Face Recognition: Eigenfaces and Fisherfaces

Face recognition: once you've detected and cropped a face, try to recognize it

Face recognition: overview

- Typical scenario: few examples per face, identify or verify test example
- What's hard: changes in expression, lighting, age, occlusion, **viewpoint**
- Basic approaches (all nearest neighbor)
  1. Project into a new subspace
  2. Measure face features

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”

What makes face recognition hard?

Expression

What makes face recognition hard?

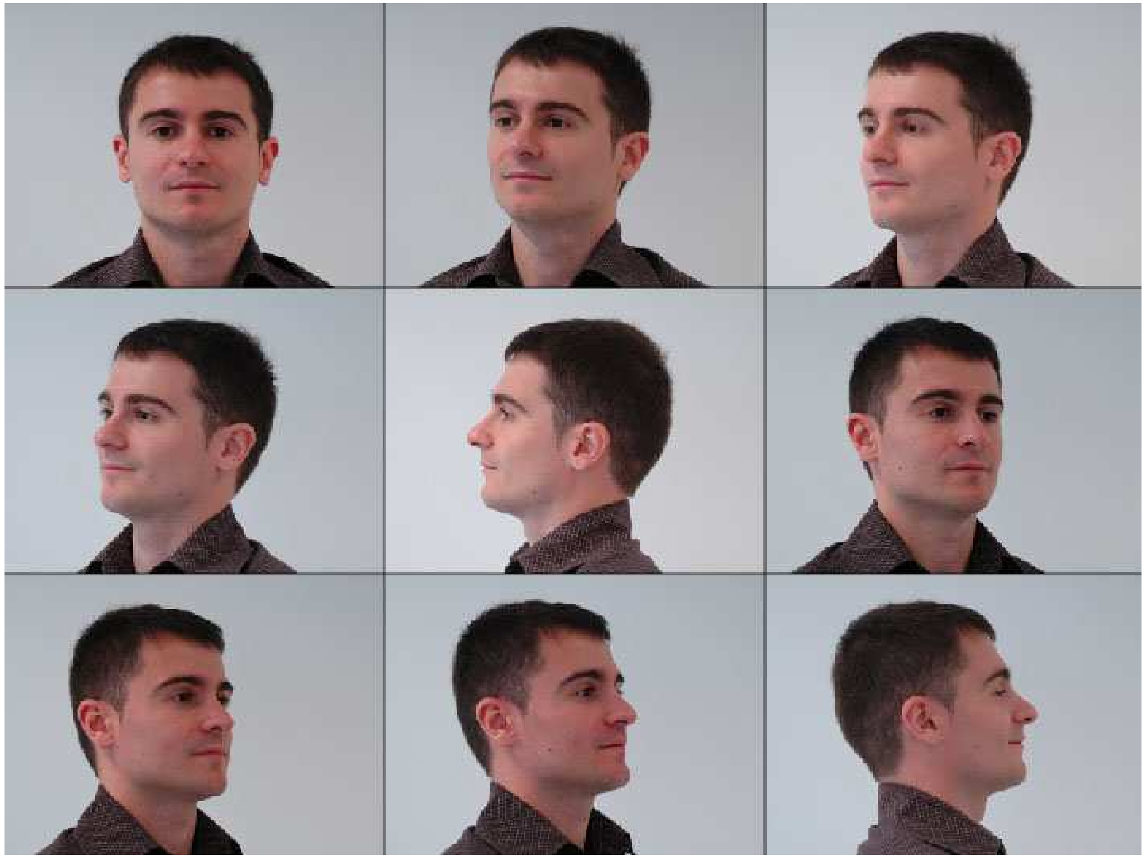
Lighting

What makes face recognition hard?

Occlusion

What makes face recognition hard?

Viewpoint



Simple idea for face recognition

1. Treat face image as a vector of intensities

$\rightarrow \mathbf{x}$

2. Recognize face by nearest neighbor in database

$\rightarrow \mathbf{y}_1 \ldots \mathbf{y}_n$

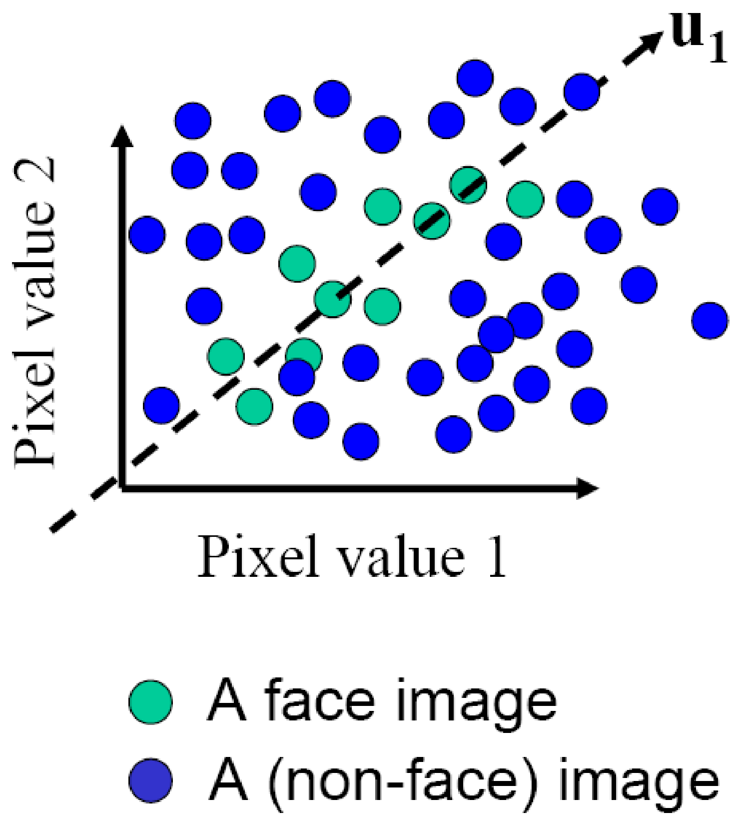
$$k = \underset{k}{\operatorname{argmin}} \; \left\| \mathbf{y}_k - \mathbf{x} \right\|$$

