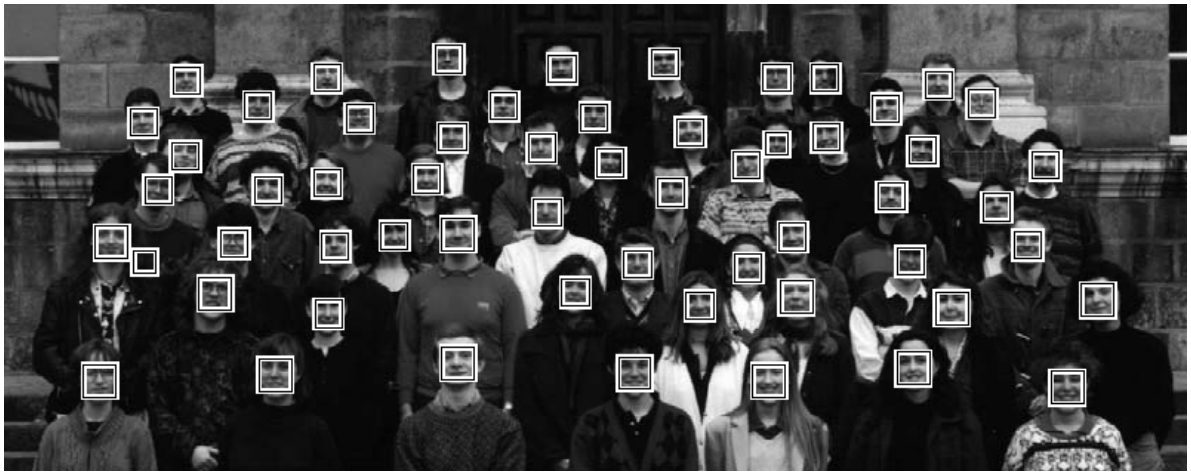
# Face Detection and Recognition
## Readings: Ch 8: Sec 4.4, Ch 14: Sec 4.4

- Bakic flesh finder using color (HW4)
- Fleck, Forsyth, Bregler flesh finder using color/texture and body parts (the naked people paper)
- Rowley & Kanade face detector using neural nets to recognize patterns in grayscale
- Eigenfaces: the first “appearance-based” recognition

# Object Detection

- Example: Face Detection



(Rowley, Baluja & Kanade, 1998)

- Example: Skin Detection

(Jones & Rehg, 1999)

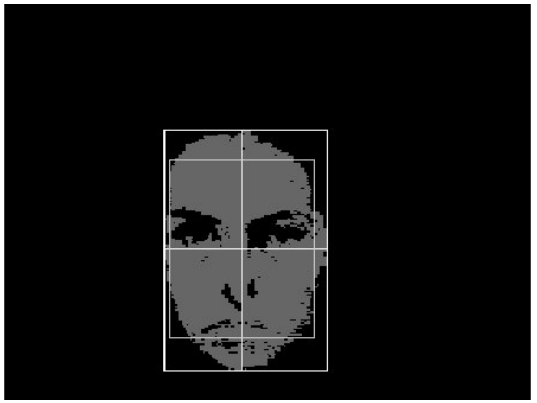
# Review:  Bakic Flesh Finder

- Convert pixels to normalized (r,g) space
- Train a binary classifier to recognize pixels in this space as skin or not skin by giving it lots of examples of both classes.
- On test images, have the classifier label the skin pixels.
- Find large connected components.

# Finding a face in a video frame

input video frame

pixels classified in normalized r-g space



largest connected component with aspect similar to a face

(all work contributed by Vera Bakic)

# Fleck and Forsyth's Flesh Detector

- Convert RGB to HSI
- Use the intensity component to compute a texture map
  texture = med2 ( | I - med1(I) | )
- If a pixel falls into either of the following ranges, it's a potential skin pixel

  texture < 5, 110 < hue < 150, 20 < saturation < 60
  texture < 5, 130 < hue < 170, 30 < saturation < 130

median filters of radii 4 and 6

* Margaret Fleck, David Forsyth, and Chris Bregler (1996) "Finding Naked People," 1996 **European Conference on Computer Vision** , Volume II, pp. 592-602.

# Algorithm

1. **Skin Filter**: The algorithm first locates images containing large areas whose color and texture is appropriate for skin.

2. **Grouper**: Within these areas, the algorithm finds elongated regions and groups them into possible human limbs and connected groups of limbs, using specialized groupers which incorporate substantial amounts of information about object structure.

