**FACIAL RECOGNITION SOFTWARE**

Name: Denisa Bianca Deac

Technical University of Cluj-Napoca, Computer Science

FACIAL RECOGNITION SOFTWARE

Contents

[1. Introduction 3](#_Toc121207297)

[1.1. Context 3](#_Toc121207298)

[1.2. Specifications 3](#_Toc121207299)

[1.3. Objectives 4](#_Toc121207300)

[2. Bibliographic study 4](#_Toc121207301)

[3. Design Analysis 6](#_Toc121207302)

[4. Implementation 8](#_Toc121207303)

## Introduction

### Context

Facial recognition software is widely used in many different areas of technology, such as identity verification, touchless access control, data security and face blurring for privacy.

This kind of application can be designed and implemented in many different forms: from a very simple and straightforward way to an extremely complex and accurate implementation. The complex form can serve in more serious matters as data security and access.

The goal of this project is to design and implement a versatile software program that is able to recognize the face and its distinctive features like nose, eyes and mouth. This kind of software represents a simpler smaller part of many other possible applications that require face identification. It can be adapted and enhanced in such a way that it could cover a larger area: not only recognizing facial features but also processing them on a larger scale (editing face features, comparing it for security and access purposes).

### Specifications

My application will be implemented in Java language and run in IntelliJ IDEA. For complex image processing such as face/object detection, the OpenCV library is used. My program loads the native OpenCV library in order to use Java API.

I want my facial recognition software to be user friendly: uncomplicated, easy to run by any user. In order to do that, the input must be intuitive and simple to set for the user and the output must be nicely displayed and easy to understand.

The way my facial recognition software works is extremely straightforward. The application will use the camera connected to the computer on which the program runs. This will be considered as the input of the program: using the camera, the user will have a simple method of taking a picture that will be sent to be processed by the application.

Once all the face features are identified (meaning the image was processed and the face of the subject was recognized), the face area will be highlighted: marked with a rectangle. This will represent the output of the program and it will be displayed on the screen.

### Objectives

The main objective of the project is to create a facial recognition application that identifies distinct facial features and processes them in order to identify the whole face.

One other objective is to understand and learn how image processing works in Java. These aspects are achieved by using OpenCV library, along many others that will make the work easier.

## Bibliographic study

The recognition of the facial features for this application will be done by using OpenCV. It stands for Open-Source Computer Vision and it’s a popular computer vision library.   Computer Vision allows computer software to understand and learn about the visualizations in the surroundings.

OpenCV is a video and image processing library, and it is used for image and video analysis, like facial detection, license plate reading, photo editing, advanced robotic vision, and many more.

While OpenCV has the job of identifying the face of a person, the whole process will be initiated and viewed by the user through a graphical user interface that will be created using JavaFX.

With JavaFX, a window will be created, having a button to take the photo, one to use an already existing image and, in center of the window, an ImageView to display the processed imaged . The button will start the work of the OpenCV library.

For the Face Recognition, OpenCV faceRec module includes three algorithms: using Eigenfaces, Fisherfaces, Local Binary Patterns Histograms. Each algorithm follows the different approaches to extract the image information and perform the matching with the input image.

The first steps to start the face detection is to load the OpenCV native library and then instantiate the CascadeClassifier class of the package org.opencv.objdetect.This class is used to load the classifier file. Instantiate this class by passing the xml file lbpcascade and haarcascade xml files for each indivitual facial feature.

This class contains a method detectMultiScale() that accepts an object of the class Mat holding the input image and an object of the class MatOfRect to store the detected faces. This function is prepared to detect faces even with a small size (the minimum size possible is given as a parameter).

 We will need to process the image. For that first, we will convert the image to Grayscale using OpenCV’s cvtColor function. We first create an empty Mat. Then we pass the source image as the first parameter and empty mat we just created as the second destination parameter. The third parameter describes what color space are we converting to and from. In our case, it is RGB to Grayscale. We can also equalize the image histogram to improve the result.

In order to display the result, we need to transform the MatOfRect object into an array<Rect>.

