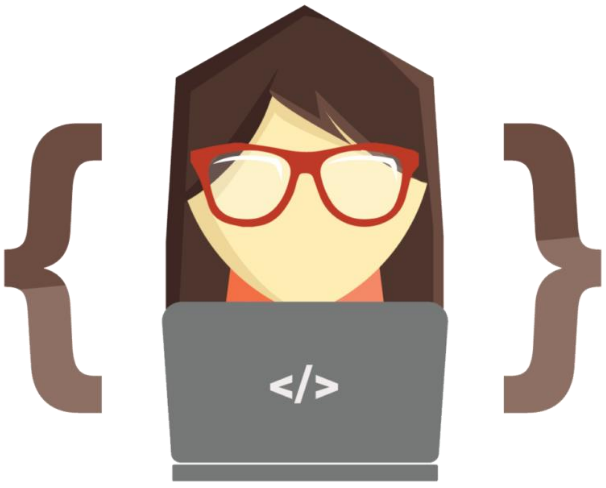


FACIAL RECOGNITION AND TRACKING

A DAY WISE PROJECT REPORT

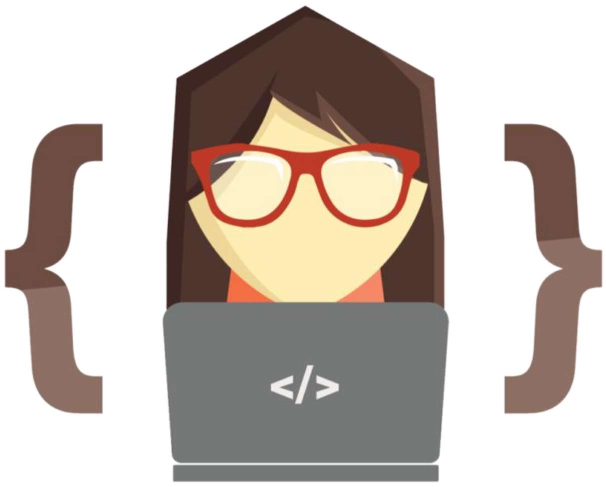
GIRL SCRIPT DEVELOPER TECH CAMP HACK-IN PROJECT

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

HARSH PATEL

Sardar Vallabhbhai Patel Institute of Technology

ACKNOWLEDGEMENT



I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to our Project Mentor, Maitree Rawat, whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project.

Furthermore I would also like to acknowledge with much appreciation the crucial role of the entire team of GirlScript Jaipur, for guiding the mentees in achieving the goal. I have to appreciate the guidance given by the GirlScript Jaipur Team in our project presentation that has improved our presentation skills thanks to their comment and advices.

ABSTRACT

With an Initiative by Girlscript Jaipur for Hack-In week, the Facial Recognition Project with OpenCV stands as an important aspect for future innovation in Facial Recognition Model Training and AI Race, Haar cascade algorithm combined with three additional weak classifiers is proposed in this Report. The three weak classifiers are based on skin hue histogram matching, eyes detection and mouth detection. First, images of people are processed by a primitive Haar cascade classifier, nearly without wrong human face rejection (very low rate of false negative) but with some wrong acceptance (false positive). Secondly, to get rid of these wrongly accepted non-human faces, a weak classifier based on face skin hue histogram matching is applied and a majority of non-human faces are removed. OpenCV is used alongside different trackers having different time complexity of algorithms used by them, from which KCF and CSRT are widely used here.

