Computer Vision for HCI

Principal Components Analysis (PCA)

1

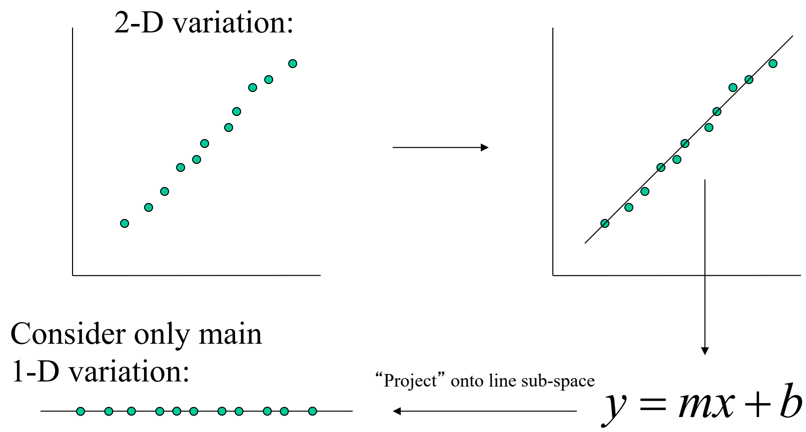
Feature Sub-Spaces

- Many times a high-dimensional feature vector is contained within a lower-dimensional “sub-space”
- When processing the feature data (for modeling and/or recognition), it is beneficial to deal with the lower-dimensional sub-space
- PCA offers **<u>linear</u>** approximation to the sub-space which can be reduced to only the *major* sub-space dimensions

2

2

Dimensionality Reduction



3

3

Linear Basis Set

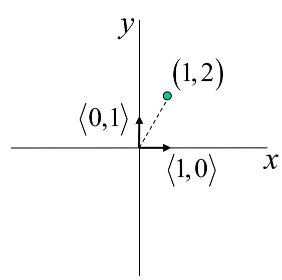
- 2-D basis set

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 1 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

x,y coordinates

x-axis

y-axis

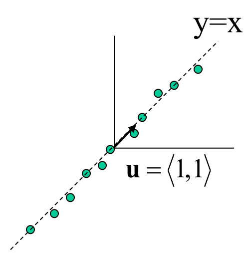


4

4

Linear Basis Set

- 1-D sub-space basis set



2-D space:

$$\mathbf{x_i} = a_i \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + b_i \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

1-D sub-space:

$$y_i = \gamma_i \cdot \mathbf{u} = \gamma_i \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

5

5

Linear Basis Set

- Generate smaller dimension basis for feature vectors

$$\mathbf{x_i} = \gamma_1 \cdot \mathbf{u_1} + \gamma_2 \cdot \mathbf{u_2} + \cdots + \gamma_m \cdot \mathbf{u_m}$$

$$\text{where} \quad \dim(\mathbf{x}) > m$$

6

6

Principal Components Analysis

7

PCA

- The main idea for PCA
  - Fit a multi-dimensional Gaussian around data
    - Use covariance of data to model Gaussian
  - Select only those dimensions capturing most of the variance in data
    - Reduce dimensionality
  - Sub-space is uncorrelated

8

8

PCA Primer: Equation for Eigenvalues

Square matrix

$$\mathbf{Ax} = \lambda \underline{\mathbf{x}}$$

Stretch/Scale **x** (Eigenvalue)

Same direction as **Ax**

Certain vector (Eigenvector)

$$\mathbf{Ax} - \lambda\mathbf{x} = \mathbf{0}$$

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$$

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

A is required to be square matrix for PCA

Want **x** to be non-zero, thus **A** is not invertible (determinant is zero). Each root leads to **x**.