From one picture, the algorithm will be able to detect multiple faces.

Using the same principal as above, distinct face features will be detected.

## Design Analysis

My application will have three main functionalities available to the user through the Graphical User Interface. The user can:

* Use the computer camera to take an instant photo to be given to scan;
* Use an input image to be given to scan
* Visualize the image with the detected face features in the Graphical User Interface

Diagram

Description automatically generated

The user can access these functionalities by pressing the available buttons named “Take Photo” and “Scan Image”.

For each use case, several messages will be displayed on the application console to be easier to verify errors.

1. Take photo with computer camera:

* Success Scenario: the “Take Photo” button is pressed, the image with scanned features is displayed on the window and messages stating success are displayed
* Alternative Scenario: display message if camera is not opening, fxml files used to detect face features are not found or exceptions are thrown

2. Scan input image:

* Success Scenario: the “Scan Image” button is pressed, the image with scanned features is displayed on the window and messages stating success are displayed
* Alternative Scenario: display message if image is not found, fxml files used to detect face features are not found or exceptions are thrown

The internal design of the project is represented by several packages and classes. The package containing the implementation of the project is called com.example.facialrecognition and contains a class MainApplication (where the main method runs) and Controller(where the components from the GUI is connected to the functionalities).

Diagram

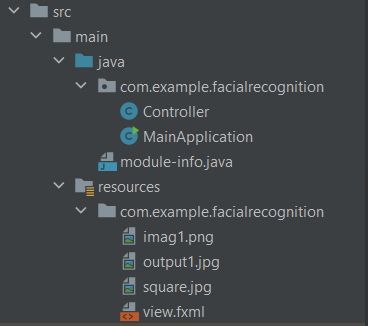
Description automatically generated

The MainApplication class extends Application, from javaFX in order to start the interface.

Controller class uses the methods of openCv library to identify the facial features of from the given input.

## Implementation

The implementation of my project is done according to the described design. The classes representing the actual implementation are located in the packages “main”, “java”, “com.example.facialrecognition”. The other package called “resources” contains the file for JavaFX interface and the pictures used and process by the application.



The class called “MainApplication” loads the fxml file of the graphical user interface and handles.



The “Controller” class contains methods that performs all the needed operations. The linkage between the graphical user interface and the code is made through three methods: “inputPhotoButtonClick”, “takePhotoButtonClick” and “showPhotoButton” that have the required “@FXML” annotation.

The first method takes a photo from the project “resources” and gives it to the “processFrame” method in order to be scanned for facial features

**Chart

Description automatically generated with medium confidence**

The second one uses the features provided by OpenCV. It opens the computer camera by using VidepCapture and takes a photo . It is transformed into mat object such that it can be processed.

**Text

Description automatically generated**

The classifiers needed for OpenCv functions to work are loaded in the method called “initClassifiers” and is appended in the “processFrame” method that does the actual scanning of the given image. Error messages are displayed in the console in case the path for the cascade file is wrong.

The method “processFrame” uses, at first, the “face cascade“ to identify the face in the photo.

Text

Description automatically generated

If it is found, in that area the eyes, nose and mouth are processed in the same manner. All these facial features will be marked by a squares of different colors.

Text

Description automatically generated

## TESTING AND RESULTS

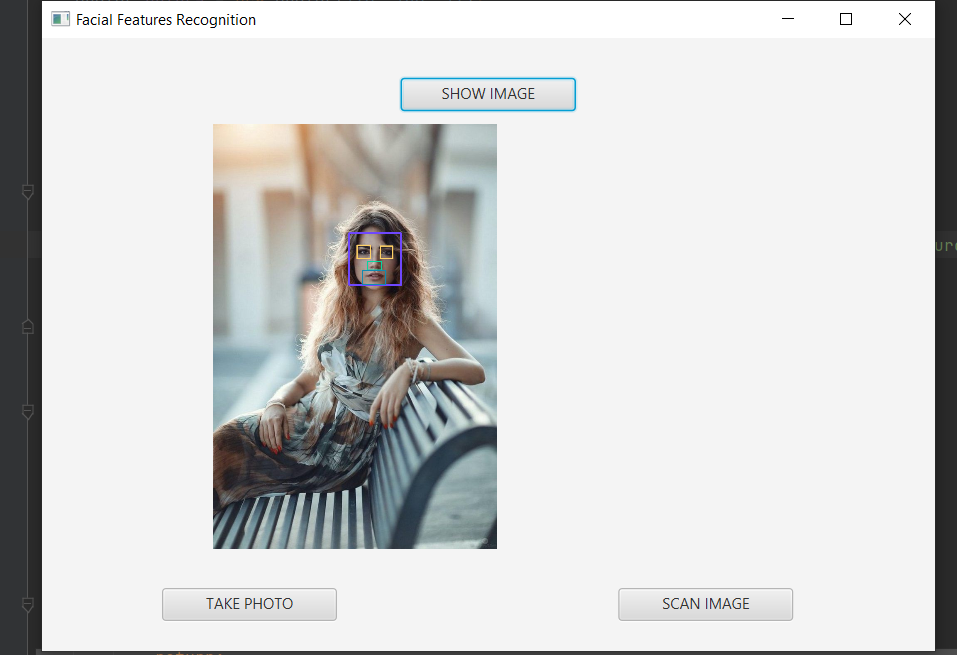
In order to test the application, I created the Graphical User Interface Using JavaFx where the processing of the facial features is done by pressing a button with the wanted option: to scan an image already given(hardcoded) or to take an instant picture.

When first opening the application, this is the format of the GUI.

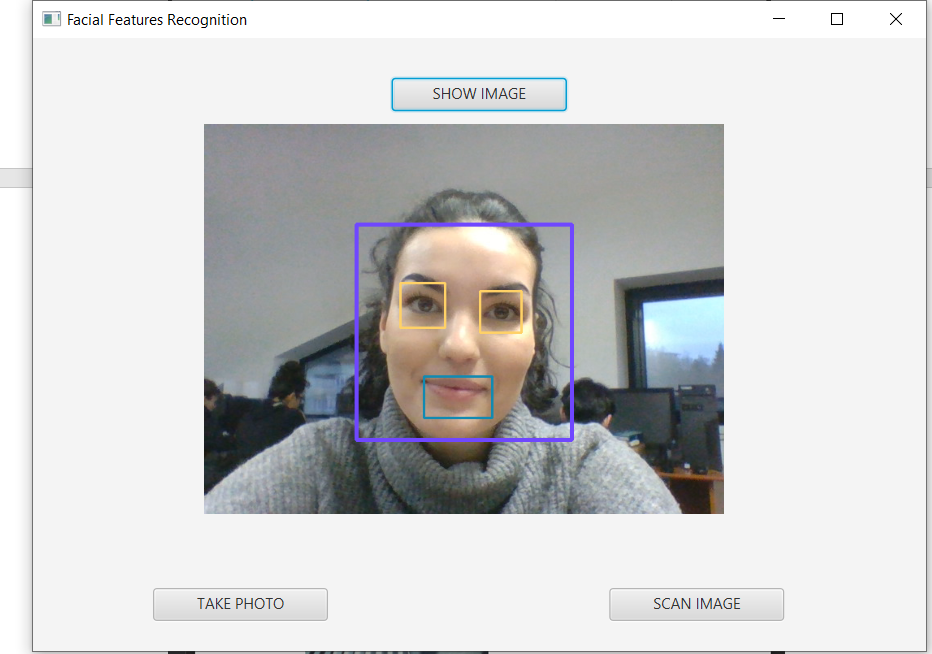
Graphical user interface, text, application, chat or text message

Description automatically generated

By pressing the “Scan Image” button, the given image is processes and messages are shown in the console. After the processing is done, by pressing the “Show Image” button the output will appear on the screen.



The “Take Photo” button does practically the same thing as described above, put the image being processes is taken instantly, reason for some inaccuracies.



The operations can be repeated without restarting the application.

## CONCLUSION

In conclusion, the facial recognition application was an entertaining way of learning to work with the OpenCV library and with all the features it brings.