

---

# ECSE 415- Introduction to Computer Vision



PCA and Eigenfaces

---

# Face Recognition

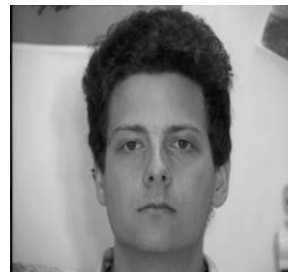
- Problem statement:

*Identify face from a  
database of known faces.*

?



From MIT database



---

# Typical face recognition scenarios

---

- Verification: a person is claiming a particular identity; verify whether that is true
  - E.g., security
- Closed-world identification: assign a face to one person from among a known set
- General identification: assign a face to a known person or to “unknown”

---

# Face Recognition

---

- Why is this problem hard?

---

# Face recognition: Challenges

---

- What's hard?
  - Faces are similar, same set of features (e.g. eyes, nose, mouth) in same configuration.
  - Changes in expression, lighting, age, **occlusion**, **viewpoint**

---

# What makes face recognition hard?

---

## Expression



---

# What makes face recognition hard?

---

## Lighting



---

# What makes face recognition hard?

---

## Occlusion





---

# What makes face recognition hard?

---

## Viewpoint



---

# Face recognition: Challenges

---

- Have to find representation that captures enough features to identify face from database of faces.

---

# Early Attempts

---

- Automated face recognition is a relatively new concept.

---

# Early Attempts

---

- Automated face recognition is a relatively new concept.
- In 1960s, the first semi-automated system required the administrator to locate features (such as eyes, ears, nose, and mouth) on the photographs before it calculated distances and ratios to a common reference point, which were then compared to reference data.

---

# Early Attempts

---

- Automated face recognition is a relatively new concept.
- In 1960s, the first semi-automated system required the administrator to locate features (such as eyes, ears, nose, and mouth) on the photographs before it calculated distances and ratios to a common reference point, which were then compared to reference data.
- In the 1970s, another system used 21 specific subjective markers such as hair color and lip thickness to automate the recognition.

---

# Early Attempts

---

- Early attempts –
  - Based on object recognition literature: attempt to learn underlying models that define faces
- Popular approach: model relations between face features.
- Ignores important information about texture and shape
- Doesn't capture realistic differences between faces

---

# Appearance-based Techniques

- Model-based approaches to recognition are difficult in cases where:
  - the scene is complex,
  - the relationship between the data and model less-obvious,
  - the requirements are more general.
- Appearance-based techniques were developed for these cases.

---

# Appearance-based Techniques

- Representations are *viewer-centered* and the data/model relationship made explicit through empirical evidence gathered during training.
- These techniques (Kirby and Sirovich, Nayar and Murase, Turk and Pentland) became popular due to their:
  - simplicity
  - fast indexing time
  - Accuracy in constrained environments
  - Relative insensitivity to small perturbations in the image



---

# Appearance-based Techniques

---

- Appearance-based strategies often employ a method referred to as *Principal Component Analysis* (PCA) to find an optimal lower dimensional subspace in which to represent data of high dimension.

---

# Appearance-based Techniques

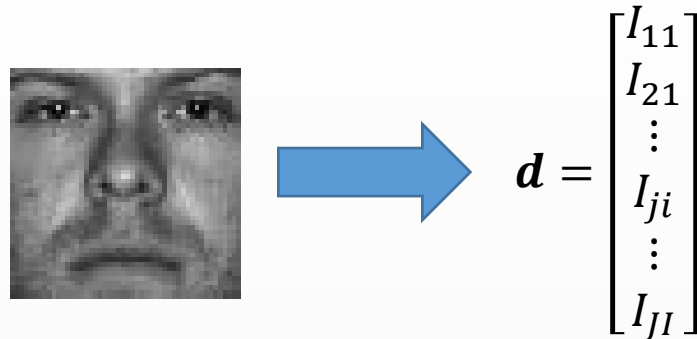
---

- Typically, an image field is represented as a vector of very high dimensionality (equal to the number of pixels).
- In the case of appearance-based representations, each image can be represented by a single vector  $\mathbf{d}$  of length  $M$ .

---

Representing an object (e.g. a face image)  
by a single vector  $\mathbf{d}$  of length  $M$

---



- A face image is represented as a vector of very high dimensionality (equal to the number of pixels)
  - $M = Height \times Width$  (total number of pixels)
    - E.g.: a  $100 \times 100$  pixels face image  $\rightarrow$  a vector with  $M = 10,000$  dimensions
- Note that each element of  $\mathbf{d}$  represents the coefficient for each dimension

---

# Appearance-based Techniques

---

Let  $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N\}$  denote the set of  $N$  such images in the database.

# Forming the data matrix $D$

$D = \{d_1, d_2, \dots, d_N\}$  is the matrix whose columns are the  $N$  image vectors in the database



$d_i$



$D = \{d_1, d_2, \dots, d_N\}$

---

# The space of all face images

- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - a  $100 \times 100$  pixels face image = 10,000 dimensions
  - Slow and lots of storage
- But very few 10,000-dimensional vectors are valid face images

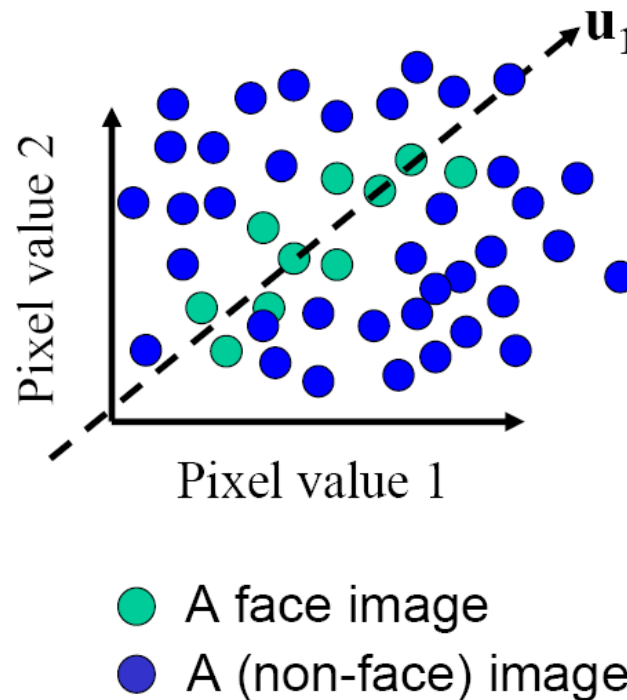
---

# The space of all face images

- If we only consider images that are valid faces, such images may lie in a small *subspace* of the set of all images
- If this *subspace* is a hyperplane spanned by  $S$  vectors, then we can represent any point that lies on this plane with only a set of  $S$  coefficients, where  $S \ll M$

# The space of all face images

- Eigenface idea: construct a low-dimensional *linear subspace* that best explains the *variation* in the set of face images





---

# Linear Models

---

- We will now investigate *deterministic* models of the following form:

$$I(x, y) = \sum_i a_i \Psi_i(x, y)$$

---

# Linear Models

---

$$I(x, y) = \sum_i a_i \Psi_i(x, y)$$

- These models are known as *linear models* due to their linearity in the weights  $a_i$ .
- The functions  $\Psi_i$  are typically nonlinear, and are often referred to as *basis* functions, when they come from a basis set.
- An image can then be represented as a summation of a set of coefficients x basis functions.

