Recognition:
Face Recognition

Linda Shapiro

EE/CSE 576

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

Detection

Recognition

"Sally"

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 (or kernel space) (e.g., "Eigenfaces"=PCA)
  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

$$\mathbf{x}$$

2. Recognize face by nearest neighbor in database

$$\mathbf{y}_1 \cdots \mathbf{y}_n$$

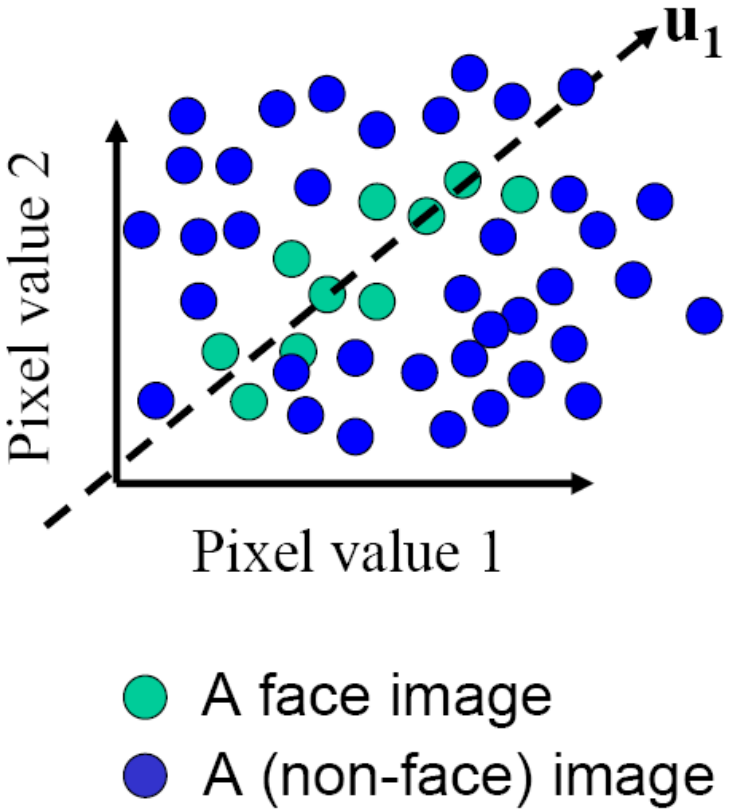
$$k = \operatorname*{argmin}_k \left\| \mathbf{y}_k - \mathbf{x} \right\|$$

The space of all face images

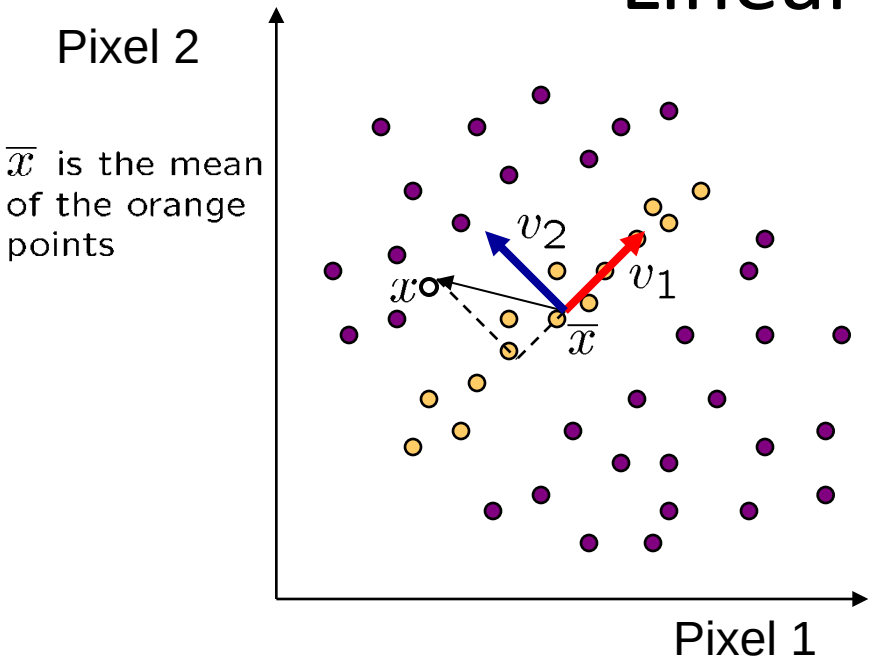
- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 100x100 image = 10,000 dimensions
  - Slow and lots of storage
- But very few 10,000-dimensional vectors are valid face images
- We want to effectively model the subspace of face images

The space of all face images

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



Linear subspaces



$\mathbf{v_1}$ is the major direction of the orange points and $\mathbf{v_2}$ is perpendicular to $\mathbf{v_1}$.

