De-Identification of Facial Images to preserve Data Privacy using L-Diversity over images (Averaging of images)

A PROJECT REPORT

Submitted by

18BCE0213	NAVYAA SHARMA
18BCE0296	P.G.V. SAIKARTHIK
18BCI0247	ROHAN ALLEN

CSE3501

Information Security Analysis and Audit

Under the guidance of

Dr. Anil Kumar K

SCOPE

VIT University, Vellore.

Department of ….

SCHOOL OF COMPUTER SCIENCE

AND ENGINEERING

November 2020

Contents

ABSTRACT

With the advent of the Internet, the amount of data being shared over the internet has increased drastically over the recent years. With this, comes the need of preventing hackers/adversaries from accessing sensitive information. In this project, our aim is to implement one such method of anonymizing facial images using de-identification techniques. There are many methods available to implement this such as blurring out the image, hiding certain facial features or modifying the facial features by adding some noise, calculated by taking out average values of certain facial features such as skin-color, shape etc. We will be using averaging over images to reduce the granularity of our image data.

1. INTRODUCTION

In recent times, many software MNCs have implemented facial recognition software for services like surveillance, photo-tagging, maintaining identity records etc. So, the question has arisen whether our identity is safe from leaks of sensitive data and are the companies following proper safety guidelines. Image recognition has become an integral part of many organisation's operations such as facebook implementing tagging people in uploaded pics, government installing surveillance cameras, cctv cameras in banks and other institutions, etc. Thus there is a need of protecting this info from untrusted sources by privatizing the data.

1.1 PURPOSE

With video surveillance becoming an integral part of our security infrastructure, privacy rights are beginning to gain importance. The key concern is the fact that private citizens, who are not suspects, are being recorded and recordings archived through the use of video surveillance systems. Such a record-everything-and process-later approach has serious privacy implications. The same privacy issues arise when surveillance cameras routinely record highway traffic as vehicle tags are recorded. The solution of removing the identities by blurring/blackening the portions of video is not acceptable to security personnel as they may have legitimate need to review the videos. On the contrary, leaving the videos with identities of people and vehicles public is a breach of privacy. A solution to the problem is selective encryption of portions of the video that reveal identity (e.g., faces, vehicle tags) in surveillance applications. Regions of a video can be encrypted to ensure privacy and still allow decryption for legitimate security needs

at any time in the future. The goals of the video surveillance are still met as selective encryption allows monitoring the activities without knowing the identities of those being monitored. When a suspicious activity needs to be investigated, the identities can be uncovered with proper authorization. The few existing solutions are specific to video and image compression algorithms used and require modification to the video encoders.

2.BACKGROUND WORK

2.1 TECHNOLOGY REQUIREMENTS:-

We are going to use Machine Learning Algorithms and run our program using python. We will run our python program on Spyder

.

• Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging, and introspection features.

• Spyder is also pre-installed in Anaconda Navigator, which is included in Anaconda.

• Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment.
We will use some of the inbuilt libraries of python like numpy,cv2,argparse,dlib,etc.

• Numpy:- NumPy is a python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform, and matrices.

• Cv2:- OpenCV-Python is a library of Python bindings designed to solve computer vision problems. cv2. imread() method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

• ArgParse:- argparse — Parser for command-line options, arguments and subcommands. The argparse module makes it easy to write user-friendly command line interfaces.

2.2 LITERATURE SURVEY:

Compression Independent Reversible Encryption for Privacy in Video Surveillance[1]

With video surveillance becoming an integral part of our security infrastructure, privacy rights are beginning to gain importance. The key concern is the fact that private citizens, who are not suspects, are being recorded and recordings archived through the use of video surveillance systems. Such a record-everything-and-process-later approach has serious privacy implications. The same privacy issues arise when surveillance cameras routinely record highway traffic as vehicle tags are recorded. The solution of removing the identities by blurring/blackening the portions of video is not acceptable to security personnel as they may have legitimate need to review the videos. On the contrary, leaving the videos with identities of people and vehicles public is a breach of privacy. A solution to the problem is selective encryption of portions of the video that reveal identity (e.g., faces, vehicle tags) in surveillance applications. Regions of a video can be encrypted to ensure privacy and still allow decryption for legitimate security needs at anytime in the future. The goals of the video surveillance are still met as selective encryption allows monitoring the activities without knowing the identities of those being monitored. When a suspicious activity needs to be investigated, the identities can be uncovered with proper authorization. The few existing solutions are specific to video and image compression algorithms used and require modification to the video encoders.

