University of St.Gallen

Handbook Coding Project:
‘Real Time Face Recognition’

Tania Regueiro
14-126-973

Nadia Regueiro
14-112-379

Anina Patricia Sax
14-109-177

FS19-8,789,1.00 Skills: Programming with Advanced Computer Languages

Silic Mario, PhD

Thursday, May 9, 2019

Table of Contents

1. Introduction 1

2. Software, Programs and Packages needed 2

2.1. How to use the program 2

2.2. Install Python 3.7 and Anaconda 3

2.3. Install Python packages 3

2.4. Basics/Principles of the Database 5

3. Visualization of the project and common issues 5

3.1. Visualization 5

3.2. Common Issues - Face Recognition Accuracy Problems 7

3.2.1 Similar faces 7

3.2.2 Incorrect name association with unknown people 8

3.3 Solution to common issues 8

3.3.1 Adjusting Tolerance/Sensitivity 8

3.3.2 Upload more pictures 9

4. Sources 10

List of Figure

Figure 1 - Face Recognition Process 1
Figure 2 - Required Software and Packages 2
Figure 3 - Github File 2
Figure 4 - Database and Subfolders 2
Figure 5 - Run the Code 3
Figure 6 - Terminal Application 4
Figure 7 - Visualization 6
Figure 8 - Real time face recognition 6
Figure 9 - Face Recognition 'Unknown' 7
Figure 10 - Similar Faces 7
Figure 11 - incorrect Association 8

1. Introduction

The aim of the project ‘Real Time Face Recognition’ is to uniquely identify a person by comparing and analyzing patterns based on the person's facial contours. The information needed for the recognition is stored in a database. The program created is a basic version for real time face recognition. Facial recognition technology is constantly being developed further. It has received significant attention as it has the potential for a wide range of applications related to law enforcement as well as other projects.

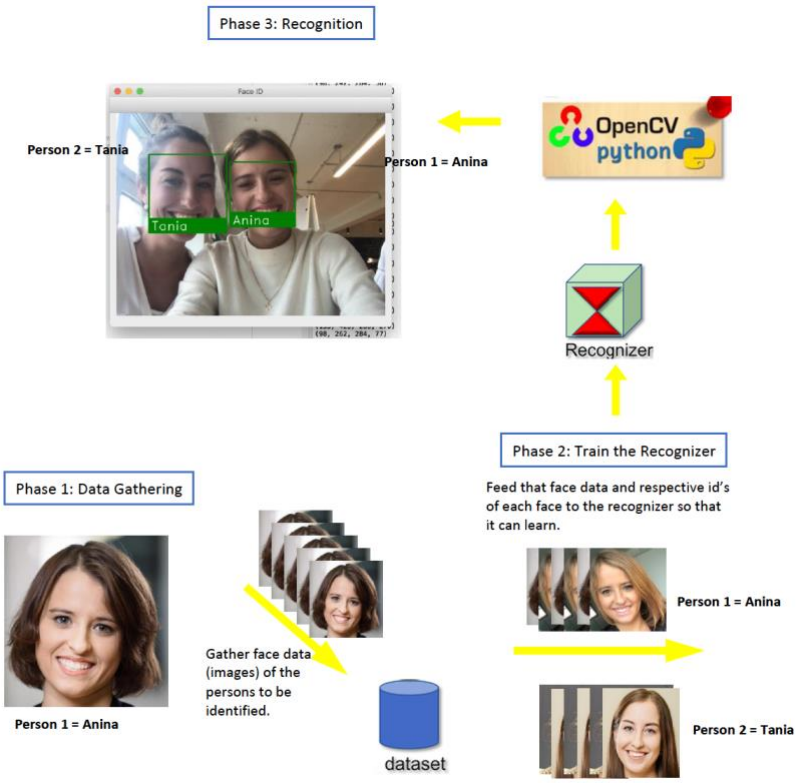

Figure 1 - Face Recognition Process

This handbook is divided into three parts: first it explains which software is needed to run the program (see chapter 2), second it will give information on how to run the program (see chapter 3), finally the project is shown visually using an example (see chapter 4).