The space of all face images

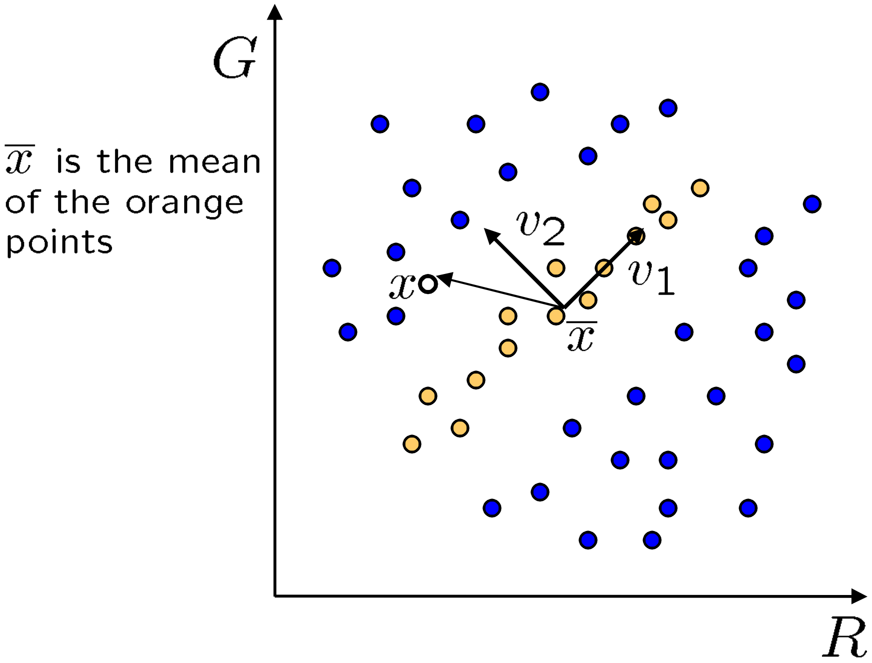
- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 100x100 image = 10,000 dimensions
  - Slow and lots of storage
- But very few 10,000-dimensional vectors are valid face images
- We want to effectively model the subspace of face images

The space of all face images

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



Linear subspaces



Consider the variation along direction **v** among all of the orange points:

$$var(\mathbf{v}) = \sum_{\text{orange point } \mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{T}} \cdot \mathbf{v}\|^2$$

What unit vector **v** minimizes *var*?

$$\mathbf{v}_2 = min_{\mathbf{v}} \ \{var(\mathbf{v})\}$$

What unit vector **v** maximizes *var*?

$$\mathbf{v}_1 = max_{\mathbf{v}} \ \{var(\mathbf{v})\}$$

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\| \\ &= \sum_{\mathbf{x}} \mathbf{v}^{\mathrm{T}}(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}}\mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}} \left[\sum_{\mathbf{x}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \right] \mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}} \mathbf{A} \mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \end{aligned}$$

Note: there's an error, the expression in the sum should be squared

Solution: $\mathbf{v_1}$ is eigenvector of **A** with *largest* eigenvalue
$\mathbf{v_2}$ is eigenvector of **A** with *smallest* eigenvalue

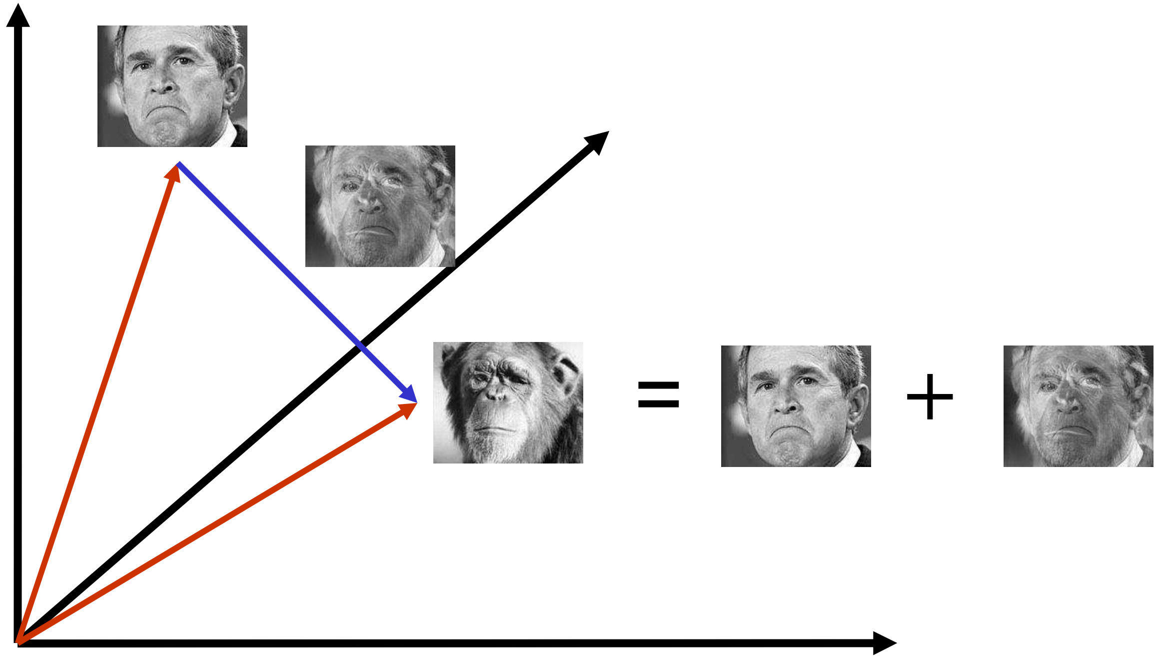
Principal component analysis (PCA)

- Suppose each data point is N-dimensional
  - Same procedure applies:

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x}-\overline{\mathbf{x}})^{\mathbf{T}} \cdot \mathbf{v}\| \\ &= \mathbf{v}^{\mathbf{T}}\mathbf{A}\mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}} (\mathbf{x}-\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^{\mathbf{T}} \end{aligned}$$