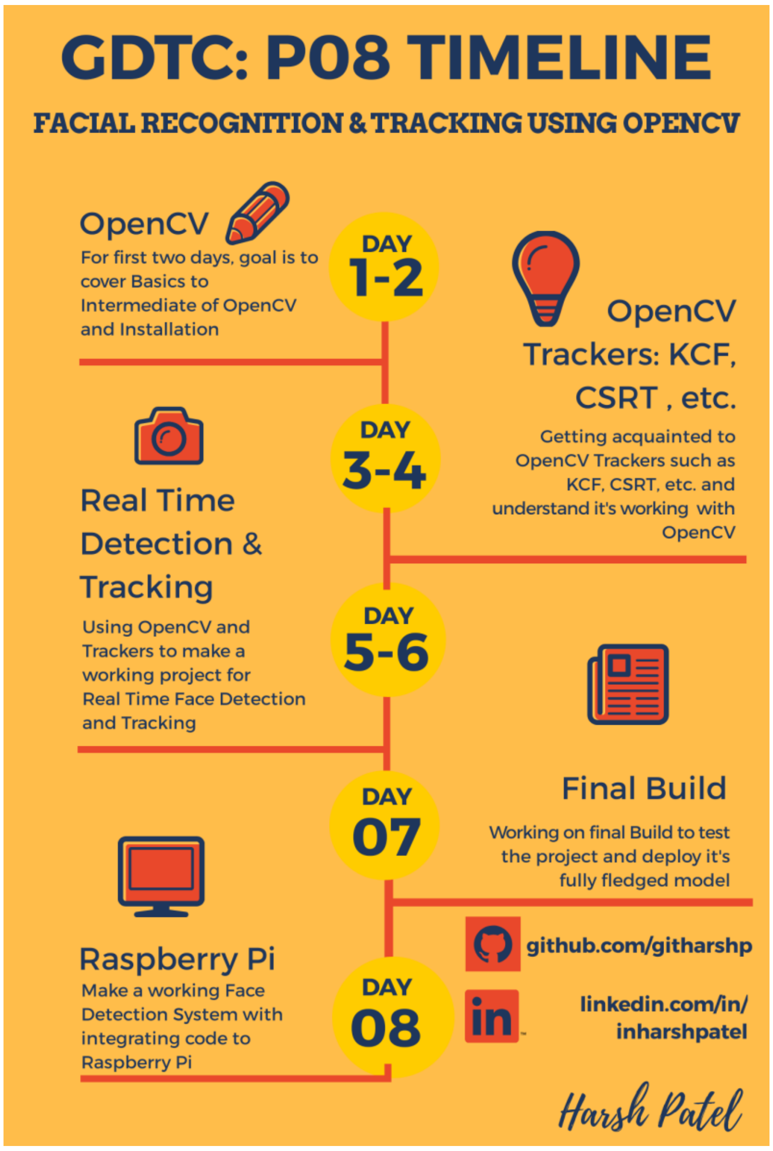
TECHNOLOGY STACK

1. PYTHON 3.7.3
2. OPENCV
3. IMUTILS VIDEO
4. PIP3 (PACKAGE MANAGEMENT SYSTEM)
5. PIL (PYTHON IMAGING LIBRARY)
6. MULTI-OBJECT TRACKER (BOOSTING, KCF, MIL, TLD, MEDIANFLOW, GOTURN, MOOSE, CSRT)
7. VSCODE (EDITOR)

DAY 1: CREATING TIMELINE

❖ Objective: To establish a tentative flow of work for Hack-In week, so that each topic can be covered with proper theoretical explanation and practical application

❖ Timeline Image:



GDTC: P08 TIMELINE

FACIAL RECOGNITION & TRACKING USING OPENCV

DAY 1-2

OpenCV

For first two days, goal is to cover Basics to Intermediate of OpenCV and Installation

DAY 3-4

OpenCV Trackers: KCF, CSRT , etc.

Getting acquainted to OpenCV Trackers such as KCF, CSRT, etc. and understand it's working with OpenCV

DAY 5-6

Real Time Detection & Tracking

Using OpenCV and Trackers to make a working project for Real Time Face Detection and Tracking

DAY 07

Final Build

Working on final Build to test the project and deploy it's fully fledged model

DAY 08

Raspberry Pi

Make a working Face Detection System with integrating code to Raspberry Pi

github.com/githarshp

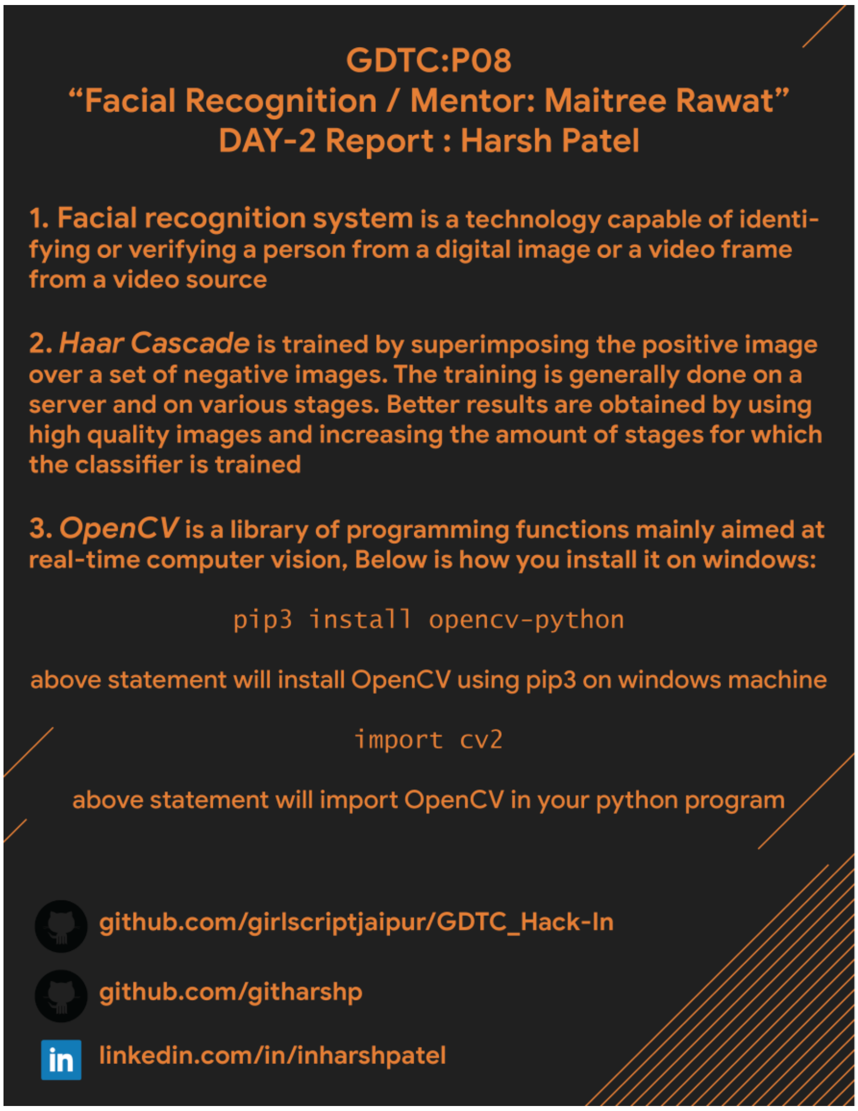
linkedin.com/in/inharshpatel

Harsh Patel

DAY 2: OPENCV

❖ Objective: Study about OpenCV to understand it's usage in Face Recognition using Python Programming Language, and also understand the process of Haar Cascade used during Face Recognition process with OpenCV

❖ Images:



GDTC:P08
"Facial Recognition / Mentor: Maitree Rawat"
DAY-2 Report : Harsh Patel

1. Facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source

2. *Haar Cascade* is trained by superimposing the positive image over a set of negative images. The training is generally done on a server and on various stages. Better results are obtained by using high quality images and increasing the amount of stages for which the classifier is trained

