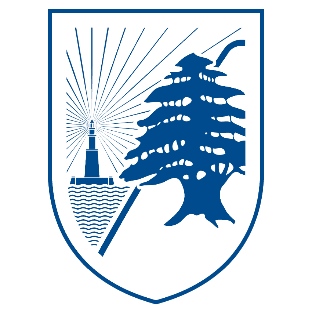
C M P S 3 2 7 P R O J E C T



F A C E A T T E N D A N C E S O F T W A R E



**Presented by**

Mohamed-Hammoud Ahmed Chokor

202101712

**Proposed to**

Dr. Lama Affara

Table of Contents

[Abstract 3](#_Toc88688405)

[Introduction 4](#_Toc88688406)

[Topic Motivation 4](#_Toc88688407)

[General Ideas about the topic 4](#_Toc88688408)

[Ideas related to the project and what was done in the project 4](#_Toc88688409)

[Work/Report Division 4](#_Toc88688410)

[Related Work and Technologies 5](#_Toc88688411)

[Teck Stack used 5](#_Toc88688412)

[How these programs were used 5](#_Toc88688413)

[State of the art techniques 5](#_Toc88688414)

[How Face Recognition Works 6](#_Toc88688415)

[Step 1: Finding the Faces 6](#_Toc88688416)

[Step 2: Posing and Projecting Faces 8](#_Toc88688417)

[Step 3: Encoding Faces 9](#_Toc88688418)

[Step 4: Finding the person’s name from the encoding 9](#_Toc88688419)

[Implementation 10](#_Toc88688420)

[Project Structure 10](#_Toc88688421)

[1. Deciding the Project idea 10](#_Toc88688422)

[2. Checking the web for similar projects 10](#_Toc88688423)

[3. Choosing the tools needed 10](#_Toc88688424)

[4. List of all the features 10](#_Toc88688425)

[How I implemented the Project 10](#_Toc88688426)

[Importing the needed Libraries 10](#_Toc88688427)

[Explanation of every Code section 11](#_Toc88688428)

[Results 15](#_Toc88688429)

[Results and Techniques 15](#_Toc88688430)

[Visual Results 15](#_Toc88688431)

[Console view after done encoding 15](#_Toc88688432)

[Webcam View 15](#_Toc88688433)

[Attendance.csv file view 16](#_Toc88688434)

[Email View 16](#_Toc88688435)

[Conclusion 17](#_Toc88688436)

[Reference 18](#_Toc88688437)

# Abstract

The Project is about a Face Attendance Software, which is developed using many kinds of image processing techniques, like Histogram of Gradients HOG, face landmark estimation, and other. This software is written using Python Programming Language.

# Introduction

## Topic Motivation

The passion for learning more about how face recognition works, combined with the passion for automating mundane everyday tasks that take more time than they should, resulted in the concept of Face Attendance Software, which takes the task of taking attendance of every student from the teacher by checking that he is present and gives it to a computer that works more efficiently in this task.

## General Ideas about the topic

A Face Attendance Software is a piece of software that allows university or school instructors to take attendance in their classes using a face recognition approach.

## Ideas related to the project and what was done in the project

In the project we accomplished the main goal that was talked about in the General ideas about the topic section right above, a face attendance software was made, and it marked the attendance of the students with the time that they entered the class in. The program also send the csv file to the instructor email after scanning the instructors face.

## Work/Report Division

Everything that was written as source code for this project, the document that you are currently reading, and the PowerPoint presentation was all prepared by the student Mohamed-Hammoud Ahmed Chokor, university I.D. 202101712.

# Related Work and Technologies

## Teck Stack used

* Git
* GitHub
* Visual Studio Code

## How these programs were used

Git & GitHub was used to store the code in an online repository and for version control, you can find the repository at the following link: <https://github.com/mmchokor/Face-Attendace-Program>

Visual Studio Code was used by the member to edit the code and run it.

## State of the art techniques

Alongside using Git & GitHub for version control and storing the code on an online repository, we also explained in comments properties of every function or snippet of code created in this project which aided us to understand every single piece of code that was written in the project and it can also aid anyone who isn’t a part of this project, and even if he has little to nothing of experience in programming he can still understand what every piece of code is meant to represent and do in the functioning of the software.

