Abstract—In this project, face recognition algorithms for mobile phone are investigated. Eigenface recognition is applied due to its fast speed. Skin color segmentation by Cr classifier is used for face detection step. Experimental results show 80% accurate rate of face recognition.

Keywords—face recognition; face detection; mobile phone; eigenface

I. INTRODUCTION

Face recognition using mobile phone is getting popular since it provides a convenient way to recognize a person and retrieve information of that person. Individuals share their public profile on the internet, and anyone can access that information by taking the picture and recognizing the person. It also can be used for ID authentication to protect owner’s personal information even when the mobile phone is lost or stolen, as mobile devices are carrying ever more personal information including address books, schedules and payment information. The goal of this project is to implement a face recognition application on the DROID phone, which could be used to unlock the phone when the registered user is recognized.

I used skin color segmentation method for face detection and eigenface algorithm for face recognition. Basic process is to take a picture using mobile phone and detects faces, send detected faces to the server/database, match the face with training images, and return a face tag and other info on the mobile phone. System architecture is shown in the figure 1.

II. REVIEW OF PREVIOUS WORK

Face recognition has been a very active research area in the last decade. Numerous techniques have been designed to detect and recognize faces [1, 2]. The first step of face recognition is face detection. Face detection algorithms are often implemented as a binary pattern-classification task. That is, the content of a given part of an image is transformed into features, after which a classifier trained on example faces decides whether that particular region of the image is a face, or not. Often faces can be detected by color, motion and model-based algorithms. Skin color algorithms are considered in this project since it has proven to be a useful and robust cue for face detection. Existing skin color modeling methods [3] include explicitly defined skin region, nonparametric skin distribution modeling and parametric skin distribution modeling. The first method of building a skin classifier is to define explicitly (through a number of rules) the boundaries skin cluster in some colorspace. For example [4]:

\[(R, G, B)\text{ is classified as skin if:}\]
\[R>95 \text{ and } G>40 \text{ and } B>20 \text{ and}\]
\[\text{Max}\ {R, G, B}-\text{min}\ {R, G, B}>15 \text{ and}\]
\[|R-G|>15 \text{ and } R>G \text{ and } R>B\]

The obvious advantage of this method is simplicity of skin detection rules that leads to construction of a very rapid classifier. The main difficulty achieving high recognition rates with this method is the need to find both good colorspace and adequate decision rules empirically. The key idea of the non-parametric skin modeling methods is to estimate skin color distribution from the training data without deriving an explicit model of the skin color. Popular methods are normalized lookup table [5], Bayes classifier [6] and self organizing map [7]. The advantages of the non-parametric methods are fast in training and usage, but they require more storage space for the training data. On the contrary, parametric skin distribution models have the ability to generalize and interpolate the training data. Various models are introduced such as single Gaussian [8],
mixture of Gaussians [9], multiple Gaussian clusters [10] and elliptic boundary model [10]. Obviously, the goodness of fit is more dependent on the distribution shape; therefore the model performance varies significantly from colorspace to colorspace.

For face recognition algorithm, there are appearance-based methods and model-based methods. The former includes linear and nonlinear analysis; model-based method requires shape and texture information. As a commonly discussed linear analysis method, eigenface scheme was considered in this project. The approach of using eigenfaces for recognition was developed by Sirovich and Kirby (1987). It is considered the first successful example of facial recognition technology. Eigenfaces have advantages over other techniques available, such as the system's speed and efficiency. Using eigenfaces is very fast, and able to functionally operate on lots of faces in very little time. Here I reproduce a brief introduction of this algorithm.

The eigenface scheme is pursued as a dimensionality reduction approach, more generally known as principal component analysis (PCA), or Karhunen-Loeve method. Eigenfaces decomposition and similarity detection relies on measuring the similarity between a new face image and a reference one, projecting both the images into an eigenspace, previously created by training, and calculating the distance between the projections. The idea behind the eigenface technique is to extract the relevant information contained in a facial image and represent it as efficiently as possible. Rather than manipulating and comparing faces directly, one manipulates and compares their representations.

The general eigenface recognition procedure includes three steps.

1) Initialization: Obtain training faces and calculate the eigenfaces.
2) Operating: Calculate a set of weights by projecting the test face into eigenface space.
3) Recognition: If the test face is close to a certain training face, it is recognized.