9

9

PCA Primer: Example for Finding Eigenvalues

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

$$\mathbf{A} - \lambda\mathbf{I} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} 1-\lambda & 2 \\ 2 & 4-\lambda \end{bmatrix}$$

$$\det(\mathbf{A} - \lambda\mathbf{I}) = (1-\lambda)(4-\lambda) - (2)(2) = \lambda^2 - 5\lambda = 0$$

two roots: $\lambda_1 = 0, \quad \lambda_2 = 5$

10

10

PCA Primer: Finding the Eigenvectors

$$(\mathbf{A}-\lambda_i\mathbf{I})\mathbf{x}_i=\mathbf{0}$$

$$(\mathbf{A}-\lambda_1\mathbf{I})\mathbf{e}_1=\mathbf{0}$$

$$\left(\begin{bmatrix}1&2\\2&4\end{bmatrix}-\begin{bmatrix}0&0\\0&0\end{bmatrix}\right)\mathbf{e}_1=\mathbf{0}$$

$$\begin{bmatrix}1&2\\2&4\end{bmatrix}\begin{bmatrix}x_1\\y_1\end{bmatrix}=\mathbf{0}$$

$$\mathbf{e}_1=\begin{bmatrix}x_1\\y_1\end{bmatrix}=\begin{bmatrix}2\\-1\end{bmatrix}$$

$$(\mathbf{A}-\lambda_2\mathbf{I})\mathbf{e}_2=\mathbf{0}$$

$$\left(\begin{bmatrix}1&2\\2&4\end{bmatrix}-\begin{bmatrix}5&0\\0&5\end{bmatrix}\right)\mathbf{e}_2=\mathbf{0}$$

$$\begin{bmatrix}-4&2\\2&-1\end{bmatrix}\begin{bmatrix}x_2\\y_2\end{bmatrix}=\mathbf{0}$$

$$\mathbf{e}_2=\begin{bmatrix}x_2\\y_2\end{bmatrix}=\begin{bmatrix}1\\2\end{bmatrix}$$

11

11

PCA Primer: Test Our Results

$$\mathbf{A}\mathbf{e}_1\overset{?}{=}\lambda_1\mathbf{e}_1$$

$$\begin{bmatrix}1&2\\2&4\end{bmatrix}\begin{bmatrix}2\\-1\end{bmatrix}=\begin{bmatrix}(1\cdot2-2\cdot1)\\(2\cdot2-4\cdot1)\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}=0\begin{bmatrix}2\\-1\end{bmatrix}$$

$$\mathbf{A}\mathbf{e}_2\overset{?}{=}\lambda_2\mathbf{e}_2$$

$$\begin{bmatrix}1&2\\2&4\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}=\begin{bmatrix}(1\cdot1+2\cdot2)\\(2\cdot1+4\cdot2)\end{bmatrix}=\begin{bmatrix}5\\10\end{bmatrix}=5\begin{bmatrix}1\\2\end{bmatrix}$$

12

12

PCA Primer: Symmetric Matrices

- If **A** is <u>symmetric</u> ($\mathbf{A} = \mathbf{A}^T$)
  - Real-valued eigenvalues***
  - Eigenvectors can be chosen orthonormal***
  - Can be factorized as

    $$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$$

    where **Q** orthonormal ($\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$)

    and Λ is diagonal
  - Eigenvalues go into diagonal entries of Λ
    - **Convention: put <u>largest</u> eigenvalues <u>first</u> (descending values)**
    - **Alternative: put <u>smallest</u> eigenvalues <u>first</u> (ascending values)**
  - Corresponding orthogonal eigenvectors are normalized (to become orthonormal) and go into columns of **Q**

13

13

PCA Primer: Matlab

```
>> A= [ 1 2; 2 4]
A =
    1    2
    2    4

>> [Q, S] = eig(A)
Q =
  -0.8944    0.4472
   0.4472    0.8944
S =
    0    0
    0    5
```

14

14

PCA Primer: Matlab

```
>> A= [ 1 2; 2 4];

>> [Q, S] = eig(A);

>> Q'*Q
ans =
    1.0000         0
         0    1.0000

>> A - Q*S*Q'
ans =
     0     0
     0     0
```

15

15

Positive Definite Matrices

- Matrix A is <u>positive definite</u> for all non-zero **x** if
  - Symmetric <u>and</u> $\mathbf{x}^T \mathbf{A}\mathbf{x} > 0$

