Android application for
Face Recognition

Kévin Letupe

18 Juin 2013
Abstract

This report presents all the processes I use to program an android application of face recognition. At the beginning, I used the android API, after a long study of the android literature, to make this application. Because of devices problem, I had to forsake this option, and I used the OpenCV library, a library created for image processing. Use this library was really hard, because it is a complex one, but the OpenCV literature on internet permit me to master its utilization. After that, I thought with my tutors that it could be more interesting to make my own face detection library. Now, I arrive to had six specific features on each face. Those six parameters permit us to make the difference between two faces. I will describe all my work in this report.

Acknowledgement

To begin, I really want to thank my two supervisors, Ulrich Bjerne and Hans Henrik Hansen, who help me to make my project by giving me advices. With them, I learned a lot of interesting things about face recognition.

Thanks to Ingrid Billeso, who manage all the Erasmus placements. Thank to her, my placement was really well organised and I have no problems to easily integrate the Denmark way of life.

I want also to thank my parents, because they permit me to make this placement in Denmark by helping me financially.

To finish, I really want to thank my friend, to support me during this placement. Their absence has been difficult for me, but they gave me the motivation to work harder for this project.
Contents

I  Introduction 4

II  Literature review 7

1 Introduction to the programming languages. 8
   1.1 The Java programming language 8
   1.2 The XML language 9
   1.3 The PHP programming language 10

2 Introduction to the work environment 11
   2.1 Presentation of Android 11
   2.2 Eclipse IDE 12
   2.3 The OpenCV library 13
   2.4 Theory of an android application 13

3 Mathematical theories 15
   3.1 The cosine law (Al Kashi theorem) 15
   3.2 Calculation of a distance in a Cartesian plan 16

4 The Methodology 17

III  Analysis of the problem 18

5 Creation of a basic camera application 19

6 Face Recognition with the google API 22
   6.1 FaceDetectionListener implementation 22
   6.2 FaceDetector implementation 23
Part I

Introduction
Nowadays, the quality of our mobile phone cameras permit us to take high quality pictures. So it is possible to compute many kinds of recognition on those pictures. Face detection is a kind of recognition mostly used in our actual society. We use it all the days on Facebook to tag people on our pictures. It is also used in video game, with the kinect concept, or in security, to permit access to private spaces. And it’s just few examples of face recognition uses, because in our modern society, face detection will be everywhere.

The aim of this project was to create a mobile application for Aarhus University able to compute the recognition of humains faces. The goal of this application is to allow the access of private room in an hospital just for specific employees. First of all, I used the android API to create this, but paradoxically, just few of android devices can support that. We had the OpenCV library option also, but it is really complex. That’s why the Aarhus University decided to create their own algorithm for face detection. On this project, I had to work on the creation of the application and on the implementation of basic algorithms to detect face.

To create the application, I was in front of some problems :

• Against all expectations, the android API was useless for this application. Many android devices were not able to support the application.

• To implements the algorithm, we had to realize big research on grey colors.

• The detection of the eyes and the mouth implies to find a methods to make the different between those and the face.

• Create a database was necessary to store all the obtained informations.

Face to this problems, the solution was to transform the picture taken by the camera in a grayscale picture. After a long research on the grey color, we were able to make the different (by studing the pixels of the picture) between the face and the features of the face. That permit us to compute six important distances or angles on each face. By storing facial features of each people in a database, we are now able to detect face and make the difference between two peoples.
Problem statement

In order to protect some important areas in the Denmark hospital, the Aarhus university needs a powerful system of face recognition that can be easily install in hospitals.

But to have our own algorithm, We decided to recreate all the processes from the beginning. How did we do to implements a powerful method of face recognition?

I will describe you the way we took to implement as well as possible a new face recognition algorithm.
Part II

Litterature review
Chapter 1

Introduction to the programming languages.

1.1 The Java programming language

Figure 1.1: The Java logo.

The Java language was created in 1995 by James Gosling and Patrick Naughton, Sun Microsystems employees. This is the language we have used on this project. Java can work on all the operating system. Its motto: 
"write once, run anywhere". That is why Java became one of the most popular programming language during the ten last years.
Java is an object-oriented object. All the Java components are objects that interact each other. Those objects are represented by classes. All classes have their own attributes and methods. This is an example of a basic class:

```
public class MyClass {
    // declaration of the class
    private int class_attribute; // attribute
    public int class_attribute2;

    public int class_methods() { // declaration of the methods
        return class_attribute; // body of the methods.
    }
}
```

The keyword "private" doesn’t allow the other classes to use "class_attribute". It is a way to hide important attributes or methods. Indeed, with this keyword, the other classes can’t use those. Sometimes it is necessary to improve the security of a class. Normally, all attributes have to be private, and programmers must add "getters" methods to the class to catch them. This is exactly that the example above show. We have to pass by the method "class_methods" to catch "class_attribute". It is the principle of encapsulation. The "public" keyword allow all the other classes to use methods/attributes.