Facial expression preserving privacy protection using image melding[2]

An enormous number of images are currently shared through social networking services such as Facebook. These images usually contain the appearance of people and may violate the people's privacy if they are published without permission from each person. To remedy this privacy concern, visual privacy protection, such as blurring, is applied to facial regions of people without permission. However, in addition to image quality degradation, this may spoil the context of the image: If some people are filtered while the others are not, missing facial expression makes comprehension of the image difficult.

Say cheese! Privacy and facial recognition[3]

With many images being shared over the internet and many companies utilizing this data for their services comes the need of implementing guidelines for dealing with data and protecting it. This paper discusses various guidelines under European data protection laws.

Efficient Privacy-Preserving Facial Expression Classification[4]

- This paper proposes an efficient algorithm to perform privacy-preserving (PP) facial expression classification (FEC) in the client-server model. The server holds a database and offers the classification service to the clients. The client uses the service to classify the facial expression (FaE) of subject. It should be noted that the client and server are mutually untrusted parties and they want to perform the classification without revealing their inputs to each other. In contrast to the existing works, which rely on computationally expensive cryptographic operations, this paper proposes a lightweight algorithm based on the randomization technique. The proposed algorithm is validated using the widely used JAFFE and MUG FaE databases. Experimental results demonstrate that the proposed algorithm does not degrade the performance compared to existing works. However, it preserves the privacy of inputs while improving the computational complexity by 120 times and communication complexity by 31 percent against the existing homomorphic cryptography based approach.

Privacy-Preserving Face Recognition[5]

Biometric techniques have advanced over the past years to a reliable means of authentication, which are increasingly deployed in various application domains. In particular, face recognition has been a focus of the research community due to its unobtrusiveness and ease of use: no special sensors are necessary and readily available images of good quality can be used for biometric authentication. The development of new biometric face-recognition systems was mainly driven by two application scenarios

1) To reduce the risk of counterfeiting, modern electronic passports and identification cards contain a chip that stores information about the owner, as well as biometric data in the form of a fingerprint and a photo. While this biometric data is not widely used at the moment, it is anticipated that the digitized photo will allow to automatize identity checks at border crossings or even perform cross-matching against lists of terrorism suspects .

2) The increasing deployment of surveillance cameras in public places sparked interest in the use of face recognition technologies to automatically match faces of people shown on surveillance images against a database of known suspects. Despite massive technical problems that render this application currently infeasible, automatic biometric face recognition systems are still high on the agenda of policy makers .

The ubiquitous use of face biometrics raises important privacy concerns; particularly problematic are scenarios where a face image is automatically matched against a database without the explicit consent of a person (for example in the above-mentioned surveillance scenario), as this allows to trace people against their will. The widespread use of biometrics calls for a careful policy, specifying to which party biometric data is revealed, in particular if biometric matching is performed at a central server or in partly untrusted environments.

