ABSTRACT

An Android application FaceAnalysis is designed to analyze a human face from a photograph. Users can take a photo by a custom camera or choose a picture by the Google Gallery. Then the application analyzes the input image and give some improvement advice about the input face to users. To achieve this function, three problems need to be solved: face detection, facial feature point detection and the aesthetic analysis. Both face detection and facial feature point detection are achieved using OpenCV. The face detection uses a Haar Cascade classifier and a LBP Cascade classifier. In the application, an image is checked by LBP Cascade classifier first. It is faster and costs less system resources, which is important for mobile devices. Then if there is no face detected by LBP Cascade classifier, Haar is used due to its high accuracy. The facial feature point detection uses ASM algorithm. Two aspects of Aesthetics is considered: the symmetry of a face and the Golden Ratios on a face. The UI of the FaceAnalysis is designed as a multi-pane layout, which adapts for different size screens. The FaceAnalysis has been tested on animal photos, beautiful human photos, ordinary human photos, non-ideal human photos, and the group with multiple faces.
TABLE OF CONTENTS

Abstract ................................................................. ii
Table of Contents ....................................................... iii
List of Figures ........................................................... iv
List of Tables ............................................................. vi
1 Introduction and Background ........................................ 1
   1.1 Image Processing .................................................. 2
      1.1.1 Face Detection ............................................... 2
      1.1.2 Facial Features Tracking .................................... 3
   1.2 Concepts of Aesthetics ........................................... 5
2 The App FaceAnalysis ............................................... 7
   2.1 The Basic Information ............................................ 7
   2.2 The User Interface ................................................ 7
3 System Design ........................................................ 12
   3.1 The Start Activity ............................................... 14
   3.2 The Camera Activity ............................................. 16
      3.2.1 The Navigation Drawer ..................................... 16
      3.2.2 The Custom Camera .......................................... 17
      3.2.3 Facial Feature Point Detection ............................ 17
   3.3 The FaceGrading Service ........................................ 19
   3.4 The Gallery activity ............................................. 24
   3.5 The Result Activity .............................................. 24
4 Results ............................................................... 26
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Example of the basic LBP operator.</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Example of the basic Haar Cascade feature extraction.</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>The 14 reference points used for component learning.</td>
<td>4</td>
</tr>
<tr>
<td>1.4</td>
<td>Golden Ratio Modules.</td>
<td>5</td>
</tr>
<tr>
<td>2.5</td>
<td>The OpenCV Manager check view.</td>
<td>8</td>
</tr>
<tr>
<td>2.6</td>
<td>The view of the Start activity.</td>
<td>9</td>
</tr>
<tr>
<td>2.7</td>
<td>Views from the Camera activity.</td>
<td>10</td>
</tr>
<tr>
<td>2.8</td>
<td>Views from the Result activity.</td>
<td>11</td>
</tr>
<tr>
<td>3.9</td>
<td>The view of the Start activity.</td>
<td>13</td>
</tr>
<tr>
<td>3.10</td>
<td>The view of the Start activity.</td>
<td>15</td>
</tr>
<tr>
<td>3.11</td>
<td>The ragb bitmap and mirrored bitmap with landmarks.</td>
<td>19</td>
</tr>
<tr>
<td>3.12</td>
<td>The label number of facial feature points.</td>
<td>21</td>
</tr>
<tr>
<td>3.13</td>
<td>The same image processed with and without regression.</td>
<td>23</td>
</tr>
<tr>
<td>3.14</td>
<td>The result activity shows on a tablet.</td>
<td>25</td>
</tr>
<tr>
<td>4.15</td>
<td>The animal faces.</td>
<td>26</td>
</tr>
<tr>
<td>4.16</td>
<td>The animal face results.</td>
<td>27</td>
</tr>
<tr>
<td>4.17</td>
<td>The images with multiple faces.</td>
<td>28</td>
</tr>
<tr>
<td>4.18</td>
<td>The results of images with multiple faces.</td>
<td>28</td>
</tr>
<tr>
<td>4.19</td>
<td>The sample of the beautiful faces.</td>
<td>29</td>
</tr>
<tr>
<td>4.20</td>
<td>The sample of the normal faces.</td>
<td>31</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.21</td>
<td>The sample of the beautiful faces.</td>
<td>32</td>
</tr>
<tr>
<td>4.22</td>
<td>The sample of the normal faces.</td>
<td>35</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>The correlation table of Horizontal Golden Ratio</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>The correlation table of Vertical Golden Ratio</td>
<td>22</td>
</tr>
<tr>
<td>4.3</td>
<td>The sample of beautiful human</td>
<td>30</td>
</tr>
<tr>
<td>4.4</td>
<td>The sample of ordinary human</td>
<td>32</td>
</tr>
<tr>
<td>4.5</td>
<td>The sample of non-ideal human</td>
<td>34</td>
</tr>
</tbody>
</table>
1. INTRODUCTION AND BACKGROUND

A recent trend has been to do variety things on mobile devices. There are more than one million Apps in Google Play, and the number keeps growing [9]. The Android App FaceAnalysis is designed to give an analysis result to users based on analyzing an input human face. The App cannot give an accurate result of the beauty level, which is influenced by many incomparable factors, such as race, temperament, and age. The purpose is to give a Golden Ratio analysis of faces. There are three major problems in this App that need to be solved: face detection, facial feature point detection, and human face analysis.

Both face detection and facial feature point detection belong to image processing. Due to the improvement of hardware, the mobile devices can process images through OpenCV in real time recently. The face detection uses OpenCV to test the photo pixel by pixel, in order to find where the face is [10]. Through the face detection, the app should have the raw data about the facial features. After processing the photo, the app will abstract the data of facial features. This is facial features collecting procedure. The raw data of the face that obtained earlier is analyzed based on the Golden Ratio from an aesthetic aspect.

The application allows users to input a photo and get the analysis result back immediately. This paper will proceed as follow. Chapter 1 introduces the background of this project. Chapter 2 presents a description of the external designs, which mainly is the user interface. Chapter 3 describes the system internal designs, which is the algorithm and codes part. Chapter 4 analyzes the results. Chapter 5 presents the conclusion.
1.1 Image Processing

Face detection and facial features tracking are concepts of image processing. They are particular cases of object tracking.

1.1.1 Face Detection

Face detection is to find a face or multiple faces in a picture or a snippet of video. There are four approaches: analysis of facial motion, model-based techniques, feature-based approaches and holistic analysis [1]. To analysis of facial motion, Mase allocates an axis for each muscle to estimate it. Meanwhile, Cohn et al. defined several facial feature points and track it. Model-based techniques map the images of faces onto predefined physical model [3]. Feature-based approaches are based on facial feature measurements: nose length, chin shape and so on. Holistic analysis is to analyze the holistic gray-level of the photo.

The Local Binary Patterns (LBP) Cascade classifier uses model-based techniques for face detection [8]. The LBP Cascade uses pattern recognition. It makes the checking pixel as center to capture a 3 by 3 table for analyzing the eight neighbors, as Figure 1.1 shows. The eight neighbors go through a threshold to find out the result of themselves, either be 1 or 0.

