Principal Component Analysis

Jenn-Jier James Lien (連 震 杰)
Professor

Computer Science and Information Engineering
National Cheng Kung University

(O) (06) 2757575 ext. 62540
jjlien@csie.ncku.edu.tw
http://robotics.csie.ncku.edu.tw

0.1 Major Issues

1. **VQ Vs. PCA Vs. GAN ➔ PCA Vs. SVM**
   1) VQ: (1) Unsupervised learning, (2) Space reduction by clustering encoding
   2) PCA: (1) Unsupervised learning, (2) Dimensionality reduction
2. **Fundamental of machine learning:**
   - PCA, PPCA to Factor analysis (including noise) by Markov Model.
3. **PCA:**
   - Dimensionality reduce => Domain knowledge
   - Reconstruction
   - Low frequency components (PCA, Gaussian) +
     high frequency components (ICA, non-Gaussian) + noise
4. **Covariance matrix Vs. correlation matrix**
5. **Covariance matrix => PCA ➔ SVD: $C=UWV^T$**
   - Gaussian Model
   - Affine transform:
     - » translation (mean shift),
     - » rotation (eigenvectors) and
     - » scaling (eigenvalue)
5. **Camera 2D image plane Vs. Object surface/plane rotation (2D)**
   - **Optical axis Vs. Normal direction of 2D plane/surface at object**

0.2 Preprocssing

- **Geometric Normalization: Affine (Vs. Perspective)**
  - Translation
  - Rotation
  - Scaling
- **Illumination Normalization (Gray Value (0~255) Histogram) /Histogram Equalization**
  - Translation
  - Rotation
  - Scaling

$$(a-m_a)/\sigma_a = (b-m_b)/\sigma_b$$

- **Similarity Measure (Gaussian Model ?): Absolute Relation => Relative Relation**
  - Translation (shift): Zeroing
  - Rotation: ? (eigenvector ? Scale-Invariant Feature Transform (SIFT), Histogram Of Gradient (HOG) ?), Rotation invariance
  - Scaling: Normalization to the same unit (NT$ vs US$), Scaling invariance

0.3 Domain Knowledge

- **Input Data => PCA (Reduce the dimension => Input) =>**
  - => Similarity Measure (=> Machine Learning)
  - => Detection or Recognition Result

- **Domain Knowledge**

- **Find the best space => Manifold**
  - Lower-dimensional space
    - PCA, LDA
  - Higher-dimensional space
    - SVM, Kernel function

- **Template Matching: Correlation Coefficient**
  - One image per person for few persons
- **PCA => PCA (Gaussian) + ICA (Non-Gaussian)**
  - One image per person for many persons
  - Consider relation between persons
- **LDA**
  - One image set per person for many persons
  - Consider relation within the same person and between different persons
- **LLE: Locally Linear Embedding**
- **SVM: Linear and Non-Linear Discriminant**
  - Subspace => Kernel Function to High Dimension

- **PCA**
  - Gaussian Model
  - Orthogonal Vs. Orthonormal
- **ICA**
  - Non-Gaussian Model
  - Independent

•

- **Orthogonal is independent**
  - x+y: Linear combination or orthogonal … can be separated
  - x*y: correlation cannot be separated, => log x + log y
- **Independent does not need to be orthogonal (ICA)**
  - Why independent? => P(A,B) = P(A) P(B)

- **Orthonormal = orthogonal + unit vector (normalization) = eigenvectors**

- **PCA: Physical meaning, why eigenvector, eigenvalue**
- **Gaussian Model: Mean, variance**

- **SVD => Factorization**

0.4 Dimensionality Reduction

Linear Vs Non-Linear
Supervised Vs. Unsupervised

- **PCA => PCA (Gaussian) + ICA (Non-Gaussian)**
  - One image per person for many persons
  - Consider relation between persons
- **LDA**
  - One set image per person for many persons
  - Consider relation within the same person and between different persons

- **LLE: Locally Linear Embedding**
- **ISOMAP**
- **Laplacision Eigenmap**

- **SVM: Linear and Non-Linear Discriminant**
  - Subspace => Kernel Function to High Dimension

1.0 Principal Component Analysis

- **Principal Component Analysis = Hotelling Transform = Karhunen-Loeve (K-L) Transform (or Expansion)**
- **Principle components = Eigenvectors**
- **Properties: (Covariance matrix)**
  - Compression (not regular compression, it reduces the dimensions)
  - Correlation (self and cross correlations – Significant variance)

- **Compression:**
  - DCT, Wavelets… (ex. MPEG, H.264)
    - Spatial domain
    - Temporal domain
  - PCA
- **Decomposition/Factorization (SVD) Vs. 3D Reconstruction**

1.1 Why PCA

1. Dimensional Reduction Vs. Descriptor

1) 1M 100x100-pixel facial images => 100x100 (2D) Matrix

=> 100x100-dimensional (1D) Vector $(\ldots)_{100x100}$

- One sample/image at high-dimensional space/coordinate

2) Similarity Measure: Detection or Recognition

- Compare: 100x100-dim vector Vs. 100x100-dim vector

➔ PCA ➔ 50-dim vector Vs. 50-dim vector

3) - Computational time

- Storage

2. Reconstruction as GAN

Principal component analysis, or PCA: Is a statistical procedure that allows you to summarize the information content in large data tables by means of a smaller set of "summary indices" that can be more easily visualized and analyzed.

1.1 PCA

1. Covariance Matrix A Vs. Correlation Matrix A'

2. $A=(A'-m)=UWV^T$

➢**Gaussian distribution** $N(m, Lamda^2)$,

➢**as affine transform (translation by m, rotation by U and V, scaling by W)**

3. Affine transform –

m: **as the translation terms**

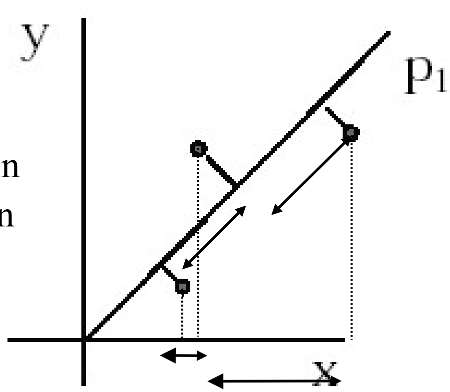
U: **Eigenvector (orthonormal vector) matrix as the rotation matrix.**

W: **Eigenvalues (Lamda^2) as the variance/scaling terms at denomination for normalization.**

V: **Rotation matrix**

- We are given:
  - a set of M objects” (images)
  - ex.
  - each is represented, initially, by a set of N^2 features (128^2) or pixels
  - This data is organized as an (very large!) N^2xM matrix
- Let’s look at a small example

Separate in y direction



Separate in x direction

projection

- points are 2-D points
- we find the axis that most closely passes through these points
- if the axis passed exactly through these points, then we would need only one coordinate to represent each point.
  - ➢ That is, sum of all shortest vertical distance of samples or points to the major axis

Eigenvectors, orthonormal vector

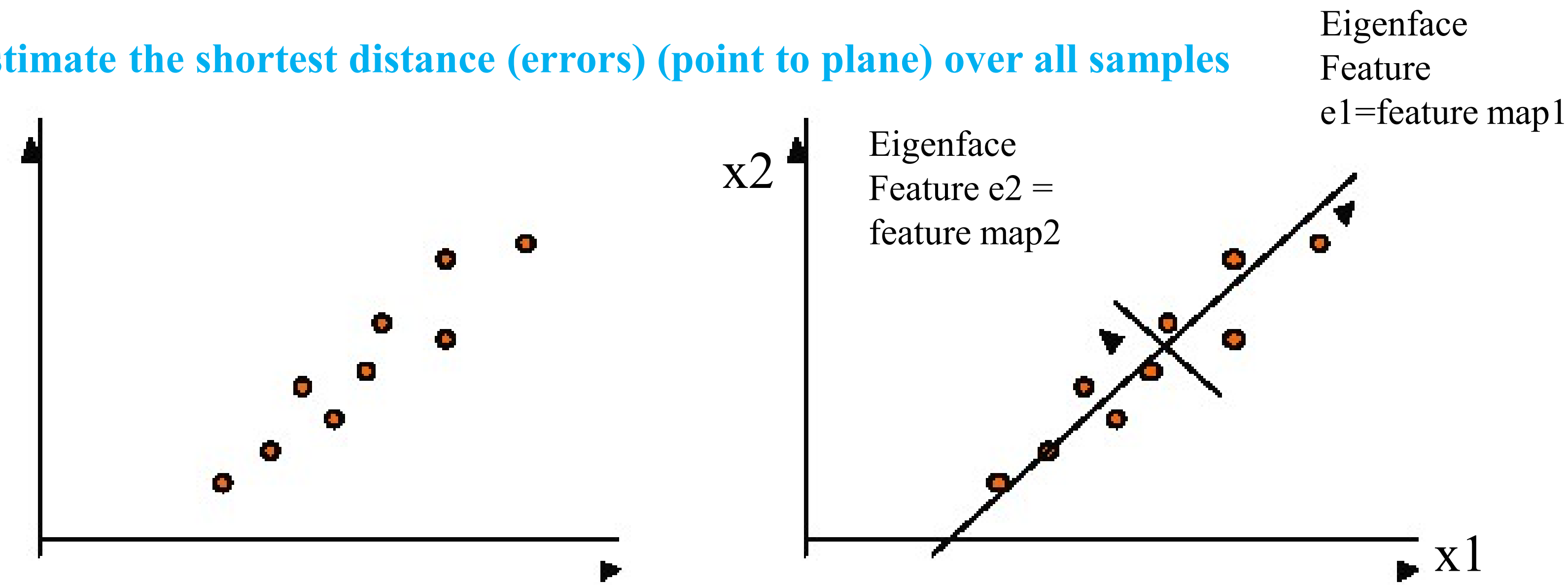
- Find an orthogonal set of basis vectors for the feature space, subject to the requirement that the new features have zero correlation with each other.

eigenvectors

- Dimensional reduction.