2. Software, Programs and Packages needed

To run the program two software need to be installed: Python 3.7, Anaconda and the Python necessary packages. The next section describes how to use the program in more detail.

2.1. How to use the program

The program can be run by following these four steps:

1. Install Python 3.7, Anaconda and the Python necessary packages (see paragraph 2.2 -2.3). Spyder, an environment available in Anaconda, will be later used to run the code.

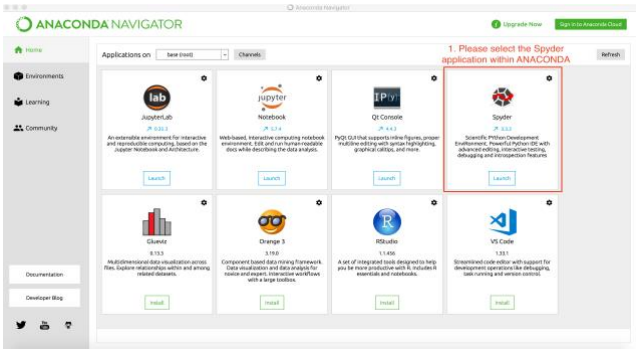

Figure 2 - Required Software and Packages

2. Download the file from Github (myFaceIdentification.py) and save the file on your device.

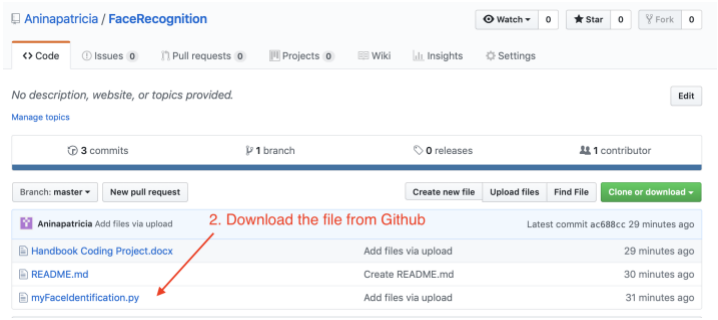

Figure 3 - Github File

3. Create a folder 'Database' and store it in the same place as the previous file. Further, create subfolders including pictures of the people you want to identify and label them with the names of the corresponding person

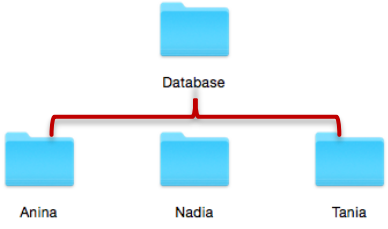

Figure 4 - Database and Subfolders

4. Open the previously downloaded file including the code in Spyder and run the program.

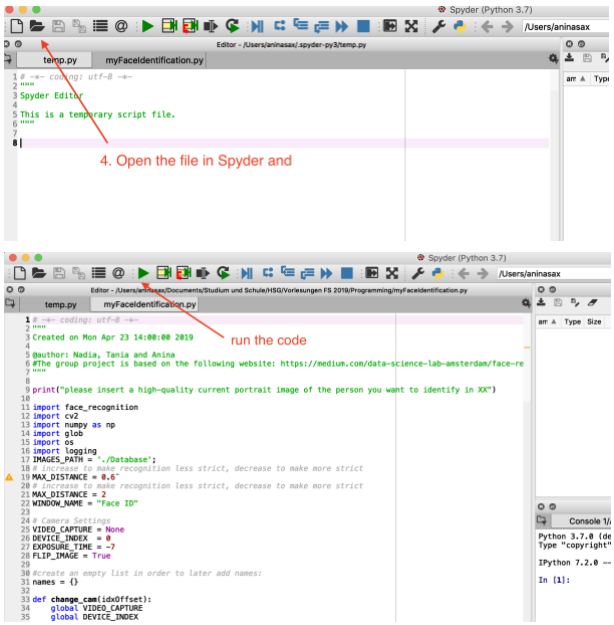

Figure 5 - Run the Code

2.2. Install Python 3.7 and Anaconda

In order to run the program the following software needs to be installed: Python 3.7 and the Anaconda package, which will be used to access Spyder[1]. The links for downloading the two softwares are the following:

https://www.python.org/downloads/

https://www.anaconda.com/download/

The installation of this software should be easy and straightforward as the installation works in a similar way to any other program.