The last important concept of Java is the principle of inheritance. It is possible to reuse the old Java code thanks to the keyword "extends" during the declaration of the class:

```
public class MyClass2 extends MyClass {
    // class who inherit from MyClass
}
```

Thanks to this syntax, "MyClass2" have already "class_attributes2" and "class_methods". Programmers don’t have to rewrite those. However, "MyClass2" doesn’t have "class_attribute", because it is private. The keyword "protected" can be used also to declare functions or attributes. This keyword allow access only to the children classes. Only the public/protected attributes and methods can be inherit. Thank to inheritance, developers can always reuse their codes.

1.2 The XML language

The XML (Extensible Markup Language) is a computer markup language. It permit to organize informations in a way that allow both computer and
human to understand. XML has been declined in a lot of formats, like XHTML, used for the internet.

The informations are organized as a tree form. That means a XML file contain a root markup who include all the other markups of the file. Thanks to this, there is a real hierarchy of informations, always in the goal to organize those as well as possible. This is an example of a xml file:

```xml
<html>
  <head>
    <title> My title </title>
  </head>
  <body>
    <p> My text </p>
  </body>
</html>
```

This is a really basic example of an internet page, using the XHTML. The "html" markup is the root of the file, and the other markups form the information hierarchy.

### 1.3 The PHP programming language.

![PHP logo](image)

Figure 1.2: The PHP logo.

PHP: Hypertext Preprocessor, or PHP, is a web programming language using to create web dynamic pages. It only works on server side. Indeed, a HTTP server is needed to interpret PHP syntax for after creating a full web page understandable by any web browser.
PHP is a weak typing language. That means the programmers don’t have to indicate the type of a variable. It is the task of the server to detect the type of those, in function of the way they are binary representing.

Since the fifth version (2004), PHP integrate an oriented-object dimension (similar to Java).

Chapter 2

Introduction to the work environment

2.1 Presentation of Android

![Bugdroid, the green Android logo.](image)

For this project, we had to create an application for mobile phone. Indeed, mobile can be easily put anywhere. It was one of our goals: install as easily as possible our application in a hospital. We have chosen to use the android platform. Nowadays, 56.1% of mobile phones work on android operating system.
It’s an open source operating system, distributed under the Apache license (a free software license). It uses the Linux kernel, implements in language C. Android’s graphical interface is written in Java.

The greatest interest of Android is his software development kit, or SDK. Thanks to the SDK, every programmers who know C or Java language can implement an Android application. That permit Android to have a huge community of developers.

2.2 Eclipse IDE

Eclipse is a free Integrated Development Environment implemented in Java. The first version was launched in 2001 by the eclipse foundation which includes important computers companies, such as IBM, Google or Oracle. It has been created at first for Java development. Nowadays, Eclipse support more than 20 programming languages like C, XML, PHP, etc...

Eclipse include a plugin created for android development, named ADT (Android Development Tool). Add to the Android Software Development Kit, Eclipse is a complete IDE to program android applications. The Android SDK include all the android libraries, a debugger, an emulator to test the application, samples of code and tutorials. It also permit Eclipse to understand the XML android syntax.

![Android SDK Manager](image)

Figure 2.2: The Android SDK for Eclipse.
2.3 The OpenCV library

The Open Computer Vision library is a free graphic library developed by Intel in C. It is under the free BSD license. OpenCV has a huge community because this library is used by individuals and businesses. It permit to make really difficult calculations on pictures and video streams. Its features are large:

- image processing
- video stream processing
- graphic processing in real time
- form recognition
- image analysis.

OpenCV has several API (Application Programming Interface) that allow this library to be used in different programming language, like C, C++, Java or Python. Each module of OpenCV is specialized in a specific function. For example, the module highgui permit to create graphical user interface. OpenCV is available on every operating systems.

2.4 Theory of an android application

To program a basic android application, programmers need to know the Java programming language. Moreover, some important rules have to be respected to obtain a valid project.

All the features of an android project are combined in the file "Android-Manifest.xml". It is a xml file. All the important informations about the project are in this file:

- the minus and target sdk version used by the application.
- the devise permissions, for example to permit the application to use the camera.
- the official name of the application.
- the icon or the background picture of the application.
All this features can be set by the programmers.

The folder "res" is an other important part of an android project. This folder contain a lot of other folders. All the pictures and the important files used by the application must be stored in a subfolder of "res". The graphical interface settings of the project can be set in the file "myactivity_main.xml" which are in the folder "res/layout/". Here again, it is a XML file. Programmers can drag and drop components, called widgets, to built the interface or set all the parameters in the XML file. This is an example of a widget properties in XML:

```xml
<Button
    android:layout_marginLeft="30dp"
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Take Picture"
    android:onClick="takePicture" />
```

As we can see, a lot of properties can be set for each widget, such as its height or its margin. Here the most important lines are the lines `android:id="@+id/button1"` and `android:onClick="takePicture"`. The first one gives a name to the widget, whereas the second calls the function `takePicture()` in the main class of the application (we will see this farther) when we click on it.

