XFace: A Face Recognition System for Android Mobile Phones

Jiawei Hu
Dept. of Computer Science and Technology,
Tsinghua University
Beijing, China
hjw13@mails.tsinghua.edu.cn

Liangrui Peng
Tsinghua National Laboratory for Information Science and Technology,
Dept. of Electronic Engineering,
Tsinghua University
Beijing, China
penglr@tsinghua.edu.cn

Li Zheng
Dept. of Computer Science and Technology,
Tsinghua University
Beijing, China
zhengli@tsinghua.edu.cn

***Abstract*—With the rapid development of mobile computing, biometric identification technologies including face recognition on smart phones become increasingly attractive. There are two main challenges of face recognition for mobile computing: one is the problem of the variations of face images acquired in unconstrained environment; the other is the computing limitation of mobile phone platform. This paper presents the design and implementation of an open source face recognition system named XFace for Android mobile phone platform, which provides a feasible solution to meet these challenges. The proposed face recognition approach includes face detection, eye detection, preprocessing for ROI (Region of Interest), LBP (Local Binary Pattern) feature extraction, feature dimensionality reduction based on PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis), and minimum distance classifier. The implementation of the proposed approach is mainly based on OpenCV (Open Source Computer Vision) SDK. The system framework and user interface are also presented. Experimental results on practical face image samples collected on Android mobile phones proved the effectiveness of the proposed method. In future work, it is possible to integrate deep learning based face recognition method to the XFace system.**

Keywords-component; Face Recognition; Face Detection; Eigenfaces; Fisherfaces; Android Application

I. INTRODUCTION

Biometric identification technology, especially human face recognition, has been a very popular research topic in the last decade. With the rapid development of smart phones, it becomes an attractive topic to implement face recognition system on mobile phone platforms such as Android. There are mainly two challenges of face recognition for mobile computing: one is the problem of the variations of face images acquired in unconstrained environment; the other is the computing limitation of mobile phone platform.

The face images are often acquired under a certain environment in the traditional way of implementing a face recognition system, but things are different and more complicated when implementing a face recognition system on mobile phone platform. The current face recognition techniques are much less reliable when used in real-world conditions for the fact that current face recognition techniques are very sensitive to exact conditions in the images, such as the direction of lighting and shadows, exact orientation of the face, expression of the face, and the current mood of the person [5].

On the other hand, as a mobile operating system, Android has reached great success in smartphones and tablets recently. When compared with traditional mobile phones, Android smart phones have improved greatly in computing ability and storage capacity. New versions of Android system have become more and more powerful by integrating functions like image capture and image processing. In June 2009, Google released Android Native Development Kit (Android NDK [1]) to support native development in C/C++, besides the Android Software Development Kit (Android SDK) that supports Java. Android NDK allows the developers to develop Android applications using C/C++ and transplant C/C++ libraries like OpenCV [2] to Android platform [10]. Though it has made great improvements in recent years, Android still remains the problem of computing limitation when compared with the PC platform. So we have to make a tradeoff between the computation complexity and the processing efficiency when implementing this system.

This paper presents a prototype face recognition system that provides a feasible solution to meet the challenges and achieves a satisfactory face detecting and recognizing performance on Android mobile phones. For the face detection, we used the LBP-based frontal face detector, and an effective way of ROI preprocessing method has been applied to overcome the problem of the variations in face images. Eigenfaces and Fisherfaces are both employed and tested for face recognition.

This document is organized as follows: in Section II, the methodologies we applied in this face recognition system are discussed. Section III illustrates the design and implementation of the face recognition system. Section IV shows the results obtained from the experiments and the analysis performed. Finally, conclusions and future work are presented in Section V.

II. METHODOLOGY

Face recognition procedure basically consists of two steps. The first step is the face detection, which rapidly finds a human face in the photo or video when the human is located within a proper distance by the smart phone. The second step is the face recognition that is recognizing the

face as someone known according to the face database collected before. In the system we implemented, another step of the ROI preprocessing is performed between face detection and face recognition for the fact that the accuracy of the face recognition system by using the ROI preprocessing method increased significantly.

A. Face Detection

Face detection is the process of locating one or more face regions in an image. There are two common types of face detector: one is Haar-based face detector; the other is LBP-based face detector. We have studied these two face detection algorithms and evaluated the tradeoff between detection accuracy and computational complexity, as the face recognition system has to be implemented on mobile phones. In this prototype system, we used the LBP-based frontal face detector as the face detector. Here we reproduce a brief introduction of LBP, while referring the reader to the relevant reference.