2.3. Install Python packages

To use some specific functions, different Python packages need to be downloaded. The majority of the packages can simply be installed using the command pip install. However, a package (called 'dlib') is used which cannot be installed by simply using this command. It requires the Python environment cmake.

[1] Please note, that the following description applies for MacBook

2.3.1. Install Cmake

The procedure to install cmake works as follows:

1. Press Command and Space and type Terminal and press enter key.
2. Run the following code in the terminal application on your MacBook:

   ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)" < /dev/null 2> /dev/null

and press enter/return key.

In case you should enter a password, enter your Mac's user password to continue and press enter. While you type the password, it will not be displayed on the screen. However, the system will accept it. Wait for the command to be finished.

3. Run the command:

   brew install cmake

2.3.2. Installation of the packages

The procedure to install the various packages is the same for each package. As an example, here below are the different steps required to install the tensorflow package.

1. Open the Terminal Application on your MacBook

Figure 6 - Terminal Application

2. Paste 'pip install tensorflow' into the code (without the ' ') and press enter
3. Repeat the procedure for the other packages by using the following instructions:

   pip install PyHamcrest

   C:

   pip freeze > requirements.txt

   pip install -r requirements.txt –upgrade

```
# re-run, if error occurs

# conda install python=3.6.5

pip install numpy scipy matplotlib scikit-learn jupyter

pip install imutils

conda install -c menpo ffmpeg

pip install opencv-contrib-python

pip install dlib

pip install -I numpy

pip install tensorflow

pip install pillow pandas h5py keras

pip install face_recognition
```

2.4. Basics/Principles of the Database

In order to recognize different people, the program needs a dataset as an input. It gathers information from this data to identify and recognize the people visible in the camera. In our case the data will be taken from pictures uploaded in subfolders of the folder ‘Database’ (see figure 4). More precisely, the subfolders are labelled with the names of the corresponding people and contain several images of this specific person. The more images uploaded in the subfolder of a person the better the program can recognize him or her. The program goes through the data and identifies patterns in the pictures saved. The program converts a particular face into numbers. Based on this numerical representation of a face, the program can recognize and identify a variety of people.

3. Visualization of the project and common issues

3.1. Visualization

We tried the program by ourselves. Therefore, we saved the file (which includes the code) and the ‘Database’ folder in the same place on our MacBookPro. The folder

“Database” includes subfolders (labelled ‘Tania’, ‘Nadia’ and ‘Anina’) with photos of us three (See figure 4). Subsequently, we opened Spyder and run the code. The camera opened and displayed the following image:

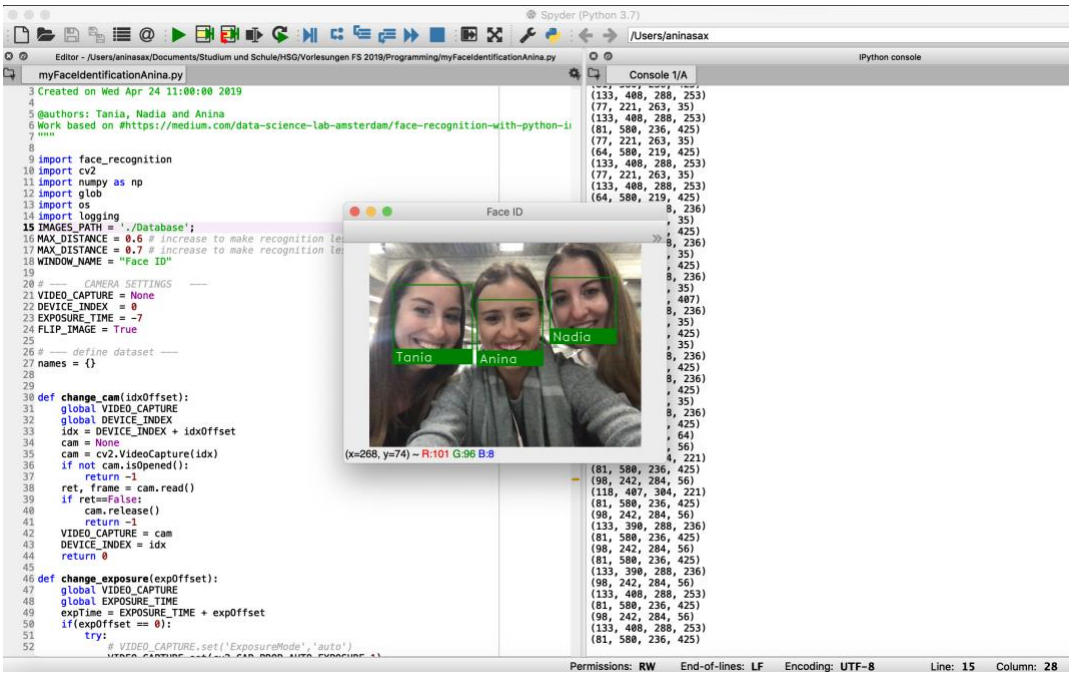

Figure 7 - Visualization

Afterwards, we experimented with different people, male and female. The program worked well if pictures of the corresponding people were added to the Database, regardless if they were male or female.

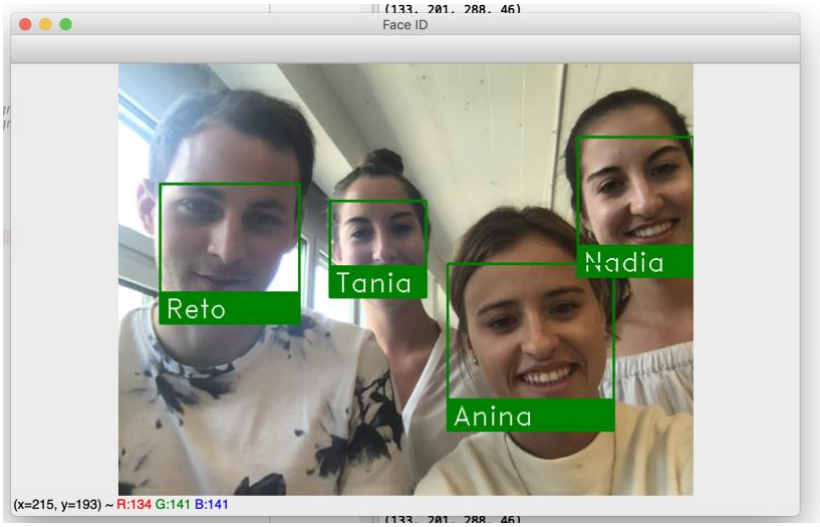

Figure 8 - Real time face recognition

Additionally, if the person is not added to the database the camera displays a rectangular around the face of the person labeled as ‘Unknown’. However, some difficulties were encountered (see section 3.2.).

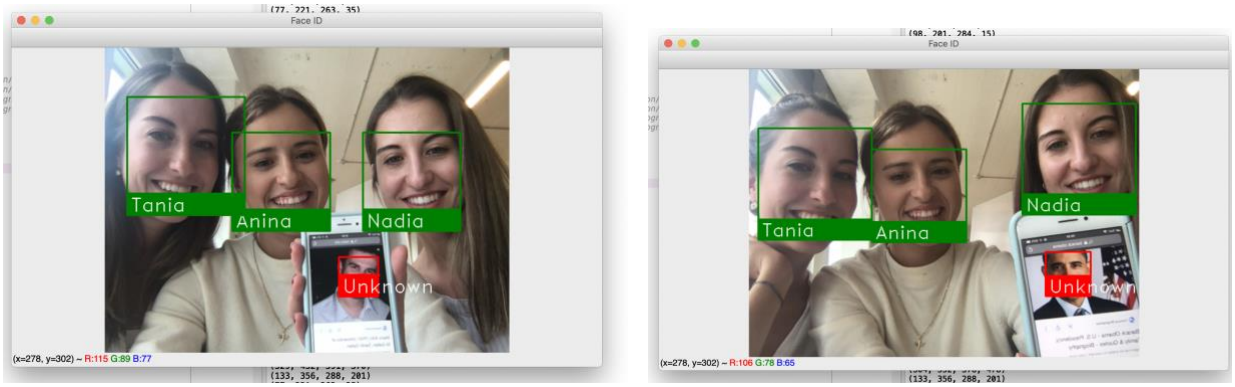

Figure 9 - Face Recognition 'Unknown'

3.2. Common Issues - Face Recognition Accuracy Problems

We ran the code several times and tried the real face recognition with different people. Generally, the recognition worked well but there were some issues relating to accuracy problems. The first issue relates to the identification of similar faces. The second issue relates to the association of people whose identity has not been saved in the database.