![Figure 2.3: The interface to change android graphical properties in eclipse.](image)
All the project have to contain one class who inherit from ”Activity” :

```java
public class MyApplication extends Activity {

This class is the hearth of the application. All the interactions between the
graphical interface and the rest of the application must be make here. The
android project can contain only one class who inherit from ”Activity”. The
most important method in this class is :

```java
protected void onCreate(Bundle savedInstanceState) {
```
This method is automatically called during the creation of the application.
It is recommended to initialize all the application settings in ”onCreate”.
One function permit programmers to make the link beetwen this class and
the widgets of the graphical interface seen above :

```java
Button button = (Button) this.findViewById(R.id.button1);
```
With this method, we can interact with the Object button in the class.
We can for example set the text in it. It is also possible to make this in
”myactivity_main.xml”.

Chapter 3

Mathematical theories

3.1 The cosinus law (Al Kashi theoreme).

![Figure 3.1: A basic triangle.](image)

Figure 3.1: A basic triangle.
In any triangle, the cosine law says:

\[ c^2 = a^2 + b^2 - 2ab \cos(\gamma) \]

So we can also know the value of an angle by changing a bit the equation above:

\[ \gamma = \arccos\left(\frac{a^2 + b^2 - c^2}{2ab}\right) \]

This two equations will be useful for us to compute three distances on the face: the distance between the two eyes, the distance from mouth to left eye and the distance from mouth to right eye. After we will compute also the value of the three angles of the triangle formed by the three lengths.

![Figure 3.2: the cosine law on a face.](image)

### 3.2 Calculation of a distance in a Cartesian plan

A Cartesian plan is a plan formed by two perpendicular number lines called axes. If we have two points named A, with the coordinates \((x_A, y_A)\), and B, with the coordinates \((x_B, y_B)\), the way to calculate the distance between the two points in this plan is:

\[ AB = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \]
Chapter 4

The Methodology

During my project, I always realised my tasks in 5 steps.

First, I had to make research on the internet to know how the technologies and how the programming languages work. The targets of my research were tutorials explaining the rules about the librairies I used. I used only internet because all the informations for computer science can be find on the net. However, I had to take care about the informations I found. Many tutorials were useless and unsuitable for my problems.

Then, I had to install all the tools to use the good technologies. For this project, I installed for example the android SDK and the OpenCV library. Sometimes this step can take more time than expected.

The more interesting step is the implementation of the application, when the objectives are well defined and the tools well installed.

Then, I had to test all the functions I created to remove bugs and errors. This step is really long, because it is hard to find it own bugs and errors in the code.

The last step is the improvement of the code, to make it more powerful. If the changes are too important, a new step of tests is necessary.
Part III

Analysis of the problem
Before the start of this project, I had no ideas about how to programme for android application. So I had to learn it before start the project. My supervisor Hans learned me all the important rules on the creation of android project, and I saw that I had to used Java to program applications. Fortunately, I’m confident with Java, so it was easy for me to understand how android applications have to be implemented.

Chapter 5

Creation of a basic camera application

At the beginning I had to create a basic application using the front Camera of a mobile phone. To create this application, there are two important classes that I needed to use: Camera and CameraPreview. Camera is a class of the android API. It permit us to set all the parameters of the camera device, and make action like take pictures. This is the way to create a Camera instance in the application:

```java
private Camera mCamera;

protected void onCreate(Bundle savedInstanceState) {
    mCamera = MainActivity.getCameraInstance();
    // We have to call this method at the creation of the application.
}

public static Camera getCameraInstance(){
    Camera c = null;
    try{
        c = Camera.open(1);
    }
    catch(Exception e){
        Log.d(NOTIFICATION_SERVICE, "Camera is not available (in use or does not exist)");
    }
```

19
CameraPreview is a class I have created. It permit to display on the screen things that the camera is seeing. CameraPreview extends SurfaceView. SurfaceView is a basic widget that permit to display things on the device screen. CameraPreview also implements SurfaceHolder.Callback. That permit to the class to receive all the informations about changes of the surface (rotation of the device, end of the application, etc.). This is a part of the code of CameraPreview:

```java
public class CameraPreview extends SurfaceView
implements SurfaceHolder.Callback {

private SurfaceHolder mHolder;
// the instance of this class permit us
// to change the parameters of the SurfaceView.

private Camera mCamera;
// our Camera.

public CameraPreview(Context context, Camera camera) {
    super(context);

    this.mCamera = camera;
    this.mHolder = this.getHolder();
    this.mHolder.addCallback(this);
    this.mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    // required on Android versions prior to 3.0
}

