Computer Vision
---*Face Recognition Using Principal Components Analysis (PCA)*

Dr. WU Xiaojun

2020.10.30

Face recognition

- What makes face detection and recognition hard?

Hilary

NCC, SSD,……?

Face recognition

- What makes face detection and recognition hard?

Variation in appearance: can't match a single face template and expect it to work.

Face recognition

- What makes face detection and recognition hard?

Lighting

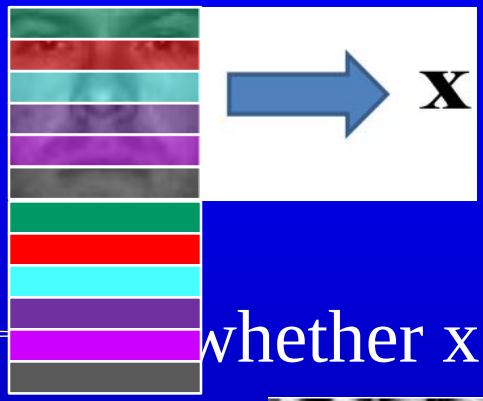
Occlusion

Viewpoint

Do this image contain faces? Where?

Face recognition

- Simple Idea for Face Detection
  - Treat each window in the image like a vector



x

- whether x matches some y_j in the database

SSD: $(y_j - x)^2$

Cross-correlation: $y_j \cdot x$

NCC, zero-mean NCC…

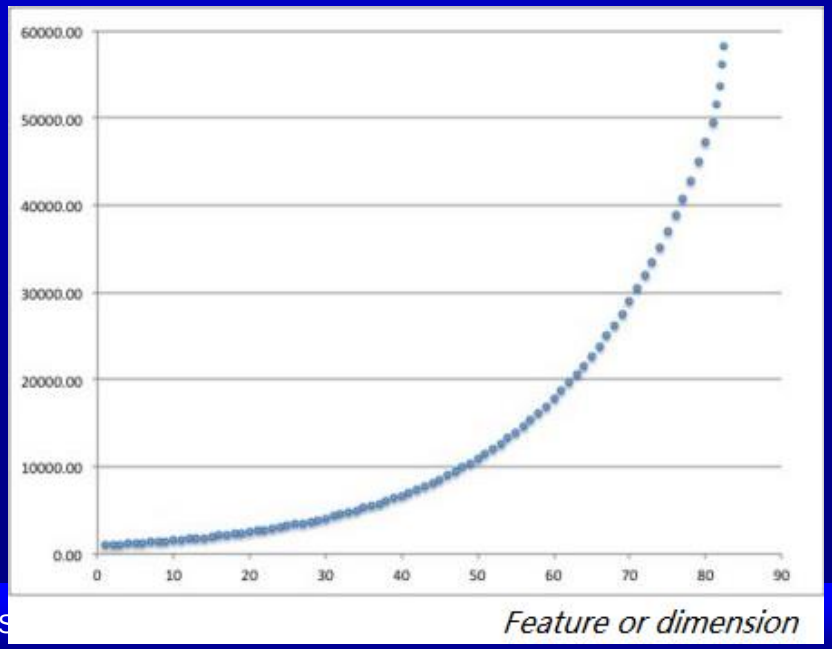
Face recognition

- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 100x100 image = 10,000 dimensions
  - Slow and lots of storage

- But very few 10,000-dimensional vectors are valid face images.
- We want to effectively model the subspace of face images.

Principal Component Analysis (PCA)

- Pattern recognition in high-dimensional spaces
  - Problems arise when performing recognition in a high-dimensional space (curse of dimensionality--维数灾难).
  - As the number of features or dimensions grows, the amount of data we need to generalize accurately grows Exponentially!



Principal Component Analysis (PCA)

- Pattern recognition in high-dimensional spaces
  - Problems arise when performing recognition in a high-dimensional space (curse of dimensionality--维数灾难).
  - Significant improvements can be achieved by first mapping the data into a lower-dimensional sub-space.

$$x = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_N \end{bmatrix} \text{--->} \textit{reduce dimensionality} \text{--->} y = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix} \quad (K << N)$$

  - The goal of PCA is to reduce the dimensionality of the data while retaining as much information as possible in the original dataset.

Principal Component Analysis (PCA)

- Dimensionality reduction
  - PCA allows us to compute a linear transformation that maps data from a high dimensional space to a lower dimensional sub-space.

$$y = Tx \text{ where } T = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1N} \\ t_{21} & t_{22} & \dots & t_{2N} \\ \dots & \dots & \dots & \dots \\ t_{K1} & t_{K2} & \dots & t_{KN} \end{bmatrix}$$

K x N $K \ll N$

$$\begin{aligned} b_1 &= t_{11}a_1 + t_{12}a_2 + \dots + t_{1n}a_N \\ b_2 &= t_{21}a_1 + t_{22}a_2 + \dots + t_{2n}a_N \\ &\dots \\ b_K &= t_{K1}a_1 + t_{K2}a_2 + \dots + t_{KN}a_N \end{aligned}$$

$y = (b_1, b_2, \cdots, b_K)^T$

$x = (a_1, a_2, \cdots, a_N)^T$

Principal Component Analysis (PCA)

- Lower dimensionality basis
  - Approximate vectors by finding a basis in an appropriate lower dimensional space.

(1) Higher-dimensional space representation:

$$x = a_1 v_1 + a_2 v_2 + \cdots + a_N v_N$$

$v_1, v_2, ..., v_N$ is a basis of the N-dimensional space

(2) Lower-dimensional space representation:

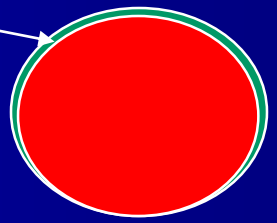
$$\hat{x} = b_1 u_1 + b_2 u_2 + \cdots + b_K u_K$$

$u_1, u_2, ..., u_K$ is a basis of the K-dimensional space

- *Note:* if both bases have the same size ($N = K$), then $x = \hat{x}$)

Principal Component Analysis (PCA)

- Information loss
  - Dimensionality reduction implies information loss!
  - PCA preserves as much information as possible, that is, it minimizes the error:



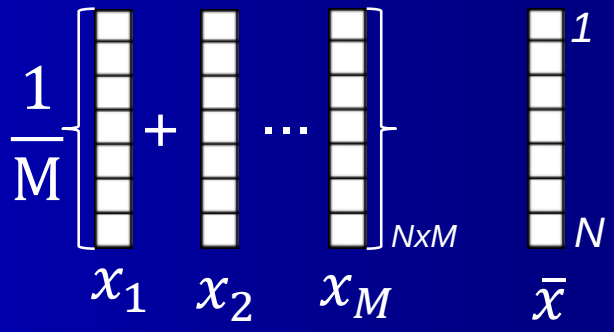
$$\|x - \hat{x}\|$$

- How should we determine the best lower dimensional sub-space?

The best low-dimensional space can be determined by the "best" eigenvectors of the covariance matrix of x (i.e., the eigenvectors corresponding to the "largest" eigen-values -- also called "principal components").

Principal Component Analysis (PCA)

- Methodology
  - Suppose $x_1, x_2, \cdots, x_M$ are *N* x 1 vectors



Step1: $\bar{x} = \frac{1}{M}\sum_{\{i=1\}}^{M} x_i$

Step2: subtract the mean: $\Phi_i = x_i - \bar{x}$ (i.e. center at zero)

Step3: from the matrix $A = [\phi_1\ \Phi_2 \cdots \Phi_M]$ $(N \times M\ matrix)$, then compute

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

(sample convariance matrix, $N \times N$, characterizes the scatter of the data)

Step4: compute the eigenvalue of C: $\lambda_1 > \lambda_2 > \cdots > \lambda_N$ $\quad AA^T u_i = \lambda_i u_i$

Step5: compute the eigenvectors of C: $u_1, u_2, \cdots, u_N$

Principal Component Analysis (PCA)

- Methodology – cont.

- Since C is symmetric, $u_1, u_2, \ldots, u_N$ form a basis, (i.e., any vector x or actually $(x - \bar{x})$, can be written as a linear combination of the eigenvectors):

$$x - \bar{x} = b_1 u_1 + b_2 u_2 + \cdots + b_N u_N = \sum_{i=1}^{N} b_i u_i \qquad b_i = u_i^T (x - \bar{x})$$

Step 6: (**dimensionality reduction step**) keep only the terms corresponding to the K largest eigenvalues:

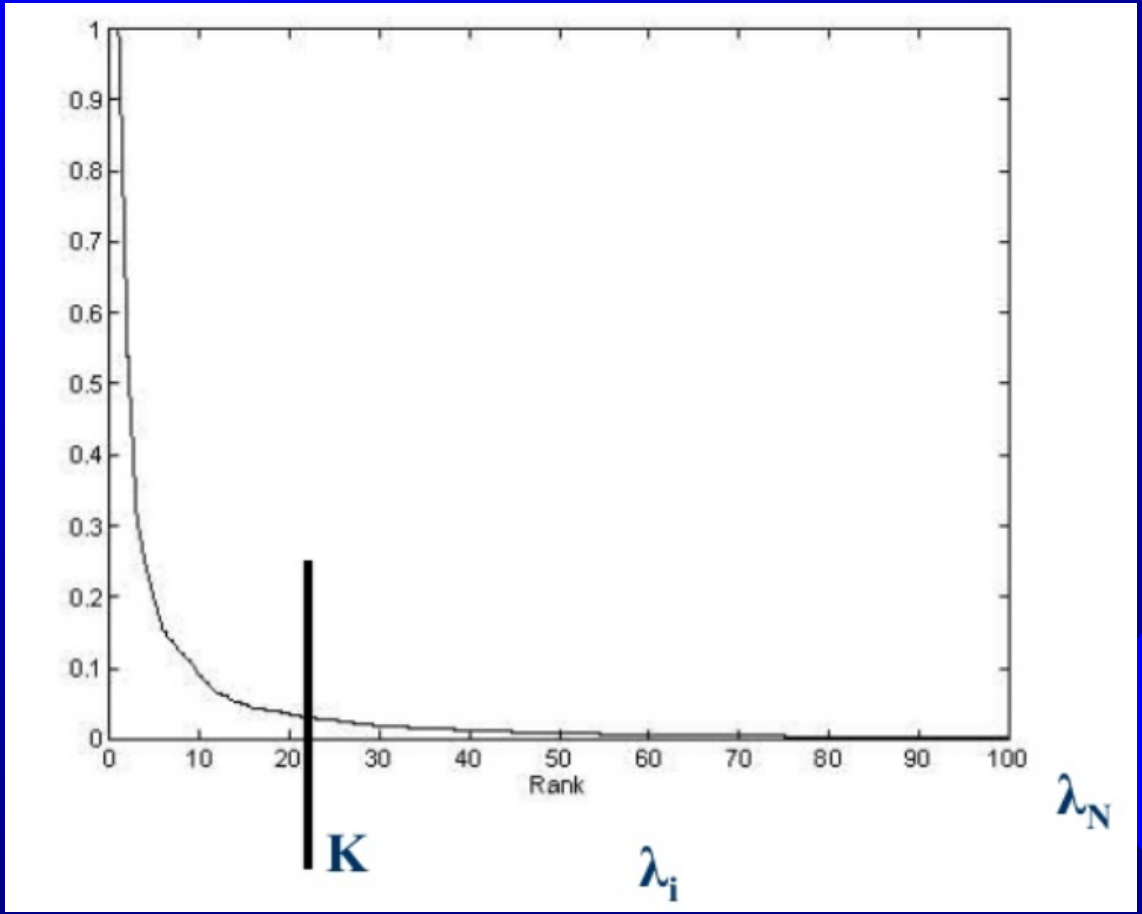
$$\hat{x} - \bar{x} = \sum_{i=1}^{K} b_i u_i \text{ where } K << N \qquad u_i - \text{dimension } N$$

- The representation of $\hat{x} - \bar{x}$ into the basis $u_1, u_2, \ldots, u_K$ is thus

$$\begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_K \end{bmatrix}$$

Principal Component Analysis (PCA)

- Eigenvalue spectrum.