Convert $\mathbf{x}$ into $\mathbf{v_1}$, $\mathbf{v_2}$ coordinates

$$\mathbf{x} \rightarrow \left((\mathbf{x} - \overline{x}) \cdot \mathbf{v_1}, (\mathbf{x} - \overline{x}) \cdot \mathbf{v_2}\right)$$

What does the $\mathbf{v_2}$ coordinate measure?

- distance to line
- use it for classification—near 0 for orange pts

What does the $\mathbf{v_1}$ coordinate measure?

- position along line
- use it to specify which orange point it is

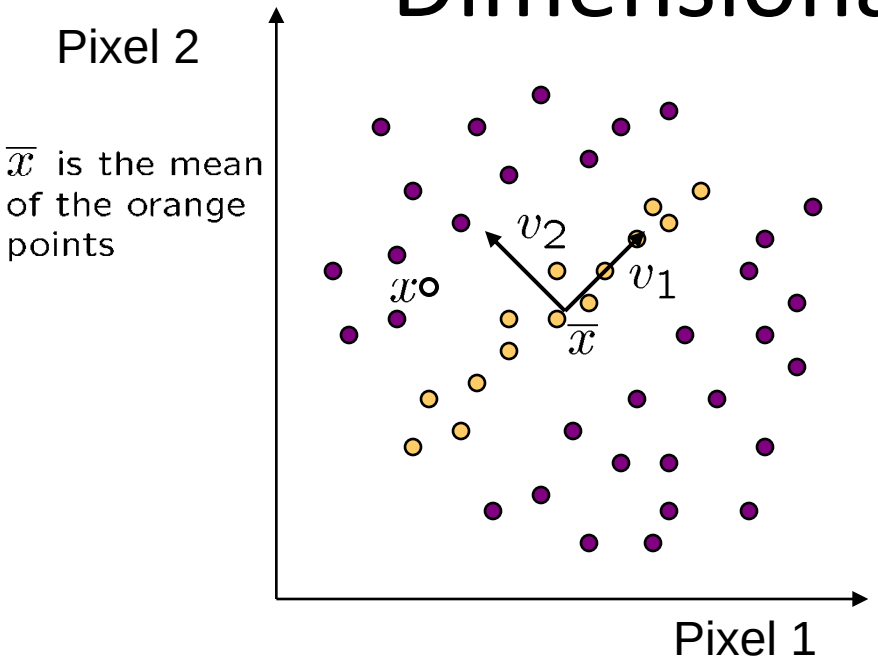
- Classification (to what class does x belong) can be expensive
  - Big search problem

Suppose the data points are arranged as above

- Idea—fit a line, classifier measures distance to line

Selected slides adapted from Steve Seitz, Linda Shapiro, Raj Rao

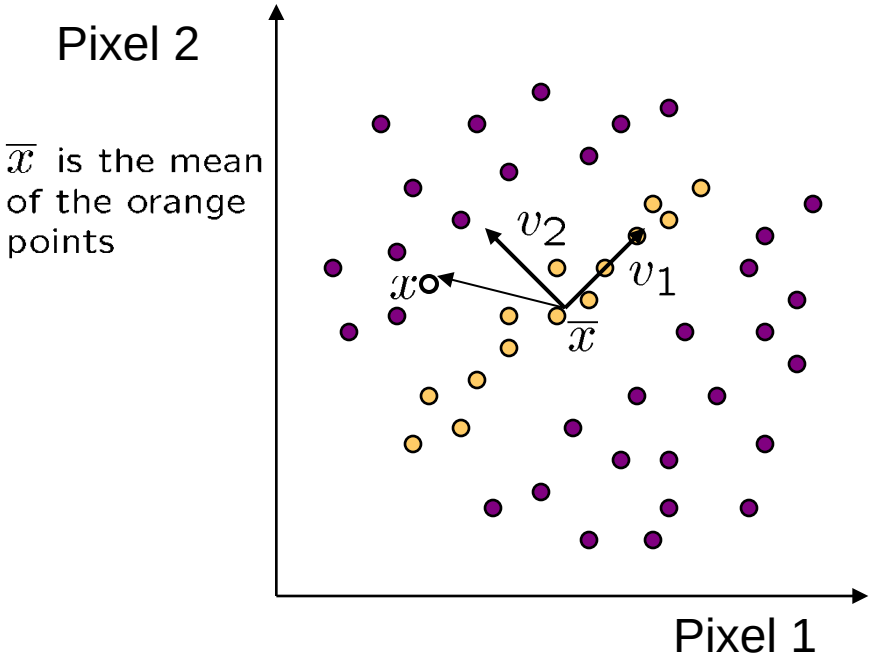
Dimensionality reduction



Dimensionality reduction

- We can represent the orange points with *only* their $\mathbf{v}_1$ coordinates
  - since $\mathbf{v}_2$ coordinates are all essentially 0
- This makes it much cheaper to store and compare points
- A bigger deal for higher dimensional problems (like images!)