3. *OpenCV* is a library of programming functions mainly aimed at real-time computer vision, Below is how you install it on windows:

```
pip3 install opencv-python
```

above statement will install OpenCV using pip3 on windows machine

```
import cv2
```

above statement will import OpenCV in your python program

github.com/girlscriptjaipur/GDTC_Hack-In

github.com/githarshp

linkedin.com/in/inharshpatel

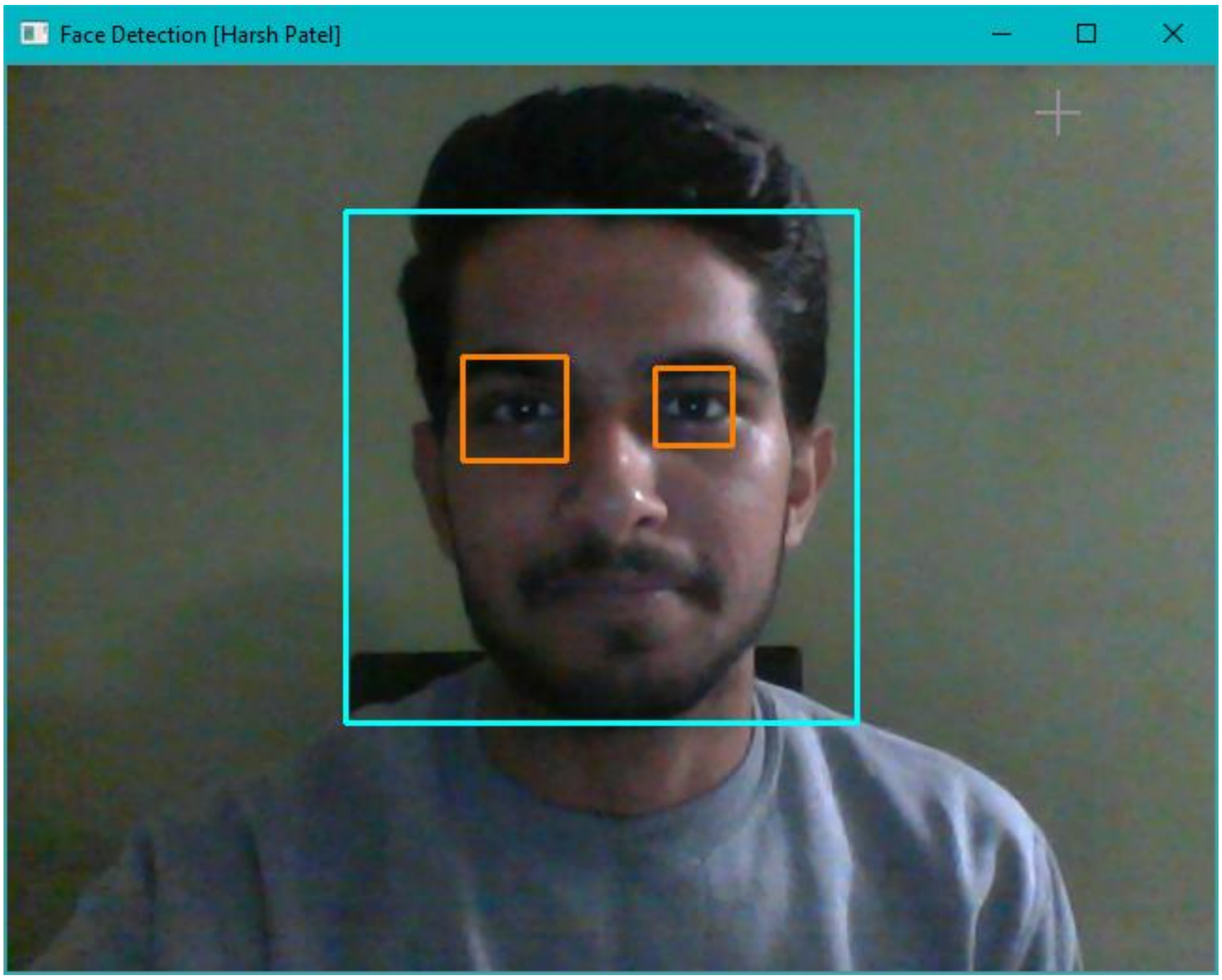
❖ Code:

```
# GDTC:P08 [Facial Recognition] OpenCV program to detect face in real time
[Harsh Patel]
# Reference: https://github.com/Itseez/opencv/blob/master
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Trained XML file for detecting eyes
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

# capture frames from a camera
cap = cv2.VideoCapture(0)

# loop runs if capturing has been initialized.
while 1:

    # reads frames from a camera
    ret, img = cap.read()

    # convert to gray scale of each frames
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Detects faces of different sizes in the input image
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:
        # To draw a rectangle in a face
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]

        # Detects eyes of different sizes in the input image
        eyes = eye_cascade.detectMultiScale(roi_gray)

        #To draw a rectangle in eyes
        for (ex,ey,ew,eh) in eyes:
            cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,127,255),2)

    # Display an image in a window
    cv2.imshow('Face Detection [Harsh Patel]',img)

    # Wait for Esc key to stop
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
```

