Face Recognition Using Eigenfaces

Matthew A. Turk and Alex P. Pentland

Presented by Pundik Dmitry
IDC, March 16, 2005

Main Goals

- Detection of faces and personal recognition
- Near-realtime operation
- Automatically learning new faces
- Insensitivity to small changes
- Simplicity

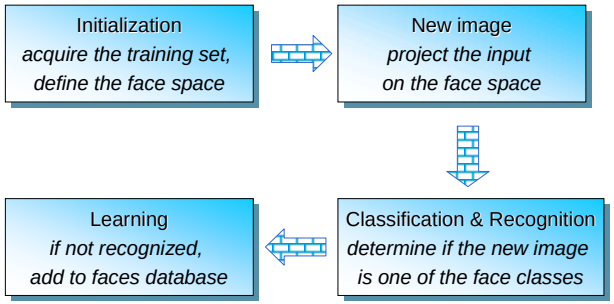
Constrains

- Only frontal view
- No scaling

Analysis Method

- Work on a training set of images
- Select only the significant features, that span the face space – the eigenfaces
- Reduce dimensionality by using Principal Components Analysis
- Use Euclidean distance metric

Recognition Process



Eigenfaces

- Each image I(x,y) is NxN array, or a vector of dimension N^2

$$I_k = \begin{pmatrix} i_1 \\ i_2 \\ . \\ . \\ i_{N^2} \end{pmatrix}$$

- These vectors are **not** equally distributed in high dimension space
- **Eigenfaces** are K orthogonal vectors, that best describe the faces distribution

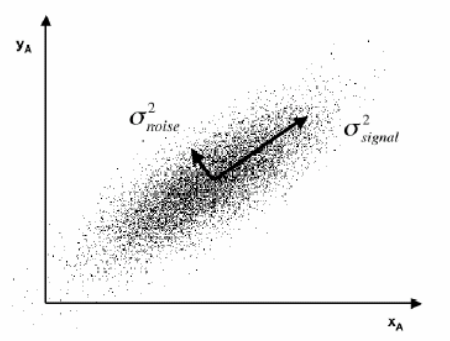
Faces

Average face

Principal Component Analysis

- Each face image is a random vector
- Each **component** in this vector is a **pixel**
- The most correlated components best describe the faces distribution
- Find the components that describe the **highest variance**

PCA - example



PCA – cont.

- **Covariance** matrix:

$$C = E\{(I-\mu_i)(I-\mu_i)^T\} = \begin{pmatrix} c_{11} & \dots & c_{1N^2} \\ \dots & \dots & \dots \\ c_{N^2} & \dots & c_{N^2N^2} \end{pmatrix}$$

$$I = [I_1, I_2, \dots, I_M]$$

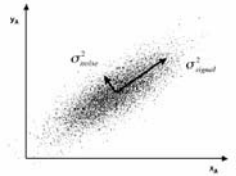
- c_{kk} - variance of variable k
- c_{ij} - covariance of variables i and j
- μ_i - mean for each components on all the images

PCA – cont.

- Now, let's calculate the *eigenvalues* and *eigenvectors* of the covariance matrix
- Take the first K eigenvectors, corresponding to the **highest** eigenvalues
- Those vectors have the direction of **largest variance** of the data
- Our space has now K dimensions

PC Vectors

- The Principal Component vectors are orthonormal vectors
- The PC vectors transform the face to a new representation
- They serve as a basis for the face space



PCA - cont

- When a new random vector I arrives:
  - Project it on the K selected eigenvectors
  - Work in a K-dimensional space
- Using PCA, we preserve as much information as possible, in the mean-square sense

Back To Faces

- Use PCA, to reduce dimensionality…
- Training set of images: $\Gamma_1, \Gamma_2, ..., \Gamma_M$
- Average image: $\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n$
- Covariance matrix: $C = AA^T$

  where: $A = [\Phi_1, \Phi_2, ..., \Phi_M]$

  $\Phi_i = \Gamma_i - \Psi$

Calculating Eigenfaces

- We have to find eigenvectors and eigenvalues of C
- The size of C is $N^2 \times N^2$
- The eigenvectors corresponding to the biggest eigenvalues are taken
- Those are the eigenfaces!

Calculations Optimization

- How can we speed up?

$$M << N^2$$

- We have only M training images:
- There can be at most M-1 useful (nonzero) eigenvectors

Calculations Optimization – cont.