![Fig. 1.1. Example of the basic LBP operator.](image)
Haar Cascade classifier is another face detection method. It scans lots of images as two types: positive images with human face and negative images without human face to train itself [2]. The value of each feature is extracted by calculating the result of the pixels under white background minus the pixels under the black background. Figure 1.2 shows the Haar features used in face detection. It uses a fixed window to scan images. The MUCT dataset is chosen as pre-annotated training dataset for LBP and Haar Cascade classifier.

![Example of the basic Haar Cascade feature extraction.](image)

Fig. 1.2. Example of the basic Haar Cascade feature extraction.

### 1.1.2 Facial Features Tracking

Heisele, Serre, and Poggio presented a double layer classifier system [6]. First, they scanned the image through a fixed size window (50 × 50 pixel) to find the pyramid that is similar to the module composed by eyes, nose, and mouth. This is the component classifiers layer. If the system detected the pyramid module, this window will pass to the second layer classifier, or the system will detect the following pixel. The second layer classifier is a combination classifier. In this level the sub-
image will get rid of the background and hair. Figure 1.3 shows the 14 reference points used for component learning. Before doing component learning, the system will train data for identification. After detecting the sub-image, Farajzadeh, Faez, and Pan did a pre-processing [5]. They placed the center of the eyes in a line to locate the face, instead of using reference points.

Fig. 1.3. The 14 reference points used for component learning.

In this Android application, Active Shape Models (ASM) algorithm was chosen. This algorithm was proposed by T.F. Cootes, G. J. Edwards, and C. J. Taylor in 1995 [3]. This algorithm uses the target image to match a statistical model to find out the facial feature points. To implement this algorithm, machine learning must be involved, which requires a dataset with images, annotations, symmetry indices, and connectivity indices. The images means at least 200 human faces must be contained. The annotations means to annotate the facial features points. The symmetry indices means to make the indices for the mirrored points. The connectivity indices was to define the connectivity of the structure of points. In this project XM2VTSDB multi-modal face database was chosen, which contains four different images of 295 people, which was taken in four mouths.
1.2 Concepts of Aesthetics

In Aesthetics, there is a way to judge a human face by geometric measurement. First, the more symmetric the face is, the more beautiful the person is [7]. Second, the Golden Ratio, which approximates to 1.618, exists in Aesthetics [7]. As shown in Figure 1.4, there are vertical and horizontal dimensions for calculating Golden Ratio. Each largest rectangle is 1.618 times bigger of the smaller rectangle on its own color.

The following list defines the rectangle of each color and size. Vertical golden ratios:

1. White - Hairline : Eyebrow top : Eye top
2. Gold - Eyebrow top : Eyebrow bottom : Eye top : Eye bottom

Fig. 1.4. Golden Ratio Modules.
3. Blue - Eye pupil : Nose flair : Nose bottom
4. Green - Eye pupil : Nose bottom : Mouth
5. Green - Eye pupil : Nose bottom : Chin
6. Green - Eye pupil : Mouth : Chin

Horizontal golden ratios:

1. Gold - Face side : Eyebrows : Face side
2. Gold - Face side : Eye inside : Face side
3. Gold - Face side : Nose width : Face side
4. White - Face side : Eye outside : Nose center
5. Blue - Eye outside : Eye inside : Nose center
6. Green - Mouth outside : Lip cupids bow : Mouth outside
2. THE APP FACEANALYSIS

This chapter shows the basic information of FaceAnalysis and the user interface. The basic information explains the supported API version, the device requirements, and the expected results. The user interface part presents the number of activities, the number of screens, and the relationship between the activities and the screens.

2.1 The Basic Information

The FaceAnalysis is developed with OpenCV sdk and Android NDK except the regular Android develop sdk [?][11]. When the project was being developed, the project documents must under the OpenCV sdk folder and the OpenCV Library must be imported into the project. The App FaceAnalysis only supports API 14 and above. The OpenCV is for API 8 and above. The fragment design requires API 14 and above. If the user wants to use all the functions of the App, the device has to have a camera at least. Meanwhile the App also requires the devices with dual-core chips for real time face analysis. The result is two processed images with the suggestion of improving the facial features.

2.2 The User Interface

The App has four activities and one service: the Start activity, the Camera activity, the Gallery activity, the Result activity, and the FaceGrading service. Each activity is associated with a screen. There are seven self-defined classes in the project to work with the four activities.

The first time the App is used, the App checks if an OpenCV Manager has been installed. If no OpenCV Manager is detected, an alert dialog is displayed to help users install it manually under instructions, as Figure 2.5 shows.

The start screen is the main screen of FaceAnalysis. It is the soul which defines
the overall style of the user interface. It has a liner layout filled with a four options list view and a blank child liner layout, as Figure 3.10 shows. Users can go other screens from this screen by pressing the selections on the list view: Camera, Gallery, Help, and About. The Camera is linked to the Camera activity. The Gallery option is linked to the Gallery activity. The Help is linked to an alert dialog with the helping information. The About option is associated with a developing information alert dialog.

The beginning of Camera activity is the camera view. On the camera view screen, there is an action bar, a navigation drawer and two option menu items, as Figure 2.7 (a) shows. Figure 2.7 (b) displays that a hint shows up when a user press the Hint option menu item. If the user select Face Analysis item at this time, a warning of no taken picture shows up, see Figure 2.7 (c). The user can take a picture by clicking
the screen. After the picture was taken, the screen jumps to another screen, where the taken picture is displayed. The user can decide to process the picture or go back to retake. When the Face Analysis item is chosen, the App will check if a human face exists in this picture. If a human face has been detected, the process will link it to the post-process screen. Meanwhile, the analyzing and calculating of the detected face is processed in the background. Otherwise, an alter dialog pops up shown by Figure 2.7 (d).
Fig. 2.7. Views from the Camera activity.
Once the user choose the gallery section in the start screen, the screen will jump from FaceAnalysis to the Google Gallery to let users choose their desired pictures. After the user chooses a photo, the operation will jump to the Gallery activity. The rest screens are the same as the Camera activity.

The Result activity does nothing but shows results. Two fragment views are in it. The list view offers the options to the users. When the user chooses the Horizontal Result option, the right view shows the horizontal result of the detected face. It has a picture, which has been modified by colorful lines. Below the modified picture, the horizontal Golden Ratio analysis of the face is posted. It gives a few advice for this face based on the Golden Ratio. For example, in Figure 2.8 (a) it gives suggestion You need longer eyes. When the user chooses another option, he or she can see the vertical result image with a text result as Figure 2.8 (b) shows.

![Fig. 2.8. Views from the Result activity.](image)
3. SYSTEM DESIGN

The system design of App FaceAnalysis is explained in this chapter. This App has four activities and one service: Start, Camera, Gallery, Result, and FaceGrading. The flow chat of the system is shown in Figure 3.9. The general hierarchy is: Start - Camera/Gallery - FaceGrading - Result. However, the system is more complicated than this. Besides the Start activity, users also may enter to their needed activities from the Camera activity, the Gallery activity, and the Result activity. Therefore, the relationship between activities is not a liner relationship. It is a combination of liner and circle relationships.

