BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Emotion Kids

-ITSG Raport-

Team 8
Suciu Alexandra
Jurja Alexandru

Chapter 1

1. Introduction and motivation

1.1. Introduction

In preschool education, tracking Range of Emotion (ROE) is a standard approach to measuring progress in patient emotions. Often, ROE is measured subjectively and documentation is inconsistent between teachers. Teachers may come to wrong conclusions if ROE is tracked incorrectly between classes. The problem is that kids become sad and they want school hours to end as quickly as possible.

That's why we want to make a face recognition app that makes use of a phone / pc camera to objectively calculate real-time ROEs and automatically produces a report that tracks progress over the classes.

1.2. Motivation

Our motivation is to help kids to be more relaxed and happy when they use educational applications. We want to track pre-school emotions to improve educational applications to build new and healthy behaviors for kids.

This application is designed to track the pre-school emotions helping teachers to know which of educational apps are making kids happier or make them sad and help them for a long term. In this case we have a solution by creating a pc/mobile app which will be a useful tool for teachers help to reach their goals, like make kids happy and relaxed to have a good mood in the class by automated tracking of ROE.

Chapter 2

2. Scientific problem

2.1. Problem definition

Build an application for teachers for tracking kids emotions Via a camera, in real-time the kids will be monitored to track their face when they use the educational apps. This is very important to make a rank between the apps, which make the children happier or which not. Kids also are logged-in the applications based on their face with the helping from the recognition app. We also need to store the activity of the users, to have a sort of audit, this will help the owners to have a better image of what happening and how the user will use the educational apps, and how fast solve the problems, exercises.

Chapter 3

3. Related work

3.1 FDDB Dataset (Benchmark)

FileFacialRecognitionBenchmark project

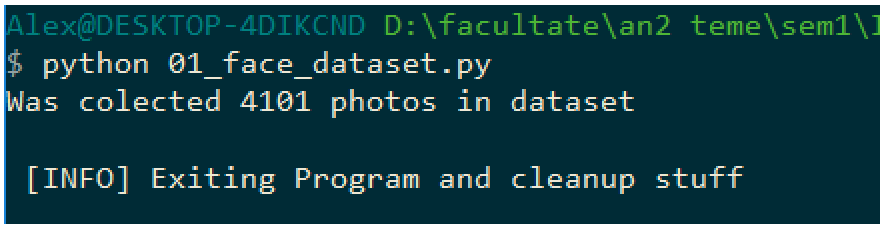
Face Detection Data Set and Benchmark (FDDB), a data set of face regions designed for studying the problem of unconstrained face detection. This data set contains the annotations for 5171 faces in a set of 2845 images taken from the Faces in the Wild data set. More details can be found in the technical report below.

This is the link to the database: http://tamaraberg.com/faceDataset/originalPics.tar.gz

Performance and comparisons

As an input we have the "users" folder which after running the dataset script will duplicate the pictures in black and white in the dataset folder .

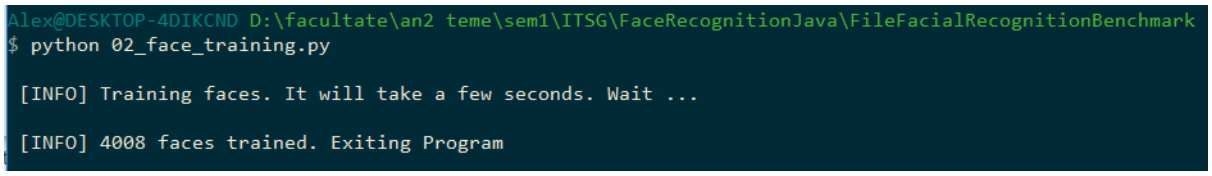
The program was able to detect the smile in about **7 minutes** of running a data set containing **4101** pictures, which can be found in the "dataset" folder



```
Alex@DESKTOP-4DIKCND D:\facultate\an2 teme\sem1\I
$ python 01_face_dataset.py
Was colected 4101 photos in dataset

 [INFO] Exiting Program and cleanup stuff
```

