Face recognition using Eigenfaces

Based on the Principal component analysis (PCA) theory:
http://en.wikipedia.org/wiki/Principal_component_analysis

Eigenfaces

http://en.wikipedia.org/wiki/Eigenface

Eigenfaces is the name given to a set of eigenvectors (https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors) when they are used in the computer vision problem of human face recognition.[1] 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.[2] The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images. The eigenfaces themselves form a basis set of all images used to construct the covariance matrix. This produces dimension reduction by allowing the smaller set of basis images to represent the original training images. Classification can be achieved by comparing how faces are represented by the basis set.

History

The Eigenface approach began with a **search for a low-dimensional representation of face images**. Sirovich and Kirby (1987) showed that Principal Component Analysis could be used on a collection of face images to form a **set of basis features**. These basis images known as **Eigenpictures**, could be **linearly combined** to **reconstruct images in the original training set**. If the **training set** consists of ***M* images**, principal component analysis could form **a basis set of *N* images**, where $N < M$. The reconstruction error is reduced by increasing the number of eigenpictures, however the number needed is always less than M. For example if you need to generate a number of N eigenfaces for a training set of M face images , **you can say**

that each face image can be made up of "proportions" of all this K "features" or eigenfaces : **Face image1 = (23% of E1) + (2% of E2) + (51% of E3)+...+(1% En).**

In 1991 M. Turk and A. Pentland expanded these results and presented the **Eigenface method of face recognition**.[3] As well as designing a system for automated face recognition using eigenfaces, they showed a way of calculating the eigenvectors of a covariance matrix in such a way as to make it possible for computers at that time to perform eigen-decomposition on a large number of face images. Face images usually occupy a high dimensional space and conventional principal component analysis was intractable on such data sets. Turk and Pentlands paper demonstrated ways to extract the eigenvectors based on matrices sized by the number of images rather than the number of pixels.

Once established, the eigenface method was expanded to include methods of preprocessing to improve accuracy.[4] Multiple manifold approaches were also used to build sets of eigenfaces for different subjects[5][6] and different features, such as the eyes.[7]

Eigenface generation

A **set of eigenfaces** can be generated by performing a mathematical process called principal component analysis (PCA) on a large set of images depicting different human faces. Informally, **eigenfaces can be considered a set of "standardized face ingredients**", derived from statistical analysis of many pictures of faces. Any human face can be considered to be a combination of these standard faces. **For example, one's face might be composed of the average face plus 10% from eigenface 1. 55% from eigenface 2, and even -3% from eigenface 3**. Remarkably, it does not take many eigenfaces combined together to achieve a fair approximation of most faces. Also, because a person's face is not recorded by a digital photograph, but instead as just a list of values (one value for each eigenface in the database used), **much less space is taken for each person's face**.

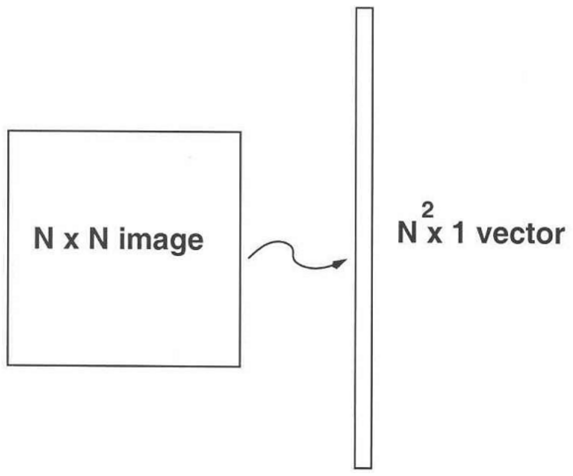
The eigenfaces that are created will appear as light and dark areas that are arranged in a **specific pattern**. This pattern is how different features of a face are singled out to be evaluated and scored. There will be a **pattern** to evaluate symmetry, if there is any style of **facial hair**, **where the hairline is**, or **evaluate the size of the nose or mouth**. Other eigenfaces have patterns that are less simple to identify, and the image of the eigenface may look very little like a face.

The technique used in creating eigenfaces and using them for recognition is also used outside of facial recognition. **This technique is also used for** handwriting analysis, lip reading, voice recognition, sign language/hand gestures interpretation and medical imaging analysis. Therefore, some do not use the term eigenface, but prefer to use '**eigenimage**'.

Practical implementation

To create a set of eigenfaces, one must:

1. **Prepare a training set of face images**. 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 a **common** pixel **resolution ($N \times N$).**
2. Each image is treated as **one vector**, simply by concatenating the rows of pixels in the original image, resulting in a **single row with $N \times N$ elements (Γ_i)**. For this implementation, it is assumed that all images of the training set are stored in a single matrix, where **each column of the matrix is an image**.



