

FACE RECOGNITION AND ATTENDANCE LIST USING PYTHON

A THESIS SUBMITTED TO
THE FACULTY OF ARCHITECTURE AND ENGINEERING
OF
EPOKA UNIVERSITY

BY

KEVIN MAMILLO

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JULY, 2022

Approval sheet of the Thesis

This is to certify that we have read this thesis entitled **“Face recognition and attendance list using Python”** and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Dr. Arban Uka
Head of Department
Date: July 13, 2022

Examining Committee Members:

Assoc. Prof. Dr. Carlo Ciulla (Computer Engineering) _____

Dr. Arban Uka (Computer Engineering) _____

Dr. Igli Hakrama (Computer Engineering) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Surname: Kevin Mamillo

Signature: _____

ABSTRACT

FACE RECOGNITION AND ATTENDANCE LIST USING PYTHON

Mamillo, Kevin

M.Sc., Department of Computer Engineering

Supervisor: Dr. Arban Uka

Face recognition is a must-have technology these days. In addition, they have various uses, such as Cell phones, etc. In this project, I recognised the face as the student attendance system, which can replace the regular paper attendance system and fingerprint attendance system. I used Python to create this project, as it is constructive in the face and other usage logs.

This project is based on the main program, which detects and detects faces using the HOG algorithm with values and parameters. In addition, the subsystem is in a CSV file sheet (later, you can create an Excel spreadsheet, database integration and analysis with the deep neural network). This is integrated into the program and filled with names and face recognition time.

The system has two main sections, which are 1) Face detection with face recognition algorithm and keep updating as data is fed in. 2) When the images are uploaded to the system (i.e., by students, administrators, or anyone can upload), it is then saved as a CSV file, which we can use for deeper analysis and statistical data..

Keywords: *Face detection, face recognition system, attendance system, machine learning, object detection, HOG algorithm,*

ABSTRAKT

SISTEMI I FREKUENTIMIT NEPERMJET NJOHJES SE FYTYRES

Mamillo, Kevin

Master Shkencor, Departamenti i Inxhinierisë Kompjuterike

Udhëheqësi: Dr. Arban Uka

Në ditët e sotme teknologjia e njohjes së fytyrës po luan një rol të rëndësishëm në cdo fushë të jetës sonë. Atë e kemi të pranishme kudo, si prsh: në celularët tanë, në aeroporte, në zyrat e shërbimit biometrik etj. Sistemet e njohjes së fytyrës përdorin një sërë matjesh dhe teknologjish për të analizuar fytyrën, duke analizuar përmasat e karakteristikave të saj, distancën midis tipareve kryesore apo duke përfshirë edhe imazhin termik.

Ky punim është fokusuar kryesisht në lehtësirën që sistemi i njohjes së fytyrës mund të ofroj në fushën e mësimdhënies. Në ditët e sotme pjesëmarrja në universitete është shumë e rëndësishme, po aq e rëndësishme është edhe procesi i marrjes së mungesave. Aktualisht në Shqipëri po operohet mënyra konvencionale e marrjes së mungesave, pra nëpërmjet thirrjes së emrit të studentit apo numrit në listën e prezencës. Kjo mënyrë jo vetëm që konsumon kohë, por kërkon dhe energji. Për këtë arsye një mënyrë optimale që kam sugjeruar unë është nëpërmjet sistemit të njohjes së fytyrës ku me anë të një pajisje identifikuese në hyrje të sallave identifikohet studentin nëpërmjet njohjes së fytyrës. Zbulimi i fytyrës përdoret për të lokalizuar pozicionin e rajonit të fytyrës, ndërsa njohja e saj përdoret për shënim frekuentimin e studentit.

Gjithashtu, ky punim mund të zbatohet edhe në seancat e provimit ose në veprimtari të tjera mësimore ku frekuentimi është shumë thelbësor. Ky sistem eliminon studentin klasik, pra thirrja e emrit të studentit, ose kontrolli i kartës së identitetit, të

cilat mund të ndërpresin procesin mësimor ose mund të jenë bezdisëse për studentët gjatë seancave mësimore. Kjo njohje automatike realizohet nëpërmjet regjistrimit të studentit në bazën e të dhënave. Baza e të dhënave e të gjithë studentëve në klasa ruhet kur fytyra individuale e nxënësit përputhet me një nga fytyrat e ruajtura në bazën e të dhënave , ku me pas regjistrohet frekuentimi.

Fjalët kyçe: Njohja e fytyres , Baza e te dhenave, frekuentimi , regjistrimi, student klasik, student, imazh termik, analiza e fytyres, etj,

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Dr. Arban Uka for the continuous support of my thesis study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this study.

TABLE OF CONTENTS

ABSTRACT	iii
ABSTRAKT	iv
ACKNOWLEDGEMENTS	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	ix
CHAPTER 1	1
INTRODUCTION	1
1.1 Project Definition	1
1.2 Project Objectives	1
1.3 Project Specifications.....	2
1.4 Organization of the thesis.....	3
CHAPTER 2	4
LITERATURE REVIEW.....	4
2.1 Project background	4
2.2 Previous work.....	4
CHAPTER 3	10
METHODOLOGY	10
3.1 Project background	10
3.2 Using Machine Learning to solve higly complex problem.....	11
3.2.1 Systematic face recognition.....	12

3.2.2	Systematic face recognition	12
3.2.3	Projecting and showing faces	18
3.2.4	Encoding face part.....	21
3.2.5	Econcoding our face image	24
3.3	System testing and analysis.....	26
3.4	Eigenvalues and Eigenvector analyze of faces.....	30
3.4.1	Singular value decomposition.....	33
3.5	Face Recognition using SVD	40
CHAPTER 4.....		44
CONCLUSIONS		44
4.1	Conclusions	44

LIST OF FIGURES

<i>Figure 1.</i> Project flow diagram (flowchart).....	3
<i>Figure 2.</i> Current project flow diagram.....	5
<i>Figure 3.</i> Current project flow diagram of attendance system in the experiement	7
<i>Figure 4.</i> Cgan model, where G is the main function and D are the presentyed as follow, G is supposed to be Generative Network.....	8
<i>Figure 5.</i> The current figure represents the researchers approach for combining face recognition and morphing detection in one single system.	9
<i>Figure 6.</i> This images shows two doifferent persons with almost identic faces	10
<i>Figure 7.</i> People taking selfie yusing a digital camera from mobile phone	12
<i>Figure 8.</i> Image used for some testing	13
<i>Figure 9.</i> Image used for some testing	14
<i>Figure 10.</i> The arrow shows the direction when it is getting darker	14
<i>Figure 11.</i> Image that represents the key points	15
<i>Figure 12.</i> Image replaced by pixel gradients to extract those key components	15
<i>Figure 13.</i> Converting the original image By HOG algorithm.....	16
<i>Figure 14.</i> Using the HOG face pattern methodology to detect the faces.....	17
<i>Figure 15.</i> HOG Representation of processed images.	18
<i>Figure 16.</i> Humans can recognise booth the two images, but the computers will differently see the booth images of will ferrell.	18

Figure 17. We will see that there in the upper image will be the results of 68 located face landmarks of testing images.....	19
Figure 18. Snapchat uses also the same 3D technique for realtime video processing filters.	20
Figure 19. We can centre the eyes and mouth roughly in the same position in the image.....	21
Figure 20. Face recognition on a TV show	22
Figure 21. A single triplet-training step. Repeating the same step many times for different images and different peoples, the neural network will learn and generate 128 measurements of each person.	23
Figure 22. 128 measurements from the image	25
Figure 23. First structure of project code	26
Figure 24. Code run part 1.....	27
Figure 25. Code run part 1.....	28
Figure 26. The test was run successfully and we got the expected results.	29
Figure 27. Five images of four different faces which include part of Troy.	30
Figure 28. Average images generated from 4 actors in the Troy movie.	32
Figure 29. First step of the image converted black and white.	34
Figure 30. Getting an image where each column of pixels is different weighting of the same value.	35
Figure 31. Reconstructed image using the first n vectors of the singular value decomposition, in this case $n = 2$	36
Figure 32. Reconstructed image using the first n vectors of the singular value decomposition, in this case $n = 2$	37

Figure 33. Reconstructed image using the first n vectors of the singular value decomposition, in this case the n will vary from 5 to 40.....38

Figure 34. Reconstructed image using the first n vectors of the singular value decomposition, in this case the n will vary from 45 to 50.....39

Figure 35. Reconstructed image using the first n vectors of the singular value decomposition, in this case the n will vary from 45 to 50.....40

CHAPTER 1

INTRODUCTION

1.1 Project Definition

We are integrating sophisticated face recognition technology into your existing payroll systems. This way, instead of having to go through all that troubleshooting, save us time and effort, and use a far more accurate and efficient system that replaces the need for fingerprint identifiers. Our software will recognise faces automatically and update a CSV file, which later will use a database system.

The algorithm can convert this CSV to Excel data, used in databases and analyse with deep learning to improve future face identification algorithms using machine learning and advanced plugins.

1.2 Project Objectives

We are making the best use of Python programming to recognise faces from images and videos and recognise the faces after.

We are designing and developing an attendance system that will be much quicker, more efficient and save time and money.

Make good use of Machine learning.

An automated process that will lead us to a digital environment.

Increasing accuracy of the algorithm and preventing suspicious, malicious actions.

Encourage the technology used in daily life.

1.3 Project Specifications

Uses a real-time face recognition system algorithm. This will require Python 3.3+ or Python 2.7.”

This algorithm can recognise a vast number of users simultaneously

Metric camera resolution is required

Uses a folder named “face_detection_images” that will store images. If there are no images, it will use a sash portal to grab the images from a server and update the MYSQL database and the temporary CSV file, which can be digitally distributed and maintained.

The project can be applicable on Docker servers and cloud instances too.

1.4 Organization of the thesis

This is a presentation flowchart of project flow.

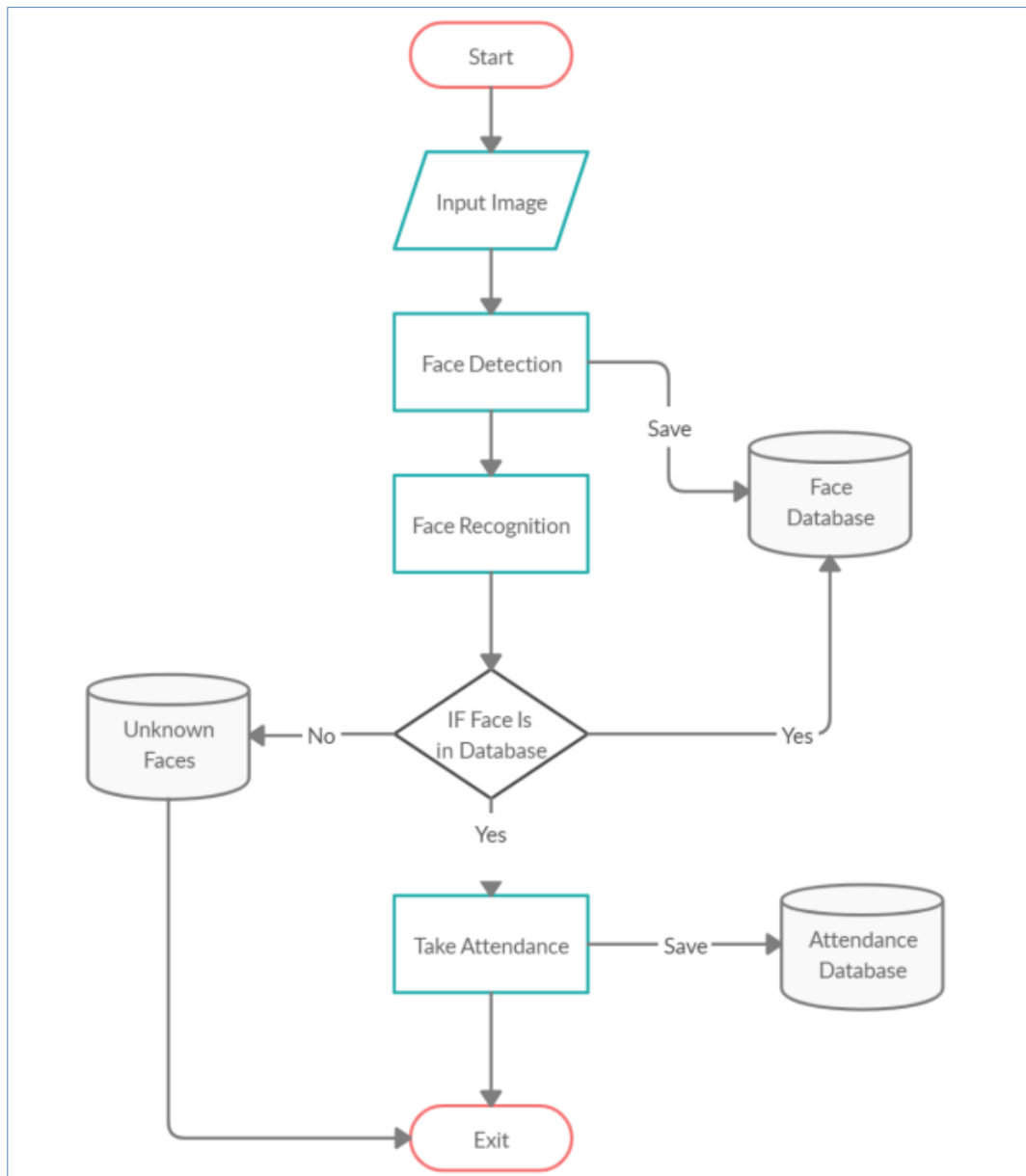


Figure 1. Project flow diagram (flowchart)

CHAPTER 2

LITERATURE REVIEW

2.1 Project background

In a face recognition system, the process flow begins with recognising facial features from a video camera, images, or content stored on a hard drive (cloud or local). Algorithm itself will proceed with the captured images and identifies the number of faces in the images. To analyse them using the learned patterns and compare them with known photographs in the database.