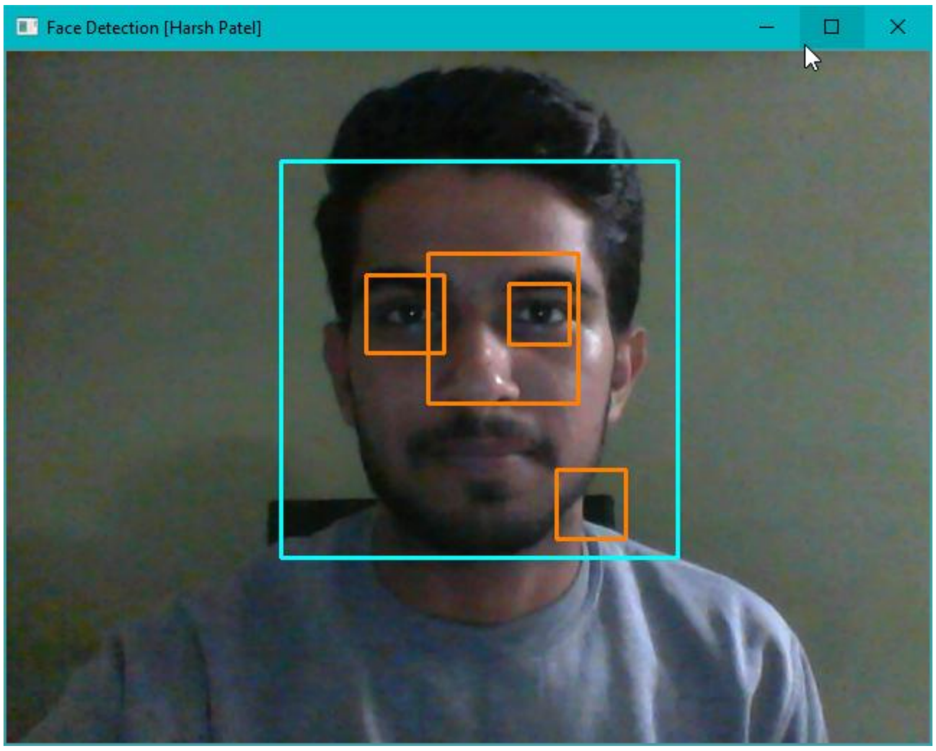
```
# Close the window
cap.release()

# De-allocate any associated memory usage
cv2.destroyAllWindows()
```

❖ Sample Output Screenshots:

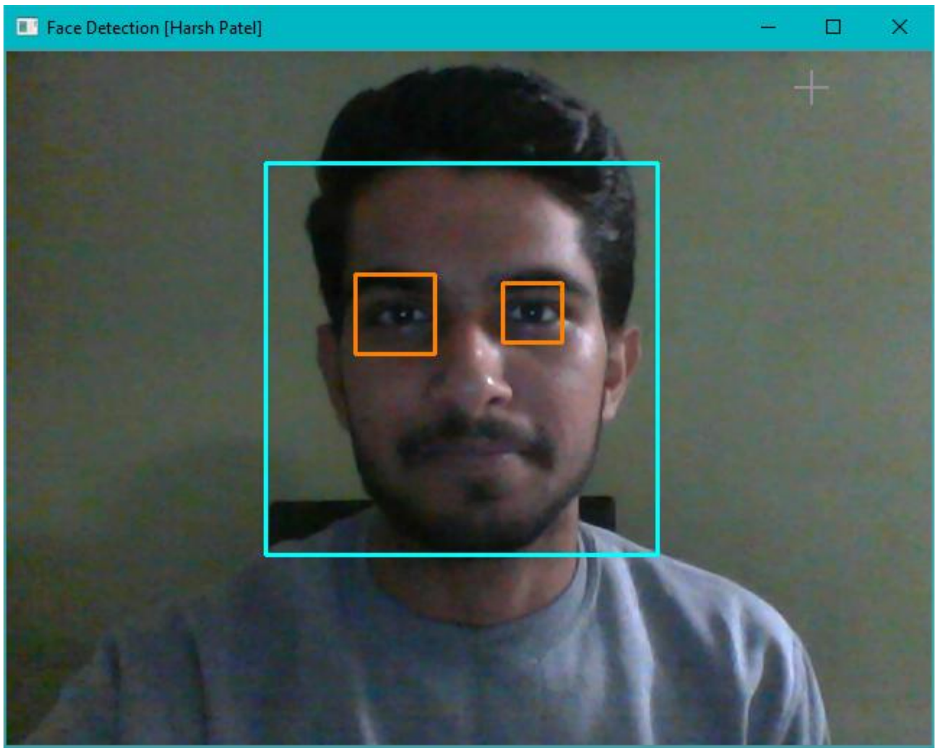
```
# Final Output with Improved Module for Face Detection
```



```
# Output 2 with glitch in Face Detection Module
```



