Face Recognition for Mobile Phone Using Eigenfaces

Hao Yu (14917715)

[Introduction]

Human face is the primary identification method. Therefore, by recognizing face, we are able to gather information about the person. 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.

This project deals with the topic of face recognition using mobile phone. Many face recognition algorithms have been proposed in literature [1,2]. However, the application of the well known and established techniques to mobile devices, such as mobile phones, raises some challenges related to two major issues: the inefficiency of the algorithms and the limited processing capabilities of the mobile devices.

In this project, eigenfaces for face recognition will be applied. The approach of using eigenfaces for recognition was developed by Sirovich and Kirby (1987) and used by Matthew Turk and Alex Pentland in face classification [3]. 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.

[Technical Approaches]

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.

Detailed technical procedures applied in this project will be discussed below. Basic eigenface decomposition and detection include creating eigenfaces from training images, calculating the eigenfaces and classifying a face image. In order to reduce the computational cost and speed up the operation process for large dataset, possible performance improvement technique will be discussed.

Creating eigenfaces

To create a set of eigenfaces, one must:
1. Prepare a training set of face images. In this work, Yalefaces database is selected as the training dataset (15 faces with 11 sample images per face). The pictures constituting the training set should have
been taken under the same lighting conditions, and must be normalized to have the eyes and mouths aligned across all images. They must also be all resampled to the same pixel resolution. Each image is treated as one vector, simply by concatenating the rows of pixels in the original image, resulting in a single row with \( r \times c \) elements. For this implementation, it is assumed that all images of the training set are stored in a single matrix \( T \), where each row of the matrix is an image.

2. Subtract the mean. The average image \( \mathbf{a} \) has to be calculated and then subtracted from each original image in \( T \).

3. Calculate the eigenvectors and eigenvalues of the covariance matrix \( S \). Each eigenvector has the same dimensionality (number of components) as the original images, and thus can itself be seen as an image. The eigenvectors of this covariance matrix are therefore called eigenfaces. They are the directions in which the images differ from the mean image. Usually this will be a computationally expensive step (if at all possible), but the practical applicability of eigenfaces stems from the possibility to compute the eigenvectors of \( S \) efficiently, without ever computing \( S \) explicitly, as detailed below.

4. Choose the principal components. The \( D \times D \) covariance matrix will result in \( D \) eigenvectors, each representing a direction in the \( r \times c \)-dimensional image space. The eigenvectors (eigenfaces) with largest associated eigenvalue are kept.

**Calculating Eigenfaces**

Let the training set of face images be \( \Gamma_1, \Gamma_2, \Gamma_3, \ldots, \Gamma_M \). The average face of the set if defined by \( \Psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma_n \). Each face differs from the average by the vector \( \Phi_n = \Gamma_n - \Psi \). 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

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

is a maximum, subject to

\[
\mu_l^T \mu_k = \begin{cases} 1, & l = k \\ 0, & \text{otherwise} \end{cases}
\]

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

\[
C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T = AA^T
\]

Where, the matrix \( A = [\Phi_1 \Phi_2 \ldots \Phi_M] \). 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 \( L_{mn} = \Phi_m^T \Phi_n \), and find the \( M \) eigenvectors \( \nu_n \) of \( L \). These vectors determine linear combinations of the \( M \) training set face images to form the eigenfaces \( \mu_n \) :

\[
\mu_n = \sum_{k=1}^{M} \nu_{nk} \Phi_k = A \nu_n, n = 1, \ldots, M
\]

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 (\( M \)).
Classifying a face image

The eigenfaces span an \( M' \) dimensional subspace of the original \( N^2 \) image space. The \( M' \) most significant eigenvectors of the \( L \) matrix are chosen as those with the largest associated eigenvalues.

A new face image \( \Gamma \) is transformed into its eigenface components (projected onto “face space”) by a simple operation

\[
\omega_n = \mu_k (\Gamma - \Psi)
\]

for \( n=1, \ldots, M' \). This describes a set of point-by-point image multiplications and summations.

The weights 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. The vector may then be used in a standard pattern recognition algorithm to find which of a number of predefined face classes, if any, best describes the face. The simplest method for determining which face class provides the best description of an input face image is to find the face class \( k \) that minimizes the Euclidian distance

\[
\varepsilon_k^2 = \| (\Omega - \Omega_k)^2 \|
\]

where \( \Omega_k \) is a vector describing the \( k \)th face class. The face classes \( \Omega_k \) are calculated by averaging the results of the eigenface representation over a small number of face images (as few as one) of each individual. A face is classified as “unknown”, and optionally used to create a new face class.

Because creating the vector of weights is equivalent to projecting the original face image onto to low-dimensional face space, many images (most of them looking nothing like a face) will project onto a given pattern vector. This is not a problem for the system, however, since the distance \( \varepsilon \) between the image and the face space is simply the squared distance between the mean-adjusted input image \( \Phi = \Gamma - \Psi \) and \( \Phi_f = \sum_{i=1}^{M'} \omega_i \mu_i \), its projection onto face space:

\[
\varepsilon^2 = \| \Phi - \Phi_f \|^2
\]

Thus there are four possibilities for an input image and its pattern vector: (1) near face space and near a face class; (2) near face space but not near a known face class; (3) distant from face space and near a face class; (4) distant from face space and not near a known face class. In the first case, an individual is recognized and identified. In the second case, an unknown individual is present. The last two cases indicate that the image is not a face image. Case three typically shows up as a false positive in most recognition systems; in this framework, however, the false recognition may be detected because of the significant distance between the image and the subspace of expected face images.

Possible performance improvement technique [4, 5]

Preprocessing of the training or input image region by aligning a face horizontally will be considered to improve the face recognition accuracy with different head tilts after the basic eigenface recognition is implemented successfully. Firstly, an eye feature extraction is applied to locate the eyes; based on the orientation of the eye pair, the selected image is rotated so that their inclination is removed. Also, the distance between the eyes is used to resize the same image to a given value of width and height. The final step of the image processing consists in cropping the face’s area with an elliptical mask. This reduces the influence of hair and background pixels at the four corners of the rectangular region and speed up the process.
[Evaluation criterion]
Time for the training set

Time for the image recognition

Recognition accuracy with different head tilts, with varying illumination

[Milestones achieved so far]

Sep. 30th Literature Review
Oct. 5th Toy project for communicating mobile phone with server
Oct. 10th Get training dataset of Yale faces
Oct. 20th Basic eigenface calculation and recognition algorithm
Oct. 26th Classify a face image by minimizing the Euclidian distance

[Remaining milestones]

Nov. 19th Algorithm optimization with performance improvement technique
Nov. 26th Recognition evaluation
Dec. 17th Final Report

[References]