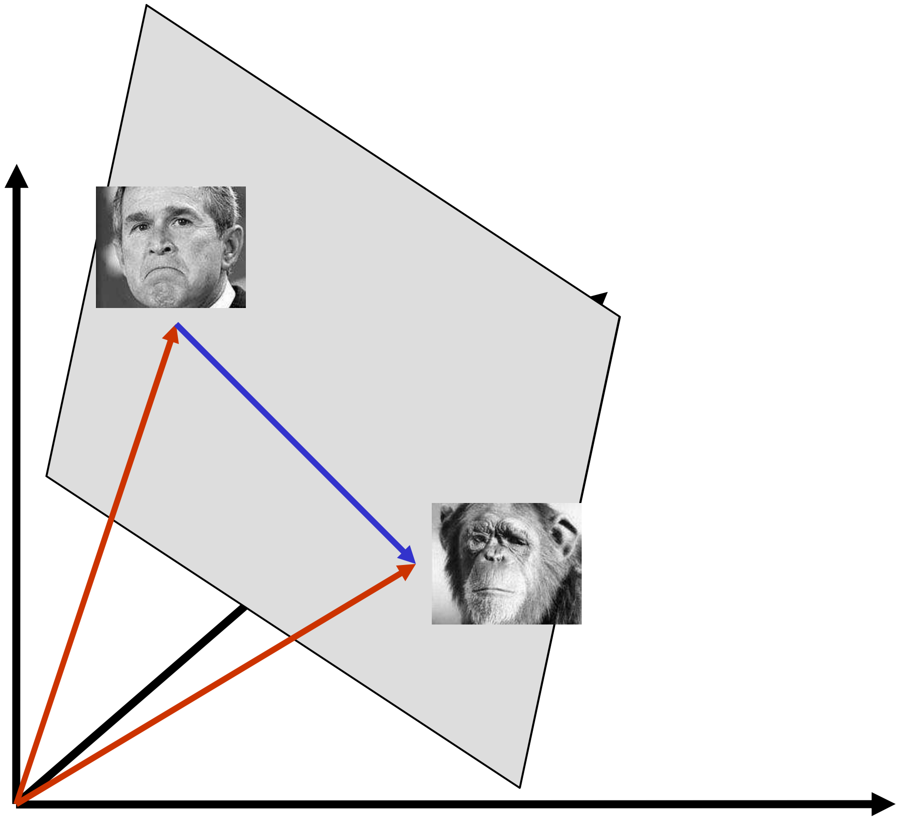
  - The eigenvectors of **A** define a new coordinate system
    - eigenvector with largest eigenvalue captures the most variation among training vectors **x**
    - eigenvector with smallest eigenvalue has least variation
  - We can compress the data by only using the top few eigenvectors
    - corresponds to choosing a “linear subspace”
      - represent points on a line, plane, or “hyper-plane”
    - these eigenvectors are known as the ***principal components***

The space of faces



- An image is a point in a high dimensional space
  - An N x M image is a point in R^{NM}
  - We can define vectors in this space as we did in the 2D case