Eigenvectors and Eigenvalues



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

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

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

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

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

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

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\|^2 \\ &= \sum_{\mathbf{x}} \mathbf{v}^{\mathrm{T}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}} \left[\sum_{\mathbf{x}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \right] \mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}} \mathbf{A} \mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \end{aligned}$$

A = covariance matrix of data points (if divided by no. of points)

Solution: $\mathbf{v}_1$ is eigenvector of **A** with *largest* eigenvalue
$\mathbf{v}_2$ is eigenvector of **A** with *smallest* eigenvalue

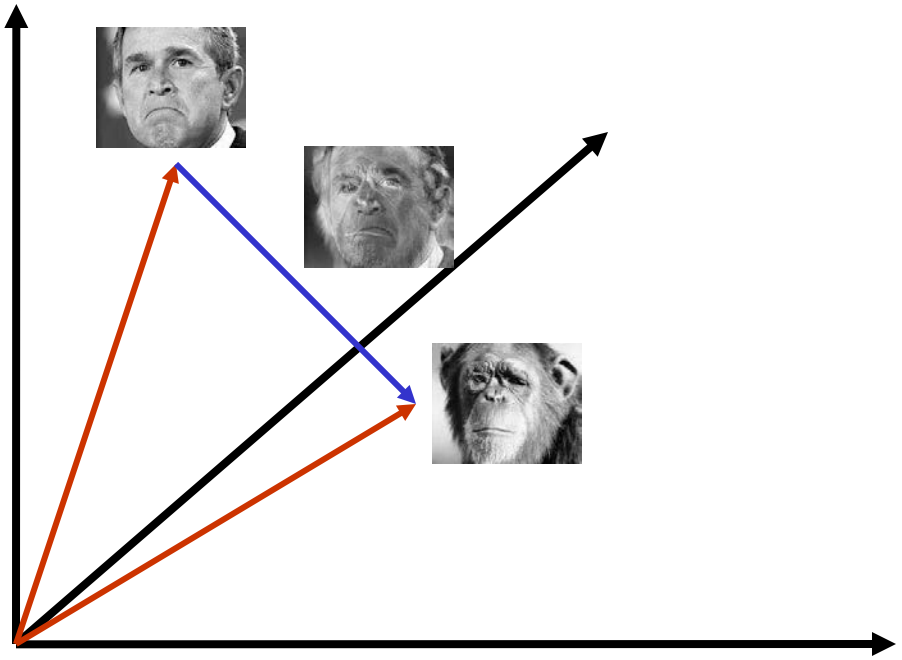
Principal component analysis (PCA)

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

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

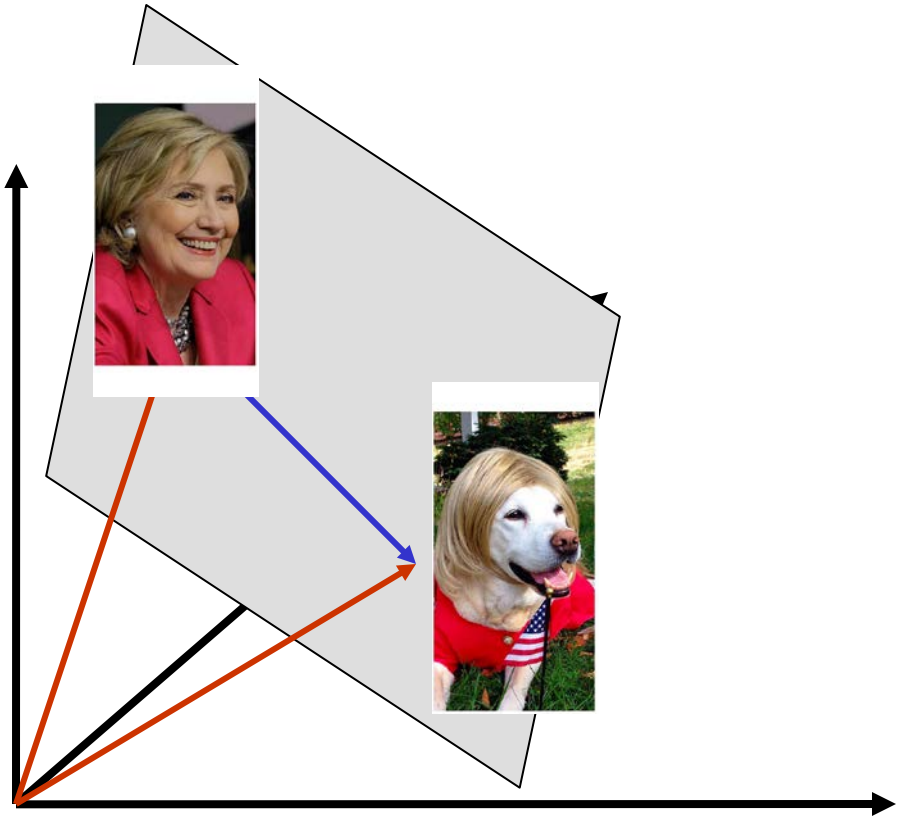
  - The eigenvectors of **A** define a new coordinate system
    - eigenvector with largest eigenvalue captures the most variation among training vectors **x**
    - eigenvector with smallest eigenvalue has least variation
  - We can compress the data by only using the top few eigenvectors
    - corresponds 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