There are three goals for the system: first, an accurate face analysis; second, a functional UI; third, a dynamic and multi-pane UI. To obtain an accurate face analysis result, the Camera and Gallery activities are designed to process the pictures by using OpenCV and analyze those processed pictures based on the Golden Ratio Algorithm [7]. A navigation drawer is added into Camera and Gallery for a better user experience. In order to create a dynamic and multi-pane UI, the Start activity and the Result activity use Fragments to build their main layouts.

In this App, Each Activity, has a multi-pane UI, which is a host screen with several nested sub-screens. To embed sub-screens into the host screen, an Android component fragment has to be used repeatedly. A fragment is a modular section, which can be considered a mini Activity. A fragment has a similar life cycle as an activity. Besides onCreate(), onStart(), onPause(), and onStop(), the Fragment has extra functions such as Attach(), onCreateView(), and onDestroyView() to communicate with its host activity. The Fragment is also the key component to achieve the dynamic UI. The rest of the chapter is divided into five sections. Each of them represents an activity or a service. The details of how an activity are associated with others and the fragments is presented in each section.
Fig. 3.9. The view of the Start activity.
3.1 The Start Activity

The Start activity is the first screen when users launch the FaceAnalysis. The Start activity does two major actions: first, checking the OpenCV Manager; second, connecting to other activities. Since the OpenCV Manager App has to be installed before any OpenCV involved App uses, the FaceAnalysis App has to check if the OpenCV Manager has been installed. Hence an OpenCV object BaseLoaderCallback is declared in this Activity to check the existence of OpenCV the Manager. In the BaseLoaderCallback object, a function named onManagerConnected(int status) needs to be overridden. When the status = LoaderCallbackInterface.SUCCESS (an OpenCV defined static value), it means the OpenCV Manager exists on the device. Otherwise, the function has to call super.onManagerConnected(status) to install the OpenCV Manager automatically.

Even though in the current phase there is no difference between using a single-pane layout and using a multi-pane layout, the dynamic and multi-pane design is adopted for future updates, which is discussed in Chapter 5. To accomplish the multi-pane interface, the Start Activity is associated with two .xml files: options.xml in layout folder and options.xml in layout-large folder. The reason of creating two .xml files with the same name in different folders is making the interface of this activity can fit for different screen size. The options.xml in layout folder, which is for the handset screens, only defines a fragment container. The options.xml in layout-large folder, which is for the tablet screens, defines two fragments: option_fragment and show_fragment. The weight of option_fragment is assigned 1, while the weight of show_fragment is assigned 2. Therefore the list view takes up one third of the screen on the left and the blank screen takes up to two thirds as shown in Figure 3.10.
Fig. 3.10. The view of the Start activity.

In the source code, two fragment classes are defined, one for each fragment nested in the main layout. Because the show_fragment is a blank fragment for now, only one Fragment class is defined: OptionsFragment. The OptionsFragment class associated with option_fragment defines the ListView. The OptionsFragment extends ListFragment. The onCreate() function is rewritten. In onCreate(), the API level is checked first to decide which version of the list item layout should be used. The class also defines the names of the list items and listens to the click to the list items. To pass the click action in the fragment back to the Start activity, both OptionsFragment class and the Start activity is modified.

1. The Start activity defines a public function onOptionSelected(int position) to connect the selected options to other activities.

2. The OptionsFragment class defines an interface called OnOptionsSelectedList-
tener, in which the onOptionSelected(int position) is called.

3. The Start activity implements the OptionsFragment.onOptionSelectedListener. The OptionsFragment class uses Attach() function to pass the position of clicked option back to Start activity.

3.2 The Camera Activity

As the name suggested, the Camera activity helps users access the camera. It is one of the two main functional activities. The other one is the Gallery activity. Two .xml files and three classes are associated with Camera activity. The design of Camera activity can be divided into three parts: the navigation drawer, the custom camera, the facial feature point detection.

3.2.1 The Navigation Drawer

For a better navigation experience, the navigation drawer is used in this activity. The two layouts used in this activity are activity_take_photo.xml and fragment_camera.xml. The activity_take_photo.xml, which is a drawer layout, is the main layout to create the view of the Camera activity. In this layout, a frame layout and a list view are defined. The list view is the navigation drawer. The frame layout is a fragment container. A drawer layout only can have two children: a frame layout and a list view, so the other views have to add to the frame layout as fragment. In the onCreate() function of Camera activity, after the navigation drawer and the action bar are set up, the fragment for the custom camera is declared and defined. In overridden function onPrepareOptionMenu(Menu menu) of navigation drawer, the option menu is defined to unclickable when the navigation drawer is open to avoid the misoperations.
3.2.2 The Custom Camera

In this activity to define a custom camera, two classes are created first: the self-defined surface view MySurfaceView and the CameraFragment. MySurfaceView controls the runtime camera view and its feedback. MySurfaceView is set by fragment_camera.xml as the view of CameraFragment. The CameraFragment implements OnClickListener to listen to the photo-taken action, which is triggered by click events. After a photo has been taken, the option menu item Face Analysis can be pressed. The procedure in Face Analysis item is as following:

1. Get the view of CameraFragment and get the path of the view, which displays the taken photo.

2. Check if path = null or path exists. If path = null, it means no photo has been taken. If path exists, it means there is no new photo has been taken since the App jumped from other activities to Camera activity. Then an alert dialog will pop up: You need to take a photo.

3. If path != null and path does exist, put path into existing path list and call face detection function imgProcess(path) to start process this photo.

4. If no face detected, finish and an alert dialog will pop up: No human Face Detected.

5. Otherwise, go face analysis then jump to Result activity.

3.2.3 Facial Feature Point Detection

In the App, OpenCV is implemented to process the images. To detect the face, the Haar Cascade classifier and the LBP Cascade classifier are used. To detect the facial feature points, ASM algorithm is used. An ASM library developed by Wei Yao
is imported to achieve those two algorithms [12]. The ASM library has one java class ASMFit and four C++ files: asmfitting.h, asmlibrary.h, vjgacedetect.h, and DemoFit.cpp. The trained data my68_1d.amf, haarcascade_frontalface_alt2.xml, and lbpcascade_fountalface.xml also come from the ASM library. For the use of this App, the pseudo code of detect facial feature points is describe.

1. Load asmlibrary and jni-asmlibrary.

2. Declare an object of ASMFit.

3. Read files my68_1d.amf, haarcascade_frontalface_alt2.xml, and lbpcascade_fountalface.xml.

4. Convert the taken photo into bitmap format and get a mirror copy of the bitmap.

5. Convert the bitmap and mirror bitmap from rgba to gray.

6. Use ASMFit.fastDetectedAll() function to detect face. If no face detected, use ASMFit.detectAall() function to detect the face. If still no face can be detected, then end the imgProcess(path) and return to activity.

7. If there are any face was detected by rule No. 6, use ASMFit.fitting() find the 67 facial feature points on the rgba bitmap and its mirror bitmap.

The bitmap of the processing image has to be converted into gray scale because the facial feature points only can be detected in gray color pictures in OpenCV. The ASMFit.fastDetectedAll() function uses LBP cascade algorithm and the ASMFit.detectAll() uses Haar cascade function. The LBP cascade algorithm is less accurate than Haar cascade algorithm. However, LBP cascade is faster and uses less hardware resources. Because of the limitations of mobile devices, using LBP cascade to detect faces makes the App more efficient. The feature points are marked by red circles in the rgba bitmap (Figure 3.11 (a)). In the mirrored bitmap the points also
are found (Figure 3.11 (b)). The mirror copy bitmap is for the Golden Ratio Analysis. It is explained in the next section.