3. The **average image Ψ** has to be calculated:

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

4. The **average image Ψ** has to be subtracted from each original image **(Γ_i)**.

$$\Phi_i = \Gamma_i - \Psi$$

5. Calculate the **eigenvectors** (u_i) and eigenvalues of the covariance matrix **C** of **A**. 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 **C** efficiently, without ever computing **C** explicitly, as detailed below.

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T \quad (N^2 \text{x} N^2 \text{ matrix})$$

$$\text{where } A = [\Phi_1 \ \Phi_2 \cdots \Phi_M] \quad (N^2 \text{x} M \text{ matrix})$$

6. **Computing the eigenvectors**

Performing PCA directly on the covariance matrix of the images is often computationally infeasible. If small, say 100 × 100, greyscale images are used, each image is a point in a 10,000-dimensional space and the covariance matrix **C is a matrix of 10,000 × 10,000 = 10^8 elements**. However the rank of the covariance matrix is limited by the number of training examples: if there are N training examples, there will be at most $N - 1$ eigenvectors with non-zero eigenvalues. **If the number of training examples is smaller than the dimensionality of the images**, the principal components can be computed more **easily as follows**:

Let A be the matrix of preprocessed training examples, where each column contains one mean-subtracted image. The **covariance matrix** can then be computed as $\mathbf{C} = \mathbf{AA}^T$ and the eigenvector decomposition of **C** is given by:

$$\mathbf{C}u_i = \mathbf{AA}^T u_i = \mu_i u_i$$

However $\mathbf{AA}^T$ is a large matrix (N^2 x N^2), and if instead we take the eigenvalue decomposition of:

$$\mathbf{A}^T\mathbf{A}v_i = \mu_i v_i$$

then we notice that by pre-multiplying both sides of the equation with **A**, we obtain:

$$\mathbf{AA}^T\mathbf{A}v_i = \mathbf{CA}v_i = \mu_i \mathbf{A}v_i$$

$$u_i = \mathbf{A}v_i \quad \Rightarrow \quad \mathbf{C}u_i = \mu_i u_i$$

Meaning that, if $\mathbf{v}_i$ is an eigenvector of $\mathbf{A}^T\mathbf{A}$, then $\mathbf{u}_i = \mathbf{Av}_i$ is an eigenvector of **C**.

The **M eigenvalues** (eigenvectors) of $\mathbf{A}^T\mathbf{A}$ correspond to the **M largest eigenvalues** (eigenvectors) of $\mathbf{C=AA}^T$

⇒ Compute the **M** best eigenvectors of $\mathbf{AA}^T$: $u_i = \mathbf{A}v_i$

Note1: If we have a training set of **M = 300** images of 100 × 100 (**NxN**) pixels, the matrix $\mathbf{A}^T\mathbf{A}$ is a 300 × 300 matrix (it can have up to **M = 300 eigenvectors** /

eigenvalues), which is much more manageable than the (**N**^{2} x **N**^{2} = **10,000** × **10,000**) covariance matrix **C** (it can have up to **N**^{2}=**10,000 eigenvectors** / eigenvalues). Notice however that the resulting vectors **u**$_i$ are not normalised; if normalisation is required it should be applied as an extra step such as || **u**$_i$ ||=1.

Note 2: Each eigenvector represents a **direction** in the (**N**x**N**) image space

7. From the **M** best eigenvectors keep only **K** eigenvectors (corresponding to the K largest **eigenvalues ≈ Eigenfaces / Eigenpictures**)

These **eigenfaces** can now be used to represent both existing and new faces: we can project a new (mean-subtracted) image on the eigenfaces and thereby record how that new face differs from the mean face. **The eigenvalues associated with each eigenface represent how much the images in the training set vary from the mean image in that direction**. We lose information by projecting the image on a subset of the eigenvectors, but we minimize this loss by keeping those eigenfaces with the largest eigenvalues. For instance, if we are working with a 100 x 100 image, then we will obtain 10,000 eigenvectors. In practical applications, most faces can typically be identified using a projection on between **100 and 150 eigenfaces**, so that most of the 10,000 eigenvectors can be discarded.