Local Binary Pattern was invented by Ahonen, Hadid and Pietikäinen in 2004 [4]. The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighborhood. A formal description of the LBP operator can be given as:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(v_p - v_c), s(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \quad (1)$$

Where (x_c, y_c) denotes the central pixel in the local image with intensity v_c, v_p is the intensity of the neighbor pixel. P is the total number of involved neighbor pixels.

B. Eye detection and ROI preprocessing

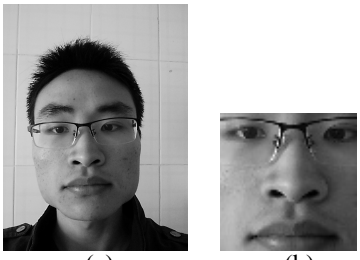
As face recognition is extremely vulnerable to the changes of the nearby circumstances, we have performed an experiment and evaluated the performances of the system with and without the ROI preprocessing method (this method will be discussed later), the detailed results presented in Section IV show that, with ROI preprocessing method the accuracy of the face recognition system increased significantly.

For the reliability in real-world conditions, the ROI preprocessing method should apply many other sophisticated techniques including facial feature detection (for example, eyes detection, nose detection, and mouth detection). Among these facial feature detections, eyes detection can be very helpful for ROI preprocessing, because for frontal faces we can always assume a person's eyes should be horizontal and on opposite locations of the face and should have a fairly standard position and size within a face, despite changes in facial expressions, lighting conditions, camera properties, distance to the camera, and so on [5].

With both face and eyes detected, we then perform a preprocessing method proposed by Shervin Emami in [5]. The whole preprocessing procedure combines the following four steps, among which, the first step is to get the Region of Interest:

1) Geometrical transformation and cropping: This process would include scaling, rotating, and translating the input face image so that the eyes are aligned. The distance between the eyes is used to normalize the image [9]. Then the ROI image of N*N pixels is extracted according to the position of the eyes. In our system, N is set to 70 as an empirical value.

Figure 1 shows a grayscale face image acquired from the mobile phone and its corresponding ROI image obtained by the first step.



(a) (b)

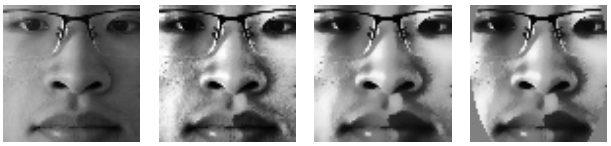
Figure 1. (a) A grayscale face image acquired from the mobile phone; (b) The corresponding ROI image obtained by the first step.

2) Separate histogram equalization for left and right sides: This process standardizes the brightness and contrast on both the left- and right-hand sides of the face independently.

3) Smoothing: This process reduces the image noise using a bilateral filter which is very good at smoothing most of an image while keeping edges sharp.

4) Elliptical mask: The elliptical mask removes some remaining hair and background from the face image.

The face images before and after the ROI preprocessing are shown in Figure 2.



(a) (b) (c) (d)

Figure 2. The four steps in preprocessing a face image: (a) Geometrical transformation and cropping; (b) Separate histogram equalization for left and right sides; (c) Smoothing; (d) Elliptical mask.

C. Face Recognition

As two of the commonly discussed face recognition algorithms, Eigenfaces and Fisherfaces are both employed and tested in this work.

1) Eigenfaces

Eigenfaces, also known as PCA, first used by Turk and Pentland in 1991 [6]. PCA is a commonly used technology for dimensionality reduction in computer vision (particularly in face recognition). It chooses a dimensionality reducing linear projection that maximizes the scatter of all projected samples.

To create a set of eigenfaces, we need a training set of n face images. Each image is treated as one column vector, simply by concatenating the columns of pixels in the original image, resulting in a single column with r x c elements. Then we could turn the training set into a big matrix as

$$X = [\chi_1, \chi_2, \cdots \chi_n] \quad (2)$$

where χ_i is a column vector corresponding to an image. After that we can calculate the mean u and the covariance matrix S as

$$u = \frac{1}{n}\sum_{i=1}^{n}\chi_i \quad (3)$$

$$S = \frac{1}{n}\sum_{i=1}^{n}(\chi_i - u)(\chi_i - u)^T \quad (4)$$

Now consider a linear transformation W_{opt} mapping the original image space into a k-dimensional feature space (k <= n << r x c). PCA chooses the projection W_{opt} that maximizes the determinant of the covariance matrix of the projected images, that is,

$$W_{opt} = \arg\max_W \left|W^T S W\right| = [w_1, w_2, \cdots w_k] \quad (5)$$

where w_i 's are k eigenvectors of covariance matrix S corresponding to the k largest eigenvalues. Each of them represents an "eigenface".