After the training was run, only 4008 faces were recognized



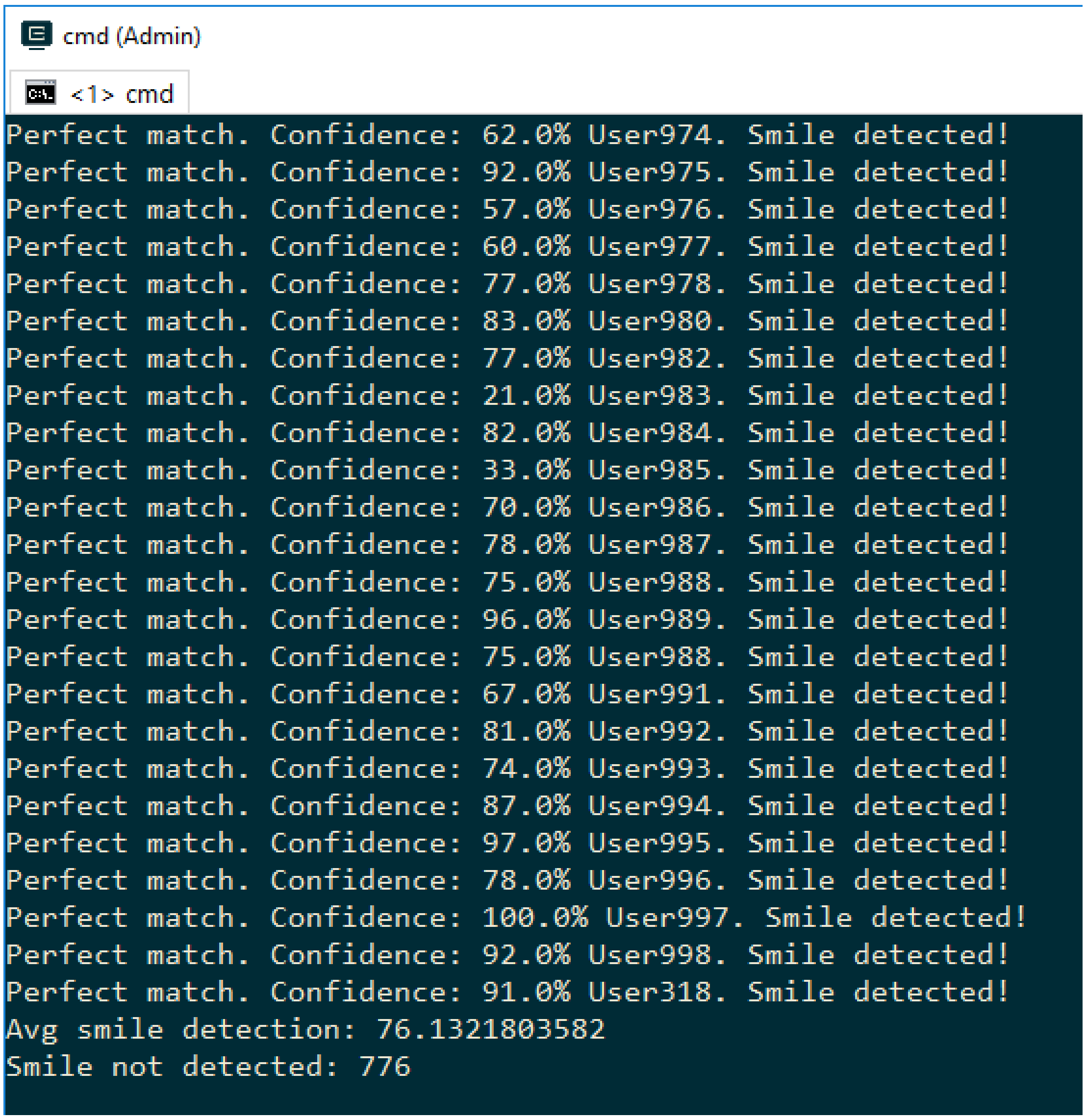
```
Alex@DESKTOP-4DIKCND D:\facultate\an2 teme\sem1\ITSG\FaceRecognitionJava\FileFacialRecognitionBenchmark
$ python 02_face_training.py

 [INFO] Training faces. It will take a few seconds. Wait ...

 [INFO] 4008 faces trained. Exiting Program
```

After a run of about 7 minutes, the program managed to identify a number of 3200 smiles, of which 700 faces contained another emotion. Some of the faces was not recognized.