# How Face Recognition Works

There are 4 steps that are done in the process of face recognition, and they are the followings:

1. Finding all the Faces
2. Posing and Projecting Faces
3. Encoding Faces
4. Finding the person’s name from the encoding

## Step 1: Finding the Faces

Face detection is the initial stage in our workflow. Obviously, we must first detect the faces in a snapshot before attempting to distinguish them. Face detection is an excellent camera function. When the camera can detect faces automatically, it can ensure that all the faces are in focus before taking the photo. But we'll use it for something else: we'll use it to discover the regions of the picture that we want to send on to the next phase in our pipeline.

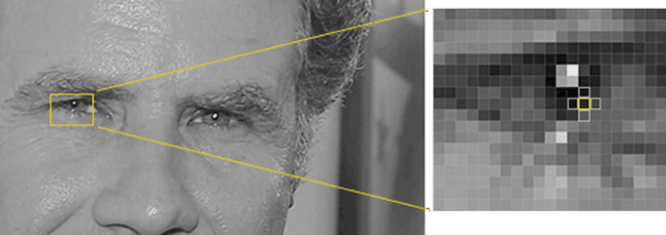
We'll employ a technique called Histogram of Oriented Gradients, which was developed in 2005.

To locate faces in a picture, we'll first convert it to black and white because we don't require color data to do so:

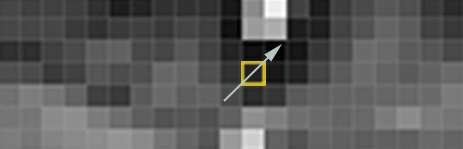
 A person with curly hair

Description automatically generated with medium confidence

Then each pixel in our image will be examined one by one. We want to look at the pixels that are directly surrounding each pixel:



The goal is to determine how dark the current pixel is in comparison to the pixels immediately surrounding it. Then we'll draw an arrow to show which direction the image is becoming darker.



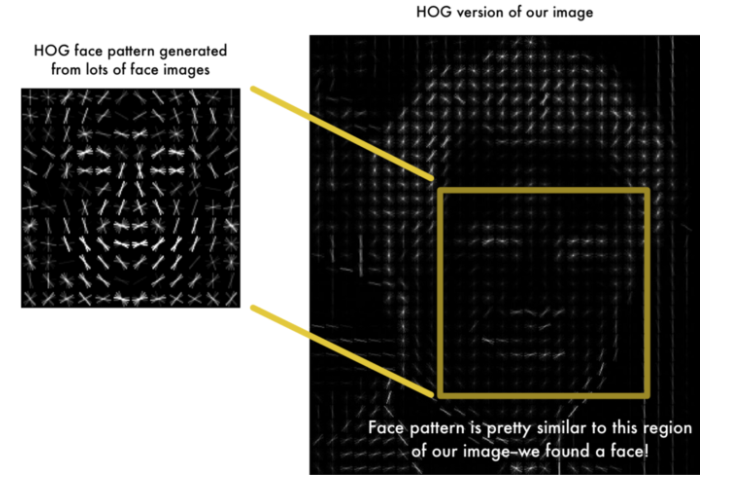
If we evaluate pixels directly, we will find that incredibly dark and really bright photographs of the same individual have completely distinct pixel values. However, if just the direction of brightness change is considered, both extremely dark and extremely brilliant pictures will end up with the same precise representation.

To do this, we'll divide the image into 16x16 pixel squares. We'll count how many gradients point in each main direction in each square. Then we'll replace that square in the picture with the strongest arrow directions.

Consequently, we convert the original image into a very simple representation that represents the core structure of a face in a straightforward manner.

All we must do to locate faces in this HOG image is find the area of our image that appears the most similar to a known HOG pattern taken from a number of previous training faces:

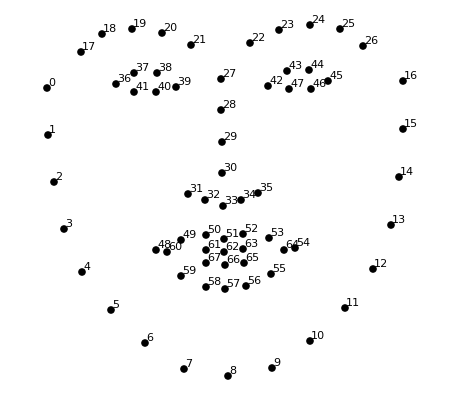
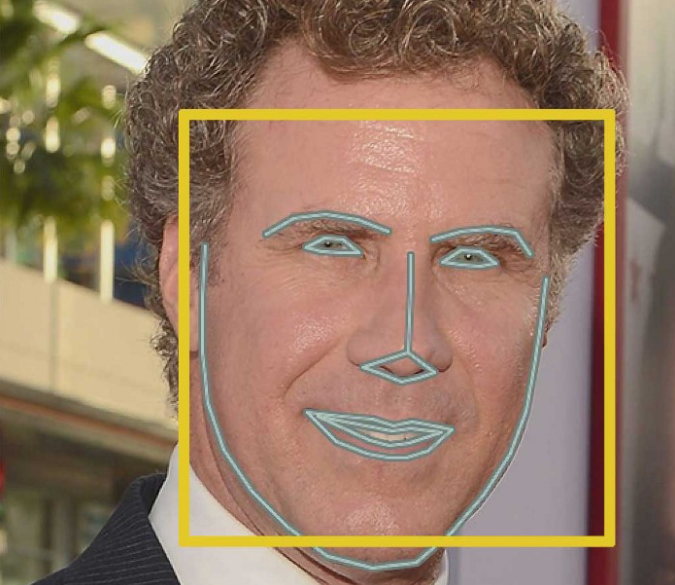


## Step 2: Posing and Projecting Faces

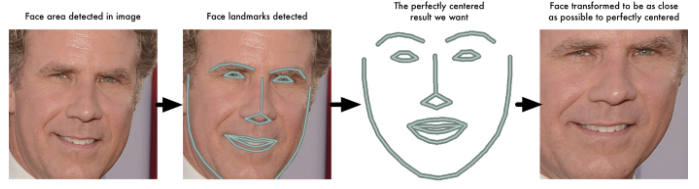
Now we must deal with the issue that faces oriented in various directions appear completely different to a computer. To accommodate for this, we will distort each image such that the eyes and mouth are always in the sample location. This will make comparing faces much easier in the next phases.

To do this, we will employ a technique known as facial landmark estimation. There are several approaches to this. However, we will employ the technique developed in 2014 by Vahid Kazemi and Josephine Sullivan.

The main concept is that we will generate 68 distinct points that exist on each face. Then, we'll train a machine learning system to discover these 68 precise spots on any face.

Now that we know where the eyes and mouth are, we'll just rotate, scale, and shear the picture to center the eyes and mouth as much as feasible. We will not utilize any sophisticated 3D warps since they will create distortions into the image. We will only employ fundamental picture changes such as rotation and scaling that retain parallel lines, which are known as affine transformations. Now, no matter how the face is oriented, we can center the eyes and mouth in the picture in nearly the same location. This will make our following step much more precise.



## Step 3: Encoding Faces

The simplest approach to face recognition is to directly compare the unknown face we found in Step 2 with all the pictures we have of people that have already been tagged. When we find a previously tagged face that looks very similar to our unknown face, it must be the same person. But the computer isn’t capable to distinguish two faces from looking at their eye color, but it needs measurement to do so. The best measurement for this task isn’t decided by human but by a neural network that generated 128 measurements for each face.

Training a convolutional neural network to generate facial embeddings necessitates a large amount of data and computational resources. Even with a high-end NVidia Telsa video card, it takes around 24 hours of continuous training to get decent accuracy.

However, once taught, the network can create measures for any face, even ones it has never seen before! As a result, this step only must be completed once. Fortunately for us, the good people at OpenFace have already done this and provided multiple trained networks that we can easily utilize.

So, all we have to do is feed our face photos into their pre-trained network to retrieve the 128 measures for each face.

Note that we don’t know what part of the face the 128 generated belongs to, but all that we care about that network generates nearly the same numbers when looking at two different pictures of the same person. A picture containing shape

Description automatically generated

## Step 4: Finding the person’s name from the encoding

This final step is the simplest of the entire procedure. All we must do is discover the individual with the closest dimensions to our test image in our database of known persons. You may accomplish this by employing any fundamental machine learning classification technique. All that remains is to train a classifier that can take measures from a fresh test image and determine which known individual is the closest match. It takes milliseconds to run this classifier. The classifier's output is the person's name.

# Implementation

## Project Structure

### Deciding the Project idea

After reviewing all of the suggestions offered by our lecturer, Dr. Lama Affara, as well as additional ideas obtained on Google, some brainstorming was conducted, and the idea of a face attendance program was chosen.

