Face Recognition in Mobile Phones

Guillaume Dave, Xing Chao, Kishore Sriadibhatla
Department of Electrical Engineering
Stanford University
Stanford, USA
gdavo@stanford.edu, chaox@stanford.edu, kishores@stanford.edu

***Abstract*—In this paper, we investigate various algorithms for face recognition on mobile phones. First step in any face recognition system is face detection. We investigated algorithms like color segmentation, template matching etc. for face detection, and Eigen & Fisher face for face recognition. The algorithms have been first profiled in MATLAB and then implemented on the DROID phone. While implementing the algorithms, we made a tradeoff between accuracy and computational complexity of the algorithm mainly because we are implementing the face recognition system on a mobile phone with limited hardware capabilities.**

Keywords-face recognition; face detection; mobile phones

I. Introduction

As mobile phones are becoming increasingly powerful, security of the data stored in mobile phones like email addresses, sensitive documents, etc., becomes very important. Most of the current phones have password protection to address security. However, a face recognition scheme is much more secure and flexible as it provides distinctive print to gain access and also the user need not remember passwords. The goal of this project is to implement a face recognition application on the DROID phone, which could be used to unlock the phone or authorize a transaction when the registered user is recognized.

We used color segmentation combined with template matching for face detection and Eigen/fisher face algorithms for face recognition. We first profiled our algorithm in MATLAB and then implemented it on DROID.

II. Prior and Related Work

Face recognition has been a very active research area in the last decade. Numerous techniques have been designed to detect and recognize faces. For the face detection algorithm, we referred to the work done by Spring 2003 EE368 students. Majority of the students used Color Segmentation and Template Matching for face detection and obtained good results. We have studied the same algorithms and evaluated the tradeoff between accuracy and computational complexity, as the face recognition system has to be implemented on mobile phone.

In [1] Vezhnevets et al. realized a quite exhaustive survey of color segmentation techniques. It appears that parametric skin modeling methods are better suited to cases like ours with limited training and target sets. A lot of different colorspaces can be used, like RGB [2], YCbCr or HSV. In [3], Chai et al. proposed to use Hue and Cr to detect skin pixels.

For basic template matching algorithm, we referred to the EE368 lecture notes.

As two of the commonly discussed face recognition methods [4-5], Eigenface and Fisherface schemes were employed and tested in this work. Here we reproduce a brief introduction of the two algorithms, while referring the reader to the relevant references.

The eigenface scheme is pursued as a dimensionality reduction approach, more generally known as principal component analysis (PCA), or Karhunen-Loeve method. Such method chooses a dimensionality reducing linear projection that maximizes the scatter of all projected images. Given a training set of N images $\Gamma_i (i = 1,2,\ldots N)$, each of size m x n, we could turn the set into a big matrix as

$$A = \left[\Phi_1 \Phi_2 \cdots \Phi_N\right]$$

where Φ_i's are column vectors, each corresponding to an image as

$$\Phi_i = \phi_i - \mu$$

$$\phi_i = reshape\left(\Gamma_i{}', [mn, 1]\right)$$

$$\mu = \underset{i}{mean}\left(\phi_i\right)$$

The total scatter matrix is defined as

$$S_T = AA^T .$$

Consider a linear transformation W mapping the image space into a p-dimensional feature space, p<=N<<mn. PCA chooses the projection W_{opt} that maximizes the determinant of the total scatter matrix of the projected images, i.e.,

$$W_{opt} = \arg\max_W \left|W^T S_T W\right| = \left[w_1 w_2 \cdots w_p\right]$$

where w_i's are eigenvectors of S_T corresponding to the p largest eigenvalues. Each of them corresponds to an "eigenface". The dimension of the feature space is thus reduced to p. The weights of the training set images and test images could be then calculated and the Euclidean distances are obtained. The test face is recognized as the face of training set with the closest distance, if such distance is below a certain distance.