- AA^T has dimensions of $N^2 \times N^2$
- But, $A^T A$ has dimensions of $M \times M$
- Consider the eigenvectors v_j of $A^T A$ and the eigenvalues μ_j of $A^T A$

$$A^T A v_j = \mu_j v_j$$

$$\Downarrow$$

$$AA^T A v_j = \mu_j A v_j$$

Av_j : the eigenvalues of AA^T

Calculating Eigenfaces – Final

- Use a new matrix L

$$L = A^T A \qquad C = AA^T$$

C – the covariance matrix

- Eigenfaces:

$$u_i = \sum_{k=1}^{M} v_{ik} \Phi_k \qquad i = 1,2,...,M$$

u_i : the i^{th} eigenface

v_{ik} : the k^{th} value of the i^{th} eigenvector

Eigenfaces Space

- The found eigenfaces span the face space
- Actually we need less then M eigenvetors
- The space has M' dimensions

Faces Projection

- Each new face is projected onto the face space, using the eigenfaces basis

$$\omega_k = u_k^T (\Gamma - \Psi)$$

$$k = 1...M'$$

$$\Omega = [\omega_1, \omega_2, ..., \omega_{M'}]$$

- Ω - the projected face
- ω_k - contribution of a single eigenface

Face Projection - example

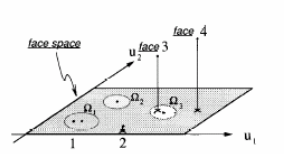
Face Classification

- Select a number of face classes:
- Average a few faces (maybe one), representing a class – a single individual
- Project the average face onto the face space

Face Classification – cont.

- Find the Euclidian distance between the new face projection and the class projection

$$\varepsilon_k = \left\| (\Omega - \Omega_k) \right\| < \theta$$



Ω, Ω_k - projected face, and projected class face

ε_k - Euclidian distance

θ - threshold

Face Space Revisited

- If the face not recognized, but close to the face space, it can be added
- New category of unknown individuals
- The system learns new faces

Recognition Procedure Summary

1. Collect a set of face images.
2. Calculate the L matrix, and find its eigenvalues and eigenvectors.
3. Select the M' eigenfaces.
4. For each new face, project it onto the face space, and find the distance to all the known face classes.
5. If the person is recognized, the new face may enter the database as this person.
6. If the face was not recognized, it may enter the database as new face class.

Faces Detection

- How to detect faces in a scene?
- "Faceness" of an image:

  The distance between the image to its projection onto the face space
- Projected faces do not change radically, but projected non-faces appear different

Faces Detection – cont.

- For every sub-image, calculate its faceness

$$\Phi = \Gamma - \Psi$$

$$\Phi_{face} = \sum_{i=1}^{M'} \omega_k u_k$$

$$\varepsilon = \left\| \left(\Phi - \Phi_{face} \right) \right\|$$

Φ - mean - adjusted image

Φ_{face} - projection onto face space

ε - the faceness of the image

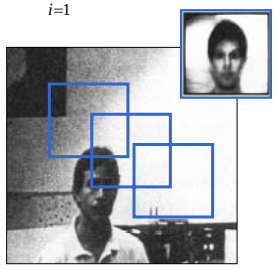
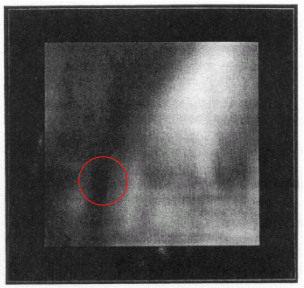


Image Faceness - example



The dark areas indicate the presence of face

Recognition Experiments

- The authors used database of 2,500 faces
- Each training set contained 16 faces

	Light	Orientation	Size
100% Acc.	19% Unkn.	39% Unkn.	60% Unkn.
20% Unkn.	100% Acc.	94% Acc.	74% Acc.

Recognition Experiments – cont.

- The changing lighting causes relatively few errors
- Size changes drop the performance dramatically
- Multiscale approach is needed!

Limitations and Further Research

- Range of aspects: defining new face classes for each person
- Clustering the “unknown” faces, to create a new identified individual
- Optimizing speed in images analyzing
- Noise or occlusion cause performance degradation

Conclusion – Eigenfaces

- Practical solution
- Nearly real-time operation
- Relatively simple
- Works well in a constrained environment
- Proved math background