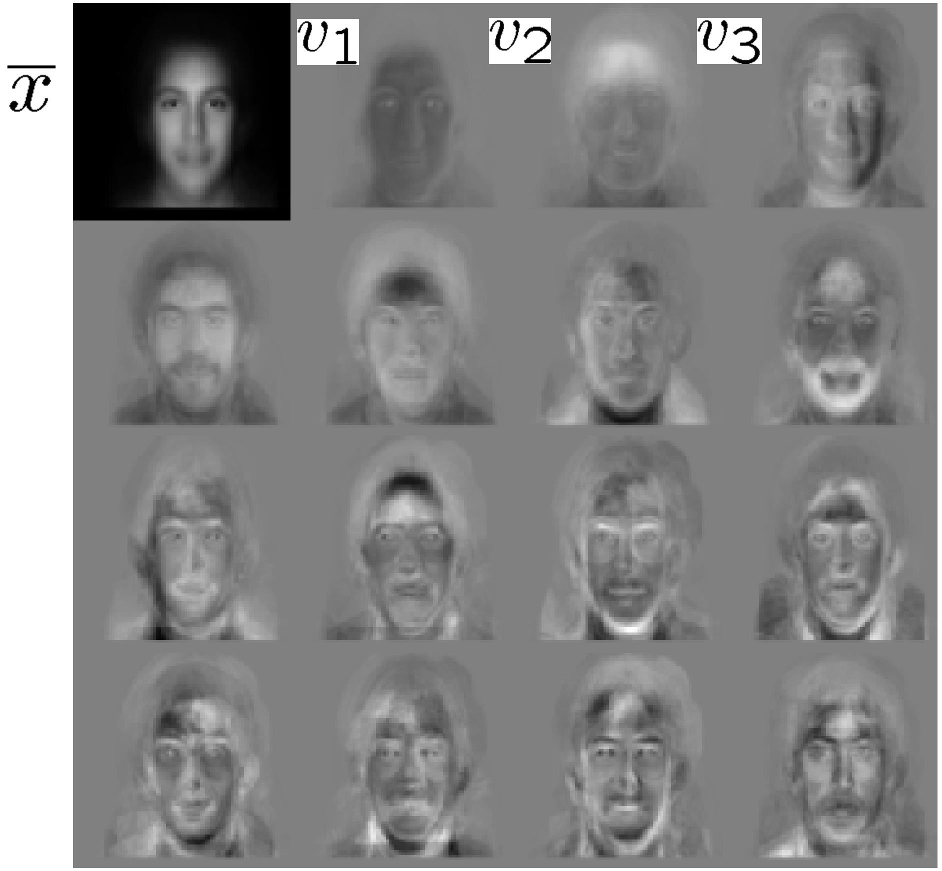
Dimensionality reduction



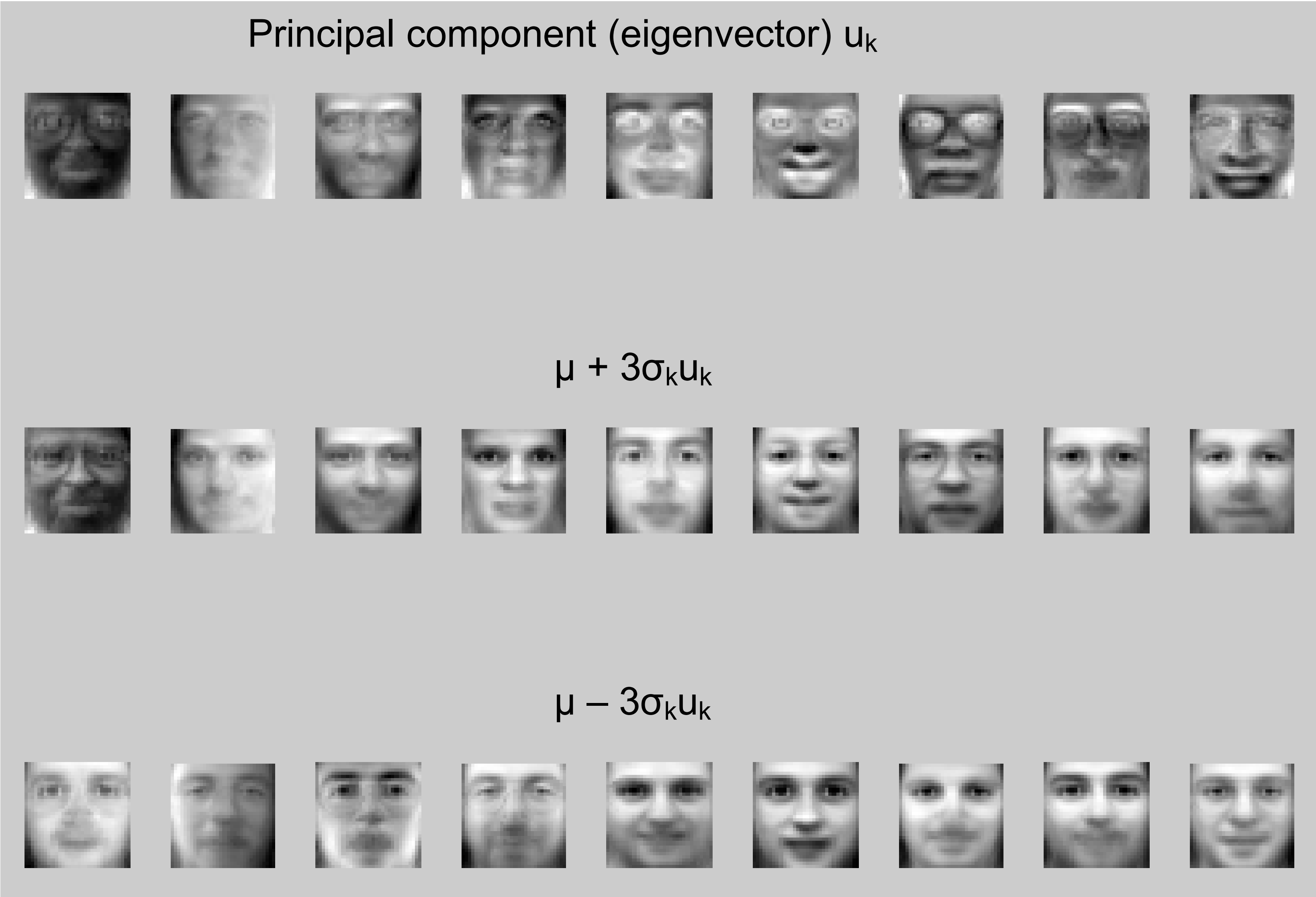
- The set of faces is a "subspace" of the set of images
  - Suppose it is K dimensional
  - We can find the best subspace using PCA
  - This is like fitting a "hyper-plane" to the set of faces
    - spanned by vectors $\mathbf{v_1}$, $\mathbf{v_2}$, ..., $\mathbf{v_K}$
    - any face $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

Eigenfaces

- PCA extracts the eigenvectors of **A**
  - Gives a set of vectors $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$, ...
  - Each one of these vectors is a direction in face space
    - what do these look like?



Visualization of eigenfaces



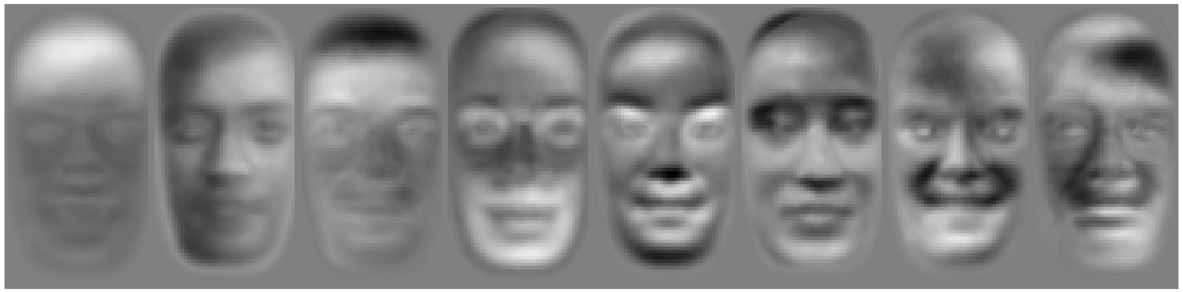
Projecting onto the eigenfaces

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

$$\mathbf{x} \rightarrow (\underbrace{(\mathbf{x} - \overline{\mathbf{x}}) \cdot \mathbf{v_1}}_{a_1}, \ \underbrace{(\mathbf{x} - \overline{\mathbf{x}}) \cdot \mathbf{v_2}}_{a_2}, \ldots, \ \underbrace{(\mathbf{x} - \overline{\mathbf{x}}) \cdot \mathbf{v_K}}_{a_K})$$

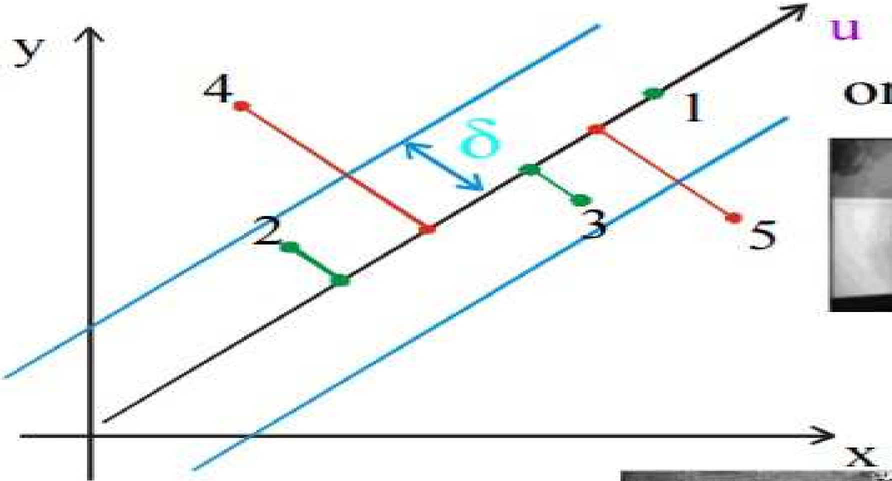
$$\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K}$$

$\mathbf{x}$

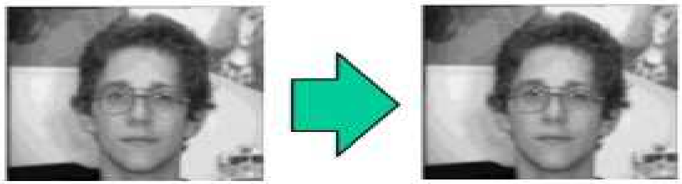


$a_1\mathbf{v_1}$ $a_2\mathbf{v_2}$ $a_3\mathbf{v_3}$ $a_4\mathbf{v_4}$ $a_5\mathbf{v_5}$ $a_6\mathbf{v_6}$ $a_7\mathbf{v_7}$ $a_8\mathbf{v_8}$

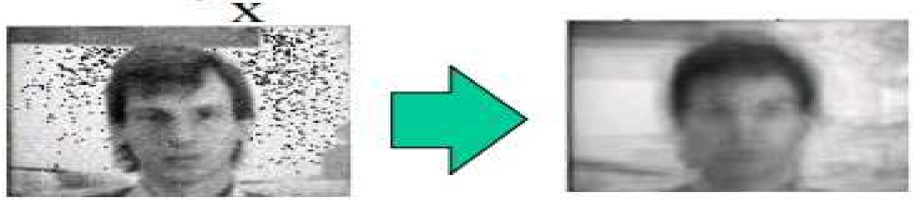
Determine If the Image is a Face at All



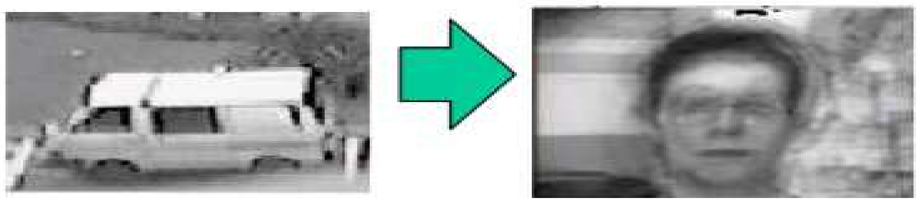
original projection



A face, used for training



A face, not used for training



Not a face, not used for training

1 *The best case: on the subspace*

2,3 *Close enough*

4,5 *Too far – not a face*

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) **x**, calculate K coefficients

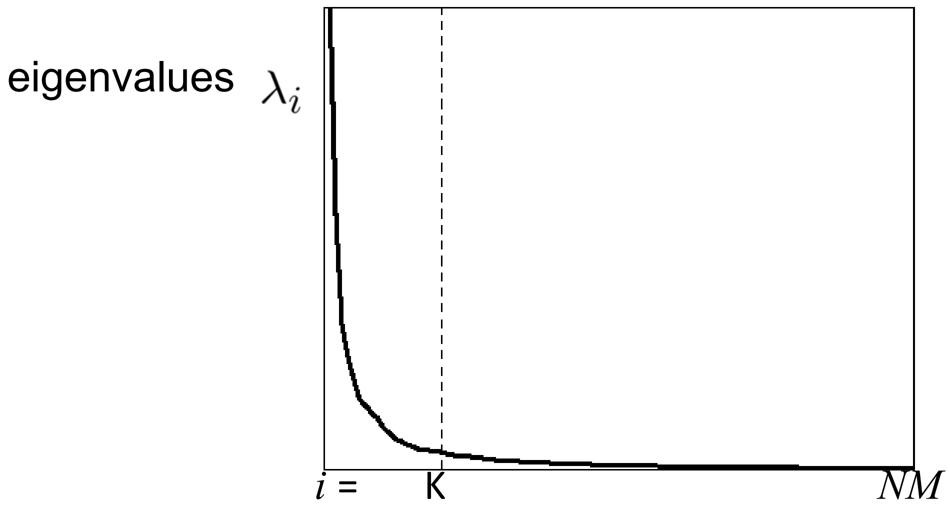
$$\mathbf{x} \rightarrow (a_1, a_2, \ldots, a_K)$$

  3. Detect if x is a face

$$\|\mathbf{x} - (\overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K})\| < \text{threshold}$$

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

Choosing the dimension K



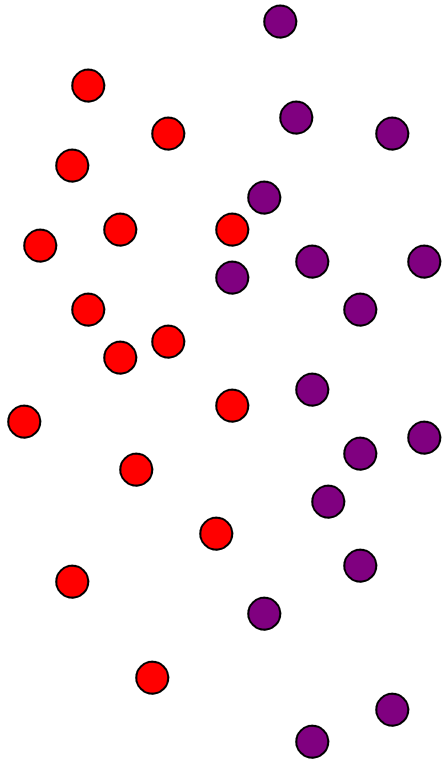
- How many eigenfaces to use?
- Look at the decay of the eigenvalues
  - the eigenvalue tells you the amount of variance "in the direction" of that eigenface
  - ignore eigenfaces with low variance

PCA

- General dimensionality reduction technique
- Preserves most of variance with a much more compact representation
  - Lower storage requirements (eigenvectors + a few numbers per face)
  - Faster matching

Limitations

- The direction of maximum variance is not always good for classification



A more discriminative subspace: FLD

- Fisher Linear Discriminants → "Fisher Faces"

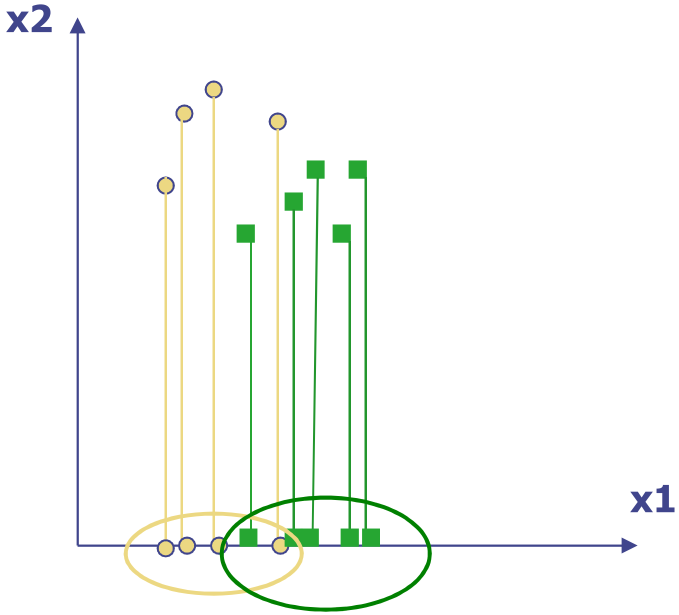
- PCA preserves maximum variance

- FLD preserves discrimination
  - Find projection that maximizes scatter between classes and minimizes scatter within classes

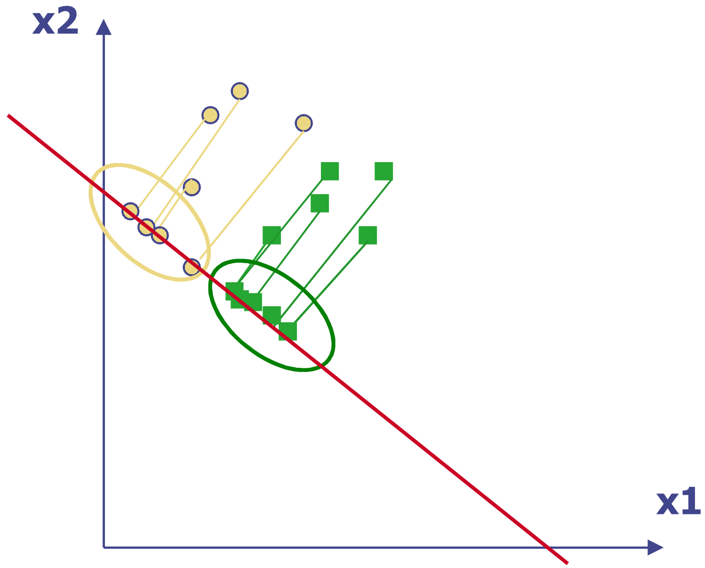
Reference: [Eigenfaces vs. Fisherfaces, Belheumer et al., PAMI 1997]

Illustration of the Projection

- Using two classes as example:

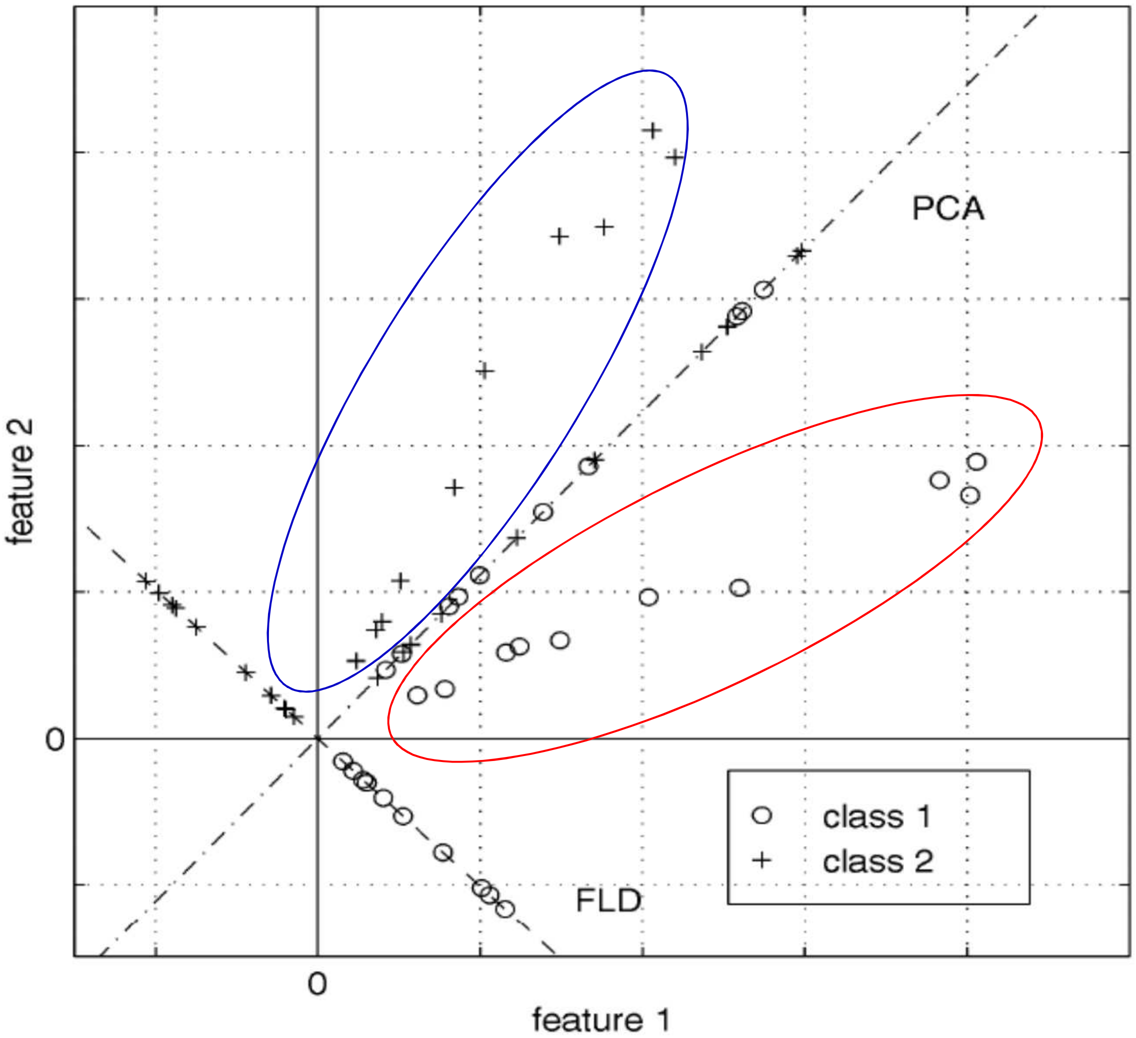


Poor Projection



Good

Comparing with PCA



Variables

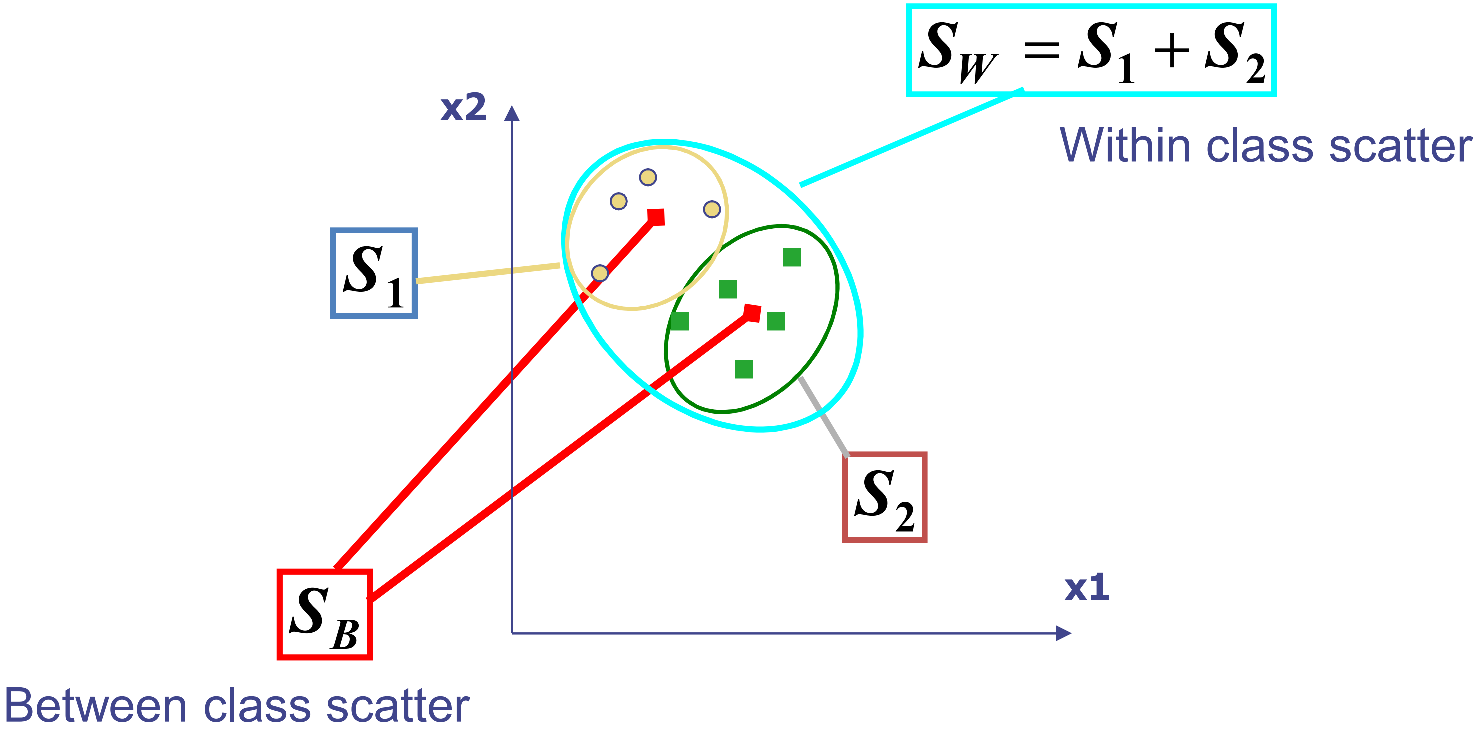
- N Sample images: $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$
- c classes: $\{\chi_1, \cdots, \chi_c\}$

- Average of each class: $\mu_i = \frac{1}{N_i} \sum_{x_k \in \chi_i} \boldsymbol{x}_k$
- Average of all data: $\mu = \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{x}_k$

Scatter Matrices

- Scatter of class i: $$S_i = \sum_{x_k \in \chi_i} (x_k - \mu_i)(x_k - \mu_i)^T$$
- Within class scatter: $$S_W = \sum_{i=1}^{c} S_i$$
- Between class scatter: $$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

Illustration



Mathematical Formulation

- After projection $y_k = W^T x_k$
  - Between class scatter $\tilde{S}_B = W^T S_B W$
  - Within class scatter $\tilde{S}_W = W^T S_W W$
- Objective

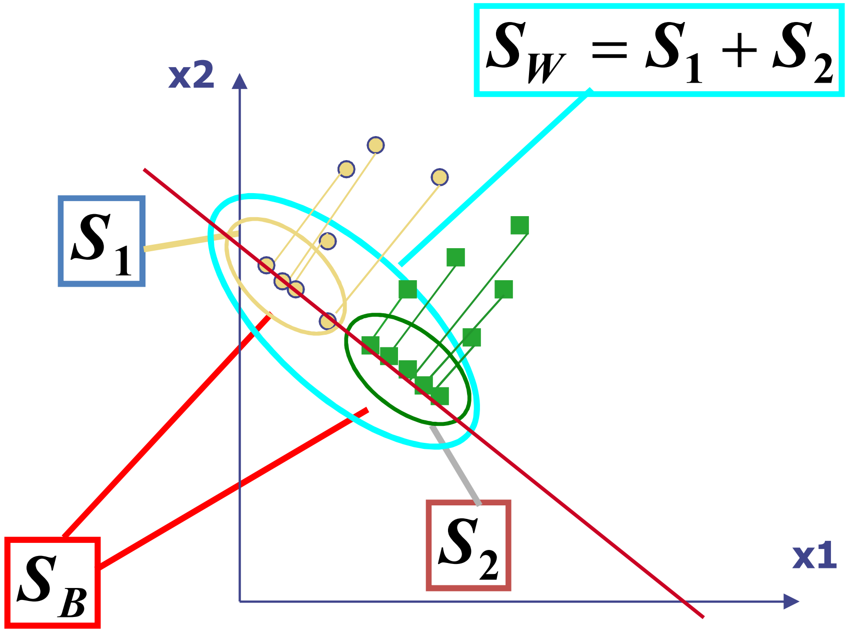
$$W_{opt} = \arg\max_{W} \frac{\left|\tilde{S}_B\right|}{\left|\tilde{S}_W\right|} = \arg\max_{W} \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}$$

- Solution: Generalized Eigenvectors

$$S_B w_i = \lambda_i S_W w_i \qquad i = 1, \ldots, m$$

- Rank of W_{opt} is limited
  - Rank(S_B) <= |C|-1
  - Rank(S_W) <= N-C

Illustration



Recognition with FLD

- Use PCA to reduce dimensions to N-C

$$W_{pca} = \mathrm{pca}(X)$$

- Compute within-class and between-class scatter matrices for PCA coefficients

$$S_i = \sum_{x_k \in \chi_i} (x_k - \mu_i)(x_k - \mu_i)^T \qquad S_W = \sum_{i=1}^{c} S_i \qquad S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

- Solve generalized eigenvector problem

$$W_{fld} = \arg\max_{\mathrm{W}} \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|} \qquad S_B w_i = \lambda_i S_W w_i \qquad i = 1, \ldots, m$$

- Project to FLD subspace (c-1 dimensions)

$$\hat{x} = W_{opt}{}^T x$$

- Classify by nearest neighbor

Note: x in step 2 refers to PCA coef; x in step 4 refers to original data

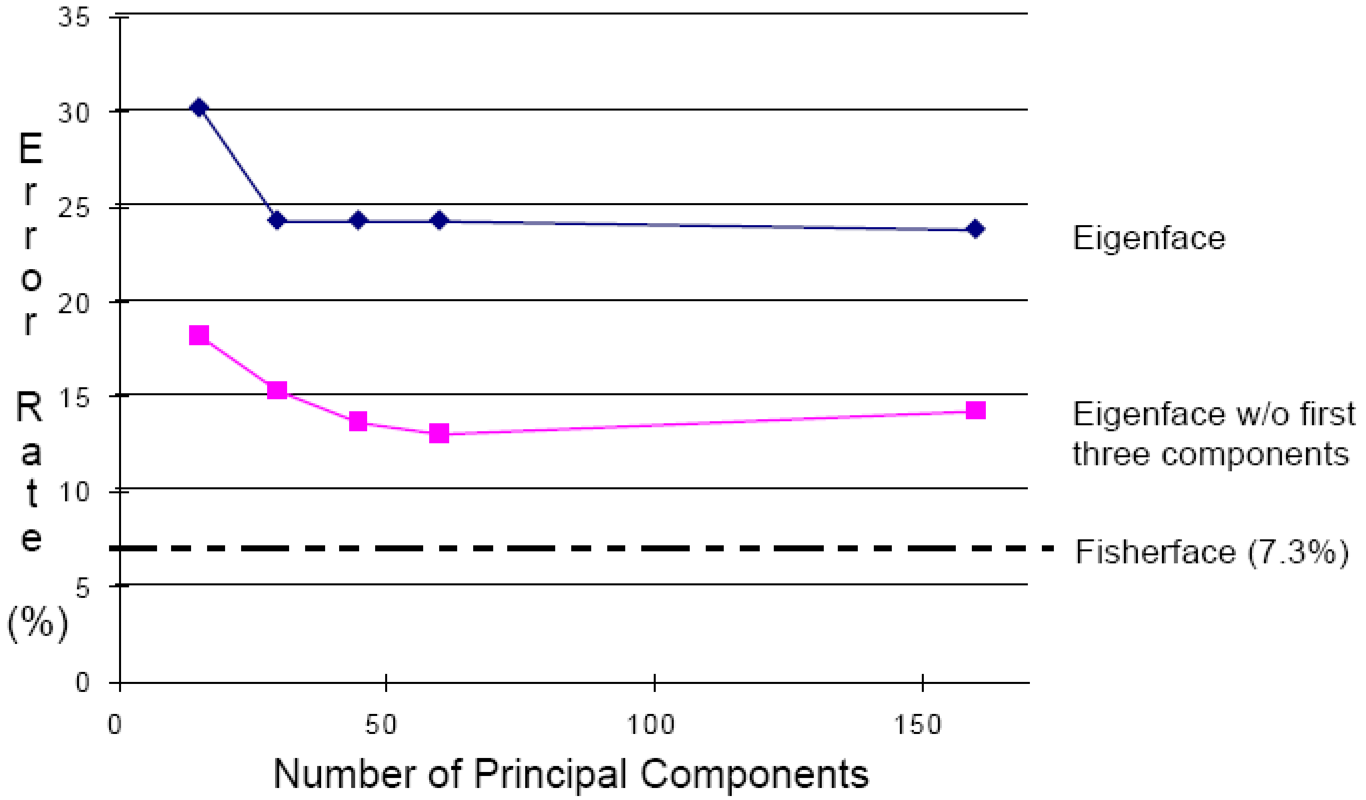
Results: Eigenface vs. Fisherface

- Input: 160 images of 16 people
- Train: 159 images
- Test: 1 image

- Variation in Facial Expression, Eyewear, and Lighting

Reference: [Eigenfaces vs. Fisherfaces, Belheumer et al., PAMI 1997]

Eigenfaces vs. Fisherfaces



Reference: [Eigenfaces vs. Fisherfaces, Belheumer et al., PAMI 1997]