3. Images containing sufficiently large skin-colored groups of possible limbs are reported as potentially containing naked people.

This algorithm was tested on a database of 4854 images: 565 images of naked people and 4289 control images from a variety of sources. The skin filter identified 448 test images and 485 control images as containing substantial areas of skin. Of these, the grouper identified 241 test images and 182 control images as containing people-like shapes.

# Grouping

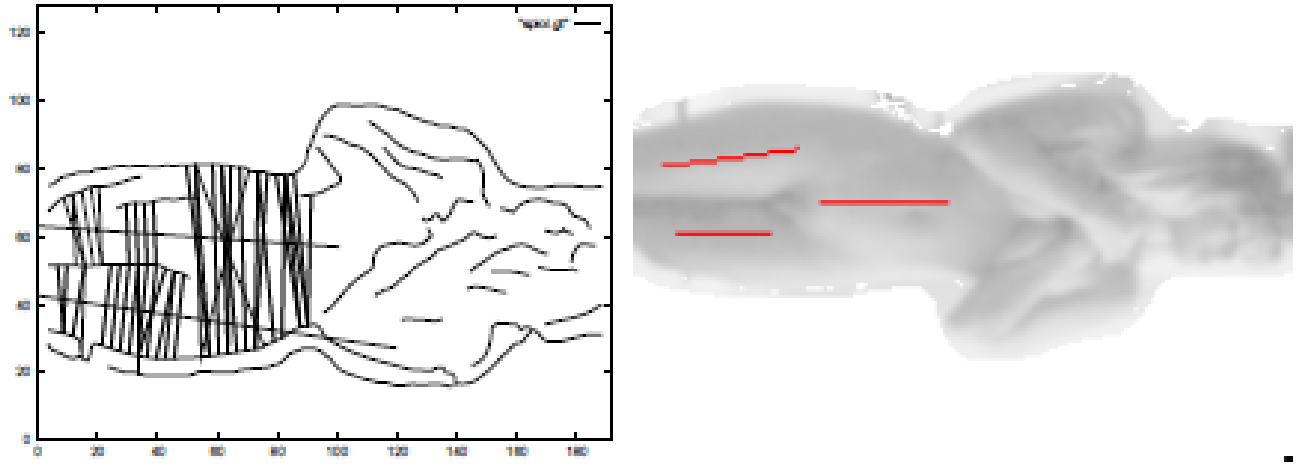
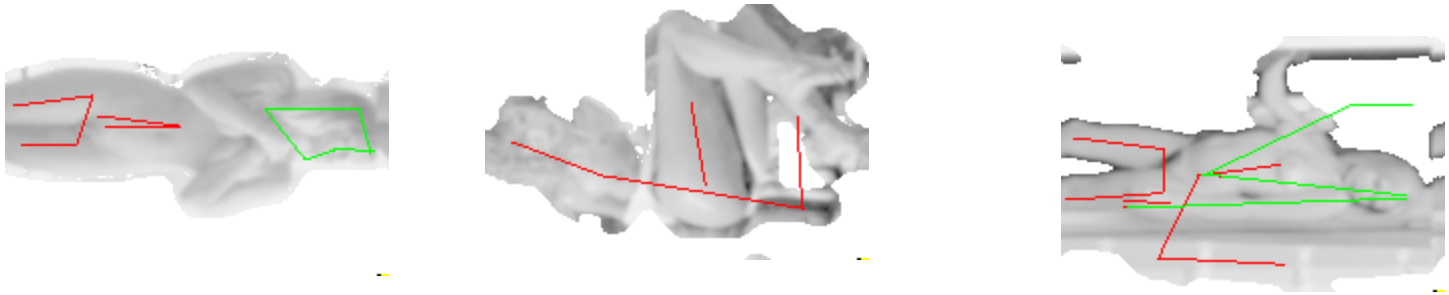


**Fig. 2.** Grouping a spine and two thighs: *Top left* the segment axes that will be grouped into a spine-thigh group, overlaid on the edges, showing the upper bounds on segment length and the their associated symmetries; *Top right* the spine and thigh group assembled from these segments, overlaid on the image.