The Average **Smile** detection was around 76.1 %



```
Perfect match. Confidence: 62.0% User974. Smile detected!
Perfect match. Confidence: 92.0% User975. Smile detected!
Perfect match. Confidence: 57.0% User976. Smile detected!
Perfect match. Confidence: 60.0% User977. Smile detected!
Perfect match. Confidence: 77.0% User978. Smile detected!
Perfect match. Confidence: 83.0% User980. Smile detected!
Perfect match. Confidence: 77.0% User982. Smile detected!
Perfect match. Confidence: 21.0% User983. Smile detected!
Perfect match. Confidence: 82.0% User984. Smile detected!
Perfect match. Confidence: 33.0% User985. Smile detected!
Perfect match. Confidence: 70.0% User986. Smile detected!
Perfect match. Confidence: 78.0% User987. Smile detected!
Perfect match. Confidence: 75.0% User988. Smile detected!
Perfect match. Confidence: 96.0% User989. Smile detected!
Perfect match. Confidence: 75.0% User988. Smile detected!
Perfect match. Confidence: 67.0% User991. Smile detected!
Perfect match. Confidence: 81.0% User992. Smile detected!
Perfect match. Confidence: 74.0% User993. Smile detected!
Perfect match. Confidence: 87.0% User994. Smile detected!
Perfect match. Confidence: 97.0% User995. Smile detected!
Perfect match. Confidence: 78.0% User996. Smile detected!
Perfect match. Confidence: 100.0% User997. Smile detected!
Perfect match. Confidence: 92.0% User998. Smile detected!
Perfect match. Confidence: 91.0% User318. Smile detected!
Avg smile detection: 76.1321803582
Smile not detected: 776
```

3.2 FileFacialRecognition

Performance and comparisons

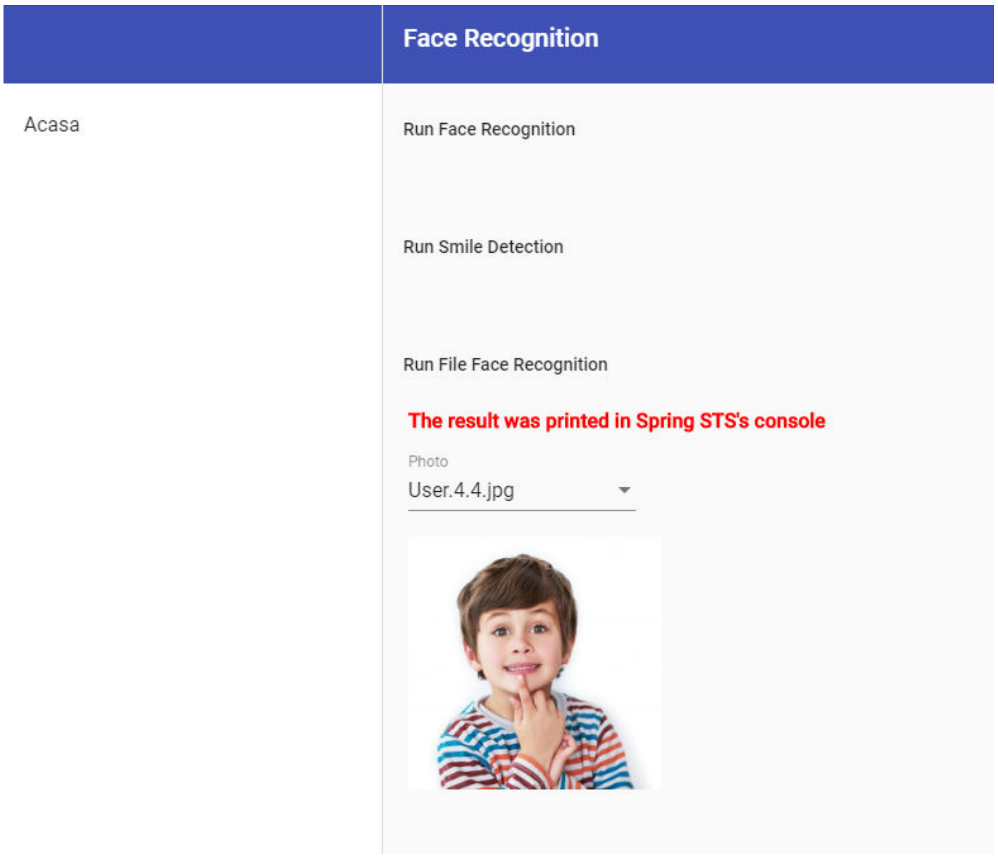
The same there is an input "users" folder which after running the dataset script will duplicate the pictures under a predefined naming in the dataset folder.