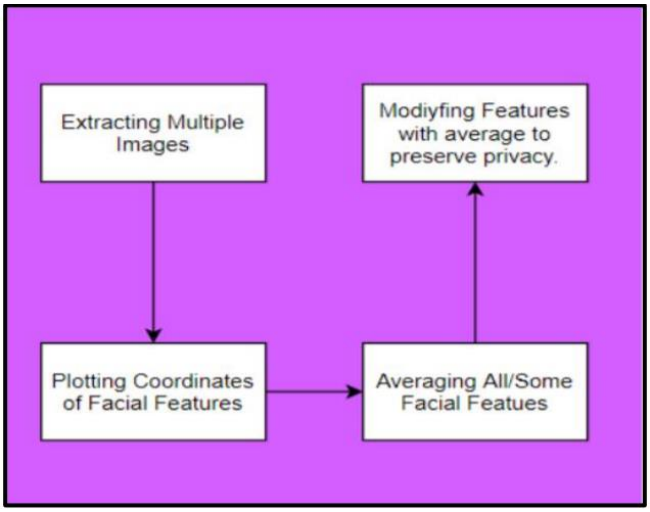
3.PROBLEM DESCRIPTION

3.1 Problem Description

In recent times, many software MNCs have implemented facial recognition software's for services like surveillance, photo-tagging, maintaining identity records etc. So, the question has arisen whether our identity is safe from leaks of sensitive data and are the companies following proper safety guidelines. Image recognition has become an integral part of many organization's operations such as Facebook implementing tagging people in uploaded pictures, government installing surveillance cameras, CCTV cameras in banks and other institutions, etc. Thus there is a need to protect this info from non trusted sources by privatizing the data.

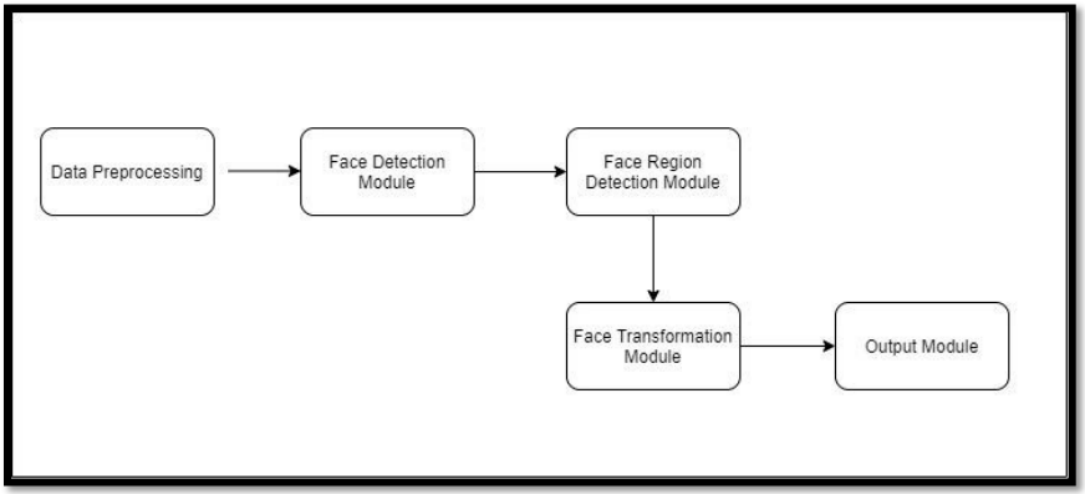
3.2 Working Model

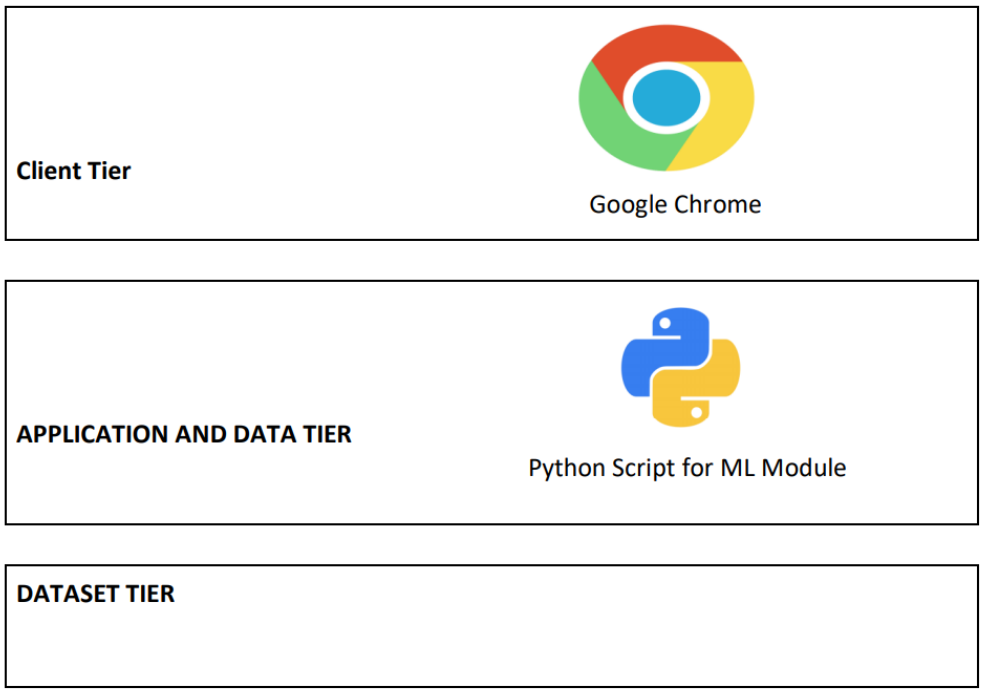
The system functions will go in this order:
1) Extracting Multiple Images
2) Plotting Coordinates of Facial Features
3) Averaging All/Some Facial Feature
4) Modifying Features with average to preserve privacy