Simplifying and making face recognition technology more cost-effective was our motivation for this project. This also by connecting the concept of cloud storage, computing, and making face recognition more practical and usable in any environment.

2.2 Previous work

Project #1

This project is performed in the work of "Face Recognition Attendance System Based on Real-Time Video Processing" and represents a test of a face recognition algorithm that creates attendance records based on real-time video processing.

The complete assistance system consists of several modules, each of which fulfils different functions. It is conducive to maintaining and managing the whole system so it can be implemented quickly and easily, reducing program complexity and facilitating code reuse. This system's face recognition time and attendance design include video capture terminals, cable transmission modules, data storage hardware, face recognition hardware and computer terminals.

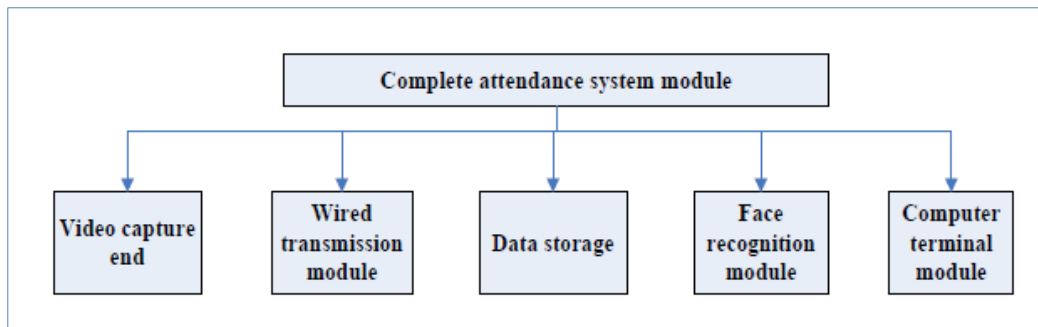


Figure 2. Current project flow diagram

Face recognition is an increasingly popular way to identify people in images. However, using a face recognition system to process video is not without its challenges. The authors of this paper thoroughly analyse the status of face recognition technology and examine its limitations. They also describe how these challenges can be overcome by developing new technologies to enhance face recognition accuracy or by developing new methods to increase the speed at which we can implement face recognition systems.

In this paper, the authors will analyse the application of face recognition assistance systems to real-time video processing. They will examine the accuracy and stability of these systems against changes in lighting conditions, background noise, and facial expressions. They will also critique the role played by face recognition systems in existing registration technologies and highlight any difficulties encountered by product developers.

To deeply observe the software of face-recognition systems, researchers analysed the accuracy of their reputation structures and the steadiness of historic structures. They also investigated whether they used these systems in critical check-in processes. To better understand the application of facial recognition attendance systems to real-time video processing, researchers have investigated the accuracy and stability of existing facial recognition systems and the role they play in important check-in processes. Rice field.

Face recognition systems are used for real-time video processing because they are accurate and stable. However, they are also used as a form of biometric identification to determine whether a person is allowed access to certain areas. The effectiveness of face recognition in this area is limited by the fact that it can be fooled easily by masks or make-up.

Researchers found that recognising a face was less accurate than recognising a name; that recognition was unstable, mainly when people were in groups; and that face recognition could be a valuable mechanism for the check-in process.

A study was conducted to analyse the feasibility of using real-time face recognition technology in virtual check-in applications. They chose two universities from a province for this experiment: the same number of students for each group were collected; they were counted and analysed; Researchers analysed the application space and development prospects of this face recognition system. They identified problems with implementing this technology into virtual check-in.

Project #2

The PCA and LDA methods easily learn and train data. However, training massive data sources becomes increasingly difficult. Face recognition algorithms can be tricked into learning fake or false examples of identities, slightly modifying them meanwhile the face recognition algorithm fails to recognise them.

The system's results were incorrect. These changes are almost not detectable so peoples will not notice them, but the classifier will do wrong and make mistakes. The attacker can attack the System of Machine Learning and disturb the result without knowing the basic face recognition model. As is shown in Fig. 13, taking the classic bi-classification problem as an example, this model learns a segmentation plane by training on samples in face recognition.

At present, generative adversarial networks (GANs) are one of the effective ways to resist attacks; Ian Good fellow proposed a generative adversarial network in 2014. They applied its neural networks to GANs and showed that GANs can be used to learn from images or videos with little or no prior knowledge about pixels or objects.

The points on one side of the segmentation plane are identified as category 1, and the points on the other side are identified as category 2. We use some algorithms to calculate the change amount for attack samples when generating images using GAN.

In the case of image generation, we use some algorithms to calculate the change amount for the specified models when generating attack samples. GAN is also an unsupervised learning model widely used in unsupervised and semi-supervised learning. At present, an interesting application is to use GAN in image style migration, image noise reduction and repair, and image super-resolution, which have better results in face recognition. Face recognition technology has been continuously studied using new data sources under natural conditions.

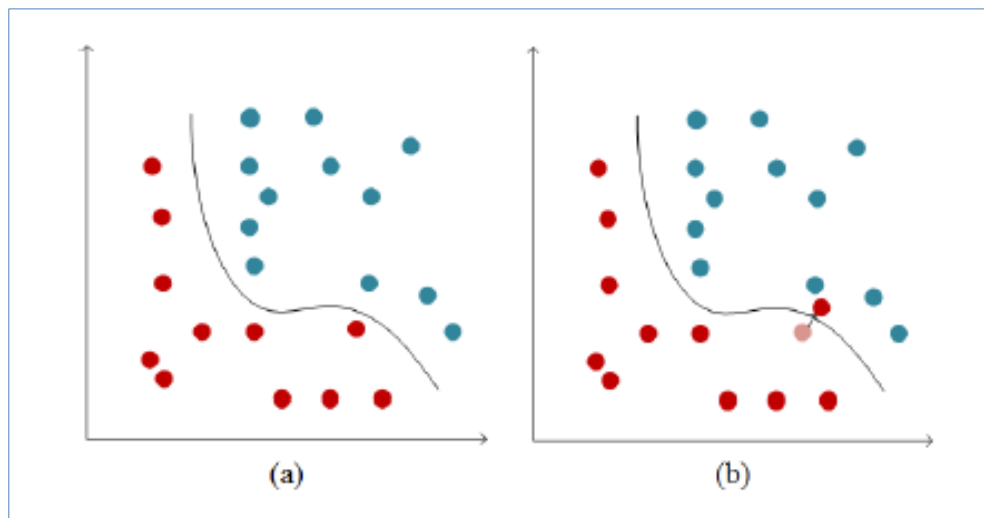


Figure 3. Current project flow diagram of attendance system in the experiment

The Generative Adversarial Network (GAN) is a model of generative adversarial networks. The main functions of GANs are shown below. GANs are generative neural networks that receive random noise z and generate an image through this noise. Discriminative neural networks are biased neural networks that judge whether a picture is "real". Their input parameter is x , representing a picture, and the output $D(x)$ represents the probability that x is an accurate picture. One represents 100% of the precise picture; if it is zero, it means it cannot express anything resembling a shot.

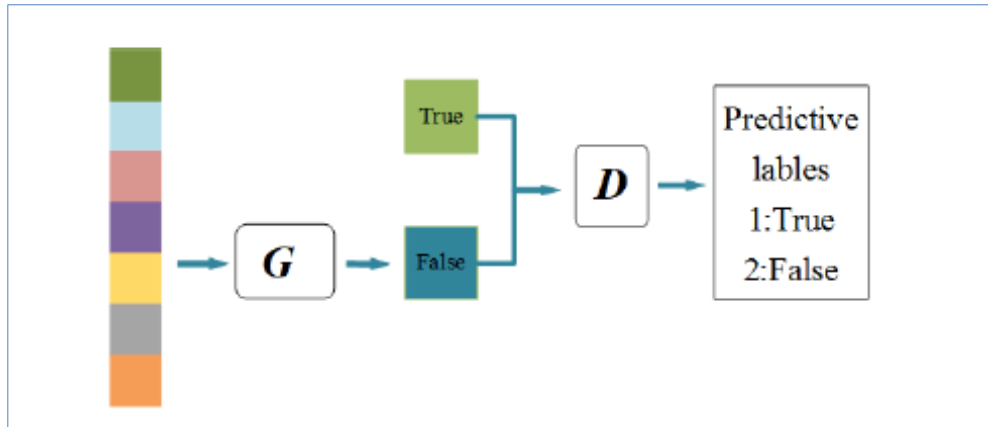


Figure 4. Cgan model, where G is the main function and D are the presented as follow, G is supposed to be Generative Network

Project #3

They obtained this experimental report from the paper "Morphing Detection Using a General-Purpose Face Recognition System". This section describes how we collected and analyzed the data for our research. We used the multi-PIE dataset as the basis for facial morphs. It was automatically generated by stitching the images together. The procedure used to collect this data was like the procedure required for eMRTD. Therefore, we expect our results to be the same as those obtained with such an approach. The original dataset contains images from 337 subjects from four sessions, including changes in viewing angle, lighting, and facial expressions. This work only considers neutral poses, facial expressions, and front lighting ideas.

Their algorithm uses DLib shape prediction and facial features to identify border areas. The boundary area is refined with a constant blend factor of 0.5. Use only the image from the first session and limit it to motifs that will be displayed in at least one subsequent session. In addition, to avoid unnatural image artifacts, consider only non-spectacle wearing issues that belong to the same gender. The dataset is organized into five train test splits to avoid unnecessary correlation between trains and test sets.

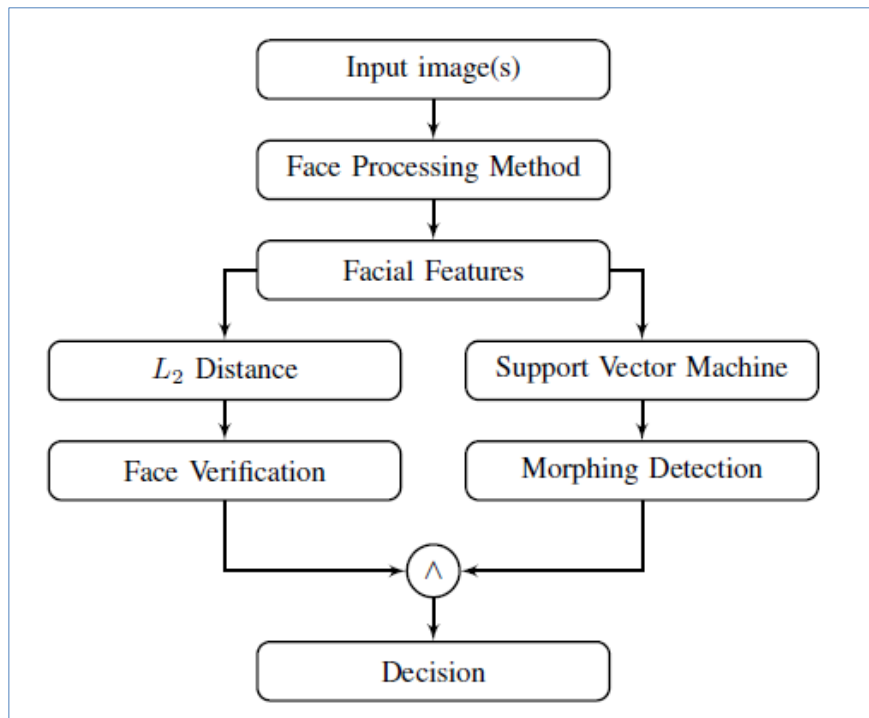


Figure 5. The current figure represents the researchers approach for combining face recognition and morphing detection in one single system.

The image shows our approach to combining face verification and morphing detection in one system. One of the methods for face recognition we introduce can replace face processing: FaceNet, DLib, VG-Face or High-Dim LBP.

SVM passed by reference vector image, which outputs a decision for morphing detection. These binary choices are combined to produce the final decision, either an acceptance or rejection of the individual.

CHAPTER 3

METHODOLOGY

3.1 Project background

Facebook has developed sophisticated facial recognition software. In the old days, Facebook users will need just to upload the pictures meanwhile facebook will tag their friends automatically. While an image is uploaded, Facebook recognises faces automatically so that users can like or comment on the pictures without clicking on individual faces.



Figure 6. This images shows two doifferent persons with almost identic faces

Face recognition technology is a fantastic innovation. By tagging a face with a friend's name, you can make it so that other applications will recognise that person's face and allow access to their profile. The accuracy rate of this technology is 98%, which is as good as humans can do.