Training will detect the faces after we can run the face detection script which only will recognize the faces

Firstly you need to select a photo after press the Run File Face Recognition. A red message will show after the button after you can look in the Eclipse/Spring's console:

```
2020-01-17 17:51:11.100  INFO 7444 --- [nio-
Perfect match. Confidence: 99.0% User2
```

Example: for User 2 will be detected a 99% match.

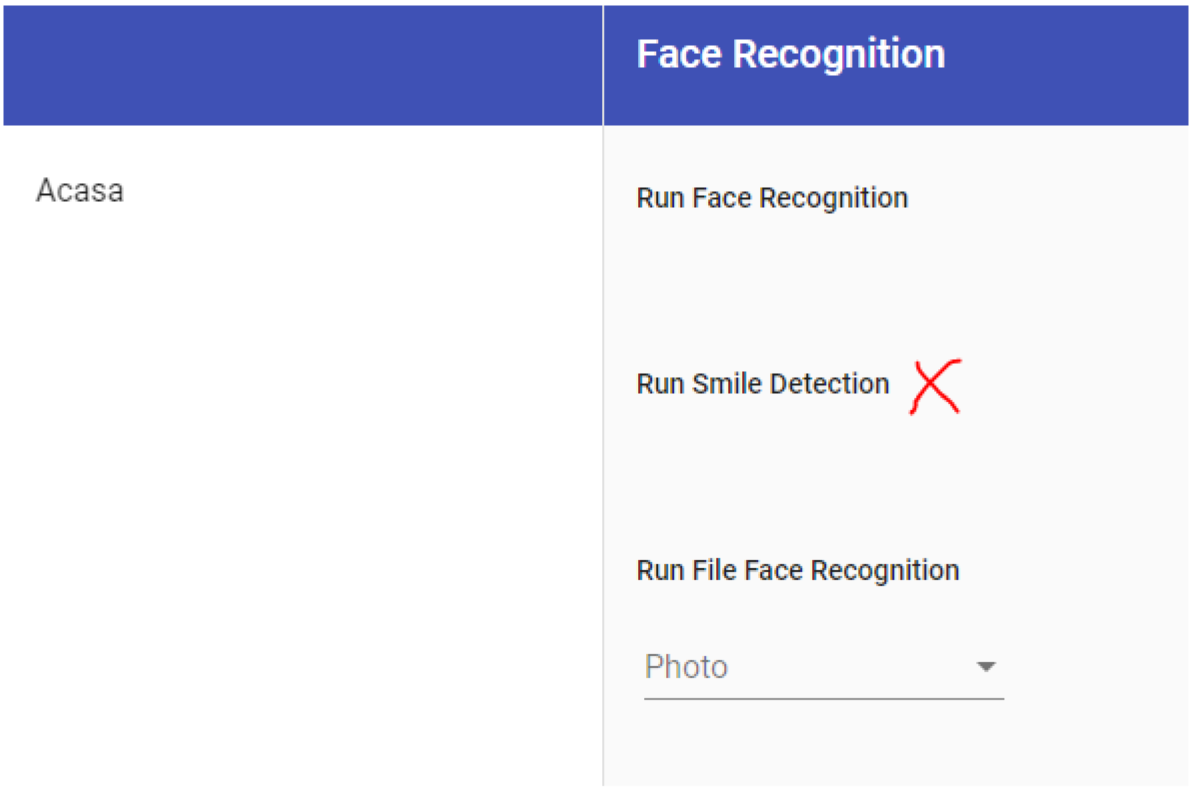


The result was printed in Spring STS's console