3.3 Design Description

1. Architecture Design:

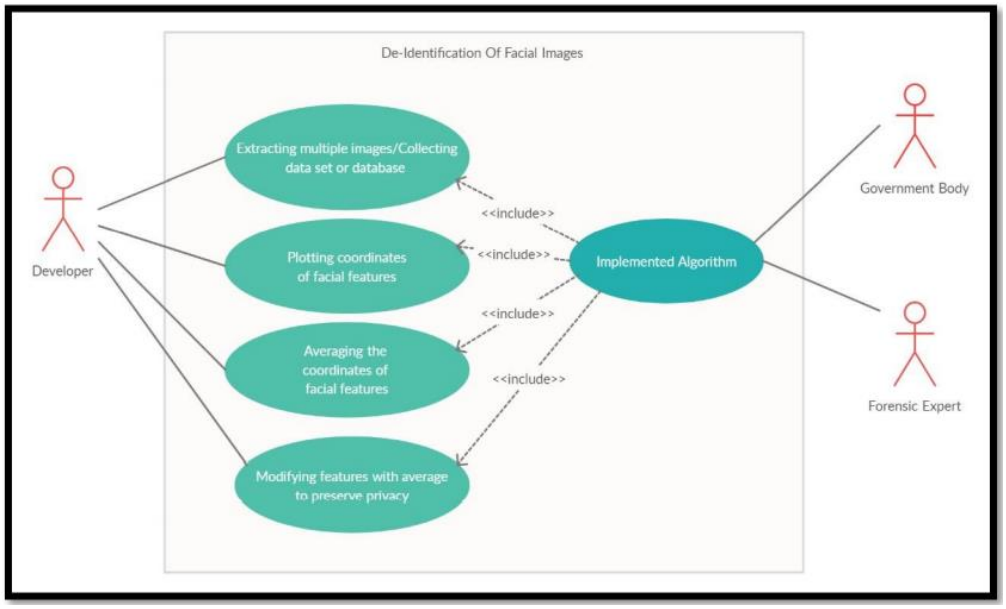




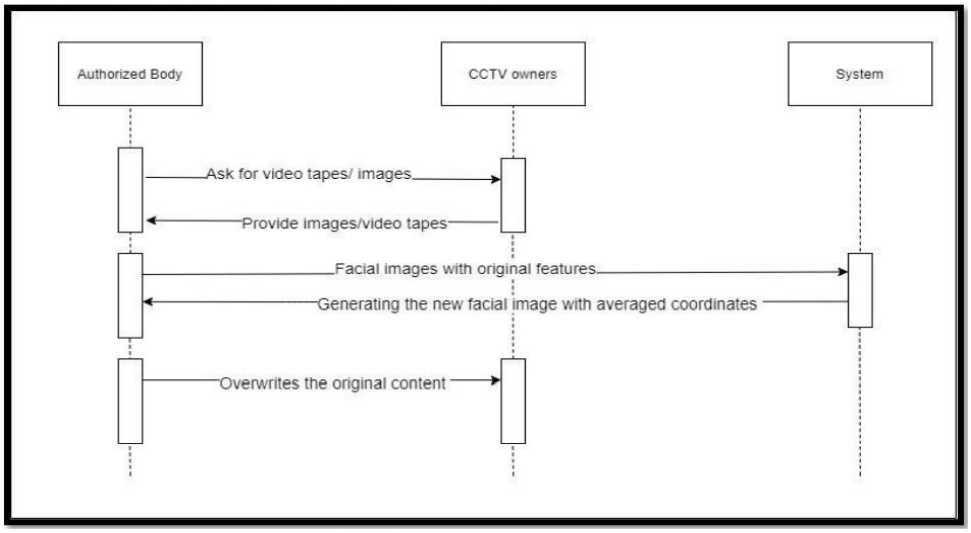
2. Use Case Diagram:

The purposes of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements of the actors.