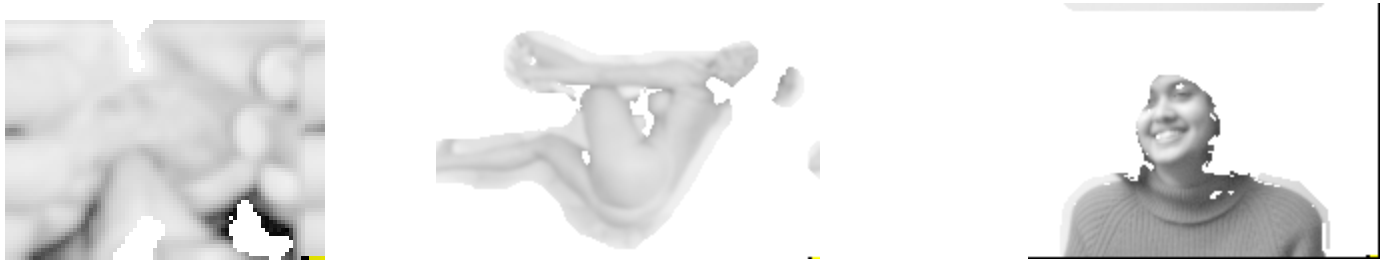
# Results

| | eliminated by skin filter | eliminated by geometrical analysis | marked as containing naked people |
|---|---|---|---|
| test images | 13.8% (19) | 34.1% (47) | 52.2% (72) |
| control images | 92.6% (1297) | 4.0% (56) | 3.4% (48) |

**Table 1.** Overall classification performance of the system.



Some True Positives



False Negatives

True Negative

# Object Detection: Rowley's Face Finder

1. convert to gray scale
2. normalize for lighting*
3. histogram equalization
4. apply neural net(s) trained on 16K images

What data is fed to the classifier?

20 x 20 windows in a pyramid structure

* Like first step in Laws algorithm, p. 220

# Preprocessing

**Oval mask for ignoring background pixels:**

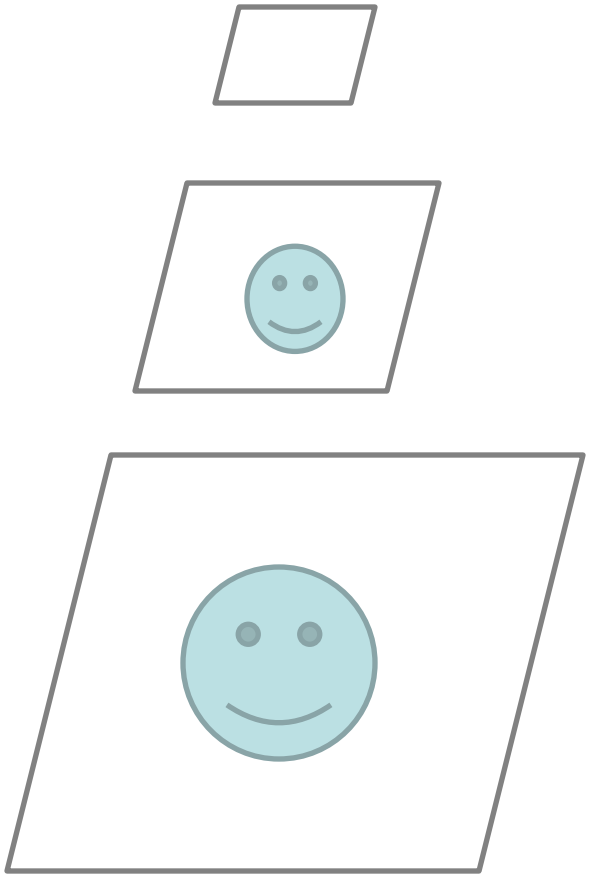
**Original window:**

**Best fit linear function:**

**Lighting corrected window:**
**(linear function subtracted)**

**Histogram equalized window:**

# Image Pyramid Idea



even lower resolution (1/16 of original)

lower resolution image (1/4 of original)

original image (full size)

# Training the Neural Network

## Positive Face Examples

- Nearly 1051 face examples collected from face databases at CMU, Harvard, and WWW
- Faces of various sizes, positions, orientations, intensities
- Eyes, tip of nose, corners and center of mouth labeled manually and used to normalize each face to the same scale, orientation, and position

