Preserving Data security in facial images using De-identification

BHANUJ KUMAR-18BCE0903
DIVYA KUMARI-18BCE0914
AMBATI YESWANTH KUMAR-18BCI0200
SRIRAM USHA SREE-18BCB0063

A report submitted for the J component of

CSE3502- Information Security Management

Supervisor: **Dr.D.RUBY**

School of Computer Science and Engineering Vellore institute of Technology, Vellore

May 30, 2021

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 4795

Student Name: Bhanuj Kumar,Divya Kumari,Ambati Yeswanth Kumar,Sriram Usha Sree

Date of Submission:30.05.2021

Signature: Bhanuj Kumar, Divya Kumari, Ambati Yeswanth Kumar, Sriram Usha Sree

Table of Contents

Abstract

With the advent of 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.

Project Specification

Project Name : Preserving Data security in facial images using De-identification

Date : 30.05.2021

Project Developers : Bhanuj, Divya,Yeswanth,Usha

Professor : Ruby D

Purpose : To develop a technology that allows the sharing of video recordings for other valuable purposes, while recording images in public places and reasonably preserving the anonymity of those involved in legal activities.

Description : We used averaging over images to reduce the granularity of our image data. The facial landmark detection implemented inside dlip produces coordinates that map coordinates to specific facial features.

Objective :

1.Highly secured system for de-identification of facial features.

2.Details remaining on the face are minimally distorted.

3.To share only anonymized data.

4.Determining the facial landmarks for the face region.

Functionality :

1.Extracting Multiple Images.

2.Plotting Coordinates of facial features.

3.Averaging some/all Facial features.

4.Modifying features to preserve privacy.

Performance:

1.The overall code of this project is organized in a very efficient way.

2.Best possible techniques have been used to enhance the overall performance of the project.

Language Used: Python

Interface : Jupyter Notebook

1.Introduction

Many software MNCs have recently implemented facial recognition software for services such as surveillance, photo tagging, and identity record keeping. Therefore, the question has been raised as to whether our identities are safe from the leakage of sensitive data and follow appropriate safety guidelines. Image recognition has become an integral part of many organizations' operations, including tagging people on Facebook's uploaded photos, government installing surveillance cameras, and installing CCTV cameras in banks and other institutions. data.

The purpose of this work is to develop a technology that allows the sharing of video recordings for other valuable purposes, while recording images in public places and reasonably preserving the anonymity of those involved in legal activities. The method presented in this white paper does this by digitally modifying the image so that facial recognition software cannot reliably associate a person with the captured image. To fight crime, video surveillance cameras have proliferated in public places like shops, , ATMs, schools, buses, subway stations, and airports.Preliminary ideas were tested such as covering the eyes and nose, reducing the number of pixels in the face by averaging grayscale values from a block of square pixels, and converting grayscale pixels to black and white values. We report on experiments showing that none of these kinds of solutions are effective guards. Facial recognition software can accurately identify what's still left. Faces that are similar to each other have many characteristics left, but the resulting images are averaged in such a way that they are not reliably recognized by facial recognition software.

Idea

The idea is to develop a technology that allows the sharing of video recordings for other valuable purposes, while recording images in public places and reasonably preserving the anonymity of those involved in legal activities.

Expected Result

Highly secured system for preserving data security in facial images .

2.Literature Survey

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

As video surveillance becomes an integral part of your security infrastructure, your privacy rights are becoming more important. The main concern is the fact that civilian citizens, not suspects, are being recorded and recorded using video surveillance systems. The approach of recording all of this and processing it later has a serious impact on privacy. The same privacy concerns arise when surveillance cameras routinely record highway traffic when vehicle tags are recorded. Solutions that remove identities by blurring/blacking parts of the video are not acceptable to security personnel as there may be a legitimate need to review the video. Conversely, revealing the identities of people and vehicles is a violation of privacy. The solution to the problem is the selective encryption of the part of the video that reveals the identity (e.g. face, vehicle tag) in surveillance applications. You can encrypt areas of the video to protect your privacy and allow decryption for legitimate security requirements at any time in the future. The goal of video surveillance is still met, as selective encryption allows you to monitor activity without knowing the identity of the monitored object. If you need to investigate suspicious activity, you can identify yourself with proper authorization. Several existing solutions are specific to the video and image compression algorithm used and require modification of the video encoder.

Facial expression preserving privacy protection using image melding[2]

Currently, a huge number of images are shared through social networking services such as Facebook. These images usually contain a person's appearance and can invade an individual's privacy if posted without the permission of each individual. To address these privacy issues, visual privacy protections such as blurring are applied to areas of people's faces without permission. However, in addition to poor image quality, the context of the image can be compromised. If some people are filtered and others are not, facial expressions are missing, making the image difficult to understand.

Say cheese! Privacy and facial recognition[3]

Since many images are shared over the Internet and many companies use this data for services, you need to implement guidelines to process and protect your data. This document describes the various guidelines under European data protection law.

Efficient Privacy-Preserving Facial Expression Classification[4]

This paper proposes an efficient algorithm for performing privacy (PP) facial expression classification (FEC) in a client-server model. The server holds the database and provides classification services to clients. The client uses the service to classify the subject's facial expressions (FaE). The client and server are parties that can't trust each other and want their input to be done without revealing each other. Unlike conventional work that relies on cryptographic operations that are computationally expensive, this paper proposes a lightweight algorithm based on a randomization technique. The proposed algorithm is using the famous JAFFE and MUG FaE databases. Experimental results show that the proposed algorithm does not degrade the performance compared to the existing work. However, it maintains the privacy of the input while improving computing complexity by 120 times and communication complexity by 31% compared to traditional homogeneous encryption-based approaches.

Privacy-Preserving Face Recognition[5]

Biometrics technology has evolved over the past few years as a reliable means of authentication, and is increasingly being deployed in various application domains. Face recognition, in particular, has been the focus of the research community because it is less conspicuous and easy to use. No special sensor is required and easy-to-use, high-quality images can be used for biometric authentication. The development of a new biometric facial recognition system has been driven mainly by two application scenarios. To reduce the risk of counterfeiting, modern e-passports and ID cards contain chips that store information about the owner and biometric data in the form of fingerprints and photographs. Although this biometric data is currently not widely used, it is expected that digitized photos could automate identification when crossing borders or perform crossmatching against terrorist suspect lists. Increasing placement of surveillance cameras in public places has sparked interest in the use of facial recognition technology to automatically match the faces of people displayed in surveillance images with databases of known suspects. Despite the tremendous technical challenges that make this application currently infeasible, automatic biometric facial recognition systems are still high on the agenda of policy makers. The ubiquitous use of facial biometrics poses important privacy concerns. What is particularly problematic is the scenario where the database and face images are automatically matched without the explicit consent of a person (e.g. the surveillance scenario mentioned above). The widespread use of biometrics requires a prudent policy stating to whom biometric data is disclosed, especially if biometric matching is performed on a central server or in a partially untrusted environment.

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

Indicators for 'health' status of user rights in the digital world, new Facebook Basic privacy settings and facial recognition features are covered in the first part of this article. The Facebook case also constitutes an example of the use of special categories of personal data (biometrics) on social networking sites with legal implications in terms of data protection, privacy and management of users' digital identities. In particular, it is based on the results of the Eurobarometer (EB) on the attitudes of users to the protection of personal data and identity, the recent comments of the Article 29 Working Group (Art29 WP), and the recent reports of Frank, the UN Special Rapporteur. La Rue, On Freedom of Expression (2011), the analysis included in the second part of the paper reveals several shortcomings of the current data protection legal framework, in which a recent proposal by the European Commission on Data Protection proposed regulations on data protection. I will emphasize. It's easy to partially solve.