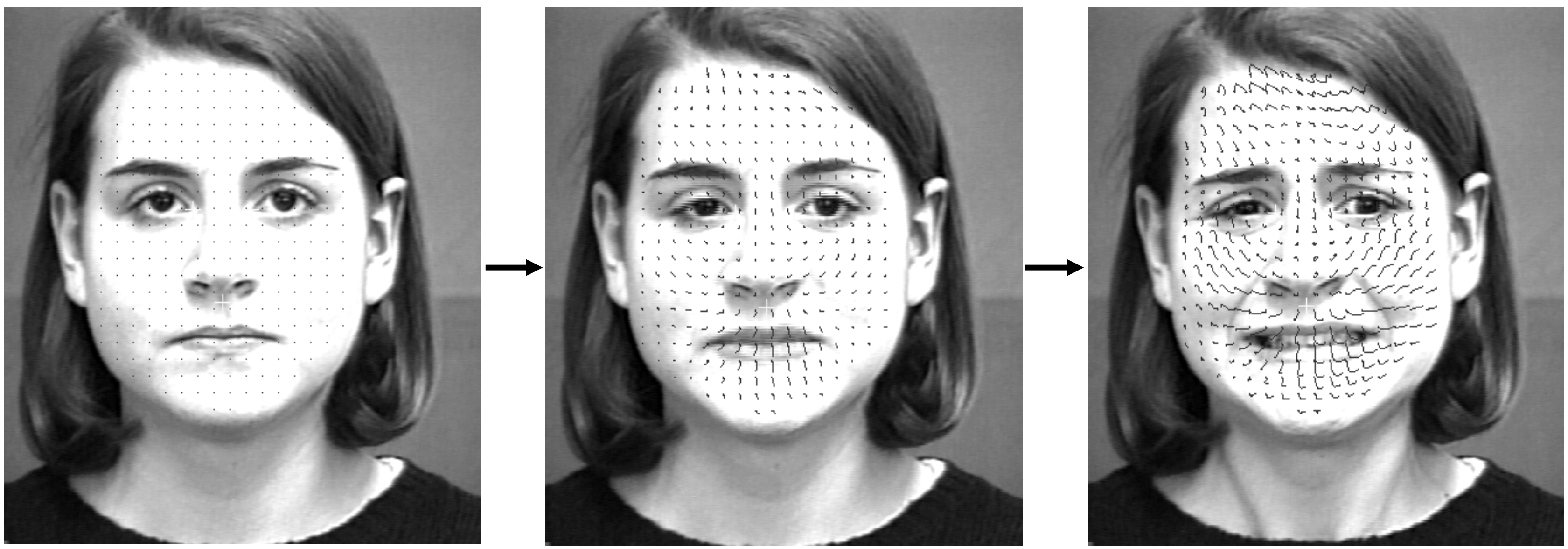
- **Affine**

Estimate the shortest distance (errors) (point to plane) over all samples

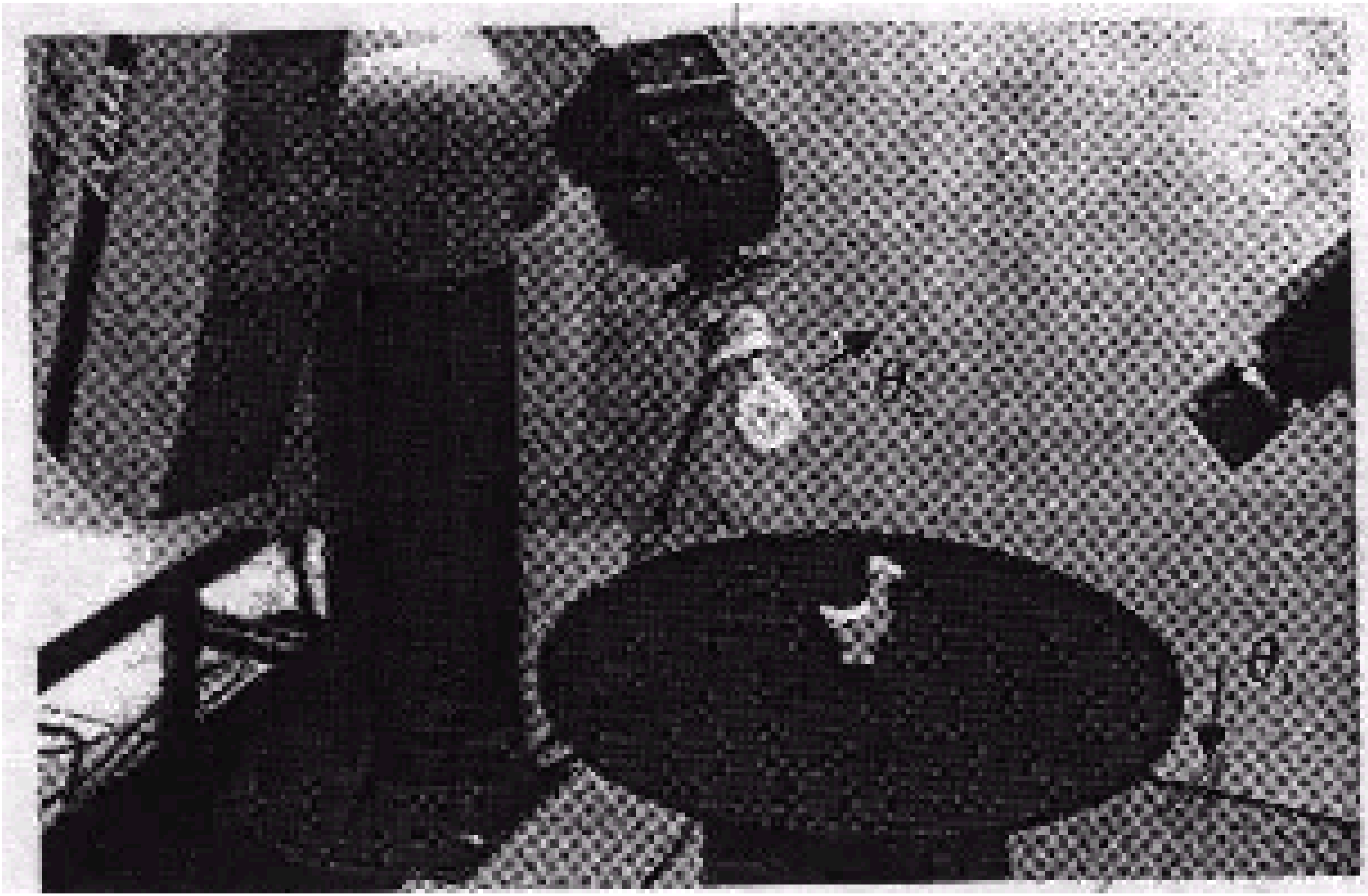


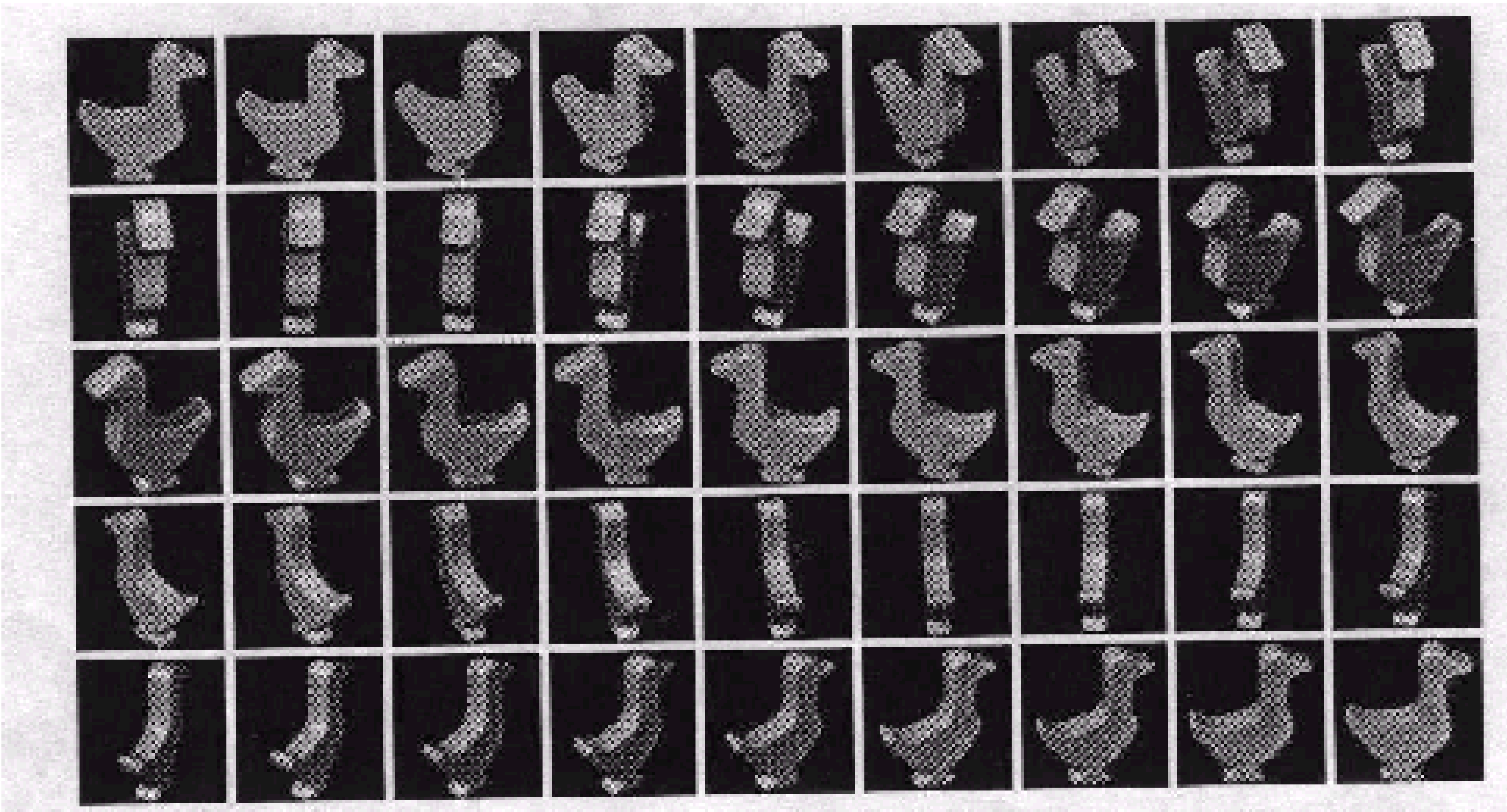
1.2 PCA Applications

1. Facial Expression Recognition: Input/output dense flows (x-axis and y-axis components)



2. Object Recognition

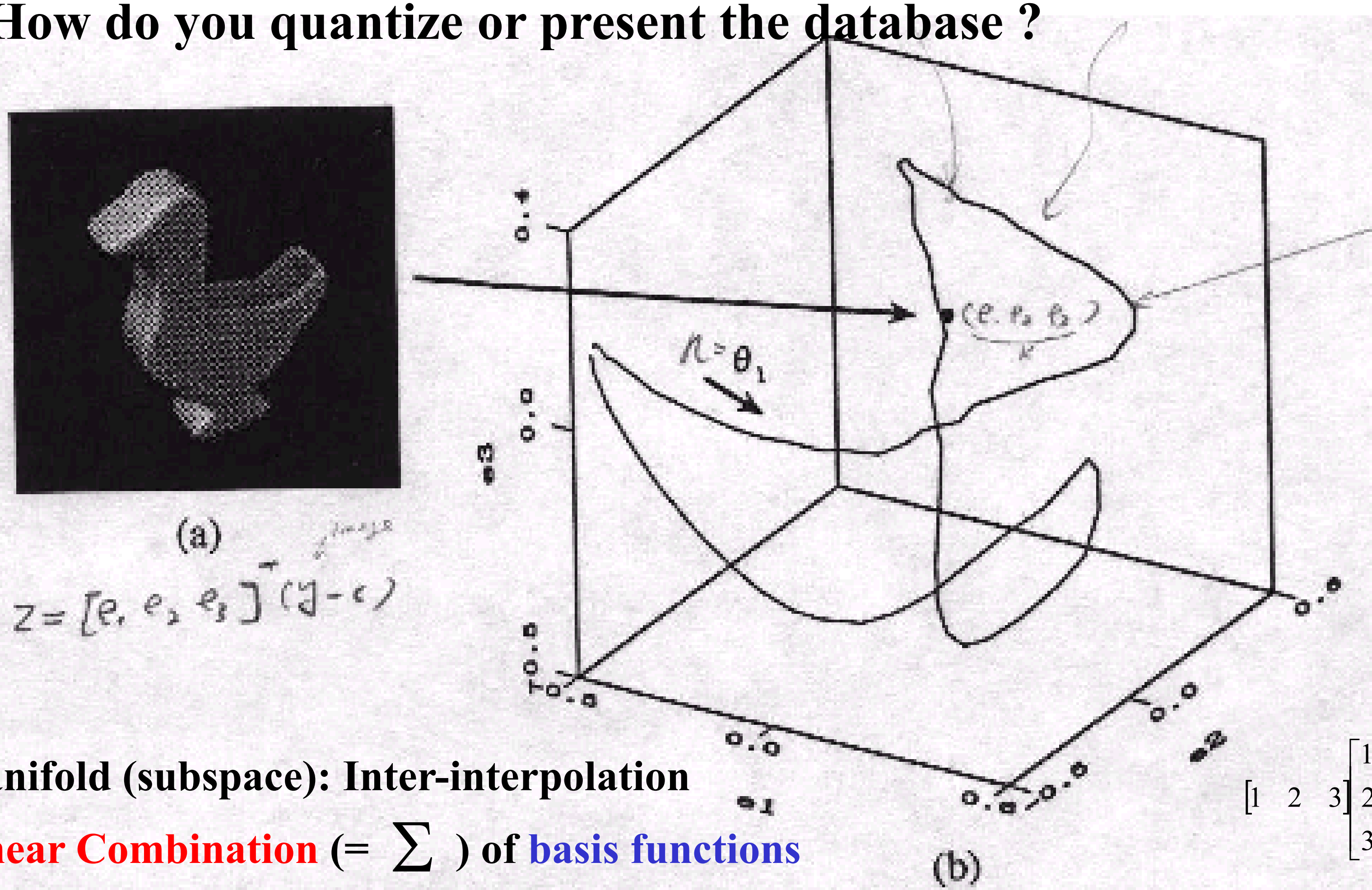




360 /（9x5） = 8 degrees

1.3 Domain Knowledge: Manifold

- How do you quantize or present the database ?



Manifold (subspace): Inter-interpolation

Linear Combination (= $\sum$) of basis functions

$$[1\ 2\ 3]\begin{bmatrix} 11 & 12 & 13 \\ 21 & & \\ 31 & & 33 \end{bmatrix}$$

(=>Homogenous Matrix or Matrix * [1 1 … 1]^T):

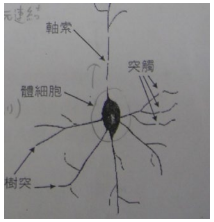
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}\begin{bmatrix} 11 & 12 & 13 \\ 21 & & \\ 31 & & 33 \end{bmatrix}$$

$$1*v_1+1*v_2+\ldots+1*v_M=\lambda_1*e_1+\ldots+\lambda_{M'}*e_{M'}+\ldots+\lambda_M*e_M$$

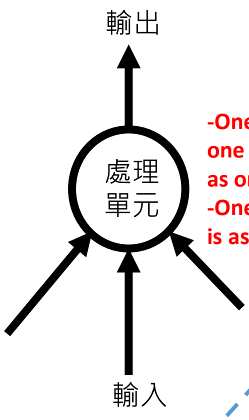
given any $v_i = \boxed{w1*e1+w2*e2+\ldots+w_{M'}*e_{M'}+\ldots}+ w_M*e_M$

Independent Vs. Orthogonal

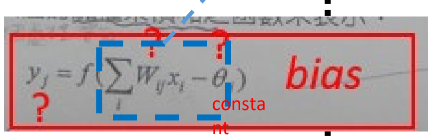
Review NN (Ch1)：



生物神經元模型



人工神經元模型



A neuron unit

-One weak classifier is one haar filter is also as one neuron unit
-One strong classifier is as one layer

Neural Network

寫程式不會以equation的方式，會以vector，matrix (Ax = b), homogenous matrix, 的方式撰寫。

Linear Combination: Aw = z

J input data vectors (here J=1)

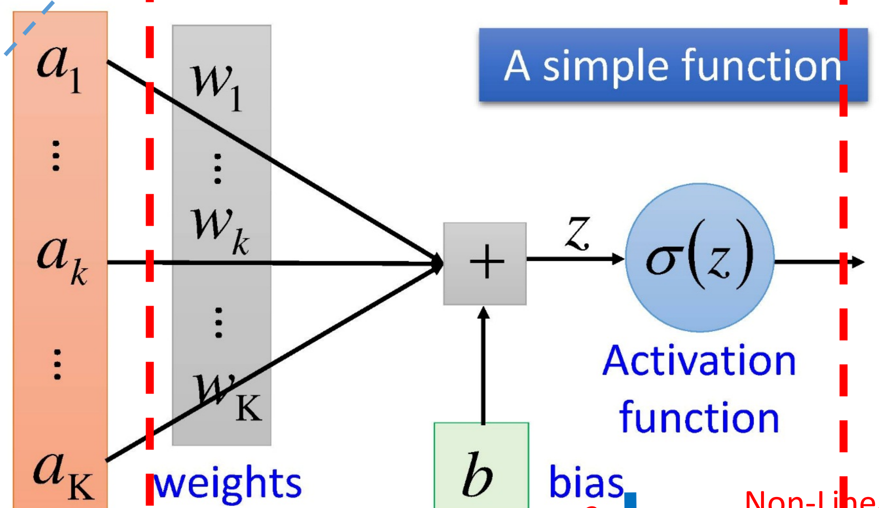
$$\begin{array}{r} \text{Feature map vector 1} \\ \\ \\ \text{Feature map vector j} \end{array} \begin{bmatrix} w_{11} w_{21} \cdots w_{k1} \cdots w_{N1} \; b_1 \\ w_{12} w_{22} \cdots w_{k2} \cdots w_{N2} \; b_2 \\ \vdots \\ w_{1j} w_{2j} \cdots w_{kj} \cdots w_{Nj} \; b_j \end{bmatrix}_{J \times (N+1)} \overset{\text{I/P}}{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \\ \vdots \\ a_N \\ 1 \end{bmatrix}}_{(N+1)\times 1} = \overset{\text{O/P}}{\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \\ \vdots \\ z_j \end{bmatrix}}_{J \times 1}$$

?