@Override
public void surfaceCreated(SurfaceHolder holder) {
    // The surface has been created, now tell the camera
    // where to draw the preview.
    try {
        this.mCamera.setPreviewDisplay(holder);
        this.mCamera.startPreview();
    } catch (IOException e) {
        Log.d(VIEW_LOG_TAG, "Error setting camera preview:");
        e.getMessage();
    }
}
```


The method `surfaceCreated()` is automatically called when an instance of CameraPreview is created. The function `this.mCamera.startPreview()` allow CameraPreview to start to display the Camera. There are also two other methods in this class, `onChange()` and `onDelete()`, but there are not interesting in our case. You can see the code of those two methods in the appendix.

Now it is important to link CameraPreview with the main class of the application, MainActivity:

```java
public class MainActivity extends Activity {
    private Camera mCamera;
    private CameraPreview mPreview;
    private Bitmap myBitmap;

    protected void onCreate(Bundle savedInstanceState) {
        mCamera = MainActivity.getCameraInstance();
        mPreview = new CameraPreview(this, mCamera);
        FrameLayout preview = (FrameLayout) this.findViewById(R.id.frameView1);
        preview.addView(this.mPreview);
    }

    "frameView1" is a FrameLayout widget I have put in the file "activity_layout.xml".
    Now with this code it is possible to display everything the camera is seeing in the widget "frameView1". The last important methods is :

    public void takePicture(View view){
        mCamera.takePicture(null, null, mPicture);
    }

    The function `takePicture()` call a method named `OnPictureTaken()`. The goal of this method is to take a picture and store it in a `Bitmap` (here, `myBitmap`). The code of `OnPictureTaken()` is in appendix.
Chapter 6

Face Recognition with the google API

With my two supervisors we decided first to use classes of the android API to make our application of Face Recognition. Indeed, Google decided to add to his API two important classes for our project:

- FaceDetectionListener, that permits to compute the position of the two eyes and of the mouth together with a rectangle that represents the bound of the face. This class works with the API level 14.

- FaceDetector, who permits to compute the distance between the two eyes and the mid-point between the eyes. This class works with the API level 1.

All the code I will show you in this chapter is simplified. I made it in order to have a chapter easily understandable. The real code is available in the appendix.

6.1 FaceDetectionListener implementation.

I had the choice between this two classes for the application. FaceDetectionListener was more suitable for our project, because I needed the position of the mouth. Indeed, I wanted to calculate the distance between the two eyes and the mouth and the three angles of the triangle formed by the three lengths. This listener permit to detect faces in real time on video streams. I created a new class to implement this listener:
```java
class MyFaceDetectionListener implements FaceDetectionListener {
    // All the class that implements FaceDetectionListener have to
    // override the function onFaceDetection()
    @Override
    public void onFaceDetection(Face[] faces, Camera camera) {
        if (faces.length > 0) {
            Log.d("FaceDetection", "face detected: ", faces.length +
            "Location left eye:", faces[0].leftEye.toString() +
            "Location right eye:", faces[0].rightEye.toString() +
            "Location mouth:", faces[0].mouth.toString();
        }
    }
}
```

All the classes that implement `FaceDetectionListener` have to override the function `onFaceDetection()`. `Face[] faces` is an array automatically generated by the function. This table is full of faces detected on the video stream. Here the video stream is the instance of Camera. Now we have to change the function `onSurfaceCreated()` in `CameraPreview`:

```java
mCamera.setPreviewDisplay(holder);
mCamera.startPreview();
mCamera.setFaceDetectionListener(new MyFaceDetectionListener());
mCamera.startFaceDetection();
// the line above will automatically called the function
// onFaceDetection()
```

This solution was really easy to establish. Unfortunately, the Aarhus university’s devices couldn’t support this class. Indeed, the camera of the device was not precise enough to detect the position of the mouth and the eyes. So I had to use the other class gave by the Android API: `FaceDetector`.

### 6.2 FaceDetector implementation.

We decided my supervisors and me to change the way to make face recognition, because this class can’t give the eye positions and the mouth position. `FaceDetector` is only able to give the distance between the two eyes and the position of the middle point between those. So we chose 5 other reference distances to make the difference between two faces:

- the distance between the two eyes.
- the distance between the left eye and the left side of the Bitmap.
- the distance between the right eye and the left side of the Bitmap.
• the distance between the left eye and the top side of the Bitmap.
• the distance between the left eye and the middle point.

FaceDetector is able to detect face only on a Bitmap. Fortunately, the method onPictureTaken() store the picture taken in a Bitmap. So I created a new function in MainActivity:

```java
public void faceRecognition(View view) {
    int height = myBitmap.getHeight();
    int width = myBitmap.getWidth();

    // I create an instance of FaceDetector.
    FaceDetector faceDetector = new FaceDetector(height, width, 1);

    // I create here an array of face to store
    // the detected face. The length of this array is one
    // because we had to recognize only one face.
    Face[] faces = new Face[1];

    int findfaces = faceDetector.findFaces(myBitmap, faces);

    Log.i("FaceDetector", "Eye distance: " + faces[0].eyeDistance +
        " Midpoint: " + faces[0].midPoint.toString());
}
```

It was a very easy way to implement the face recognition. The function findFaces run through all the Bitmap given in parameters and store found faces in the array. But here again the devices I had were not able to detect faces by this way, because they don’t tolerate the class FaceDetector. So this solution was obsolete.
Chapter 7

Face Recognition with the OpenCV library.

The OpenCV library contains an android API. So it was possible for me to create an new application of face recognition using this library, entirely in Java.

I created an application that detected faces with OpenCV. But results of that application are only rectangles who contain the face. No important features, like the eyes or the mouth is detected with this method. So it was useless to use this library. However, you can see the code of this application in the appendix.

In more, the OpenCV library used highly complex algorithm. Indeed, it implements the recognition method of Viola and Jones. This method compare the picture with thousands of other examples of faces stored in a XML file called ”Cascade Classifiers”. It is a really powerful method, but really complex. So we decided to leave the OpenCV library to create our own face recognition algorithm.
Chapter 8

The Aarhus face recognition algorithm.

After many successive failures to create this application, my tutors and me thought it could be really interesting to create our own face detection algorithm. We decided to make a great work on the grey color. Indeed, our idea for this algorithm was to detect changes of grey color on a greyscale face picture.

8.1 The grey color.

Figure 8.1: Sample of our face detection algorithm.
When the picture is taken by the camera, we automatically transform it into a grayscale picture. After that, our idea is to run through the picture, pixel by pixel, to see if a pixel has a different gray than the pixel before. If it is, we detected the face, or one of the features of the face, like eyes or mouth.

Before that, I had to study the way the gray color is represented in computer science. For this project, I used the RGB (Red-Blue-Green) encoding method, that is to say every colors are represented by three values: the level of red, the level of green and the level of blue in this color, between 0 and 255. For example, white is represented by RGB(255,255,255) and black by RGB(0,0,0). A gray color has always the same level of red, green and blue. For example, a light gray is represented by RGB(196,196,196), and a dark gray is represented by RGB(82,82,82). In Java, a function, `int rgb(red, green, blue)` permit to transform the RGB color in an integer between -1 and -16777216. -1 is white, and -16777216 is black. But we were afraid that the function was not linear. Indeed, if results of that function are not linear, compare a pixel with the pixel before make no sense, because it is impossible to say if the pixel is more lighter or darker. I realized some test with different gray colors:

<table>
<thead>
<tr>
<th>Color</th>
<th>Level of red/blue/green</th>
<th>result of function Color rgb( r, g, b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>white</td>
<td>255</td>
<td>-1</td>
</tr>
<tr>
<td>very light grey</td>
<td>210</td>
<td>-29966996</td>
</tr>
<tr>
<td>light grey</td>
<td>192</td>
<td>-4144900</td>
</tr>
<tr>
<td>gray</td>
<td>153</td>
<td>-6710887</td>
</tr>
<tr>
<td>dark grey</td>
<td>102</td>
<td>-16065330</td>
</tr>
<tr>
<td>very dark grey</td>
<td>51</td>
<td>-13921773</td>
</tr>
<tr>
<td>black</td>
<td>0</td>
<td>-16777216</td>
</tr>
</tbody>
</table>

Figure 8.2: Test Results on the grey color.
As we can see I obtained linear results. The more dark the color is, the bigger the return integer is. In contrast, the more white the color is, the smaller the return value is. So our idea to compare each pixel one by one was workable. We decided to make a tolerance range for the pixel comparison of 5%. The maximum value is -16777216. So 0.05 * 16777216 = 838861. For example, a pixel has a value of 6500000. If the next pixel has a value under 6500000-838861 or above 6500000+838861, that means the algorithm detect something different on the face.
But to use this method, the face must be on a white background.

8.2 The FaceRecognition library.

To make this project reusable, I made it as a library. Thereby, any android project can use my algorithm and make a face recognition. In first, I created the class BitmapProcessing. It’s a basic class used to process all the operations on the Bitmap we will use to make the face recognition. For example, the function Bitmap toGrayScale(Bitmap originalBitmap) (as its name suggests, this function transform a Bitmap in a grayscale one) is in BitmapProcessing.