Representing faces onto this basis

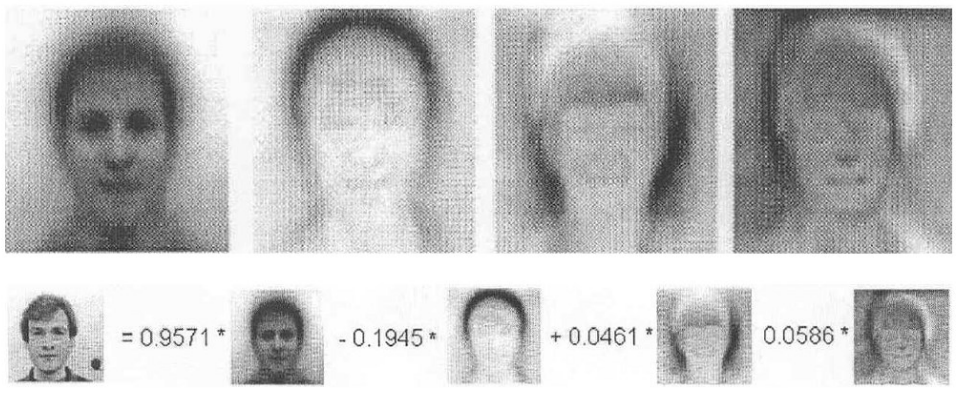
Each face (minus the mean) **Φ*i*** in the training set can be represented as a linear combination of the best K eigenvectors:

$$\hat{\Phi}_i - mean = \sum_{j=1}^{K} w_j u_j, \quad (w_j = u_j^T \Phi_i)$$

u$_j$ - eigenfaces / eigenvectors (vectors of size N^2)

Each normalized training face Φi is represented in this basis by a $\Omega_i = \begin{bmatrix} w_1^i \\ w_2^i \\ \dots \\ w_K^i \end{bmatrix}, \quad i = 1, 2, \dots, M$

vector:



Use in facial recognition

Facial recognition was the source of motivation behind the creation of eigenfaces. For this use, eigenfaces have advantages over other techniques available, such as the system's speed and efficiency. As eigenface is primarily a dimension reduction method, a system can represent many subjects with a relatively small set of data. As a face recognition system it is also **fairly invariant to large reductions in image sizing**, however it begins to **fail considerably when the variation between the seen images and probe image is large**.

To recognize faces, gallery images, those seen by the system, are saved as collections of weights describing the contribution each eigenface has to that image. When a **new face is presented to the system for classification**, its **own weights are found by projecting the image onto the collection of eigenfaces**. This provides a **set of weights describing the probe face**. These weights are then classified against all weights in the gallery set to find the closest match. A **nearest neighbour method** is a simple approach for finding the Euclidean Distance between two vectors, where the minimum can be classified as the closest subject.(Turk & Pentland 1991, p. 590)

The **weights of each gallery image only convey information describing that image**, **not that subject**. An image of one subject under **frontal lighting** may have very different weights to those of the same subject under strong **left lighting**. This limits the application of such a system. Experiments in the original Eigenface paper presented the following results: an average of 96% with light variation, 85% with orientation variation, and 64% with size variation. (Turk & Pentland 1991, p. 590).

Other methods

Various extensions have been made to the eigenface method such eigenfeatures. This method combines facial metrics (measuring distance between facial features) with the eigenface representation. Another method similar to the eigenface technique is '**fisherfaces**' which uses (LDA) Linear discriminant analysis.[8] This method for facial recognition **is less sensitive to variation in lighting and pose** of the face than using eigenfaces. Fisherface utilizes labeled data to retain more of the class specific information during the dimension reduction stage.

A further alternative to eigenfaces and fisherfaces is the active appearance model. This approach use an Active Shape Model to describe the outline of a face. By collecting many face outlines, Principal Component Analysis can be used to form a basis set of models which, encapsulate the variation of different faces.

http://www.vision.jhu.edu/teaching/vision08/Handouts/case_study_pca1.pdf

http://www.face-rec.org/algorithms/PCA/jcn.pdf

http://www.cs.columbia.edu/~belhumeur/journal/fisherface-pami97.pdf

FACE RECOGNITION HOMEPAGE

http://www.face-rec.org/algorithms/
http://www.face-rec.org/databases/
http://www.face-rec.org/source-codes/

Implementations:

Philipp Wagner, **FaceRecognizer - Face Recognition with OpenCV** (Eigenfaces, Fisherfaces, LBP)

http://docs.opencv.org/trunk/modules/contrib/doc/facerec/

http://docs.opencv.org/trunk/modules/contrib/doc/facerec/facerec_tutorial.html

Ver. MTALAB (Eigenfaces, Fisherfaces):

https://github.com/bytefish/facerecognition_guide/raw/master/facerec_octave.pdf

PCA / Eigenfaces (MATLAB): http://www.cs.ait.ac.th/~mdailey/matlab/

PCA on the FERET Dataset (EigenFaces): http://www.face-rec.org/source-codes/pca.zip