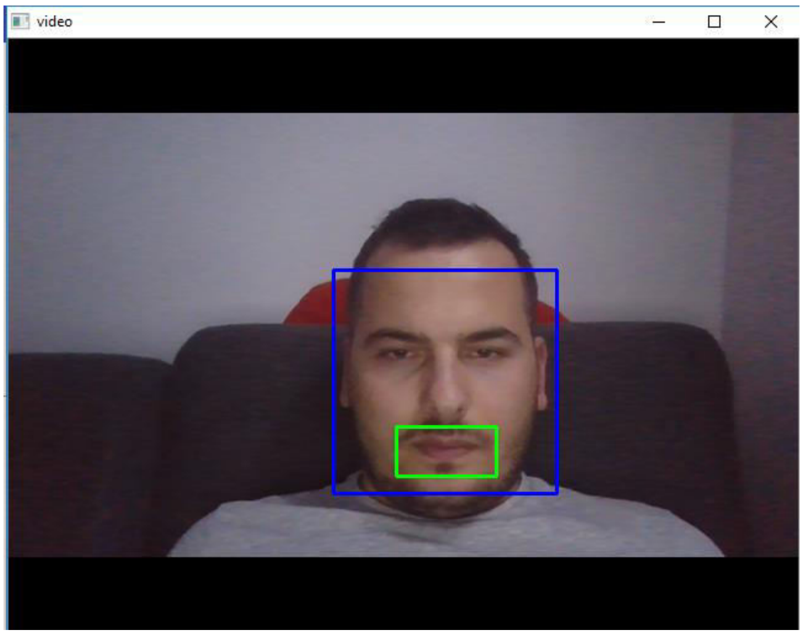
Photo: User.4.4.jpg

3.3 FaceDetection

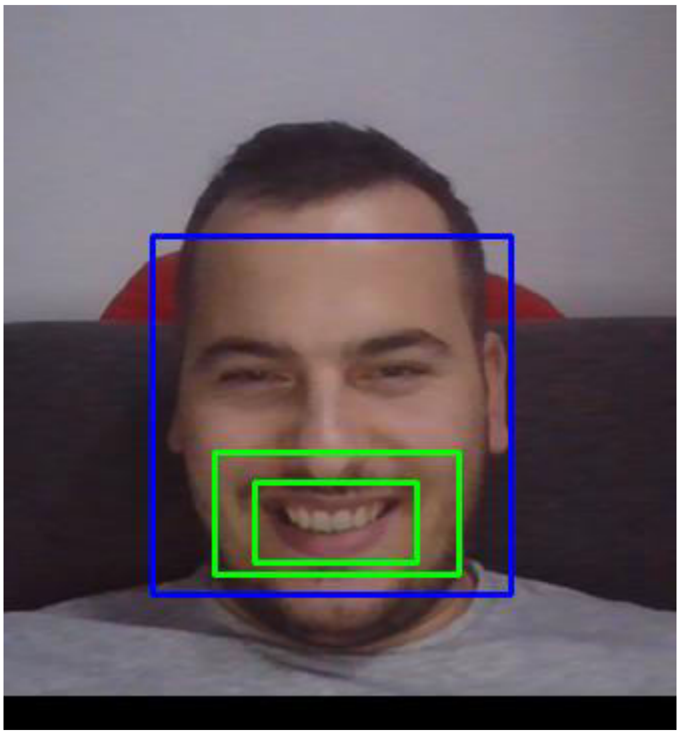
After the java app was started and the angular app also was started, you need to press the "Run Smile Detection" button from the UI.



After the button was pressed the camera will start and will detect your face. A bigger blue frame will cover your face and a smaller one will cover your mouth.

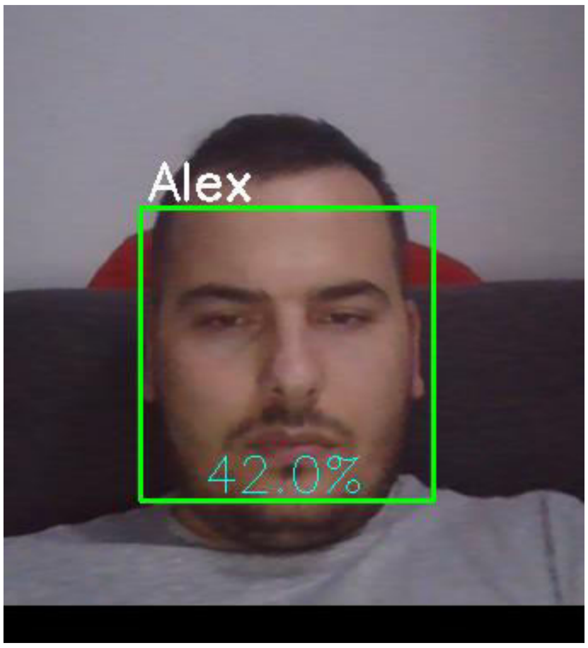


If you will smile a little bigger frame will enclose the mouth border.