Result: set of 20 X 20 face training samples

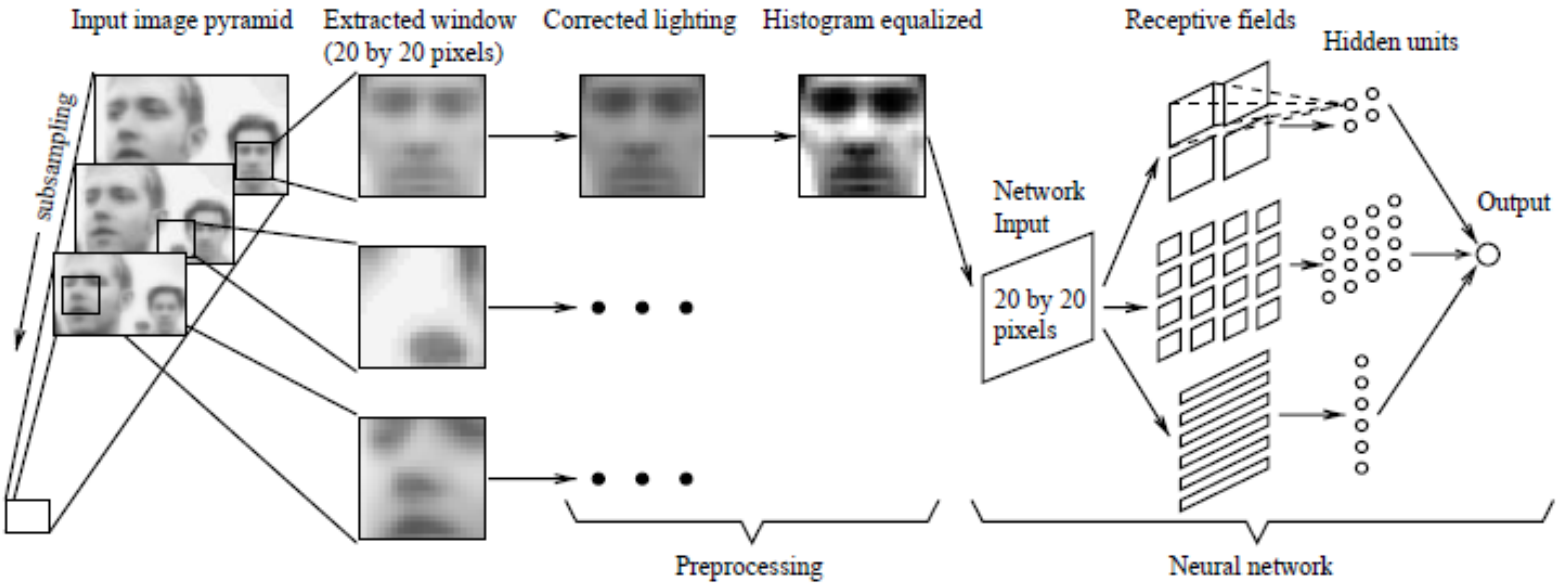
# Training the Neural Network

## Negative Face Examples

- Generate 1000 random nonface images and apply the preprocessing
- Train a neural network on these plus the face images
- Run the system on real scenes that contain no faces
- Collect the false positives
- Randomly select 250 of these and apply preprocessing
- Label them as negative and add to the training set

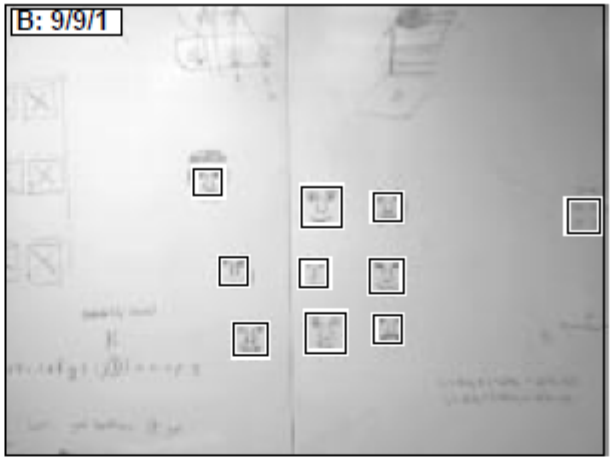
# Overall Algorithm

*Rowley, Baluja, and Kanade: Neural Network-Based Face Detection (PAMI, January 1998)* 17