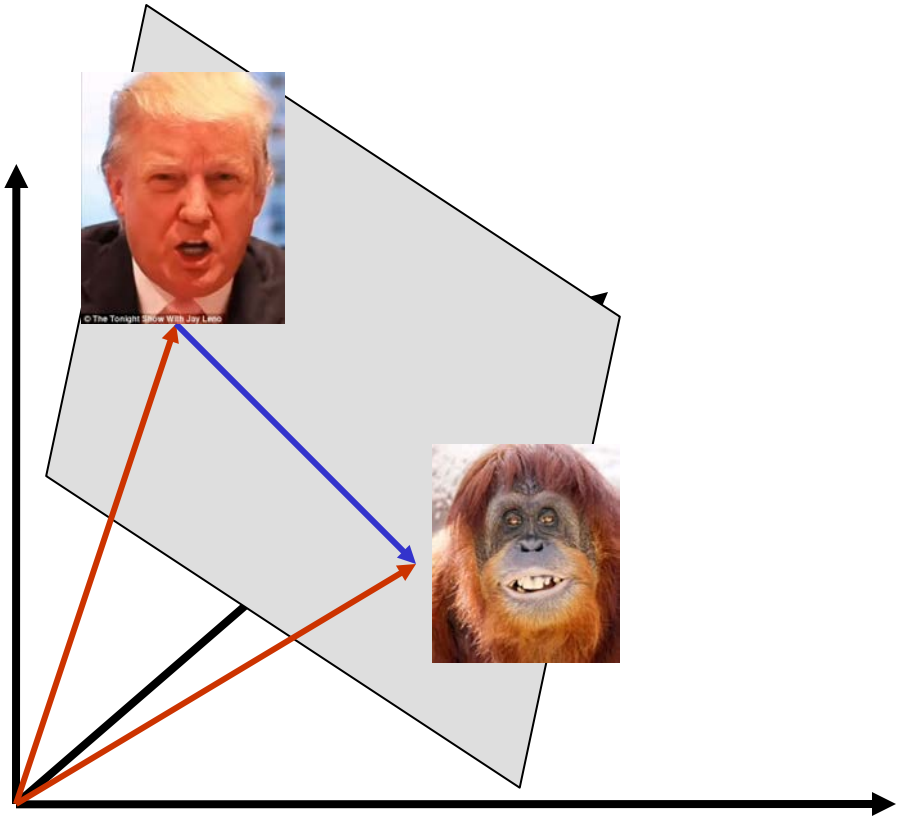
- An image is a point in a high dimensional space
  - An N x M image is a point in R^{NM}
  - We can define vectors in this space as we did in the 2D case

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 $\mathbf{x} \approx \bar{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