Given a training image set M images \( \Gamma_i \), each size of \( N \) by \( N \), we could turn the set into a big matrix as

\[
A = [\Phi_1, \Phi_2, \ldots, \Phi_M]_{N \times N}
\]  

Where \( \Phi_i \)'s are column vectors, each corresponding to an image as

\[
\Phi_i = \Gamma_i - \Psi
\]  

\( \Psi \) is the average face defined by

\[
\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i
\]

This set of very large vectors is then subject to principal component analysis, which seeks a set of \( M \) orthonormal vectors, \( \mu_n \), which best describes the distribution of the data. The \( k \)th vector, \( \mu_k \) is chosen such that

\[
\lambda_k = \frac{1}{M} \sum_{i=1}^{M} (\mu_k^T \Phi_n)^2
\]

is a maximum, subject to

\[
\mu_i^T \mu_k = \begin{cases} 1, & i = k \\ 0, & otherwise \end{cases}
\]

The vectors \( \mu_i \) and scalars \( \lambda_k \) are the eigenvectors and eigenvalues, respectively, of the covariance matrix

\[
C = \frac{1}{M} \sum_{i=1}^{M} \Phi_i \Phi_i^T = AA^T
\]

The matrix \( C \), however, is \( N^2 \times N^2 \), and determining the \( N^2 \) eigenvectors and eigenvalues is an intractable task for typical image sizes. Assuming the number of data points in the image space is less than the dimension of the space (\( M < N^2 \)), which is usually the case, we construct the \( M \) by \( M \) matrix \( L = A^T A \), where \( L_{mn} = \Phi_m \Phi_n^T \), and find the \( M \) eigenvectors \( v_n \) of \( L \). These vectors determine linear combinations of the \( M \) training set face images to form the eigenfaces \( \mu_n \):

\[
\mu_i = \sum_{k=1}^{M} v_{ik} \Phi_k = AV_i, i = 1, \ldots, M
\]

With this analysis the calculations are greatly reduced, from the order of the number of pixels in the images (\( N^2 \)) to the order of the number of images in the training set (\( M \)). The eigenfaces span an \( M \) dimensional subspace of the original image space. The \( M \) most significant eigenvectors of the \( L \) matrix are chosen as those with the largest associated eigenvalues. The weights of the training set images are calculated from a simple operation:

\[
\omega_i = \mu_i (\Gamma - \Psi), i = 1, \ldots, M
\]

And they form a vector \( \Omega^T = [\omega_1, \omega_2, \ldots, \omega_M] \) that describes the contribution of each eigenface in representing the input face image, treating the eigenfaces...
as a basis set for face images. Weights of test images are 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.

### III. Algorithm

The following diagram describes the main steps in my face recognition algorithm.

![Block diagram of face recognition algorithm](image)

#### IV. FACE DETECTION

There is a great trend showing that face recognition is not so much about face recognition at all - it is much more about face detection. When faces could be located exactly in any scene, the recognition step afterwards would not be so complicated. There are several assumptions if the use takes the correct photo:

- 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

In this project, I use an algorithm that performs well and fast instead of the most accurate algorithm. As shown in the figure 2, firstly input image with resolution of is down sampled by a factor of 4 to reduce the computational burden; the aliasing introduced is negligible. Then skin color segmentation is applied to find skin pixels. Morphological operations are performed to eliminate isolated pixels. A simple algorithm is designed to remove the neck to further improve the detection accuracy and an initial bounding box is created. But detected face area may contain some background and also some non-skin part of the face. Finally I remove 10% area from top, left and right side of the boundary and consider the remaining part as detected face area. Technical details will be shown next.

#### IV.1 SKIN COLOR SEGMENTATION

In this work, two explicitly defined skin region methods are explored and compared.

One firstly transfers RGB image into LAB color space, then two global thresholds are selected based on the A space and B space. Pixels with A, B values above two thresholds are determined as skin pixels. This algorithm performed well in general, but other options are explored, in particular because the speed of the LAB classifier was slower than expected due to whole image color space transformation.

The other widely used color segmentation methods are based on Cr classifiers. A Cr classifier was derived from [11]: A pixel is considered as skin if Cr is within the interval of 136 and 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. I adopted this classifier.