3. SEQUENCE DIAGRAM



4. IMPLEMENTATION:-

4.1 MODULES

We will be using averaging over images to reduce the granularity of our image data

We will be making use of facial landmark detector implemented inside dlib produces 68 (x, y)-coordinates that map to specific facial structures. These 68 point mappings were obtained by training a shape predictor on the labeled iBUG 300-W dataset.

Below we can visualize what each of these 68 coordinates map to:

The facial expressions are indexed via simple python indexing.

- ❏ The mouth can be accessed through points [48, 68].
- ❏ The right eyebrow through points [17, 22].
- ❏ The left eyebrow through points [22, 27].

- The right eye using [36, 42].
- The left eye with [42, 48].
- The nose using [27, 35].
- And the jaw via [0, 17].

FACIAL_LANDMARKS_IDXS dictionary is encoded with these mappings inside the face_utils of the imultils library.

Resources

We have used the following packages in python:

- NumPy
- OpenCV
- dlib
- imutils

4.2 SOURCE CODE

```
import numpy as np
import cv2
import argparse
import dlib
import imutils
import math



def func(out,inp):
    for (x, y, w, h) in inp:
        for i in range(x,x+w):
            for j in range(y,y+h):
                out.append([i,j])
        cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)

def func1(inp1,inp2,out):
    for i in range(0,300):
        for j in range(0,300):
            a=math.pow(inp1[i][j],2)
            b=math.pow(inp2[i][j],2)
```

```
            y=np.add(a,b)

            z=math.sqrt(y)
            out[i][j] = np.divide(z,2).astype(int)


facial_features_cordinates = {}
faces_array =  []
FACIAL_LANDMARKS_INDEXES = OrderedDict([
    ("Mouth", (48, 68)),
    ("Right_Eyebrow", (17, 22)),
    ("Left_Eyebrow", (22, 27)),
    ("Right_Eye", (36, 42)),
    ("Left_Eye", (42, 48)),
    ("Nose", (27, 35)),
    ("Jaw", (0, 17))
])

ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
     help="path to facial landmark predictor")
ap.add_argument("-i", "--image", required=True,
     help="path to input image")
ap.add_argument("-i2", "--image2", required=True,
     help="path to input image")
ap.add_argument("-i3", "--image3", required=True,
     help="path to input image")
ap.add_argument("-i4", "--image4", required=True,
     help="path to input image")
args = vars(ap.parse_args())


def shape_to_numpy_array(shape, dtype="int"):

    coordinates = np.zeros((68, 2), dtype=dtype)

    for i in range(0, 68):
        coordinates[i] = (shape.part(i).x, shape.part(i).y)

    return coordinates
```

```
def visualize_facial_landmarks(image, shape, colors=None,
alpha=1):

    overlay = image.copy()
    output = image.copy()

    if colors is None:
        colors = [(0,0,0), (0,0,0), (0,0,0),
                  (0,0,0), (0,0,0),
                  (0,0,0), (0,0,0)]

    for (i, name) in enumerate(FACIAL_LANDMARKS_INDEXES.keys()):
        print("\n\n")
        print(i,name,"\n")
        (j, k) = FACIAL_LANDMARKS_INDEXES[name]
        pts = shape[j:k]
        facial_features_cordinates[name] = pts

        if name == "Jaw":
            for l in range(1, len(pts)):
                ptA = tuple(pts[l - 1])
                ptB = tuple(pts[l])
                cv2.line(overlay, ptA, ptB, colors[i], 2)

        else:
            hull = cv2.convexHull(pts)
            cv2.drawContours(overlay, [hull], -1, colors[i], -1)

    cv2.addWeighted(overlay, alpha, output, 1 - alpha, 0,