---

# Linear Models

---

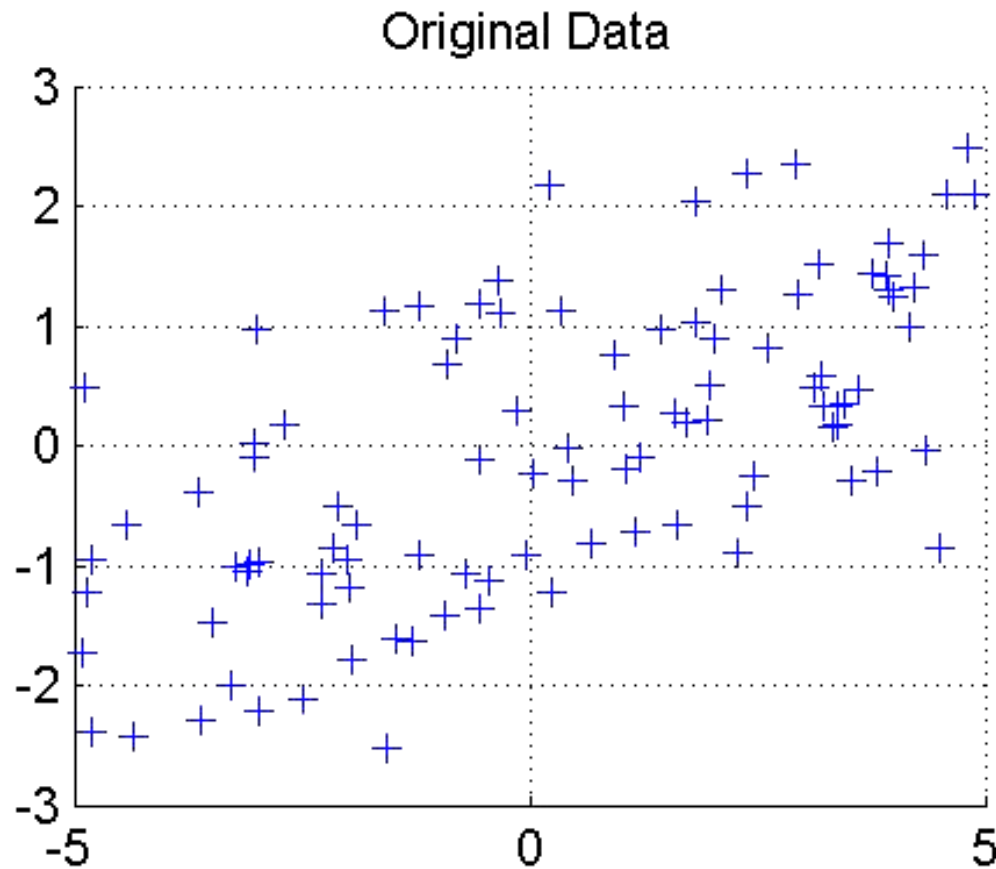
- An example of such a linear model is probably familiar to you, that of a Fourier representation, where the basis functions are sine and cosine functions.
- Linear models are most effective when only a few of their associated basis functions are needed to capture the variation of  $I$  with respect to  $x$  and  $y$ .

---

# Principal Components Analysis

---

- *Principal Components Analysis* (or **PCA**) provides a very popular linear model – one that captures most of the variation in the data (e.g. an image).



- *For example, the data points in the figure above seem to be concentrated along a tilted axis.*

---

# Principal Components Analysis

- How can we determine the lower-dimensional subspace?
- One approach is to find a single axis that best models the data points  $\mathbf{d}_i$ .
  - This will give a one-dimensional model for the data.
- Once we find this axis, we can try to find another axis which best models the deviation of the data points from the first axis.
- And so on... for as many axes as we want
  - up to the dimensionality of the data space

---

# Principal Components Analysis

Assume that the axis is defined by a unit vector  $v$ .  
There are two main approaches for finding  $v$   
given a set of data points:

---

# Principal Components Analysis

1 - We can try to find the vector  $v$  that minimizes the **square modeling error**:

$$E_R = \sum_i (d_i - v^T d_i v)^2$$

In the above equation  $v^T d_i v$  represents the projection of the data point onto the axis  $v$ .



---

# Principal Components Analysis

2 - We can try to find the vector  $v$  that maximizes the **variance of the data** after projection onto the axis  $v$  (assuming the projection has zero mean) :

$$\sigma^2 = \frac{1}{N-1} \sum_i (d_i^T v)^2$$

The result is the same for both of these approaches.

---

# Principal Components Analysis

Let us consider the second approach.

First, we will rewrite the equation for the variance as a Matrix equation

(remember that the data points  $\mathbf{d}$  and the axis  $\mathbf{v}$  are vectors):

$$\sigma^2 = \frac{1}{N-1} \sum_i (d_i^T \mathbf{v})^2 = \frac{1}{N-1} \mathbf{v}^T \mathbf{D} \mathbf{D}^T \mathbf{v}$$

where  $\mathbf{D}$  is the matrix whose columns are the data point vectors.

---

# Principal Components Analysis

The unit vector  $v$  that maximizes this variance can be found by solving the following constrained optimization problem:

$$\begin{aligned} v &= \arg \max_v (v^T D D^T v + \lambda(1 - v^T v)) \\ &= \arg \max_v (v^T [D D^T - \lambda I] v - \lambda) \end{aligned}$$

The second term in the above equation is a Lagrange Multiplier term. The Lagrange Multiplier  $\lambda$  must be determined so as to enforce the constraint (that  $v$  is a unit vector, or that  $v^T v = 1$ ).

---

# Principal Components Analysis

$$v = \arg \max_v (v^T [DD^T - \lambda I] v - \lambda)$$

We can find the optimal  $v$  by taking the derivative of the argument in the above equation and equating to zero:

$$\frac{\partial}{\partial v} (v^T [DD^T - \lambda I] v - \lambda) = 0$$

---

# Principal Components Analysis

$$\frac{\partial}{\partial \mathbf{v}} (\mathbf{v}^T [\mathbf{D}\mathbf{D}^T - \lambda \mathbf{I}] \mathbf{v} - \lambda) = 0$$

This gives:

$$2[\mathbf{D}\mathbf{D}^T - \lambda \mathbf{I}] \mathbf{v} = 0$$

or

$$[\mathbf{D}\mathbf{D}^T] \mathbf{v} = \lambda \mathbf{v}$$

---

# Principal Components Analysis

$$\left[ DD^T - \lambda I \right] v = 0$$

or

$$\left[ DD^T \right] v = \lambda v$$

This can be identified as an eigenvalue problem, where  $v$  is an eigenvector of the square, symmetric matrix  $DD^T$  and  $\lambda$  is the corresponding eigenvalue.

We choose the eigenvalue so as to make the eigenvector have unit length.

---

# Principal Components Analysis

The most common application of PCA is to dimensionality reduction.

It can be shown that the modeling error, for a given number of model dimensions,  $K$ , is minimized by choosing the  $K$  principal components with the largest eigenvalues.

---

# Principal Components Analysis

This is based on the result that the modeling error is equal to the sum of the eigenvalues of the principal components that are not used in the model.

This is obviously minimized when we choose the components with the largest eigenvalues.