Since eigenface method maximizes the scatter within the whole training set, the points corresponding to the same class may not be well clustered in the projected space, or may be smeared with each other. The variation due to lighting may cause a within-class scatter to be larger than the between cluster. One method proposed to overcome, or at least, reduce the impact from such variation, is the Fisherface algorithm. It is designed to maximize the between-class scatter while minimizing the within-class scatter, by calculating the between-class scatter matrix SB, and within-class scatter S_W, and the optimal projection is chosen as

$$W_{opt} = \frac{\arg\max_W \left|W^T S_B W\right|}{\arg\max_W \left|W^T S_W W\right|} = \left[w_1 w_2 \cdots w_p\right]$$

Since the rank of S_W is at most N-c, where c is the number of classes in the training set, PCA is used as a first step to reduce the dimensionality to avoid singularity. Note that there are at most c-1 generalized eigenvectors, therefore at most c-1 "fisherfaces".

III. Algorithm

The below block diagram depicts the major steps in our face recognition algorithm.

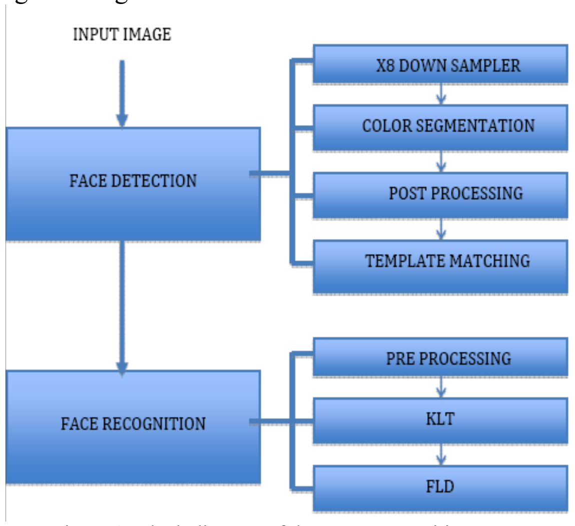


Figure 1: Block diagram of the Face Recognition system

IV. Face Detection

The first step in our face recognition algorithm is the face detection. We used color segmentation, morphological processing and template matching algorithms for the face detection. If the user takes the photo correctly, we can make the following assumptions:

- The face is centered and takes a big part of the image, since the photo is shot closely
- The illumination conditions are correct
- The user is facing the camera

So the face detection needs not to use the most performing algorithm; we rather want an algorithm that can perform well and fast in the cited conditions.

In consequence, we decided to use the following for face detection:

1) Use color segmentation to find skin pixels
2) Use morphological operations to eliminate isolated pixels (false acceptances in 1)
3) Use template matching to extract only the face, which we will use for face recognition.

IV.1 Color Segmentation

Detection of skin color in color images is a very popular and useful technique for face detection. In the skin color detection process, each pixel was classified as skin or non-skin based on its color components values.

To reduce the computation time, we first down sample the image by a factor of 8. This is done without pre-filtering to avoid the extra computation required; the aliasing introduced is negligible. Scale-by-max color balancing is also performed to reduce the effects of illumination variations. Scale-by-max was chosen over gray-world because it can be done in the gamma pre-distorted world and the gray-world assumption is not quite true for face pictures.

In the literature, the color segmentation can be done in many different ways, with some very advanced methods to process the images in extreme illumination conditions or with a cluttered background. Here we looked for a simple rule to detect the skin pixels as fast as possible. Two methods in particular were explored.

First we can work in the RGB space to avoid any calculation. We modified a rule from [2]:

A pixel with color values (R, G, B) is classified as skin if:

- $R > 95$ and $G > 40$ and $B > 20$ and
- $R > G$ and $R > B$ and
- $R-G > 15$

This algorithm performed well in general, but we wanted to explore other options, in particular because the speed of the RGB classifier was slower than expected. Other widely used color segmentation methods are based on Cr or Hue classifiers. We tested various rules for Hue, but the classification can be too strict or on the contrary too loose for some lighting conditions.

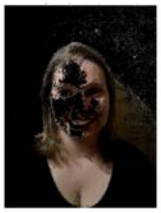


Figure 2: Example of bad performance for the Hue classifier

Finally, a Cr classifier was derived from [Chai and Ngan, 1999]: A pixel is considered as skin if $Cr \in [136\ 173]$. As Cr component is easy to compute from RGB (affine transformation) and there are only two tests to perform, the classification is really fast, and surprisingly good results were obtained. So, we adopted this last classifier.