3.3 FaceRecognition

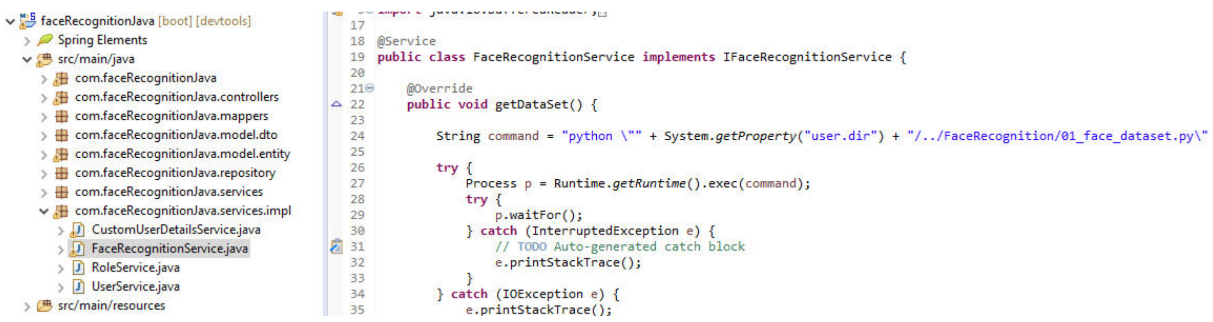
The first button "Perform face recognition" will run 3 scripts, first it will collect 30 photos with your face after executing the training script and start the webcam that will detect your face, it will write the name and display the number percent detection.



3.4 FaceRecognitionJava

This is the principal project from you can call all the functionalities. Is a Java project with an Angular 8 project as a UI project and it using an elasticsearch database to store de users.

The login use basic authentication (*as an improvement we want to use the webcam to authorize the users in the application*)



```
faceRecognitionJava [boot] [devtools]
  Spring Elements
  src/main/java
    com.faceRecognitionJava
    com.faceRecognitionJava.controllers
    com.faceRecognitionJava.mappers
    com.faceRecognitionJava.model.dto
    com.faceRecognitionJava.model.entity
    com.faceRecognitionJava.repository
    com.faceRecognitionJava.services
    com.faceRecognitionJava.services.impl
      CustomUserDetailsService.java
      FaceRecognitionService.java
      RoleService.java
      UserService.java
  src/main/resources
```

```java

@Service
public class FaceRecognitionService implements IFaceRecognitionService {

    @Override
    public void getDataSet() {

        String command = "python \"" + System.getProperty("user.dir") + "/../FaceRecognition/01_face_dataset.py\""

        try {
            Process p = Runtime.getRuntime().exec(command);
            try {
                p.waitFor();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } catch (IOException e) {
            e.printStackTrace();
```

FaceRecognitionService is the core of the application. Here all the scripts are prepared for be called.