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

When it comes to video surveillance data sharing, a significant threat to privacy is facial recognition software, such as a driver's license photo database, that can automatically identify known people to track people regardless of suspicion. In this paper, we introduce an algorithm that protects personal privacy in video surveillance data by de-identifying faces so that many facial features remain, but faces cannot be reliably recognized. A trivial solution to de-identifying faces is to blacken each face. This interferes with possible facial recognition but consequently limits its use as all facial details are obscured. Many ad hoc attempts, such as blindfolding or randomly perturbing image pixels, do not interfere with facial recognition because of the robustness of the facial recognition method. This paper presents a new privacy activation algorithm called k-Same, which scientifically limits the ability of facial recognition software to reliably recognize faces while preserving facial details in the image. The algorithm determines the similarity between faces based on the distance metric and creates a new face by averaging the image components, which can be either the original image pixels (k-Same-Pixel) or eigenvectors (k-Same-Eigen).

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

Automated human face image de-identification is a very necessary technology for privacy protection protecting social media and intelligent surveillance applications. In this work, we propose to achieve facial anonymity so that it is difficult to reveal the identity by slightly modifying the existing face image to ‘average face' in addition to the general face blurring technique. This approach preserves the aesthetics of your face image while meeting your privacy goals. Specifically, we look at deep learning based FIP (Facial Identity Preservation) capabilities. Unlike traditional face descriptors, the FIP function can significantly reduce the difference between IDs while maintaining the difference between IDs. By suppressing and modifying the FIP function, we achieve the k-anonymous face image de identification goal while maintaining the desired utility. Using a face database, we have successfully demonstrated that the resulting "average face" preserves the aesthetics of the original image while ignoring facial image identity recognition.

"All the better to see you with, my dear": Facial recognition and privacy in online social networks[9]

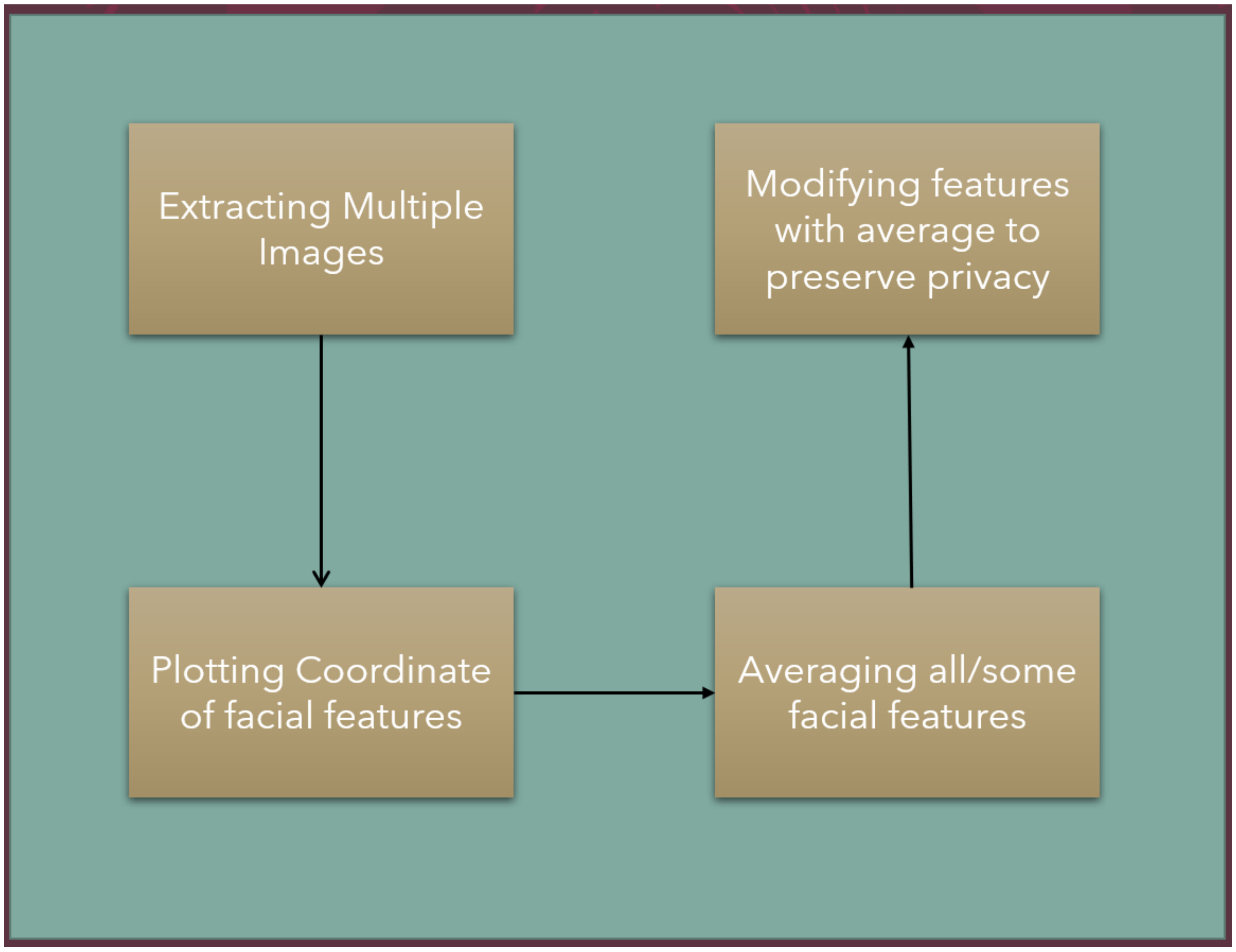
An overview of social and legal issues related to facial recognition technology used in online social networks explores how to manage the relevant privacy-related impacts, especially from a European data protection perspective.[9]

Robust human face hiding ensuring data privacy[10]

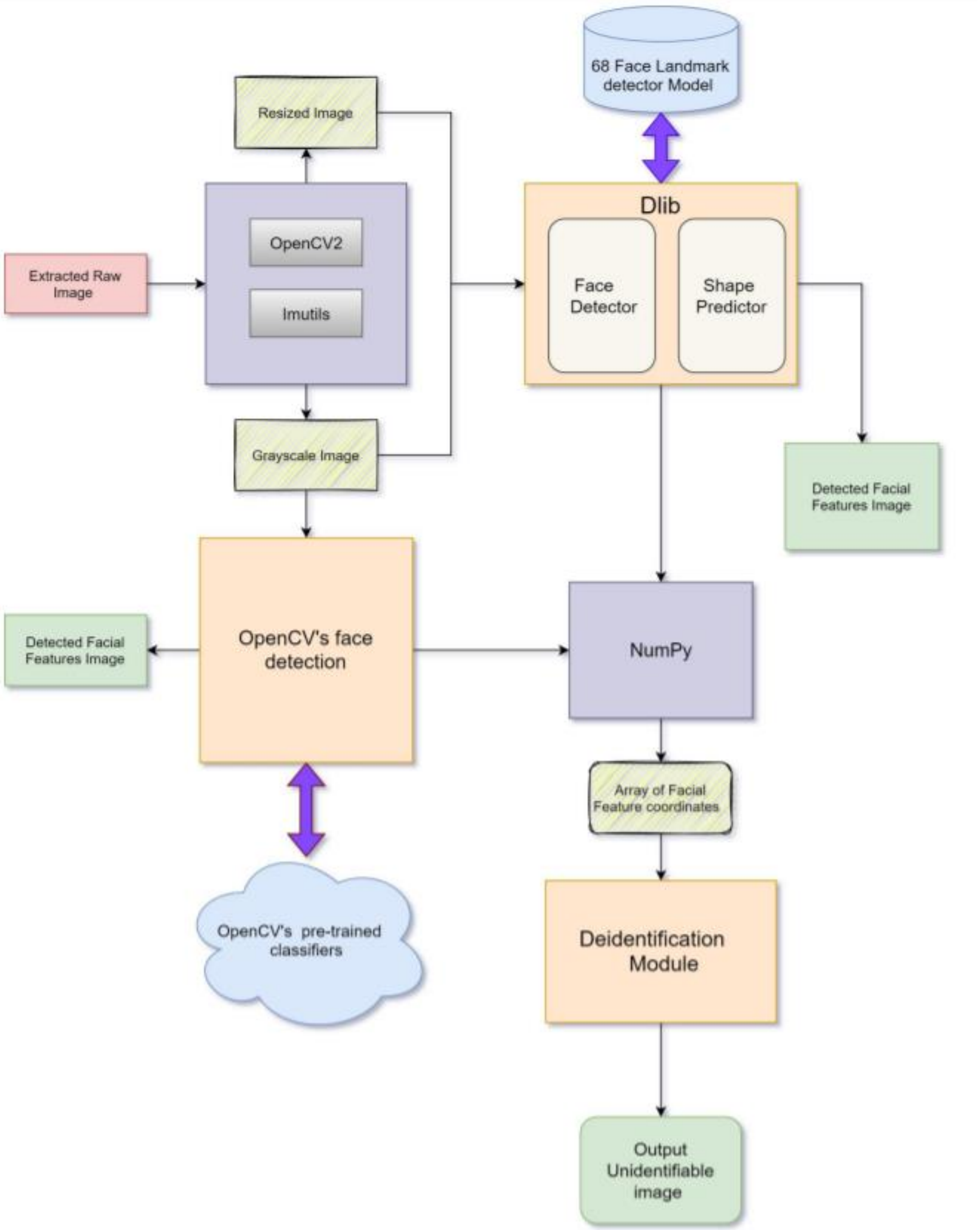
Today, people's video surveillance must ensure privacy. In this paper, we propose a complete solution to that problem by masking faces in video sequences that keep people anonymous. The system consists of two modules. First, the analysis module identifies and tracks the region of interest (ROI) where the face is detected. Second, the JPEG 2000 encoding module can limit the correct rendering of a human face by compressing frames that keep the ROI on a separate data layer. The analytics module combines two complementary methods of detecting faces, finding faces in an image, and tracking seamlessly over time. Combining these two methods improves robustness. If a face is already detected in a frame, tracking can find the face in successive frames even if the face detection algorithm does not recognize it. Also, detecting the face prevents the tracking from losing the target. The encoding module downshifts the JPEG 2000 data corresponding to the identified ROI to the lowest quality layer in the code stream. When the transmission bandwidth is limited, the human face is decoded to a lower quality and becomes invisible when needed.

System Architecture

3.1 HIGH LEVEL DESIGN


Extracting Multiple Images
Plotting Coordinate of facial features
Averaging all/some facial features
Modifying features with average to preserve privacy

3.2 LOW LEVEL DESIGN


68 Face Landmark detector Model
Resized Image
Dlib
OpenCV2
Extracted Raw Image
Face Detector
Shape Predictor
Imutils
Grayscale Image
Detected Facial Features Image
Detected Facial Features Image
OpenCV's face detection
NumPy
Array of Facial Feature coordinates
OpenCV's pre-trained classifiers
Deidentification Module
Output Unidentifiable image

4.MODULE EXPLANATION

We have three modules in this project:
1.Image processing module
2.face detection and facial feature detection module
3.Deidentification module

Image processing module:

This module helps us to achieve our goal of image pre-processing which in turn helps in deidentification of image. In here, first we take image as input using ‘imread’ function of cv2 library. Imread function is mainly used for reading and loading an image from any specified file. Image is then resized to square fit using resize function and image is interpolated to correct any distortion. After which the image is converted to greyscale using cvtColor function of Python’s OpenCV2 and imutils library. We convert to grayscale image because ‘get_frontal_face_detector()’ of dlib library works only with grayscale image only.

Face detection and facial feature detection module:

We have achieved facial feature detection using dlib Python’s library. Dlib is a advanced ML library created using c++ and it is compatible with c/c++, java and python. This library is used to solve complex realworld problems. Our face have several which can be identified such as eyes, noes, mouth etc. Using DLib algorithm to detect these features more precisely, we get a map of points that border each facial feature. Here, the map is composed of 68 points as mention in methodology. Then we used dlib’s function ‘get_frontal_face_detector()’, this function returns detector which we can use to retrieve imformation about face. Then we use ‘shape_predictor()’ function to find facial feature landmark. This function uses a pretrained model, which we have used in our project ‘shape_predictor_68_face_landmarks.dat’. We have also tried another face detection technique, HAAR CASCADE CLASSFIERS. It also uses a pretrained classifier for face which is already there is OpenCV library. To used this model we need to first load xml file stored in opencv/data/haarcascades/ folder then we used ‘v2.CascadeClassifier.detectMultiScale()’ function to detect face. we have also tried eyes detection using this technique, but result was not reliable. So we stick with shape_predictor_68_face_landmarks model to detect facial feature which is much better than cascade classifier. We get the coordinate of each feature of face which was detected using DLib shape predictor. We used those coordinates in deidentification module.

Deidentification module:

Facial feature’s detected coordinates are used here to achieve deidentification. We used coordinate of two different image and used Pythagorean Theorem and divided the result by 2 to normalise it. So we achieved to make face such that it cannot be detected using all most any facial detection model available today.

Formula used in deidentification:

Suppose $(x1,1)$ is coordinate for grayscale_image1 and $(x2,y2)$is coordinate for grayscale_image2 then out final output coordinate is:

$$\frac{\sqrt[2]{x_1^2 + x_2^2}}{2}, \frac{\sqrt[2]{y_1^2 + y_2^2}}{2}$$

4.1 Algorithm

```
1. define a dictionary that maps the indexes of the facial landmarks to specific face regions
2. Construct 2D array from facial shape i.e. for each 68 facial landmarks - convert them to a 2-tuple of (x, y)-coordinates
3. Visualize facial landmark:
    3.1. create two copies of the input image -- one for the overlay and one for the final output image.
    3.2. if the colors list is None, initialize it with a unique color for each facial landmark region
    3.3. loop over the facial landmark regions individually:
        3.3.1. grab the (x, y)-coordinates associated with the face landmark
        3.3.2. check if are supposed to draw the jawline: since the jawline is a non-enclosed facial region, just draw lines between the (x, y)-coordinates.
        3.3.3. otherwise, compute the convex hull of the facial landmark coordinates points and display it
    3.4. apply the transparent overlay and return the output.
4. initialize dlib's face detector (HOG-based) and then create the facial landmark predictor
5. load the input image, resize it, and convert it to grayscale
6. detect faces in the grayscale image.
7. determine the facial landmarks for the face region, then convert the landmark (x, y)- coordinates to a NumPy array
8. Loop over the facial landmark coordinates for two gray scaled images:
    8.1. square the coordinates of two images
    8.2. Add up them
    8.3. Square root the sum value
    8.4. Divide it by two to normalize coordinates.
9. Show the averaged grayscale image.
```

5. IMPLEMENTATION

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:

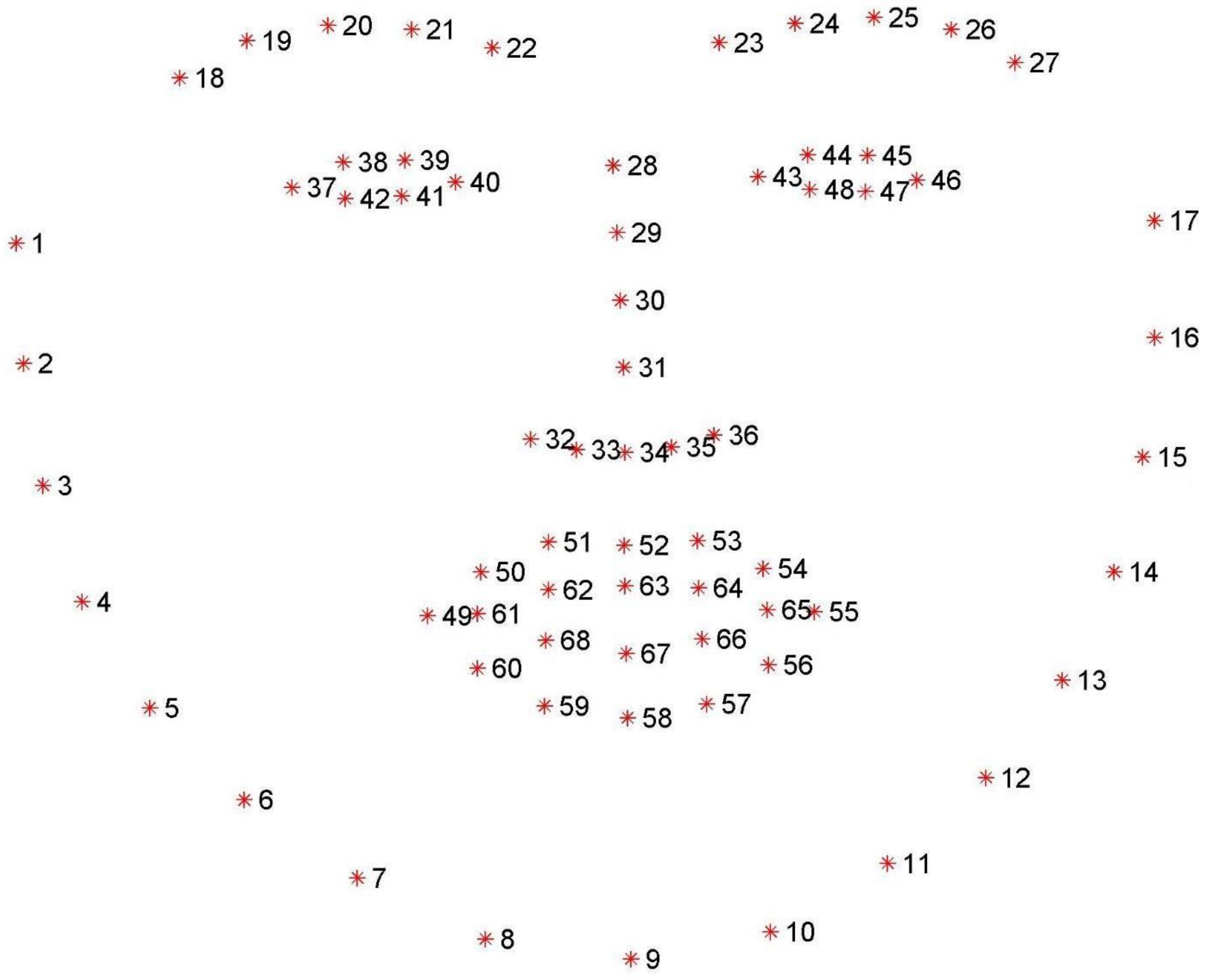


Figure 1 68 co-ordinate map for facial mapping.

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].

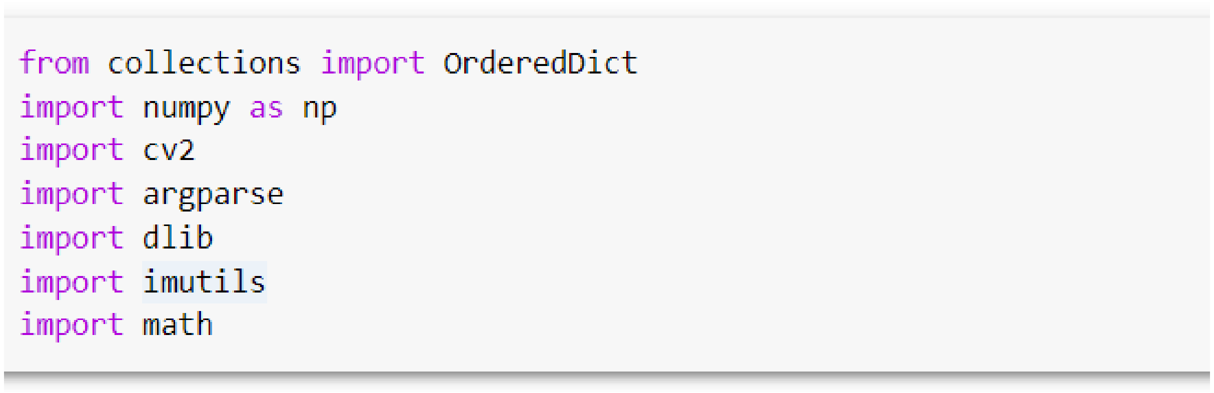
FACIAL_LANDMARKS_IDXS dictionary is encoded with these mappings inside the face_utils of the imultils library

We have used the following packages in python:
* NumPy
* OpenCV
* dlib
* imutils

We will be using images for illustration purposes. We have named them as image_1, image_2,image_3,image_4. They are present in the application root directory where we have a folder name ‘images’

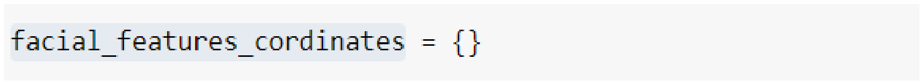
5.1 RESULTS AND DISCUSSION

We have multiple python Library : OrderedDict to store key-value pair for facial_Landmark_Indexes, numpy to handle arrays faster, cv2 for image processing and face detection, dlib toolkit for facial landmark detection, imutil is also for image resizing and other image processing, and maths for mathematical calculation.

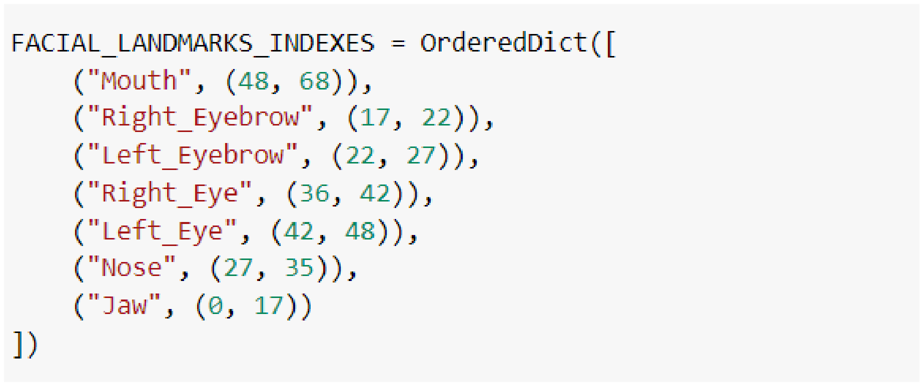


```python
from collections import OrderedDict
import numpy as np
import cv2
import argparse
import dlib
import imutils
import math
```

Created a facial landmark dictionary with facial feature and its value.

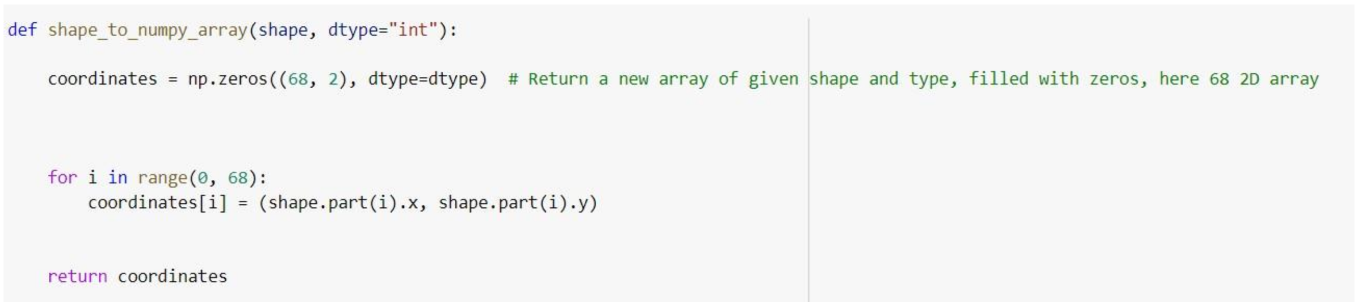


```python
facial_features_cordinates = {}
```



```python
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))
])
```

This function is created for generating coordinate array from detected facial shape.



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

    coordinates = np.zeros((68, 2), dtype=dtype)  # Return a new array of given shape and type, filled with zeros, here 68 2D array


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


    return coordinates
```

Function "visualize facial landmark" is for visualizing landmarks and coordinates for each features of detected shape of face. In this function, first the detected shape (using detector and predictor – you will find it below) is assigned a key value (facial feature i.e. mouth, lefteye, righteye etc.) and the jaw line is drawn by joining coordinate of detected jaw line coordinate and since other features have a closed shape so, using convex hull it is shaded, then *cv2.addweighted* is applied to show shaded portion on facial_feature on original image.

```
def visualize_facial_landmarks(image, shape, colors=None, alpha=0.75):

    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(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
```

Dectector is taken from dlib library to detect face for raw image , and predictor will use shape_predictor_68_landmark, a trained model to detect shape of facial image.

```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("/content/shape_predictor_68_face_landmarks.dat")
```

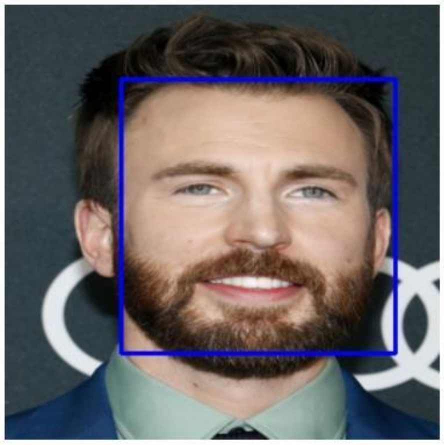
First, we read the image using "imread" and then resized it to square fit and interpolated it then converted the image to grayscale using "cvtColor".

```
image = cv2.imread("/content/images/image_1.jpg")
image = cv2.resize(image,(300,300),interpolation=cv2.INTER_AREA)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_frontalface_default.xml")
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray,1.1,4)
```

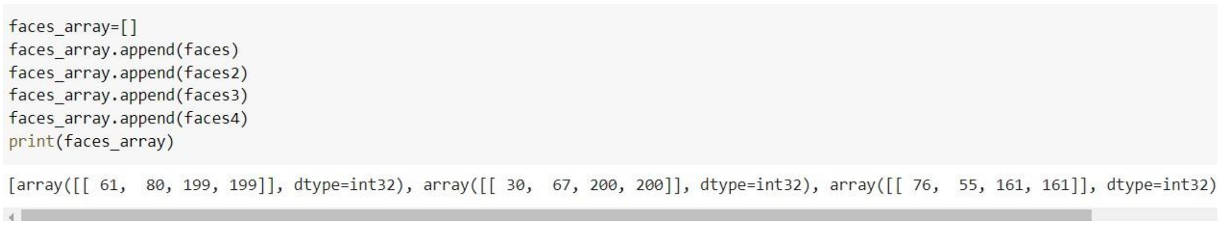
```
image2 = cv2.imread("/content/images/image_2.jpg")
image2 = cv2.resize(image2,(300,300),interpolation=cv2.INTER_AREA)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_frontalface_default.xml")
gray2 = cv2.cvtColor(image2,cv2.COLOR_BGR2GRAY)
faces2 = face_cascade.detectMultiScale(gray2,1.1,4)

image3 = cv2.imread("/content/images/image_3.jpg")
image3 = cv2.resize(image3,(300,300),interpolation=cv2.INTER_AREA)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_frontalface_default.xml")
gray3 = cv2.cvtColor(image3,cv2.COLOR_BGR2GRAY)
faces3 = face_cascade.detectMultiScale(gray3,1.1,4)

image4 = cv2.imread("/content/images/image_4.jpg")
image4 = cv2.resize(image4,(300,300),interpolation=cv2.INTER_AREA)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_frontalface_default.xml")
gray4 = cv2.cvtColor(image4,cv2.COLOR_BGR2GRAY)
faces4 = face_cascade.detectMultiScale(gray4,1.1,4)
```

Here we also used OpenCV face detection to detect face, it uses OpenCV pretrained classifier and the used a rectangle to visualized detected face.

```
image5 = cv2.imread("/content/images/image_5.jpg")
image5 = cv2.resize(image5,(300,300),interpolation=cv2.INTER_AREA)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_frontalface_default.xml")
gray5 = cv2.cvtColor(image5,cv2.COLOR_BGR2GRAY)
faces5 = face_cascade.detectMultiScale(gray5,1.1, 4)

for (x,y,w,h) in faces5:
    cv2.rectangle(image5,(x,y),(x+w,y+h),(255,0,0),2)
cv2_imshow(image5)
```



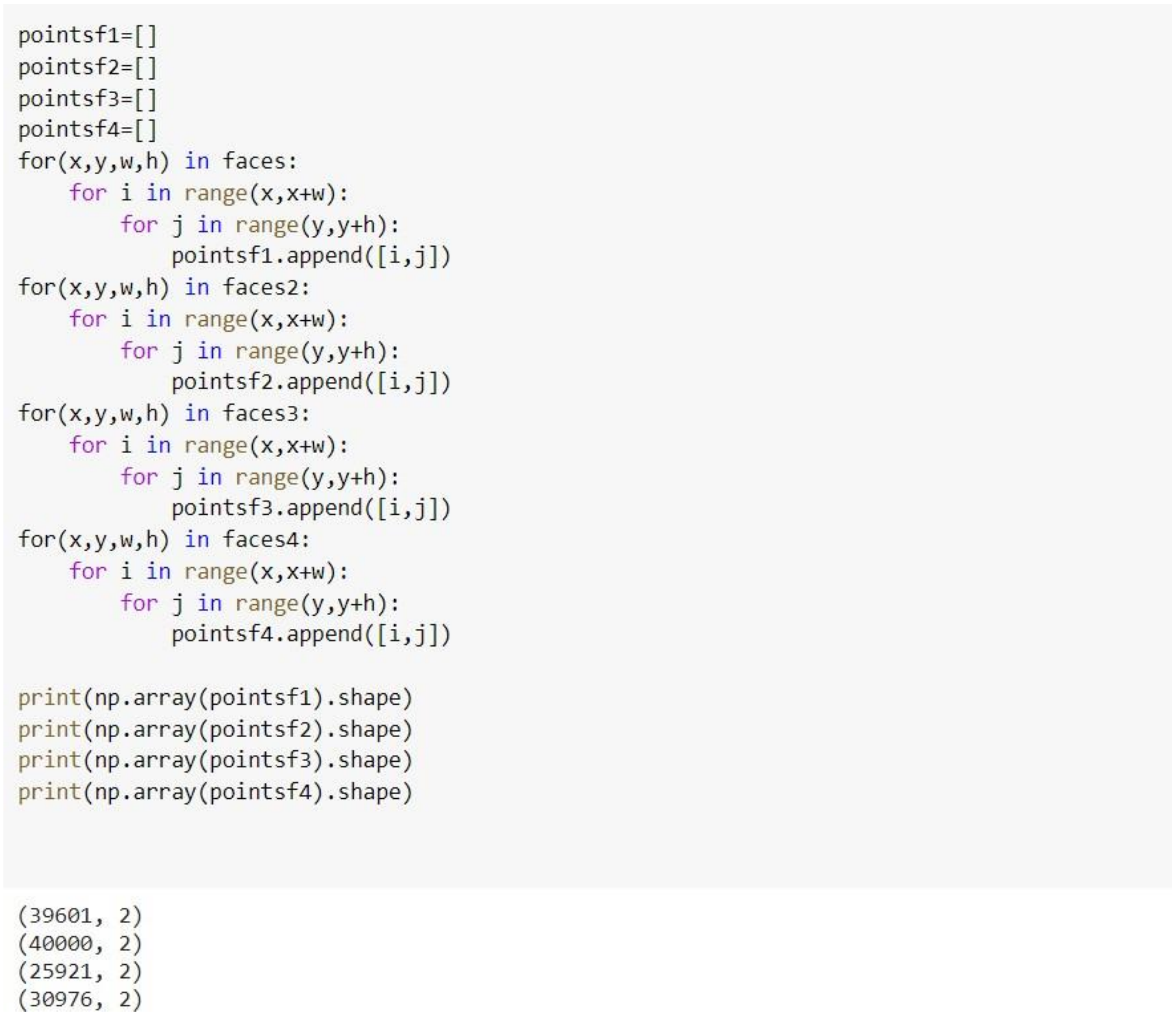
Here you can see coordinate of each face detected from corresponding images :



```
faces_array=[]
faces_array.append(faces)
faces_array.append(faces2)
faces_array.append(faces3)
faces_array.append(faces4)
print(faces_array)
```

```
[array([[ 61,  80, 199, 199]], dtype=int32), array([[ 30,  67, 200, 200]], dtype=int32), array([[ 76,  55, 161, 161]], dtype=int32)
```

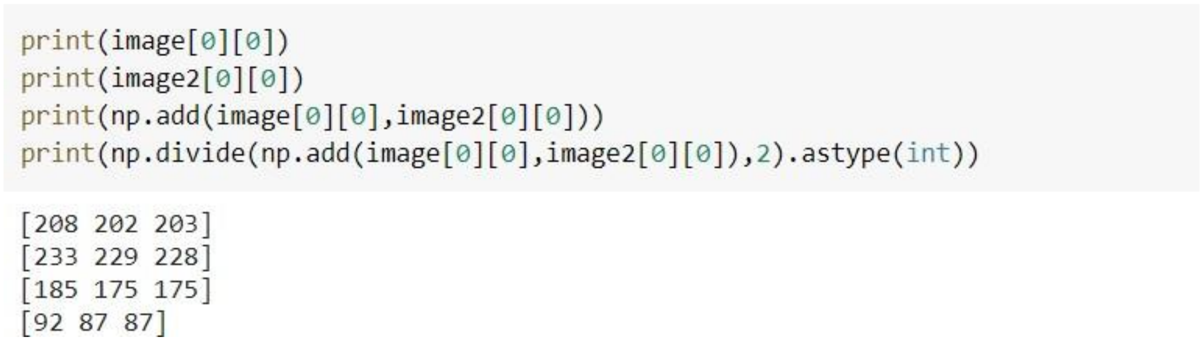
Here we made coordinates from detected face above and stored in 2-D array. And number of coordinate detected can be seen as output for each images.



```
pointsf1=[]
pointsf2=[]
pointsf3=[]
pointsf4=[]
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])
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])
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])
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])

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

```
(39601, 2)
(40000, 2)
(25921, 2)
(30976, 2)
```

A test result, manipulating coordinated for images with mathematical calculation.



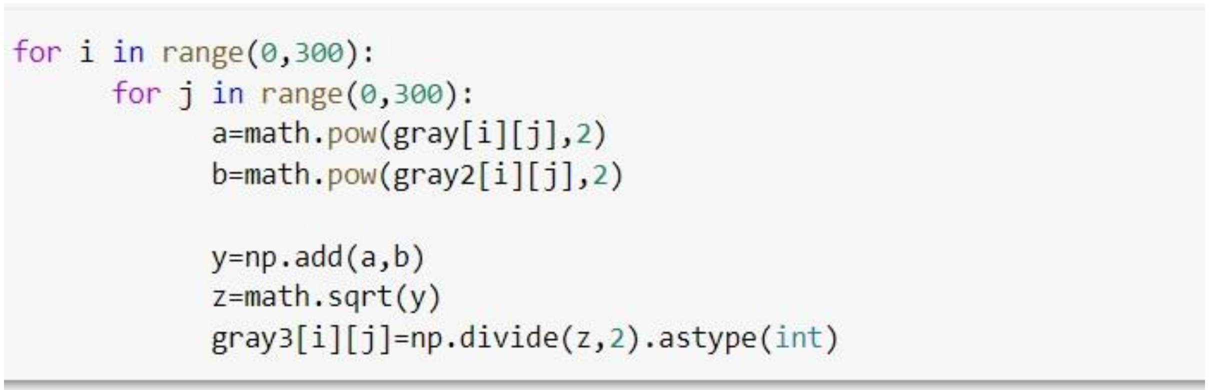
```
print(image[0][0])
print(image2[0][0])
print(np.add(image[0][0],image2[0][0]))
print(np.divide(np.add(image[0][0],image2[0][0]),2).astype(int))
```

```
[208 202 203]
[233 229 228]
[185 175 175]
[92 87 87]
```

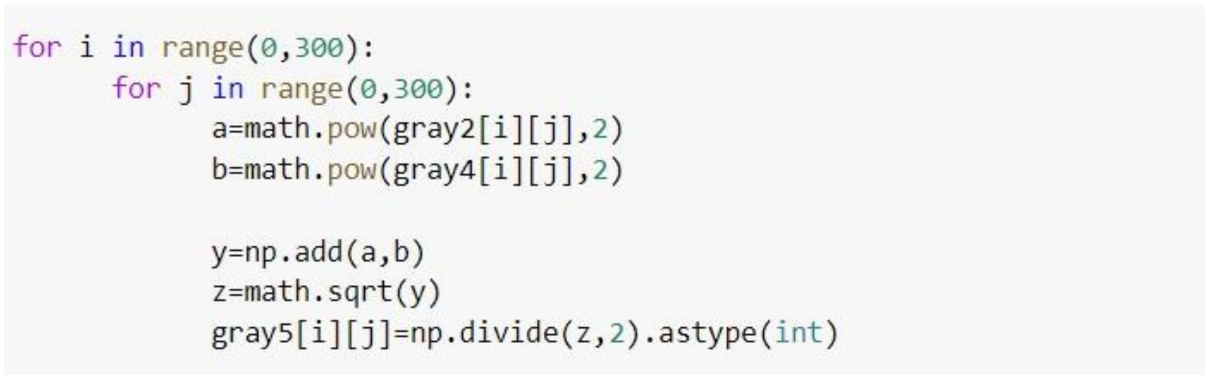
Deidentification Module:

Here we took coordinates for each grayscale image and applied Pythagorean Theorem and divided the result by 2 to normalise it. Suppose $(x1,y1)$ is coordinate for grayscale_image1 and $(x2,y2)$is coordinate for grayscale_image2 then out final output coordinate is:

$$\frac{\sqrt[2]{x_1^2+x_2^2}}{2},\frac{\sqrt[2]{y_1^2+y_2^2}}{2}$$



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



```
for i in range(0,300):
      for j in range(0,300):
            a=math.pow(gray2[i][j],2)
            b=math.pow(gray4[i][j],2)

            y=np.add(a,b)
            z=math.sqrt(y)
            gray5[i][j]=np.divide(z,2).astype(int)
```

Image of grayscale image 1

```
from google.colab.patches import cv2_imshow
cv2_imshow(gray);
```

Image of grayscale image 2

```
cv2_imshow(gray2);
```

Image after applying deidentification algorithm on image1 and image2 :

```
cv2_imshow(gray3);
```

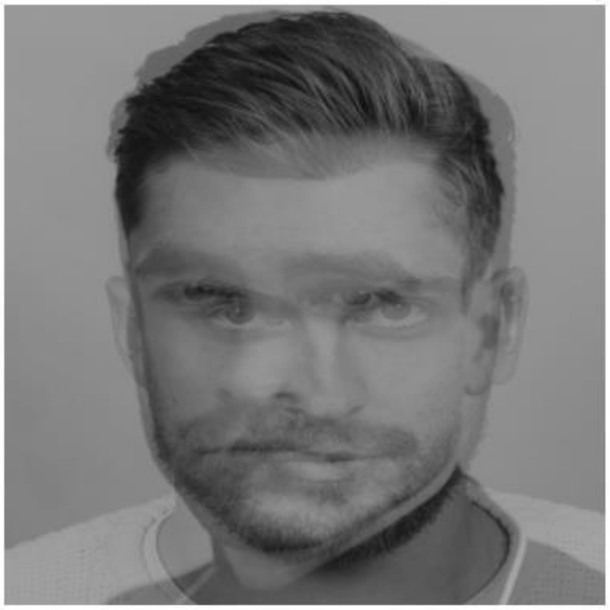
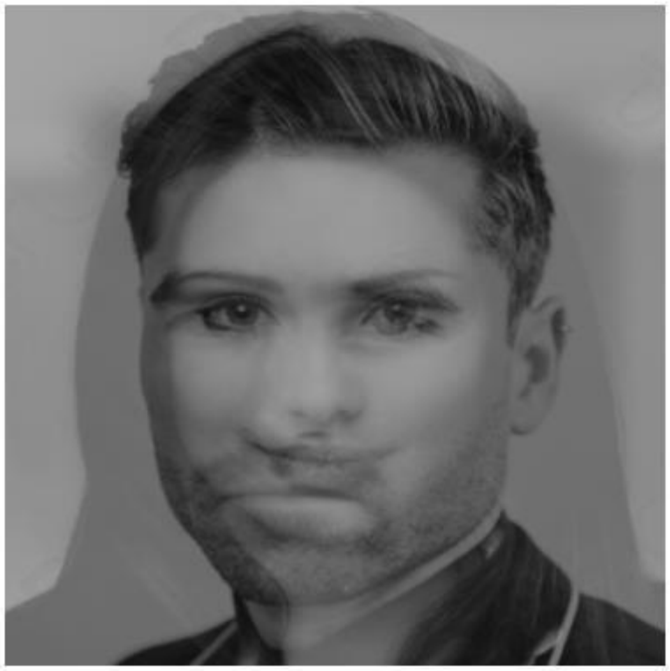


Image of grayscale image 4

```
cv2_imshow(gray4);
```

Image after applying deidentification algorithm on image2 and image4 :

```
cv2_imshow(gray5);
```

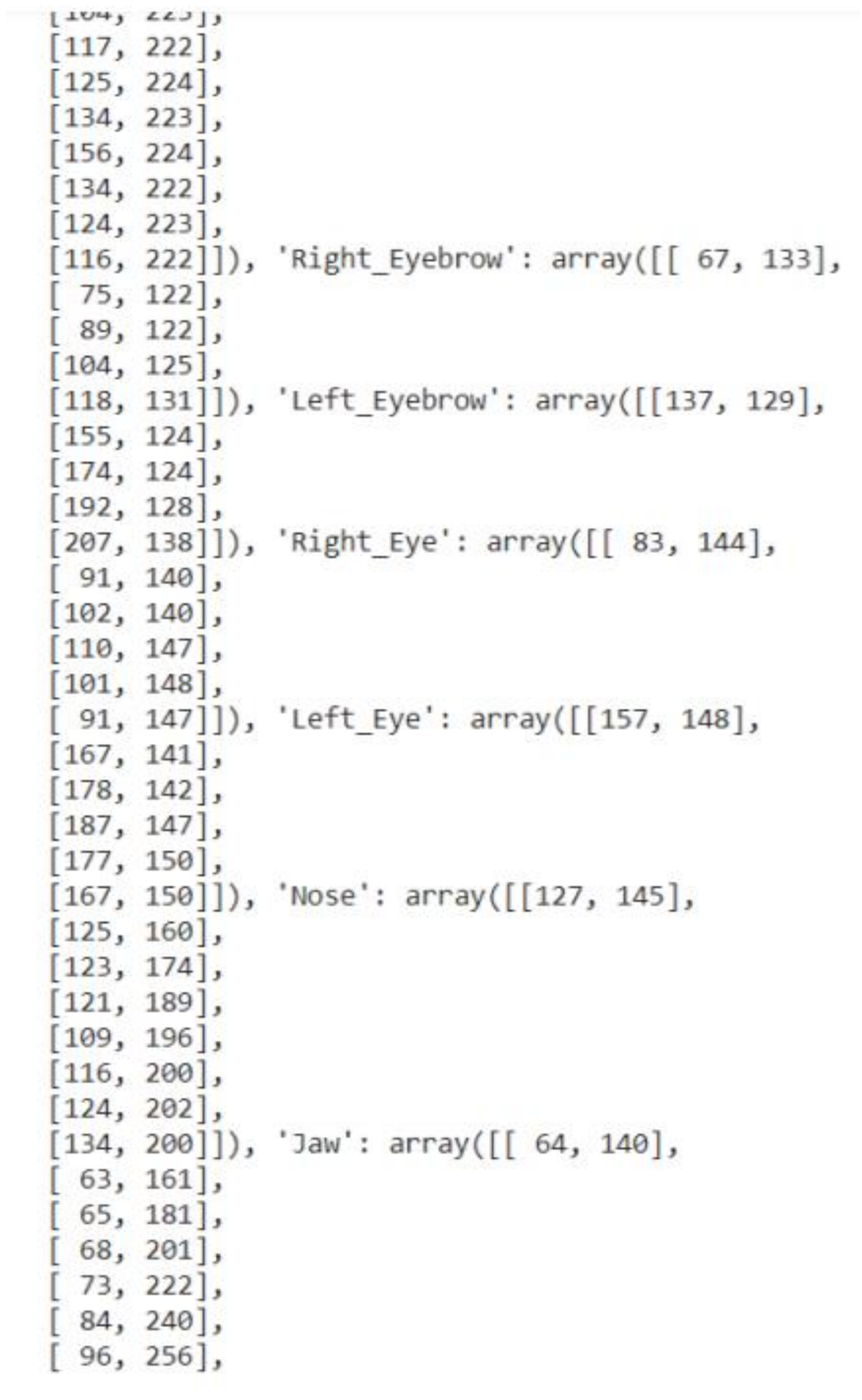


This is driver code for identification of facial feature and shape (68 facial landmark), first feed the image in detector and then the output image is feed into shape predictor and the out space feature is converted into facial features coordinate array, and then visualization function is called, as a result you can see the image each 68 landmark coordinates and and plotted predicted shape (shaded in black colour).

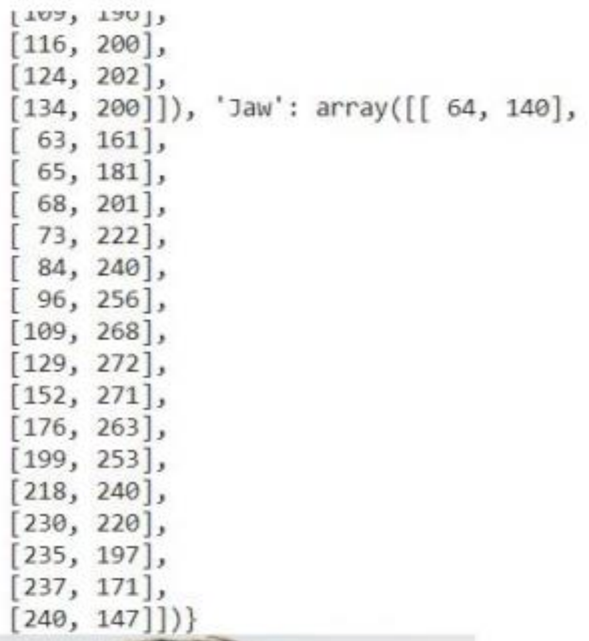
```
rects = detector(gray2, 1)
```

```
for (i, rect) in enumerate(rects):
    shape = predictor(gray2, rect)
    shape = shape_to_numpy_array(shape)

    output = visualize_facial_landmarks(image2, shape)
    cv2_imshow(output)
    cv2.waitKey(0)
```