IV.2 Morphological Image Processing

After color segmentation, a mask of non-skin pixels is obtained. However this mask is not perfect: some sparse non-skin pixels are still visible while some parts of the face can be masked (see fig. 3). Morphological image processing is thus a good way to eliminate the non-skin visible pixels and regroup the skin pixels: First, erosion is performed to remove sparse non-skin pixels. Second, dilation is performed with a larger disk to regroup the skin regions and smooth their contours. The disk diameter is bigger when scale-by-max was used because more skin pixels have been misclassified.

Below is a sample output of the color segmentation and morphological processing stages.

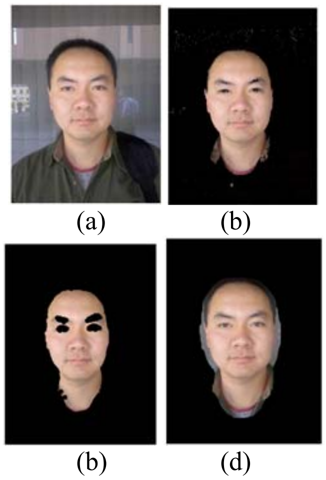


Figure 3: Early steps of the face detection: downsampling (a), color segmentation with Cr classifier (b), erosion (c), dilation (d). Aliasing effects due to downsampling are negligible for the face as it has few high frequency components.

IV.3 Template Matching

After a color segmented image is obtained, template matching is used as a final step in the face detection process. Template matching is a process of locating an object represented by a template T(x,y) in an input image I(x,y) by cross-correlating the input with the template. Cross-correlation is implemented in the frequency domain using (FFT and IFFT) as it is computationally more efficient.

Down sampled output of the color segmentation and morphological processing stages is used as an input to the template-matching block. A standard template (average of around 400 faces, both male/female and people from different ethnic backgrounds) was taken from the Internet. Then normalized 2D cross-correlation is performed with the given input image to obtain the position of the face. We also tried to manually generate the standard template using the training images that we have taken from the phone. But it didn't give good results because all the images have to be exactly aligned for the average image to be useful in matching process.

Standard template used was a gray scale image. So, input image is converted to gray scale and then cross-correlated with the template. The standard template used for matching purpose is shown in Fig. 4Figure . As we can see from Figure 4, standard template is a straight face without any rotations. So, template matching makes sense when the input image is also similar i.e. straight without any rotations.

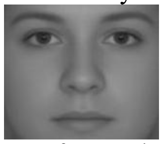


Figure 4: Standard average face template used for matching.

Faces in the test images taken using the DROID phone can be of different sizes. Correlating with a standard template size didn't give good results. So, we correlated the image with templates of different sizes (scale ratios from .6 to 1.8) and compared the correlation values. This technique worked well in detecting faces of different sizes. Here are some sample outputs of the template-matching.

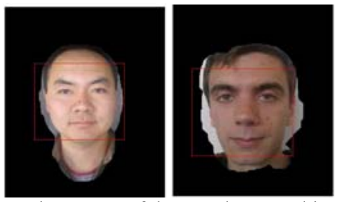


Figure 5: Sample outputs of the template matching algorithm

After color segmentation and morphological processing, ideally only the face portion of the image is left. In such cases, we probably will not need template matching. However, in cases where more skin is exposed like hands, neck etc., or if the background is similar to skin color, then template matching is helpful in detecting the face.

IV.4 Alternate approach

Template matching algorithm results are heavily dependent on the kind of template used. If the input is unaligned or doesn't closely resemble the template, then the results are not good. It also failed for images under bad illumination conditions and dark skin colors. So, we tried to detect the face using an alternate approach.

Output image of color segmentation stage is post processed using region labeling. If background color is same as skin color, then the color segmentation algorithm cannot differentiate it. Region labeling will help in removing these small skin colored background regions. The assumption here is face occupies the major portion of the image. Figure 6 shows the images before and after applying the regional labeling algorithms.

In some images, to accurately detect the face, neck region has to be removed. We do a simple post processing after the regional labeling stage to remove the neck region. If the neck region is included, usually no of skin pixel rows would we much greater than that of columns. By detecting this condition, we can remove the neck region to accurately detect the face. Figure 7 shoes this post processing.