I also created the class FaceRecognition. All the detection algorithms are in this class. FaceRecognition contains some attributes:

```java
private int width; //the width of the face
private int height; //the height of the face.

//the position of the eyes and the mouth.
private Point eyes_left;
private Point eyes_right;
private Point mouth;

//the three important distance.
private double eyes_distance;
private double mouth_eyes_distance;
private double mouth_eyebrow_distance;

//the three angle.
private double alpha;
private double beta;
private double teta;

//the three ration.ratio_eyes is 1.
```
8.3 The width and the height of the face.

To compute the width of the face, I made the function `getFaceWidth()` that run through all the bitmap. I take as a reference the first pixel of the Bitmap. It is a pixel from the white background, so it is a light grey one. I test the other pixels from the top left corner to the right down corner one by one on the Bitmap. The leftmost and rightmost different pixel are store, and I calculate the distance between those at the end of the function.

This is the code of `getFaceWidth()`:

```java
class BitmapProcessing {
    // and the other ratio are calculated in function of ratio_eye.
    private double ratio_mouth_er;
    private double ratio_mouth_el;
    private double ratio_eyes;
    // the face.
    private Bitmap picture;
    // the gray scale face
    private Bitmap picture_gray;
    // the f a c e.

    public void getFaceWidth (){
        Point min_w = null, max_w = null;
        // I take as reference the first pixel.
        int original_grey_pixels = this.picture_gray.getPixel(0, 0);
        // beginning of the trail of the Bitmap.
        for(int i = 0; i < this.picture_gray.getHeight(); i++){
            // At the beginning of each row, I take the first
            // as a reference.
            original_grey_pixels = this.picture_gray.getPixel(0, i);
            for (int j = 0; j < this.picture_gray.getWidth(); j++){
                // If the pixel isn't in the grey range
                if( this.picture_gray.getPixel(j, i) < original_grey_pixels - BitmapProcessing.GREY_COLOR_RANGE || this.picture_gray.getPixel(j, i) > original_grey_pixels + BitmapProcessing.GREY_COLOR_RANGE ){
                    // if it is the first different pixel detected.
                    if (min_w == null && max_w == null){
                        min_w = new Point(j, i);
                        max_w = new Point(j, i);
                    }
                    // if a different pixel leftmost is discovered.
                    else if (min_w.x > j){
```
```java
    min_w.set(j, i);
    } //if a different pixel rightmost is discovered.
    else if (max_w.x < j){
        max_w.set(j, i);
    }
    original_grey_pixels = this.picture_gray.getPixel(j, i);
    }
    } //I calculate here the width.
    this.width = max_w.x - min_w.x;
}
```

I use the same way to have the height of the face. The code of `getFaceHeight()` is in the appendix.

### 8.4 The detection of the eyes

In order to detect the two eyes, I used the ROI system. The ROI are the Region Of Interest of the face. Indeed, we can cut a face in 3 important parts: the forehead (useless in our case), the eyes and the mouth. I created a function called `resize_Bitmap_Eyes_Track()`. It goal is to resize the Bitmap to obtain just the ROI of the eyes. This function is in the class

![Sample of ROI on a face.](image)

Figure 8.3: sample of ROI on a face.
To track the two eyes, I try to find a dark pixel to detect pupils in the eyes ROI. The system is almost similar then to find the width. I run through the resized Bitmap. If a pixel is under RGB(15,15,15), that means it is part of the pupil. If this pixel is in the left part of the resize map, it is the left eye, if it is in the right part, it is the right eye. The positions of the eyes I obtained were in function of the left top corner of the resize Bitmap. So at the end of the function, I had to make some calculations to have the positions of the eyes in function of the top corner of the original grayscale Bitmap. This is the code of this function:

```java
public void trackEyes(){
    // resize the Bitmap to obtain eyes ROI.
    Bitmap eyesBitmap = BitmapProcessing.resize(Bitmap_EyesTrack(
        picture_gray, width, height);
    // definition of the color of the pupils.
    int pupils_color = Color.rgb(15, 15, 15);

    this.eyes_left = new Point();
    this.eyes_right = new Point();

    // beginning of the trail of the resize Bitmap.
    for (int i = 0; i < eyesBitmap.getHeight(); i++){
        for (int j = 0; j < eyesBitmap.getWidth(); j++){
            // if the pixel is under the color of the pupils,
            // that means an eye is found.
            if (eyesBitmap.getPixel(j, i) < pupils_color) {
                // if it is on the left side of the Bitmap,
                // it is the left eye.
                if (j < eyesBitmap.getWidth()/2) {
                    this.eyes_left.set(j, i);
                }
                // else it is the right eye.
                else if (j > eyesBitmap.getWidth()/2){
                    this.eyes_right.set(j, i);
                }
            }
        }
    }

    // calculations to have the position of the eyes in function
    // of the original greyscale Bitmap.
    this.eyes_right.set( (this.eyes_right.x-(int)(this.width*0.1)+
        this.width/2, (this.eyes_right.y-(int)(this.height*0.25)+
        this.height));
    this.eyes_left.set( (this.eyes_left.x-(int)(this.width*0.1)+
        this.width/2, (this.eyes_left.y-(int)(this.height*0.25)+
        this.height));
}
```
8.5 The detection of the mouth.