2) Fisherfaces

Fisherfaces, also referred to as LDA, invented by Belhumeur, Hespanha and Kriegman in 1997 [7]. While PCA is clearly a powerful way to reduce the dimensionality and represent data, it doesn't consider any classes and so a lot of discriminative information may be lost when throwing components away, whereas LDA performs a class-specific dimensionality reduction. It is designed to maximize the between-class scatter while minimizing the within-class scatter, by calculating the between-class scatter matrix S_B, and within-class scatter matrix S_W, and the optimal projection is chosen as

$$W_{opt} = \arg\max_W \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|} \quad (6)$$

Suppose c is the number of classes in the training set, i.e., there are c persons' faces collected in the face database, so there are at most c-1 generalized eigenvectors, therefore at most c-1 "Fisherfaces".

D. Design of Classfier

The classifier is simply a distance classifier based on Euclidean distance. The test image is projected onto Eigenfaces and Fisherfaces in these two approaches, respectively, and the weights are then stored as coordinates in the feature space. The Euclidean distances between the test image coordinates and the training image coordinates are calculated to find out the closest training image. If a threshold is set to the algorithm model, the system will accept a facial image to the person in the database or reject a facial image for the minimum distance is still larger than the threshold.

III. System Design and Implementation

We designed and implemented an open source face recognition system named XFace for Android mobile phone platform. While implementing the system, we have to make a tradeoff between recognition accuracy and computational complexity of the algorithm due to the limited computing ability and storage capacity on mobile phone platform. This XFace application is hosted on Github, https://github.com/hujiaweibujidao/XFace.

A. System Overview

There are two main procedures in this face recognition system. The first procedure is registration process called XFace Registration Module, which includes image acquisition, ROI preprocessing and training an algorithm model with these images. While in the recognition process called XFace Recognition Module, stages are image acquisition, ROI preprocessing and predicting the test image as someone in the face database with the trained algorithm model. The overview of the system is shown in Figure 3.

The implementation of system is mainly based on the OpenCV SDK. OpenCV started by Intel in 1999 and OpenCV 2.3.1 comes with a programming interface to Android. What's more, OpenCV 2.4 comes with the very new FaceRecognizer class for face recognition. The simplest implementation of face recognition application for Android system is taking full advantages and functionalities of OpenCV for Android SDK.

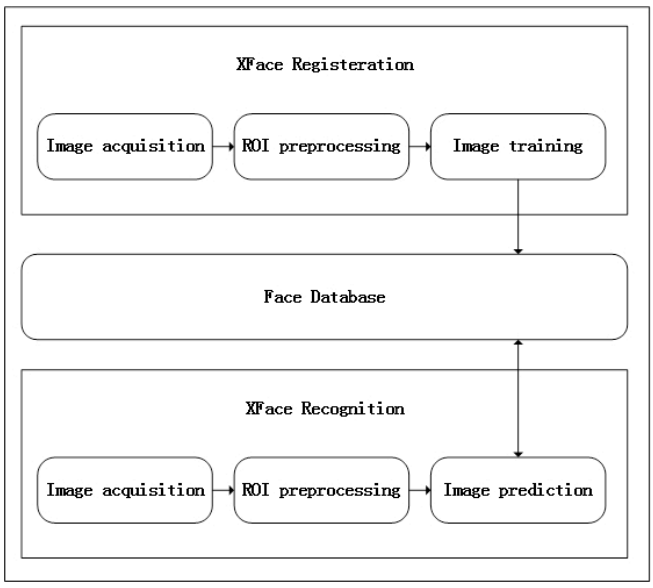


Figure 3. System Overview.