Positive <u>semi</u>-definite if
$\mathbf{x}^T \mathbf{A}\mathbf{x} \geq 0$

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} > 0$$

$$ax^2 + 2bxy + cy^2 > 0$$

- Recall equation for ellipse

$$ax^2 + by^2 = c$$

Rotated ellipse: $ax^2 + 2bxy + cy^2 = d$

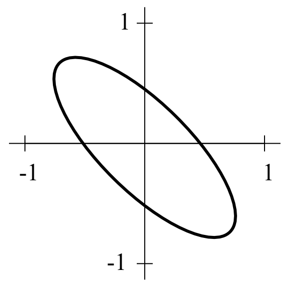
16

16

Ellipse Factorization

Consider the rotated ellipse:

$$5x^2 + 8xy + 5y^2 = 1$$



17

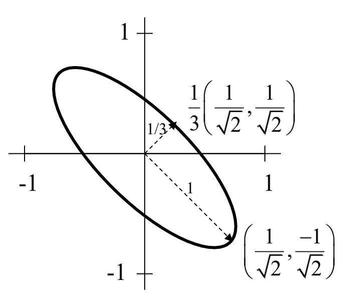
17

Ellipse Factorization

$$5x^2 + 8xy + 5y^2 = 1 \quad \rightarrow \quad a = 5,\ b = 4,\ c = 5$$

$$\mathbf{x}^T \mathbf{A}\mathbf{x} = 1$$

$$\mathbf{A} = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix} = \mathbf{Q\Lambda Q^T} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}_{\sqrt{2}} \begin{bmatrix} 9 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}_{\sqrt{2}}$$



Axes of ellipse point along eigenvectors

Half-lengths of axes are $1/\sqrt{\lambda_i}$ (NOTE: <u>bigger</u> eigenvalues give <u>shorter</u> axes!)

18

18

Eigenvector Projection

- The matrix $\mathbf{Q^T}$ acts as a rotation matrix

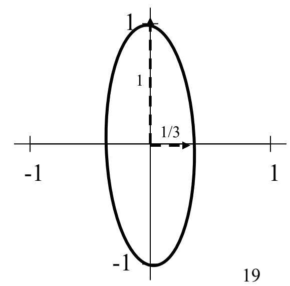
$$\mathbf{x^T A x} = \mathbf{x^T}\left(\mathbf{Q \Lambda Q^T}\right)\mathbf{x} = \left(\mathbf{x^T Q}\right)\mathbf{\Lambda}\left(\mathbf{Q^T x}\right) = \mathbf{X^T \Lambda X}$$

$$\mathbf{X} = \mathbf{Q^T x} = \begin{bmatrix} \hat{\mathbf{e}}_1^T \mathbf{x} \\ \hat{\mathbf{e}}_2^T \mathbf{x} \end{bmatrix}$$

Rotate previous axes:

$$\mathbf{X}_1 = \mathbf{Q^T x}_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}_{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}_{\sqrt{2}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathbf{X}_2 = \mathbf{Q^T x}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}_{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}_{\sqrt{2}} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



19

19

Gaussian Density

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\mathbf{K}|^{\frac{1}{2}}} \cdot e^{\left[-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T \mathbf{K}^{-1}(\mathbf{x}-\mathbf{m})\right]}$$

Remember this???

(squared) Mahalanobis distance: $(\mathbf{x}-\mathbf{m})^T \mathbf{K}^{-1}(\mathbf{x}-\mathbf{m}) = C$

Locus of all points at given distance C from mean (i.e., variance contour with C = # stdev^2)

20

20

Covariance

Covariance of vector **x**:

$$K = E\left[(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T\right] = \frac{1}{N}\mathbf{B}\mathbf{B}^\mathbf{T}$$

K is symmetric (and positive [semi-]definite)

Look at MATLAB's function **cov( )**

21

21

Plotting Gaussian Density Function Contours

Describes a (rotated) ellipse (at a *C* variance contour, or squared stdev contour) centered around mean:

$$\left(\mathbf{x}-\mathbf{m}\right)^T \mathbf{K}^{\mathbf{-1}}\left(\mathbf{x}-\mathbf{m}\right) = C$$