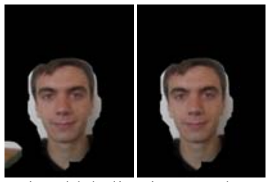


Figure 6: Regional labeling input and output images

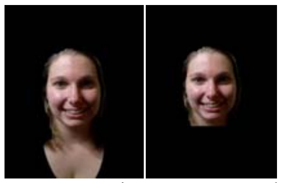


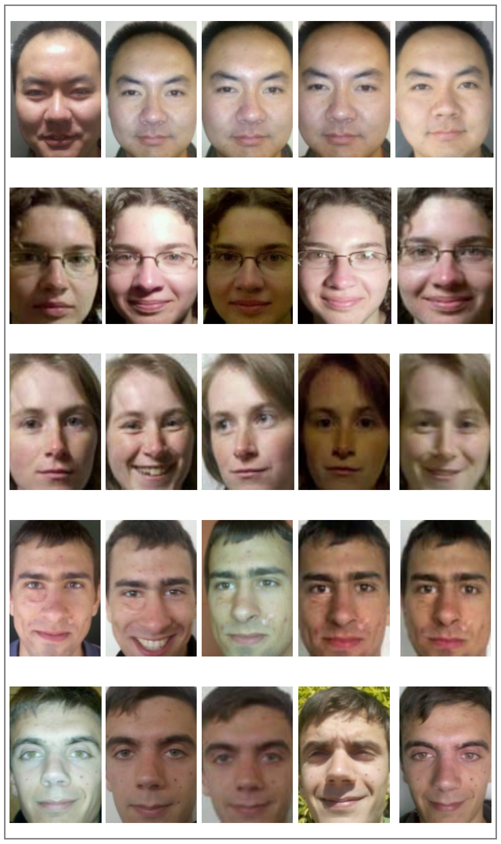
Figure7: Post processing to remove neck region

V. FACE RECOGNITION

After the face portion is detected by the previous steps, we tried to identify a person's face in the case that his or her information has been stored in the training set, or reject this person if not. Both eigenface and fisherface schemes were employed and tested.

V.1 TRAINING SET AND TEST SET

We used a training set of 45 images, containing 9 classes (persons), and 5 images per classes (Fig. 8). The cropped faces captured by the face detector were used as the inputs of the recognition step. The faces are centered and rescaled to the same size.



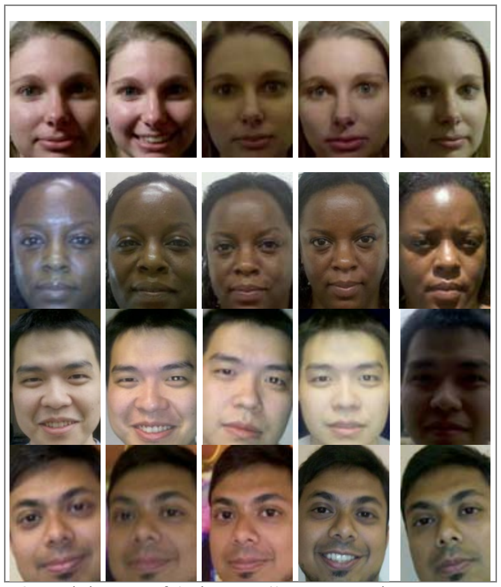
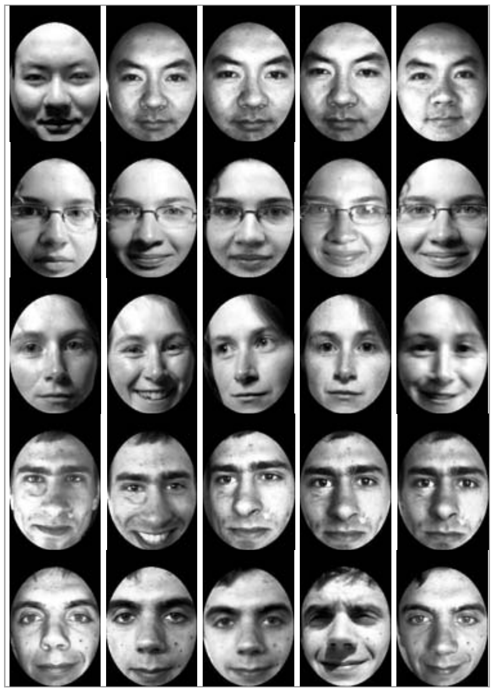


Figure 8: Training set of 45 images (9 persons, 5 images per person)

We could see from the images that for a same person, the color and direction of lighting could vary a lot, bringing in more scatter within the same class on top of the variation of facial expressions and angle. The background could also vary, introducing differences between single images of the same class. To alleviate these effects, some pre-processing step were performed, including masking with an oval shape that crops out the face part only, color to gray transformation, and histogram equalization. The resulted images are shown as following in Fig. 9.