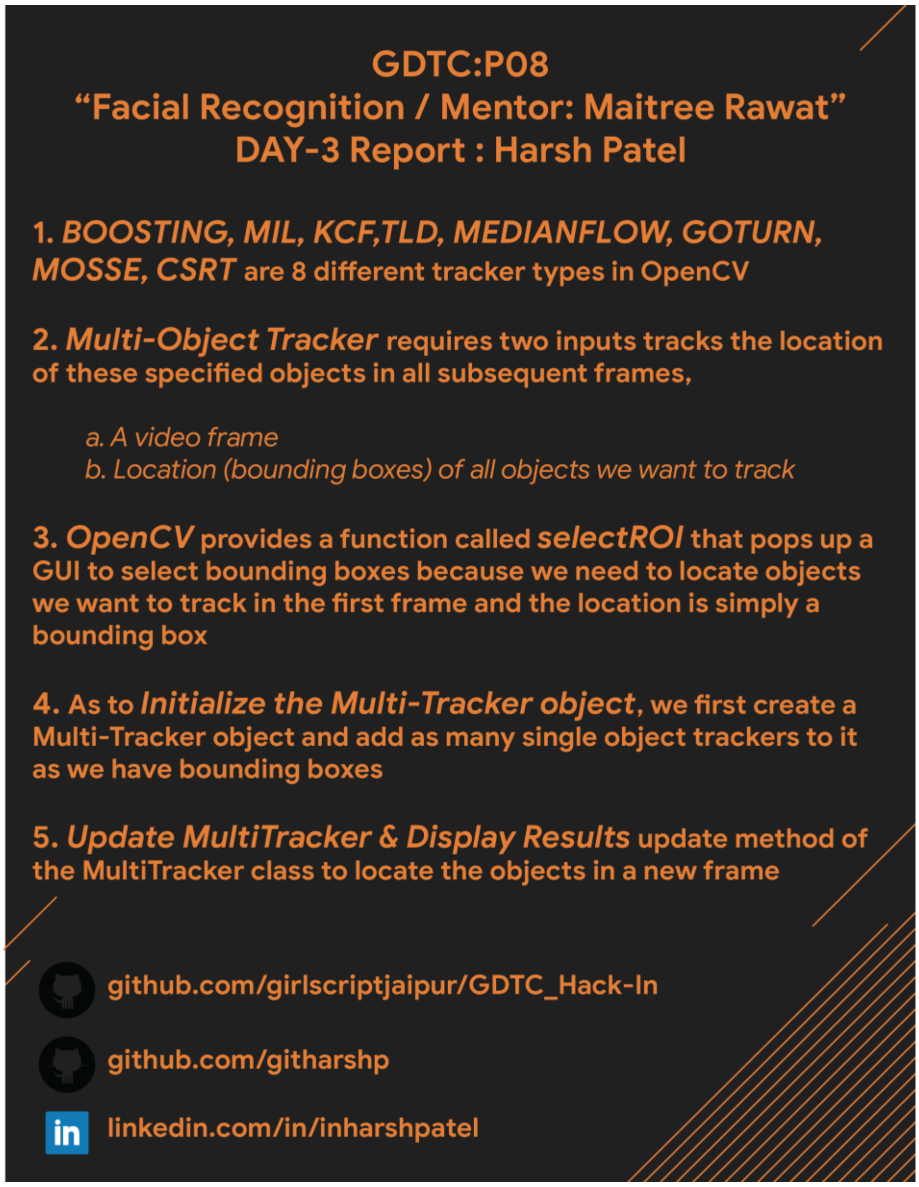
```
# Output 3 with minor ongoing improvements for Face Detection
```



DAY 3: OPENCV TRACKERS

- ❖ Objective: Learn about the usage of 8 different type of trackers with OpenCV and study the procedure for it's application using Python Programming Language

- ❖ Images:



GDTC:P08
"Facial Recognition / Mentor: Maitree Rawat"
DAY-3 Report : Harsh Patel

1. *BOOSTING, MIL, KCF,TLD, MEDIANFLOW, GOTURN, MOSSE, CSRT* are 8 different tracker types in OpenCV

2. *Multi-Object Tracker* requires two inputs tracks the location of these specified objects in all subsequent frames,

   a. A video frame
   b. Location (bounding boxes) of all objects we want to track

3. *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 is simply a bounding box

4. As to *Initialize the Multi-Tracker object*, we first create a Multi-Tracker object and add as many single object trackers to it as we have bounding boxes

5. *Update MultiTracker & Display Results* update method of the MultiTracker class to locate the objects in a new frame

github.com/girlscriptjaipur/GDTC_Hack-In

github.com/githarshp

linkedin.com/in/inharshpatel

DAY 4: OPENCV TRACKERS APPLICATION & USAGE

- ❖ Objective: Using OpenCV Trackers to Track Objects using passive file path video file and also live stream camera

