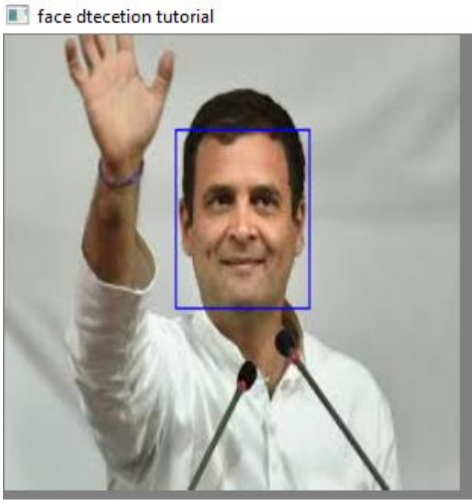
3. test.py:

```
import cv2
import os
import numpy as np
import fr

test_img=cv2.imread('TestImages/test1.jpg')
faces_detected,gray_img=fr.faceDetection(test_img)
print("faces_detected:",faces_detected)

faces,faceID=fr.labels_for_training_data('trainingImages')
face_recognizer=fr.train_classifier(faces,faceID)
face_recognizer.write('trainingData.yml')
```

```
name={0:"Narendra Modi",1:"Rahul Gandhi"}
predicted_name=""
for face in faces_detected:
    (x,y,w,h)=face
    roi_gray=gray_img[y:y+h,x:x+h]
    label,confidence=face_recognizer.predict(roi_gray)
    print("confidence:",confidence)
    print("label:",label)
    fr.draw_rect(test_img,face)
    predicted_name=label
    if(confidence>37):
        continue
   fr.put_text(test_img,predicted_name,100,100)

resized_img=cv2.resize(test_img,(300,300))
cv2.imshow("face dtecetion tutorial",resized_img)
cv2.waitKey(0)
cv2.destroyAllWindows
```

Screenshots:





FUTURE SCOPE

1.Can be used with Raspberry pi for safety detection purposes.

2.Can be used with drones to track real time person.

3.Can be used to detect over speeding vehicles in smart traffic management system

REFERENCES

1) OpenCV

https://www.pyimagesearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv/

https://docs.opencv.org/2.4/doc/tutorials/introduction/table_of_content_introduction/table_of_content_introduction.html#table-of-content-introduction

2) Face Detection using Haar Cascades

https://docs.opencv.org/2.4/doc/tutorials/introduction/table_of_content_introduction/table_of_content_introduction.html#table-of-content-introduction/

https://docs.opencv.org/trunk/db/d28/tutorial_cascade_classifier.html

3) Open CV Trackers

https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking

4) face Recognition

http://hanzratech.in/2015/02/03/face-recognition-using-opencv.html