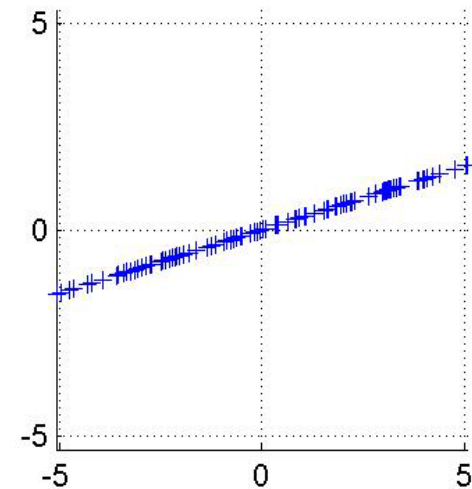
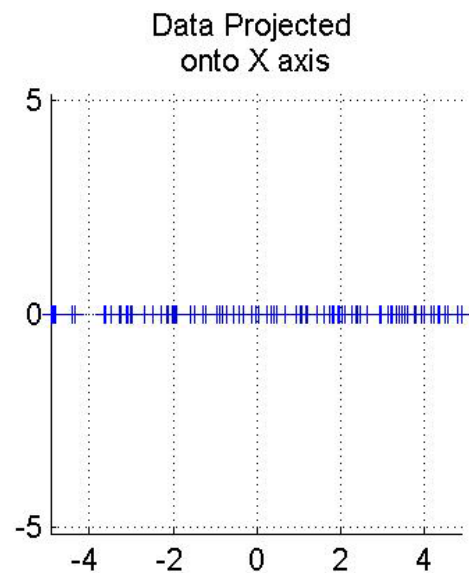
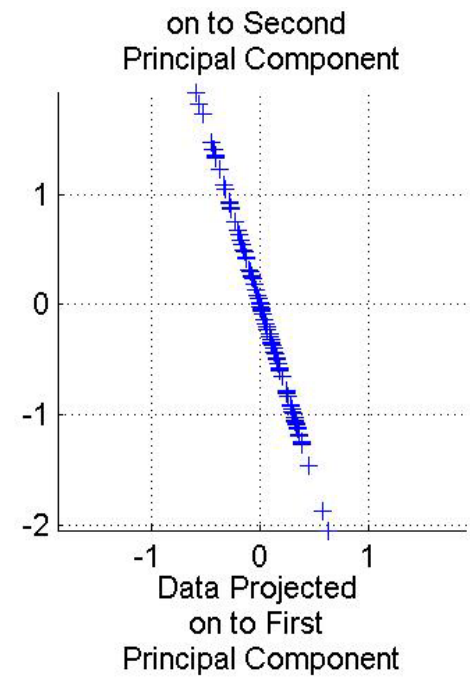
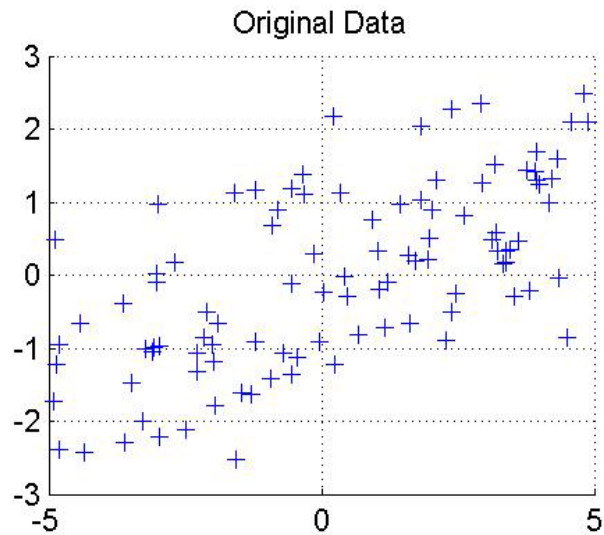
---

# Principal Components Analysis

The variance of the data along each principal component direction can be seen to be proportional to the associated eigenvalue.

$$\sigma^2 = \frac{1}{N-1} v^T D D^T v = \frac{1}{N-1} v^T \lambda v = \frac{\lambda}{N-1}$$

Thus we can see that the principal component with the largest associated eigenvalue is the one that provides the largest variance in the projected data values.



Axis of Projection	RMS Reconstruction Error	Variance of Projected Values
1st P.C.	0.91	8.63
X-Axis	1.22	7.94
Y-Axis	2.82	1.51
2nd P.C.	2.94	0.0019

We can see that the first principal component is the one that maximizes the variance of the projected values and minimizes the reconstruction error.

The last principal component (second in this case) is the one that minimizes the variance and maximizes the reconstruction error.

---

# Karhunen-Loeve Transform

---

- We can transform the data to be represented in a new coordinate system by projecting the data onto each of the principal components in turn. The resulting set of numbers is the coordinates of the data in this new coordinate system.
- One of the significant properties of this change in coordinate system is that the covariance matrix of the data,  $DD^T$ , expressed in this new coordinate system, *diagonal*.

---

# Karhunen-Loeve Transform

---

- The data components in these new directions are *uncorrelated*.
- The process of transforming coordinates to obtain uncorrelated data components is often called *whitening*, in reference to the lack of correlation between components of white noise signals.
- The transformation of the data points into an uncorrelated set is often referred to as the *Karhunen - Loeve Transform*.

---

# Computation of Principal Components

---

- The principal components analysis as presented so far requires the formation of the covariance matrix  $C = DD^T$ , followed by finding the eigenvectors and eigenvalues of  $C$ .
- If each data vector point is  $M$ -dimensional and if we have  $N$  such vectors, then  $D$  is an  $M \times N$  matrix and  $C$  is  $M \times M$ .
- The size of  $C$  can therefore be quite large, making computation of the eigenvectors very demanding of computational resources.

---

# Computation of Principal Components

---

- If the number of data points is much less than the dimension of the data points (i.e.  $N \ll M$ ) we can convert the problem of finding the principal components into one that is much less computationally intensive.

---

# Computation of Principal Components

---

- To see this, pre-multiply the eigenvector equation by the transpose of the data matrix:

$$D^T (DD^T) \mathbf{v} = D^T (\lambda \mathbf{v})$$

- or

$$(D^T D)(D^T \mathbf{v}) = \lambda (D^T \mathbf{v})$$

- Let  $\hat{C} = (D^T D)$ , then it can be seen from the above equation that  $\hat{\mathbf{v}} = (D^T \mathbf{v})$  is an eigenvector of  $\hat{C}$  (with the same eigenvalues as  $C$ ).



---

# Computation of Principal Components

---

- We can therefore get the principle component  $v$  by pre-multiplying  $\acute{v}$  by  $D$ :

$$D\acute{v} = DD^T v = \lambda v$$

- Therefore

$$v = \frac{1}{\lambda} (D\acute{v})$$

where  $\acute{v}$  is an eigenvector of  $\acute{C}$ , with eigenvalue  $\lambda$ .

The eigenvalues of  $C$  are the same as those of  $\acute{C}$ .

---

# Computation of Principal Components

---

- This method of computing the Principal Components is sometimes called the *Snapshot Method* (name due to Sirovich and Kirby, 1986).
- The advantage of this approach is that the size of  $\hat{C}$  is  $N \times N$ , which can be much smaller than the size of  $C$ .
- The principal components can be seen in this view to be linear combinations of the data point vectors.

L. Sirovich, M. Kirby, *Low dimensional procedure for the characterization of human faces*, Journal of the Optical Society of America 4 (3) (1987) 519--524.

---

# Principal Components Analysis

---

- Summary:
  - Create matrix  $D$  whose columns are the image vectors
  - Compute  $DD^T$
  - Find the eigen vector and eigen values of  $DD^T$
  - We can compress the data by only using the top few eigenvectors
    - The eigenvectors with largest eigenvalues capture the most variation among training vectors

---

# Eigenfaces

---

- One of the first applications of PCA to Computer Vision problems was by Sirovich and Kirby, who used PCA to model images of human faces.

(L. Sirovich, M. Kirby, *Low dimensional procedure for the characterization of human faces*, Journal of the Optical Society of America 4 (3) (1987) 519--524.

M. Kirby and L. Sirovich. *Application of the karhunen-loeve procedure for the characterization of human faces*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(1):103-- 108, 1990. )

---

# Eigenfaces

---

- Sirovich and Kirby did not (in their early papers) specify any particular application of the models, but just demonstrate that the principal components provide an efficient basis for face images.
- They found that 50 Principal Components give an average modeling error of 3.68% over ten test images.