3.2 Using Machine Learning to solve highly complex problem

Machine learning was used in many problems. By gathering the data and putting it on the machine learning algorithm we will solve all the problems and get the best result. The data is fed into the program to use its capabilities to solve the problem.

What we have in our project is face recognition which is completed by solving several different cases.

- Look at the photo first and calculate all the faces you see there
- The next step is to pay attention. Even if the person is pointing in a strange direction or the lighting is inadequate, they should look the same person.
- Third, you need to be able to identify unique facial features such as eye size and whether the mouth is high on one side and low on the other.
- Finally, determine the name by comparing it to every face that knows the face.

Like humans, our brains are wired to recognise faces automatically and instantly. In addition to recognising faces, humans also tend to recognise everyday objects as faces—faces that may not be there at all. To resolve each step, you need to build a separate pipeline for the face recognition process and pass the resulting results to the next step. So, to get better results, you need to combine some machine learning algorithms.

[1]

3.2.1 Systematic face recognition

As we said before, in all the entire paper we will see everything behind the scene, how we used all the algorithms and what are the result of using face recognition by the help of python (PYcharm) .

3.2.2 Systematic face recognition

Face detection is used in many photography applications, such as taking pictures of people it detects all the peoples inside the camera screen directly before photo shot.

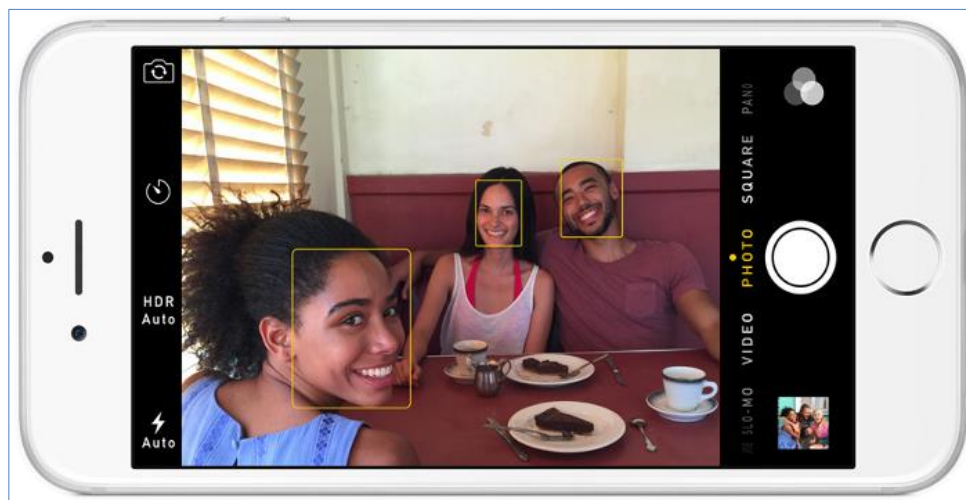


Figure 7. People taking selfie yusing a digital camera from mobile phone

We can use the face detection feature of a camera for many purposes, including finding faces within an image. We will use the HOG (Histogram of Oriented Gradients) algorithm which is developed in 2005 and will help us in further solution. [2]

The feature of face detection is excellent for cameras. By picking out faces in the image, the camera nowadays can detect all people in a screen before shoting, but here we

are using for different needs, discovering new areas for further analysis and continuing our pipeline stages.

The process of face detection was first developed in 2000 by Paul Viola and Michael Jones. By their collaboration they invented a method algorithm to detect faces in within a high optimised and cost-effective infrastructure that would work even on very cheap cameras [3]. Relying on colour data would have compromised the accuracy of their solution, so they used black-and-white images instead. Our method is the alternative method. This method is intended for use with colour images; it does not depend on colour data but instead makes use of shape data from the image. To find a face in an image, you don't need the color data to display the face, so first make the image black and white. [3]



Figure 8. Image used for some testing

In order to analyse every pixel in our image, we will determine which pixels were directly surrounding it as shown on the figure below (fig 9).

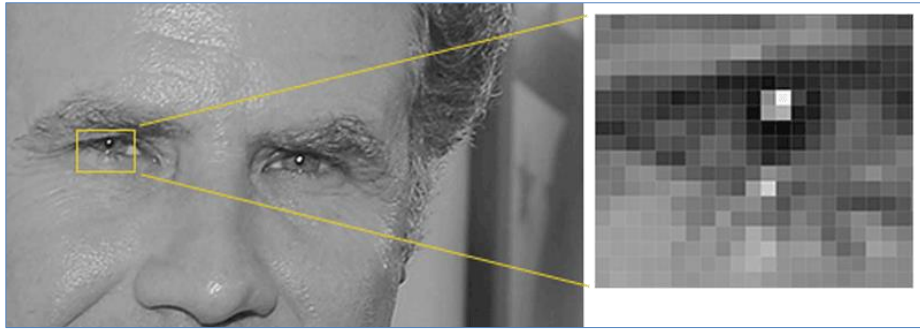


Figure 9. Image used for some testing

To get the best result our technique is to catch the darkest pixels and what pixels are surrounding it. Then we draw an arrow showing in which direction the image is getting darker:

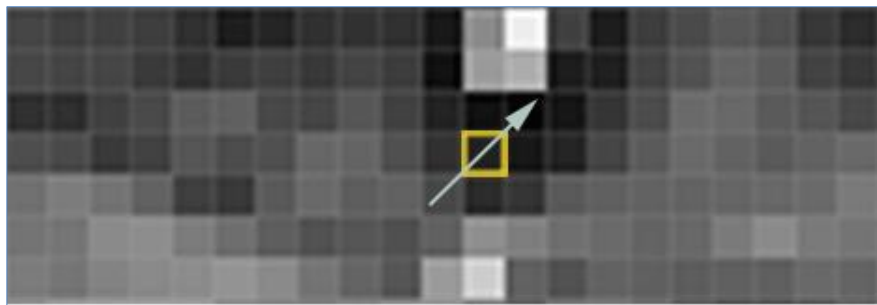


Figure 10. The arrow shows the direction when it is getting darker

Gradients are a way to apply colour to an image by replacing each pixel with one of two colours. So, in this step as we seen in figure 10, all the pixels are replaced by an arrow [5].



Figure 11. Image that represents the key points

There the image 11 will be replaced with pixel gradients

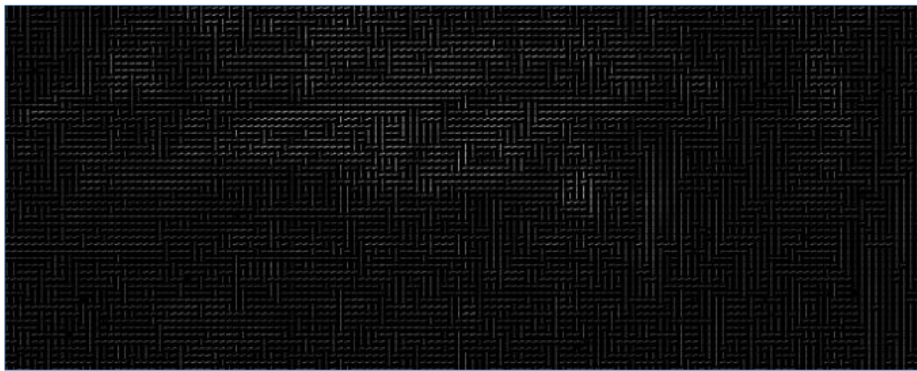


Figure 12. Image replaced by pixel gradients to extract those key components

Replacing pixels with gradients can help solve the problem of representing images from a group of people with different brightness levels. If we analyse pixels directly, the same person's dark and light photos have not the same values. [6]

But in the end the dark and bright photos will have the same representation by only considering how the brightness of photo changes. It will help the computer to solve.

We will separate the images into small square images 16X16. Next step is to count the gradients of each direction (upward, upward rightward, rightward). The algorithm will replace each square with an arrow indicating which direction was strongest.

However, when we save the gradient for every pixel in a grayscale image, we can lose some data and these can affect the accuracy of algorithm. We see only what happens briefly; we do not see the entire picture. The basic pattern of lightness/darkness is lost, and we cannot truly appreciate it as an object. [3]

There are two ways to do this: (1) count how many gradients point in each square in each significant direction (2) count how many gradients point in each square in each direction except for one that you deem significant (three points up, three points right). Then we replace that square in the image with arrows pointing in those directions that were strongest.

The result is a straightforward representation of the face that represents only the essential features of its structure

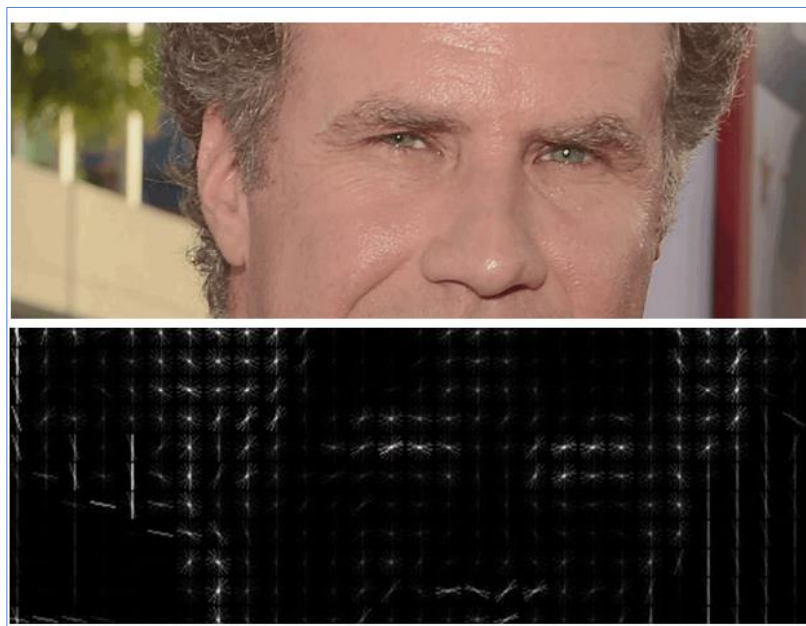


Figure 13. Converting the original image By HOG algorithm.

To extract faces from a HOG image, we need to find the part of our image that looks the most like a known HOG pattern that the Algorithm will extract from a bunch of other images used to train.

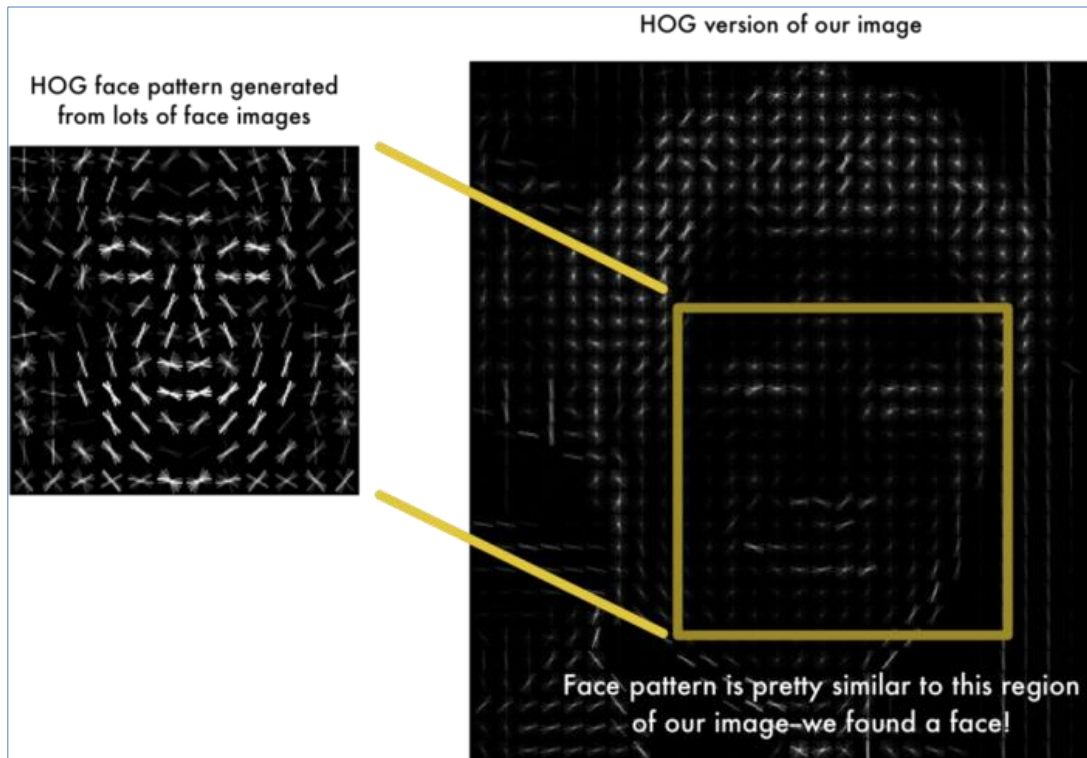


Figure 14. Using the HOG face pattern methodology to detect the faces.

This technique allows us to find faces in any image easily:



Figure 15. HOG Representation of processed images.

3.2.3 Projecting and showing faces

By seeing the images, the face isolated images now there is a problem that we will face the faces differently to a computer or device depending on different orientation.