1 J-dim output result vector, here J=1)

On Neuron

Linear Combination: $z = \underbrace{a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K}_{} + b$ Constant, bias



$a=\sigma(z)$

Here, the output is a, which is one input element for next neuron

J: Can solve by EM optimization, random (such as Particle filter), ...
(可見此份投影片 p.38+3)

input：(20X20個pixel的圖) $x_1, x_2, x_3, \ldots, x_{k=400}$

One neuron

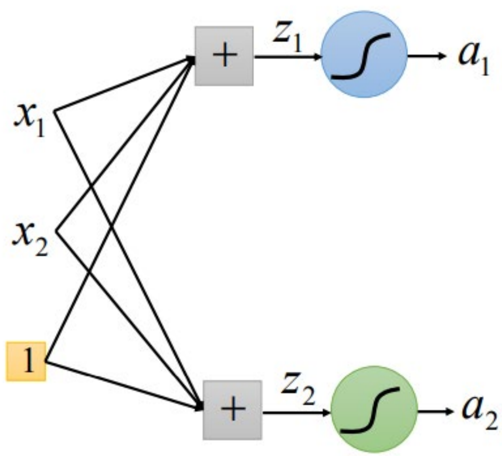
Linear Discrimination by Linear Combination

Non-Linear Discrimination by Kernel Function / Activation Function as SVM phi()

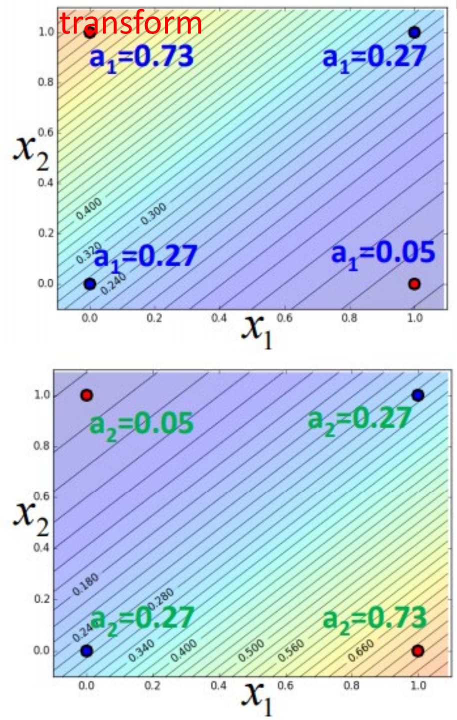
Limitation of Single Layer 2/2

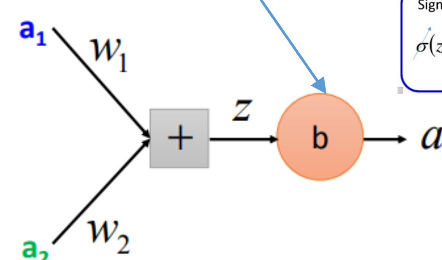
J:
1) After we transform from (x1,x2) space to (a1,a2) space,
2) then we can separate by linear discrimination
3.1) That is, instead of using SVM directly backproject to very high dimension by using kernel function or high dimensional activation function,
3.2) we can apply more layers (deeper network) to reach the same goal – space transformation layer by layer (step by step) and not so high dimension transform/activation function (nonlinear discrimination but as Fuzzy)

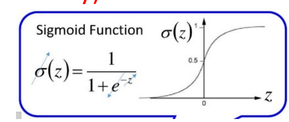
J: a1 - As eigenvector 1
a2 – As eigenvector 2



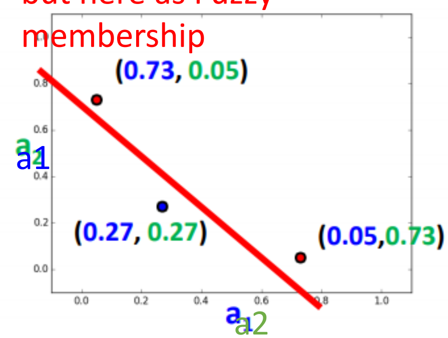
Space domain transform

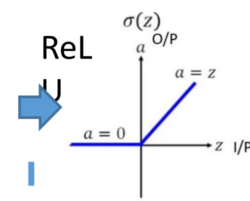


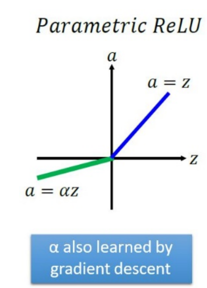


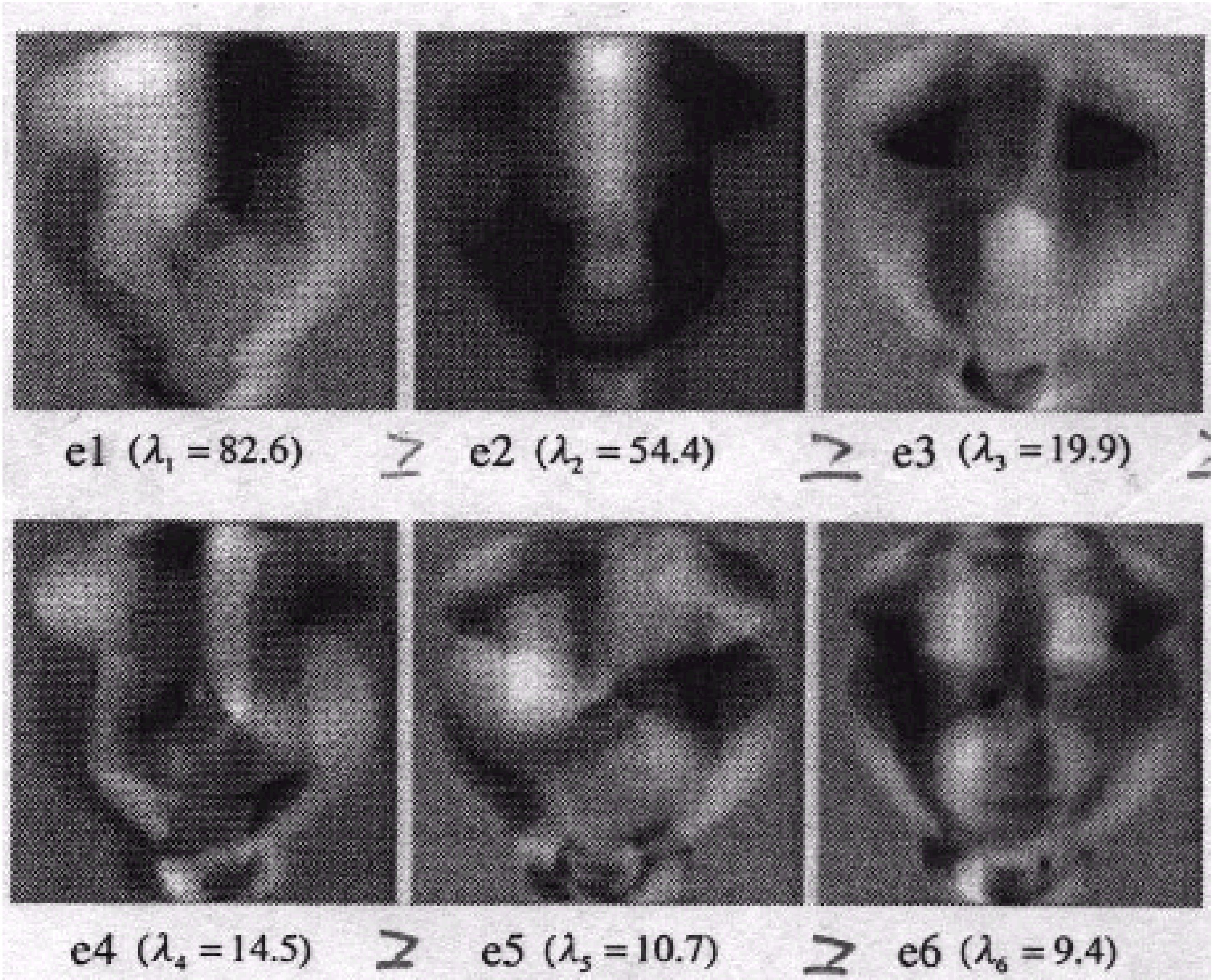


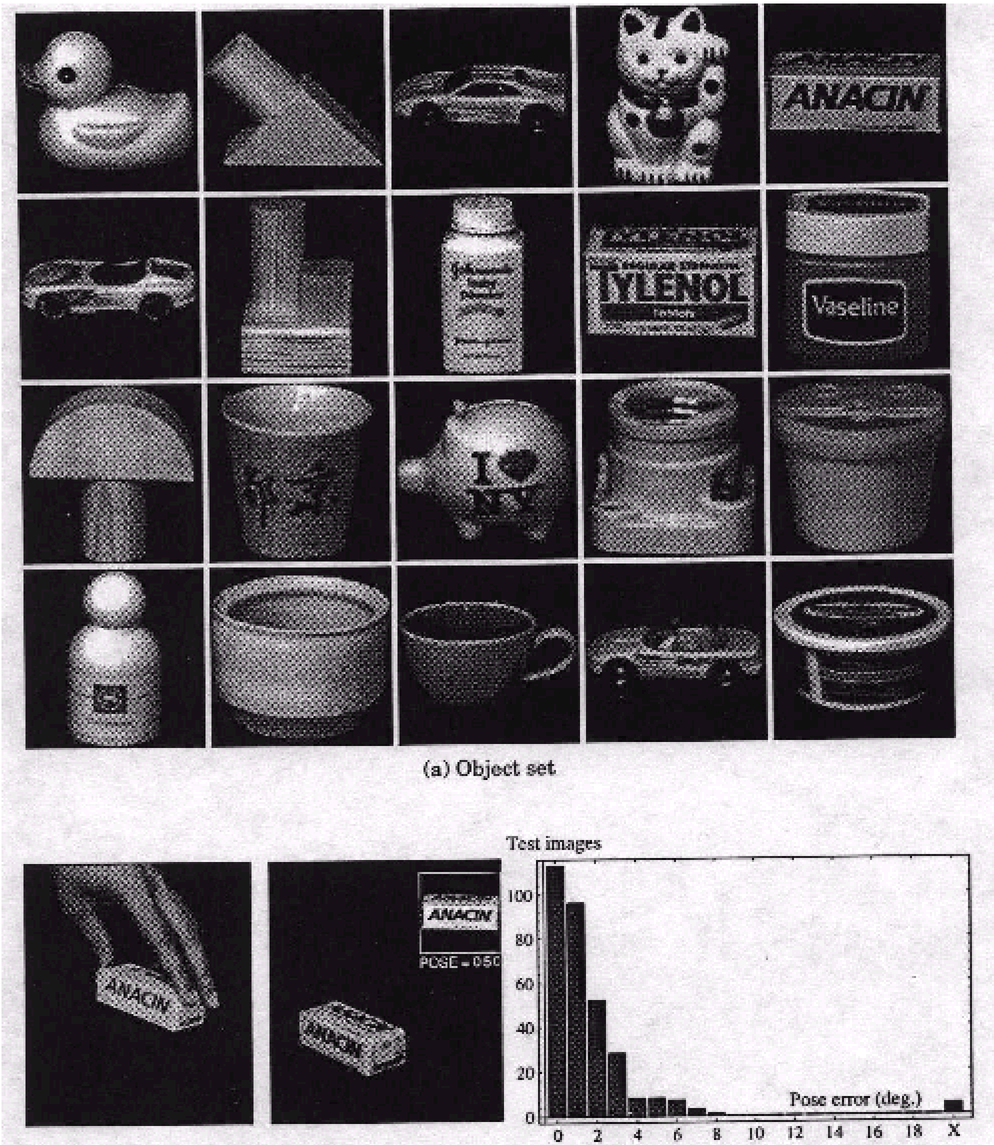
Non-Linear discriminations but here as Fuzzy membership







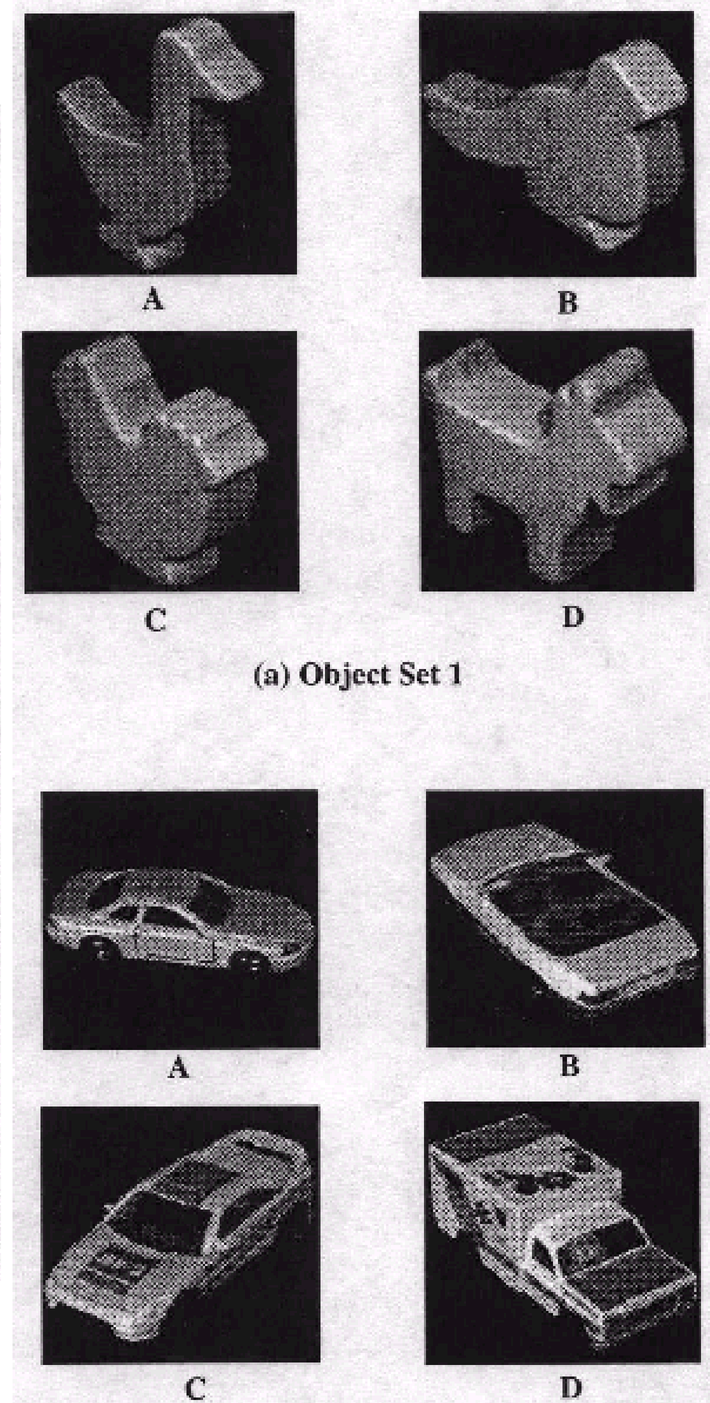

e1 ($\lambda_1 = 82.6$) ≥ e2 ($\lambda_2 = 54.4$) ≥ e3 ($\lambda_3 = 19.9$) ≥
e4 ($\lambda_4 = 14.5$) ≥ e5 ($\lambda_5 = 10.7$) ≥ e6 ($\lambda_6 = 9.4$)