![Figure 3.11](image)

Fig. 3.11. The ragb bitmap and mirrored bitmap with landmarks.

### 3.3 The FaceGrading Service

The service FaceGrading defines the algorithm to determine the Golden Ratio on human faces. Due to the large amount of computation, it takes few seconds to finish this service, which exceeds the App respond time. Therefore, the face analysis has to be processed in a service or an AsyncTask. The FaceGrading service judges the face from four aspects: the symmetry of the face, the horizontal Golden Ratio, the vertical Golden Ratio, and the variance of the Golden Ratio results. After the Camera activity passes the path of the photo and the coordinates array of the facial feature points of two bitmaps, the FaceGrading service starts to analyze the taken photo.
The formula of calculating the symmetry grade is:

\[
Grade_{sym} = \left[ F - \sum_{i=0}^{66} \frac{\sqrt{(x_i - x_{67})^2 + (y_i - y_{67})^2} - \sqrt{(x'_i - x_{67})^2 + (y'_i - y_{67})^2}}{N \cdot U} \right] \] (3.1)

In this formula, \( F \) stands for full score, which equals 100. The \( x_i \) stands for the x-coordinate of point \( i \) in original bitmap and the \( y_i \) stands for the y-coordinate. Meanwhile, the \( x'_i \) and the \( y'_i \) mean the coordinates of point \( i \) in mirrored bitmap. As Figure 3.12 shows, the point 67 is in the center of the face, so it is considered as the standard point. From point 0 to point 66, the distance of the point to the standard point is calculated while the distance of the point, under the same label number in the mirror bitmap, to the standard point is also calculate. The point in the mirror bitmap with the same label is the symmetric point of the point in the original bitmap under the same label. For an instance, the symmetric point of point 0 is point 14. In the mirror bitmap, the face is mirrored, so point 0 in the mirror bitmap is point 14 in the original bitmap. The absolute value of the difference of the two distance is divided by the product of \( N \) and \( U \) where \( N \) is the number of points. \( U \), equals 2, is the unit of minimum difference to be considered. The grade of symmetry is the rounding number of \( F \) minus the just calculated value, so the grade may be a negative integer.

The correlation table of Horizontal Golden Ratio is as following:
Fig. 3.12. The label number of facial feature points.

<table>
<thead>
<tr>
<th>The Horizontal Golden Ratio</th>
<th>The lines ( (\text{slope} = k) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold - Face side : Eyebrows : Face side</td>
<td>( \text{line}<em>1 : \text{line}</em>{24} : \text{line}_{13} )</td>
</tr>
<tr>
<td>Gold - Face side : Eye inside : Face side</td>
<td>( \text{line}<em>1 : \text{line}</em>{29} : \text{line}<em>{34} : \text{line}</em>{13} )</td>
</tr>
<tr>
<td>Gold - Face side : Nose width : Face side</td>
<td>( \text{line}<em>1 : \text{line}</em>{39} : \text{line}_{13} )</td>
</tr>
<tr>
<td>White - Face side : Eye outside : Eye inside</td>
<td>( \text{line}<em>1 : \text{line}</em>{27} : \text{line}_{29} )</td>
</tr>
<tr>
<td>Blue - Eye outside : Eye inside : Nose center</td>
<td>( \text{line}<em>{27} : \text{line}</em>{29} : \text{line}_{67} )</td>
</tr>
<tr>
<td>Green - Mouth outside : Lip cupids bow: Mouth outside</td>
<td>( \text{line}<em>{48} : \text{line}</em>{50} : \text{line}<em>{52} : \text{line}</em>{54} )</td>
</tr>
</tbody>
</table>

Table 3.1. The correlation table of Horizontal Golden Ratio.
The line means the line through the point $X$ with slope $k$. Slope $k$ is the slope of the midline of the face. Then the ratio is calculated by the distance of two lines divided by the other distance of two lines. Theoretically the midline should be through points 7, 41, 51, 57, 61, 64, 66, and 67, so any two points of them should determine the midline. However, the result is not ideal, as Figure 3.13 (a) shows. Therefore, the aggression algorithm is used to find the midline determined by points 41, 51, 57, 61, 64, 66, and 67. The result is showed in Figure 3.13 (b). The lines are drawn on the bitmap by the color defined in the correlation table. Point 7 was disposed because after comparing the x-coordinates of those eight points on different images, the x-coordinates of point 7 have a difference which can cause an unusual result.

To calculate the Vertical Golden Ratio, the following correlation table should be obeyed.

<table>
<thead>
<tr>
<th>The Vertical Golden Ratio</th>
<th>The lines (slope = $k'$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>White - Hairline: Eyebrow top : Eye top</td>
<td>(undefined)</td>
</tr>
<tr>
<td>Gold - Eyebrow top : Eyebrow bottom : Eye</td>
<td>$line_{22}/23/24 : line_{26}/25/24 : line_{28} : line_{30}$</td>
</tr>
<tr>
<td>top : Eye bottom</td>
<td></td>
</tr>
<tr>
<td>Blue - Eye pupil : Nose flair : Nose bottom</td>
<td>$line_{31} : line_{67} : line_{41}$</td>
</tr>
<tr>
<td>Green - Eye pupil : Nose bottom : Mouth chin</td>
<td>$line_{31} : line_{41} : line_{66}$</td>
</tr>
<tr>
<td>Green - Eye pupil : Mouth : Chin</td>
<td>$line_{31} : line_{66} : line_{7}$</td>
</tr>
</tbody>
</table>

Table 3.2. The correlation table of Vertical Golden Ratio

The slope $k'$ is the slope of horizontal line of the face. The horizontal line is determined by points 31 and 36, which are the center of two pupils. Since the hairline
is easy to be changed by hair style, so the ratio of hair line : eyebrow top : eye top is disposed. The eyebrow top is determined by comparing the distance of points 22, 23, and 24 to point 31. The eyebrow bottom is determined by comparing the distance of points 26, 25, and 24 to point 31. The lines are drawn on the bitmap by the colors defined in the correlation table. The grades and the suggestions based on the result ratio are stored in a local address in .txt format. The Camera activity passes the addresses of modified photos and the .txt file to the Result activity.
3.4 The Gallery activity

The Gallery activity helps users access the local pictures. It is the other one of two main activities. Two .xml files and two classes are associated with gallery activity: activity_gallery.xml, fragment_camera.xml, the GalleryFragment class, and the Face-Analysis class. The design of the Gallery activity can be divided into four parts: the navigation drawer, the Google Gallery, the facial feature point detection, and the Golden Ratio analysis. The Gallery activity is the same as the Camera activity except the Google Gallery takes the place of the camera.