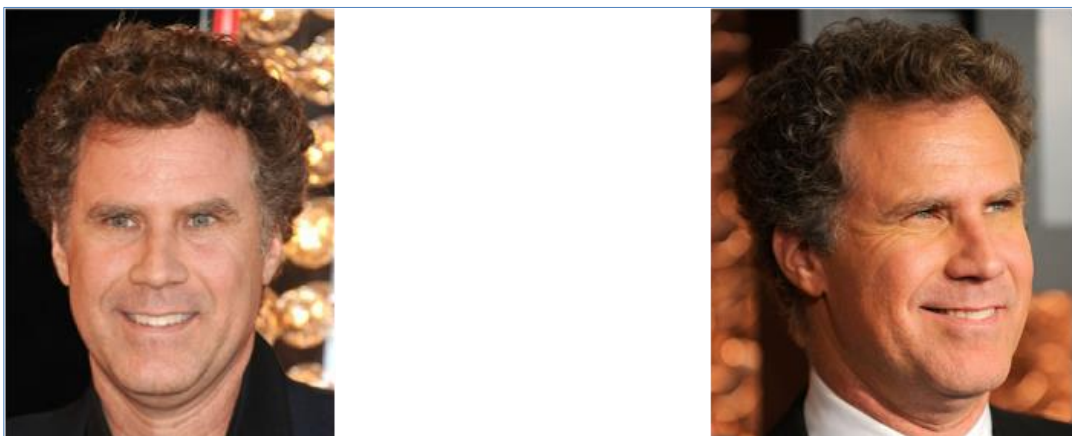


Figure 16. Humans can recognise both the two images, but the computers will differently see the both images of will ferrell.

To illustrate this, we use an algorithm called FaceLandmarkEstimation. This method was developed in 2014 by Vahid Kazemi and Josephine Sullivan. The basic idea behind this approach is to find 68 specific points (called landmarks) on each face. The top of the chin, the outer edge of each eye, the inner edge of each eyebrow, etc. The machine learning algorithm then finds these 68 and finds a specific point on each face. Use this to determine how a particular person is like another in terms of age and gender. [6]

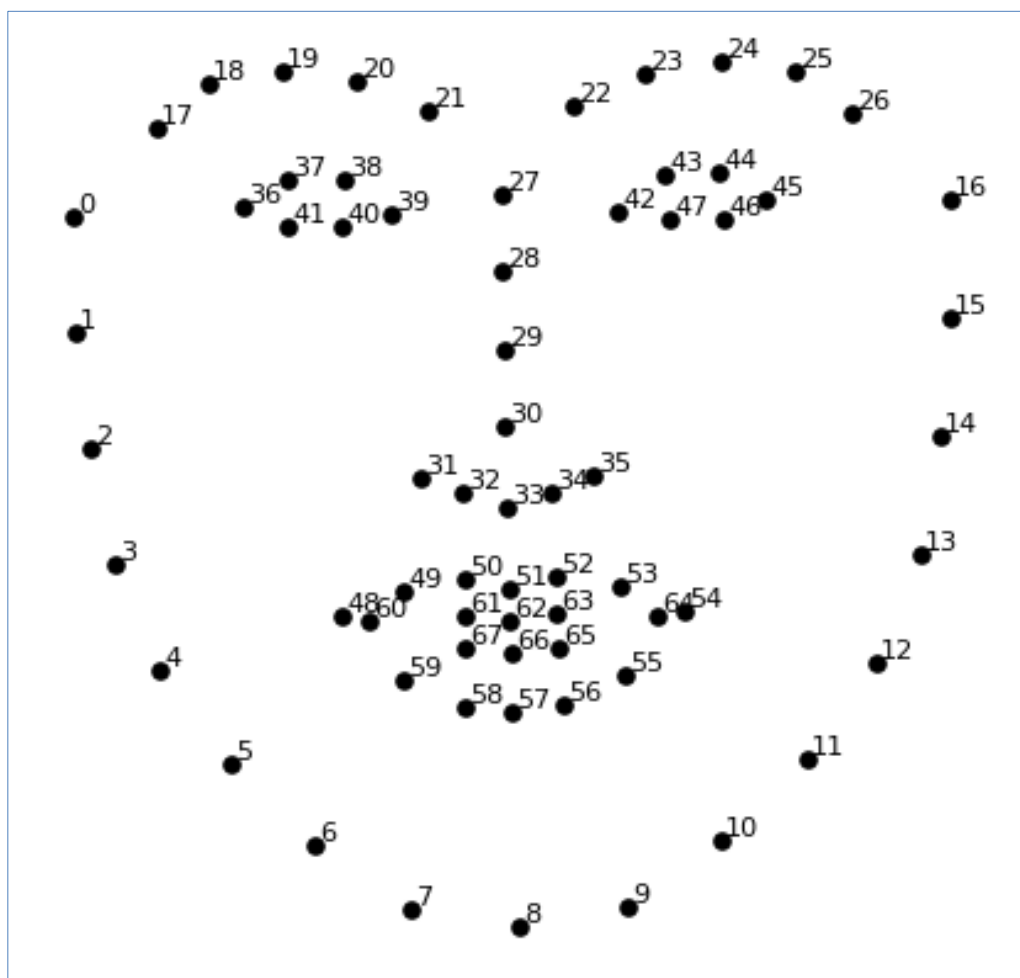


Figure 17. We will see that there in the upper image will be the results of 68 located face landmarks of testing images.

To locating the 68, face landmarks on or testing images in the image (17) number.

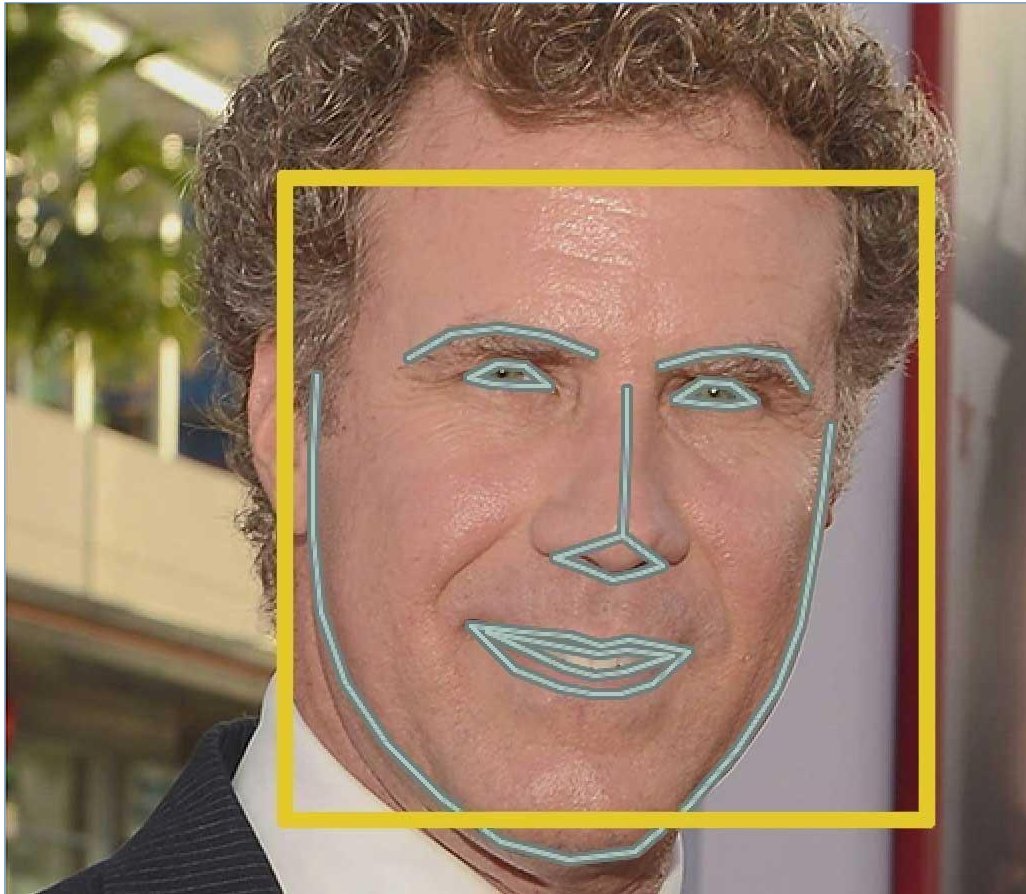


Figure 18. Snapchat uses also the same 3D technique for realtime video processing filters.

Now that you know where the eyes, mouth, and chin are in the image, you can rotate, scale, and shear the image to center the eyes, mouth, and chin as much as possible. Do not do flashy warps as the image will be distorted. Use only affine transformations.
[7]

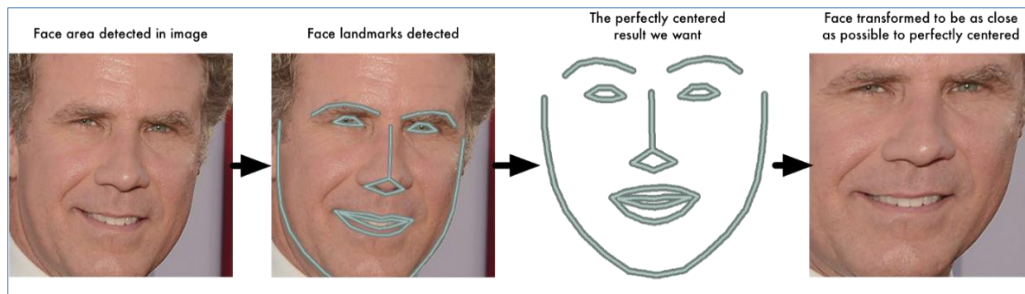


Figure 19. We can centre the eyes and mouth roughly in the same position in the image.

Here is the code for finding face landmarks and transforming an image using the same landmarks.

3.2.4 Encoding face part

Now we have reached the heart of the face-distinguishing problem. This is an interesting place! The easiest way to compare unfamiliar faces with all the images of people who are already tagged is to compare them directly. However, this approach has major flaws. Facebook and other social media sites need to recognize faces in milliseconds instead of hours. [3]

First, we need a way to extract the basic features measured from each face (ear size, eye relief, nose length, etc.). Then measure the unknown face to find the known face with the most similar dimensions (such as ear size).



Figure 20. Face recognition on a TV show

Now To build a reliable face database, we need to collect measurements from each face. Ear size, nose length and eye colour are three types of measurements we can use. We need to decide which measurement is most reliable for a particular use.

In studies, researchers have found that computers are more accurate than humans at measuring facial features. Deep learning algorithms can identify which part of the face is essential for measurement.

Train DCNN (Deep Convolutional Neural Network) as in Part 3. Here we need to generate 128 measurements for each unique face. One approach is to train DCNN (Deep Convolutional Neural Network) as in Part 3. Use this DCNN to generate 128 features for each face.

The training process works by looking at three face images at a time:

- Loading a known person's face image.
- Loading another picture of the same known person.
- Loading a picture of another person to compare and get a result.

The algorithm then examines the measurements currently being generated for these three images. Then tweak the neural network slightly to produce a measurement close to that of image # 1, making sure it is slightly further away than image # 2.:

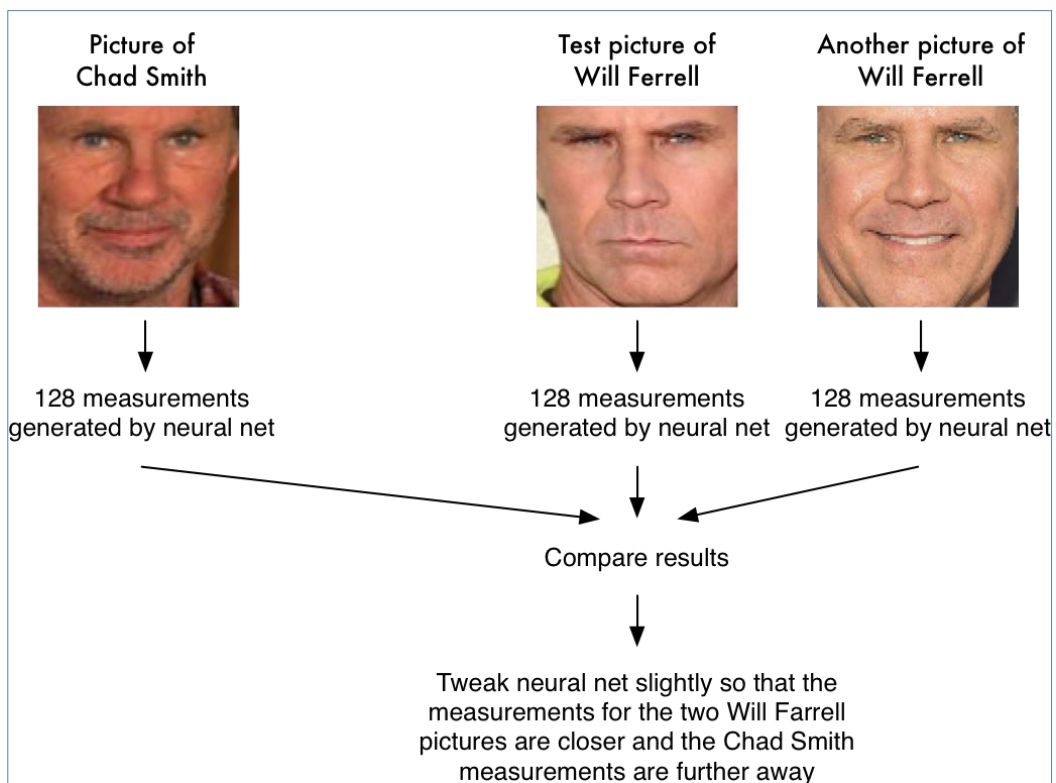


Figure 21. A single triplet-training step. Repeating the same step many times for different images and different peoples, the neural network will learn and generate 128 measurements of each person.