The library uses also the ROI system to track the mouth. I have created in the class BitmapProcessing the function `resize_Bitmap_Mouth_Track()` to make a Bitmap for mouth tracking. The code of this function is in the appendix.

The function `trackMouth()` permit to detect the mouth in the resize Bitmap. This function uses the same method than `getFaceWidth`. I take as a reference the first pixel of the resize Bitmap, and I run through the Bitmap and test every pixels to know if it is a different grey pixel (always with the grey range). With this method I try to find two points : the leftmost and the rightmost pixels of the mouth. I calculate after the middle point of those. That permit me to have approximately the center of the mouth as a point. At the end of the function, like for the eye, I must make some calculations to have the position of the mouth not in function of the left top corner of the resize Bitmap, but in function of the left top corner of the original one. I don’t show here the code of `trackMouth()`, because it is mostly the same than `trackEyes()` or `getFaceWidth()`. But we can see the comment code in the appendix.

8.6 The calculations of all the distance.

Functions `trackMouth()` and `trackEyes()` permit to compute the position of the eyes and the mouth, but not the distances between those. I created the function `computeAll()`. It computes the three important lengths, the angles\(^1\) of the triangle formed by the three distances, and three ratios. Indeed, we need ratios because the three distances can’t be use to compare two faces because they are in function of the distance between the camera and the face. So I take as a reference the distance between the two eyes to calculate the ratio of the two other lenghts. For example, if the lenght between the two eyes is 100, and the two others are 230 and 250, then the three ratios will be 1, 2.3 and 2.5. With the ratio, the three lengths don’t

\(^1\) with the cosinus law
depend of the distance between the camera and the face. The code of the function `computeAll()` is in the appendix.

Chapter 9

The database.

To compare two faces, I had to store all the important informations about the detected faces in a database. My supervisors and me created a database that contain one table (faceRecognition) on the Aarhus university network in order to store 7 informations: the id of the face, three ratios and three angle values.

<table>
<thead>
<tr>
<th>id</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>alpha</th>
<th>beta</th>
<th>gamma</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.3</td>
<td>1.5</td>
<td>1.5</td>
<td>56</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>0</td>
<td>1.4</td>
<td>2.3</td>
<td>1</td>
<td>90</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>2.5</td>
<td>1</td>
<td>90</td>
<td>45</td>
<td>45</td>
</tr>
</tbody>
</table>

Figure 9.1: The table faceRecognition.

To interact with this database, I had to implement fives new functions in the class `MainActivity`:

- A `Select()` function, in order to extract data of the faceRecognition table.
- A `Update()` function, to update a row of the table if it is necessary.
- A `Insert()` function. It is the most important function, because it permit to add new faces in the database.
- A `Delete()` function, to delete a row of the table if it is necessary.
- A `Compare()` function, to compare a face with the other in the database.
The five functions send queries to a PHP script by the HTTP protocol. The queries are in form of url. The url transmits all the important informations. This is the code of Insert() :

```java
public void Insert(View view) {
    new InsertTask().execute();
}

private class InsertTask extends AsyncTask<Void, Void, String> {
    @Override
    protected String doInBackground(Void... voids) {
        // here we extract all the important informations of the
        // application's widgets.
        int id = Integer.parseInt(idbox.getText().toString());
        double distance1 = Double.parseDouble(a.getText().toString());
        double distance2 = Double.parseDouble(b.getText().toString());
        double distance3 = Double.parseDouble(c.getText().toString());
        double angle1 = Double.parseDouble(alpha.getText().toString());
        double angle2 = Double.parseDouble(beta.getText().toString());
        double angle3 = Double.parseDouble(teta.getText().toString());
        // creation of the url with all the important informations.
        String query = url + "insert" + ";id="+id+"&a=" + distance1 +
            "&b=" + distance2 + "&c=" + distance3 + "&alpha=" + angle1 +
            "&beta=" + angle2 + "&teta=" + angle3;
        String data = "";
        try {
            // permit to send the url.
            HttpClient httpclient = new DefaultHttpClient();
            HttpProtocolParams.setUseExpectContinue(httpclient.getParams(),
                false);
            HttpPost httppost = new HttpPost(query);
            httpclient.execute(httppost);
            } catch (Exception e) {data = e.toString();}
        return data;
    }
    @Override
    protected void onPostExecute(String result) {
        idbox.setText(result);
    }
}
```

The four other methods are made on the same layout. Their code is available in the appendix.