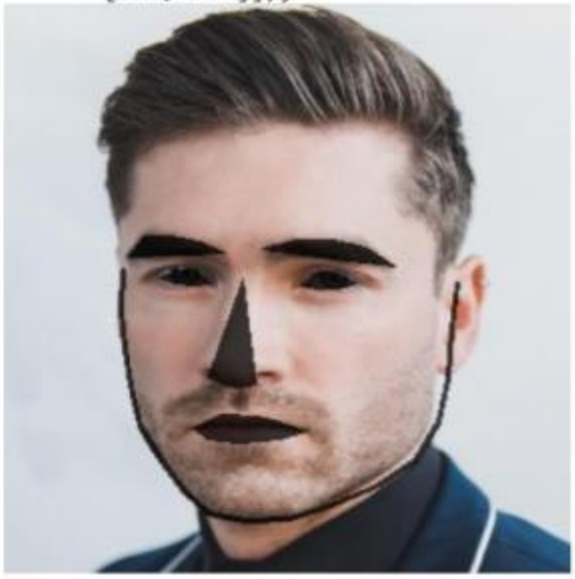
```
0 Mouth

1 Right_Eyebrow

2 Left_Eyebrow

3 Right_Eye

4 Left_Eye

5 Nose

6 Jaw

{'Mouth': array([[101, 223],
       [109, 219],
       [116, 216],
       [124, 219],
       [134, 217],
       [146, 220],
       [160, 225],
       [147, 229],
       [134, 230],
       [125, 231],
       [116, 230],
       [108, 229],
```



```
[104, 223],
[117, 222],
[125, 224],
[134, 223],
[156, 224],
[134, 222],
[124, 223],
[116, 222]]), 'Right_Eyebrow': array([[ 67, 133],
[ 75, 122],
[ 89, 122],
[104, 125],
[118, 131]]), 'Left_Eyebrow': array([[137, 129],
[155, 124],
[174, 124],
[192, 128],
[207, 138]]), 'Right_Eye': array([[ 83, 144],
[ 91, 140],
[102, 140],
[110, 147],
[101, 148],
[ 91, 147]]), 'Left_Eye': array([[157, 148],
[167, 141],
[178, 142],
[187, 147],
[177, 150],
[167, 150]]), 'Nose': array([[127, 145],
[125, 160],
[123, 174],
[121, 189],
[109, 196],
[116, 200],
[124, 202],
[134, 200]]), 'Jaw': array([[ 64, 140],
[ 63, 161],
[ 65, 181],
[ 68, 201],
[ 73, 222],
[ 84, 240],
[ 96, 256],
```



```
[109, 198],
[116, 200],
[124, 202],
[134, 200]]), 'Jaw': array([[ 64, 140],
[ 63, 161],
[ 65, 181],
[ 68, 201],
[ 73, 222],
[ 84, 240],
[ 96, 256],
[109, 268],
[129, 272],
[152, 271],
[176, 263],
[199, 253],
[218, 240],
[230, 220],
[235, 197],
[237, 171],
[240, 147]])}
```



Input image used:

Image_1.jpg

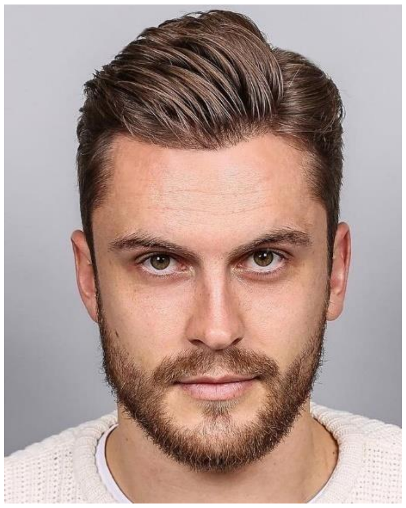


Image_2.jpg

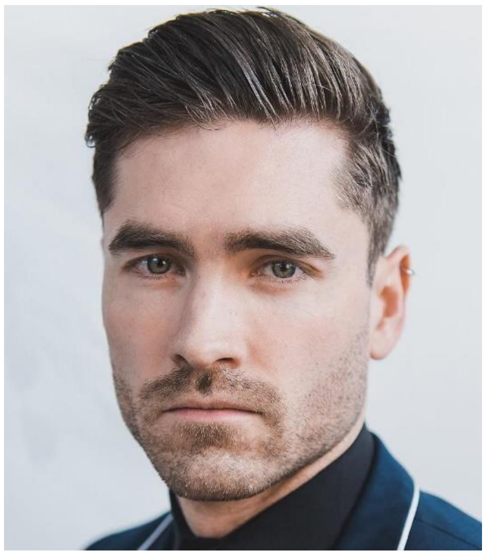


Image_3.jpg

Image_4.jpg

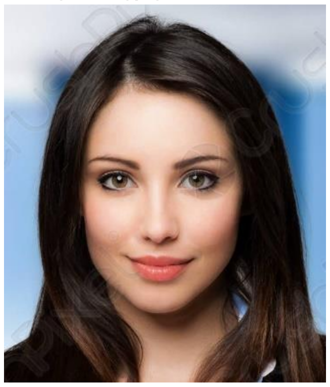


Image_5.jpg

6.CONCLUSION

With the increasing number of information generated and shared over the Internet, there is an urgent need to develop better methods for protecting the identity of facial images.
Old techniques such as attaching black bands over images and hiding facial features have made images useless while protecting sensitive data. Thus, techniques like averaging pixel features or compressing pixels preserve the usefulness and identity of the image.
This report concludes in this section as follows:
(1) What you need to do this to anonymize the image
(2) The practice necessary to unfold this kind of work
(3) How this work relates to the policy on data sharing.
Consider the actual work of anonymizing surveillance images. The k-Same algorithm provides the key elements to achieve this, but building an operating system requires more work. Consider an anonymization tool that has a set of individual faces as input and a set of faces anonymized as output. There are pre-processing steps required to convert the facial expressions in an image into a set of person-specific faces. There are also post-processing steps necessary to place anonymized faces in the image so that people's behavior and behavior are preserved in the original image, but the unidentified face replaces the original facial expression. For example, the person coughing in the original image should be an unidentified person coughing in the final image. If necessary, postprocessing also applies an encrypted identifier to the unidentified face, revealing the person's original face once given the appropriate key. Both pre- and post-processing stages cover active areas of research in computer vision, computer graphics and facial recognition.
I can't exactly determine which person with what name appears in which frame of an unidentified image, but it can identify all k people and treat them as groups. In other kinds of data, k is stipulated by policy, but you still need to pay attention. The anonymization described in this work does not guarantee anonymity. This action interferes with facial recognition to limit automatic continuous recognition of populations whose images were captured on video but did not behave suspiciously. These technologies, for example, interfere with facial recognition software, but correct identification is still possible by recognizing clothes, actions, or co-occurrences with others.
In conclusion, we review the broader policy settings. Policies generally treat personal information in individual data in a binary manner. Data collectors hold data exclusively (or almost exclusively). Or, the data is shared almost without restrictions. Various sharing policies are possible through data de identification, where the individual's identity cannot be identified in the data. Anonymization allows you to share your data with a scientific guarantee of anonymity, and the resulting data is practically useful. In the latter case, this is probably because anonymization reduces risk and improves usability. In terms of video surveillance data, policies can be polarized when society chooses whether to share the collected video data or to ban all sharing. The use of anonymization technology allows you to share your data more freely while providing the privacy protection that applicable laws pursue.

7.REFERENCES

[1]Carrillo, P., Kalva, H. and Magliveras, S., 2010. Compression independent reversible encryption for privacy in video surveillance. EURASIP Journal on Information Security, 2009, pp.1-13.

[2]Nakashima, Y., Koyama, T., Yokoya, N. and Babaguchi, N., 2015, June. Facial expression preserving privacy protection using image melding. In 2015 IEEE International Conference on Multimedia and Expo (ICME) (pp. 1-6). IEEE.

[3]Buckley, B. and Hunter, M., 2011. Say cheese! Privacy and facial recognition. Computer Law & Security Review, 27(6), pp.637-640.

[4]Rahulamathavan, Y. and Rajarajan, M., 2015. Efficient privacy-preserving facial expression classification. IEEE Transactions on Dependable and Secure Computing, 14(3), pp.326-338.

[5]Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I. and Toft, T., 2009, August. Privacypreserving face recognition. In International symposium on privacy enhancing technologies symposium (pp. 235-253). Springer, Berlin, Heidelberg.

[6]Monteleone, S., 2012. Privacy and Data Protection at the time of Facial Recognition: towards a new right to Digital Identity?. European Journal of Law and Technology, 3(3).

[7]Scheirer, W., White, R. and Boult, T., Privacy Enhancement via Adaptive Cryptographic Embedding.

[8]Chi, H. and Hu, Y.H., 2015, December. Face de-identification using facial identity preserving features. In 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP) (pp. 586-590). IEEE.

[9]de Andrade, N.N.G., Martin, A. and Monteleone, S., 2013. " All the better to see you with, my dear": Facial recognition and privacy in online social networks. IEEE security & privacy, 11(3), pp.21-28.

[10]Martínez-Ponte, I., Desurmont, X., Meessen, J. and Delaigle, J.F., 2005, April. Robust human face hiding ensuring privacy. In Proceedings of the International Workshop on Image Analysis for Multimedia Interactive Services (Vol. 4).

.