### Checking the web for similar projects

After deciding on a concept for the project, I searched the web for projects that were similar to mine in order to get more ideas to add to my and to see if I was capable of completing this project, because haven’t yet taken any complex idea like this in the course. I discovered a few projects and was inspired by many that helped to make my own unique program.

### Choosing the tools needed

Because we were taking the Python course, the programming language was Python, and I used my favorite text editor VS Code, as well as Git & GitHub for version control and online code storage.

### List of all the features

* Recognizing the face of the student from a database that contain all the student faces.
* Marking down the attendance for the student after recognizing his face.
* Marking down the exact time that the student entered the classroom.
* Saving the previous two points in a csv file.
* Sending this csv file to the instructor email address after recognizing the instructor face.
* The program is easy to use and don’t need programming skills to set it up.

## How I implemented the Project

### Importing the needed Libraries

* Numpy
* Open CV
* face\_recognition
* smtplib
* time

### Explanation of every Code section

#### Loading the Images

After I finished loading all of the required libraries, I began coding.

First, I needed to import all of the pictures of student faces from the database, which are located in the ImagesAttendance folder in the program's directory.

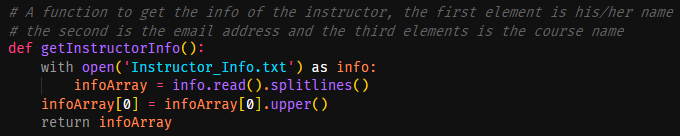
All of this is accomplished using the following line of code, which makes use of the open cv library's function to load the pictures.

Text

Description automatically generated

#### Loading the instructor’s info

The “getInstructorInfo” function was called to do this task. In it, the file “Instructor\_Info.txt” was opened and read. All the info found in it was stored in an array and was returned.



#### Encoding the loaded images

To encode the images, we called the function “findEncodings”, and parsed in the images that we loaded. In the function, each image was converted into RGB form using open cv library and used the face recognition library function “face\_encodings”, after encoding the images, the data was appended to an array and this array was returned when all the images are done encoding.

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

#### Launching the webcam and recognizing the faces scanned

To open the webcam we used the “VideaCapture” function in open cv, and the same procedure done in the section (Encoding the loaded images) was done, but this time with the picture input coming from the cam.

Text

Description automatically generated

#### Comparing the scanned face with the database

This part is done by comparing the scanned face with the faces found in the database and finding the distance between them, and the finding the minimum to see if it matched with any face.

Text

Description automatically generated

#### Labeling the faces on the cam and marking attendance

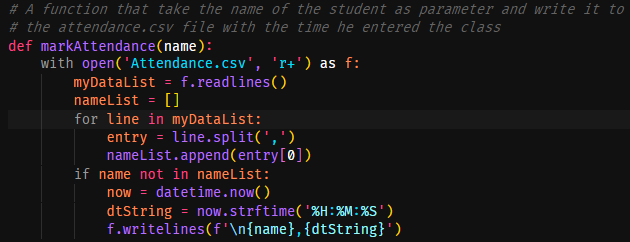
If faces matched it will be stored in the variable name and using the open cv library rectangle will be drawn around the face and a label will be placed at the bottom of the rectangle with name displayed if matched or displaying “Unknow” if the face didn’t match with any face from the database.

Text, chat or text message

Description automatically generated

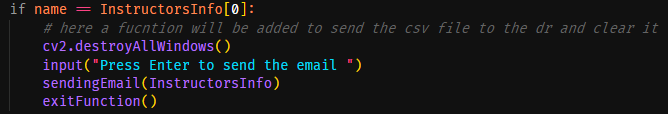
Also, if the face matched with a face from the database the “markAttendane” function is called, parsing in the name in it.

In this function, the Attendance.csv file will be opened, and the name will be written to it with the time of the student entering the class. Also, the name will be stored in an array to prohibit the duplication of the same student of student in the file if his face got scanned more than one time.



#### In the case of the instructor face is scanned

Inside the if statement (Section: Labeling the faces on the cam and marking attendance, picture one), there is an if statement that if the name of the scanned face is the same name of the instructor’s name which was obtained previously and stored in an array with the other info’s. The webcam will be closed, and the sending email function will be called. In this function we will use the smtplib library to write the code needed to send the Attendance.csv file to instructor’s email address which was parsed into the function in the array that contains all the instructor’s information. After the email is sent the “exit Function” is called and the program will close itself.