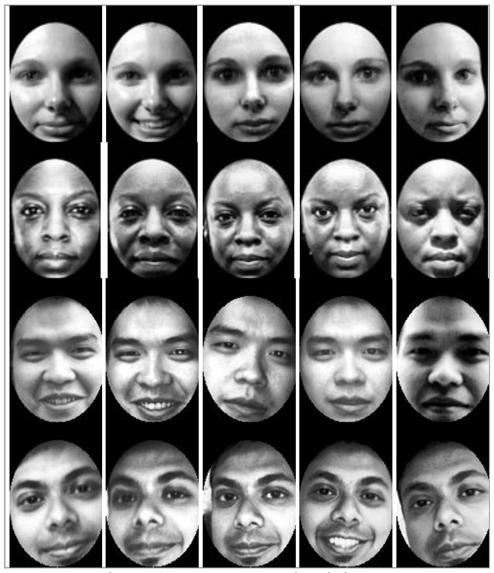


Figure 9: Pre-processed training set

V.2 EIGENFACES

The same training set was used for both eigenface and fisherface approaches. In eigenface scheme, the calculated average face is shown in Fig. 10. In our particular case, there are 45 images in the test set, therefore at most 44 eigenvectors, i.e., 44 eigenfaces could be found, and they were sorted according to descending order of their corresponding eigenvalues. The first ten eigenfaces are shown in Fig. 11. The faces in the training set are then projected onto these eigenfaces, and the weights are stored as coordinates in the feature space.

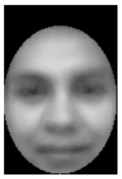


Figure 10: Average face of the training set

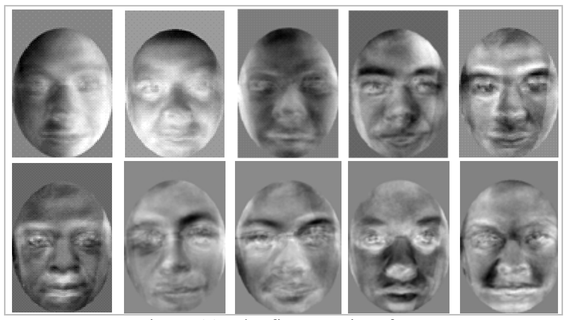


Figure 11: The first ten eigenfaces

V.3 FISHERFACES

For fisherface approach, the eigenfaces calculated above were used as a first step, but only N-c = 45-9 = 36 eigenvectors were retained for the need of dimensionality reduction. The fisher linear discriminants were then calculated in the low dimensionality space. Note that in our case, 9 classes were included in the training set. Therefore c-1 = 9-1 = 8 fisherfaces exist (Fig. 12)

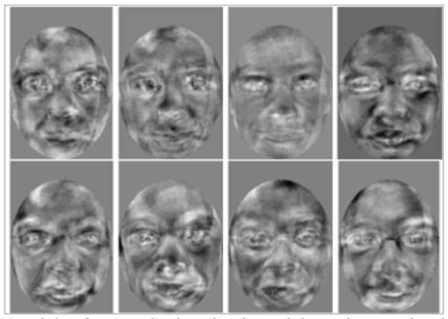


Figure 12: Fisherfaces calculated using Fisher Linear Discriminants

Similarly as above, the faces in the training set are projected onto these fisherfaces, and the weights were stored.

VI. IMPLEMENTATION ON DROID

After profiling and testing the face recognition algorithm in MATLAB, we then implemented the algorithm on the Motorola DROID phone. For the face detection part, we used the face detector available in the Android API instead of implementing our face detection algorithm. The documentation is really brief so we do not know for sure on what algorithm it is based. As it is very robust, we assumed it uses Viola-Jones. It was used to crop a face bitmap out of the picture taken with the phone camera.

We developed the face recognition algorithm on the Android platform. As we have found Fisherface to be more efficient than Eigenface, only Fisherface was implemented in Android. The KLT and Fisher LDA projection matrices were computed in Matlab for our fixed training set and stored on the Droid external storage.