Chapter 4

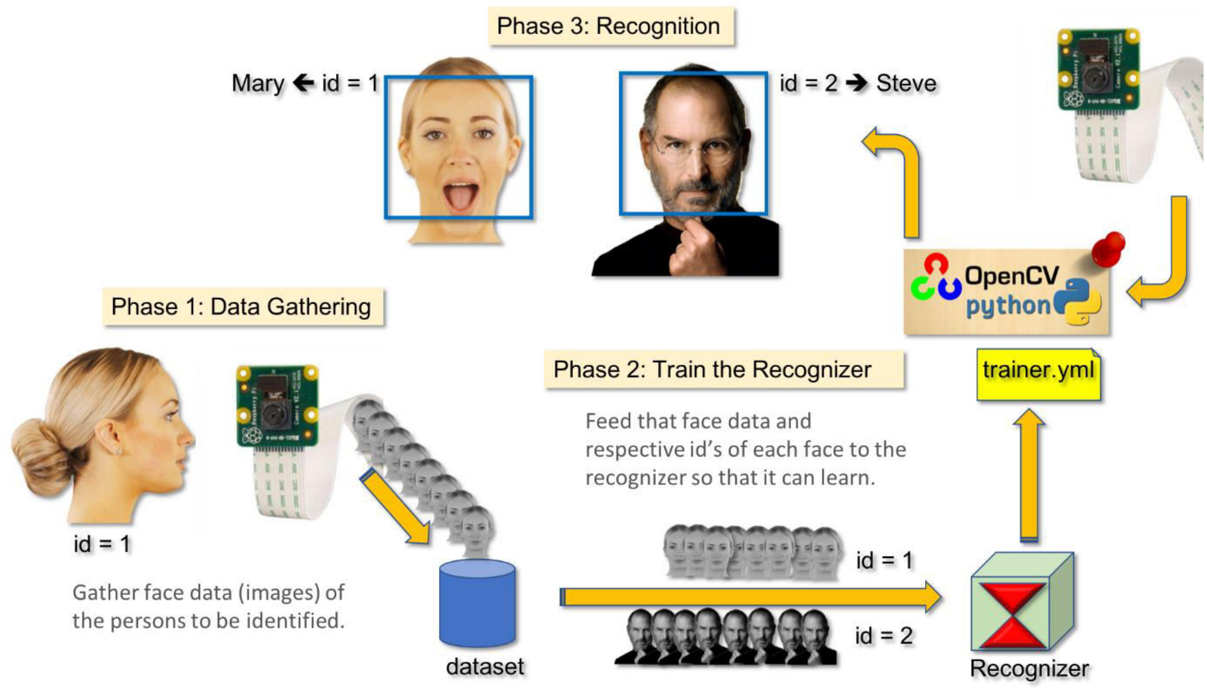
4.1 Proposed approach

4.1.1 Method description

To create a complete project on Face Recognition, we were working on 3 very distinct phases:

1. Face Detection and Data Gathering
2. Train the Recognizer
3. Face Recognition

The below block diagram resumes those phases:



Phase 3: Recognition
Mary ← id = 1
id = 2 → Steve
OpenCV
python
Phase 1: Data Gathering
Phase 2: Train the Recognizer
trainer.yml
Feed that face data and respective id's of each face to the recognizer so that it can learn.
id = 1
Gather face data (images) of the persons to be identified.
dataset
id = 1
id = 2
Recognizer

OpenCV contains many pre-trained classifiers for face, eyes, smile, etc. We are using some of them in our application.

Step 1:

- Load the image
- Call our classifier function, passing it some very important parameters, as scale factor, number of neighbors and minimum size of the detected face.

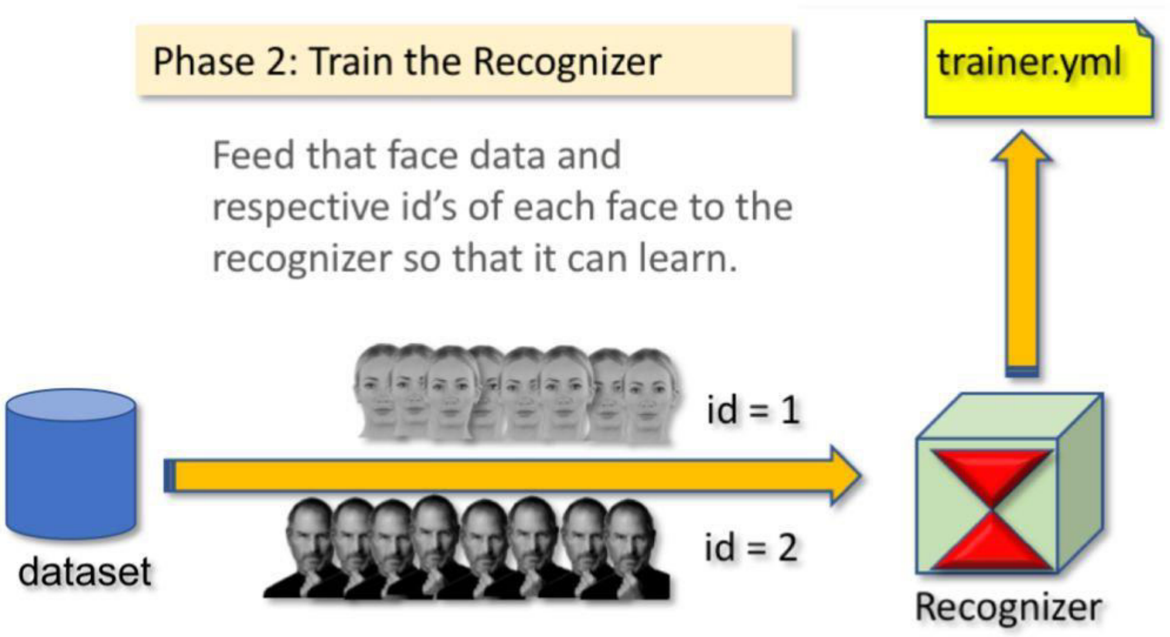
```
faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.2,
        minNeighbors=5,
        minSize=(20, 20)
    ), where: - gray is the input grayscale image
```

- scaleFactor is the parameter specifying how much the image size is reduced at each image scale. It is used to create the scale pyramid.
- minNeighbors is a parameter specifying how many neighbors each candidate should have, to retain it. A higher number gives lower false positives
- minSize is the minimum rectangle size to be considered a face.