Repeating the same step, a lot of times for different images and different peoples the Neural Network will learn and generate 128 measurements of each person the following results from this process should be like ten pictures of the same person taken under different conditions. [8] Google researchers have developed an algorithm that reduces complex raw data, such as images, into a computer-generated list of numbers called embeddings. This is used in many machines learning applications, including language translation.

3.2.5 Econcoding our face image

Convolutional neural networks must be trained to recognize faces. One of the most difficult processes is training the data to output face because we sum up on using too much time and computer power. [9]

While we test with the best graphical card would take around 24h continuous training process to achieve a good accuracy. Once the training of network finishes, it will generate the measurements of each face even for face that are not seen anywhere or used before. So, this step will be done only once.

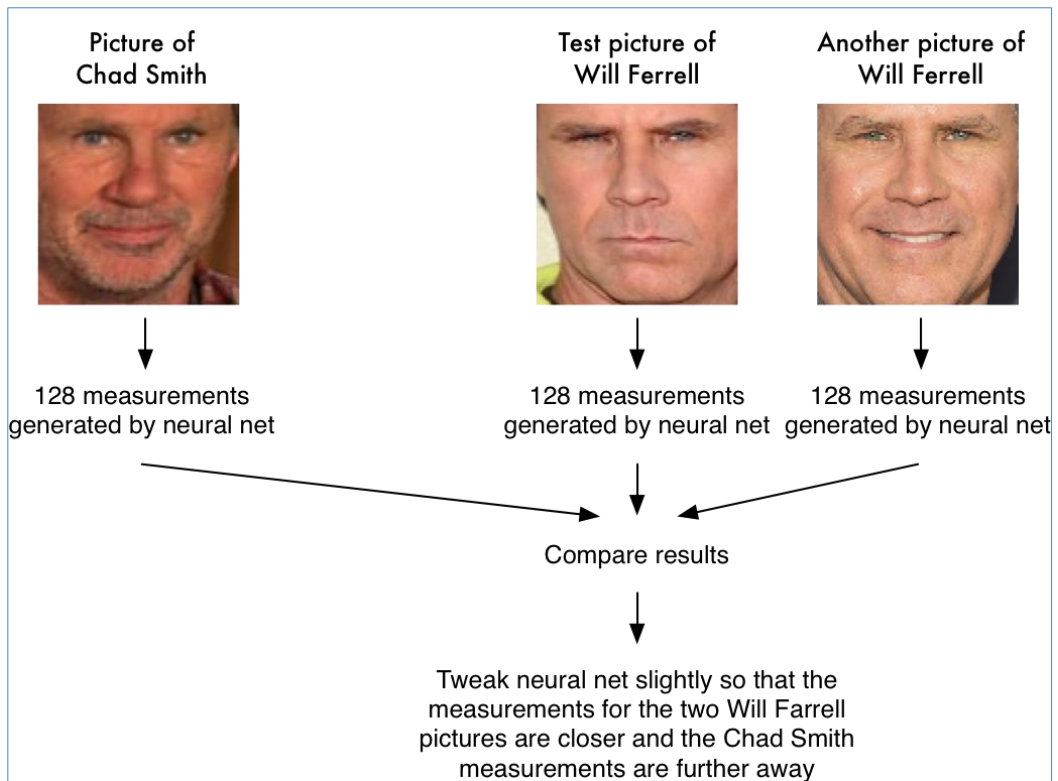


Figure 22. 128 measurements from the image

The network does not measure all parts of the face accurately; it only measures 128 points. This is not important because we do not care about how many pixels are measured on each point. By catching the same person two times we will get the same result.

3.3 System testing and analysis

Verify that the images are placed correctly on the respective folders, and we can start testing part. Checking the output records if the updates are done correctly within a respective timeframe. [9]

We run the algorithm and check the list of records, everything was updated respectively, and the images of each person were successfully recorded. We were able to successfully capture live information's shown in the video figure *Figure 23*.

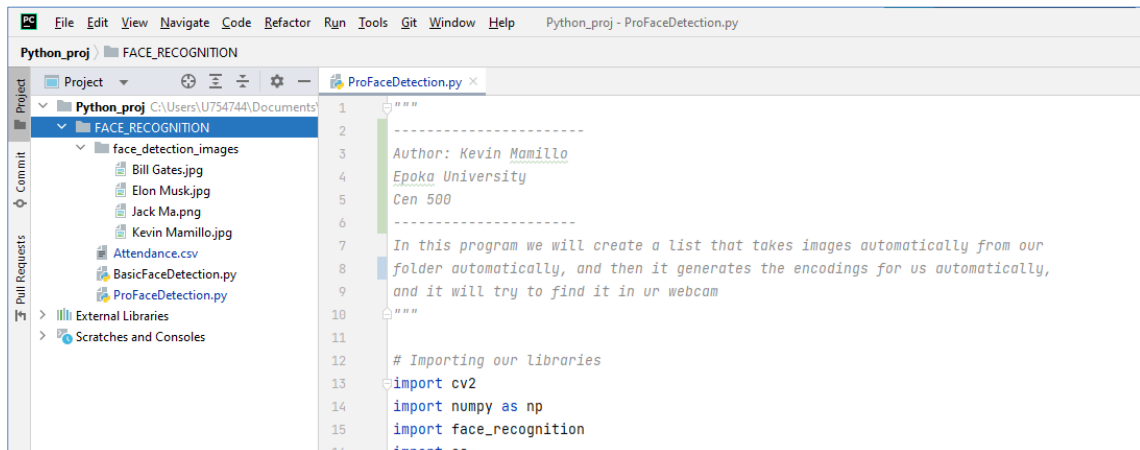
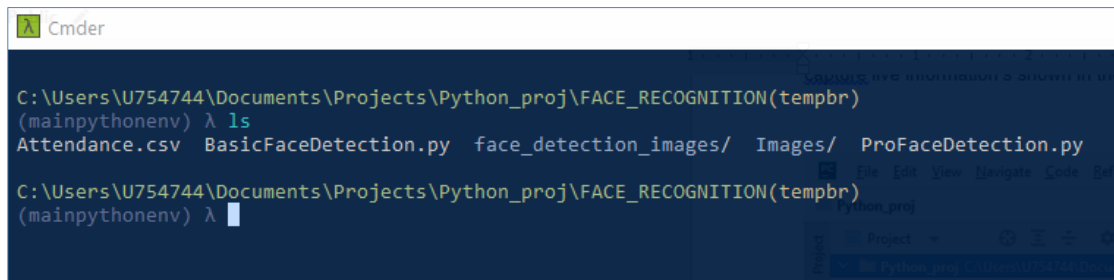


Figure 23. First structure of project code

Part of the algorithm implemented and running. A virtual environment on python 3.8.8 executed the code.

Running the tests,



```
C:\Users\U754744\Documents\Projects\Python_proj\FACE_RECOGNITION(tempbr)
(mainpythonenv) λ ls
Attendance.csv BasicFaceDetection.py face_detection_images/ Images/ ProFaceDetection.py
C:\Users\U754744\Documents\Projects\Python_proj\FACE_RECOGNITION(tempbr)
(mainpythonenv) λ
```

Figure 24. Code run part 1

Images are taken from the directory and processed, each filename signify the name and surname.

We can see that there are four columns of values are the current face coordinates and similarities that represent the accuracy.

According to the test cases, in this case, there are faces that does not occurs in the archive and the persons appear to be unknown because of the high values:

```
Cmder
C:\Users\U754744\Documents\Projects\Python_proj\FACE_RECOGNITION(tempbr)
(mainpythonenv) λ python ProFaceDetection.py
['Bill Gates.jpg', 'Elon Musk.jpg', 'Jack Ma.png', 'Kevin Mamillo.jpg']
['Bill Gates', 'Elon Musk', 'Jack Ma', 'Kevin Mamillo']
Encoding Complete
[0.80994245 0.75682206 0.91821948 0.71238657]
[0.78493524 0.74157607 0.9197509 0.65864454]
[0.73590585 0.7715143 0.87937346 0.610151 ]
[0.77037361 0.77957693 0.85297734 0.6871399 ]
[0.77161673 0.77844177 0.84122865 0.67244135]
[0.78604428 0.78047194 0.85728713 0.6977196 ]
[0.74553616 0.77873494 0.84511746 0.67020434]
[0.79403954 0.78656621 0.85709638 0.68407425]
[0.78635092 0.81796968 0.8652632 0.70948101]
[0.77424643 0.78365605 0.85335276 0.68803278]
[0.77602927 0.77537503 0.829183 0.70509654]
[0.81284507 0.81672342 0.85776928 0.6994381 ]
[0.7897417 0.80746725 0.89493532 0.70378669]
[0.79436195 0.78227135 0.8712346 0.6955896 ]
[0.8698337 0.80173607 0.89399216 0.73914995]
[0.81941454 0.82108654 0.89893264 0.73191371]
[0.83780112 0.79863892 0.8974026 0.74052778]
[0.80549211 0.81649497 0.88919074 0.73391137]
[0.79027959 0.76206112 0.86328568 0.69795187]
[0.77976577 0.78641805 0.86547663 0.70565775]
[0.80047635 0.76984028 0.87785299 0.68974078]
[0.74779351 0.7570426 0.87316041 0.71987763]
[0.75584245 0.7268143 0.83947717 0.69722081]
[0.78711399 0.78208694 0.87888746 0.71747801]
[0.83644648 0.8307507 0.8908973 0.72550061]
[0.8561668 0.78458933 0.8737434 0.69395015]
[0.82152961 0.77301988 0.86275452 0.67343576]
[0.81549935 0.71370543 0.90282628 0.649781 ]
[0.78824273 0.71710426 0.84142636 0.65859525]
[0.78407892 0.72283428 0.8264171 0.67633097]
[0.76340243 0.71907789 0.8971833 0.67862369]
[0.8012573 0.75764436 0.90072051 0.68978646]
[0.79424862 0.73762064 0.89513289 0.65165979]
[0.75162211 0.65645361 0.87306587 0.68352728]
[0.78072239 0.71863008 0.87532501 0.64005984]
```

Figure 25. Code run part 1

When we test and appear the face in front of the live camera, we can see that the system detects our face and it shows the name stored of the original stored images as a training dataset to be compared with the live images.

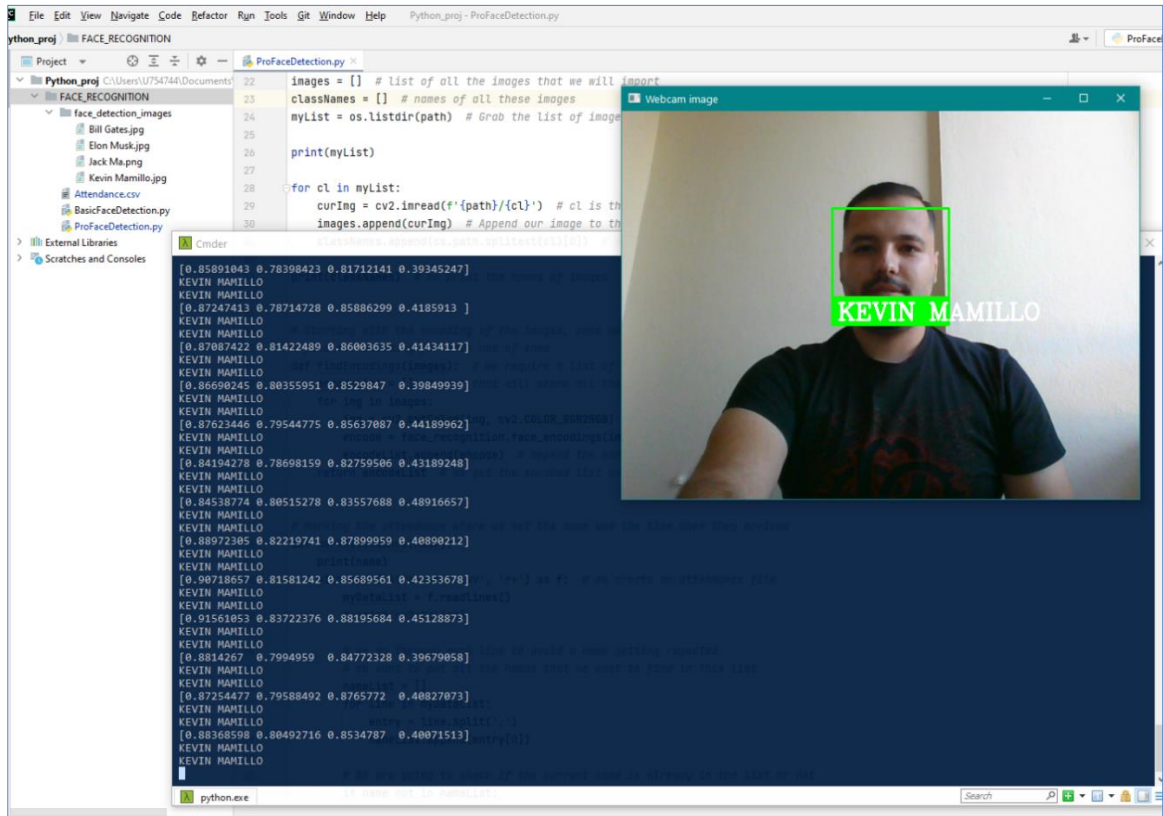


Figure 26. The test was run successfully and we got the expected results.