output)

    print(facial_features_cordinates)
    return output

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])
```

```
image = cv2.imread(args["image"])
image = cv2.resize(image, (300,300) , interpolation =
cv2.INTER_AREA)
face_cascade = cv2.CascadeClassifier('frontface.xml')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.1, 4)

image2 = cv2.imread(args["image2"])
image2 = cv2.resize(image2, (300,300) , interpolation =
cv2.INTER_AREA)
face_cascade = cv2.CascadeClassifier('frontface.xml')
gray2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
faces2 = face_cascade.detectMultiScale(gray2, 1.1, 4)

image3 = cv2.imread(args["image3"])
image3= cv2.resize(image3, (300,300) , interpolation =
cv2.INTER_AREA)
face_cascade = cv2.CascadeClassifier('frontface.xml')
gray3 = cv2.cvtColor(image3, cv2.COLOR_BGR2GRAY)
faces3 = face_cascade.detectMultiScale(gray3, 1.1, 4)

image4 = cv2.imread(args["image4"])
image4 = cv2.resize(image4, (300,300) , interpolation =
cv2.INTER_AREA)
face_cascade = cv2.CascadeClassifier('frontface.xml')
gray4 = cv2.cvtColor(image4, cv2.COLOR_BGR2GRAY)
faces4 = face_cascade.detectMultiScale(gray4, 1.1, 4)

faces_array.append(faces)
faces_array.append(faces2)
faces_array.append(faces3)
faces_array.append(faces4)
faces2 = faces2
faces3 = faces3
pointsf1 = []
pointsf2 =[]
pointsf3 = []
pointsf4=[]
print(faces_array)
func(pointsf1,faces)
func(pointsf2,faces2)
```

```
func(pointsf3,faces3)
func(pointsf4,faces4)
for (x, y, w, h) in faces:
    for i in range(x,x+w):
        for j in range(y,y+h):
            pointsf1.append([i,j])
    cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)
for (x, y, w, h) in faces2:
    for i in range(x,x+w):
        for j in range(y,y+h):
            pointsf2.append([i,j])
    cv2.rectangle(image2, (x, y), (x+w, y+h), (255, 0, 0), 2)
for (x, y, w, h) in faces3:
    for i in range(x,x+w):
        for j in range(y,y+h):
            pointsf3.append([i,j])
    cv2.rectangle(image3, (x, y), (x+w, y+h), (255, 0, 0), 2)
for (x, y, w, h) in faces4:
    for i in range(x,x+w):
        for j in range(y,y+h):
            pointsf4.append([i,j])
    cv2.rectangle(image4, (x, y), (x+w, y+h), (255, 0, 0), 2)

print(np.array(pointsf1).shape)
print(np.array(pointsf2).shape)
print(np.array(pointsf3).shape)
print(np.array(pointsf4).shape)


print(image[0][0])
print(image2[0][0])
print(np.add(image[0][0],image2[0][0]))
print(np.divide(np.add(image[0][0],image2[0][0]),2).astype(int))

func1(gray,gray2,gray2)
func1(gray3,gray4,gray3)


for i in range(0,300):
    for j in range(0,300):
        a=math.pow(gray[i][j],2)
```

```
        b=math.pow(gray2[i][j],2)

        y=np.add(a,b)

        z=math.sqrt(y)
        gray2[i][j] = np.divide(z,2).astype(int)
for i in range(0,300):
    for j in range(0,300):
        a=math.pow(gray3[i][j],2)
        b=math.pow(gray4[i][j],2)

        y=np.add(a,b)

        z=math.sqrt(y)
        gray3[i][j] = np.divide(z,2).astype(int)

cv2.imshow("image",gray);
cv2.waitKey(0);

cv2.imshow("image2",gray2);
cv2.waitKey(0);
cv2.imshow("image3",gray3);
cv2.waitKey(0);
cv2.imshow("image4",gray4);
cv2.waitKey(0);
rects = detector(gray, 1)

for (i, rect) in enumerate(rects):
    shape = predictor(gray, rect)
    shape = shape_to_numpy_array(shape)

    output = visualize_facial_landmarks(image, shape)
    #cv2.imshow("Image", output)
    cv2.waitKey(0)
```