Text

Description automatically generated

Text

Description automatically generated

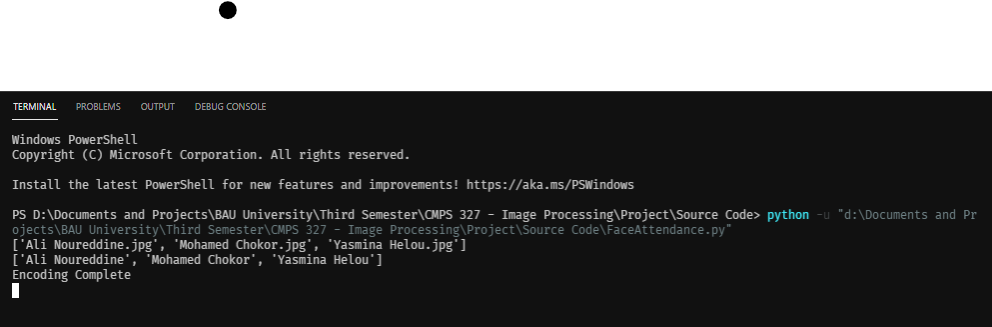
# Results

## Results and Techniques

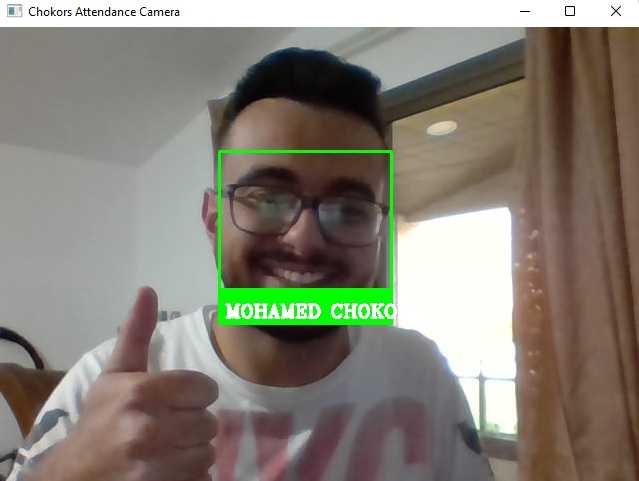
First every part of the program was tested alone, from the face recognition, marking attendance and email sending features. After passing all the test, the program was ran for the first time and worked almost perfectly but it needed some light tweaking and some little bug fixes here and there. After that is done, the program ran perfectly.

## Visual Results

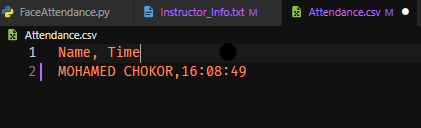
### Console view after done encoding



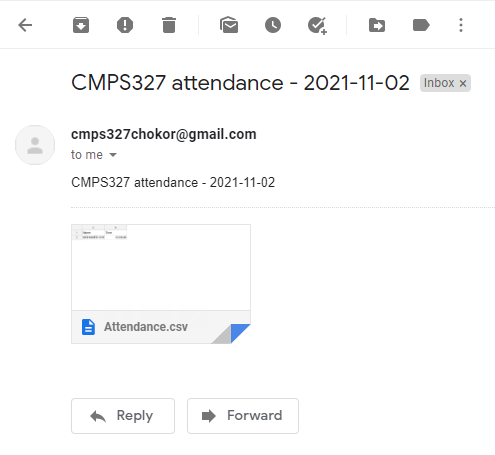
### Webcam View



### Attendance.csv file view



### Email View



# Conclusion

In the end, I can say that the project was a success, and that it helped me develop the skills that I learned from this course, skills that will help me in my life in general, but especially in my professional life when I work for companies and with other programmers, because it taught me the core of using Image Processing by developing a piece of software to serve a certain purpose.

Although I chose to undertake the project alone, it demonstrated to me that teamwork would have made it less stressful and more pleasurable if done with the proper individuals as partners.

Finally, it demonstrated that having a wonderful instructor can make a significant difference when producing this type of project, since they teach us the core of the material that we would need to utilize, and we would not have been able to accomplish what we did without them teaching us these ideas brilliantly.

# Reference

* Dr. Lama Affara CMPS327 course.
* <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78>
* <https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients>
* <https://www.youtube.com/watch?v=sz25xxF_AVE>