Thus we can factorize $\mathbf{K}^{\mathbf{-1}}$ into eigenvectors (axes) and eigenvalues to give direction and half-lengths of ellipse axes

$$\mathbf{K} = \mathbf{Q\Lambda Q^T}$$

$$\mathbf{K^{-1}} = \mathbf{Q\Lambda^{-1}Q^T}$$

22

22

Axes

- To compute the ellipse at Gaussian "variance" contour C (= # stdev^2):
  - **Method #1**: From matrix $\mathbf{K}^{-1}$ (<u>inverse</u> covariance)
    - Axes are columns in **Q**,
    - Half-lengths of axes are $\frac{\sqrt{C}}{\sqrt{\lambda_i}}$, with λ_i from $\mathbf{\Lambda}^{-1}$
  - **Method #2**: From matrix **K** (covariance)
    - Axes are columns in **Q**,
    - Half-lengths of axes are $\sqrt{C\lambda_i}$, with λ_i from $\mathbf{\Lambda}$

Putting it all together…

Ellipse formula:

$$x^T A x = 1$$

Ellipse defined by eigenvectors/values of A

Gaussian & Mahalanobis:

$$(x - m)^T K^{-1} (x - m) = C$$

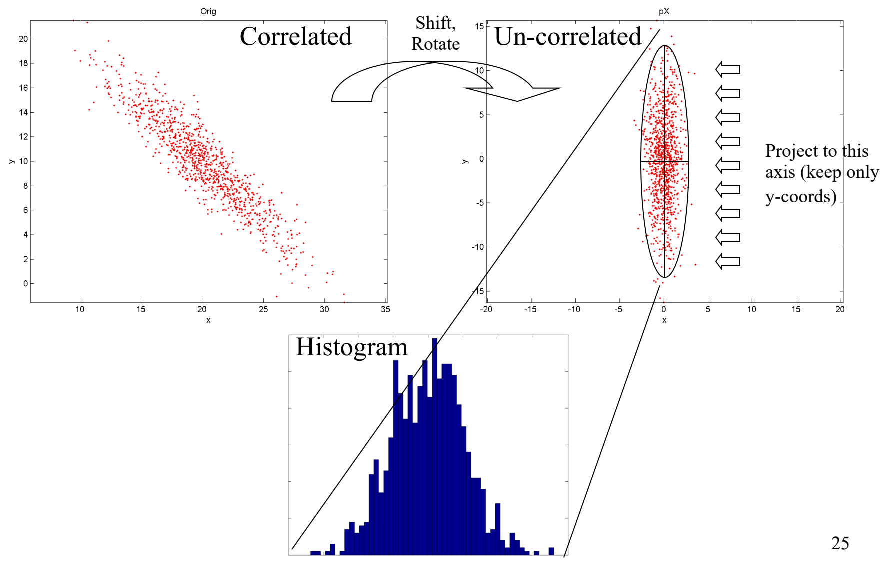
$$A = K^{-1}$$

Ellipse defined by eigenvectors/values of K^{-1}

<u>Larger</u> values of C give <u>bigger</u> variance contour ellipses

Or use K and just "flip" the eigenvalues (saves doing the inverse operation)

Gaussian Spaces



25

25

Face Recognition
(early days…)

26

Face Recognition Using EigenFaces (Turk & Pentland 91)

- Treat face recognition as 2-D view-based problem (images) rather than 3-D reconstruction and matching
- Project face images onto Gaussian feature space spanning significant variations among known face images
- Significant features in eigenspace for projection are called "EigenFaces"
  - Those eigenvectors capturing the majority of "variance" in data
    - Largest eigenvalues correspond to largest component variances
  - Project face image (vector) onto selected top eigenvectors