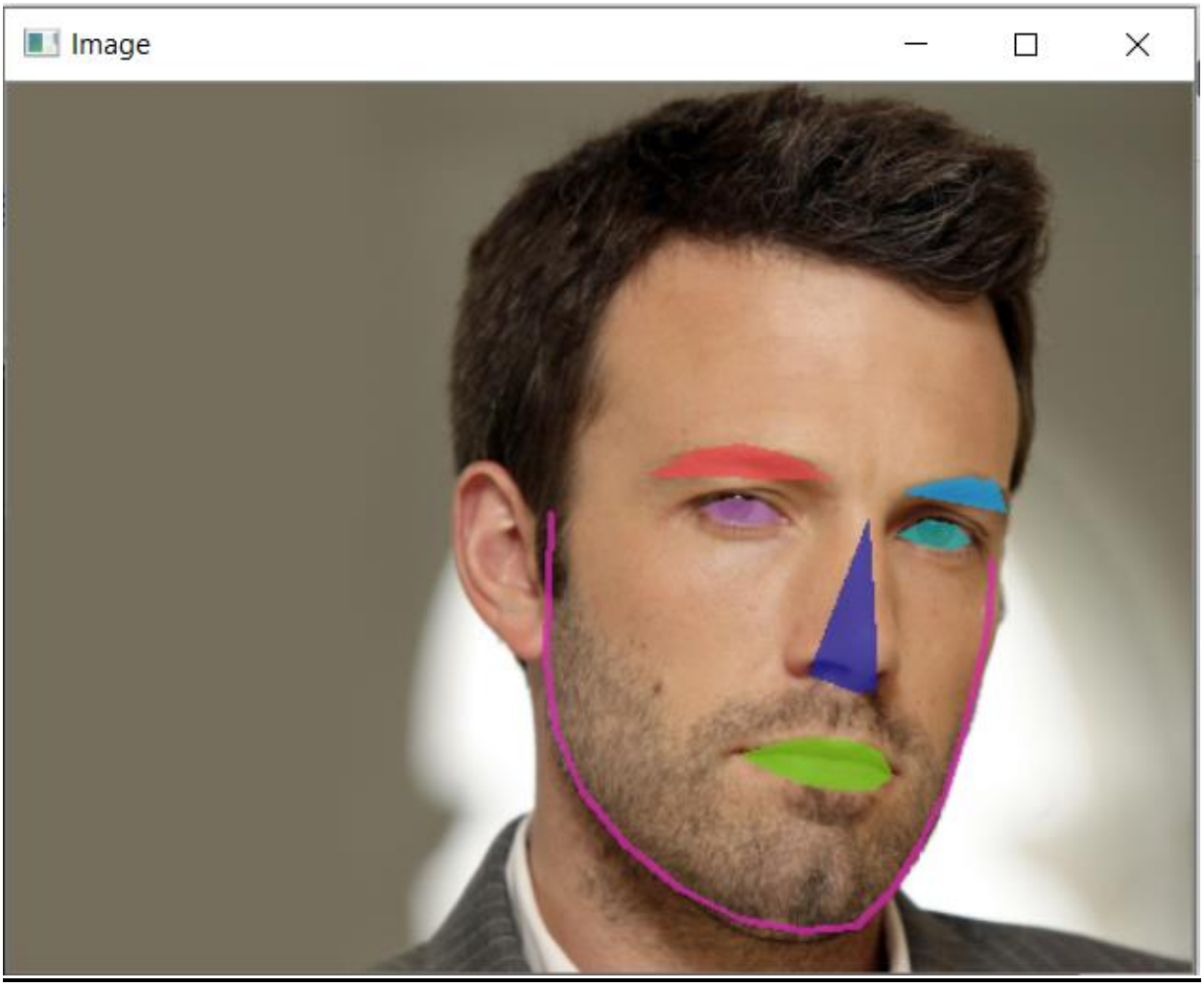
4.3 TEST CASES

TEST CASE ID	TEST CASE OBJECTIVE	PREREQUISITE	STEPS	INPUT DATA	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
TC_01	Extracting Multiple Images	Data preprocessing is done perfectly	Reading files (images) from the folder with proper path	Images	Successful extraction of multiple images	Successful extraction of multiple images	PASS
TC_02	Plotting coordinates of facial features	Proper extraction of multiple images and resolution	Run the portion of the program	Extracted Images with proper resolution	Successful plotting of Coordinates of facial features	Successful plotting of Coordinates of facial features	PASS
TC_03	Averaging the facial features	Proper plotting of Coordinates of facial features	Runt the portion of the program	Proper images with coordinates plotted	Successful formation of averaged image	Successful formation of averaged image	PASS