3.5 The Result Activity

The UI of Result activity is similar to the Start activity: a list view on the left and an associated fragment on the right of the screen. Therefore the design is similar. The host activity Result is associated with two Fragment classes: the optionsFragment and the resultFragment. As the same mentioned above, a frame container activity_show_result.xml in layout folder defines the activity view for small screens while the two-pane layout activity_show_result.xml in large-layout folder for tablet screens (Figure 3.14). The layout for handsets has not be tested due to lack of the suitable devices. The resultFragment display the processed images and the result .txt file.
Your symmetry score is 80.
You need a pair of longer eyes.
You need a pair of long eyebrows.
You need a narrower nose.
The length of your mouth is perfect.

Fig. 3.14. The result activity shows on a tablet.
4. RESULTS

All the results shown in this chapter were generated using a Nexus 7 with API 4.0.3.

To test the accuracy of the face analysis, five aspects were tested: first, if the App would consider animal faces as human faces; second, if the App can detect more than one face in a photo; third, the result of beautiful faces analysis; fourth, the result of ordinary faces analysis; fifth, the result of non-ideal faces analysis. To test, five samples were chosen for each group.

4.1 The Animal Group

Five animals (cat, dog, rabbit, gorilla, and monkey) were tested as Figure 4.15 shows. No face was detected in the photos of cat, dog, rabbit, or gorilla. However, the App miss recognizes the monkey face as a human face. This is because the monkey has a similar face structure to that of a human being (Figure 4.16). Therefore it is possible that some objects with the similar structure could be recognized as human faces.

Fig. 4.15. The animal faces.
4.2 The Multiple Face Group

Five images with more than one human faces was tested. The App can detect the human faces on every image, but only one. The Figure 4.17 shows the five chosen images. The result can be seen in Figure 4.18.
4.3 The Beautiful Human Group

Five beautiful humans (Caucasian, Indian, Asian, Latino, and African) were tested as Figure 4.19. The ratio result is showed in Table 4.3. The symmetry grades of them
are 99, 100, 97, 98, and 99. The Average stands for the average of the absolute value of each ratio minus the Golden Ratio.

![Beautiful faces](image)

**Fig. 4.19.** The sample of the beautiful faces.

<table>
<thead>
<tr>
<th></th>
<th>Caucasian</th>
<th>India</th>
<th>Asian</th>
<th>Latino</th>
<th>Africa</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face side : Eye-brows : Face side</td>
<td>1.579</td>
<td>1.652</td>
<td>1.54</td>
<td>1.681</td>
<td>1.597</td>
<td>1.610</td>
</tr>
<tr>
<td>Face side : Eye inside : Face side</td>
<td>1.6935</td>
<td>1.747</td>
<td>1.605</td>
<td>1.725</td>
<td>1.693</td>
<td>1.693</td>
</tr>
<tr>
<td>Face side : Nose width : Face side</td>
<td>1.703</td>
<td>1.737</td>
<td>1.592</td>
<td>1.736</td>
<td>1.745</td>
<td>1.703</td>
</tr>
<tr>
<td>Face side : Eye outside : Nose center</td>
<td>1.51</td>
<td>1.341</td>
<td>1.499</td>
<td>1.346</td>
<td>1.424</td>
<td>1.424</td>
</tr>
<tr>
<td>Eye outside : Eye inside : Nose center</td>
<td>1.705</td>
<td>1.692</td>
<td>1.707</td>
<td>1.712</td>
<td>1.573</td>
<td>1.6778</td>
</tr>
<tr>
<td>Mouth outside : Lip cupids bow : Mouth outside</td>
<td>1.610</td>
<td>1.681</td>
<td>1.652</td>
<td>1.705</td>
<td>1.625</td>
<td>1.655</td>
</tr>
<tr>
<td></td>
<td>1.327</td>
<td>1.378</td>
<td>1.401</td>
<td>1.621</td>
<td>1.444</td>
<td>1.434</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Eye pupil : Nose flair : Nose bottom</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eyebrow top : Eye top : Eye bottom</td>
<td>1.507</td>
<td>1.537</td>
<td>1.457</td>
<td>1.237</td>
<td>1.449</td>
<td>1.437</td>
</tr>
<tr>
<td>Eyebrow top : Eyebrowbottom : Eye top</td>
<td>1.591</td>
<td>1.62</td>
<td>1.45</td>
<td>1.514</td>
<td>1.575</td>
<td>1.550</td>
</tr>
<tr>
<td>Eye pupil : Nose bottom : Mouth : chin</td>
<td>1.85</td>
<td>2.146</td>
<td>1.918</td>
<td>1.537</td>
<td>1.430</td>
<td>1.776</td>
</tr>
<tr>
<td>Eye pupil : Mouth : Chin</td>
<td>1.759</td>
<td>1.437</td>
<td>1.774</td>
<td>1.544</td>
<td>1.638</td>
<td>1.630</td>
</tr>
<tr>
<td>Average</td>
<td>0.109</td>
<td>0.157</td>
<td>0.124</td>
<td>0.126</td>
<td>0.097</td>
<td>0.096</td>
</tr>
</tbody>
</table>

Table 4.3.: The sample of beautiful human

### 4.4 The Ordinary Human Group

Five ordinary humans, the same races as the samples of the beautiful group, were tested as Figure 4.20 showed. The ratio result is showed in Table 4.4. The symmetry grades of them are 97, 92, 98, 97, and 97.
Fig. 4.20. The sample of the normal faces.

<table>
<thead>
<tr>
<th></th>
<th>Caucasian</th>
<th>India</th>
<th>Asian</th>
<th>Latino</th>
<th>Africa</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face side : Eye-brows : Face side</td>
<td>1.553</td>
<td>1.506</td>
<td>1.556</td>
<td>1.57</td>
<td>1.597</td>
<td>1.556</td>
</tr>
<tr>
<td>Face side : Eye inside : Face side</td>
<td>1.688</td>
<td>1.754</td>
<td>1.602</td>
<td>1.725</td>
<td>1.693</td>
<td>1.692</td>
</tr>
<tr>
<td>Face side : Nose width : Face side</td>
<td>1.649</td>
<td>1.744</td>
<td>1.785</td>
<td>1.736</td>
<td>1.745</td>
<td>1.732</td>
</tr>
<tr>
<td>Face side : Eye outside : Nose center</td>
<td>1.444</td>
<td>1.577</td>
<td>1.62</td>
<td>1.346</td>
<td>1.424</td>
<td>1.482</td>
</tr>
<tr>
<td>Eye outside : Eye inside : Nose center</td>
<td>1.751</td>
<td>1.336</td>
<td>1.483</td>
<td>1.712</td>
<td>1.573</td>
<td>1.571</td>
</tr>
<tr>
<td>Mouth outside : Lip cupids bow : Mouth outside</td>
<td>1.576</td>
<td>1.664</td>
<td>1.75</td>
<td>1.705</td>
<td>1.625</td>
<td>1.664</td>
</tr>
<tr>
<td>Eye pupil : Nose flair : Nose bottom</td>
<td>1.442</td>
<td>1.301</td>
<td>1.489</td>
<td>1.621</td>
<td>1.444</td>
<td>1.459</td>
</tr>
<tr>
<td></td>
<td>1.532</td>
<td>1.501</td>
<td>1.457</td>
<td>1.237</td>
<td>1.449</td>
<td>1.435</td>
</tr>
<tr>
<td>------------------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Eyebrow top :</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eye top : Eye</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>top : Eye bottom</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eyebrow top :</td>
<td>1.567</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eyebrow bottom :</td>
<td></td>
<td>1.706</td>
<td>1.418</td>
<td>1.514</td>
<td>1.575</td>
<td>1.556</td>
</tr>
<tr>
<td>Eye top</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eye pupil : Nose</td>
<td>1.859</td>
<td>1.618</td>
<td>1.457</td>
<td>1.537</td>
<td>1.43</td>
<td>1.58</td>
</tr>
<tr>
<td>bottom : Mouth :</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eye pupil :</td>
<td>1.667</td>
<td>1.649</td>
<td>1.593</td>
<td>1.544</td>
<td>1.638</td>
<td>1.618</td>
</tr>
<tr>
<td>Mouth : Chin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>0.102</td>
<td>0.118</td>
<td>0.108</td>
<td>0.124</td>
<td>0.097</td>
<td>0.110</td>
</tr>
</tbody>
</table>

Table 4.4.: The sample of ordinary human.