The script "database.php" receives informations and transforms them into SQL queries. Later, the SQL query are send to the database to interact with the table faceRecognition. This is a part of this script, always for the insert order :

```php
<?php
```
$mode = \$_GET[ 'mode' ]; // get mode from URL path

// connection to the database.
$dbHandler = mysql_connect( 'localhost', '*****', '*****' );
mysql_select_db( 'X13101', $dbHandler ); // select database

switch ( $mode ) {
    case 'insert':
        // we extract the information from the url.
        $id = \$_GET[ 'id' ];
        $a = \$_GET[ 'a' ];
        $b = \$_GET[ 'b' ];
        $c = \$_GET[ 'c' ];
        $alpha = \$_GET[ 'alpha' ];
        $beta = \$_GET[ 'beta' ];
        $teta = \$_GET[ 'teta' ];

        // the SQL query.
        mysql_query( "INSERT INTO faceRecognition( id, a, b, c, al, be, teta ) VALUES( '$id', '$a', '$b', '$c', '$alpha', '$beta', '$teta' )" );
        break;
}

With this system, all the interactions with the database are available, and it's now possible to compare a face on a picture with the informations store in the database. Indeed, the Compare() function permit me to make this. It return the id of the face in the database if a similar one is found. My supervisors and me reached our goal.
Chapter 10

Debate about the results.

My library permit to obtain six important features of a face. Six is a good numbers, because that allow the database to have an infinit of different faces. In more, the three distances we have after the recognition, are normally unique for each faces, so the angles also.

The worse ennemy of that method is the luminosity. At the same place, but with a different luminosity, results can be different, because the pixels don’t have the same grey. Sometimes it is a reason of incoherent values. An other problem it’s the obligation to have a white background. That implies to put the mobile phone in a specific place, and it is not possible all the time.

An other problem is the ”pixels noise”. Indeed if just one pixel is different in the background, the calculation of the width can be wrong. The risk of this problem increase during the detection of the mouth. The distances can be wrong because of this problems.
Part IV

Conclusion
The aim of my project for the Aarhus university was to create an application for face recognition. At the beginning we decided to work on existing classes or libraries, in particular the android classes and the OpenCV library. The devices we have can't support programs using the android classes. As for OpenCV, use that library, even if it is really optimized, was not possible, because we have no ideas about the extremely complex algorithms behind the used functions.

So we decided to create an new algorithm for the face recognition. After a big work on the way the colors are representing in computer science (especially the grey), I created my algorithm. The goal of this algorithm is to run through a picture to test each pixels. If a pixel is different than the pixel before, that means the algorithm detected a new feature on the face. That permit to obtain six different distances and the three angles of the triangle formed by that three distances. Those distances are stored in a database to compare two faces in a second time.

However, some factors can give random results: the luminosity and the "pixels noise". Futur improvements on this project have to be done in order to fix the two problems. Maybe it could be interesting to study the HSV color encoding format, because this format permit to have an influence on the lightness of each pixels. It could be great to find a way to remove all the "pixels noise".

Some other great improvements could be beneficial for this library. For example, it could be really interesting to find a way to detect more than one face on a picture, always with the algorithm I created. In a society where the face recognition is more and more used, it is important to have a powerful algorithm of face detection.
Part V

Literature list.


Android foundation, 2013, FaceDetector, Available at: http://developer.android.com/reference/android/media/FaceDetector.html


OpenCV foundation, 2013, Cascade Classifiers, Available at: http://docs.opencv.org/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html#cascade-classifier

OpenCV foundation, 2013, Android with OpenCV. Available at: http://opencv.org/platforms/android.html


The Eclipse foundation, 2013, Eclipse. Available at: http://www.eclipse.org/

The Php Group, 2013, Documentations. Available at: http://php.net/docs.php

Wikipedia, 2013, PHP. Available at: https://en.wikipedia.org/


# List of Figures

1.1 The Java logo. ........................................... 8  
1.2 The PHP logo. ........................................... 10  

2.1 Bugdroid, the green Android logo. ...................... 11  
2.2 The Android SDK for Eclipse. ........................... 12  
2.3 The interface to change android graphical properties in eclipse. 14  

3.1 A basic triangle. ......................................... 15  
3.2 the cosinus law on a face. ............................... 16  

8.1 Sample of our face detection algorithm. ................. 26  
8.2 Test Results on the grey color. ........................ 27  
8.3 sample of ROI on a face. ................................ 30  

9.1 The table faceRecognition. ............................... 33
Appendix A

Location of the code.

A.1 Application based on the Google API.

You can find all the code for this application on the CD in the android project "FaceRecognitionGoogleAPI"

A.2 Application based on OpenCV library.

You can find all the code for this application on the CD in the android project "FaceRecognitionOpenCV"

A.3 The Library FaceRecognition.

A.3.1 The library

You can find all the code for this application on the CD in the android project "FaceRecognitionLibrary-KevinL"

A.3.2 The application that uses the library.

You can find all the code for this application on the CD in the android project "FaceRecognition".