At present, there are mainly three approaches to integrate OpenCV with Android development:

a) Using JavaCV. JavaCV [3] provides most of OpenCV functions with a programming interface to Java: This way is the easiest for only using Java programming language, but soemtimes it may be less efficient when compared with the performance of pure C++ implemented OpenCV SDK.

b) Using OpenCV for Android SDK. This SDK is developed for Android platform and provides a variety of functions that support the programming of operating cameras on Android devices in both Java and native way, but the FaceRecognizer class can not be instantiated in Java for now (version 2.4.10).

c) Using pure C++ implemented OpenCV SDK. The performance of pure OpenCV SDK is the most efficient for its many optimizations, but it increased the difficulty of development by integrating Android NDK technology.

Each approach has its own advantages and disadvantages, and it depends on the functionalities of the specific application to choose the best way. For face recognition application, to achieve the best tradeoff between the computation complexity and the processing efficiency, Android NDK with pure OpenCV SDK is used rather than pure Android SDK with JavaCV. But when it comes to operating cameras on Android devices, OpenCV for Android SDK provides the most helpful and efficient way to obtain the image frames from camera and convert the raw image to matrix format, so OpenCV for Android SDK is also adopted in this prototype application.

B. *Registration Module*

The first step in registration module is letting the user choose an existing user name from the user list or create a new user name, and then it comes to the image acquisition step.

In the common way of implementing registration module, the step of image acquisition is always required to take images on Android devices. But considering the fact that current face recognition techniques are very sensitive to exact conditions in the images, we proposed an automatic image acquisition method to collect the most useful face images for our algorithm model. The basic idea here is that if we find a proper face, i.e. the ROI detected in the current frame of camera, this application automatically passes this frame of image to the ROI preprocessing stage and then saves it to the face database stored on the smart phone, which means the training face images are images needed by our algorithm model, not the images users casually give us. Although it somewhat increased the difficulty of registration module, the accuracy of the system increased significantly.

The interface of registration procedure is shown in Figure 4. The user can click on the preview image to switch between the front and rear cameras.

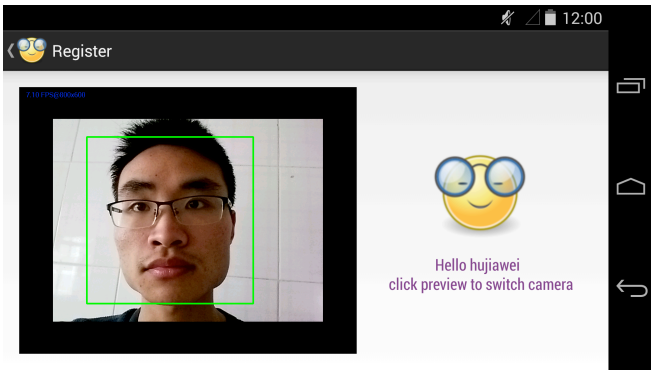


Figure 4. The procedure of registration.

Image training process is choosing Eigenfaces or Fisherfaces algorithm to obtain a trained algorithm model and saving the model data to the external storage of the smart phone. Though it seems simple, the OpenCV for Android SDK does not provide face recognition functions in Java for now as mentioned earlier. Here we use Android NDK to implement the face recognition functions in C++ and compile the native code to a dynamic library, then call these functions in Java programming code through JNI (Java Native Interface) technology.

C. *Recognition Module*

After training the Eigenfaces or Fisherfaces algorithm model with training images and corresponding user names, the recognition process is trying to figure out who a person is from a facial image captured from the camera on smart phone. The face recognition process is performed with image frames continuously coming from either the front or rear camera. When a proper facial image detected, this image frame will be passed to the ROI preprocessing procedure and predicted by the trained algorithm model. If a threshold is set to the algorithm model, the system will accept a facial image to the person in the database or reject a facial image for the minimum distance is still larger than the threshold. The interface of the recognition procedure is shown in Figure 5.