### 4.5 The Non-Ideal Human Group

The ratio result of the non-ideal human group is showed in Table 4.5. The symmetry grade of them are 92, 91, 95, 94, and null. The sample of African was not recognized as a human face.

![Fig. 4.21. The sample of the beautiful faces.](image)
<table>
<thead>
<tr>
<th>Location</th>
<th>Caucasian</th>
<th>India</th>
<th>Asian</th>
<th>Latino</th>
<th>Africa</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face side : Eyebrows : Face side</td>
<td>1.915</td>
<td>1.560</td>
<td>1.461</td>
<td>1.831</td>
<td>null</td>
<td>1.692</td>
</tr>
<tr>
<td>Face side : Eye inside : Face side</td>
<td>1.748</td>
<td>1.838</td>
<td>1.960</td>
<td>1.978</td>
<td>null</td>
<td>1.881</td>
</tr>
<tr>
<td>Face side : Nose width : Face side</td>
<td>1.352</td>
<td>1.442</td>
<td>2.116</td>
<td>2.104</td>
<td>null</td>
<td>1.754</td>
</tr>
<tr>
<td>Face side : Eye outside : Nose center</td>
<td>1.778</td>
<td>1.411</td>
<td>1.442</td>
<td>1.331</td>
<td>null</td>
<td>1.491</td>
</tr>
<tr>
<td>Eye outside : Eye inside : Nose center</td>
<td>2.076</td>
<td>1.917</td>
<td>1.399</td>
<td>1.509</td>
<td>null</td>
<td>1.725</td>
</tr>
<tr>
<td>Mouth outside : Lip cupids bow : Mouth outside</td>
<td>1.196</td>
<td>1.832</td>
<td>1.748</td>
<td>1.681</td>
<td>null</td>
<td>1.614</td>
</tr>
<tr>
<td>Eye pupil : Nose flair : Nose bottom</td>
<td>1.573</td>
<td>1.436</td>
<td>1.368</td>
<td>1.417</td>
<td>null</td>
<td>1.462</td>
</tr>
<tr>
<td>Eyebrow top : Eye top : Eye bottom</td>
<td>1.294</td>
<td>1.567</td>
<td>1.480</td>
<td>1.653</td>
<td>null</td>
<td>1.499</td>
</tr>
<tr>
<td>Eyebrow top : Eyebrow bottom : Eye top</td>
<td>1.478</td>
<td>1.630</td>
<td>1.560</td>
<td>1.815</td>
<td>null</td>
<td>1.621</td>
</tr>
</tbody>
</table>
4.6 Analysis of Three Groups

There is a small difference between the ordinary human group and the beautiful human group. The facial structure is not the determined factor influenced their beauty difference. The result of the non-ideal human group has a significant difference from the other groups. Each face of the non-ideal human group has at least one ratio which has a big residual to the Golden Ratio. The total average of the non-ideal human group is 0.227 comparing with 0.096 and 0.110 which are the results of the beautiful group and the ordinary group. The Figure 4.22 shows the comparison chart of the three groups.
Fig. 4.22. The sample of the normal faces.
5. CONCLUSION

The App FaceAnalysis achieved its design goals. The UI also can fit for different size screens. The navigation drawer can help users go to their desired activity easily. Though the test the App can detect human face and the facial feature points. The test result is not perfect but reasonable. The limitations and the future works of the App are discussed in this chapter.

5.1 Limitations of The FaceAnalysis

The limitations of the App are divided into three types: the algorithm limitations, the hardware limitations and the design limitations. The algorithm limitations are caused by the face detection algorithm, the ASM algorithm, and the training datasets. Due to the face detection algorithm, the App only detects one face if there are more than one face in the picture. Because of the ASM algorithm, the facial feature point detection depends on the pose of human face, the face expression, and the lighting condition. The incorrect recognition of the monkey and the non-ideal African is caused by the face detection algorithms and the training datasets.

Because only one Android device is used for developing, the App functions are limited by the device. Users only can access one camera. When the users take photo, they cannot switch cameras if there are two cameras. The screen size adapting UI cannot be tested on handsets. Some of the limitations can be fixed. The design limitations affects the functionality of UI. The users cannot delete pictures from the Gallery activity. The App cannot detect a face from a rotated images. The App does not support all types of images.
5.2 Future Work

There are several ways to update current activities. The Camera activity can be designed to support camera switches. The App can be designed to detect all the faces in an image. The custom grid view can be designed to replace the Google Gallery and the Gallery activity. To have the similar layout with Google Gallery, the custom grid view needs to use hash map to store the tag of images and their parent folder name. The most difficult part of designing a custom grid view is to handle the App memory. Because the grid view has to display lots of thumbnails of images, the memory runs out easily. There are three ways to solve that problem: first, loading bitmaps in a thread or an Asynctask; second, using disk memory to associate the memory; third, loading the bitmaps dynamically. The custom grid view will add the delete option to help users to delete their unwanted photos.

A new activity Review will be added to display the previous results to users by implementing the fragments nested into a fragment. The UI will have a better navigation. Because the fragment nested into fragment is only supported by the API 4.2 and above, to support more API versions the Result activity does not have a navigation drawer currently. A Help activity can be designed for the App. It will only show once at the first time users using the App. It can contain the modified graphics with instructions to give users a better guiding experience. For the test, the sample should be chosen by each photo for one gender and one race. The babies face should be tested either. The App should be test on more devices. The App can be tested by the Android JUI in the future.
BIBLIOGRAPHY AND REFERENCES


APPENDIX A – CODE FOR THE PROJECT

The following is parts of the java code for the Camera activity.

```java
package org.asmlibrary.fit;

public class Camera extends FragmentActivity{
    private static final String TAG = "TakePhoto";
    public Camera() {
        Log.i(TAG, "Instantiated new " + this.getClass());
    }

    private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
        private File getSourceFile(int id, String name, String folder){
            File file = null;
            try {
                InputStream is = getResources().openRawResource(id);
                File cascadeDir = getDir(folder, Context.MODE_PRIVATE);
                file = new File(cascadeDir, name);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
            return file;
        }
    }
```
FileOutputStream os = new FileOutputStream(file);

byte[] buffer = new byte[4096];
int bytesRead;
while ((bytesRead = is.read(buffer)) != -1) {
    os.write(buffer, 0, bytesRead);
}
is.close();
os.close();
}catch (IOException e) {
    e.printStackTrace();
    Log.e(TAG, "Failed to load file " + name + ".
    Exception thrown: " + e);
}
return file;
}