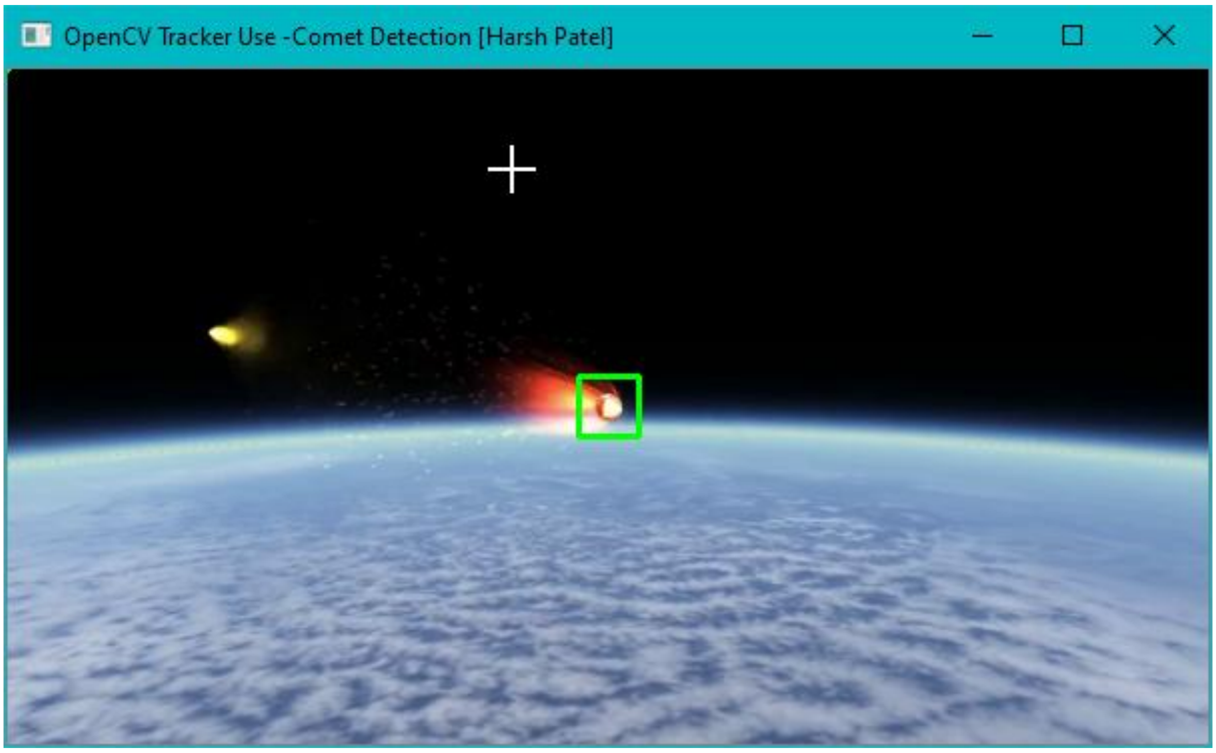
- ❖ Code:

```
# OpenCV Object Tracker usage for Comet Detection Program [Harsh Patel]


# import the necessary packages
from imutils.video import VideoStream
import argparse
import imutils
import time
import cv2

# 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())

# initialize a dictionary that maps strings to their corresponding
# OpenCV object tracker implementations
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
}

# initialize OpenCV's special multi-object tracker
```

```
trackers = cv2.MultiTracker_create()

# if a video path was not supplied, grab the reference to the web cam
if not args.get("video", False):
    print("[INFO] starting video stream...")
    vs = VideoStream(src=0).start()
    time.sleep(1.0)

# otherwise, grab a reference to the video file
else:
    vs = cv2.VideoCapture(args["video"])

# loop over frames from the video stream
while True:
    # grab the current frame, then handle if we are using a
    # VideoStream or VideoCapture object
    frame = vs.read()
    frame = frame[1] if args.get("video", False) else frame

    # check to see if we have reached the end of the stream
    if frame is None:
        break

    # resize the frame (so we can process it faster)
    frame = imutils.resize(frame, width=600)

    # grab the updated bounding box coordinates (if any) for each
    # object that is being tracked
    (success, boxes) = trackers.update(frame)

    # loop over the bounding boxes and draw then on the frame
    for box in boxes:
        (x, y, w, h) = [int(v) for v in box]
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # show the output frame
    cv2.imshow("OpenCV Tracker Use -Comet Detection [Harsh Patel]", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the 's' key is selected, we are going to "select" a bounding
    # box to track
    if key == ord("s"):
        # select the bounding box of the object we want to track (make
        # sure you press ENTER or SPACE after selecting the ROI)
        box = cv2.selectROI("Frame", frame, fromCenter=False,
            showCrosshair=True)

        # create a new object tracker for the bounding box and add it
        # to our multi-object tracker
        tracker = OPENCV_OBJECT_TRACKERS[args["tracker"]]()
        trackers.add(tracker, frame, box)

    # if the `q` key was pressed, break from the loop
    elif key == ord("q"):
        break

# if we are using a webcam, release the pointer
if not args.get("video", False):
    vs.stop()

# otherwise, release the file pointer
else:
    vs.release()

# close all windows
cv2.destroyAllWindows()
```

❖ Sample Output Screenshots:

```
# OpenCV Tracker Use for Comet Detection Model
```



DAY 5:FINAL BUILD [FACE DETECTION / LIVE STREAM DETECTION]

- ❖ Objective: Improve the previously used code to use OpenCV for live stream face detection in real time

- ❖ Code:

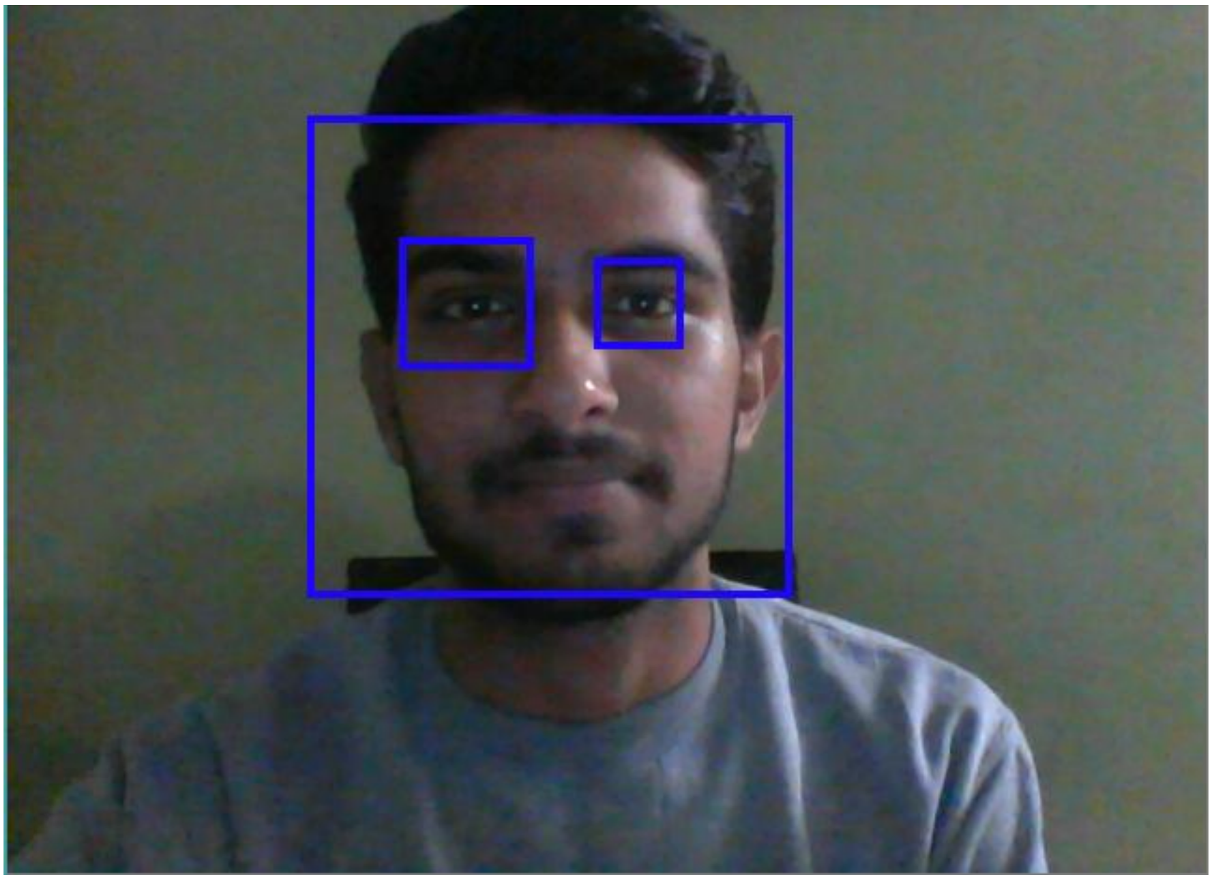
```python
# Face Detection Improvement with Live Stream Face Recognition [Harsh Patel]

import cv2
import imutils


# using the Haar Cascade Classifier
faceCascade = cv2.CascadeClassifier('../day
2/haarcascade_frontalface_default.xml')

id = input("Enter the Face Recognition ID:")
sampleNum = 0
cap = cv2.VideoCapture(0)

while True:
   ret, image = cap.read()
   image = imutils.resize(image, height=300)
   gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

   # Detect faces in the image
   faces = faceCascade.detectMultiScale(
       gray,
       scaleFactor=1.2,
       minNeighbors=5,
       minSize=(30, 30),
       flags=cv2.CASCADE_SCALE_IMAGE
   )

   print("Found {0} faces!".format(len(faces)))
   # Draw a rectangle around the faces
   for (x, y, w, h) in faces:
       sampleNum = sampleNum + 1
       cv2.imwrite("dataset/user." + str(id) + "." + str(sampleNum) + ".jpg",
gray[y:y + h, x:x + w])
       cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
       cv2.waitKey(100)

   cv2.imshow("Faces found", image)
   cv2.waitKey(1)
   if (sampleNum > 40):
       break
cv2.destroyAllWindows()


# Training the Module for Facial Recognition

import os
import numpy as np
import cv2
from PIL import Image

recognizer = cv2.face.LBPHFaceRecognizer_create()
path = 'dataset'


def getImageWithID(path):
   imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
   faces = []
   IDs = []
   for imagePath in imagePaths:
       faceImg = Image.open(imagePath).convert('L')
       faceNp = np.array(faceImg, 'uint8')
       ID = int(os.path.split(imagePath)[-1].split('.')[1])
       faces.append(faceNp)
       IDs.append(ID)
       cv2.imshow("traning", faceNp)
       cv2.waitKey(10)
   return IDs, faces


Ids, faces = getImageWithID(path)
recognizer.train(faces, np.array(Ids))
recognizer.save('recognizer/trainningData.yml')
cv2.destroyAllWindows()
```

```
# Recognizing the Face impressions saved and trying for Face Recognition


import cv2

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('recognizer\\trainningData.yml')
id = 0
font = cv2.FONT_HERSHEY_SIMPLEX
names = []
cap = cv2.VideoCapture(0)
frame_num = 1
faceCascade = cv2.CascadeClassifier('../day
2/haarcascade_frontalface_default.xml')

while True:
   ret, image = cap.read()
   # Create the haar cascade
   gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
   if frame_num == 1:

       # Detect faces in the image
       faces = faceCascade.detectMultiScale(
           gray,
           scaleFactor=1.2,
           minNeighbors=5,
           minSize=(30, 30),
           flags=cv2.CASCADE_SCALE_IMAGE
       )

       print("Found {0} faces!".format(len(faces)))

       # Draw a rectangle around the faces
       for (x, y, w, h) in faces:
           cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
           tracker = cv2.TrackerCSRT_create()
           tracker.init(image, (x, y, w, h))

   (success, box) = tracker.update(image)
   # check to see if the tracking was a success
   if success:
       frame_num = frame_num + 1
       (x, y, w, h) = [int(v) for v in box]
       cv2.rectangle(image, (x, y), (x + w, y + h),
                     (255, 0, 0), 2)
       id, conf = recognizer.predict(gray[y:y + h, x:x + w])
       print(str(id) + "  matched")
```

```
        cv2.putText(image, names[int(id)], (x, y + h + 20), font, .6, (0, 255,
0), 2)

    cv2.imshow("Faces found", image)
    if cv2.waitKey(1) == 27:
        break
cv2.destroyAllWindows()
```

❖ Sample Output Screenshots:

```
                 # Live Stream Face Recognition Model with OpenCV
```



DAY 6: FINAL BUILD [OBJECT TRACKING]

- ❖ Objective: Finalized Improved Module for Multi-Object Tracking with OpenCV Trackers for a Space Shuttle Thrust passive video file path method