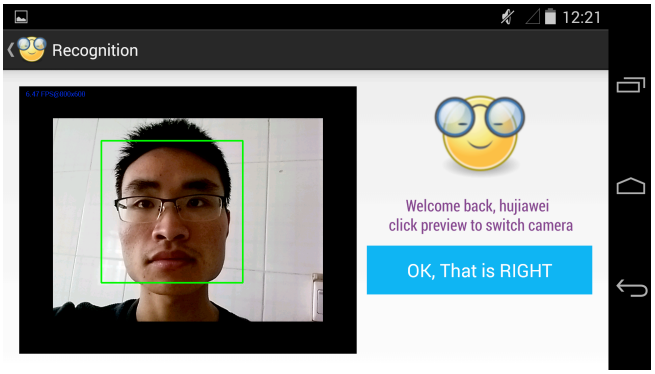
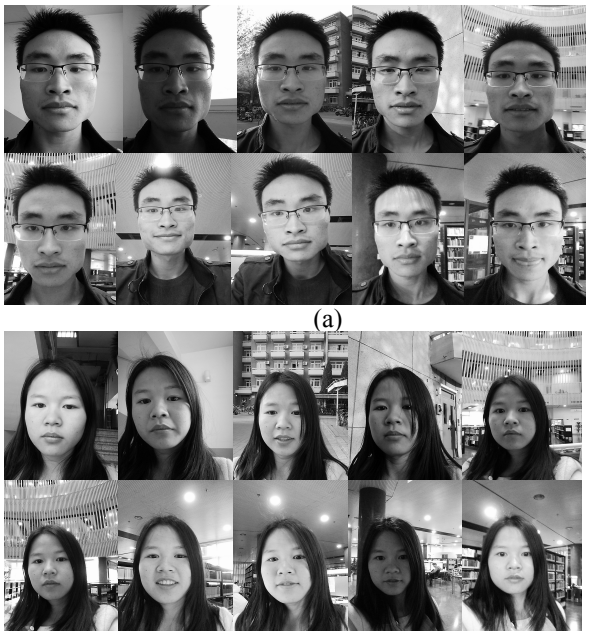


Figure 5. The procedure of face recognition.

IV. EXPERIMENT RESULTS

A. *Results of ROI preprocessing*

In real-world conditions, the face images acquired from the mobile phones vary a lot from the type of the lighting, the direction of lighting and shadows, the background and the emotion of the person, and so on. Figure 6 shows 20 face images of two persons with different emotions, different type of lighting and different backgrounds.



(a)

(b)

Figure 6. Face images of two persons under different environments.

We have made a simple experiment using 10-fold cross validation. Fisherfaces algorithm is adopted to test the performance with and without the ROI preprocessing method. Each time 5 face images of each person are randomly selected as the training set and the rest become the test set. The results are shown in TABLE I and show that, with ROI preprocessing method, the performance of the face recognition system increased significantly.

TABLE I. RECOGNITION ACCURACY WITH AND WITHOUT ROI PREPROCESSING

Times	Without ROI preprocessing	With ROI preprocessing
1	90%	90%
2	90%	100%
3	90%	100%
4	90%	100%
5	100%	100%
6	90%	90%
7	90%	100%
8	90%	100%
9	90%	100%
10	80%	80%
Average	90%	96%

B. *Results of Eigenfaces and Fisherfaces*

In order to evaluate the performance of this face recognition system, we collected a face dataset with all the images taken by the front camera of Google Nexus 5, which runs the Android system of version 4.4.4. This face dataset contains 15 classes (persons), and each person has at least 10 face images. All the face images were preprocessed using the method mentioned above and rescaled to the same size of 70x70 for the computational limitation of mobile phones.

The experiment procedure is that every time randomly select 5 face images from each person to form a training set and the rest face images of each person are left as the test set. This procedure will be conducted 10 times for each algorithm's parameter group and the accuracy on both the training set and the test set will be evaluated as the final result. The results of Eigenfaces and Fisherfaces are shown in Figure 7 and Figure 8.