3.4 Eigenvalues and Eigenvector analyze of faces

Considering a set of images in the figure (x) that will be used as a training set of images that are take from 4 different peoples from the “Troy” movie with screenshots of faces.

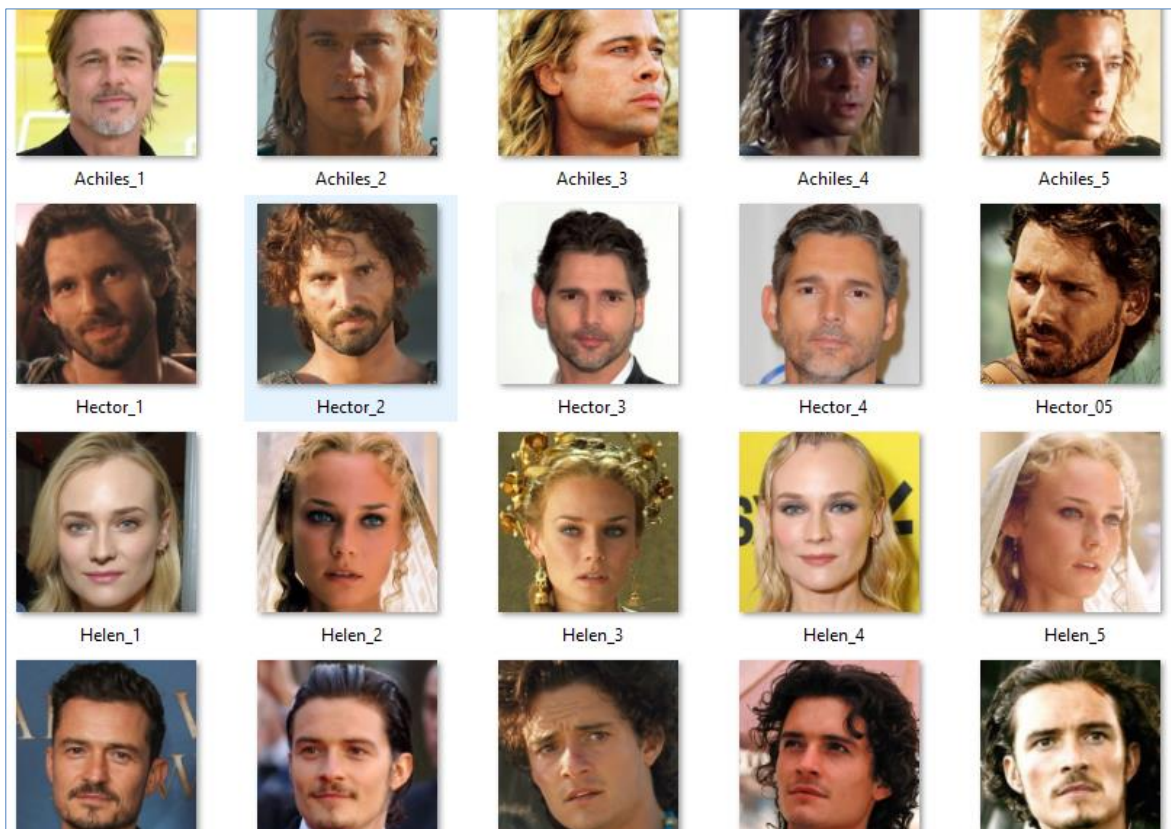


Figure 27. Five images of four different faces which include part of Troy.

The main point of this further experiment is by making the eigenface methodology to perform better. There are used 5 images of 4 individuals.

We can use SVD to decompose high dimensional data matrices X and we will use python to implement the eigenfaces concept.

IF X is a matrix or a library of human faces where each column is a human face. We make use of SVD in the principal component of that data to find a new representation of human faces to be called the Eigen face representation. [10]

We begin by importing all our 20 images and converting into the same number of pixels by resizing them into 120 pixels and 80 pixels directions. `pic.jpg`, converts it into grayscale and turns the image format into double precision numbers for manipulation. Resizing is also accomplished.

```
A1=imresize(double(rgb2gray(imread('D:\DEV\MATLAB\Eigenimages\Achiles_1.jpg','jpg'))), [120 120]);
```

This preprocessing is done for each picture to create an image matrix A . A convenient way to view the image is with the `pcolor` command

```
pcolor(flipud(A1)), shading interp, colormap(gray);  
set(gca,'Xtick',[1],'Ytick',[1]);
```

Note that the `flipud` command flips the image to be right-side-up for visualization. Further, the `imresize` command is part of the image processing toolbox.

To understand face recognition, we can first understand the concept of an average face. [11]

Based on **Figure 27** we will take the average face of each person for each of their images by getting the average of each image.

```
AveAc=(A1+A2+A3+A4+A5)/5;
```

```
AveHe=(B1+B2+B3+B4+B5)/5;
```

```
AveHel=(C1+C2+C3+C4+C5)/5;
```

```
AvePa=(D1+D2+D3+D4+D5)/5;
```

In the *Figure 28* are generated the average image of each face. If we can see in detail the images retain the fundamental aspects of the individuals.

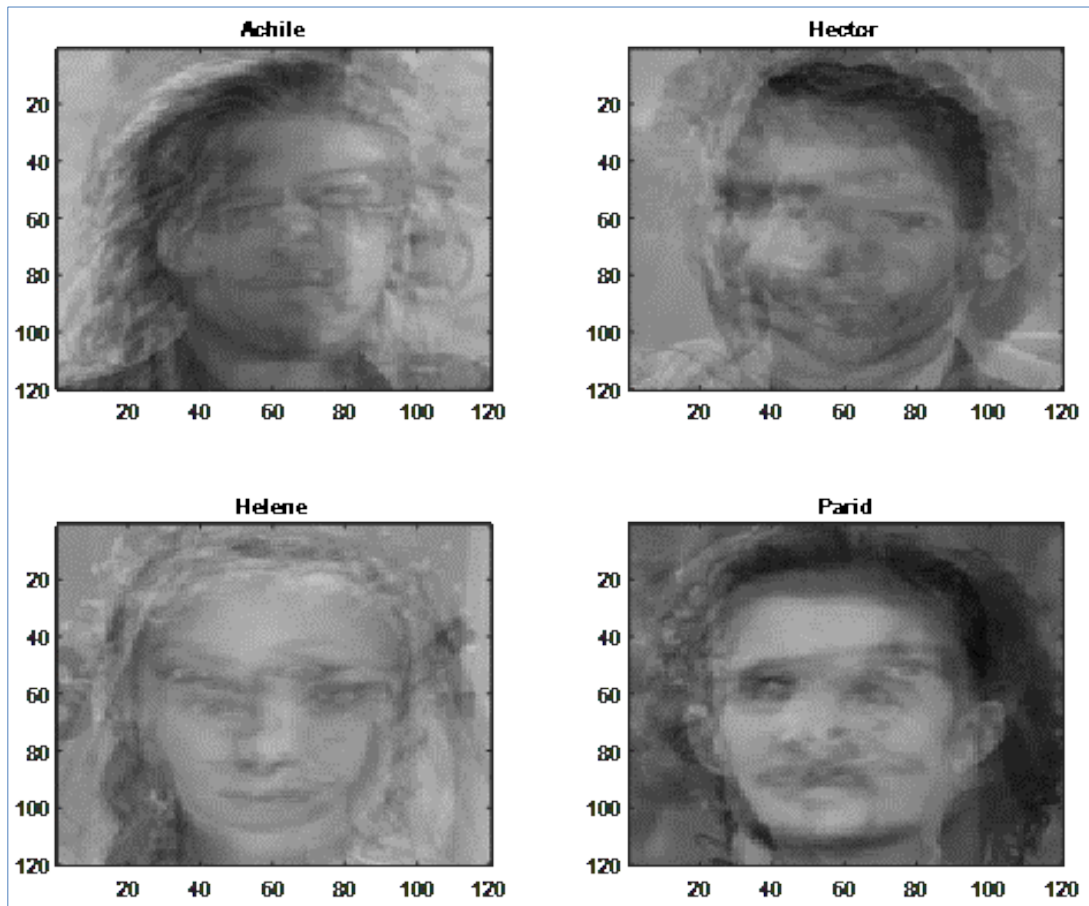


Figure 28. Average images generated from 4 actors in the Troy movie.

```
%Averages of all images plotted
```

```
subplot(2,2,1)  
image(AveAc)  
title('Achile');
```

```
subplot(2,2,2)  
image(AveHe)  
title('Hector');
```

```
subplot(2,2,3)  
image(AveHel)  
title('Helene');
```

```
subplot(2,2,4)
image(AvePa)
title('Parid');
```

3.4.1 Singular value decomposition

Singular value decomposition has applications in image compression. Here I'll give a bit more explanation of how that works and showcase some of the tools for manipulating images in python. The key here is that a black and white image is just a matrix where the numbers represent the intensity of a given pixel, which can be decomposed just like any other.

Importing the main libraries, we proceed with our analyse,

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import time

from PIL import Image
```

Here we load an image of the Parid, prince of “Troy” and convert it to black and white.

```
img = Image.open('input_images/Parid_3.jpg')
imggray = img.convert('LA')
plt.figure(figsize=(9, 6))
plt.imshow(imggray);
```

By running the following commands, we get just the

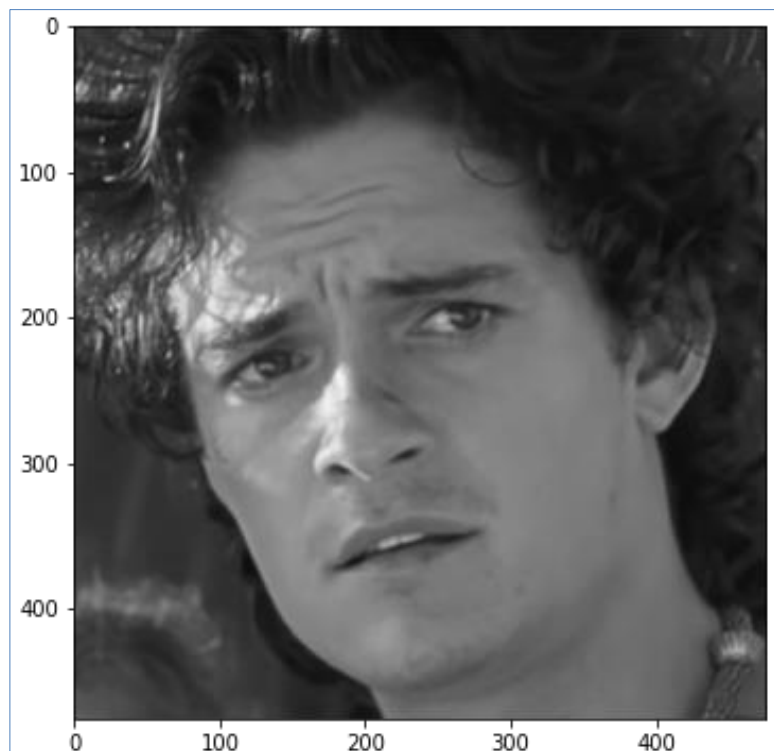


Figure 29. First step of the image converted black and white.

Now we will convert the image data into a numpy matrix, plotting the result to show the data is unchanged.

```
imgmat = np.array(list(imggray.getdata(band=0)), float)
imgmat.shape = (imggray.size[1], imggray.size[0])
imgmat = np.matrix(imgmat)
plt.figure(figsize=(9,6))
plt.imshow(imgmat, cmap='gray');
```

Now to compute the singular value decomposition we use the following command:

```
U, sigma, V = np.linalg.svd(imgmat)
```

Computing an approximation of the image using the first column of U and first row of V reproduces the most prominent feature of the image, the light area on top and the dark area on the bottom. The darkness of the arch causes the extra darkness in the middle of the reconstruction. Each column of pixels in this image is a different weighting of the same values, \vec{u}_1 :

```
reconstimg = np.matrix(U[:, :1]) * np.diag(sigma[:1]) *  
np.matrix(V[:1, :])  
plt.imshow(reconstimg, cmap='gray');
```

By running the following commands, we get just the

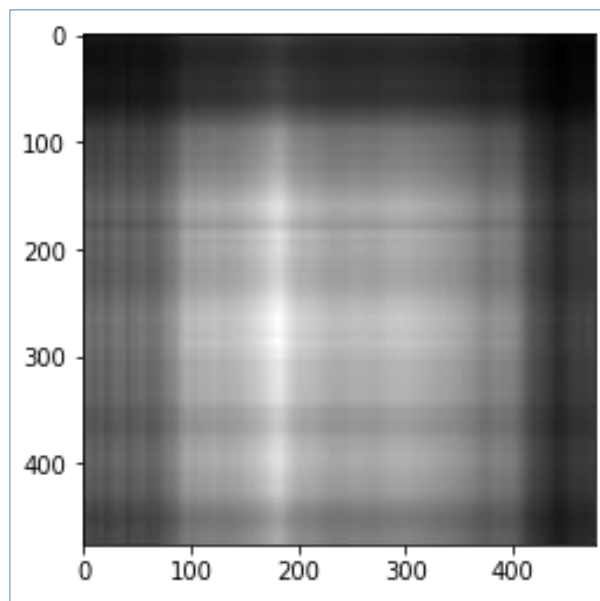


Figure 30. Getting an image where each column of pixels is different weighting of the same value.

Even with just the second and third vectors, the shape of the Parid's face it begins to appear.

```
for i in range(2, 4):
    reconstimg = np.matrix(U[:, :i]) * np.diag(sigma[:i])
* np.matrix(V[:i, :])
    plt.imshow(reconstimg, cmap='gray')
    title = "n = %s" % i
    plt.title(title)
    plt.show()
```

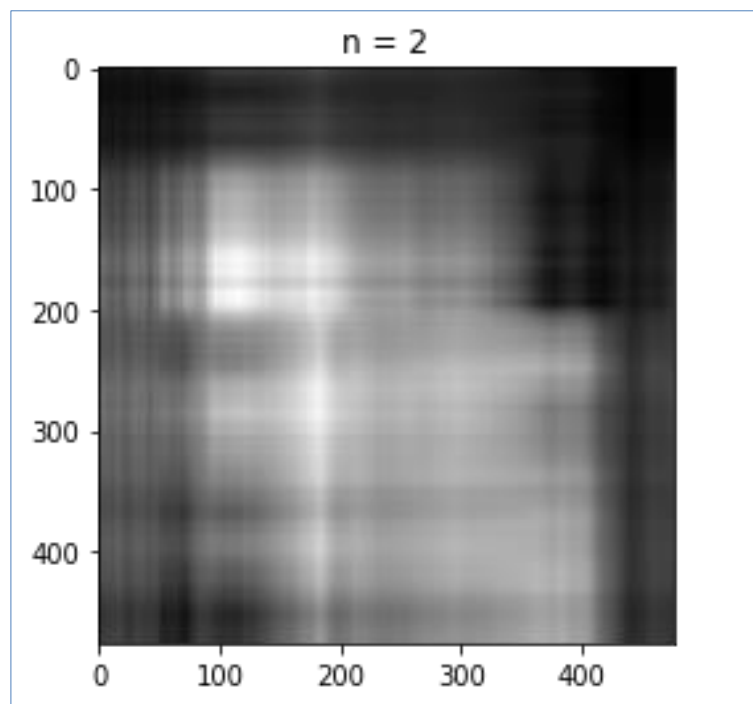


Figure 31. Reconstructed image using the first n vectors of the singular value decomposition, in this case $n = 2$

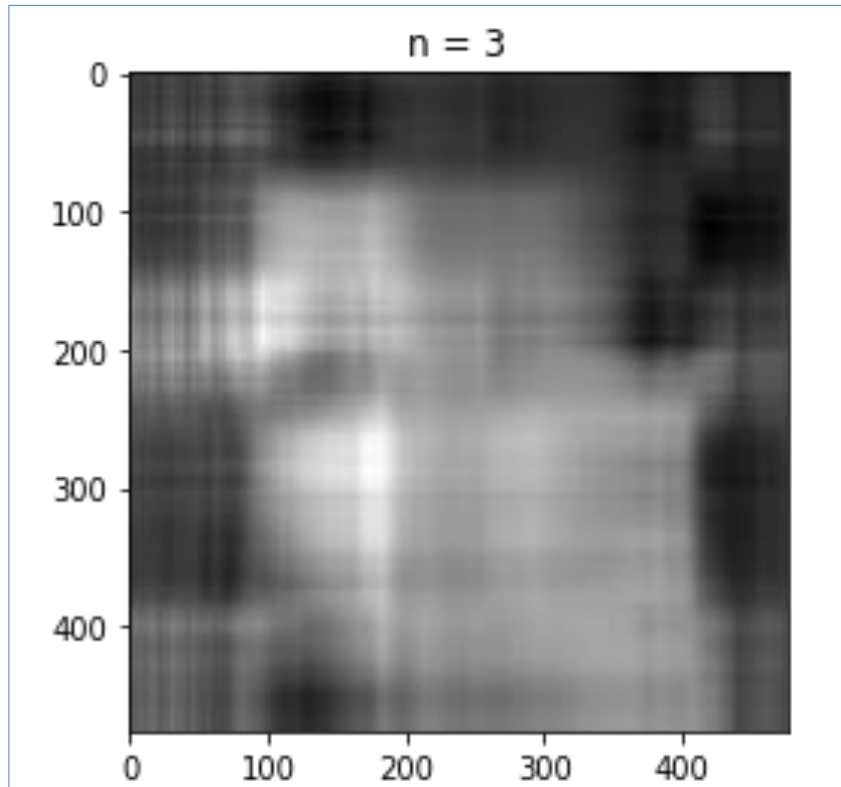


Figure 32. Reconstructed image using the first n vectors of the singular value decomposition, in this case $n = 2$

The loop below shows the reconstructed image using the first n vectors of the singular value decomposition (n is shown in the title of the plot). The first 50 vectors produce an image very close the original image,

while taking up only image.png as much space as $\frac{50*3900+50+50*2600}{3900*2600}$ the original data.

```
for i in range(5, 51, 5):
    reconstimg = np.matrix(U[:, :i]) * np.diag(sigma[:i])
    * np.matrix(V[:i, :])
```

```
plt.imshow(reconstimg, cmap='gray')
title = "n = %s" % i
plt.title(title)
plt.show()
```

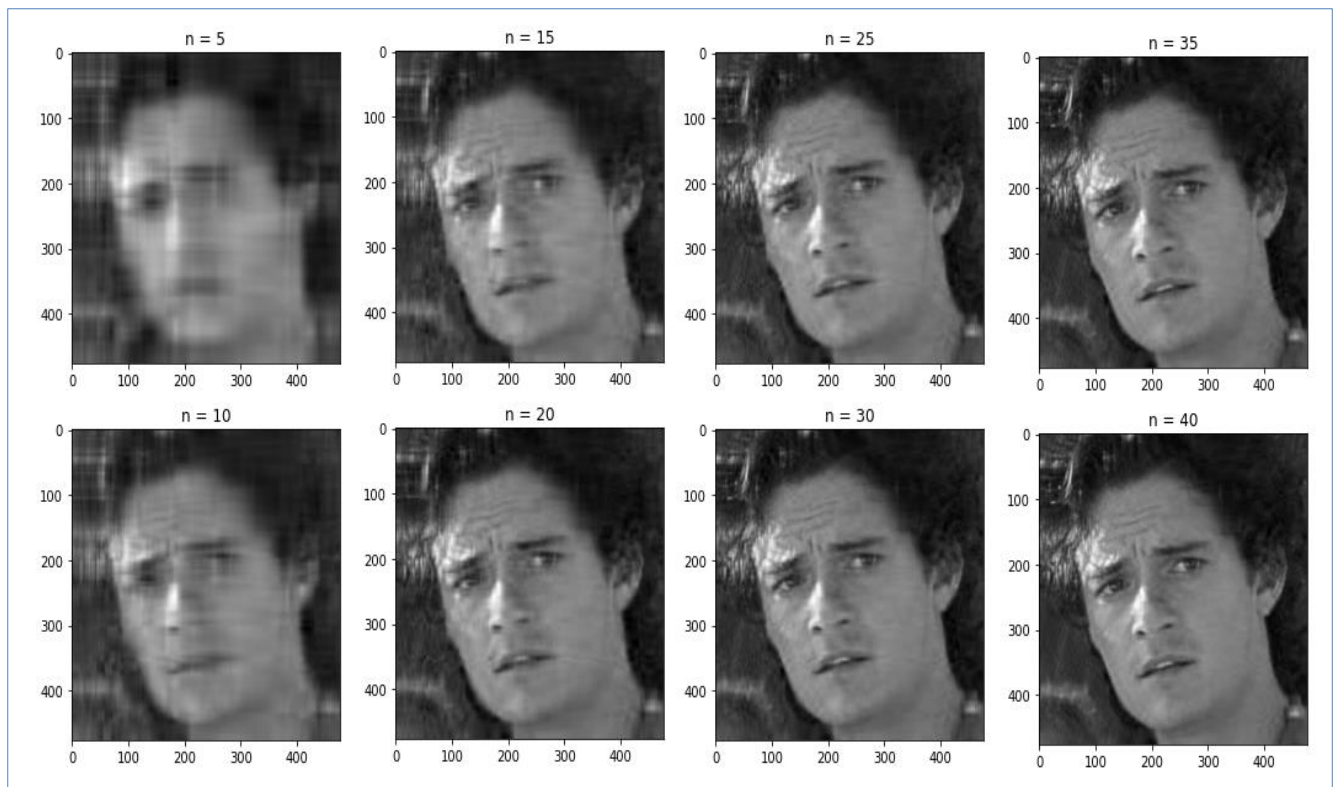


Figure 33. Reconstructed image using the first n vectors of the singular value decomposition, in this case the n will vary from 5 to 40

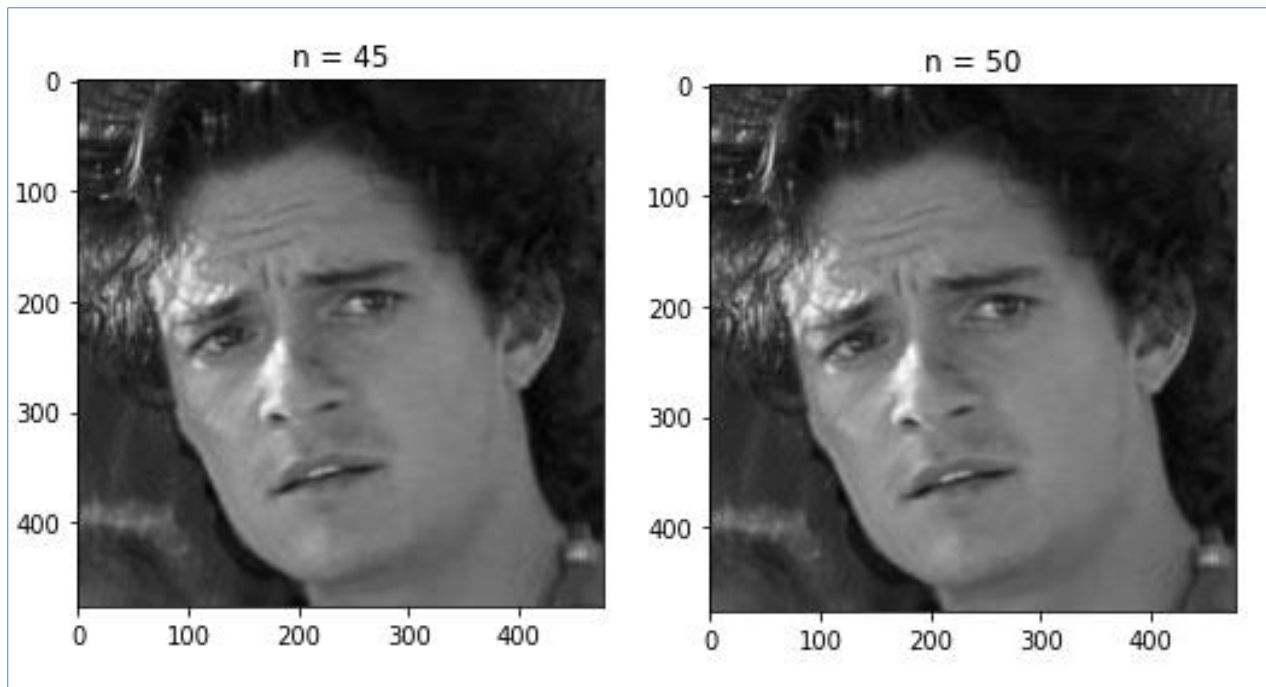


Figure 34. Reconstructed image using the first n vectors of the singular value decomposition, in this case the n will vary from 45 to 50

We clearly can see how the images are getting better and better, alsomst the same as original ones.

In this overview of what is eigenface, how SVD helps to construct it, and how it can be applied in image processing and face recognition areas. As I mentioned, this is only the fundamental step towards industrial applications, and various algorithms for preprocessing images are being developed. [8]

3.5 Face Recognition using SVD

Face recognition algorithm will use the basic object recognition method which is singular value decomposition. Using SVD function we can train our own images and perform SVD on that image. After that you can test using some mathematical operations with U(eigenface) vector returned by SVD function.

The dataset of the facial images is in "images_dataset". The code is show in "SVD_FaceRecognition.py" file. The images that we will test are in "test_images" folder. The accuracy achieved is ~85-90%.

After running the algorithm, we got out image successfully,

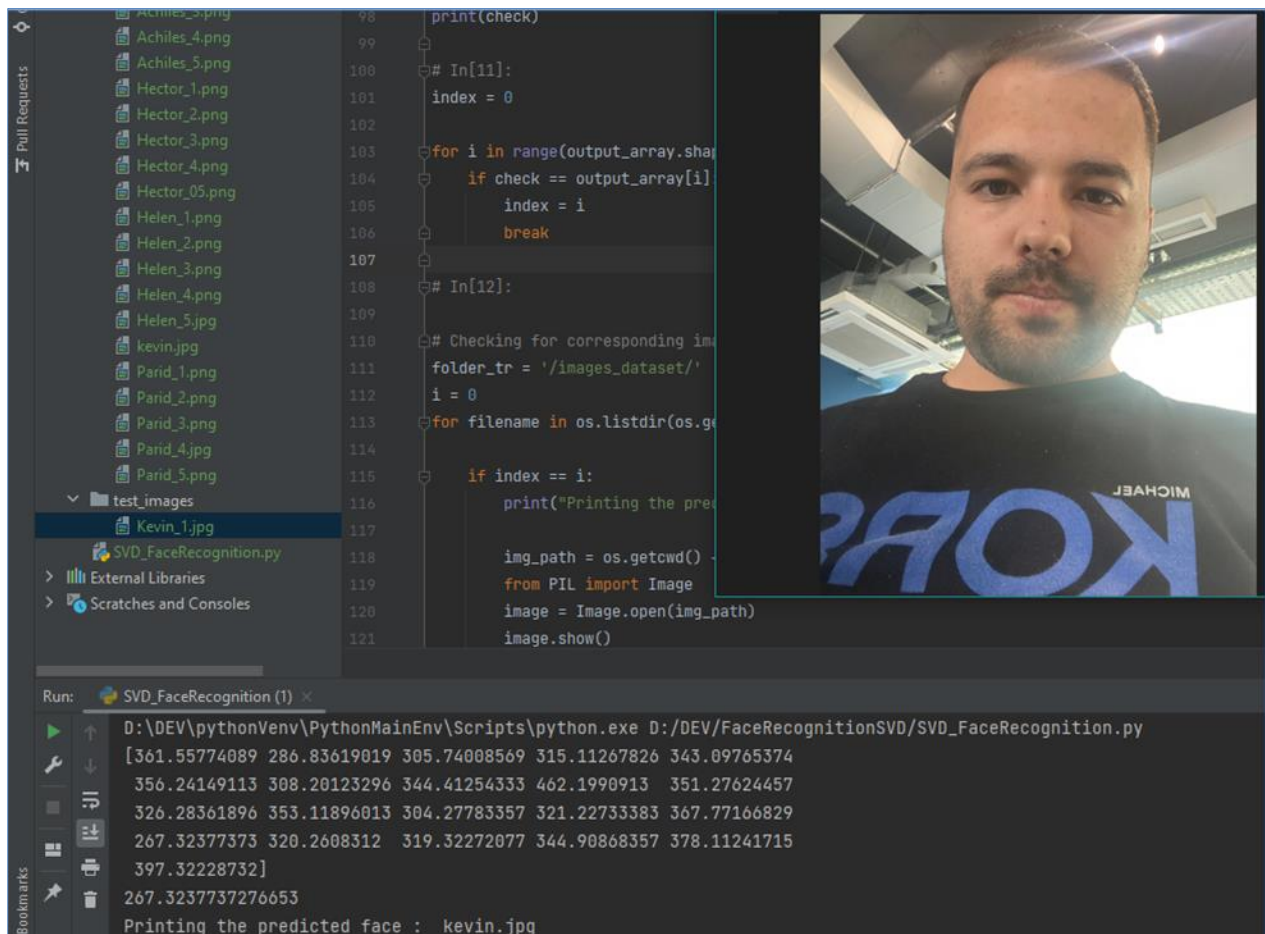


Figure 35. Reconstructed image using the first n vectors of the singular value decomposition, in this case the n will vary from 45 to 50

Project Description.

So, the idea I have done in this project is applying the concepts of vector space and subspace to face recognition. The set of known faces with $m \times n$ pixels forms a subspace, and this subspace is inside of the images space which contain all the images with $m \times n$ pixels. This subspace best defines the variation of the known faces. This subspace which we can call face spaces is defined by singular vectors of the set of known faces. These singular vectors do not necessarily correspond to the features like ears, eyes and noses. The projection of a new image onto this face space is then compared to the available projections of the known faces to identify the person. Since the dimension of face subspace is much less than the whole image space, it is much easier to compare projections than origin images pixel by pixel.

Our dataset consists of 21 images that are checked by algorithm and according to their SVD values the image with the smallest value will be printed, which means that for the image there is a match with the testing images.