---

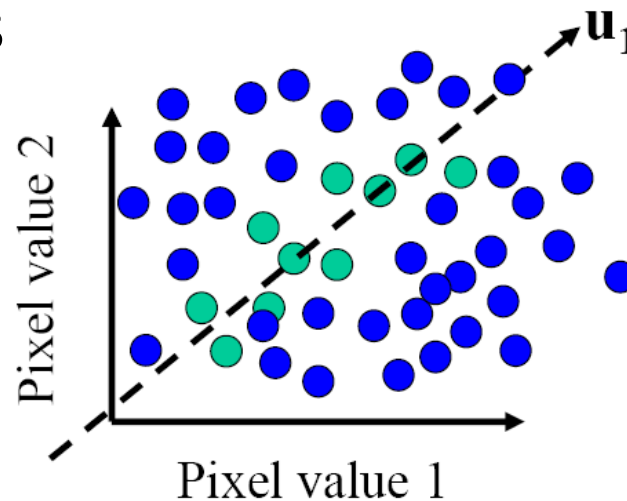
# Eigenfaces

---

- Claimed advantages:
  - Good for vague recognition that face is “known” or “unknown”.
  - Capture global, non-subjective features of faces.

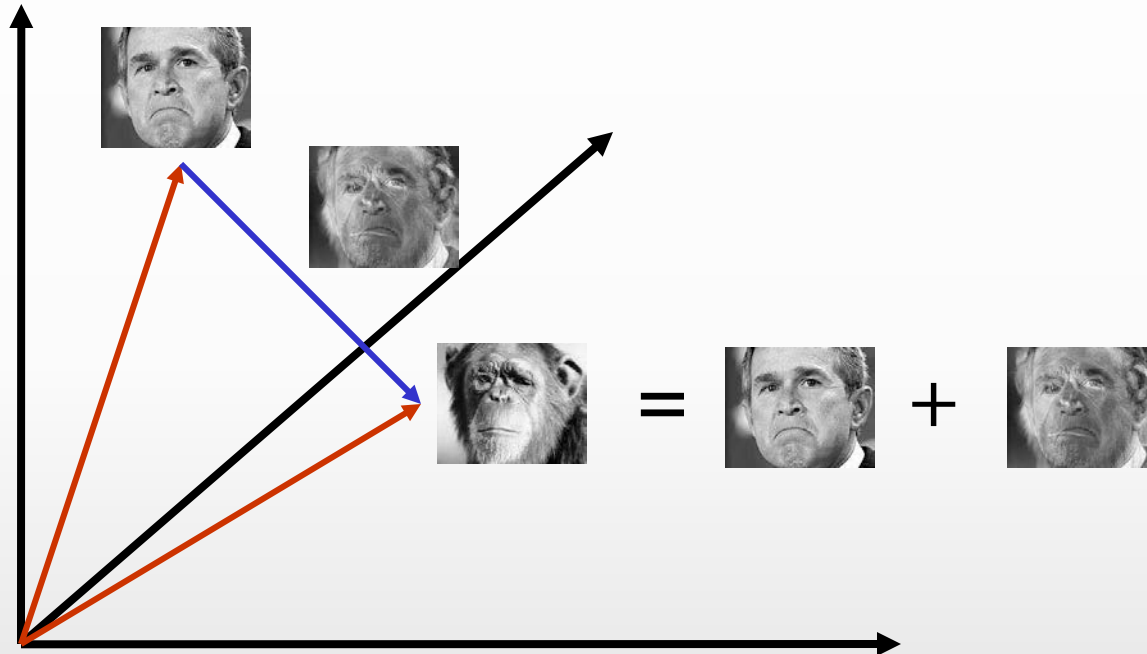
# The space of all face images

- Eigenface idea: construct a low-dimensional *linear subspace* that best explains the *variation* in the set of face images



- A face image
- A (non-face) image

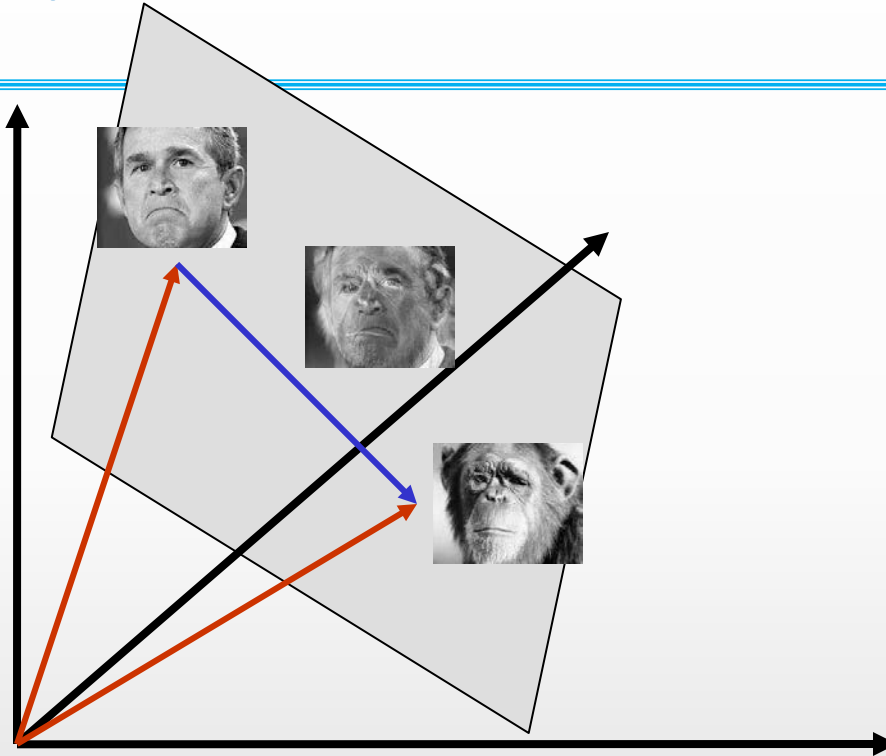
# The space of faces



- An image is a point in a high dimensional space



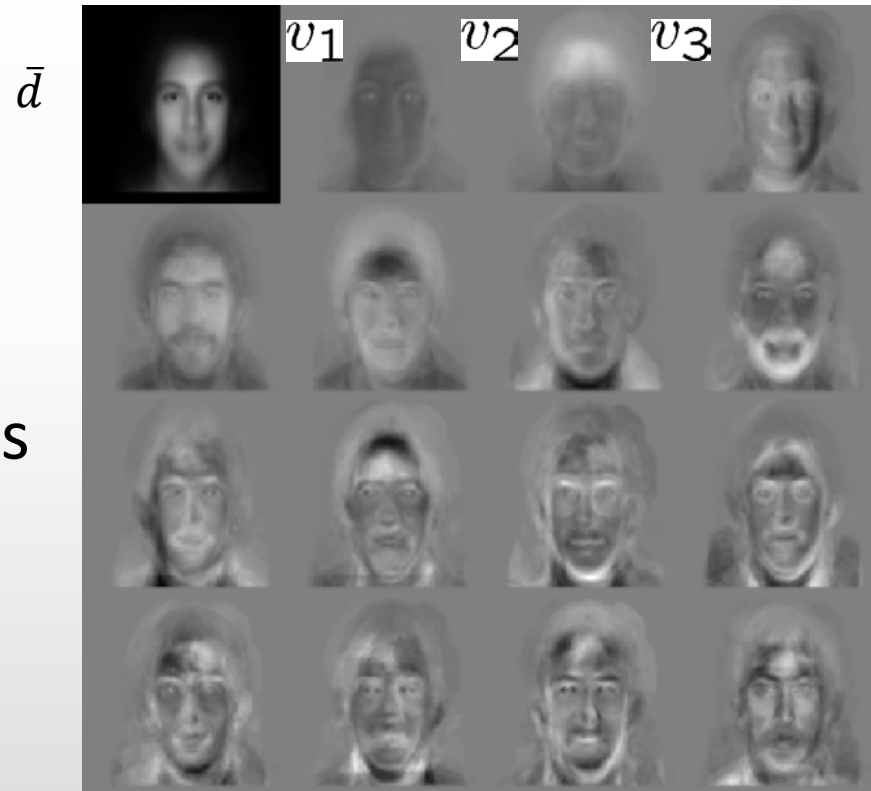
# The space of faces



- The set of faces is a subspace of the set of images
  - We can find the best subspace using PCA
  - This is like fitting a “hyper-plane” to the set of faces

# Eigenfaces

- PCA extracts the eigenvectors of  $DD^T$
- Gives a set of vectors  $v_1, v_2, v_3, \dots$
- Each one of these vectors is a direction in face space
- what do these look like?