(a) Object set

(b) Real-time recognition

(c) Pose estimation accuracy

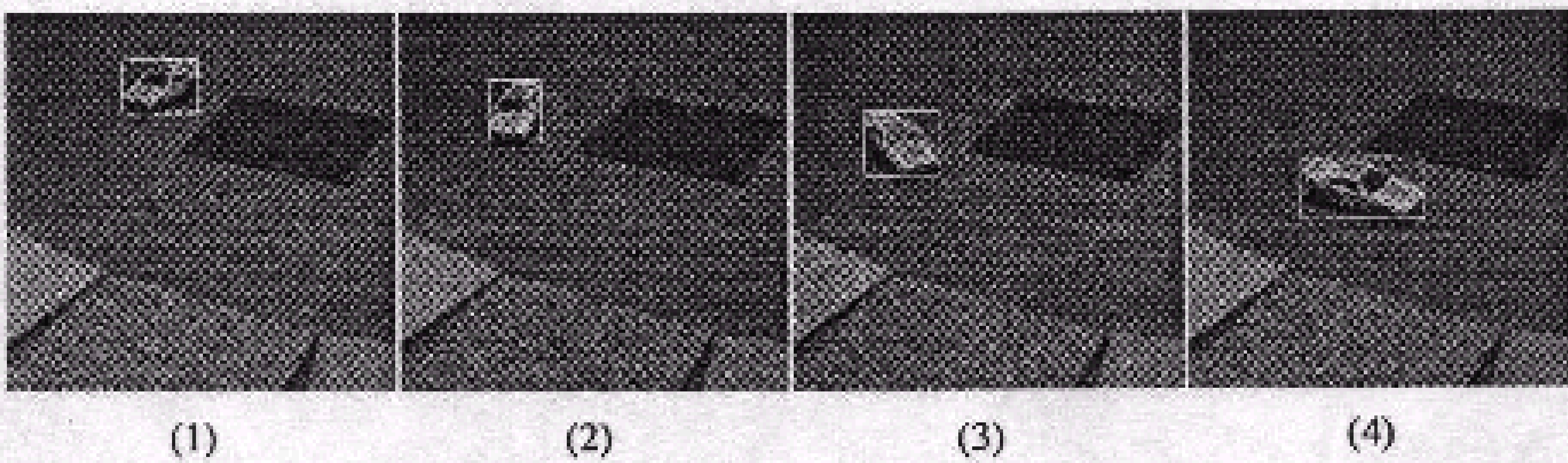


A B C D

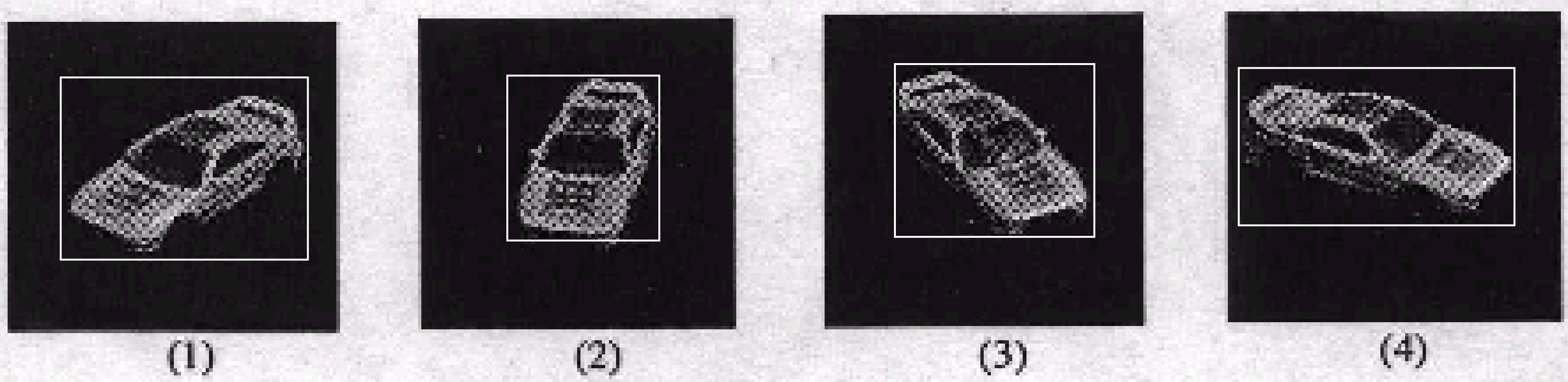
(a) Object Set 1

A B C D

(b) Object Set 2



(a) Automatic segmentation of the moving object.

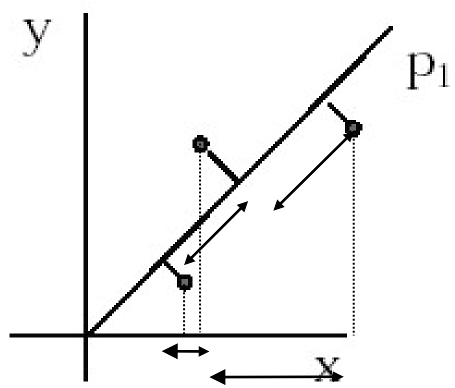


(b) Learning sample with closest pose.

2. PCA Applications: Face Recognition

- Database - Face Recognition: Input/output gray values

- We are given:
  - a set of M objects" (images)
  - each is represented, initially, by a set of N^2 features (128^2) ex. or pixels
  - This data is organized as an (very large!) N^2xM matrix
- Let's look at a small example



projection

- points are 2-D points
- we find the axis that most closely passes through these points
- if the axis passed exactly through these points, then we would need only one coordinate to represent each point.

Step 1 – Covariance Matrix

2.1.1 Computation of PCA

1) **Preprocessing: Normalization**
   - Geometric normalization (ex. Affine transformation) (Template)
   - Illumination normalization: ex. Histogram equalization => Purpose ?
2) **Let face images in the training set be** $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$
   - Represent the images as column vectors, e.g., an image Γ_i of size NxN becomes an N^2x1 column vector or row vector $[\ ,\ ,\ ,\ldots,\ ]^T$
3) **The average/mean face of the set is:**

   (an N^2x1 column vector)

$$\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n$$

4) **Each face differing from the average one is the vector: ➔ Covariance matrix**

   (i.e., centralized image vector unbiased $\Phi_i = \Gamma_i - \Psi$

   i.e., deviations/variances from the mean face => Covariance matrix

   i.e., absolute difference => relative difference)

   (an N^2x1 column vector)

Feel enhancing ?

Γ_i Ψ $\Phi_i = \Gamma_i - \Psi$

Plate 1. From left to right: sample face, average taken over extended ensemble, caricature of sample face.

$$\Phi_i = \Gamma_i - \Psi \qquad \Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n$$

- PCA seeks the axis which the cloud of points are closest to
  - this is mathematically identical to find the axis on which the variance of the point projections is greatest (that is, on which the projections are most spread out).
    spreading out to be easily separated.)
    Vs. LDA (within+btw)
  - for high dimensional objects, like pictures, it is unlikely that there will be a single axis that passes close to all of the objects.
    - So, in this case, after we find the best axis (u_1), we then find the next best one (orthogonal to the first - u_2), and then the third best (u_3), etc.
    - weight
    - Images are then represented by their projections on these axes: $w_i = \Phi_i \bullet u_i$. This is exactly analogous to the Fourier transform, with the u_i replacing the sinusoids.

Basis function

Why orthogonal ? => Liner combination

2.1.2-1 From Covariance Matrix to Eigenvectors and Eigenvalues (see DIP book)

I) In high dimension, you can not solve by equation process

- The principle components are the eigenvectors of the covariance matrix
- Covariance matrix:

Pixel over all M training images

(Linear Combination => Matrix)

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

Note : this is an outer product

II: Integral Image

$$C=\frac{1}{M}\sum_{n=1}^{M}(\Gamma_n-\Psi)(\Gamma_n-\Psi)^T=\frac{1}{M}\left(\sum_{n=1}^{M}\Gamma_n\Gamma_n^T\right)-\Psi\Psi^T=C'-D=E[\Gamma^2]-E[\Gamma]^2$$

Covariance matrix

Correlation matrix

$$\Psi=\frac{1}{M}\sum_{n=1}^{M}\Gamma_n$$

Correlation matrix

$$A=[\Phi_1,\Phi_2,\ldots,\Phi_M]$$

A is a N^2xM matrix, so C is an N^2xN^2 huge symmetric matrix ! Wow !!

By the way,

C is positive semi-definite: $uCu^T \geq 0 \quad for\ all\ u\neq 0$

Square term $= W >= 0$

$U^TCU=W$ *landa*^{2}

Singular ?

Non-invertible

So the eigenvalues of a positive semi-definite matrix are real and non-negative.

ex. Covariance matrix

Square term **C, W: Covariance matrix**

$$u = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{N^2} \end{bmatrix} = \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1N^2} \\ e_{21} & e_{22} & \cdots & e_{2N^2} \\ \vdots & \vdots & \cdots & \vdots \\ e_{N^2 1} & e_{N^2 2} & \cdots & e_{N^2 N^2} \end{bmatrix}$$

0 Actually only M objects not N^2

Let $\lambda = \begin{bmatrix} 0 \\ \lambda_i \\ \vdots \\ 0 \end{bmatrix} = u(\Gamma_i - \Psi)$

Average variance

$$\Psi_\lambda = \frac{1}{N^2}\sum_{i=1}^{N^2} u(\Gamma_i - \Psi) = u\left(\frac{1}{N^2}\sum_{i=1}^{N^2}\Gamma_i - \frac{1}{N^2}N^2\Psi\right) = 0$$

??

Actually only M objects not N^2

$$C_\lambda = \frac{1}{N^2}\sum_{i=1}^{N^2}(\lambda_i - \Psi_\lambda)(\lambda_i - \Psi_\lambda)^T = \frac{1}{N^2}\sum_{i=1}^{N^2}\lambda_i\lambda_i^T - \boxed{\Psi_\lambda\Psi_\lambda^T} = E[\lambda^2] - (E[\lambda])^2$$

$$= \frac{1}{N^2}\sum_{i=1}^{N^2}\lambda_i\lambda_i^T = \frac{1}{N^2}\sum_{i=1}^{N^2}u(\Gamma_i - \Psi)\cdot[u(\Gamma_i - \Psi)]^T$$

$$= \frac{1}{N^2}\sum_{i=1}^{N^2}u(\Gamma_i - \Psi)(\Gamma_i - \Psi)^T u^T = u\left(\frac{1}{N^2}\sum_{i=1}^{N^2}(\Gamma_i - \Psi)(\Gamma_i - \Psi)^T\right)u^T$$

$$\boxed{= uCu^T (= W) \geq 0}$$

( )²>=0

Square term

DIP book

$1*v_1+1*v_2+\ldots+1*v_M=\lambda_1*e_1+\ldots+\lambda_{M'}*e_{M'}+\ldots+\lambda_M*e_M$

Step 2.1 –
Solve Covariance Matrix to Eigen by SVD

given any $\boxed{v_i = w1*e1+w2*e2+\ldots+w_{M'}*e_{M'}}+\ldots+w_M*e_M$

II) In high dimension, you can solve by SVD matrix factorization process

- Eigenvalues are numbers λ such that $Cu = \lambda u$ for some vectors $u \neq 0$.

  (A u = Lamda v)

- Eigenvectors are the vectors u such that $Cu = \lambda u$. They are orthogonal to each other.
- For general matrices, the eigenvalues are complex numbers.
- The eigenvalues of a positive semi-definite matrix are real and non-negative.

In reality, for example, by using book - numerical recipes, negative value happens - error.

$CU=WU$ $\quad$ $C=U^TWU$

$U^TCU=W$ $\quad$ $landa^2$

- **$C = AA^T = UWU^T$, W => Lamda^2**
- **$A=UWV^T$, W => Lamda =?**

N^2-d eigenface

$$u = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{N^2} \end{bmatrix} = \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1N^2} \\ e_{21} & e_{22} & \cdots & e_{2N^2} \\ \vdots & \vdots & \cdots & \vdots \\ e_{N^21} & e_{N^22} & \cdots & e_{N^2N^2} \end{bmatrix} \quad 0$$

Actually only M objects not N^2

- The eigenvectors are the principal components.
- Each eigenvector can be determined up to a scaling factor, so usually they are normalized to unit length:

$$u = \frac{u'}{\|u'\|}$$

vector = direction (unit vector) + magnitude (scale)

- When stacked in a matrix, the eigenvectors form the Karhunen-Loeve transform matrix u with the property $C = UWV^T$ where W is a diagonal matrix, whose diagonal entries are the eigen-values of C. U orthonormal.

$$u_l^T u_k = e_l^T e_k = \delta_{lk} = \begin{cases} 1, & if \quad l=k \\ 0, & otherwise \end{cases}$$

Orthonormal ?