```
# Importing libraries
import cv2
import numpy as np
import os

# In[2]:

# Giving path for training images
dataset_path = './images_dataset/'
dataset_files = os.listdir(dataset_path)
images = []

# Reading all the images and storing into an array
for person_name in dataset_files:
    temp = cv2.imread(dataset_path + person_name)
    temp = cv2.cvtColor(temp, cv2.COLOR_BGR2GRAY)
    temp = cv2.resize(temp, (100, 100), interpolation=cv2.INTER_AREA)
    images.append(temp.flatten())

# Substring mean from all images for normalization
images = np.array(images)
mu = np.mean(images)
images = images - mu
images = images.T
# print(images.shape)
# In[3]:
# SVD function
u, s, v = np.linalg.svd(images, full_matrices=False)
# print(u.shape, s.shape, v.shape)
```

```

# In[4]:

# Reading test image as an input, converting into 100*100
test_dataset =
np.array(cv2.imread("D:\\DEV\\FaceRecognitionSVD\\test_images\\kevin_1
.jpg"))
test_dataset = cv2.cvtColor(test_dataset, cv2.COLOR_BGR2GRAY)
test_dataset = cv2.resize(test_dataset, (100, 100),
interpolation=cv2.INTER_AREA)
img = test_dataset.reshape(1, -1)

# Subtracting mean
img = img - mu
img = img.T
# print(img[:,50])

# In[5]:

# Dot product of test image and U matrix
test_x = np.empty(shape=(u.shape[0], u.shape[1]), dtype=np.int8)
# print(test_x.shape)

for col in range(u.shape[1]):
    test_x[:, col] = img[:, 0] * u[:, col]

dot_test = np.array(test_x, dtype='int8').flatten()

# In[6]:

# Dot product of all the images and U matrix
dot_train = np.empty(shape=(u.shape[0] * u.shape[1], u.shape[1]),
dtype=np.int8)
temp = np.empty(shape=(u.shape[0], u.shape[1]), dtype=np.int8)

for i in range(images.shape[1]):

    for c in range(u.shape[1]):
        temp[:, c] = images[:, i] * u[:, c]

    tempF = np.array(temp, dtype='int8').flatten()
    dot_train[:, i] = tempF[:]

# In[7]:

# Subtracting Two dot products
sub = np.empty(shape=(u.shape[0] * u.shape[1], u.shape[1]))

for col in range(u.shape[1]):
    sub[:, col] = dot_train[:, col] - dot_test[:]

# In[8]:

# Finding norm of all the columns
output_array = np.empty(shape=(u.shape[1],))

for c in range(sub.shape[1]):
    output_array[c] = np.linalg.norm(sub[:, c])

```

```

# In[9]:
print(output_array)

# In[10]:

# Sorting answer array and retrieving first element which will be
minimum from all
# the smallest element is the matching one
temp_ans = np.empty(shape=(u.shape[1],))
temp = np.copy(output_array)

temp.sort()
check = temp[0]
print(check)

# In[11]:
index = 0

for i in range(output_array.shape[0]):
    if check == output_array[i]:
        index = i
        break

# In[12]:

# Checking for corresponding image for minimum answer
folder_tr = '/images_dataset/'
i = 0
for filename in os.listdir(os.getcwd() + "/" + folder_tr):

    if index == i:
        # If there is a match then it will directly print the image
        print("Printing the predicted face : ", filename)
        img_path = os.getcwd() + folder_tr + filename
        from PIL import Image
        image = Image.open(img_path)
        image.show()
        break
    else:
        i = i + 1

```

The upper code describes a face detection process using SDV algorithm.

CHAPTER 4

CONCLUSIONS

4.1 Conclusions

This project presents a more direct way of dealing with face recognition as it makes simple the authentication process.

The face detection phase is entirely non-intrusive and can be implemented without any issues or need for training or instruction (do note the Impact point about privacy however). Decently portable code makes this implementation likely to be useable in many contexts. Reducing time wasting during conventional class attendances. Automating the whole process so that we have digital environments Preventing fake roll calls, so one to one attendance marking is possible only.

Utilizing latest, trends in machine vision to implement a feasible solution for class attendance system. This project presents a more direct way of dealing with face recognition. The face detection phase is entirely non-intrusive and can be implemented without any issues or need for training or instruction (do note the Impact point about privacy however).

Decently portable code makes this implementation likely to be useable in many contexts.

The above article concludes that the Attendance system was made with combination of many different algorithms and is works with good accuracy and efficiency. It should be noted that plenty of other algorithms such as Haar cascade were also explored which can be replaced by HOG but it was found that the HOG method is more efficient while having a little number of faces which is generally with the case of biometric and Attendance systems. One could even improve the performance of the algorithm by employing CUDA GPU cores and more powerful CPU which would be available in a server system rather compared to a weak Personal Computer.

However, the design logic along with key points of different algorithms were discussed in the article.

REFERENCES

- [1] T.Kanade, Computer Recognition of Human Faces,, Birkhauser,Basel, 1977.
- [2] Moghaddam,B. And Pentland A., Probabilistic visual learning for object representation., 1997.
- [3] Rowley,H. A.Baluja,S. and Kanade,T., “Neural network-based face detection.,” 1998, pp. 23-38.
- [4] Sung,K. and Poggio,T., “Example-based learning for view-based human face detection,” 1998, pp. 39-51.
- [5] L. Li, X. Mu, S. Li and H. Peng, “A Review of Face Recognition,” in *IEEE Access*, Stoke-on-Trent, UK, 2020.
- [6] A. Silsanpisut, P. Petchsamutr and M. Ketcham, Chapter 14 Anti-theft Motorcycle System Using Face Recognition by Deep Learning Under Concept on Internet of Things, Springer Science and Business Media LLC, 2019.
- [7] L. Wandzik, G. Kaeding and R. Vicente , “Morphing Detection Using a General Purpose Face Recognition System,” in *26th European Signal Processing Conference (EUSIPCO)*, 2018.
- [8] H. Yang and X. Han, “Face Recognition Attendance System Based on Real-Time Video Processing,” in *IEEE Access*, 10 July 2020.
- [9] A. Tolba, A. El-Baz and A. A. El-Harby, “Face Recognition: A Literature Review,” ResearchGate, [Online]. Available: https://www.researchgate.net/publication/233864740_Face_Recognition_A_Literature_Review.
- [10] A. Rosebrock, “Face recognition with OpenCV, Python, and deep learning,” <https://pyimagesearch.com/>, 18 June 2018. [Online]. Available: <https://pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/>.
- [11] “Face Detection Algorithms & Techniques,” Facedetection.com, [Online]. Available: <https://facedetection.com/algorithms/>. [Accessed 1 6 2022].
- [12] A. Geitgey, “Face Recognition,” [Online]. Available: https://github.com/ageitgey/face_recognition. [Accessed 29 5 2022].
- [13] A. Geitgey, “Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks,” Medium.com, [Online]. Available: <https://medium.com/@ageitgey/machine->

learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721.

[Accessed 5 26 2020].

- [14] M. Vincze, S. Wachsmuth and G. Sagerer, *The Cambridge Handbook of Artificial Intelligence*, 05 July 2014.
- [15] C. M. Bishop, "Pattern Recognition and Machine Learning," New York, NY, Springer, 2006.
- [16] C. Lampert, "Kernel methods in computer vision. Foundations and Trends in Computer Graphics and Computer Vision," 2008, pp. 193-285.
- [17] Heisele, B., Serre, T. and Poggio T, "A component-based framework for face detection and identification. Internal Journal of Computer Vision," 2007, pp. 167-181.
- [18] O'Toole, A. J., Jiang, F., Roark, T., "Prediction human face recognition," in *Face Processing: Advanced Methods and Models*, 2007.
- [19] Vetter, T. and Poggio, "Linear object classes and image synthesis from a single example image.," pp. 733-742.
- [20] Rowland, D. A. and Perrett, "Manipulating facial appearance through shape and color.," 1995, pp. 70-76.
- [21] Matthews, I. and Baker, S., "Active appearance models revisited. International Journal of Computer Vision.," 2004, pp. 135-164.
- [22] Ramnath, K., Koterba, S., Xiao, J., Hu, C., Matthews, I., Baker, S., Cohn, J., and Kanade, T., "Multi-view AAM fitting and construction. Internal Journal of Computer Vision.," 2008, pp. 183-204.
- [23] Sivic, J., Zitnick, C. L., and Szeliski, R., "Finding people in repeated shots of the same scene.," Edinburgh, 2006, pp. 909-918.
- [24] Lin, D., Kapoor, A., Hua, G., and Baker, S., "Joint people, event and location recognition in personal photo collections using cross-domain context.," Heraklion, Crete., 2010.
- [25] A. Silsanpisut, P. Petchsamutr and M. Ketcham, "System Using Face Recognition by Deep Learning Under Concept on Internet of Things.," 2019.
- [26] L. Wandzik, G. Kaeding and R. Vicente, "Morphing Detection Using a General Purpose Face Recognition System," in *European Signal Processing Conference, EUSIPCO*, 2018.