# Visualization of eigenfaces

Principal component (eigenvector)  $v_k$



$\mu + 3\sigma_k v_k$



$\mu - 3\sigma_k v_k$

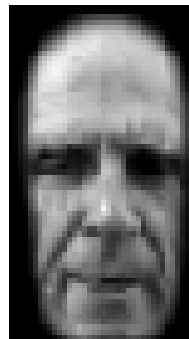


# Projecting onto the eigenfaces

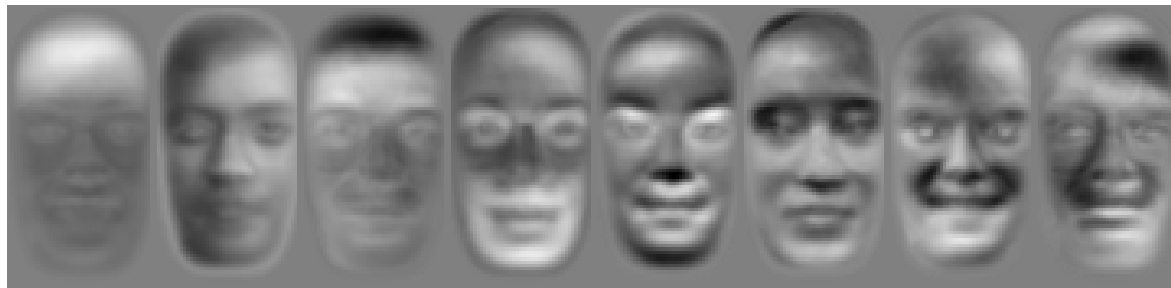
- The eigenfaces  $v_1, \dots, v_K$  span the space of faces
  - A face is converted to eigenface coordinates by

$$\mathbf{x} \rightarrow (\underbrace{(\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_1}_{a_1}, \underbrace{(\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_2}_{a_2}, \dots, \underbrace{(\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_K}_{a_K})$$

$$\mathbf{x} \approx \bar{\mathbf{x}} + a_1 \mathbf{v}_1 + a_2 \mathbf{v}_2 + \dots + a_K \mathbf{v}_K$$



$\mathbf{X}$



$a_1 \mathbf{v}_1 \quad a_2 \mathbf{v}_2 \quad a_3 \mathbf{v}_3 \quad a_4 \mathbf{v}_4 \quad a_5 \mathbf{v}_5 \quad a_6 \mathbf{v}_6 \quad a_7 \mathbf{v}_7 \quad a_8 \mathbf{v}_8$



---

# Representation and reconstruction

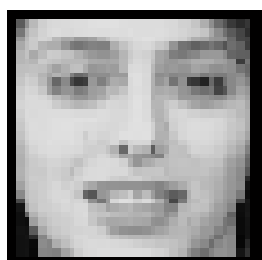
- Face  $\mathbf{x}$  in “face space” coordinates:



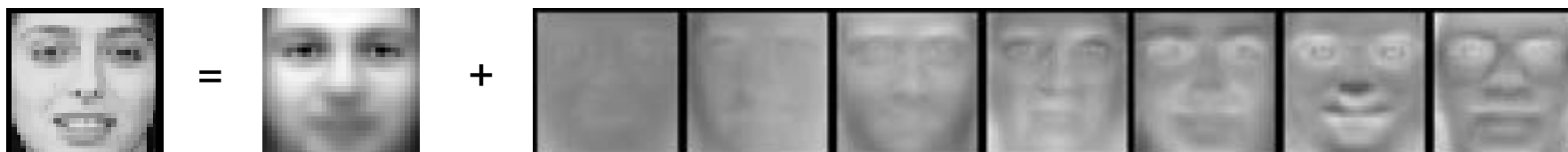
$$\begin{aligned}\mathbf{x} &\longrightarrow [\mathbf{u}_1^T (\mathbf{x} - \mu), \dots, \mathbf{u}_k^T (\mathbf{x} - \mu)] \\ &= w_1, \dots, w_k\end{aligned}$$

# Representation and reconstruction

- Face  $\mathbf{x}$  in “face space” coordinates:


$$\mathbf{x} \rightarrow [\mathbf{u}_1^T (\mathbf{x} - \mu), \dots, \mathbf{u}_k^T (\mathbf{x} - \mu)]$$
$$= w_1, \dots, w_k$$

- Reconstruction


$$\hat{\mathbf{x}} = \mu + w_1 u_1 + w_2 u_2 + w_3 u_3 + w_4 u_4 + \dots$$

---

# Reconstruction

---

- For an image that was not in the ensemble used to build the space, will the reconstruction be exact?
- For images used to build the space, if fewer eigenimages are used in the reconstruction, will the reconstruction be exact?

---

## Reconstruction

For an image that was not in the ensemble the reconstruction will not be exact, but may still be good.

Also, if fewer eigenimages are used in the reconstruction, the reconstruction will not be exact, even for images in the ensemble.



# Reconstruction

$K = 4$



$K = 200$



$K = 400$



After computing eigenfaces using 400 face images from  
ORL face database

---

# Reconstruction

The accuracy in the modeling of a given image depends somewhat on the number of images in the ensemble used to generate the principal components.

Clearly, if only a few images were used, they might not capture all of the variations found in face images in general.

---

# Reconstruction

Another way of looking at this is that images in the ensemble have a lower modeling error than images that are not in the ensemble.

So, increasing the number of images in the ensemble makes it more likely than an image outside the ensemble will resemble one that is inside, and therefore will have a relatively low modeling error.

---

# Dimensionality Estimation

An interesting and important problem involves whether we can determine the inherent dimensionality of the data.

Usually we would expect that this dimension is much lower than the dimensionality of the image (i.e. the number of pixels).

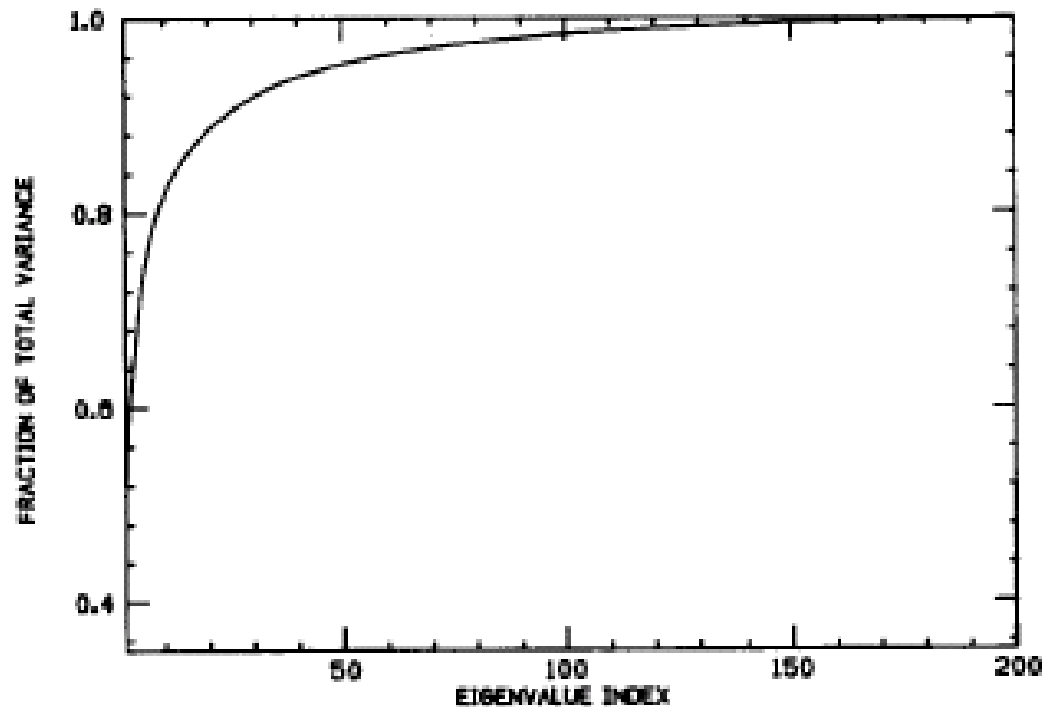


Fig. 3. Fraction of total variance  $q_N$  versus number of terms  $N$  in expansion.