TC_04	Generating the output	Proper averaged image	Run the portion of code to display	Proper Averaged Photos	Successful display of image	Successful display of image	PASS

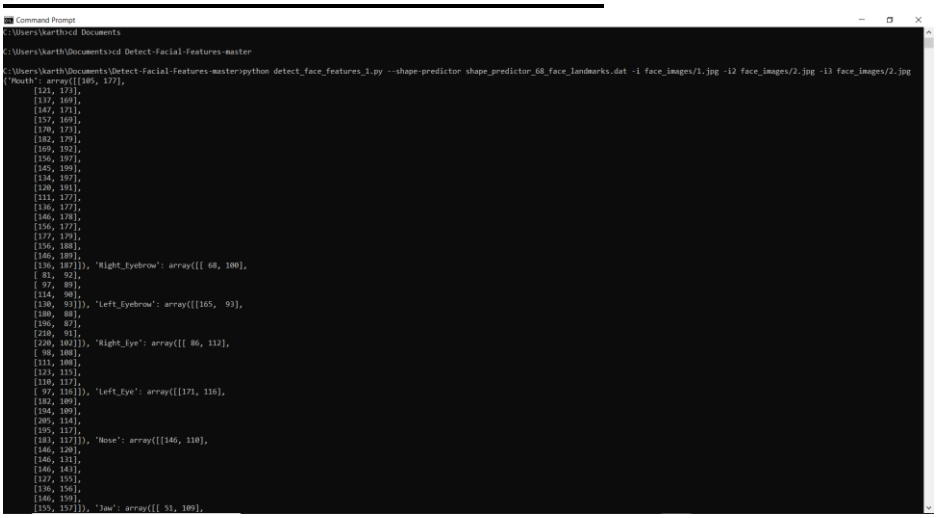
4.4 EXECUTION SNAPSHOTS

1)PLOTTING OF THE FACIAL COORDINATES:-



2)De-Identification of the facial Features.

Below we can see the hiding of the facial features being implemented



5.CONCLUSION

There is a dire need of developing better methods for identity protection in facial images with the increase in the information being generated and being shared over the internet. Older techniques like putting black strips over images and hiding facial features protected the sensitive data but also rendered the images useless. So techniques like averaging pixel features or compressing pixels **maintains the usefulness and identity of the image.**

6.REFERENCES

[1] Compression Independent Reversible Encryption for Privacy in Video Surveillance, Paula Carrillo, Hari Kalva, EURASIP Journal on Information Security,2009

[2] Facial expression preserving privacy protection using image melding, Yuta Nakashima, Tatsuya Koyama, Naokazu Yokoya, Noboru ,Aug.6,2015

[3] Say cheese! Privacy and facial recognition,Ben Buckley Matt Hunter,December 2011

[4] Efficient Privacy-Preserving Facial Expression Classification, Yogachandran Rahulamathavan , Muttukrishnan Rajarajan, 08 July 2015

[5] Privacy-Preserving Face Recognition , Zekeriya ErkinMartin FranzJorge GuajardoStefan KatzenbeisserInald LagendijkTomas Toft, PETS 2009

[6] PRIVACY AND DATA PROTECTION AT THE TIME OF FACIAL RECOGNITION: TOWARDS A NEW RIGHT TO DIGITAL IDENTITY? Shara Monteleone, 2012

[7] Preserving Privacy by De-identifying Facial Images, Elaine Newton Latanya Sweeney Bradley Malin, March 2003

[8] Face de-identification using facial identity preserving features Hehua Chi , Yu Hen Hu, 25 February 2016

[9] "All the better to see you with, my dear": Facial recognition and privacy in online social networks, Norberto Nuno Gomes de Andrade , Aaron Martin , Shara Monteleone , 14 February 2013

[10] ROBUST HUMAN FACE HIDING ENSURING PRIVACY, Isabel Martínez-Ponte, Xavier Desurmont, Jerome Meessen and Jean-François Delaigle,