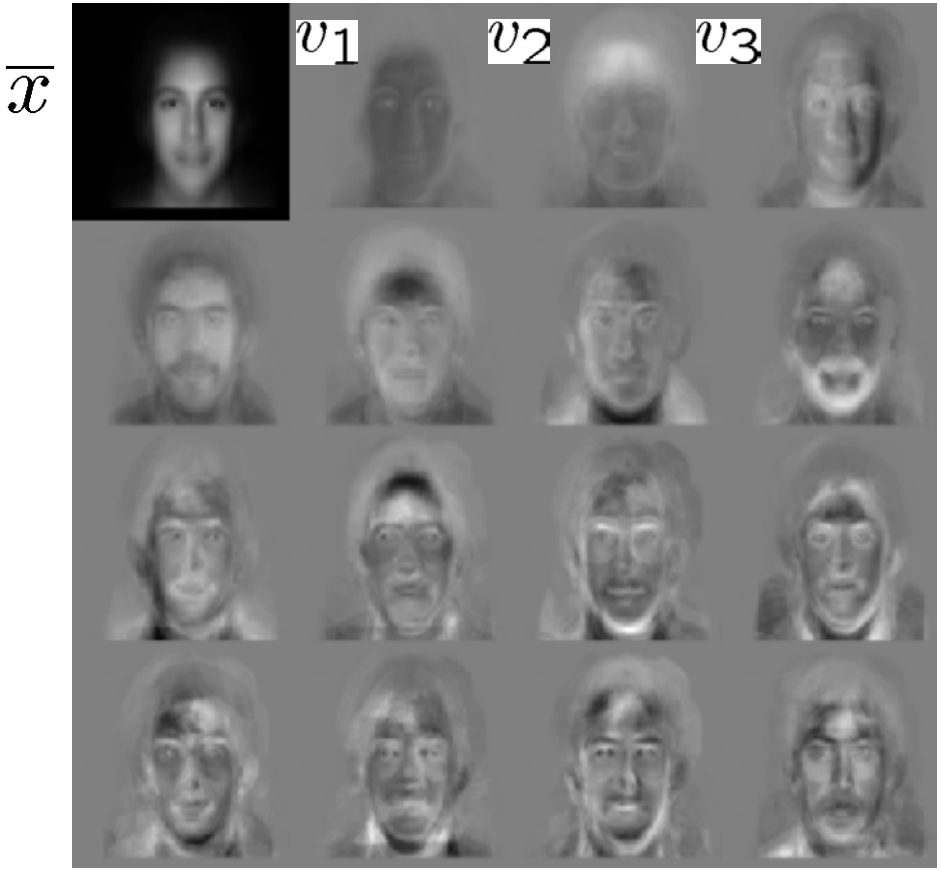
Dimensionality reduction



- 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 $\mathbf{x} \approx \bar{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

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

Principal component (eigenvector) u_k

$\mu + 3\sigma_k u_k$

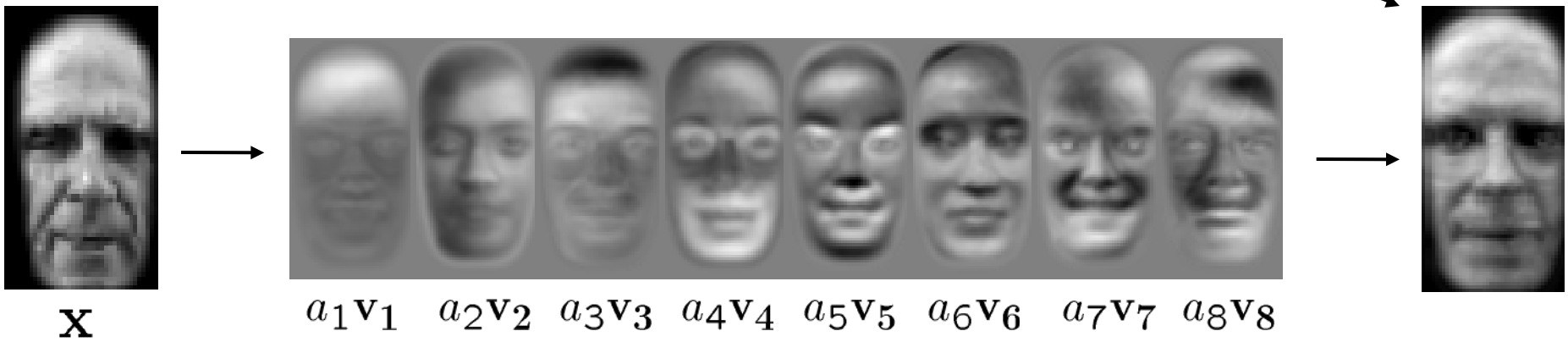
$\mu - 3\sigma_k u_k$

Projecting onto the eigenfaces

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

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

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



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

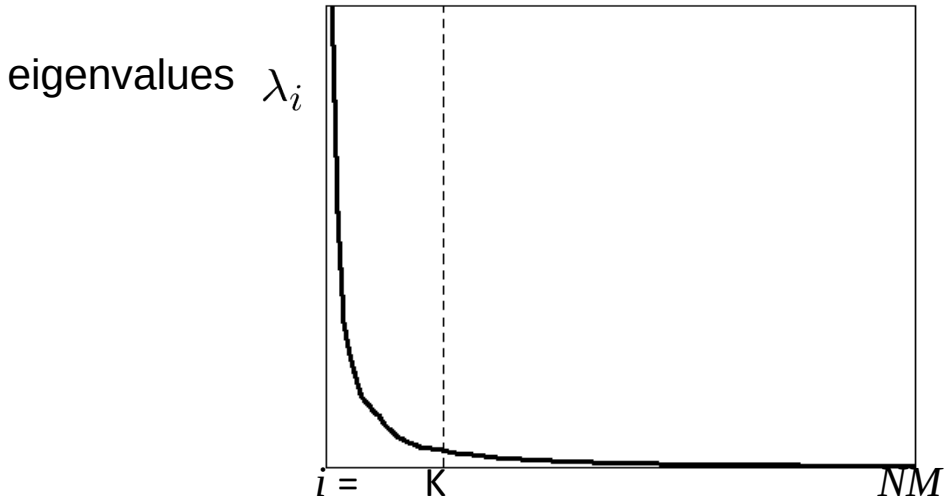
$$\mathbf{x} \rightarrow (a_1, a_2, \ldots, a_K)$$