Correlation/Covariance Matrix

$$C = uC_{\lambda}u^{T} = UWV^{T}$$

$$C_{\lambda} = \begin{bmatrix} \lambda_1^2 & & & 0 \\ & \lambda_2^2 & & \\ & & \ddots & \\ 0 & & & \lambda_{N^2}^2 \end{bmatrix}$$

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M \geq \underbrace{\lambda_{M+1} \geq \ldots \geq \lambda_{N^2}}_{\mathbf{=0}}$$

2.2.2-1 Solving for Eigenvectors and Eigenvalues: Method I:

- Solve for the roots of the characteristic polynomial:

$$Cu = \lambda u \qquad Cu - \lambda u = 0$$

$$(C - \lambda I)u = 0 \qquad u \neq 0$$

By Cramer's theorem, this has non-trivial solution

iff $\det(C - \lambda I) = 0$

If **C** is 2x2 :

$$\begin{bmatrix} C_{11} - \lambda & C_{12} \\ C_{21} & C_{22} - \lambda \end{bmatrix} = 0$$

$$(c_{11} - \lambda)(c_{22} - \lambda) - c_{12}c_{21} = 0$$

$$\lambda^2 - (c_{11} + c_{22})\lambda + c_{11}c_{22} - c_{12}c_{21} = 0$$

An Example

- Consider 4 points in 2D: (0,1), (2,2), (4,6), (6,7)
- Mean values: E[x1]=3, E[x2]=4
- Centered data: (-3,-3),(-1,-2), (1,2), (3,3)
- Covariance matrix C :

1=x, 2=y

$C_{11} = (9+1+1+9)/4 = 5$ $C_{12} = (9+2+2+9)/4 = 5.5$

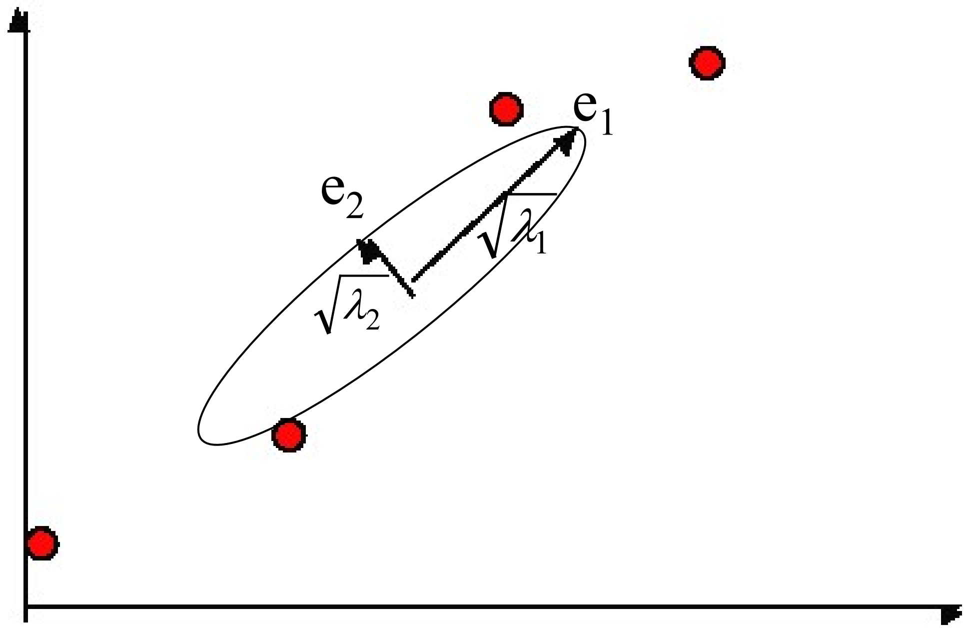
$C_{21} = (9+2+2+9)/4 = 5.5$ $C_{22} = (9+4+4+9)/4 = 6.5$

- Characteristic polynomial:

$$\lambda^2 - \underbrace{(5+6.5)}_{11.5}\lambda + \underbrace{5(6.5) - 5.5(5.5)}_{2.25} = 0$$

- Eigenvalues: $\lambda_{1,2} = (11.5 +/- \sqrt{123.25})/2 = (11.3, 0.2)$
- Eigenvectors: (0.75, 0.66), (-0.66,0.75)

Put in Lamda1 and Lamda2



- Uncertainty
- Hessian Matrix

2.2.2-2 Method II: Numerical Computation: Singular Value Decomposition (SVD)

Step 2.1 – Solve Covariance Matrix to Eigen by SVD

- A fundamental problem in science and engineering
- Singular Value Decomposition (SVD) is now the preferred method to do eigen-analysis.
- That is, there is an easier way to do PCA in using Singular Value Decomposition:

Book 'Numerical Recipes'

- $[U\ \mathbf{W}\ V^T] = \text{svd}(C)$
- U is the eigenvector arrranged in descending eigenvalues, **W** contains a diagonal matrix with descending eigenvalues.

$$U = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{N^2} \end{bmatrix} = \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1N^2} \\ e_{21} & e_{22} & \cdots & e_{2N^2} \\ \vdots & \vdots & \cdots & \vdots \\ e_{N^2 1} & e_{N^2 2} & \cdots & e_{N^2 N^2} \end{bmatrix} \quad 0$$

$$W = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_{N^2} \end{bmatrix}$$

$$0$$

2.2.3 Solving for Eigenvectors and Eigenvalues: Method for Huge Covariance Matrix

Step 2.2 – Solve Covariance Matrix to Eigen by SVD

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

$\Phi, u : N^2 x1$ *column vector* $\Phi_i = \Gamma_i - \Psi$

$A : N^2 xM$ *matrix* $A = [\Phi_1, \Phi_2, ..., \Phi_M]$

$A^T : M\ x\ N^2$ *matrix*

$$Cu = \lambda u$$

$C = AA^T, \lambda : N^2 x\ N^2$ matrix pixel base

$A^T A, v : M\ x\ M$ matrix image base

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

Pixel base $AA^T A v_i = \mu_i A v_i$

$M' < M << N^2,$

ex. $M'=7, M=16, N=128$

Image base *Eigenfaces* $\boxed{u_l} = \sum_{k=1}^{M} \boxed{v_{lk}} \Phi_k = A v_l$ *where* $l = 1,2,...,M$

Physical Meaning:
Phi_1 take/enhance nose by * v_1 +…+
Phi_M take/enhance eye by * v_M
= U_1

For each ul like extracting eye, it consists of vl over all unbiased faces

Eigenvalue/ weight $\omega_k = u_k^T(\Gamma - \Psi)$ *for* $k = 1,...,M',...,M$

2.3 Dimensional Reduction

- Given

$$C = \begin{bmatrix} 59.14 & -1.55 & 4.56 \\ -1.55 & 0.09 & -0.11 \\ 4.56 & -0.11 & 27.02 \end{bmatrix} \qquad \det(C - \lambda I) = 0$$

The resulting eigenvectors and eigenvalues are

$$u_1 = \begin{bmatrix} -0.99 \\ 0.03 \\ -0.14 \end{bmatrix}, \lambda_1 = 59.82 \quad u_2 = \begin{bmatrix} 0.14 \\ -0.00 \\ -0.99 \end{bmatrix}, \lambda_2 = 26.39 \quad u_3 = \begin{bmatrix} 0.03 \\ 1.00 \\ -0.00 \end{bmatrix}, \lambda_3 = 0.04$$

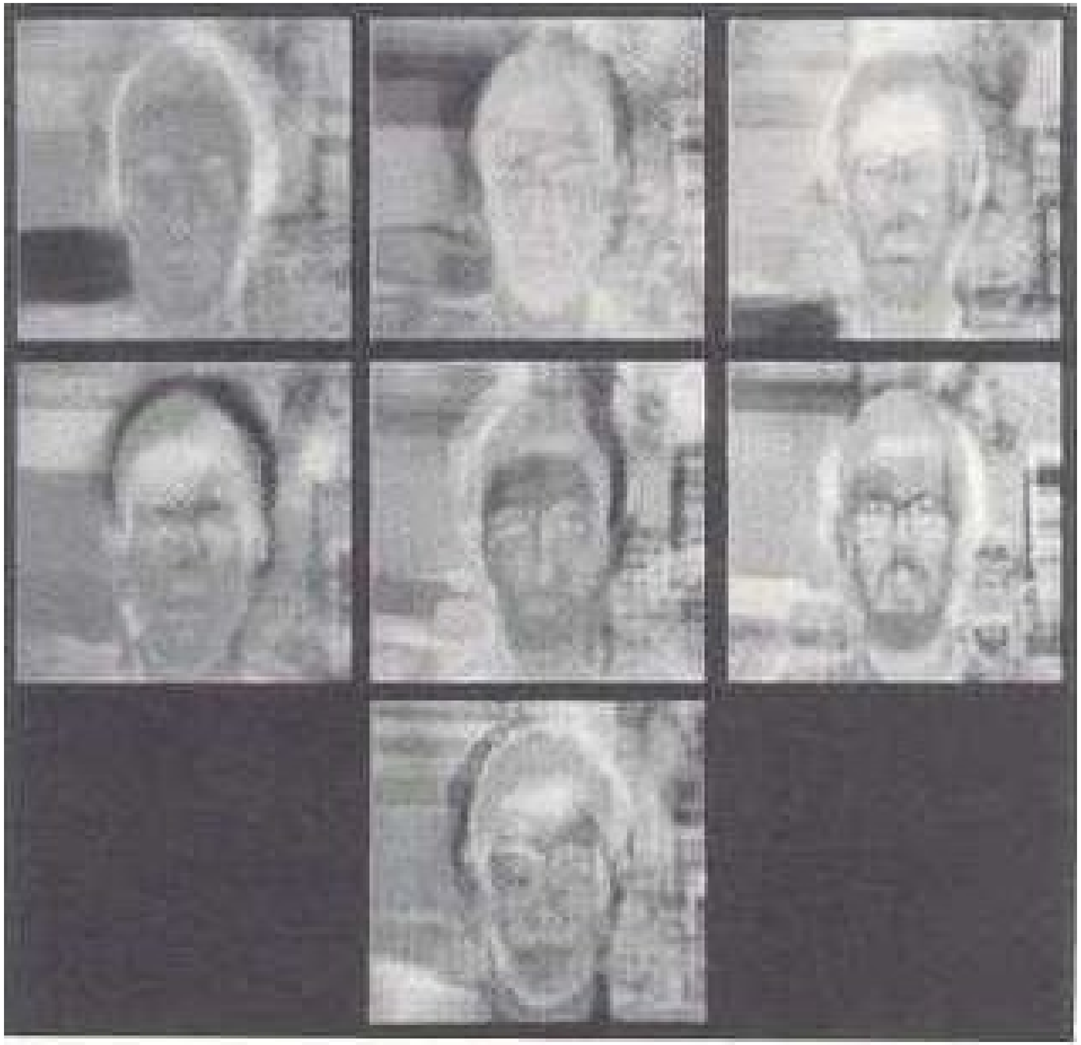
Note $\dfrac{\lambda_3}{\sum \lambda_i} < 0.001$, what is the implication?

- We can throw u_3 away, and keep $w=[\ u_1 \quad u_2\ ]$.
- You may find the different objects/textures forming nice clusters in this 2D space.
- But there is an easier way to do PCA in using Singular Value Decomposition:

Book 'Numerical Recipes'

- $[U\ W\ V^T]=svd(C)$
- U is the eigenvector arrranged in descending eigenvalues, W contains a diagonal matrix with descending eigenvalues.

- K-L transform: first basis -- data projected on which will have the maximum variance, the second basis captures the second maximum variance that is orthogonal to the first one. Once the variances are captured by some bases, the subsequent bases can be discarded.
- This helps us to achieve dimensionality reduction.

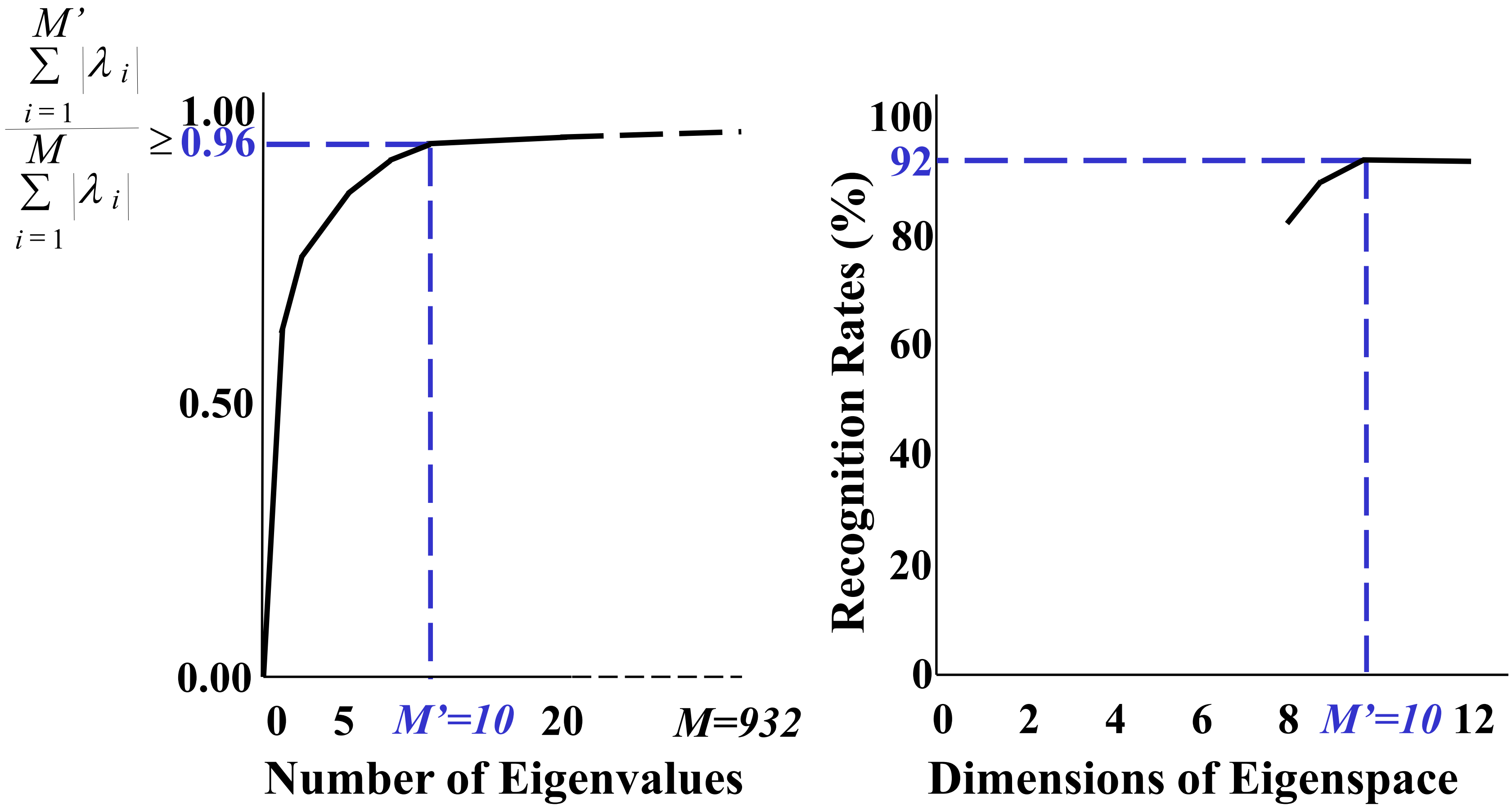


2.3 Dimension Reduction and Projection Weight Vector

Step 3 – Projection Weight Vector for Dimension Reduction

- Given your gallery of images
  - compute its principal components eigenvector u
    - this is just a set of other images that are used as a basis for representing the images in the gallery
  - determine an $M' << M$ such that the first M' principal components u are a "good" representation for the gallery
    - can be chosen based on the scores (eigenvalues) computed by the PCA
  - represent each image Γ in the gallery by its projection on these M' principal components $\longrightarrow$ Eigenvalue or weight
    - this is just the dot product of the image and the principal components. $\omega_k = u_k^T(\Gamma - \Psi) \;\; for\; k = 1,\ldots,M',\ldots,M$
    - each image now represented by M' numbers