The above function will detect faces on the image. If faces are found, we save the image as a new file on a “dataset” directory. Each file’s name will follow the structure: User.id.count.jpg.

For example, for a user with a id=1 and count=1 te file on dataset directory will be something like: User.1.1.jpg.

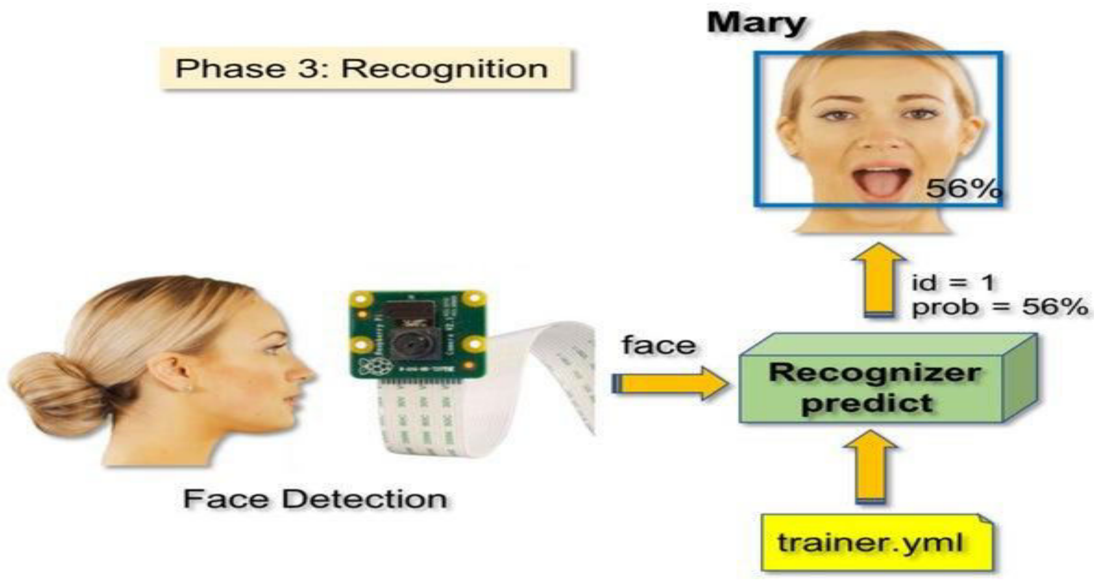
Step 2


Phase 2: Train the Recognizer
Feed that face data and respective id’s of each face to the recognizer so that it can learn.
trainer.yml
id = 1
id = 2
dataset
Recognizer

On this second phase, we take all user data from our dataset and “train” the OpenCV Recognizer. This is done directly by a specific OpenCV function. The result will be a .yml file that will be saved on a “trainer” directory.

Every time we performed Phase 1, we must run the Phase 2.

Step 3



On phase 3 we capture a face and if the person had his face captured and trained before, out recognizer will make a “prediction” returning its id and the index and shown how confident the recognizer is with this match.

The recognizer.predict(), will take as a parameter a captured portion of the face to be analyzed and will return its probable owner, indicating its id and how much confidence the recognizer is in relation with this match. Note that the confidence index will return “zero” if it will be considered a perfect match.

Chapter 5

5.1 Conclusion

Proposed system for the tracking of the faces does perform as we originally expected. There seems to be some problems regarding the determination of the smile detection, maybe due to the fact that the detected points are on a surface that is characterized by just a few shadows that appear in some tracked region.

For future development we will try to change how the login is working, to get in the app based on camera detection We also want to improve the algorithm because in some cases the complexity is n^3 and also want to add more emotions beside the smile one.

Even though the smile tracking system works for quite some time, we have been able to find the appropriate settings so that they can perform well enough to determine the girls and the smile to a large extent. Based on these calculations we could count, the number of smiles that were found in the 4100 photos.

Chapter 6

6.1 Bibliography:

1. FDDB: Face Detection Data Set and Benchmark
http://vis-www.cs.umass.edu/fddb/

2. Real-time Face Recognition: an End-to-end Project
https://www.instructables.com/id/Real-time-Face-Recognition-an-End-to-end-Project/