The fraction of the total variance contained in the first  $N$  eigenimages is given by:

$$f_N = \frac{\sum_{i=1}^N \lambda_i}{\sum_{i=1}^{2M} \lambda_i}$$

The first 10 terms contain 82 percent of the variance, while the first 50 terms contain 95 percent.

---

# Dimensionality Estimation

One way to get an estimate of the inherent dimension of the data is to look at the ***spectrum*** of the data (defined as the eigenvalue as a function of the eigenvalue number).

If there is an identifiable cutoff in the spectrum we can associate the cutoff point with the dimension.

---

# Dimensionality Estimation

One approach to finding the cutoff value is to look at the normalized eigenvalue (normalized by the maximum eigenvalue).

The cutoff is taken to be at the smallest eigenvalue index for which the normalized eigenvalue is less than some threshold.

## Are face images 21 - dimensional?

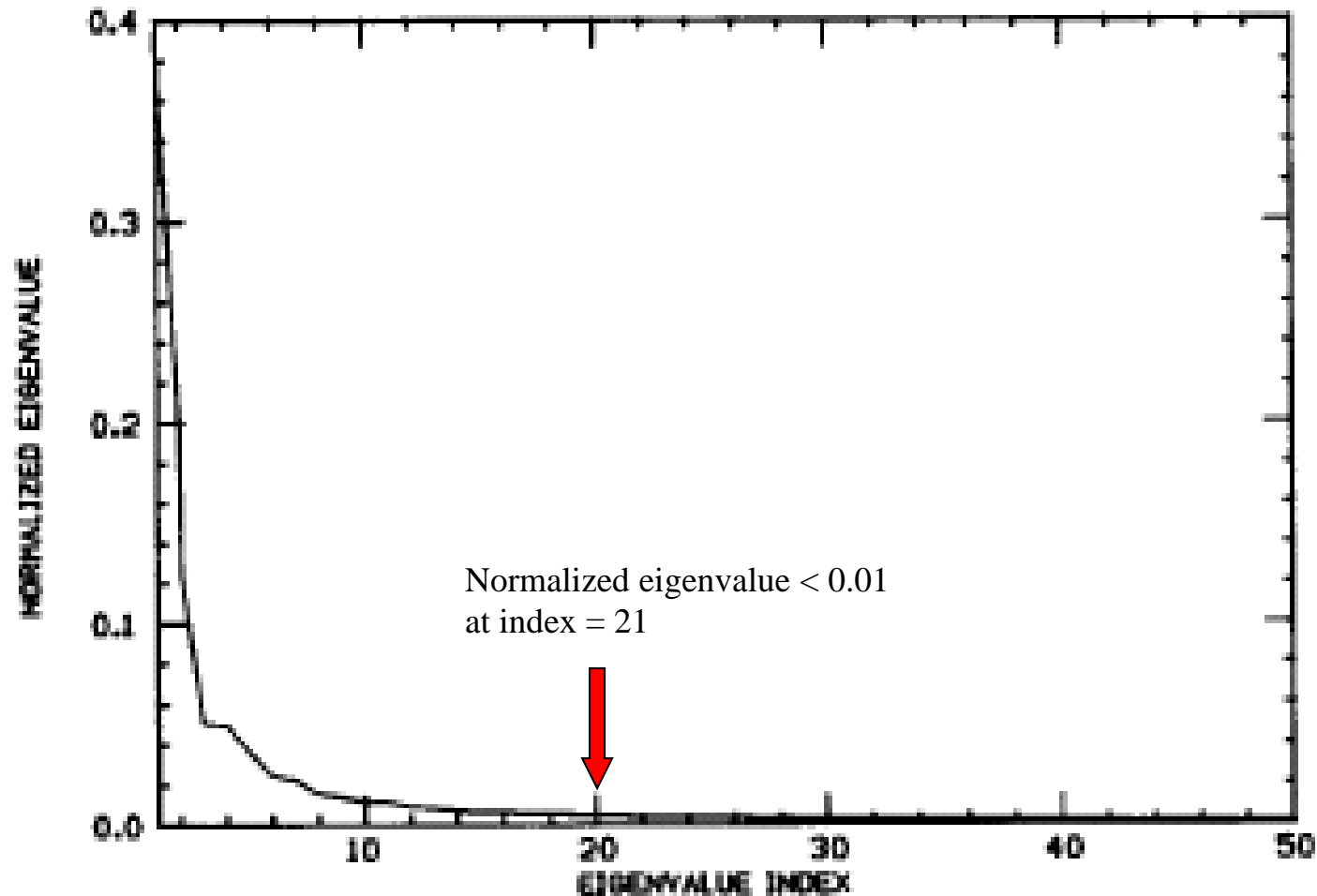


Fig. 4. Eigenvalues normalized by the maximum eigenvalue versus index.



---

# Dimensionality Estimation

Another approach is to look at the ratio of successive eigenvalues (or the difference between the log of successive eigenvalues).

We take the cutoff to be where this ratio is maximum.

Both of these methods are somewhat arbitrary.

---

# Face Recognition using Eigenfaces

Turk and Pentland use the eigenface approach of Sirovich and Kirby for the purposes of recognizing faces (rather than just encoding them).

M. Turk and A. Pentland, "Eigenfaces for recognition", *Journal of Cognitive Neuroscience*, Vol. 3, No. 1, pp. 71-86, Winter 1991.

M. Turk and A. Pentland, "Face recognition using Eigenfaces", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Maui, Hawaii, 1991.

---

# Face Recognition using Eigenfaces

Turk and Pentland use the eigenface approach of Sirovich and Kirby for the purposes of recognizing faces (rather than just encoding them).

Since recognition does not depend on highly accurate reconstruction, fewer eigenfaces are needed than for reconstruction. In the Turk and Pentland work only 7 eigenfaces were used.

M. Turk and A. Pentland, "Eigenfaces for recognition", *Journal of Cognitive Neuroscience*, Vol. 3, No. 1, pp. 71-86, Winter 1991.

M. Turk and A. Pentland, "Face recognition using Eigenfaces", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Maui, Hawaii, 1991.



**Figure 2.** Seven of the eigenfaces calculated from the input images of Figure 1.

---

# Face Recognition using Eigenfaces

The Turk and Pentland recognition algorithm works by projecting a given test image onto the (7 in their case) principal components (or eigenfaces).

This generates a 7-element vector of "face-space" coordinates that serve as a model for the input image.

This vector is compared (via a Euclidean norm in face-space) to a set of predetermined exemplar vectors, corresponding to previously seen face images.

---

# Face Recognition using Eigenfaces

The exemplar that is closest to the test image representation is taken to be the "recognized" face, as long as the distance is less than some threshold.

If there is no exemplar point closer than this threshold, the test image could be considered to be that of a new face, and a new exemplar created.

---

# Face Recognition using Eigenfaces

It might be the case that the input image is ***not of a face*** at all, in which case we don't want to generate a new exemplar.

How can we tell if an image is of a face at all?

---

# Face Recognition using Eigenfaces

It might be the case that the input image is ***not of a face*** at all, in which case we don't want to generate a new exemplar.