3.2.1 Similar faces

The first issue encountered is the correct recognition of people who look alike. We tested the program mainly with us three. It has no trouble identifying Anina (see figure 10). However, the program has trouble keeping Tania and Nadia apart as they look alike, although different pictures of each of them were stored in the folder in our database.

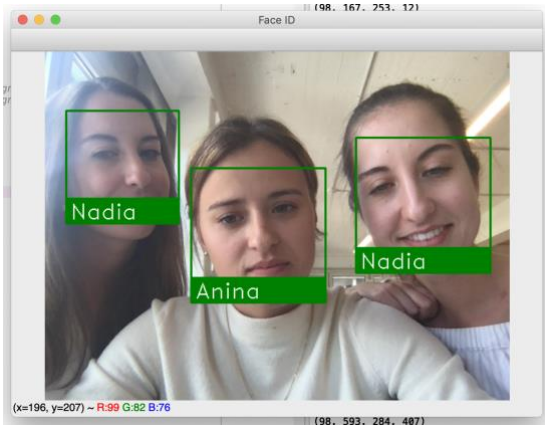

Figure 10 - Similar Faces

3.2.2 Incorrect name association with unknown people

The second issue encountered relates to the wrong association of unknown persons. Meaning the program has difficulties to identify people whose identity has not been stored in our database. It tries to find similarities to the pictures stored and associates the most similar person wrongly with the unknown person. In the example below (see figure 12) the man on the right side is recognized as ‘Reto’ although we did not upload pictures of him. The man on the left side is correctly identified as unknown. We only noticed this issue with the facial recognition of people of the same gender. This is the case as the resemblance between male and female individuals is lower.

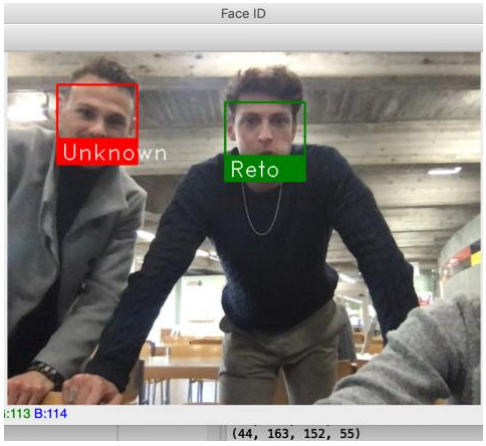

Figure 11 - incorrect Association

3.3 Solution to common issues

Possible solutions were discussed to prevent the common issues. In the end, two possible solutions were found to increase the accuracy of facial recognition. The first solution is to lower the adjusting tolerance to make the recognition stricter. The second solution is to upload more pictures to the dataset to train the recognizer better.

3.3.1 Adjusting Tolerance/Sensitivity

If the program has difficulties identifying the correct person, it might be that a lower tolerance value is needed to make the face comparisons stricter. The strictness of the comparison can be controlled with the tolerance parameter. The default tolerance value is 0.6. To make it more sensitive and consequently improve facial recognition, a lower value could be chosen.

3.3.2 Upload more pictures

As already mentioned in section 2.4 the more pictures uploaded fof a person, the better the program can identify this person. The recognizer must be trained to be able to recognize people. Therefore, the program can be improved by uploading more pictures, as the recognizer can improve the analysis of the patterns based on the person’s facial contours.

4. Sources

1. Study Materials from Coding@HSG
2. https://medium.com/data-science-lab-amsterdam/face-recognition-with-python-in-an-hour-or-two-d271324cbeb3