#### IV.2 POST 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. 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. A rectangular bounding box is then created for the largest remaining region based on the first assumption. In some images, to accurately detect the face, neck region has to be removed. I do a simple post processing to remove the neck region. Assuming the ratio of the width of a normal face and its height falls into certain range, I just remove the skin region below certain height threshold $T$, which has great possibility to be the neck skin region. Finally, bounding box is shrunk by 10% to the center area to reduce the chance of collecting background or hair portions. Below is a sample output of face detection stages.
Figure 3. Face detection steps: (a) input image, (b) downsampling, (c) skin color segmentation with Cr classifier, (d) erosion, (e) dilation, (f) face after morphological operations, (g) remove neck, (h) face output with 10% shrink.

V. FACE RECOGNITION

After the face portion is detected by the previous steps, I 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 by eigenface method.

V.1 TRAINING SET

In this work, the training set consists of 20 images with 4 classes (persons) and 5 images per person. I took the training images with slightly different lighting conditions, different facial expression and also different in scale. But the training images used in the face recognition are rescaled to the same pixel size of 256×192. To alleviate those variation effects, some preprocessing steps are conducted. First input RGB faces are converted to gray scale which reduces the dimension of image matrix. Second histogram equalization step is applied. Third each gray image is normalized. The normalized training set is shown below.

Figure 4. Pre-processed training set

V.2 EIGENFACE CALCULATION

After the training set is preprocessed, the mean image is calculated shown below.

Figure 5. mean image

All the eigenfaces are calculated and sorted according to a descending order. I then kept 8 most significant eigenfaces to span a 8 dimensional subspace shown in figure 6.

Figure 6. 8 most significant eigenfaces

The weights of each training image are computed and stored in the memory.
V.3 FACE RECOGNITION

Considering a test image, the difference between test image and mean image is computed at first. Then the weights of test image are calculated by a set of point-by-point multiplication of the difference and eigenfaces. And the minimum Euclidian distance between weights of test image and training images determines which class (person) provides the best description of the test image. A face recognition example is shown below in which class 4 is recognized with person’s ID shown above the bounding box.

VI. EXPERIMENTAL RESULTS

To test the performance of the face recognition system, 50 test images are captured including 5 different classes (persons), 10 images per person, in which one class is not in the training set. 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 (FR): if the face is associated with the correct face, but the distance is larger than threshold.
- False acceptance (FA): if the face is associated with the wrong face, but the distance is smaller than threshold.
- Correct rejection (CR): if the face is associated with the wrong face, and the distance is larger than threshold.
- Correct acceptance (CA): if the face is associated with the correct face, and the distance is smaller than threshold.

With a threshold of 125, total 40 images are determined correctly. The result table is shown below.

<table>
<thead>
<tr>
<th>CA</th>
<th>CR</th>
<th>FA</th>
<th>FR</th>
<th>Total Correct</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>13</td>
<td>3</td>
<td>7</td>
<td>40</td>
<td>80%</td>
</tr>
</tbody>
</table>

VII. IMPLEMENTATION IN MOBILE PHONE

In this project, the main face detection and recognition algorithm is implemented using matlab. A Java servlet is built on the server (Dell Inspiron 6400). Mobile phone application captures the image and sends to the server automatically. After the image is recognized, the post processed image is then sent back to the mobile phone to show the results as shown in the figure 8.

Since the final goal of this project is to implement the face recognition algorithm on mobile phone, the computation cost is a main concern. In this work, several approaches are applied to reduce the computational cost. First, down sampling of original image a factor of 4; second, Cr classifier is selected for skin color
segmentation which only has two comparison operations instead of whole image LAB conversion; third, only several most significant eigenfaces are kept for recognition. Run time for face detection step on the server (CPU T2050~1.6 GHz, 2G Ram) is 0.88s and run time for face recognition step is 0.17s. So the overall face detection and recognition process takes about 1s.

VIII. SUMMARY

In this project, I reviewed various face detection algorithms and such as RGB classifier, LAB classifier and chose Cr classifier as skin color segmentation. For face recognition, I used eigenface algorithm. Under controlled lighting condition and assuming front face case, face detection algorithm performs well. And eigenface recognition achieved 80% correct rate with a given threshold. But the face detection will fail apart if there are similar and large color region existing in the background. Hybrid face detection method needs to be considered for reliable face detection such as template matching. Eigenface recognition can be further improved with other better algorithms.

Another part of future work is to develop an application that would allow users to add/delete face classes in the training set. This would give users freedom to set training image rather than pre-defined set on the server.

IX. REFERENCE


In IEEE International Conference on Image Processing (ICIP’2002), vol. 1, 289–292.