This can be avoided by computing the distance between the original image (minus the mean face image) and its reconstruction from the principal components.



---

# Face Recognition using Eigenfaces

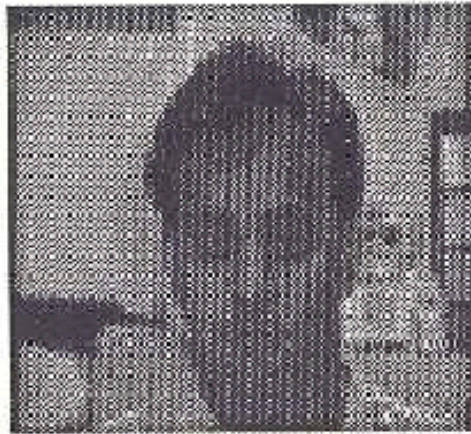
Since the PCs are modeling face-type images, a low reconstruction error means that the input looks something like a face, and we can create a new exemplar.

Otherwise, if the reconstruction error is too big, we can say that the input image is not that of a face, and do not create a new face exemplar.

original image

reconstructed image

Recon error =  
58.5



Recon error =  
5217.4



---

# Recognition with eigenfaces

- Algorithm
  1. Process the image database (set of images with labels)
    - Run PCA—compute eigenfaces
    - Calculate the  $K$  coefficients for each image
  2. Given a new image (to be recognized)  $\mathbf{x}$ , calculate  $K$  coefficients

$$\mathbf{x} \rightarrow (a_1, a_2, \dots, a_K)$$

3. Detect if  $\mathbf{x}$  is a face

$$\|\mathbf{x} - (\bar{\mathbf{x}} + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \dots + a_K\mathbf{v}_K)\| < \text{threshold}$$

4. If it is a face, who is it?
  - Find closest labeled face in database
  - nearest-neighbor in  $K$ -dimensional space

---

# If it is a face, who is it?

---

Recognize face by nearest neighbor in database



- The image  $d$  belongs to the  $k$ th subject

$$k = \operatorname{argmin}_i \|d - d_i\|$$



---

# Face Detection and Localization

Turk and Pentland also use their PCA face model to detect and localize faces in images.



---

# Face Detection and Localization

Turk and Pentland also use their PCA face model to detect and localize faces in images.

How?

---

# Face Detection and Localization

Turk and Pentland also use their PCA face model to detect and localize faces in images.

The idea is to look at sub-patches of images and project each sub-patch onto the face-space and compute the reconstruction error.

Patches for which the reconstruction error is low can be considered to contain a face, while those patches which give a large reconstruction error are taken to be non-face regions.

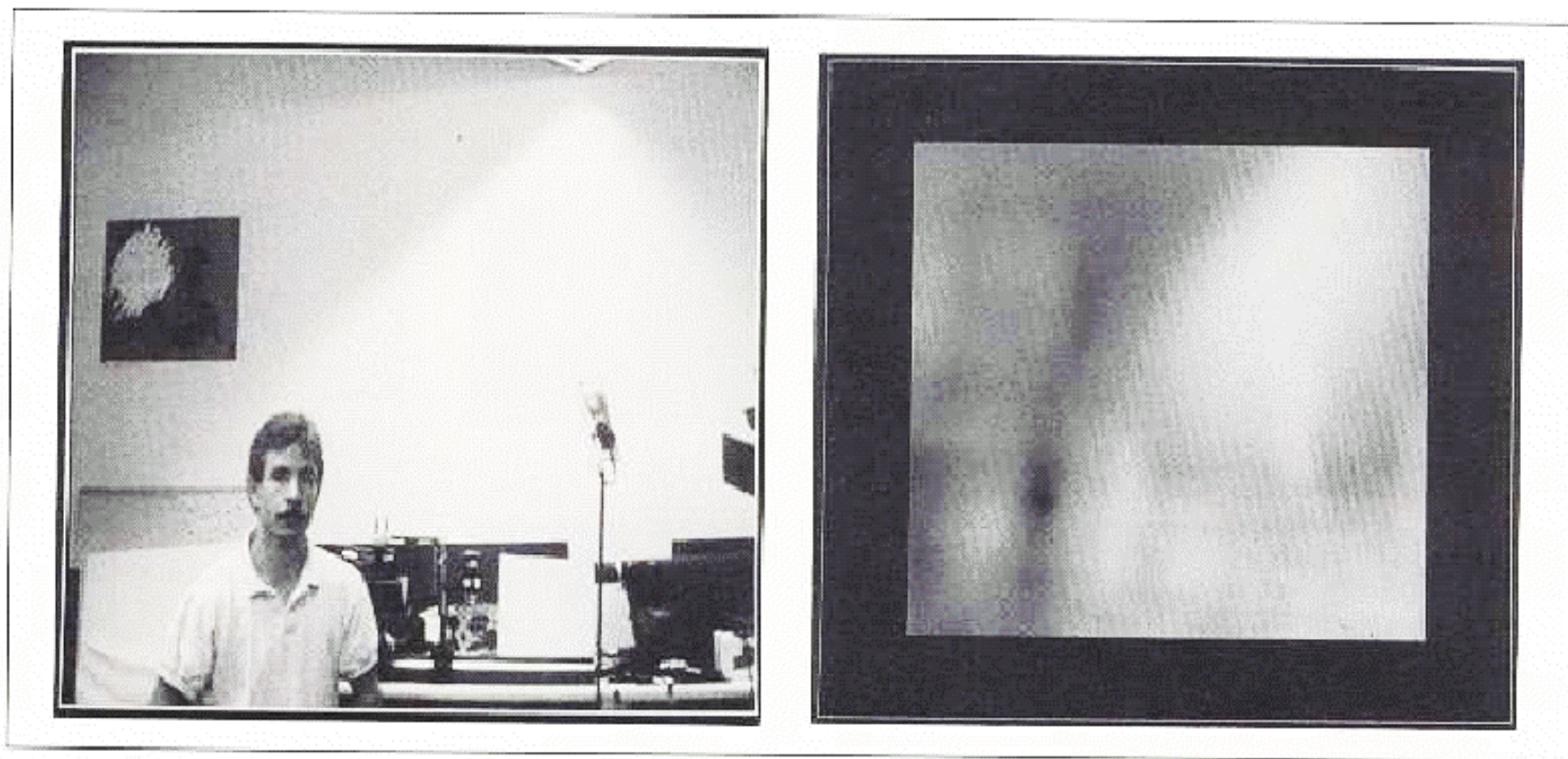


Figure 7. (a) Original image. (b) The corresponding face map, where low values (dark areas) indicate the presence of a face.