We then reproduced the steps previously described for Matlab: preprocessing, mean removal and projection. We particularly paid attention to stay close to the Matlab implementation because the same training set eigenvectors were used. We tried to use JJIL for gray-scale histogram equalization but it appeared that Gray8HistEq has an undocumented bug: output values have an offset depending on the input image. So finally JJIL was not used at all; instead a new version of Gray8HistEq was written.

Another critical challenge was to keep the computation time as small as possible, so that the user experience is the best possible. In order to achieve that goal, we first downsample the high resolution camera pictures by a factor of 8. The computation time is then roughly 64 times smaller than for a full size image. Second, the I/O management was critical. We had to convert the text files written with Matlab into binary files to store our KLT, FLD and mean face matrices in an efficient way. Then it was possible to read these values with a DataInputStream, which is much faster than to read line by line from a text file. For example, the projection took only 0.8 sec with a DataInputStream, compared to 14 sec with text files.

Finally, our algorithm can identify a user in no more than 1.6 sec (face detection and recognition), with a simple user interface depicted in Fig. 13.

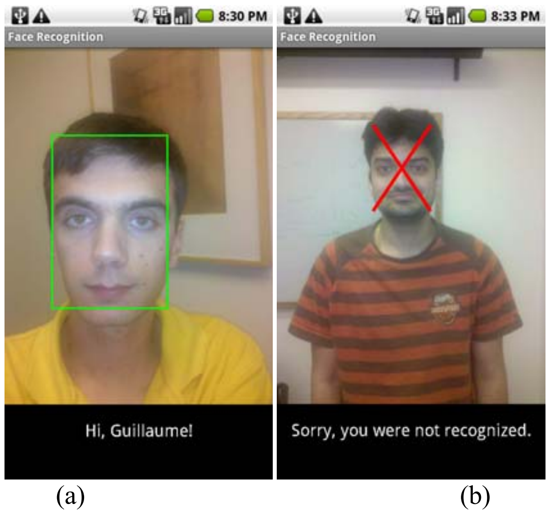


Figure 13: Result of the Face Recognition application for Android.
(a) user correctly recognized, (b) user correctly rejected

VII. RESULTS

To test how well the recognition system works, we have a test set of 134 face images, containing 10 different persons, one of which was not included in the training set. The test image is projected onto eigenfaces and fisherfaces in the 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 and the closest training image is picked out. A threshold is set such that if the closest distance is above the threshold, the test face is considered unrecognized, and if below, is associated with the identity of the closest face. The returned result is thus divided into four cases:

- False rejection: if the face is associated with the correct face, but the distance is larger than threshold.
- False acceptance: if the face is associated with the wrong face, but the distance is smaller than threshold.
- Correct rejection: if the face is associated with the wrong face, and the distance is larger than threshold.
- Correct acceptance: if the face is associated with the correct face, and the distance is smaller than threshold.

By tuning the threshold value, the number of images that fall into the four categories would change. The result is reproduced in the table below. The addition of the percentage of correct rejection and correct acceptance is considered as total correct rate. With eigenface scheme, we were able to achieve a total correct rate of 84.3%, whereas with fisherface, the correct rate goes up to 94.0%.

	Total number of test images: 134 FR: False Rejection FA: False Acceptance CR: Correct Rejection CA: Correct Acceptance									
	Eigenface Max Correct Rate (84.3%)					**Fisherface Max Correct Rate (94.0%)**				
Threshold	FR	FA	CR	CA	Total correct	FR	FA	CR	CA	Total correct
50	42	0	39	53	92	70	0	16	48	64
60	34	0	39	61	100	69	0	16	49	65
70	28	0	39	67	106	62	0	16	56	72
80	21	1	38	74	112	57	0	16	61	77
90	17	8	31	78	109	53	0	16	65	81
100	6	22	17	89	106	47	0	16	71	87
110	2	35	4	93	97	41	0	16	77	93
120	1	38	1	94	95	33	0	16	85	101
130	0	38	1	95	96	27	0	16	91	107
140	0	38	1	95	96	23	0	16	95	111
150	0	39	0	95	95	11	1	15	107	122
160	0	39	0	95	95	5	3	13	113	126
170	0	39	0	95	95	3	6	10	115	125
180	0	39	0	95	95	2	7	9	116	125
190	0	39	0	95	95	0	10	6	118	124
200	0	39	0	95	95	0	12	4	118	122
210	0	39	0	95	95	0	14	2	118	120
220	0	39	0	95	95	0	15	1	118	119
230	0	39	0	95	95	0	15	1	118	119
240	0	39	0	95	95	0	16	0	118	118
250	0	39	0	95	95	0	16	0	118	118

Figure 14: Table of recognition results with Eigenface and Fisherface scheme

We also plotted the false acceptance rate (FAR) versus the false rejection rate (FRR) and got the equal error rate (EER) where the FAR and FRR are equals. It is known that the smaller the EER, the better a recognition system functions. In our experiment, we achieved an EER of 35% for eigenface, and an EER of 25% for fisherface. As expected, the fisherface scheme worked better for recognizing faces under varying lighting conditions.

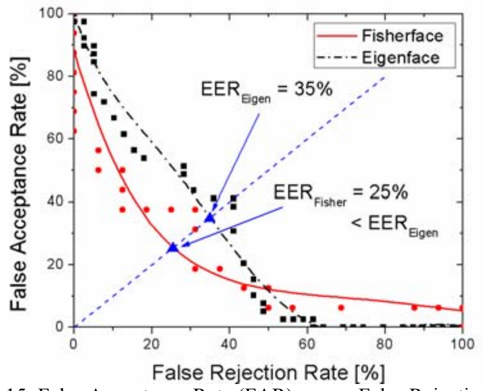


Figure 15: False Acceptance Rate (FAR) versus False Rejection Rate (FRR) and Equal Error Rate (EER) of test set recognition using both Eigenface and Fisherface scheme

Step	Droid computation time
Preprocessing	0.52s
Mean removal	0.18s
Projection	0.80s
Distance calculation and identification (including GUI)	0.08s
Total	1.58s

Figure 16: Table of various run times on Droid

The run time of different steps of DROID implementation of the fisher scheme is shown in the table above. It is also worth noting that for our particular application, which is face recognition in mobile phones that likely to be used for security reasons, we could tolerate a slightly higher false rejection rate, while a very low false acceptance is preferred. The threshold value of 150 with fisherface scheme is then chosen with such intention.

VIII. Conclusions

In this project, we investigated various algorithms for implementing face recognition system on mobile phones. We used color segmentation and template matching for face detection. There are some limitations for our face detection algorithm. If background color in an image is akin to the skin color, then the color segmentation algorithm cannot distinguish. In the template-matching algorithm, our results are dependent on the standard template chosen (orientation etc), illumination conditions etc. Also it didn't perform well while detecting faces of people from specific ethnic origins. Eigenface and fisherface algorithms are then employed and tested for face recognition. We achieved a total correct recognition/rejection rate of 84.3% with eigenface and 94.0% with fisherface. An EER of 35% was obtained with eigenface and 25% with fisher, indicating that the latter had better performance. We finally implemented face recognition algorithm on DROID and integrated it with a standard face detection application and tested the whole face recognition system.

As part of the future work, we would like to develop an application that would allow the user to add/delete face classes in the training set. This would give users the freedom to define their own user groups rather than a pre-defined set on the server. We would also like to explore better algorithms for face detection and face recognition.

IX. References

[1] V Vezhnevets, V Sazonov, A Andreeva, "A survey on pixel-based skin color detection techniques," Proc. Graphicon, 2003 – Citeseer.

[2] P. Peer,J. Kovac,F. Solina, "Human skin colour clustering for face detection," in submitted to EUROCON 2003 - International Conference on Computer as a Tool.

[3] D. Chai and K. N. Ngan,"Face segmentagion using skin-color map in videophone applications," IEEE Trans. on Circuits and System for video Technology, Vol.9, No.4, pp. 551-564, 1999.

[4] Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," IEEE transactions on pattern analysis and machine intelligence, Vol. 19, No. 7, 1997..

[5] Thomas Heseltine, Nick Pears, Jim Austin, Zezhi Chen, "Face Recognition: a comparison of appearance-based approaches," Proc. VIIth Digital image computing: Techniques and applications, 10-12, 2003.

X. Appendix: Job partition in the group

Guillaume Davo: Color Segmentation on Matlab; Implementation of face recognition algorithms on DROID phone.

Kishore Sriadibhatla: Template matching and color segmentation output post-processing on Matlab.

Xing Chao: Face recognition algorithms (Eigenface and Fisherface) on Matlab.