Principal Component Analysis (PCA)

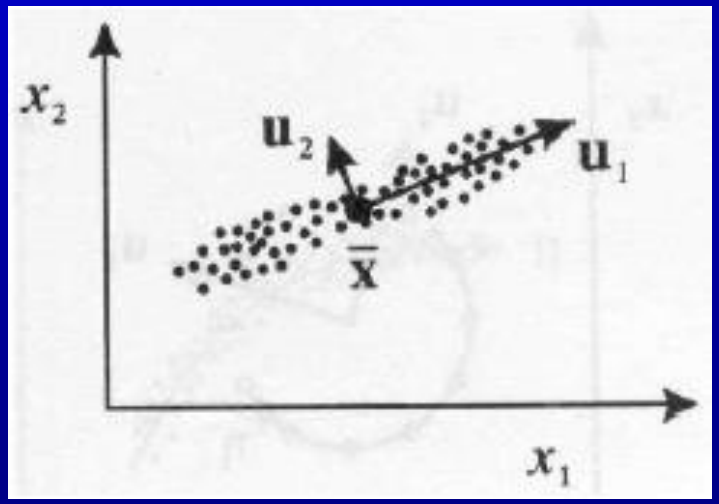
- Linear transformation implied by PCA
  - The linear transformation $R^N \rightarrow R^K$ that performs the dimensionality reduction is:

$$\begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_K \end{bmatrix} = \begin{bmatrix} u_1^T \\ u_2^T \\ \cdots \\ u_K^T \end{bmatrix} (x - \bar{x}) = U^T (x - \bar{x})$$

(i.e., simply computing coefficients of linear expansion)

Principal Component Analysis (PCA)

- Geometric interpretation
  - PCA projects the data along the directions where the data varies the most.
  - These directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues.
  - The magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions.



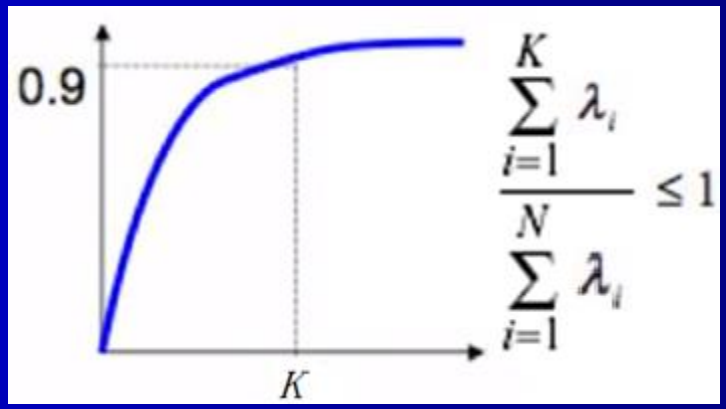
Principal Component Analysis (PCA)

- How to choose K (i.e., number of principal components) ?
  - To choose *K*, use the following criterion:

Principal Component Analysis (PCA)

- How to choose K (i.e., number of principal components) ?
  - To choose *K*, use the following criterion:

$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{N} \lambda_i} > Threshold \quad \text{(e.g., 0.9 or 0.95)}$$



  - In this case, we say that we “preserve” 90% or 95% of the information in our data.
  - If K=N, then we “preserve” 100% of the information in our data.

Principal Component Analysis (PCA)

- What is the error due to dimensionality reduction?
  - The original vector *x* can be reconstructed using its principal components:

$$\hat{x} - \bar{x} = \sum_{i=1}^{K} b_i u_i \text{ or } \hat{x} = \sum_{i=1}^{K} b_i u_i + \bar{x}$$

  - PCA minimizes the reconstruction error:

$$e = \|x - \hat{x}\|$$

  - It can be shown that the error is equal to:

$$e = 1/2 \sum_{i=K+1}^{N} \lambda_i$$

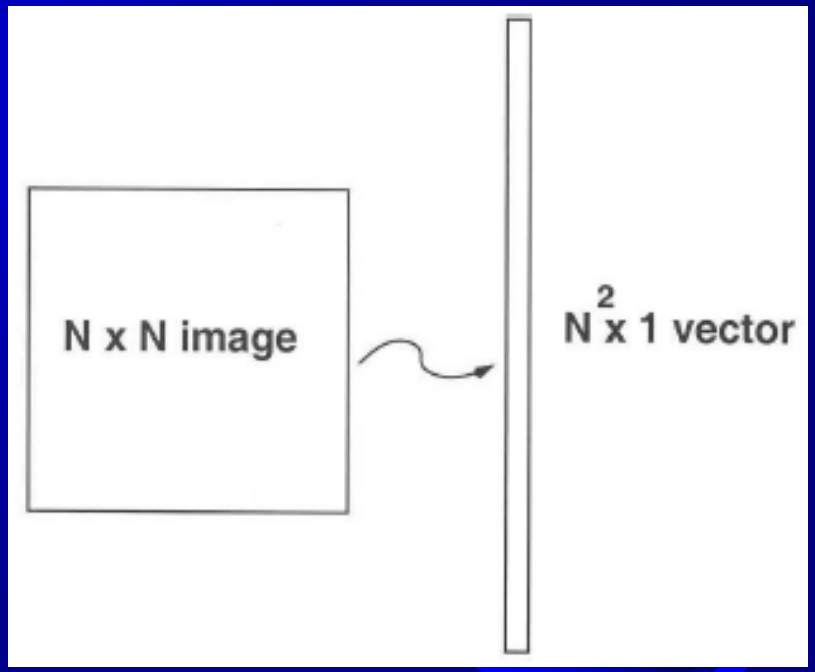
Principal Component Analysis (PCA)

- Standardization
  - The principal components are dependent on the *units* used to measure the original variables as well as on the *range* of values they assume.
  - You should always standardize the data prior to using PCA.
  - A common standardization method is to transform all the data to have zero mean and unit standard deviation:

$$\frac{x_i-\mu}{\sigma}$$ (μ and σ are the mean and standard deviation of $x_i's$)

Application to Faces

- Case Study: Eigenfaces for Face Detection/Recognition
  - M. Turk, A. Pentland, "Eigenfaces for Recognition", *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.
- Face Recognition
  - The simplest approach is to think of it as a template matching problem
  - Problems arise when performing recognition in a high-dimensional space.
  - Significant improvements can be achieved by first mapping the data into a *lower dimensionality* space.

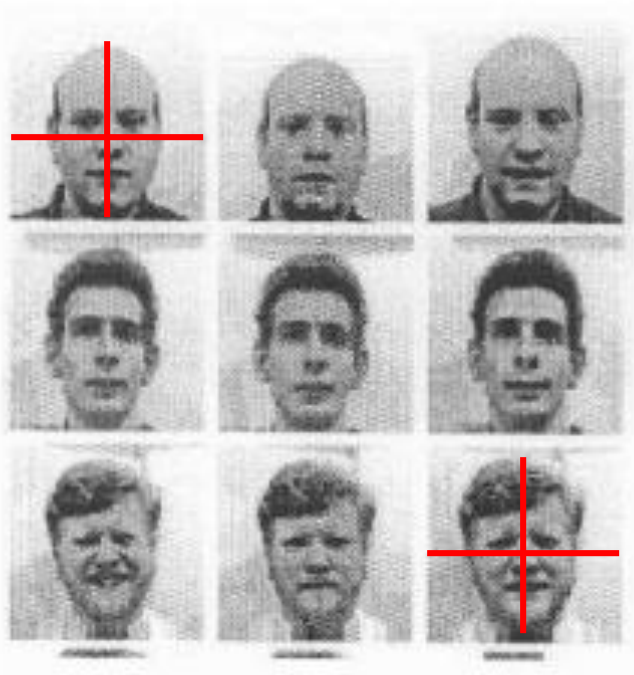


Application to Faces

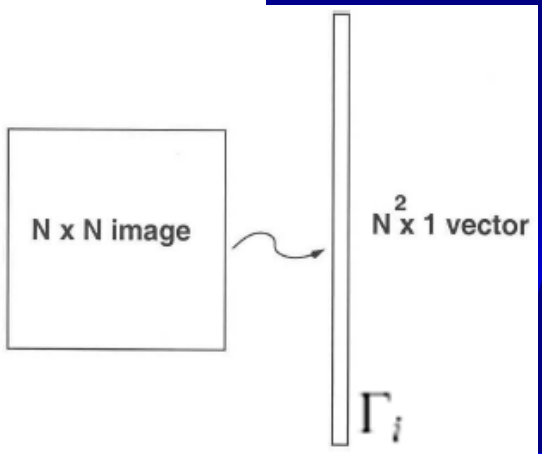
- Computation of low-dimensional basis (i.e.,eigenfaces):

Step 1: obtain face images $I_1, I_2, ..., I_M$ (training faces)

(**very important:** the face images must be *centered* and of the same *size*)

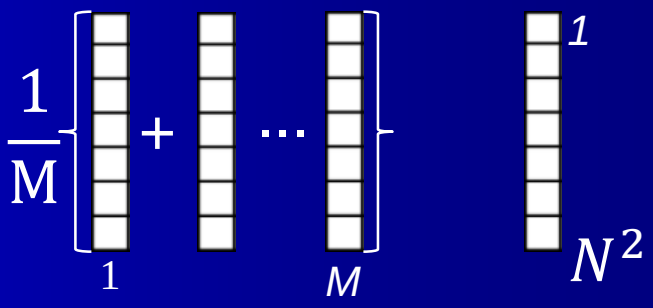


Step 2: represent every image I_i as a vector Γ_i



Application to Faces

- Computation of the eigenfaces – cont.



Step 3: compute the average face vector Ψ:

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

Step 4: subtract the mean face:

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

Step 5: compute the covariance matrix C:

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

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

Application to Faces

- Computation of the Eigen faces – cont.

Step 6: compute the eigenvectors u_i of AA^T → $AA^T u_i = \lambda_i u_i$ $Cu_i = \lambda u_i$

The matrix AA^T is very large --> not practical !!

Step 6.1: consider the matrix $A^T A$ (M x M matrix)

Step 6.2: compute the eigenvectors v_i of $A^T A$

$$A^T A v_i = \mu_i v_i$$

$A^T A$ AA^T

$\mu_i = \lambda_i$

What is the relationship between u_i and v_i?

$$A^T A v_i = \mu_i v_i \Rightarrow AA^T A v_i = \mu_i A v_i \Rightarrow C(Av_i) = \mu_i (Av_i)$$

Eigenvector of C

$$\rightarrow u_i = Av_i \quad and \quad \lambda_i = \mu_i$$

Thus, AA^T and $A^T A$ have the same eigenvalues and their eigenvectors are related as follows: $u_i = Av_i$!!

AA^T

λ_i eigenvalue

u_i eigenvector

$A^T A$

μ_i eigenvalue

v_i eigenvector

AA^T

6.4276	0	0	0	0
0	0.9822	0	0	0
0	0	0.3178	0	0
0	0	0	0.0000	0
0	0	0	0	0.0000

6.4276	0	0
0	0.9822	0
0	0	0.3178

$A^T A$

Application to Faces

- Computation of the eigenfaces – cont.

Note 1: AA^T can have up to N^2 eigenvalues and eigenvectors.

Note 2: $A^T A$ can have up to M eigenvalues and eigenvectors.

Note 3: The M eigenvalues of $A^T A$ (along with their corresponding eigenvectors) correspond to the M *largest* eigenvalues of AA^T (along with their corresponding eigenvectors).

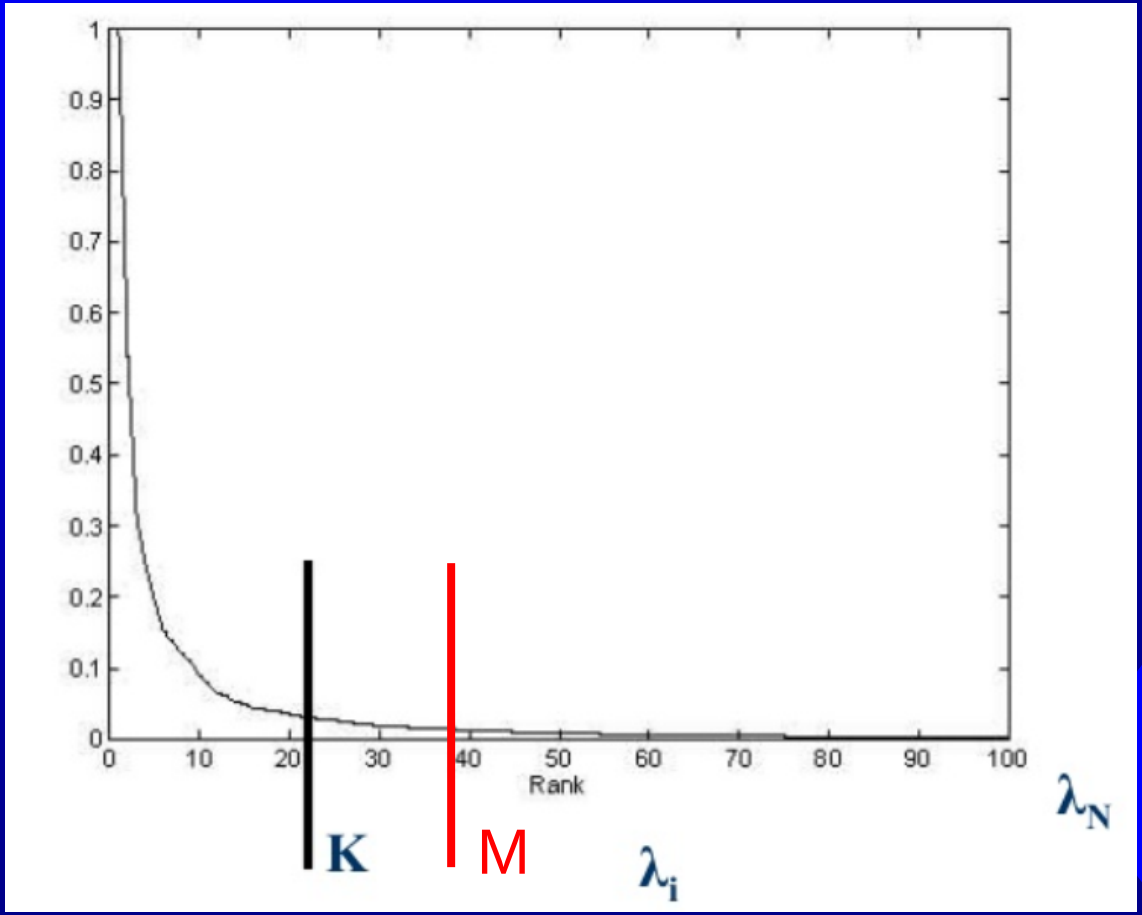
Step 6.3: compute the M best eigenvectors of AA^T: $u_i = Av_i$

(**important:** normalize u_i such that $\|u_i\| = 1$)

Step 7: keep only K eigenvectors (corresponding to the K largest eigenvalues)

Application to Faces

- Computation of the eigenfaces – cont.



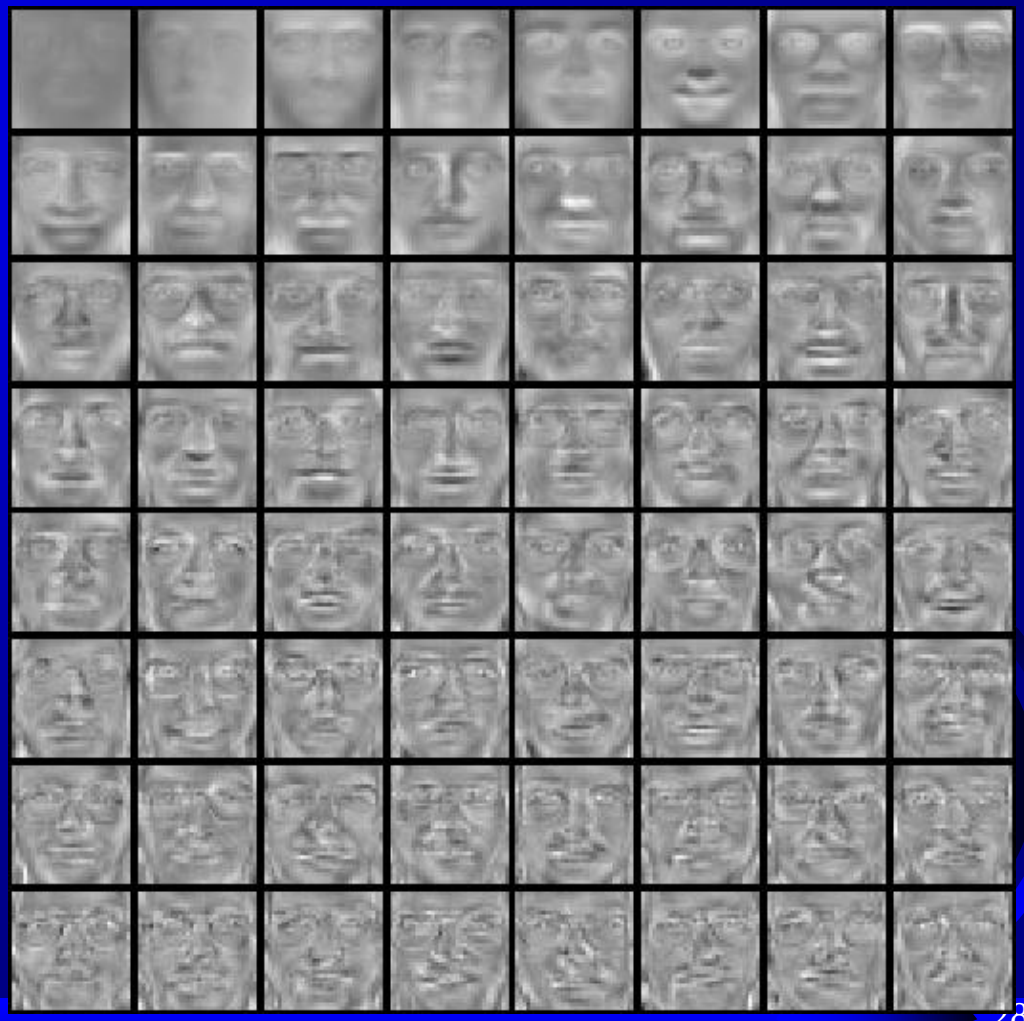
Eigenfaces example

Training images

Eigenfaces example

Mean: μ

Top eigenvectors: $u_1,\ldots u_k$



Application to Faces

- 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:

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

(we call the u_j's *eigenfaces*)

Face reconstruction:

Eigenfaces

- Face Recognition Using Eigenfaces

- Given an unknown face image Γ (centered and of the same size like the training faces) follow these steps:

Step 1: normalize Γ: $\Phi = \Gamma - \Psi$

Step 2: project on the eigenspace

$$\hat{\Phi} = \sum_{i=1}^{K} w_i u_i \quad (w_i = u_i^T \Phi) \qquad (where \| u_i \| = 1)$$

Step 3: represent Φ as: $\Omega = \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_K \end{bmatrix}$

Step 4: find $e_r = \min_k \| \Omega - \Omega^k \|$, where Ω^k is the lth face vector.

Step 5: if $e_r < T_r$, then Γ is recognized as face l from the training set.

Eigenfaces

- Face Recognition Using Eigenfaces – cont.
  - The distance e_r is called *distance within face space (difs)*
  - The *Euclidean distance* can be used to compute e_r, however, the *Mahalanobis distance (马氏距离)* has shown *to* work better:

$$\| \Omega - \Omega^k \| = \sum_{i=1}^{K} (w_i - w_i^k)^2$$

Euclidean distance

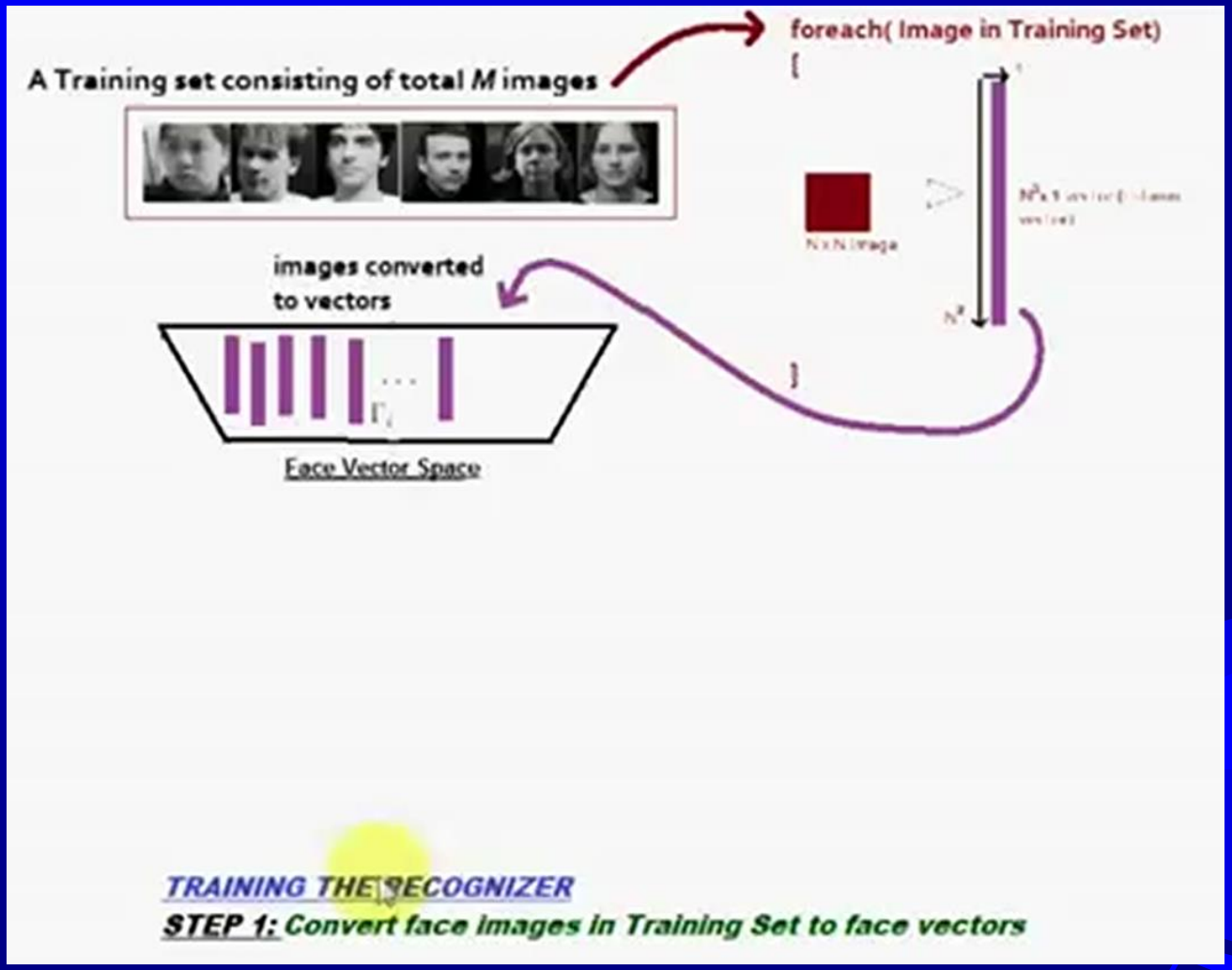
Mahalanobis distance

$$\| \Omega - \Omega^k \| = \sum_{i=1}^{K} \frac{1}{\lambda_i} (w_i - w_i^k)^2$$

(variations along all axes are treated as equally signifi cant)

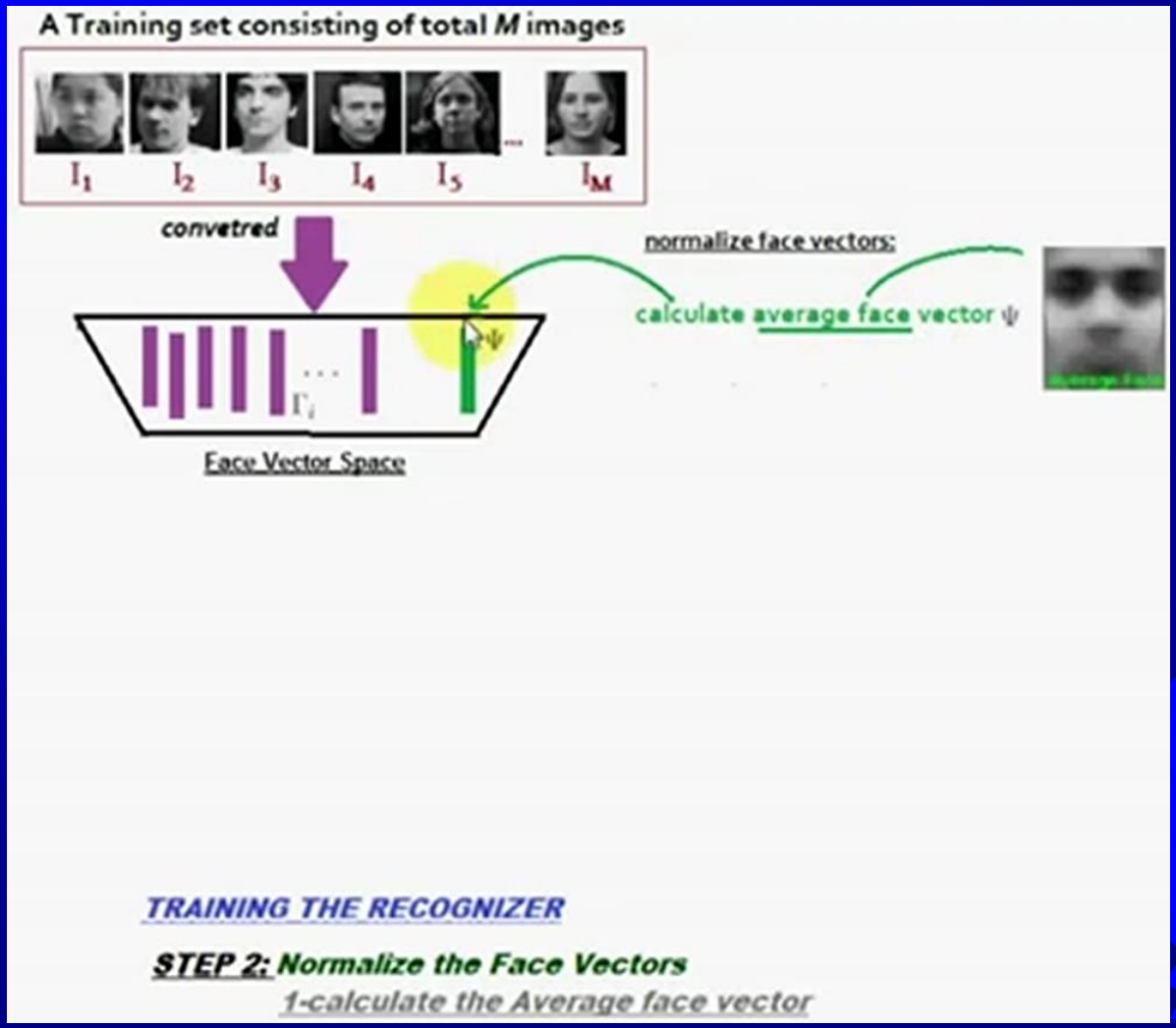
Eigenfaces

- Face Recognition Using Eigenfaces



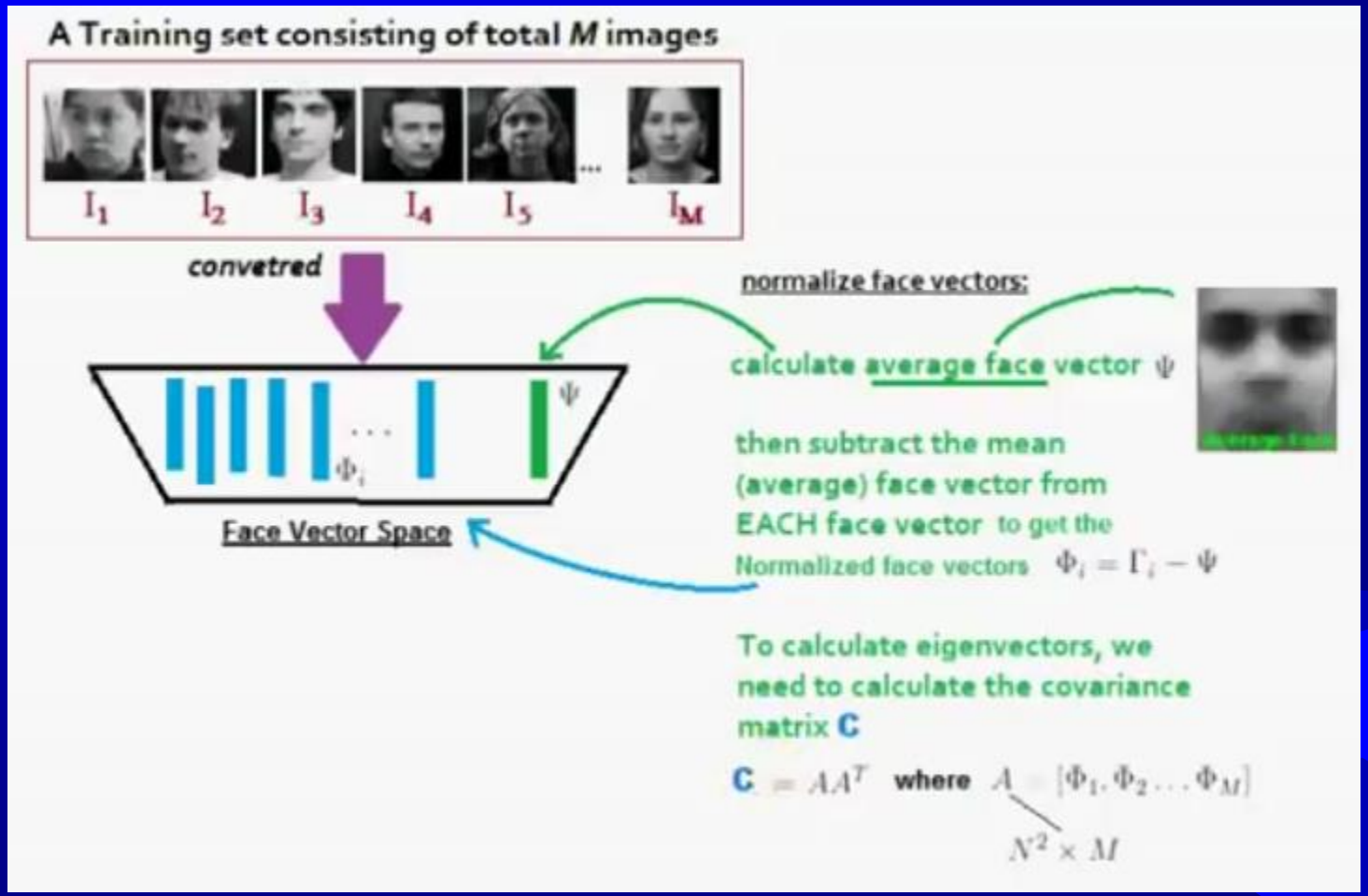
Eigenfaces

- Face Recognition Using Eigenfaces



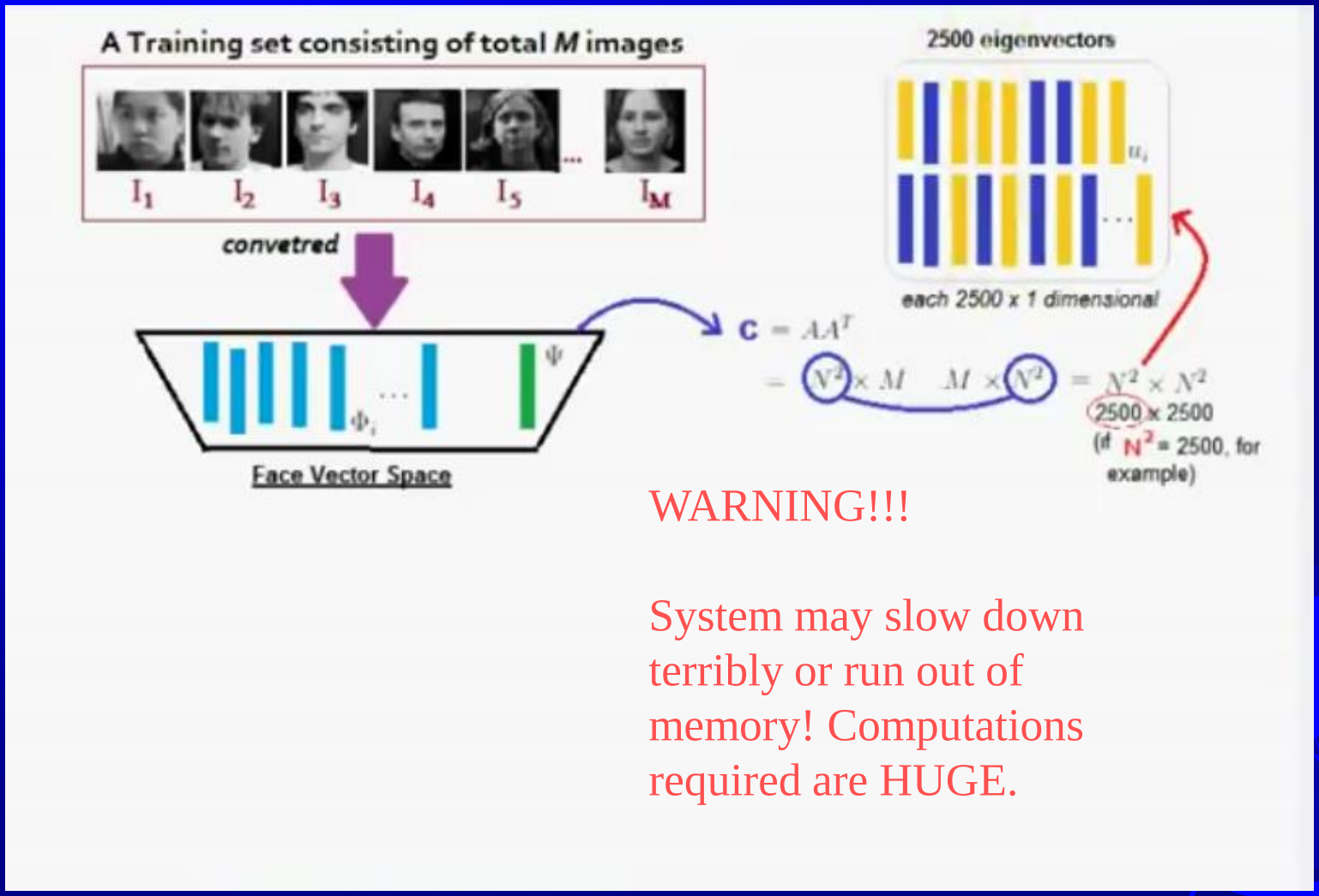
Eigenfaces

- Face Recognition Using Eigenfaces



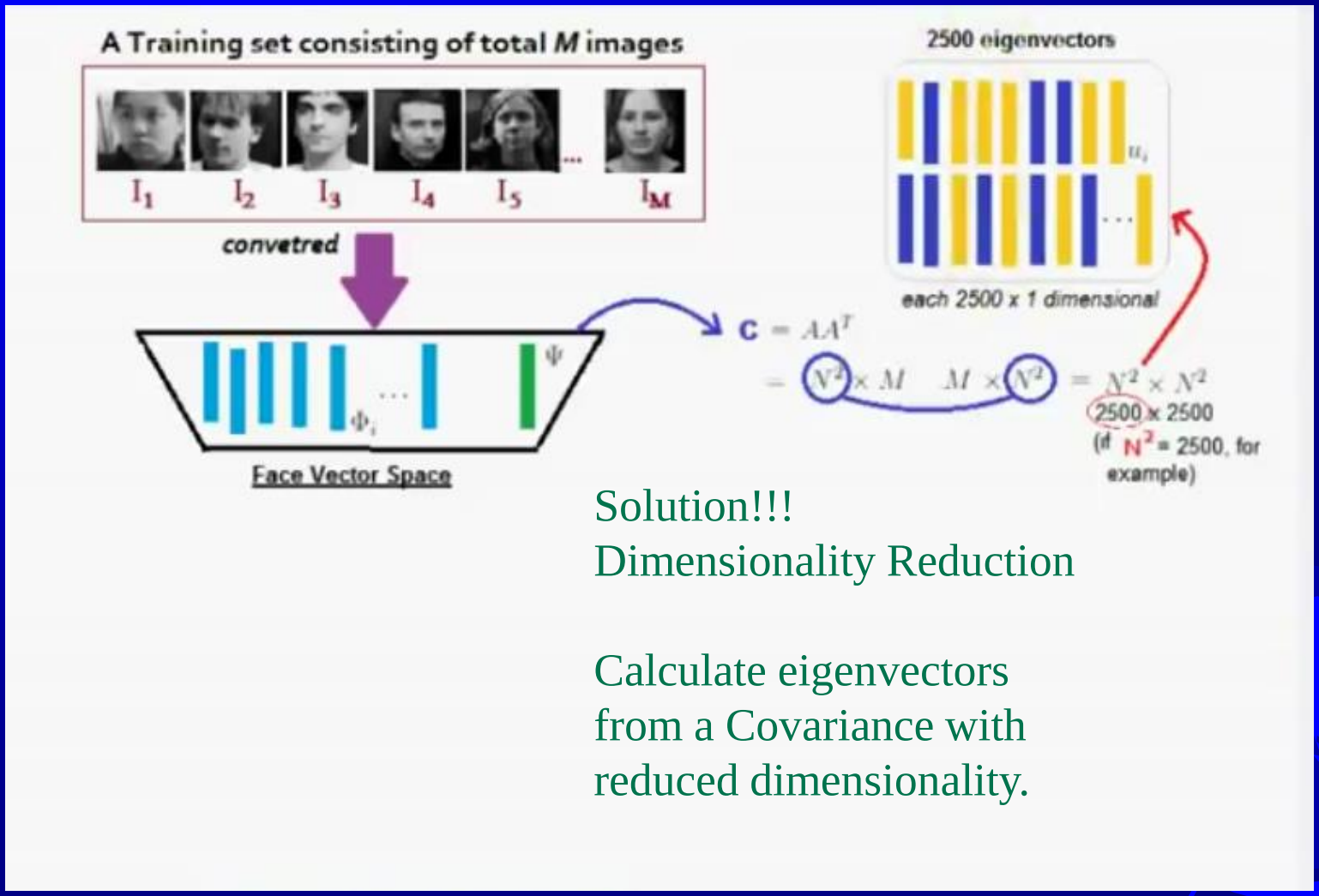
Eigenfaces

- Face Recognition Using Eigenfaces



Eigenfaces

- Face Recognition Using Eigenfaces

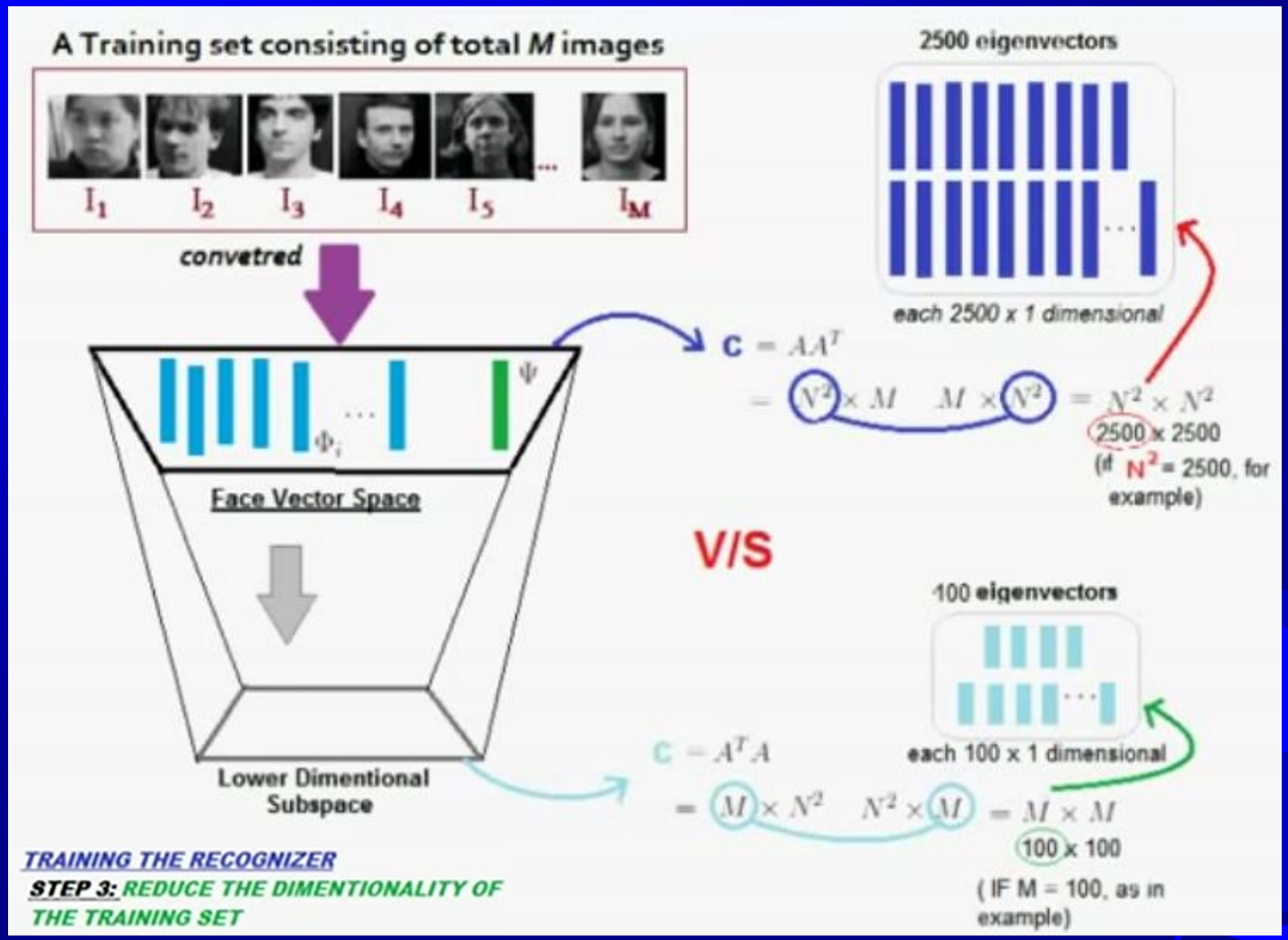


Solution!!!
Dimensionality Reduction

Calculate eigenvectors from a Covariance with reduced dimensionality.

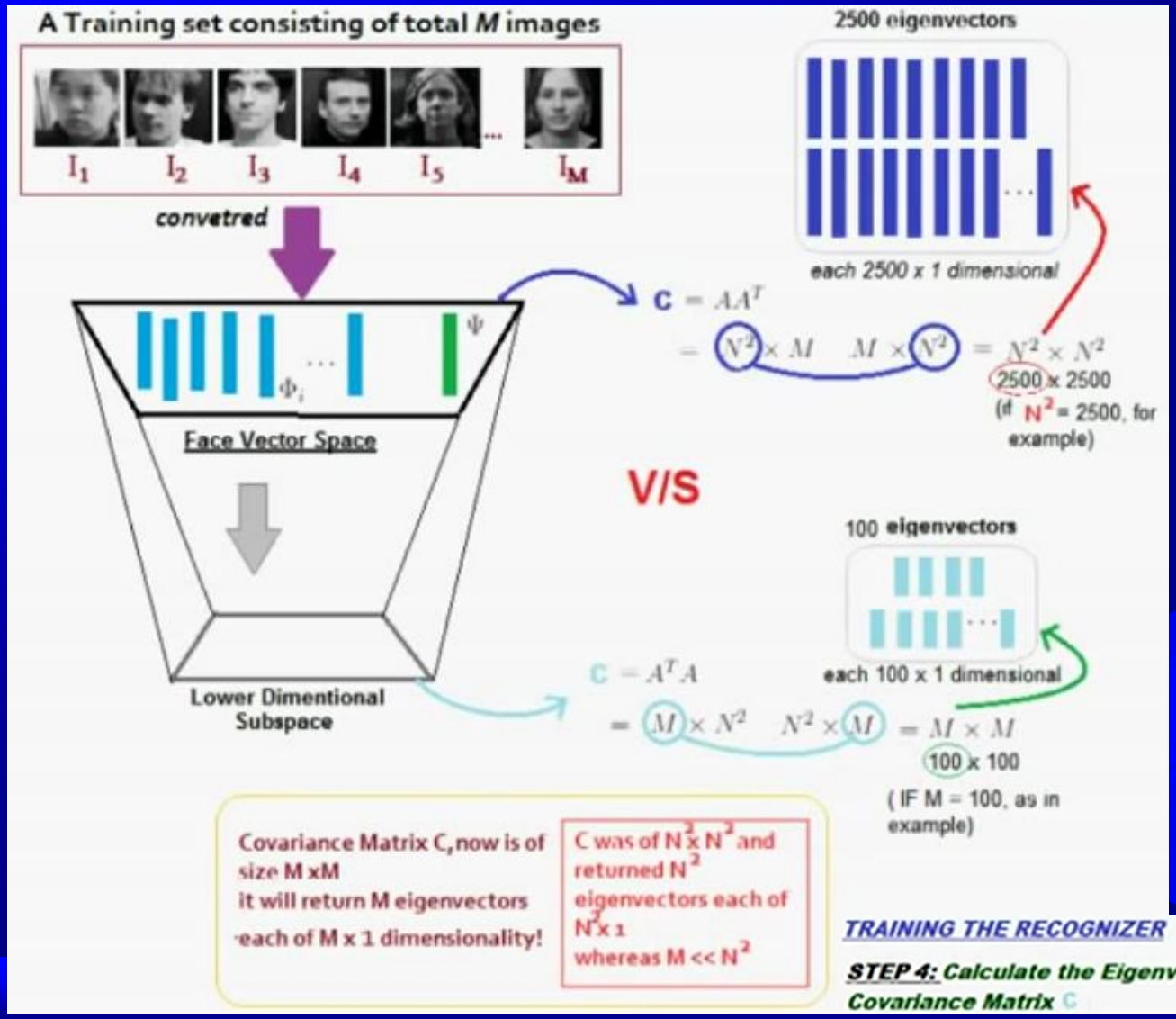
Eigenfaces

- Face Recognition Using Eigenfaces



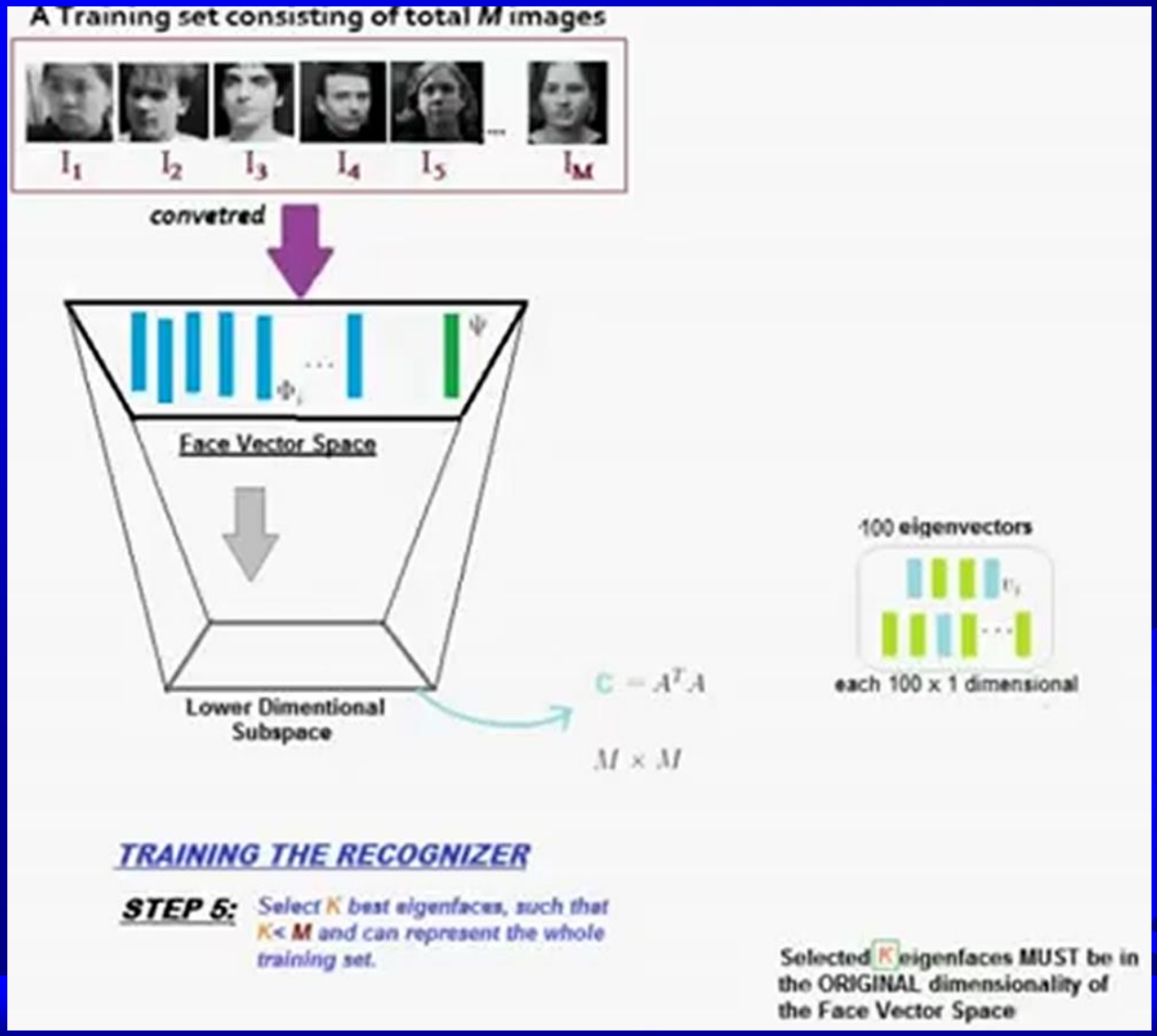
Eigenfaces

- Face Recognition Using Eigenfaces



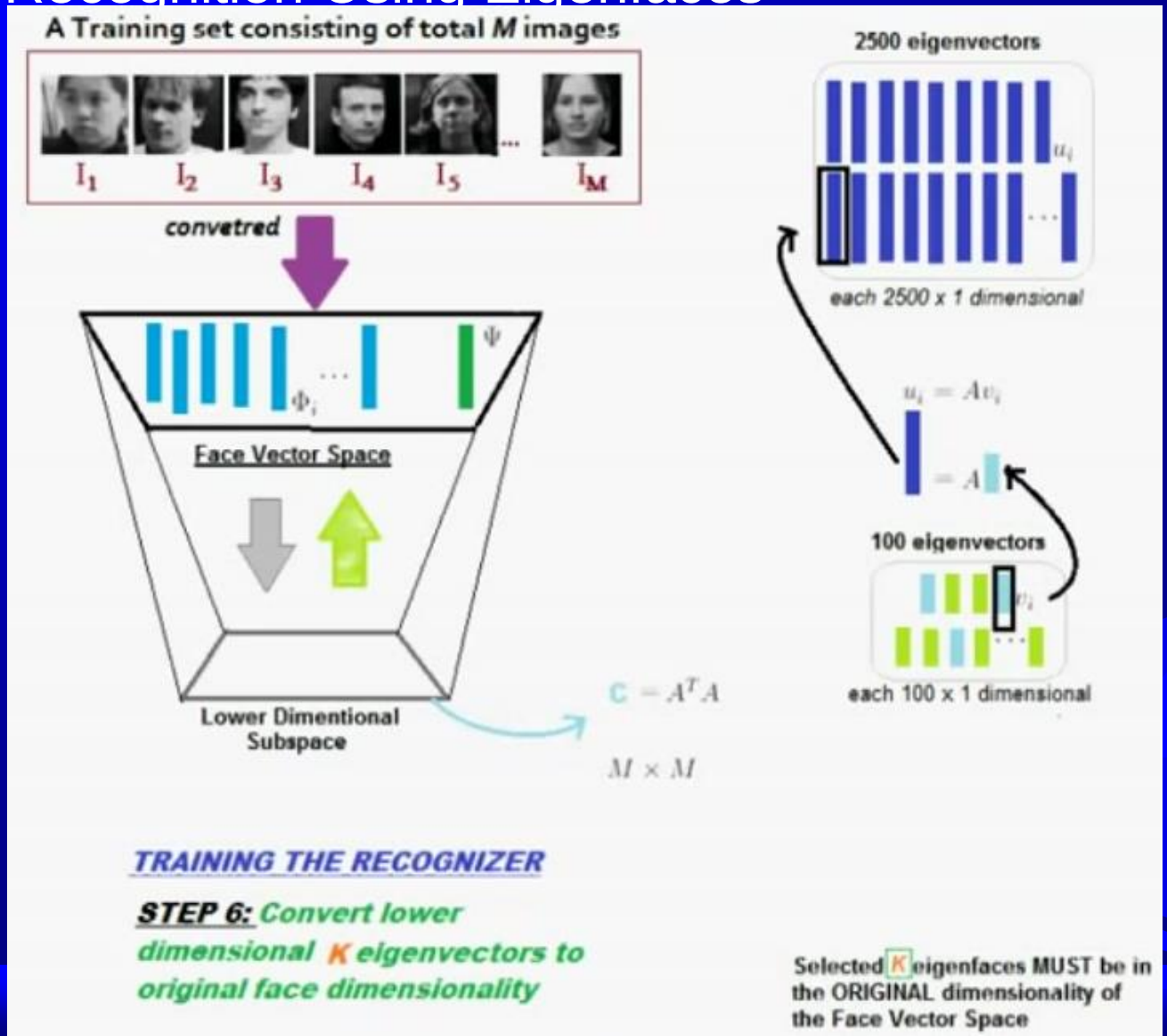
Eigenfaces

- Face Recognition Using Eigenfaces



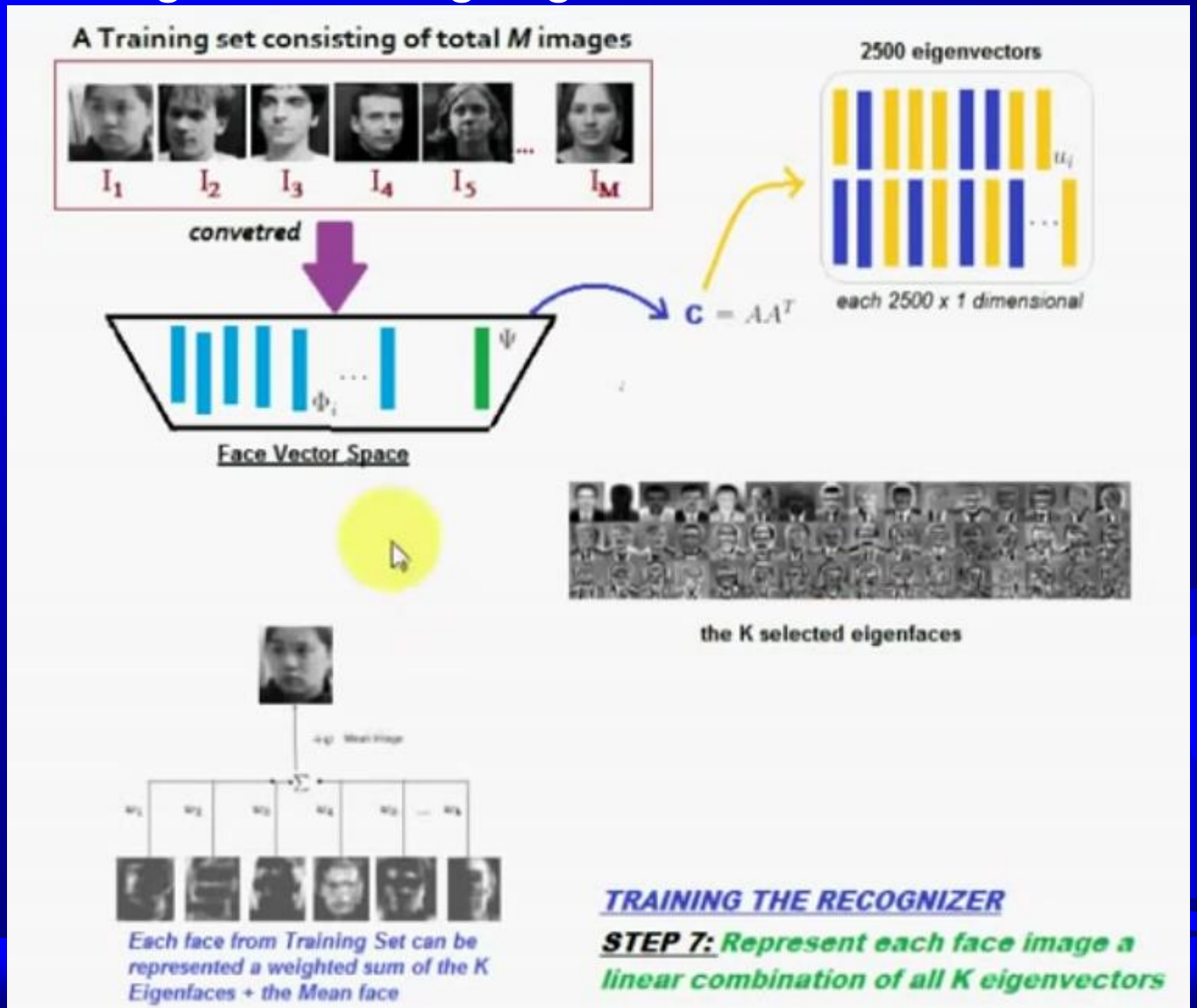
Eigenfaces

- Face Recognition Using Eigenfaces



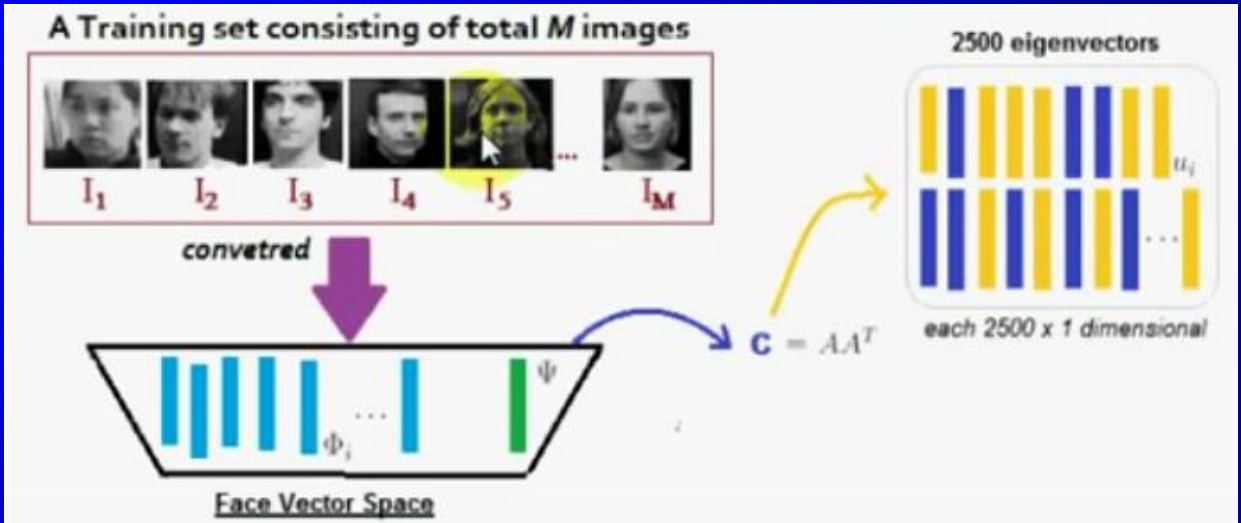
Eigenfaces

- Face Recognition Using Eigenfaces



Eigenfaces

- Face Recognition Using Eigenfaces



$$w_i = u_i^T \Phi$$

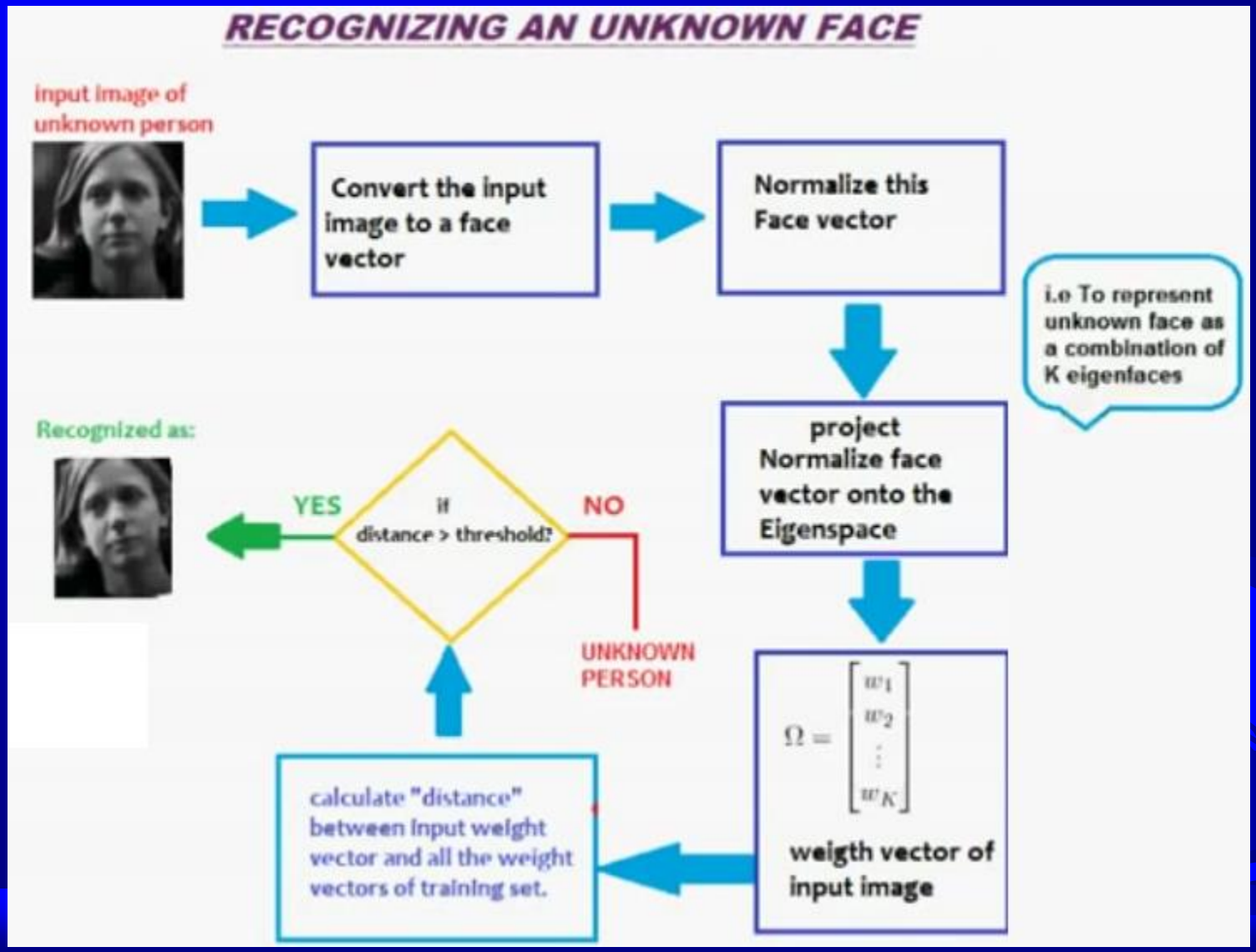
$$\Omega_i = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}$$

Each face from Training Set can be represented a weighted sum of the K Eigenfaces + the Mean face

A weight vector Ω_i which is the eigenface representation of the ith face. weight vector for each face is calculated.

Eigenfaces

- Face Recognition Using Eigenfaces



Face detection and recognition

Detection

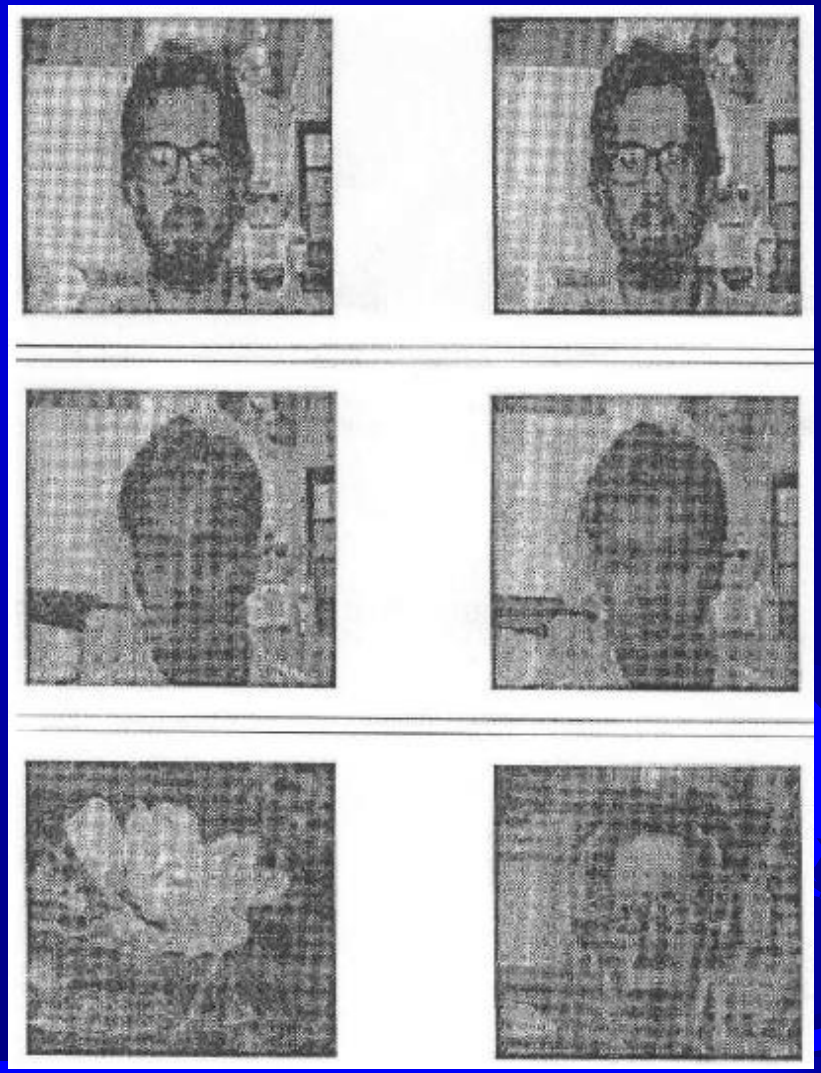
Recognition

"Sally"

Eigenfaces

- Reconstruction of faces and non-faces

Input Reconstructed



Reconstructed face looks like a face.

Reconstructed non-face looks like a fac again!

Reconstruction using partial information

- Robust to partial face occlusion.

Input Reconstructed

Limitations

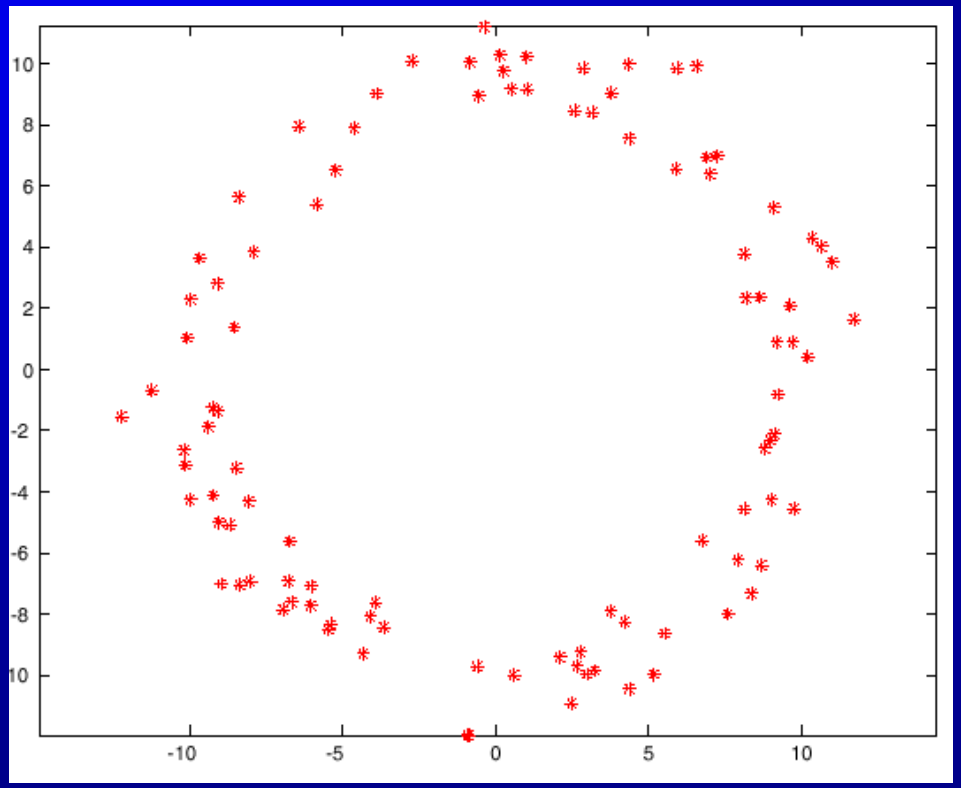
- Background changes cause problems
  - De-emphasize the outside of the face (e.g., by multiplying the input image by a 2D Gaussian window centered on the face).
- Light changes degrade performance
  - Light normalization helps.
- Performance decreases quickly with changes to face size
  - Multi-scale 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

- Not robust to misalignment

Limitations

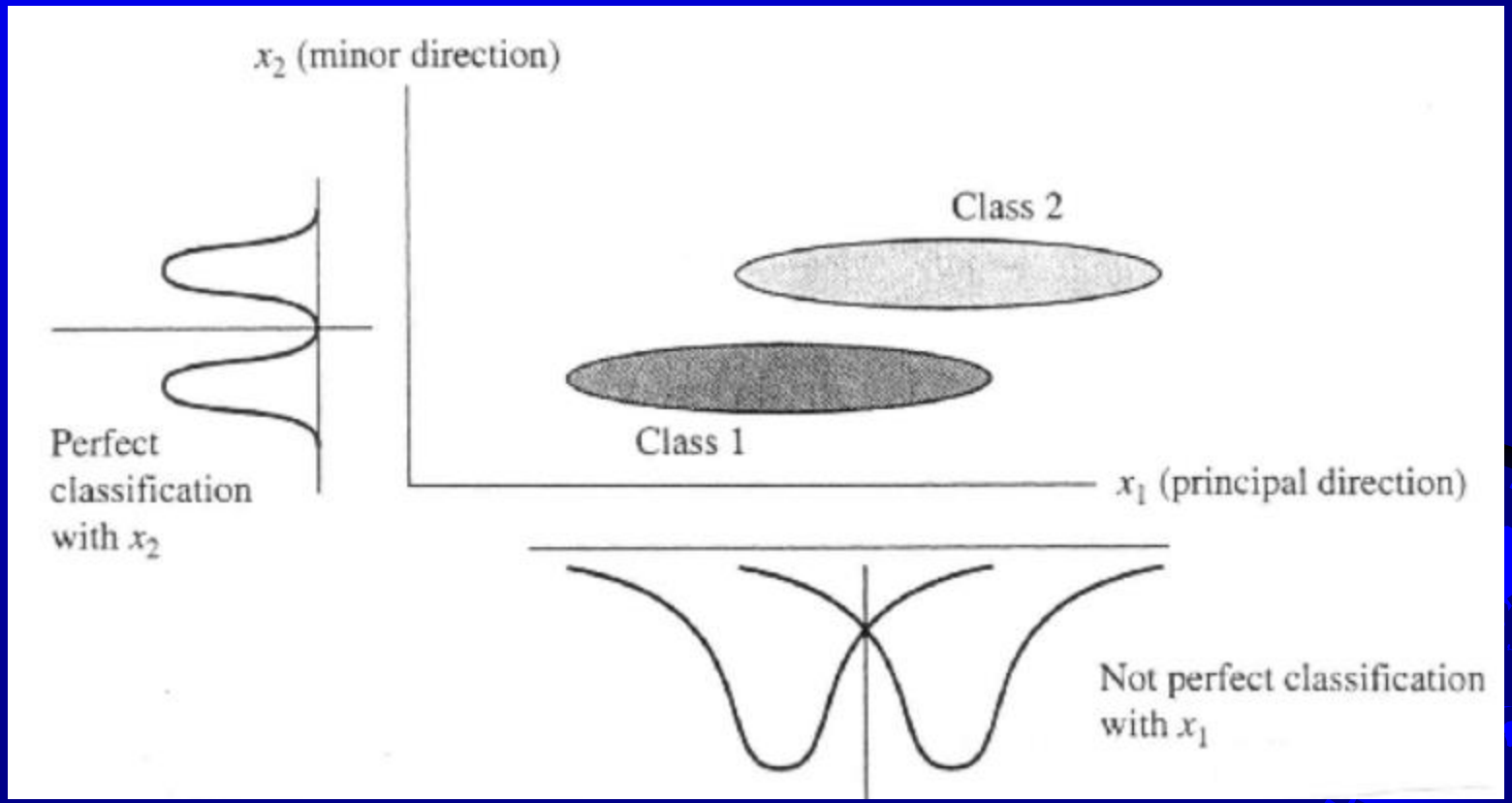
- PCA assumes that the data follows a Gaussian distribution (mean μ, covariance matrix Σ)



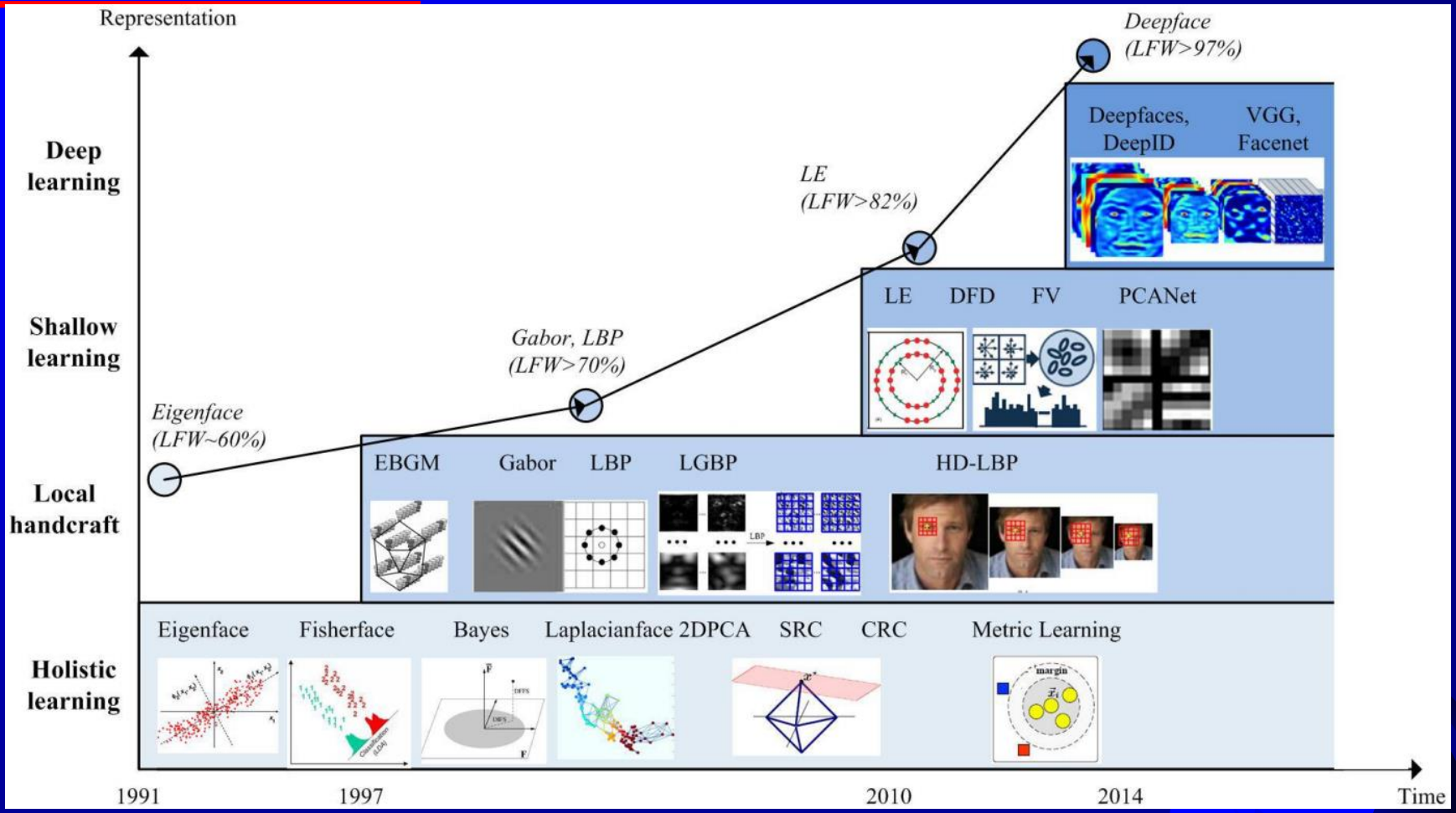
The shape of this dataset is not well described by its principal components

Limitations

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



Face Recognition-State-of-the-art



Milestones of feature representation for FR. The holistic approaches dominated the FR community in the 1990s and 2000s. In the early 2000s and 2010s, Local-feature-based FR and learning-based local descriptors were introduced successively. In 2014, DeepFace and DeepID achieved state-of-the-art accuracy, and research focus has shifted to deep-learning-based approaches. As the representation pipeline becomes deeper and deeper, the LFW (Labeled Face in-the-Wild) performance steadily improves from around 60% to above 90%, while deep learning boosts the performance to 99.80% in only three years.

See You