$$\Omega^T = [\omega_1, \omega_2 \ldots \omega_{M'}]$$



$$\lambda_1 \geq .. \geq \lambda_{M'} \geq .. \geq \lambda_M$$

$$\omega_k = u_k^T(\Gamma - \Psi) \quad for\ k = 1, \ldots, M', \ldots, M$$

2.4 Reconstruction

Step 4 – Reconstruction as GAN

- If we compute M principal axes, then we can reconstruct any image exactly from its principal components representation:

$$\Phi_f = \sum_{i=1}^{M} \omega_i u_i \qquad \Gamma_f = \Phi_f + \Psi$$

- This is just another basis for the M-vector that represents the image
  - the original basis is the natural one - (1,0,0,...0), (0,1,0,...) ...
  - the principal axes represent just a rotation of the original high dimensional coordinate system

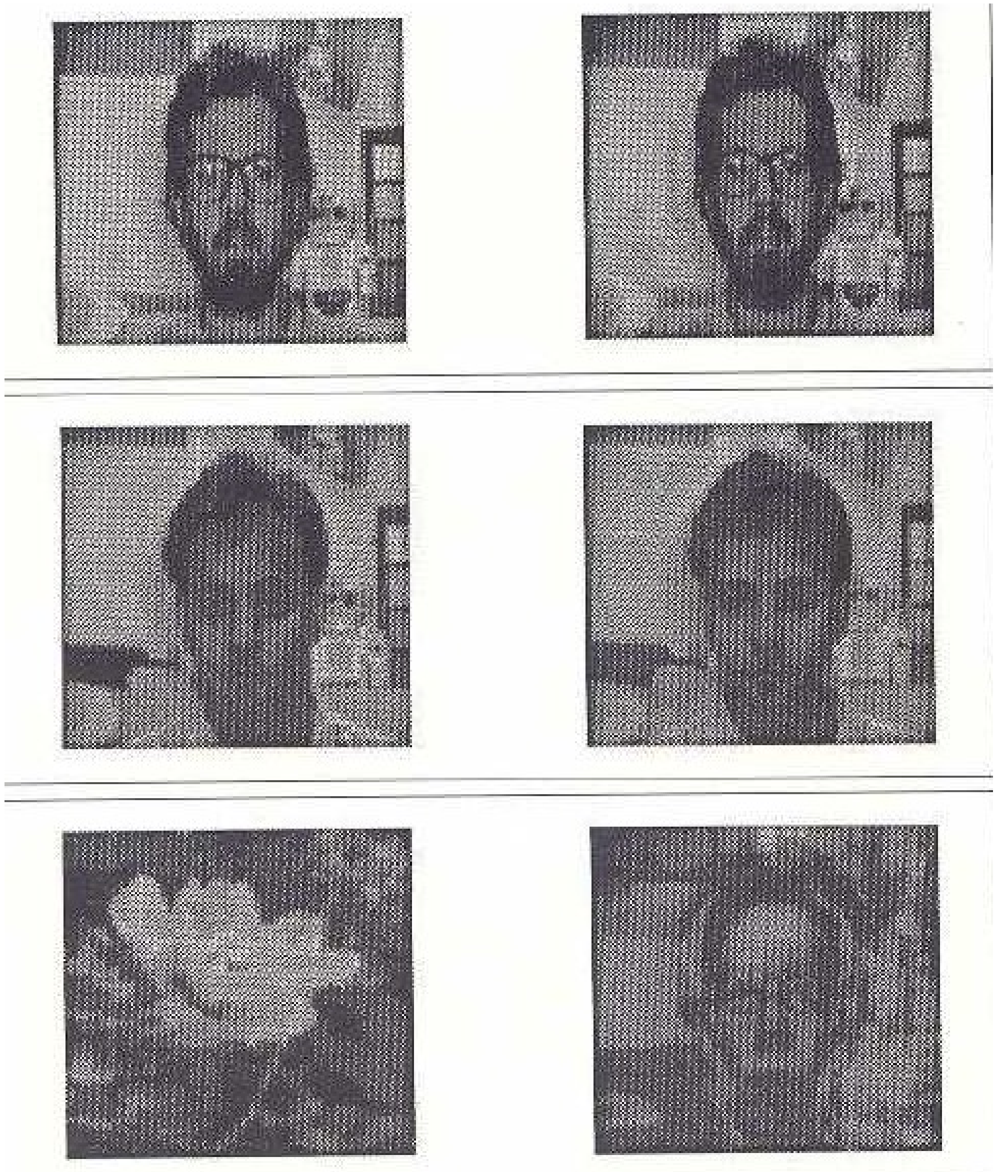
See lecture: Camera model example => World coordinate Vs. Camera Coordinate

- However, we don't need to use all of the principal axes to obtain good reconstructions of the image.
- The mathematical procedure that determines the principal axes uses an eigenvector analysis, and associates a "score" with each axis eigenvalue
  - these scores correspond to the amount of variation in the image set that the axis corresponds to and are the eigenvalues of the procedure
  - The scores generally go to zero "quickly". For a face database, we can generally reconstruct a 512x512 face using only 80-100 principal axes with very small error.

$$\Phi_f = \sum_{i=1}^{M'} \omega_i u_i \qquad \Gamma_f = \Phi_f + \Psi$$

Lost details

Novel Input Image Reconstruction



Step 5 - Recognition

2.5 Recognition

- Given an unknown image
  - compute its projection onto the principal component basis
    - this is a set of M'numbers representing the unknown image $\Omega^T = [\omega_1, \omega_2 \ldots . \omega_{M'}]$
  - compare this M'-tuple against each of the database image M'-tuples
    - simple L^2 norm $\| \ \|^2$
    - sometimes each component is weighted by the associated eigenvalue **Lamda**

J: ?? w_i is normalized by dividing corresponding eigenvalue Lamda_i

Step 6 – Eigen by NN

2.6 PCA by NN

1. Face Detection: Distance From Face Space

$$\varepsilon^2 = \left\| \Phi - \Phi_f \right\|^2 \qquad \Phi_f = \sum_{i=1}^{M'} \omega_i u_i$$

input face

Euclidean Distance

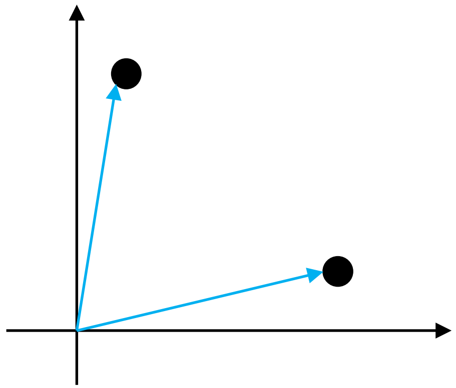
2. Face Classification

$$\Omega^T = [\omega_1, \omega_2 \ldots . \omega_{M'}]$$

Minimize the Euclidian distance

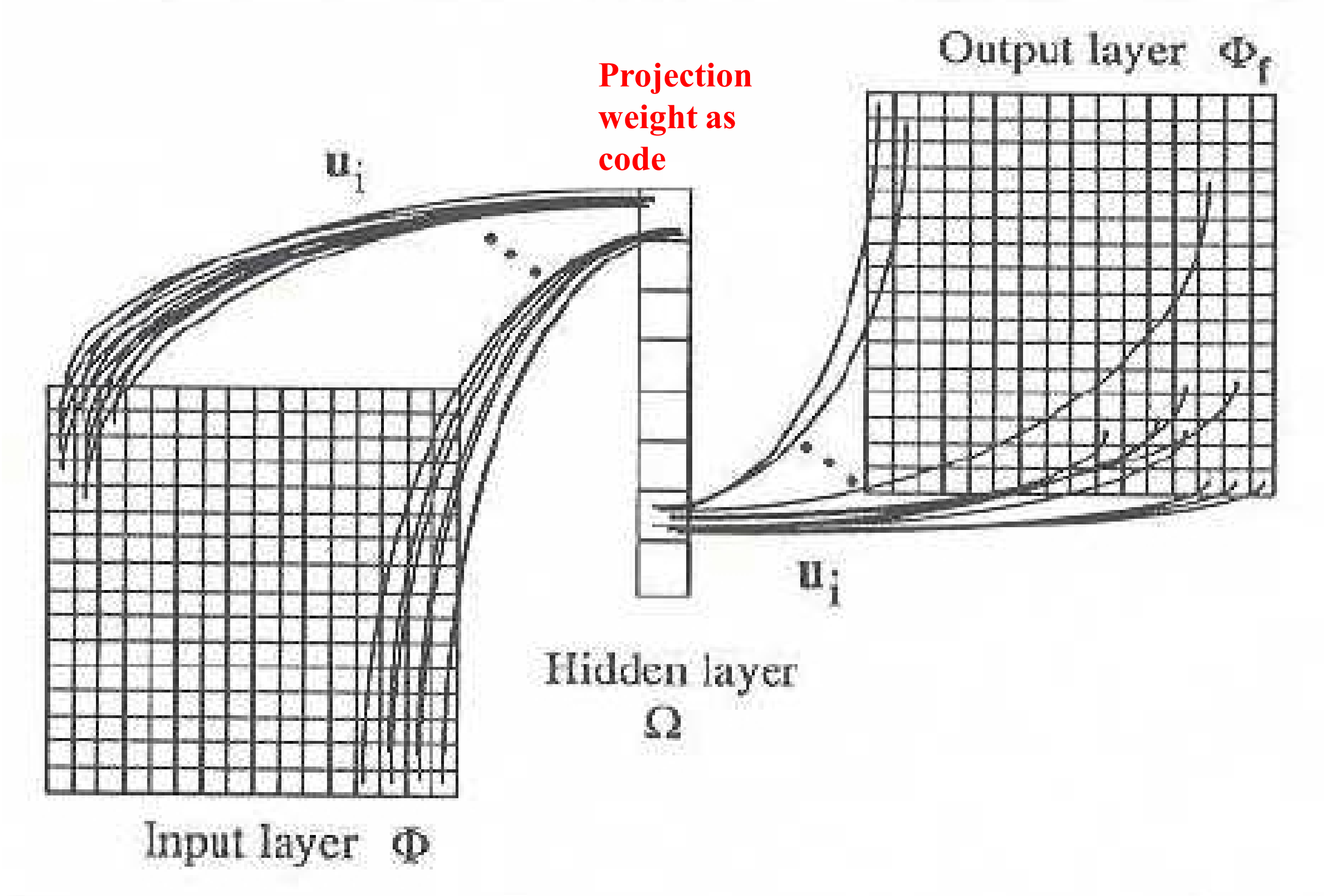
$$\varepsilon_k = \left\| \Omega - \Omega_k \right\|^2 \quad 1 \le k \le M$$

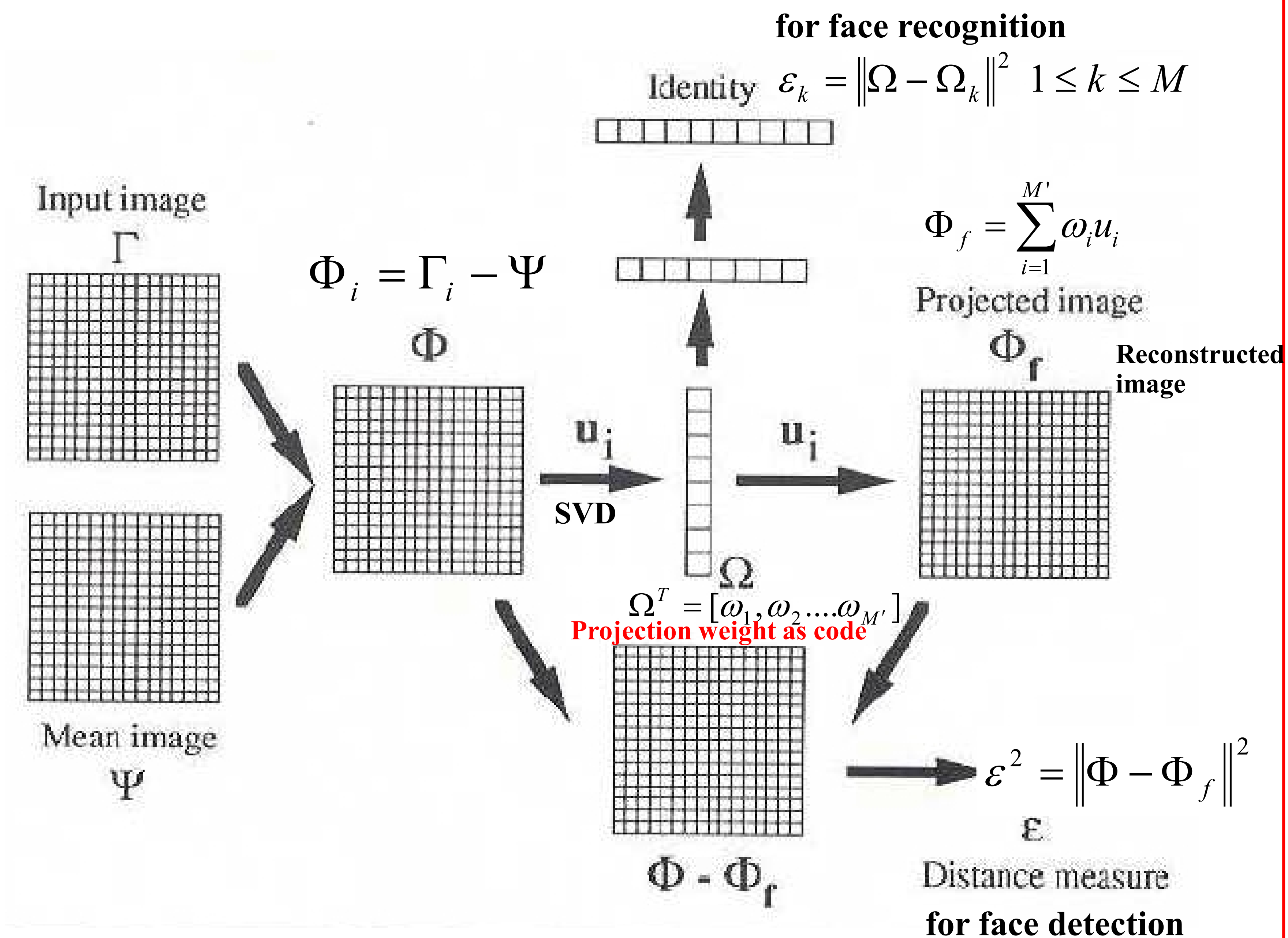
Novel face　　Known face class k



Same distance (magnitude), but different samples or vectors (directions)

PCA by Neural Networks




for face recognition
$\varepsilon_k = \|\Omega - \Omega_k\|^2 \quad 1 \le k \le M$
Identity
Input image
Γ
$\Phi_i = \Gamma_i - \Psi$
Φ
$\Phi_f = \sum_{i=1}^{M'} \omega_i u_i$
Projected image
Φ_f
Reconstructed image
u_i
SVD
u_i
Ω
$\Omega^T = [\omega_1, \omega_2 \ldots \omega_{M'}]$
Projection weight as code
Mean image
Ψ
$\varepsilon^2 = \|\Phi - \Phi_f\|^2$
ε
Distance measure
$\Phi - \Phi_f$
for face detection

3. Discussion

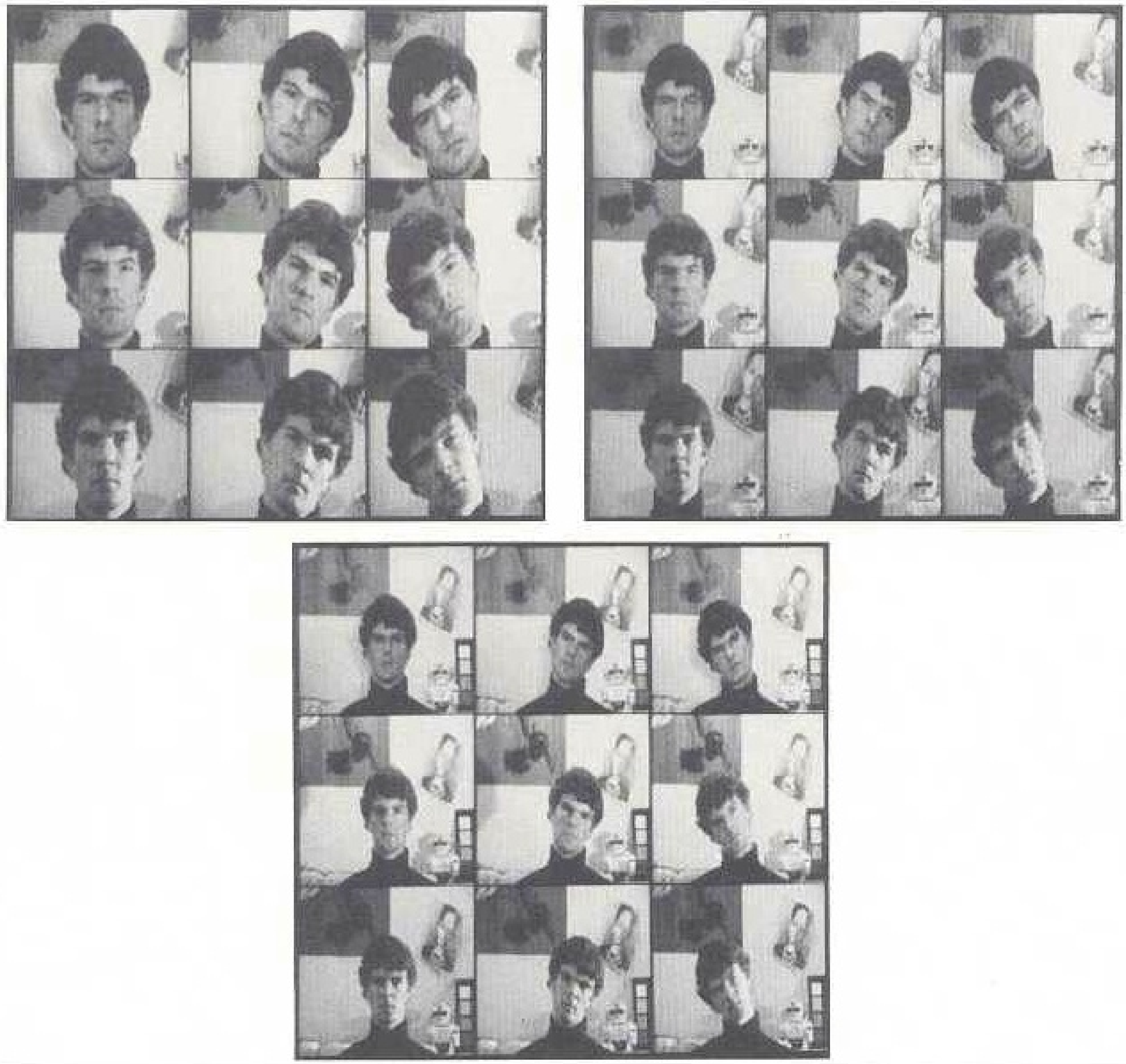
- **PCA Properties:**
  - Compression
  - Correlation
- **Normalization**
  - Histogram equalization
  - Geometric normalization

Normalize training data

or testing data ?

- **Recognition Rate: Lighting Variation (96%) > (In-Plane) Orientation Variation (85%) > Scale Variation (64%) (out-of-plane rotation ??)**
  - Under lighting changes alone the neighborhood pixel correlation remains high.
  - Under size changes, the correlation from one image to another is largely lost

Same Class



If Having New Data to PCA?

Other (Face) Recognition Methods

- **Linear Discriminant Analysis (LDA)**
  - Fisher's Discriminant, Within + Between, one subject has many images

    PCA => linear discriminant, but considering between only, one subject has only one image
- **Non-Linear Discriminant**
  - Support Vector Machine: Kernel Function (to high dimensional space ?)

Applications

- Face recognition -- the eigenface method by Pentland and Turk .
  - Collect images of human faces in the same pose and label them with the name of the person;
  - find the first 7 principle components of the images -- resemble faces, called eigenfaces.
  - For each image, store the expansion coefficients for the corresponding eigenfaces, and when an unknown face arrives, find its expansion in terms of eigenfaces and assign to it the name of the closest image in eigenspace.

$$\Omega^T = [\omega_1, \omega_2 \ldots . \omega_{M'}]$$

- Face Detection -

Principal Components

- Image compression: Use only main PCs and their coefficients in representation. Gabor wavelets are the main PC of natural images . Strategy used in Face detection algorithm of Schneiderman and Kanade.

- Facial expression recognition

  Use flow base, not gray-value base, to ignore differences across individual subjects

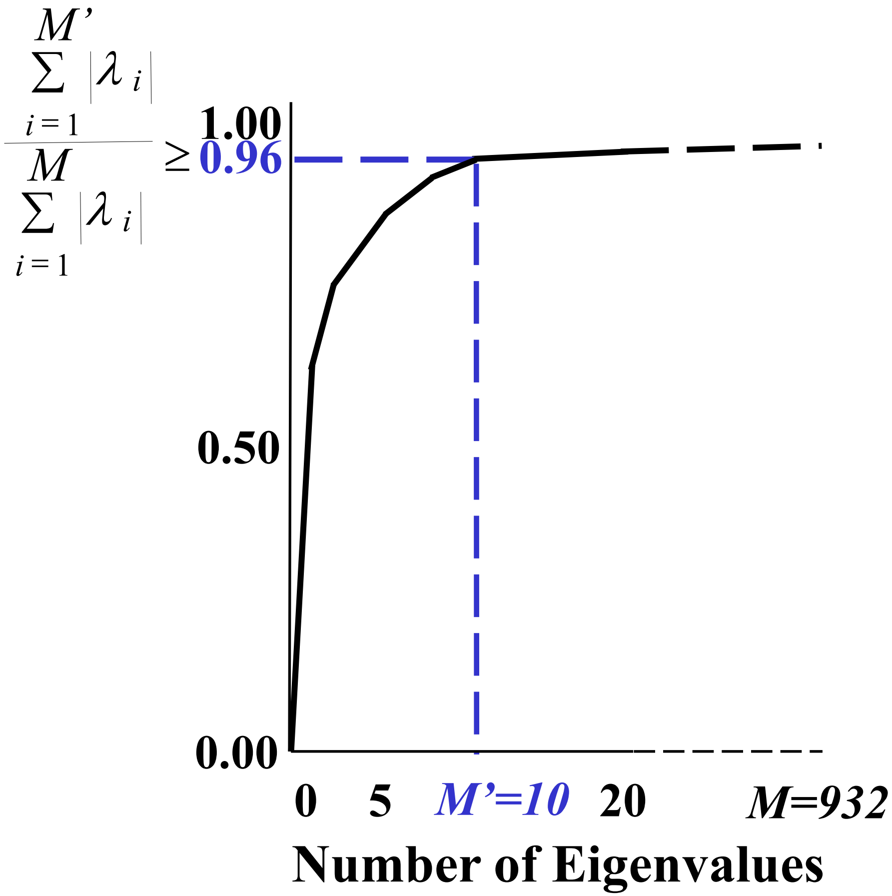
  PCA properties:

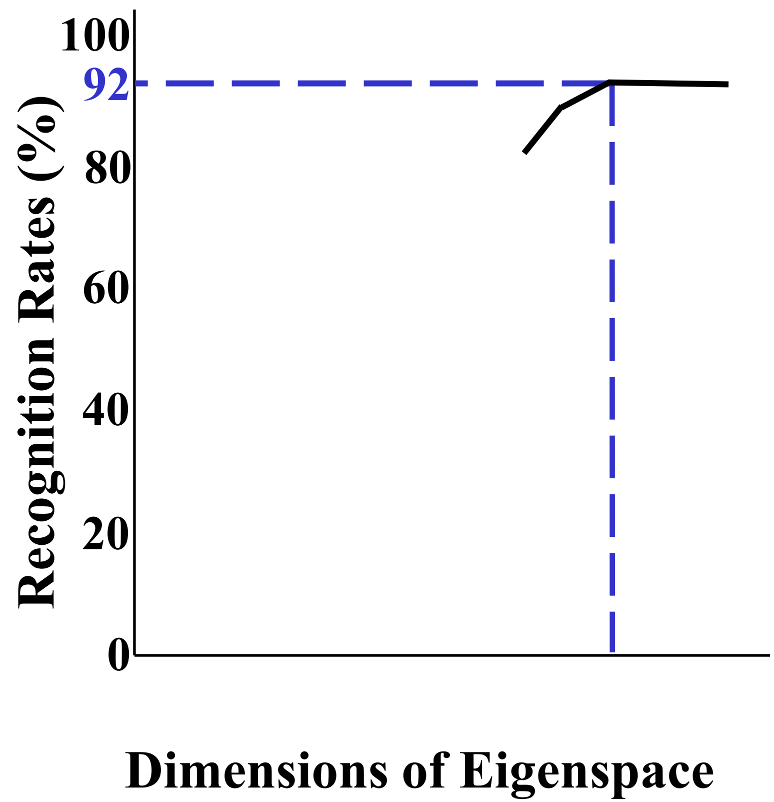
    Compression

    Correlation

❑ Facial expression recognition

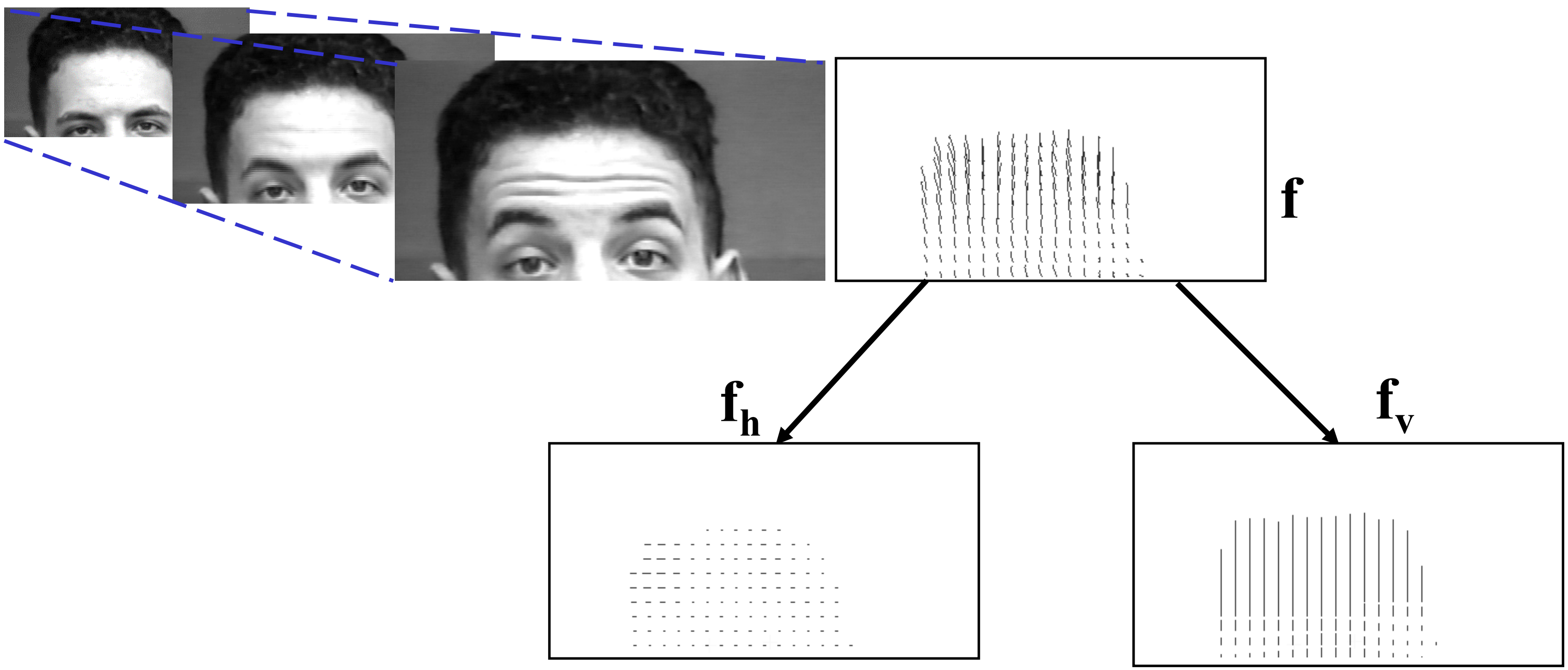
Computation of (horizontal) eigenflow number (upper facial expression)





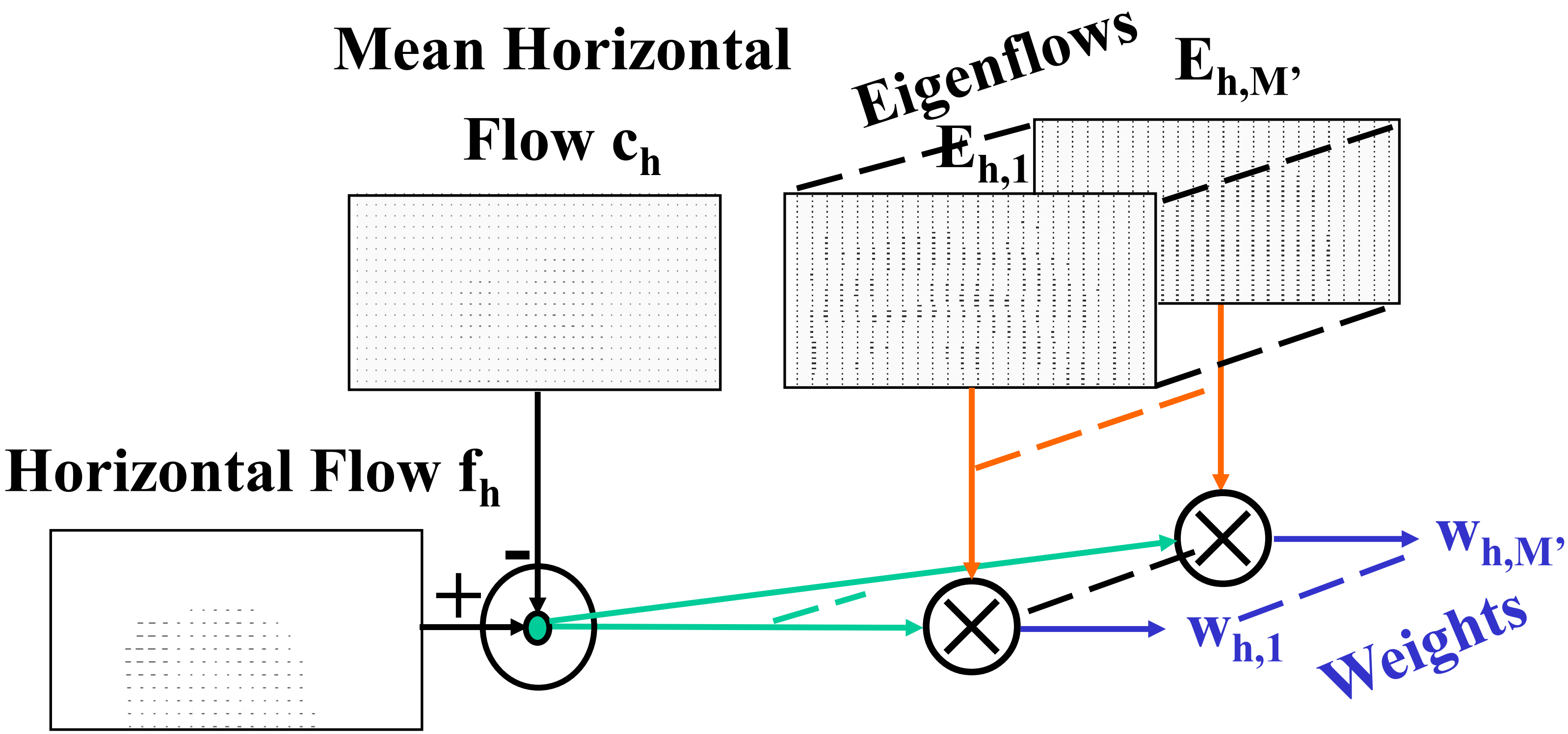
$$\lambda_1 \geq .. \geq \lambda_{M'} \geq .. \geq \lambda_M$$

Input flow image



Weight vector for horizontal flow components

(upper facial expression)

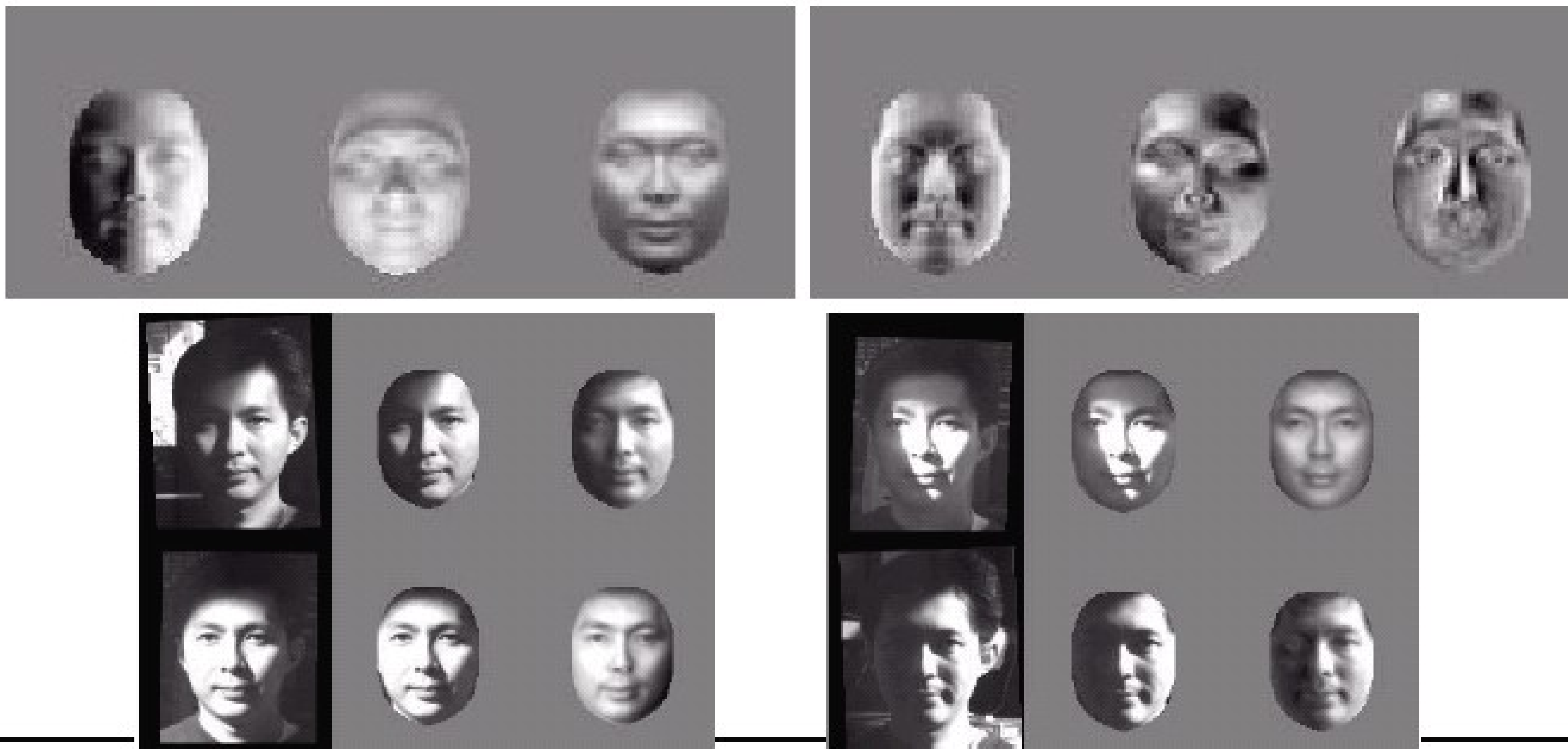


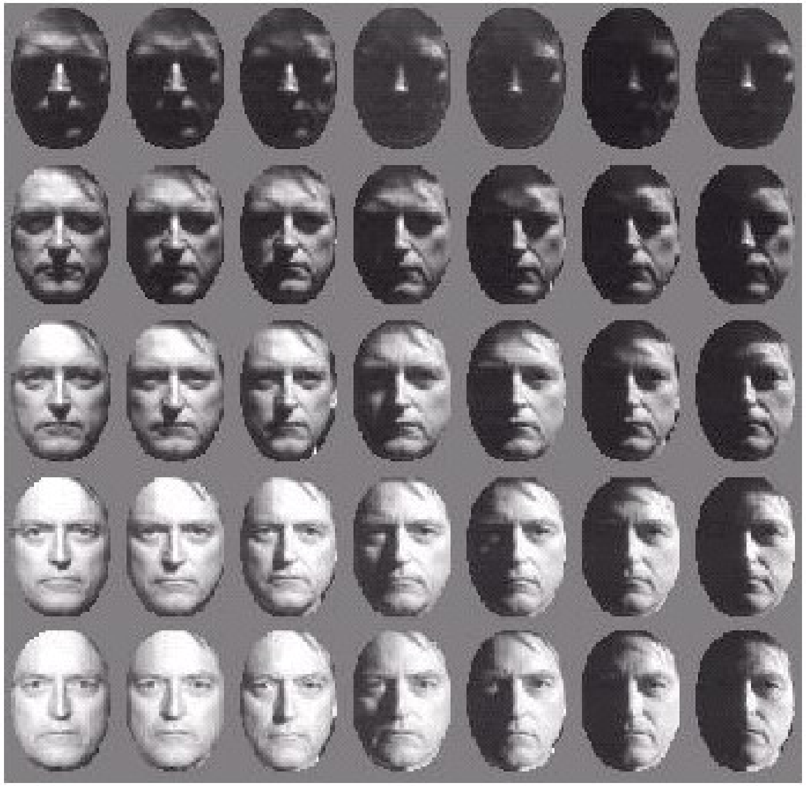
Wheelchair: EigenButts

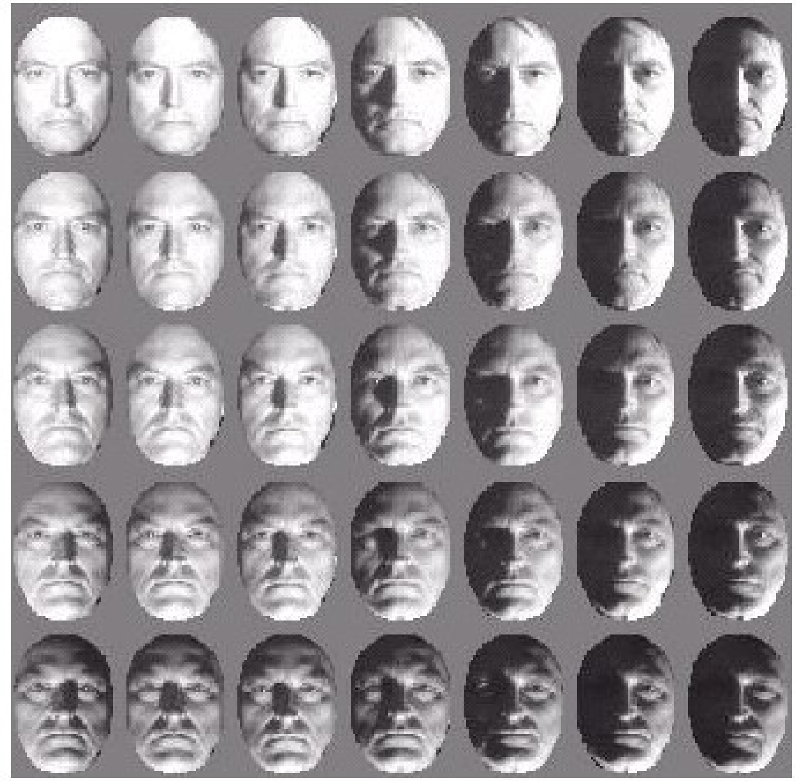
• Illumination Direction

• Suppose we have a set of images of a face with lighting from different directions, the principle components of these images are:

eigenvector







Factor Analysis

References

- **PCA:**
  - DIP book

1. M. Kirby and L. Sirovich, "Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces," IEEE Transactions on PAMI, Vol. 12, No. 1, pp. 103-108, January 1990.
2. **M.A. Turk and A. Pentland, "Eigenfaces for Recognition," Journal of Cognitive Neuroscience, Vol. 3, No. 1, pp. 71-86, 1991.**
3. H. Murase and S. Nayar, "Visual Learning and Recognition of 3-D Objects from Appearance," IJCV, 14, pp. 5-24, 1995.
4. B. Moghaddam and A. Pentland, "Probabilistic Visual Learning for Object Recognition," IEEE PAMI Vol. 19 No. 7, pp. 696-710, 1997.

- **KPCA**
- **PPCA**
- **GPCA**
- **SPCA**