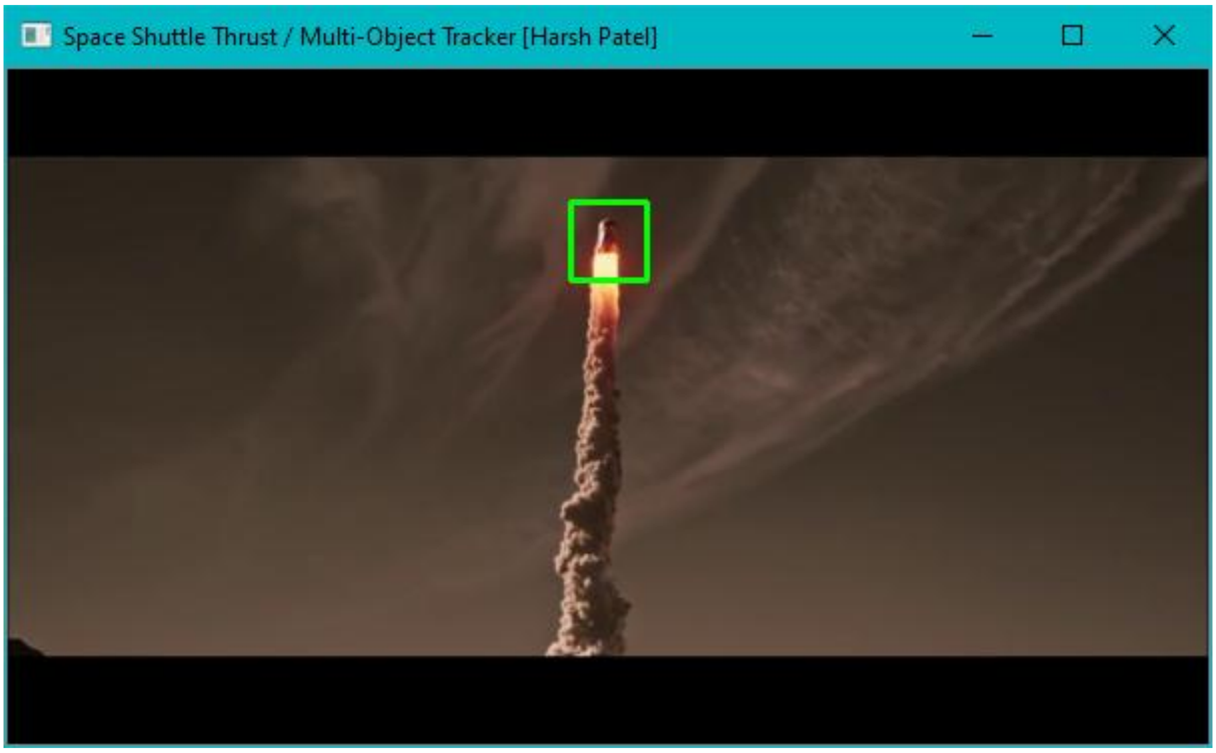
- ❖ Code:

```
# Space Thrust Tracking by OpenCV Multi-Object Tracking


# import the necessary packages
from imutils.video import VideoStream
import argparse
import imutils
import time
import cv2

# 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())

# initialize a dictionary that maps strings to their corresponding
# OpenCV object tracker implementations
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
}

# initialize OpenCV's special multi-object tracker
```

```
trackers = cv2.MultiTracker_create()

# if a video path was not supplied, grab the reference to the web cam
if not args.get("video", False):
    print("[INFO] starting video stream...")
    vs = VideoStream(src=0).start()
    time.sleep(1.0)

# otherwise, grab a reference to the video file
else:
    vs = cv2.VideoCapture(args["video"])

# loop over frames from the video stream
while True:
    # grab the current frame, then handle if we are using a
    # VideoStream or VideoCapture object
    frame = vs.read()
    frame = frame[1] if args.get("video", False) else frame

    # check to see if we have reached the end of the stream
    if frame is None:
        break

    # resize the frame (so we can process it faster)
    frame = imutils.resize(frame, width=600)

    # grab the updated bounding box coordinates (if any) for each
    # object that is being tracked
    (success, boxes) = trackers.update(frame)

    # loop over the bounding boxes and draw then on the frame
    for box in boxes:
        (x, y, w, h) = [int(v) for v in box]
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # show the output frame
    cv2.imshow("OpenCV Tracker Use -Comet Detection [Harsh Patel]", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the 's' key is selected, we are going to "select" a bounding
    # box to track
    if key == ord("s"):
        # select the bounding box of the object we want to track (make
        # sure you press ENTER or SPACE after selecting the ROI)
        box = cv2.selectROI("Frame", frame, fromCenter=False,
            showCrosshair=True)

        # create a new object tracker for the bounding box and add it
        # to our multi-object tracker
        tracker = OPENCV_OBJECT_TRACKERS[args["tracker"]]()
        trackers.add(tracker, frame, box)

    # if the `q` key was pressed, break from the loop
    elif key == ord("q"):
        break

# if we are using a webcam, release the pointer
if not args.get("video", False):
    vs.stop()

# otherwise, release the file pointer
else:
    vs.release()

# close all windows
cv2.destroyAllWindows()
```

❖ Sample Output Screenshots:

```
            # Space Shuttle Thrust OpenCV Multi-Object Tracking
```



FUTURE SCOPE

1. Can be used with Raspberry pi with web cam to detect robbery.
2. Can be used with Quadrone to track real time person.
3. Can be used to detect over speeding vehicles in smart traffic management system
4. Can be used for detecting different objects and using with Machine Learning Algorithm
5. Can be used for Home/Office security management

REFERENCES

1. https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/
2. https://realpython.com/face-detection-in-python-using-a-webcam/
3. https://www.datacamp.com/community/tutorials/face-detection-python-opencv
4. https://www.pyimagesearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv/
5. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_table_of_contents_imgproc/py_table_of_contents_imgproc.html
6. https://pythonprogramming.net/haar-cascade-face-eye-detection-python-opencv-tutorial/
7. https://ieeexplore.ieee.org/abstract/document/8265863
8. https://www.learnopencv.com/multitracker-multiple-object-tracking-using-opencv-c-python/
9. https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/