---

# Recognition from Partial Data

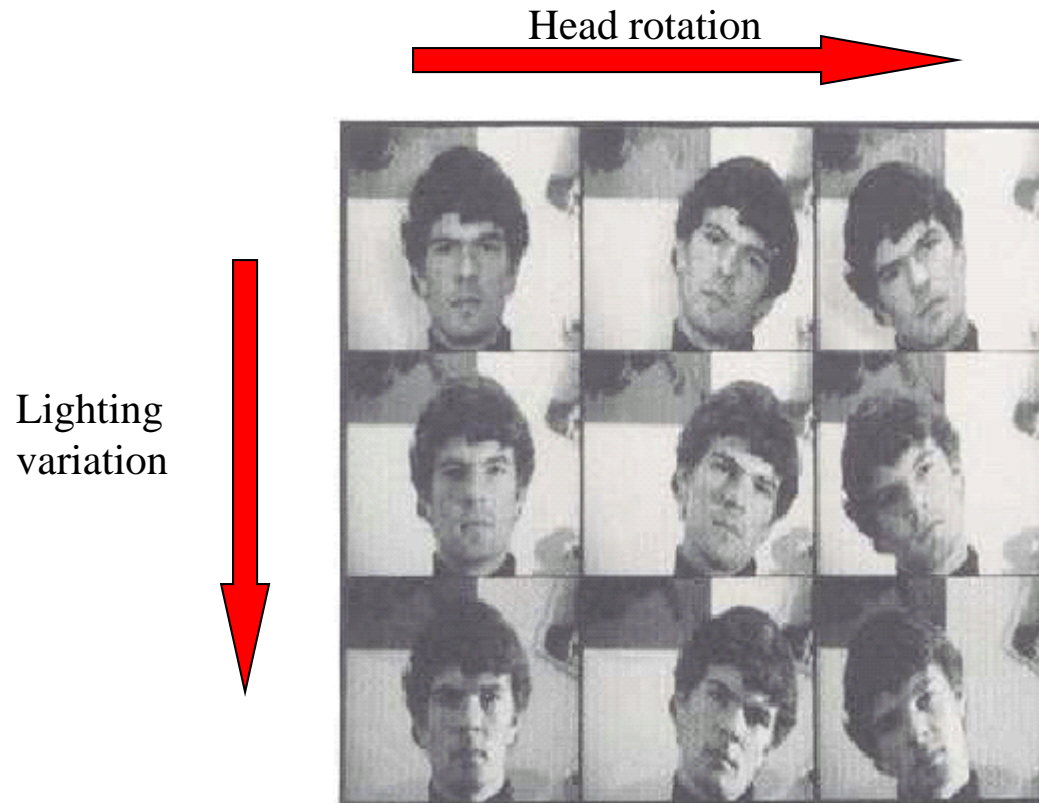
The Turk and Pentland approach can provide recognition in the presence of partial or noisy data. The image can also be reconstructed from the partial image, by using the face-space projection to define the face image.

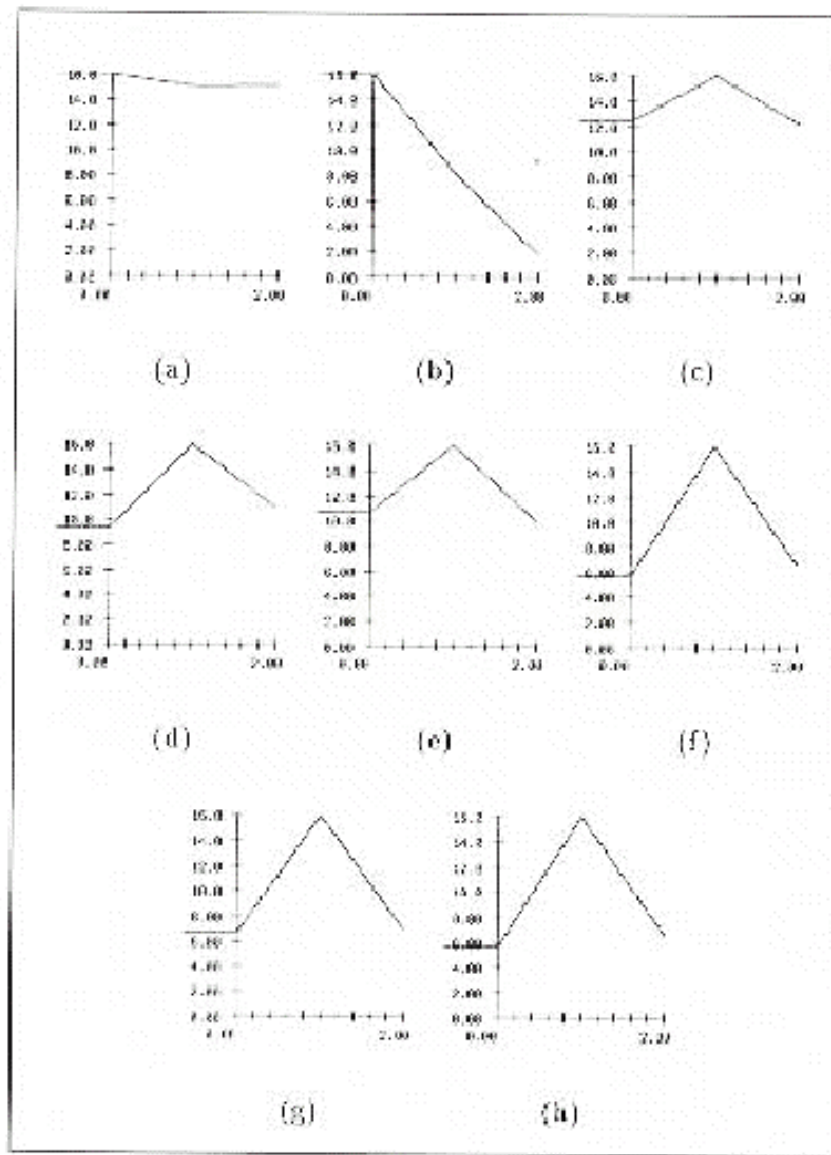


Figure 13. (a) Partially occluded face image and (b) its reconstruction using the eigenfaces.

# Invariance to Changes in Image Acquisition

How does the Turk and Pentland algorithm perform when images are taken under different conditions?





Falloff in classification performance for changes in various parameters:

- a) lighting
- b) head size (scale)
- c) head orientation
- d) orientation and lighting
- e) orientation and size (#1)
- f) orientation and size (#2)
- g) size and lighting (#1)
- h) size and lighting (#2)

Results are that the approach is quite invariant to minor changes in lighting variation, but greatly affected by scale and rotation changes.

Invariance could perhaps be achieved by taking images of faces over a greater range of configurations.

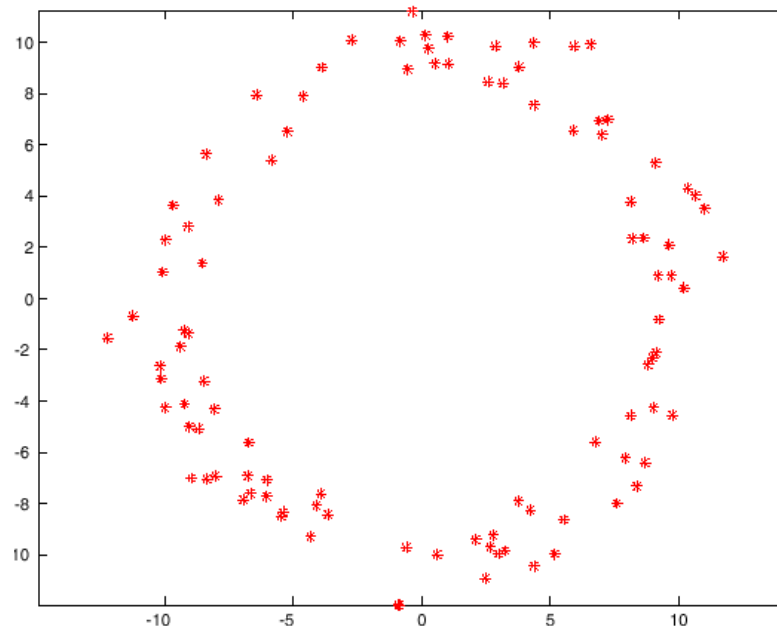
---

# Summary of Limitations

- **Background** changes cause problems
  - De-emphasize the outside of the face (e.g., by multiplying the input image by a 2D Gaussian window centered on the face).
- **Large light changes** degrade performance
  - Light normalization helps.
- Performance decreases quickly with changes to **face size**
  - Multi-scale eigenspaces.
  - Scale input image to multiple sizes.
- Performance decreases with changes to **face orientation** (but not as fast as with scale changes)
  - Plane rotations are easier to handle.
  - Out-of-plane rotations are more difficult to handle.

# Limitations

- PCA assumes that the data follows a Gaussian distribution (mean  $\mu$ , covariance matrix  $\Sigma$ )



The shape of this dataset is not well described by its principal components

# Limitations

- PCA is not always an optimal dimensionality-reduction procedure for classification purposes:

