Deliverable 1
Face Recognition Program
(Eigenface-Based Approach)

Wallun Chan
CS 297
Spring 2005
Overview

- Objective
- Principal Component Analysis (PCA)
- PCA Derivation
- Jacobian Transformation of a Symmetric Matrix
- Eigenfaces for Digital Image Recognition
- Software Implementation
- Graphical User Interface
- Digital Image Data
- Test Results
- Conclusion
Objective

- Understand theory behind Principal Components Analysis (PCA)
- Understand theory behind Jacobian matrix operations in transforming symmetric matrix and obtaining the needed Eigenvalues and Eigenvectors
- Use PCA as proof of concept in Eigenface approach for digital image recognition and reconstruction
- Implement software to apply Eigenface approach
- Validate results using database of grayscale digital face images to quantify recognition performance
Principal Component Analysis

- Picks out patterns that best represents correlations between images
- For N points in D-dimensional space, the eigenvectors of the covariance matrix for the data set form the principal axes of the D-dimensional data points.
- Image data expressed more efficiently in terms of the D orthogonal eigenvectors.
- Eigenvectors form a sort of new coordinate system.
Principal Component Analysis

- Projection (dot product) of the data points (vector from origin to each data point) onto a D-dimensional vector passing through mean in the direction of the eigenvectors is maximized.
- Direction of new principal axes is chosen such that the scatter is maximized for each (dimension) axis.
- Measure of scatter is just the covariance matrix times by N - 1.
- Covariance matrix is multi-dimensional version of variance that measures the spread of data points about the mean.
Dimensionality reduced by eliminating those eigenvectors that do not contribute much to the scatter.
- Basis for image compression such that redundancy is eliminated by exploiting the correlation between images

Features such as noses, eyes, and mouths common among face images such that similarities may be collapsed leaving only the differences.

Each eigenvector (eigenface) represents a principal component such that any training image may be reconstructed with a weighted linear combination of the eigenvectors.
Principal Component Analysis

- Each eigenvector contributes more or less to each original training image.
- Right proportions or weights must be determined to reconstruct an image.
  - Done by projecting each training image onto all N eigenvectors to form a weight vector
  - Each training image will have its own characteristic weight vector.
- These weight vectors can be used not only for image reconstruction, but also for image recognition.
  - Done by projecting each test image to form its weight vector.
- Test weight vector is then compared with the weight vectors of each training
  - Criteria used is the Euclidean distance between the weight vectors
  - Smallest Euclidean distance found of the weight vectors between the test image and all the training images is that of the recognized image.
PCA Derivation

- Assume initially there are \( n \) \( d \)-dimensional data vectors \( \mathbf{x}_1, \ldots, \mathbf{x}_n \)
- Want to find \( d \)-dimensional vector \( \mathbf{x}_0 \) that best represents \( \mathbf{x}_1, \ldots, \mathbf{x}_n \)
- Done through minimizing error function \( E \), the sum of squared difference:

\[
E(\mathbf{x}_0) = \sum_{i=1}^{n} |\mathbf{x}_0 - \mathbf{x}_i|^2 \tag{1}
\]
PCA Derivation (continued)

- It can be shown that:

\[
E(x_0) = \sum_{i=1}^{n} |x_0 - x_i|^2 = \sum_{i=1}^{n} |(x_0 - m) - (x_i - m)|^2
\]

\[
= \sum_{i=1}^{n} |x_0 - m|^2 - 2 \sum_{i=1}^{n} (x_0 - m)^T (x_k - m) + \sum_{i=1}^{n} |x_k - m|^2
\]

\[
= \sum_{i=1}^{n} |x_0 - m|^2 - 2 (x_0 - m)^T \sum_{i=1}^{n} (x_k - m) + \sum_{i=1}^{n} |x_k - m|^2
\]

\[
= \sum_{i=1}^{n} |x_0 - m|^2 + \sum_{i=1}^{n} |x_i - m|^2 \quad (2)
\]
PCA Derivation (continued)

- Since $\sum |x_i - m|^2$ is independent of $x_0$
- Clearly, $E(x_0)$ is minimized for $x_0 = m$ (mean)
PCA Derivation (continued)

- Mean $\mathbf{m}$ does not provide much information, better to project data onto line passing through $\mathbf{m}$:
  \[
  \mathbf{x} = \mathbf{m} + a_i \cdot \mathbf{e}
  \]  
  (3)
  
  $\mathbf{e}$ is unit vector in direction of line, and $a_i$ is distance factor.

- Applying (3) to (2) gives
  \[
  E(a_1, \ldots, a_n, \mathbf{e}) = \sum_{i=1}^{n} |(\mathbf{m} + a_i \cdot \mathbf{e}) - \mathbf{x}_i|^2
  \]
  \[
  = \sum_{i=1}^{n} |a_i \cdot \mathbf{e} - (\mathbf{x}_i - \mathbf{m})|^2 
  \]  
  (multiply out)
  \[
  = \sum_{i=1}^{n} a_i^2 \cdot |\mathbf{e}|^2 - 2 \cdot \sum_{i=1}^{n} a_i \cdot \mathbf{e}^t \cdot (\mathbf{x}_i - \mathbf{m}) + \sum_{i=1}^{n} |(\mathbf{x}_i + \mathbf{m})|^2
  \]  
  (4)
PCA Derivation (continued)

- To minimize error function $E(a_1, \ldots, a_n, e)$, derivative
  \[ \frac{dE}{da_i} = 2 \cdot a_i \cdot |e|^2 - 2 \cdot e^t \cdot (x_i - m) \]
  is set to 0, giving
  \[ a_i = e^t \cdot (x_i - m) \quad (5) \]
  - Geometrically, $a_i$ is the result of projecting data onto line in
direction of $e$ passing through mean $m$
- Applying (5) to (4) gives (note that $|e| = 1$ since $e$ is unit vector)
  \[ E(e) = \sum a_i^2 - 2 \cdot \sum a_i^2 + \sum |(x_i + m)|^2 \]
  \[ = -\sum a_i^2 + \sum |(x_i + m)|^2 = -\sum [e^t \cdot (x_i - m)]^2 + \sum |(x_i + m)|^2 \]
  \[ = -\sum e^t \cdot (x_i - m) \cdot (x_i - m)^t \cdot e + \sum |(x_i + m)|^2 \]
  \[ = -e^t \cdot S \cdot e + \sum |(x_i + m)|^2 \quad (6) \]
PCA Derivation (continued)

- Maximize $\mathbf{e}^t \cdot \mathbf{S} \cdot \mathbf{e}$ to minimize (6), where $\mathbf{S}$ is $(n - 1)$ times covariance matrix ($n - 1$ can be factored out)
- For optimization use Lagrange function:
  
  $L(x, \lambda) = f(x) + \lambda \cdot g(x)$ where $g(x) = 0$ and derivative $dL / de$
  
  $= df / de + d(\lambda \cdot g) / de = 0$

- So $L(e, \lambda) = e^t \cdot \mathbf{S} \cdot e - \lambda (e^t \cdot e - 1) = 0$,
  
  and derivative $dL / de = 2 \cdot \mathbf{S} \cdot e - 2 \cdot \lambda \cdot e = 0$, and

  $\mathbf{S} \cdot e = \lambda \cdot e$  \hspace{1cm} (7)
PCA Derivation (continued)

- Based on analysis, want to find largest eigenvector $e$ and corresponding eigenvalue $\lambda$ to minimize (6).
- Recall initially that there are $n$ $d$-dimensional data vectors $x_1, \ldots, x_n$.
- Can apply (7) and extend (1) for dimension $c$ up to $d$-dimensional projections to obtain

$$E(a_i, e) = \sum_{i=1}^{n} \left| \left( \sum_{j=1}^{c} a_{ij} \cdot e_j \right) - (x_i - m) \right|^2$$

(8)
PCA Derivation (continued)

- To minimize (8), must determine orthogonal eigenvectors \{ \mathbf{e}_1, \ldots, \mathbf{e}_c \} of \mathbf{S} for each basis \( i \)
- Scalars \( a_{ij} \) are the principal components for basis \( i \)
Jacobian Transformation of a Symmetric Matrix

- To solve $S \cdot e = \lambda \cdot e$ for eigenvalues and corresponding eigenvectors, need a computationally feasible algorithm.
- Assume symmetric real matrix $A$ such that $A = A^T$ ($a_{ij} = a_{ji}$).
- Jacobi Method applies sequence of transformations $T_{pq}$ to $A$ until
  \[ A' = T_{pq}^T \cdot A \cdot T_{pq} \quad (9) \]
  is diagonal (i.e. all elements except on diagonals are zero).
Jacobian Transformation of a Symmetric Matrix (continued)

Given $T_{pq} = $

- Subscripts $p$ and $q$ refer to the rows and columns of scalars $c$ and $s$
- All diagonal elements are unity except for scalars $c$
- All off-diagonals are zero except for scalars $s$
To carry out multiplication of (8), the following equations apply:

\[ a'_{rp} = c \cdot a_{rp} - s \cdot a_{rq} \quad (r \neq p, r \neq q) \quad (9a) \]
\[ a'_{rq} = c \cdot a_{rq} + s \cdot a_{rp} \quad (r \neq p, r \neq q) \quad (9b) \]
\[ a'_{pp} = c^2 \cdot a_{pp} + s^2 \cdot a_{qq} - 2 \cdot s \cdot c \cdot a_{pq} \quad (9c) \]
\[ a'_{qq} = s^2 \cdot a_{pp} + c^2 \cdot a_{qq} + 2 \cdot s \cdot c \cdot a_{pq} \quad (9d) \]
\[ a'_{pq} = (c^2 - s^2) \cdot a_{pq} + s \cdot c \cdot (a_{pp} - a_{qq}) \quad (9e) \]

Clearly only rows p and q, and columns p and q are changed for \( A' \)
Jacobian Transformation of a Symmetric Matrix (continued)

- $T_{pq}$ can be regarded as plane rotation matrix
- Rotation $\theta_{pq}$ is defined by
  \[
  \frac{c^2 - s^2}{2 \cdot s \cdot c} = \frac{a_{qq} - a_{pp}}{2 \cdot a_{pq}} \tag{10}
  \]
- Thus $\theta_{pq}$ attempts to zero-out $a'_{pq}$
  with $c = 1 / \sqrt{\theta_{pq}}$ and
  $s = \text{sgn}(\theta) / (|\theta| + \sqrt{\theta^2 + 1})$
- Assuming $a'_{pq} = 0$ and applying (9e) to (9c),
  we have $a'_{pp} = a_{pp} - t \cdot a_{pq}$
Using the same logic, we arrive at the following:

\[
\begin{align*}
a'_{qq} &= a_{qq} + t \cdot a_{pq} \\
a'_{rp} &= a_{rp} - s \cdot (a_{rq} + \tau \cdot a_{rp}) \\
a'_{rq} &= a_{rq} - s \cdot (a_{rp} + \tau \cdot a_{rq})
\end{align*}
\]

where \( \tau = \frac{s}{1 + c} \)

The idea is to zero-out \( a'_{rp} \) and \( a'_{rq} \) and allow \( a'_{pp} \) and \( a'_{qq} \) to converge.
Jacobian Transformation of a Symmetric Matrix (continued)

- When $A' = Z^T \cdot A \cdot Z$ becomes diagonal, $Z = T_1 \cdot T_2 \cdot T_3 \ldots$ where $T_i$ are the planar rotations
- Eigenvectors are located in columns of $Z$
- For each planar rotation, $Z' = Z \cdot T_i$, where
  
  \[
  z'_{rs} = z_{rs} \\
  z'_{rp} = c \cdot z_{rp} - s \cdot z_{rq} \\
  z'_{rq} = s \cdot z_{rp} + c \cdot z_{rq},
  \]

- Stopping criteria must be designed and tuned to achieve reasonable results and run-time
Eigenfaces for Digital Image Recognition

- Digital face image $I(x, y)$ is 2-dimensional $N$ by $N$ array of 8-bit intensity values (0 to 255)
- $I(x, y)$ is thus of $N^2$ dimensional space
- First “stretch” out $N$ by $N$ image array so that it is essentially 1 by $N^2$, and “stack” $M$ images on top of each other
- Formed image matrix $\Gamma$ of size $M$ by $N^2$
Eigenfaces for Digital Image Recognition (continued)

- We have image matrix $\Gamma = \{ \Gamma_1, \Gamma_2, \ldots, \Gamma_M \}$ and
  mean image $\Psi = (1 / M) \cdot \sum_{i=1}^{M} \Gamma_i$

- Adjusted image $\Phi_i = \Gamma_i - \Psi$ and covariance
  matrix $C = (1 / M) \cdot \sum_{i=1}^{M} \Phi_i \cdot \Phi_i^T$

- The Eigenvectors $v = \{ v_1, v_2, \ldots, v_M \}$ calculated from $C$ are applied to $u_i = v_i \cdot \Phi$ to obtain $i = 1$ to $M$ Eigenfaces, where $\Phi = \{ \Phi_1, \Phi_2, \ldots, \Phi_M \}$
Eigenfaces for Digital Image Recognition (continued)

- Training images $\Phi_i$ are projected onto Eigenfaces to obtain a set of weights by
  \[ \omega_i = u_i^T \cdot (\Gamma - \Psi) \]
- The set of weights formed are $\Omega^T = \{ \omega_1, \omega_2, \ldots, \omega_{M'} \}$ for all training images
- It should be noted that the subscript $M'$ denotes that the dominant $M'$ of $M$ Eigenvectors can be used for $M' \leq M$
Image recognition is performed using the following classification criteria:

Face class: $\varepsilon_i^2 = |(\Omega - \Omega_i)|^2$

where $\Omega_i$ is calculated by averaging Eigenface representation for small number of images per class

Face Space: $\varepsilon^2 = |\Phi \cdot \Phi_i|^2$

where $\Phi_i = \sum_{i=1}^{M} \omega_i \cdot u_i$
Eigenfaces for Digital Image Recognition (continued)

- \( \varepsilon_i \) classifies according to face class (which person is recognized) and \( \varepsilon \) classifies according to face space (whether image is a face or not)
- Therefore there are four classification categories:
  - \((\varepsilon_i < \theta_1 \text{ and } \varepsilon < \theta_2)\) Near face space and near known face class
  - \((\varepsilon_i > \theta_1 \text{ and } \varepsilon < \theta_2)\) Near face space but not near face class
  - \((\varepsilon_i < \theta_1 \text{ and } \varepsilon > \theta_2)\) Not near face space and near known face class
  - \((\varepsilon_i > \theta_1 \text{ and } \varepsilon > \theta_2)\) Not near face space and not near face class
- \( \theta_1 \) and \( \theta_2 \) are appropriate thresholds that must be tweaked for a given training data set to achieve maximum recognition performance
Software Implementation

- Software implemented in Visual C++ 7.0 (C++ in .Net framework) using MFC for GUI functionality and support
- Support for C++ matrix operations provided by source code (Matrix TCL Lite) downloaded from [http://www.techsoftpl.com](http://www.techsoftpl.com)
- Algorithms (Jacobian Transformations) for calculating and sorting eigenvalues and eigenvectors obtained from [Press88]
Graphical User Interface (Main Module)

Figure 1: GUI
Key GUI Functionality

- It should be noted that the code uses a specific directory structure
  - Each person (class) has a separate directory with a number of images of different facial poses of that particular person
  - Training data and test data directories each contain a number of classes (directories)
- Load classes Button – goes through training directory and loads all images of all classes
Key GUI Functionality (continued)

- Update Eigenspace Button – calculates and stores all Eigenvalues and corresponding Eigenvectors for all images in training data directory
- Min. % of Eigenvectors List Box – specifies the specified top fraction of dominant Eigenvectors to keep in calculations
- Face Space Threshold List Box – number that must be tweaked for a given set of training data to maximize the classifier’s ability to distinguish non-face images
Key GUI Functionality (continued)

- Class Threshold List Box – specifies number that must be tweaked to maximize classifier’s ability to recognize images belonging to a certain class (person)
  - Class and face space threshold values are not independent; changing one affects the other
- Test Button – attempts to classify all images (images not contained in corresponding classes in training data) of test data and generates and displays recognition performance statistics
- Reconstruct Button – attempts to reconstruct specified image using calculated (trained) Eigenvectors
Digital Image Data

- Image data was obtained from http://www.uk.research.att.com/facedatabase.html
- This database contains grayscale images of 40 different people using different facial poses
- The images have been normalized in terms of lighting, noise, image size, and face cropping (face brought to front to fill image space)
- Tests use 5 different facial poses from 20 subjects
- Image files in .pgm file format
The .pgm image file format used is extremely simple and easy to work with:

- 1 image per file
- First line is “P2” to signify ASCII number representation
- Each pixel in file represented as ASCII decimal #
- Each pixel has white space before and after it
- No line longer than 70 characters
Test Results

- Training images consists of 40 classes (persons) with each class consisting of 5 different facial poses
- Test images consists of the same 40 classes but with different facial poses as that of the training images
Facial Recognition Test Results (continued)

Figure 4: Classification statistics of test images
Facial Recognition Test Results (continued)

- Face class threshold chosen specifically to be greater than the calculated maximum face class value, for example:
  - Previous slide shows max. face class value to be 3601.88 on the first run
  - Second run, increased face class threshold to 3700 to cover all matched images at the expense of increasing wrongly classified images

- Increase face space threshold to large number so that all images will be considered a face
  - Face space threshold used in conjunction with face class threshold to classify non-face images as well
  - Since it is assumed that all images hereon will be faces, the face space threshold will not be used
Face Reconstruction Results

Figure 5: Reconstruction of training image using all Eigenvectors (perfect reconstruction)
Figure 6: Reconstruction of training image using top 3/4 of dominant Eigenvectors
Face Reconstruction Results (continued)

Figure 7: Reconstruction of training image using top half of dominant Eigenvectors
Figure 8: Reconstruction of training image using top 1/4 of dominant Eigenvectors
Mean Image

Figure 8: Mean image of training image set
Conclusion

- The Eigenface approach works reasonably well for images normalized with respect to in-plane orientation, lighting, noise, face cropping, etc.
- However, analysis suggests that PCA is not effective as a feature-based technique.
- In other words, PCA may discard certain features that may be critical in distinguishing between classes.
References

- [Smith02] A Tutorial on Principal Components Analysis. Lindsay I. Smith. 