27

27

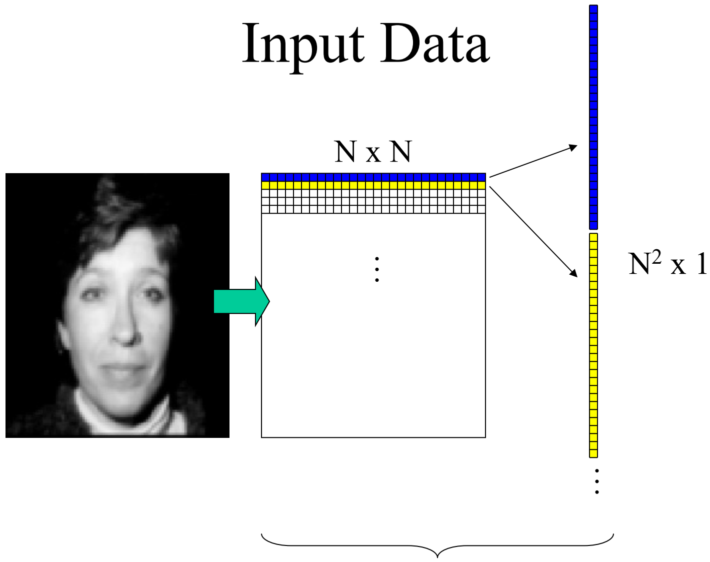
Using EigenFaces to Classify a Face Image

- Recognition achieved by comparing weights/coefficients of new face (after projection onto eigenspace) to other stored face weights/coefficients
  - Distance calculation more compact and efficient (uses small number of weights/coefficients)
- Calculate distance from face space or individual
  - Does it look like a face?
  - Does it look like "Joe"?

28

28

Input Data



Rasterize the image into vector

Input Data

- Compute mean face image $\boldsymbol{\Psi}$
  - From set of rasterized face images $\boldsymbol{\Gamma}_i$ (training images)
- Subtract mean from images
  - Remove the mean using $\boldsymbol{\Phi}_i = \boldsymbol{\Gamma}_i - \boldsymbol{\Psi}$
  - Form matrix of training faces

$$A = [\boldsymbol{\Phi}_1 \ \ \boldsymbol{\Phi}_2 \ \ \boldsymbol{\Phi}_3 \ \ \cdots \ \ \boldsymbol{\Phi}_M]$$

- Compute covariance matrix of A

Extra: Eigen "Trick"

- Compute eigenvectors and eigenvalues of A from its covariance matrix
  - Gaussian sub-space
- Number of face images (M) is much less than the dimension of the space (N^2)
- Thus only M-1 meaningful eigenvectors in matrix A
  - Remaining eigenvectors have eigenvalues = 0
- Can solve using a much smaller MxM matrix and convert back to N^2x1 dimensionality

31

31

Extra: Eigen "Trick"

- Consider the eigenvectors $\boldsymbol{v}_i$ of A^TA
  - Recall AA^T is how to compute covariance matrix K for $\boldsymbol{\Phi}_i$ data

$$(A^TA)\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$$

- Pre-multiplying both sized by A, we get

$$A(A^TA)\boldsymbol{v}_i = \lambda_i A\boldsymbol{v}_i$$

- Thus $\boldsymbol{u}_i=A\boldsymbol{v}_i$ are the eigenvectors of AA^T

$$\text{Covariance matrix} \longrightarrow (AA^T)[A\boldsymbol{v}_i] = \lambda_i [A\boldsymbol{v}_i]$$

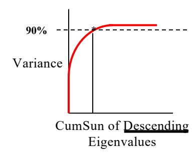
- Size comparison
  - Matrix A^TA is size MxM
  - Matrix AA^T (a covariance) is size N^2xN^2
- Hence, compute eigenvectors/eigenvalues from A^TA and use $\boldsymbol{u}_i=A\boldsymbol{v}_i$ to recover the desired dimensionality
  - Make sure to normalize the $\boldsymbol{u}_i$ to make them unit vectors!

32

32

Compute Eigenvectors

- Retain only top *m* eigenvectors ($\boldsymbol{u}_k$)
  - Determine from strength of eigenvalues
  - These eigenvectors are the "EigenFaces"
- Accumulate eigenvalues until reach desired percentage of total sum of eigenvalues
  - Pre-sort eigenvalues from largest to smallest
  - Considered as "% variance captured"
  - Typically use around 90%



33

33

Projection *into* Face Space

- Project each rasterized (mean-subtracted) face image into sub-space (*m* eigenvectors)
  - Project onto each eigenvector ("EigenFace")

$$\omega_k = \boldsymbol{u}_k^T \cdot \boldsymbol{\Phi}_i$$

- Keep projection coefficients as representation of rasterized image

$$\boldsymbol{\Phi}_i \longrightarrow \boldsymbol{\Omega}_i = [\omega_1\ \omega_2\ \omega_3\ \cdots\ \omega_m]^T$$

34

34

Reconstruction *from* Face Space

- Reconstruct face image from sub-space
  - Reconstruct from projection coefficients on each eigenvector

$$\boldsymbol{\Phi}_{recon} = \sum \omega_k \cdot \boldsymbol{u}_k$$

- Add back mean face

$$\boldsymbol{\Gamma}_{recon} = \boldsymbol{\Phi}_{recon} + \boldsymbol{\Psi}$$

35

35

Recognition Pipeline

- Get new face image
  - Rasterize
  - Mean-subtract using $\boldsymbol{\Psi}$
- Compute $\boldsymbol{\Omega}$
- Use Sum-of-Squared-Error (SSE) to other faces in the database
  - Find best match for database items $\boldsymbol{\Omega}_i$ (assumes that new face is in the database)

$$argmin_i \ \mathcal{E} = \|\boldsymbol{\Omega} - \boldsymbol{\Omega}_i\|^2$$

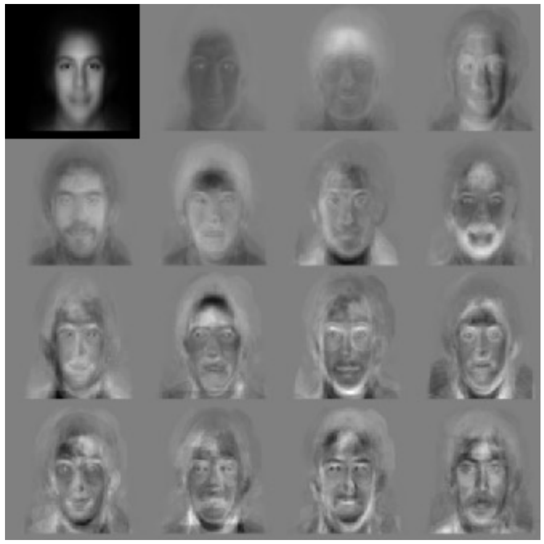
- Make sure "close enough" to face space and to person
  - Reconstruct new face image (from its projection coefficients and the eigenvectors), then threshold error distance from original face image
    - Face space only reconstructs face-like images
  - Make sure best match is within error tolerance to that person
- Other methods exist, including probabilistic methods

36

36

Example EigenFaces

Mean face

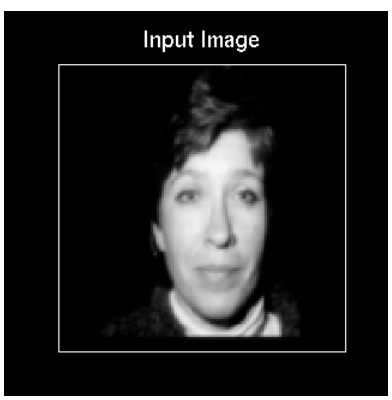


40 vectors were sufficient for 115 training face images

37

37

EigenFace Reconstruction



38

38

Face Recognition System

39

39

Summary

- Reduce high-dimensional input to lower-dimensional "sub-space"
  - Beneficial for recognition
- PCA offers <u>linear</u> approximation to the sub-space which can be reduced to only the *major* sub-space dimensions
  - Assumes Gaussian distribution
- Initial face recognition methods based on PCA
  - EigenFaces

40

40