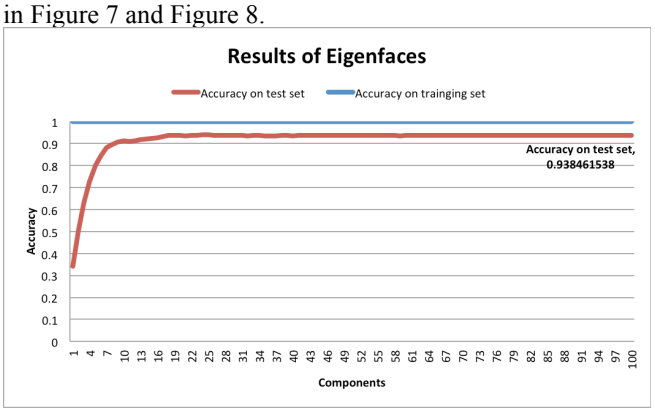


Figure 7. Results of Eigenfaces

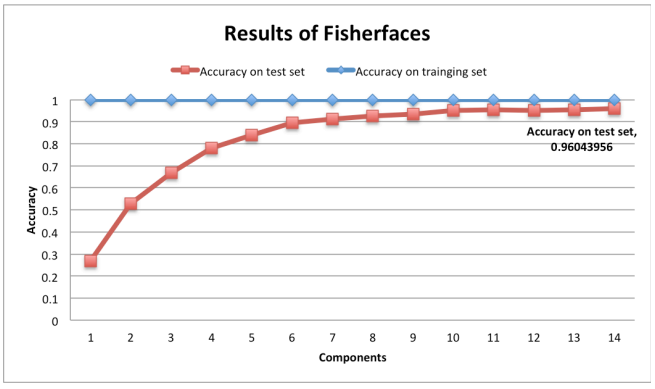


Figure 8. Results of Fisherfaces

The results we have obtained show that, both the Eigenfaces and Fisherfaces have 100% accuracy on the training set. While on the test set, the best performance has been achieved by Fisherfaces approaching the accuracy of 96.0% when components are 14, followed by 93.8% achieved by Eigenfaces when components are 29.

Besides that, when compared these two algorithms under the same components, i.e. the same dimensionality, Fisherfaces has mostly been a little better than Eigenfaces in this face dataset as we can see from Figure 9.

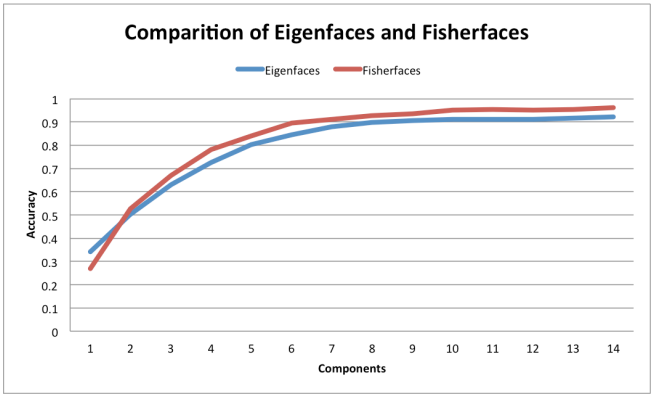


Figure 9. Comparition of Eigenfaces and Fisherfaces

V. CONCLUSIONS AND FUTURE WORK

This paper presents the design and implementation of an open source face recognition system for Android platform, which achieves a satisfactory performance on Android mobile phone platform. After face detection and ROI preprocessing, the experiment on practical face images obtained the accuracy of 93.8% with Eigenfaces and 96.0% with Fisherfaces.

In future work, more helpful functions like facial expression recognition or gender recognition may be implemented into this system, and it is possible to integrate other face recognition methods like deep learning based face recognition method to the XFace system.

ACKNOWLEDGMENTS

The second author is supported by National Natural Science Foundation of China (No. 61261130590), 973 National Basic Research Program of China (No. 2014CB340506).

REFERENCES

[1] Android NDK, http://developer.android.com/tools/sdk/ndk/.

[2] OpenCV, http://opencv.org/.

[3] JavaCV, https://github.com/bytedeco/javacv.

[4] T. Ahonen, H. Abdenour, and M. Pietikäinen. “Face recognition with local binary patterns”. Computer vision-eccv., 2004. pp. 469-481.

[5] D. L. Baggio, “Mastering OpenCV with practical computer vision projects”. Packt Publishing Ltd, 2012.

[6] M. A. Turk., and A. P. Pentland. “Face recognition using eigenfaces”, Proceedings CVPR'91, , 1991, pp. 586-591.

[7] R A. Fisher, “The use of multiple measurements in taxonomic problems”. Annals of eugenics, 7(2), 1936, pp.179-188.

[8] G. Dave, X. Chao, K. Sriadibhatla. “Face Recognition in Mobile Phones”. Tech Report, Stanford University, USA, 2010.

[9] C. Doukas, I. Maglogiannis. “A Fast Mobile Face Recognition System for Android OS Based on Eigenfaces Decomposition”. Artificial Intelligence Applications and Innovations, Volume 339, 2010, pp 295-302.

[10] G. Yang, J. Luo. “A Real-Time Face Recognition System for Android Smart Phone”. Information Technology Applications in Industry Computer Engineering & Materials Science, Vol 756-759, 2013, pp. 4006-4010.