@Override
public void onManagerConnected(int status) {
    switch (status) {
    case LoaderCallbackInterface.SUCCESS:
    {
        Log.i(TAG, "OpenCV loaded successfully");
    }
//mRgba = Highgui.imread("storage/emulated/0/image.jpg");

// Load native library after (!) OpenCV initialization
System.loadLibrary("asmlibrary");
System.loadLibrary("jni-asmlibrary");

mASMFit = new ASMFit();

mModelFile = getSourceFile(R.raw.my68_1d, "my68_1d.amf", "model");
if (mModelFile != null)
    mASMFit.nativeReadModel(mModelFile.getAbsolutePath());

mCascadeFile = getSourceFile(R.raw.
    haarcascade_frontalface_alt2,
    "haarcascade_frontalface_alt2.xml", "cascade");
if (mCascadeFile != null)
    mASMFit.nativeInitCascadeDetector(
        mCascadeFile.getAbsolutePath());
mFastCascadeFile = getSourceFile(R.raw.
  lbpcascade_frontalface,
  "lbpcascade_frontalface.xml",
  "cascade");
if (mFastCascadeFile != null)
  mASMFt.nativeInitFastCascadeDetector
  (mFastCascadeFile.getAbsolutePath()
   ());
} break;
default:
{
    super.onManagerConnected(status);
} break;
}
}@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_take_photo);
  mTitle = mDrawerTitle = getTitle();
  mPlanetTitles = getResources().getStringArray(R.array
    .option_array);
mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
mDrawerList = (ListView) findViewById(R.id.left_drawer);

// mSurface = (MySurfaceView) findViewById(R.id.mysurfaceview);

// set a custom shadow that overlays the main content when the drawer opens
mDrawerLayout.setDrawerShadow(R.drawable.drawer_shadow, GravityCompat.START);

// set up the drawer’s list view with items and click listener
mDrawerList.setAdapter(new ArrayAdapter<String>(this, R.layout.drawer_list_item, mPlanetTitles));
mDrawerList.setOnItemClickListener(new DrawerItemClickListener());

// enable ActionBar app icon to behave as action to toggle nav drawer
getActionBar().setDisplayHomeAsUpEnabled(true);
getActionBar().setHomeButtonEnabled(true);

// ActionBarDrawerToggle ties together the the proper interactions
// between the sliding drawer and the action bar app
icon
mDrawerToggle = new ActionBarDrawerToggle(
    this,  // host Activity */
mDrawerLayout,  // DrawerLayout object */
R.drawable.ic_drawer,  // nav drawer image to replace 'Up' caret */
R.string.drawer_open,  // "open drawer"
    description for accessibility */
R.string.drawer_close /* "close drawer"
    description for accessibility */
) {
    public void onDrawerClosed(View view) {
        getActionBar().setTitle(mTitle);
        invalidateOptionsMenu(); // creates call to
        onPrepareOptionsMenu()
    }

    public void onDrawerOpened(View drawerView) {
        getActionBar().setTitle(mDrawerTitle);
        invalidateOptionsMenu(); // creates call to
        onPrepareOptionsMenu()
    }
};
mDrawerLayout.setDrawerListener(mDrawerToggle);
fragment = new CameraFragment();
fragment.setArguments(getIntent().getExtras());

FragmentManager fragmentManager = getFragmentManager();
fragmentManager.beginTransaction().replace(R.id.content_frame, fragment).commit();
Log.i(TAG, "after commit?");
myDataSource = new ResultDataSource(this);
myDataSource.open();
dbHelper = new MySQLiteHelper(this);
database = dbHelper.getReadableDatabase();
}

@override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.main, menu);
    return super.onCreateOptionsMenu(menu);
}

/*@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    /* Called whenever we call invalidateOptionsMenu() */
    /*
    @Override
    public boolean onPrepareOptionsMenu(Menu menu) {
    */

46
/ If the nav drawer is open, hide action items related to the content view

boolean drawerOpen = mDrawerLayout.isDrawerOpen(mDrawerList);

menu.findItem(R.id.action_hint).setVisible(!drawerOpen);

menu.findItem(R.id.action_analysis).setVisible(!drawerOpen);

return super.onPrepareOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // The action bar home/up action should open or close the drawer.
    // ActionBarDrawerToggle will take care of this.
    if (mDrawerToggle.onOptionsItemSelected(item)) {
        return true;
    }

    // Handle action buttons
    switch(item.getItemId()) {
    case R.id.action_hint:
        AlertDialog.Builder builder2 = new Builder(
            Camera.this);
        break;
    }
builder2.setIcon(R.drawable.icon);
builder2.setTitle("Hint");
builder2.setMessage("Click the screen to take a photo.\nClick the icon to open navigation drawer.\nClick FACE ANALYSIS on the top right corner to process your photo.");
builder2.setPositiveButton("OK", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton) {
    }
});
builder2.create().show();
return true;
case R.id.action_analysis:
    // get the view of the fragment and get the path of the photo
    mSurface = (MySurfaceView) fragment.getView();
    mPath = mSurface.getPath();
    Log.i(TAG, "item path:"+mPath);
    Log.i(TAG, "item copy path:"+mPathCopy);
if ((mPath == null) || (mPathCopy == mPath)) {
    AlertDialog.Builder builder1 = new
        Builder(Camera.this);
    builder1.setIcon(R.drawable.icon);
    builder1.setTitle("Face Detection");
    builder1.setMessage("You need to take a photo.");
    builder1.setPositiveButton("OK", new
        DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {
                mSurface.overTack();
                // isClicked = false;
            }
        });
    builder1.create().show();
} else {
    mPathCopy = mPath;
    Log.i(TAG, "copy path:" + mPathCopy);
    mPath = getPath(Uri.parse(mPath));
    AnalysisProgressTask task = new
        AnalysisProgressTask();
    task.execute(mPath);
}
return true;
default:
    return super.onOptionsItemSelected(item);
}

/* The click listener for ListView in the navigation
drawer */
private class DrawerItemClickListener implements AdapterView.OnItemClickListener {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Log.i(TAG, "any click? position="+position);
        selectItem(position);
    }
}

private void selectItem(int position) {
    // update the main content by replacing fragments
    switch (position) {
    case 0:
        Toast.makeText(Camera.this, "You are using Camera now.", Toast.LENGTH_LONG).show();
        break;
    }
case 1:

    Intent intent1 = new Intent();
    intent1.putExtra("takePhoto", 0);
    intent1.setClass(Camera.this, Gallery.class);
    startActivity(intent1);
    break;

case 2:

    AlertDialog.Builder builder2 = new Builder(
            Camera.this);
    builder2.setIcon(R.drawable.icon);
    builder2.setTitle("Help");
    builder2.setMessage(R.string.help);
    builder2.setPositiveButton("OK", new
            DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {
            }
        });
    builder2.create().show();
    break;

case 3:

    AlertDialog.Builder builder3 = new Builder(
            Camera.this);
    builder3.setIcon(R.drawable.icon);
    builder3.setTitle("About");
builder3.setMessage("Developer: Shuyi Zhao\n" + "Instructors: Dr. Scott A. King\n" + "Dr. Dulal Kar\n\n" + "Texas A&M University–Corpus Christi");

builder3.setPositiveButton("OK", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton) {
    }
});

builder3.create().show();
break;
}

mDrawerLayout.closeDrawer(mDrawerList);

/**
 * When using the ActionBarDrawerToggle, you must call it

during
* onPostCreate() and onConfigurationChanged() ...
*/

@Override
protected void onPostCreate(Bundle savedInstanceState) {
    super.onPostCreate(savedInstanceState);
    // Sync the toggle state after onRestoreInstanceState has occurred.
    mDrawerToggle.syncState();
}

@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    // Pass any configuration change to the drawer toggls
    mDrawerToggle.onConfigurationChanged(newConfig);
}

@Override
public void onPause()
{
    mPath = null;
    myDataSource.close();
super.onPause();
}

@Override
public void onResume()
{
    myDataSource.open();
super.onResume();
    OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_5, this, mLoaderCallback);
mPath = null;
mFlag = false;
}

public String getPath(Uri uri) {
    String[] projection = { MediaStore.Images.Media.DATA };
    Cursor cursor = getContentResolver().query(uri, projection, null, null, null);
    // startManagingCursor(cursor);
    int column_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
    cursor.moveToFirst();
    return cursor.getString(column_index);
public boolean imgProcess(String path){
    BitmapFactory.Options BitmapFactoryOptionsbfo = new BitmapFactory.Options();
    BitmapFactoryOptionsbfo.inPreferredConfig = Bitmap.Config.RGB_565;
    // myBitmap = BitmapFactory.decodeResource(
    getResources(),
    // R.raw.florencecolgate,
    BitmapFactoryOptionsbfo);
    myBitmap = BitmapFactory.decodeFile(path,
    BitmapFactoryOptionsbfo);
    mRgba = new Mat(myBitmap.getWidth(),myBitmap.
    getHeight(), CvType.CV_8UC1);
    Utilities.bitmapToMat(myBitmap, mRgba);
    mGray = new Mat(myBitmap.getWidth(),myBitmap.
    getHeight(), CvType.CV_8UC1);
}
I mg proc. cvtColor (mRgba, mGray, Imgproc.
COLOR_RGB2GRAY);
Imgproc.equalizeHist (mGray, mGray);

// mirror the pic
mRgba2 = new Mat (myBitmap.getWidth (), myBitmap
    .getHeight (), CvType.CV_8UC1);
Core.flip (mRgba, mRgba2, 1);
mGray2 = new Mat (myBitmap.getWidth (), myBitmap
    .getHeight (), CvType.CV_8UC1);
Imgproc.cvtColor (mRgba2, mGray2, Imgproc.
COLOR_RGB2GRAY);
Imgproc.equalizeHist (mGray2, mGray2);

width = new ArrayList<Double>();
height = new ArrayList<Double>();
width2 = new ArrayList<Double>();
height2 = new ArrayList<Double>();

Mat detShape = new Mat();
Mat detShape2 = new Mat();
if (mFastDetect){
    mFlag = mASMFit.fastDetectAll
        (mGray, detShape);
    Log.i(TAG," face detect");
mASMFit2.fastDetectAll(mGray2, detShape2);
Log.i(TAG," second face detect ");
}
else{
    mFlag = mASMFit.detectAll(
        mGray, detShape);
    mASMFit.detectAll(mGray2,
        detShape2);
    Log.i(TAG," flag : "+mFlag);
}
if(mFlag){
    mShape = detShape.row(0);
    mShape2 = detShape2.row(0);
}else{
    return false;
}
//

if(mFlag)
{
    //
    mASMFit.fitting(mGray, mShape);
    mASMFit.fitting(mGray2, mShape2);
}
if (mFlag)
{
    int nPoints = mShape.row(0).cols() / 2;
    for (int i = 0; i < nPoints; i++)
    {
        double x = mShape.get(0, 2*i)[0];
        double y = mShape.get(0, 2*i+1)[0];
        double x2 = mShape2.get(0, 2*i)[0];
        double y2 = mShape2.get(0, 2*i+1)[0];
        width.add(i, x);
        height.add(i, y);
        width2.add(i, x2);
        height2.add(i, y2);
        Point pt = new Point(x, y);
    }
}
Core.circle(mRgba, pt, 3, mColor);

}
}

return true;

// Log.i(TAG, "processed photo: " + mPathCopy);
}//end of imgProcess

// An asnyTask to do analysis job
class AnalysisProgressTask extends AsyncTask<String, Integer, Boolean> {

ProgressDialog dialog = new ProgressDialog(Camera.
this);

@Override
protected Boolean doInBackground(String... params)
{
try {

if (!imgProcess(mPath)) {
    return Boolean.FALSE;
} else {
    myFaceAnalysis = new
FaceAnalysis(mPath, width, height, width2, height2);
myFaceAnalysis.analysis();
Log.i(TAG," before  add");
add();
File f = new File(mPath);
String name = f.getName();
Log.i(TAG, " FILE PATH: " + mPath);
Log.i(TAG," FILE NAME:" +name);
}
}
catch (Exception e) {
    e.printStackTrace();
}
return Boolean.TRUE; // Return your real result here
}
@Override
protected void onPreExecute() {
    this.dialog.setMessage("Analyzing...");
    this.dialog.show();
}
@Override
protected void onPostExecute(Boolean result) {

// result is the value returned from
doInBackground
dialog.dismiss();
if (result == Boolean.TRUE) {
    Intent intent = new Intent(Camera.this, Result.class);
    intent.putExtra("path", mPath);
    startActivity(intent);
} else {
    AlertDialog.Builder builder1 = new Builder(
        Camera.this);
    builder1.setIcon(R.drawable.icon);
    builder1.setTitle("Face Detection");
    builder1.setMessage("No human face detected
or there are more than.");
    builder1.setPositiveButton("OK", new
        DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            mSurface veroTack();
            mPath = null;
        }
    });
    builder1.create().show();
}
public void add() {
    ArrayList<AnalysisResult> results = new ArrayList<AnalysisResult>();
    Log.i(TAG, "IN ADD");
    File f = new File(mPath);
    String name = f.getName();
    Log.i(TAG, "NAME IN ADD:" + name);
    AnalysisResult result1 = new AnalysisResult(
        name, myFaceAnalysis.getSymGrade(),
        myFaceAnalysis.getHorEyebrow(),
        myFaceAnalysis.getVerEyebrow(),
        myFaceAnalysis.getHorEye(),
        myFaceAnalysis.getVerEye(),
        myFaceAnalysis.getLip());

    results.add(result1);
    Log.i(TAG, "result add after");
    myDataSource.add(results);
    String tmp = this.getDatabasePath("
analysis_result.db).toString();

Log.i(TAG,"DATABASE PATH:" + tmp);

File dbFile = this.getDatabasePath("result");

Boolean check = dbFile.exists();

Log.i(TAG, "exist:"+check);

\end{onespace}