  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

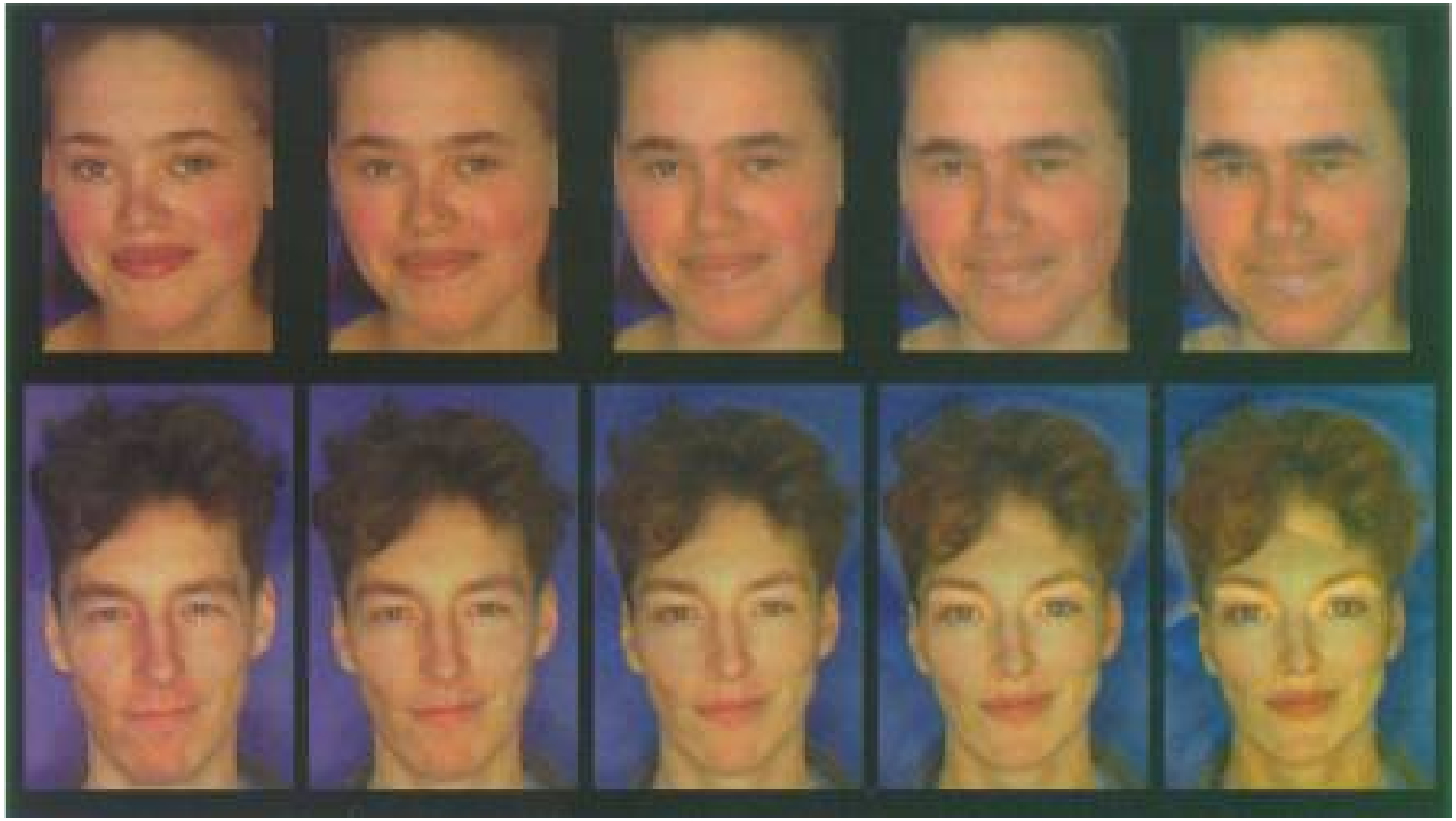


- How many eigenfaces to use?
- Look at the decay of the eigenvalues
  - the eigenvalue tells you the amount of variance "in the direction" of that eigenface
  - ignore eigenfaces with low variance

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
- What other applications?

Enhancing gender



more same **original** androgynous more opposite

D. Rowland and D. Perrett, "Manipulating Facial Appearance through Shape and Color," IEEE CG&A, September 1995

Slide credit: A. Efros

Changing age

•Face becomes "rounder" and "more textured" and "grayer"

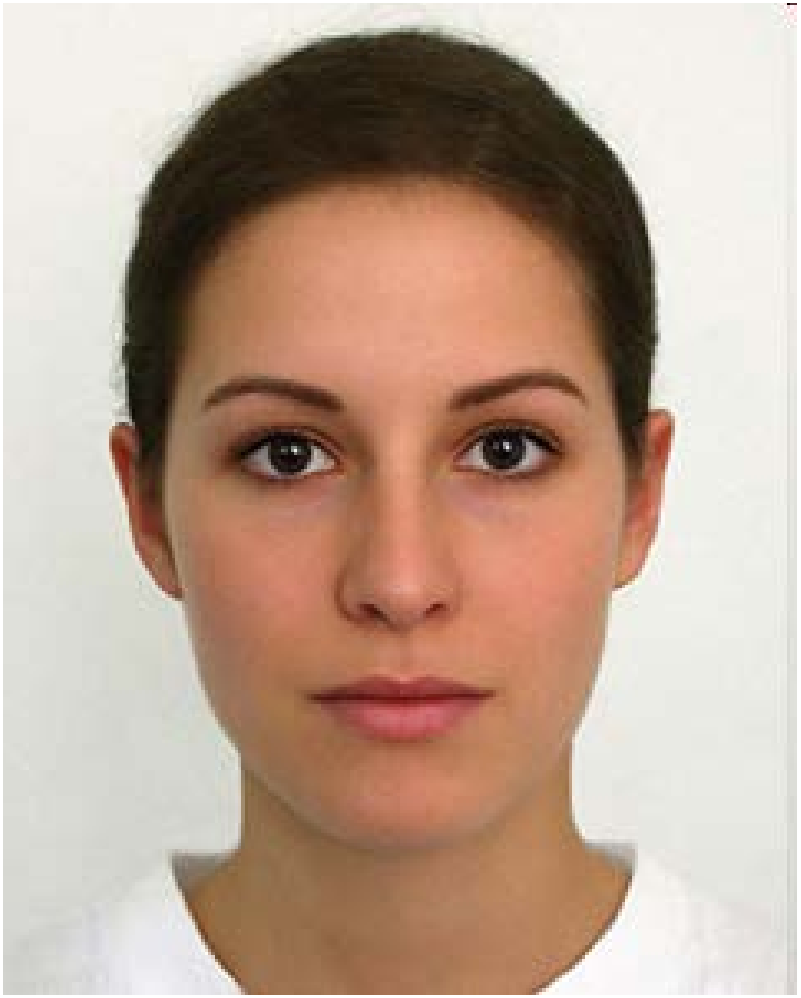
•original

• color

D. Rowland and D. Perrett, "Manipulating Facial Appearance through Shape and Color," IEEE CG&A, September 1995

Slide credit: A. Efros

Which face is more attractive?

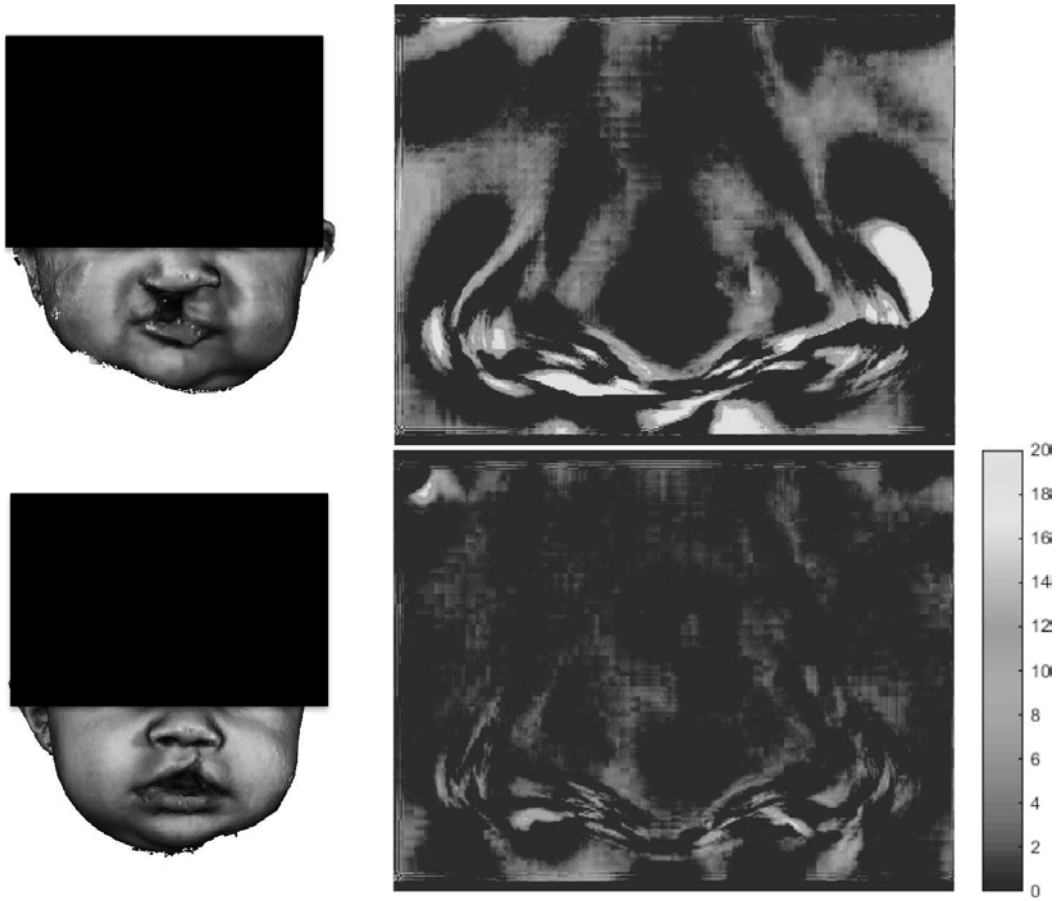


http://www.beautycheck.de

Use in Cleft Severity Analysis

- We have a large database of normal 3D faces.
- We construct their principal components.
- We can reconstruct any normal face accurately using these components.
- But when we reconstruct a cleft face from the normal components, there is a lot of error.
- This error can be used to measure the severity of the cleft.

Use of PCA Reconstruction Error to Judge Cleft Severity



Aligned head mesh

Error map from PCA reconstruction

Extension to 3D Objects

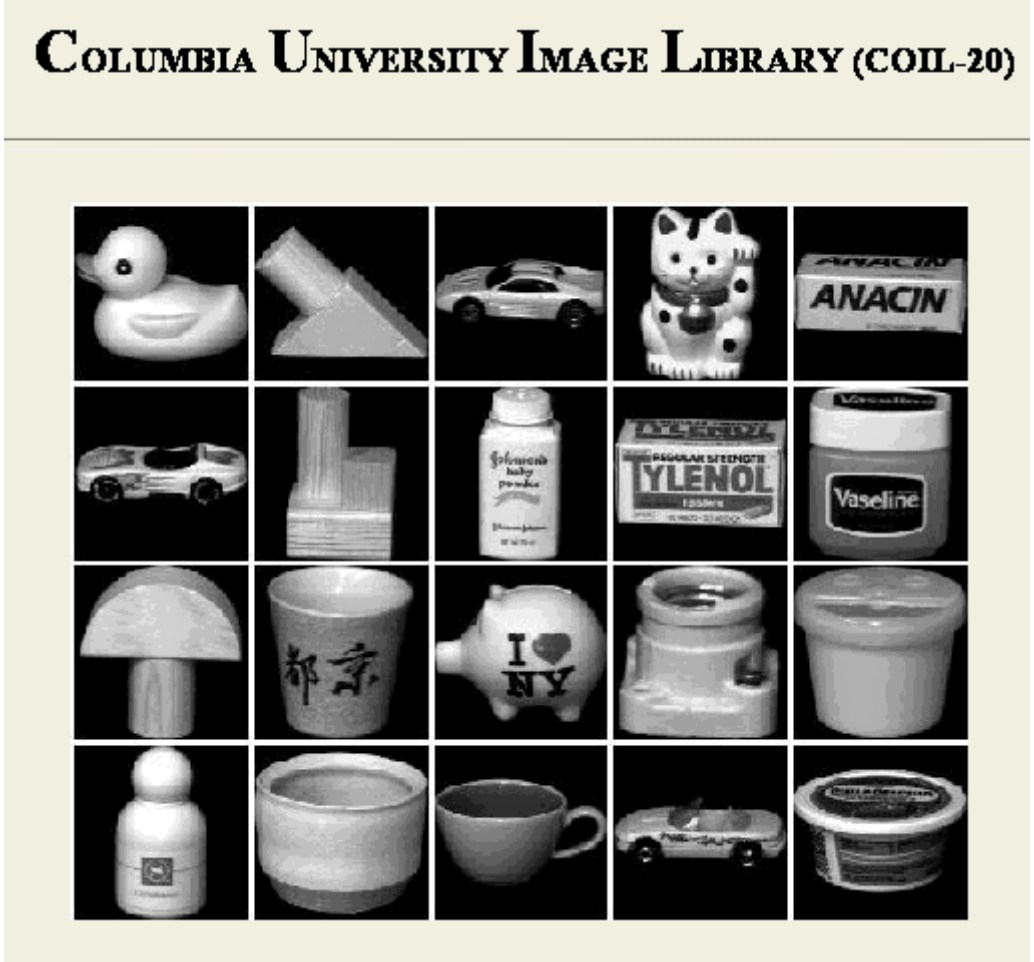
- Murase and Nayar (1994, 1995) extended this idea to 3D objects.

- The training set had multiple views of each object, on a dark background.

- The views included multiple (discrete) rotations of the object on a turntable and also multiple (discrete) illuminations.

- The system could be used first to identify the object and then to determine its (approximate) pose and illumination.

Sample Objects
Columbia Object Recognition Database



Significance of this work

- The extension to 3D objects was an important contribution.
- Instead of using brute force search, the authors observed that

  All the views of a single object, when transformed into the eigenvector space became points on a manifold in that space.
- Using this, they developed fast algorithms to find the closest object manifold to an unknown input image.
- Recognition with pose finding took less than a second.

Appearance-Based Recognition

- Training images must be representative of the instances of objects to be recognized.
- The object must be well-framed.
- Positions and sizes must be controlled.
- Dimensionality reduction is needed.
- It is not powerful enough to handle general scenes without prior segmentation into relevant objects.

* The newer systems that use “parts” from interest operators are an answer to these restrictions.