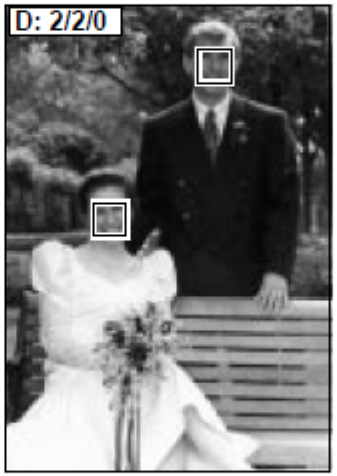


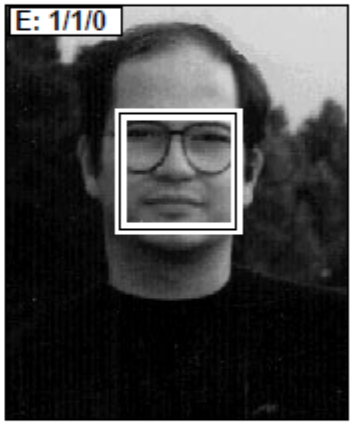
# More Pictures

# Even More

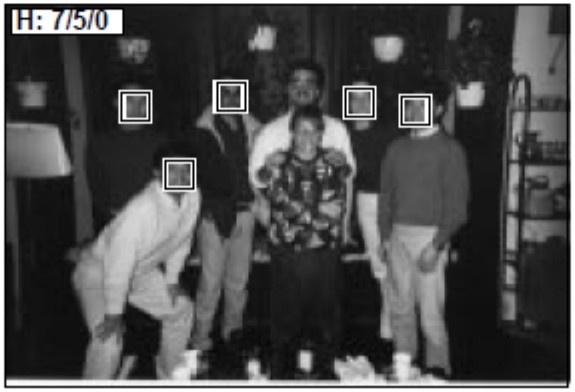



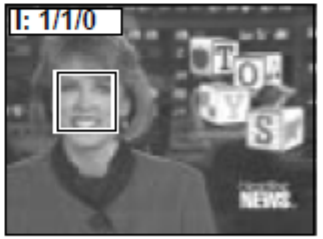


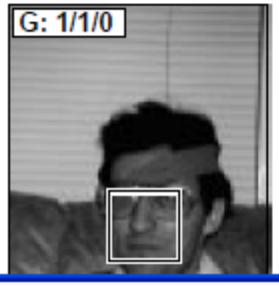







# And More

Accuracy: detected 80-90% on different image sets with an “acceptable number” of false positives

Fast Version: 2-4 seconds per image (in 1998)

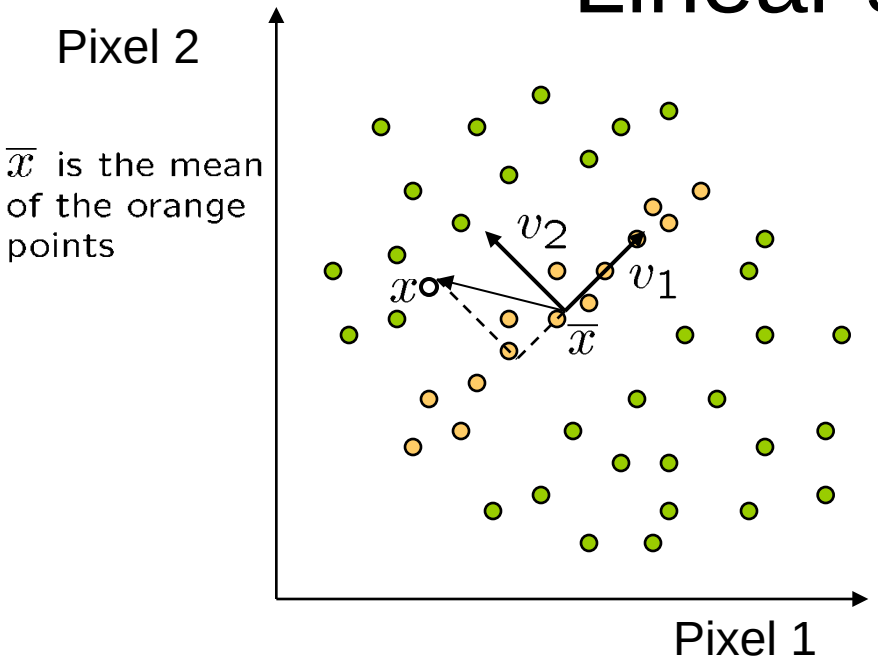
# Object Identification

- Whose face is it?

- We will explore one approach, based on statistics of pixel values, called eigenfaces

- Starting point: Treat N x M image as a vector in NM-dimensional space (form vector by collapsing rows from top to bottom into one long vector)

# 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

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

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

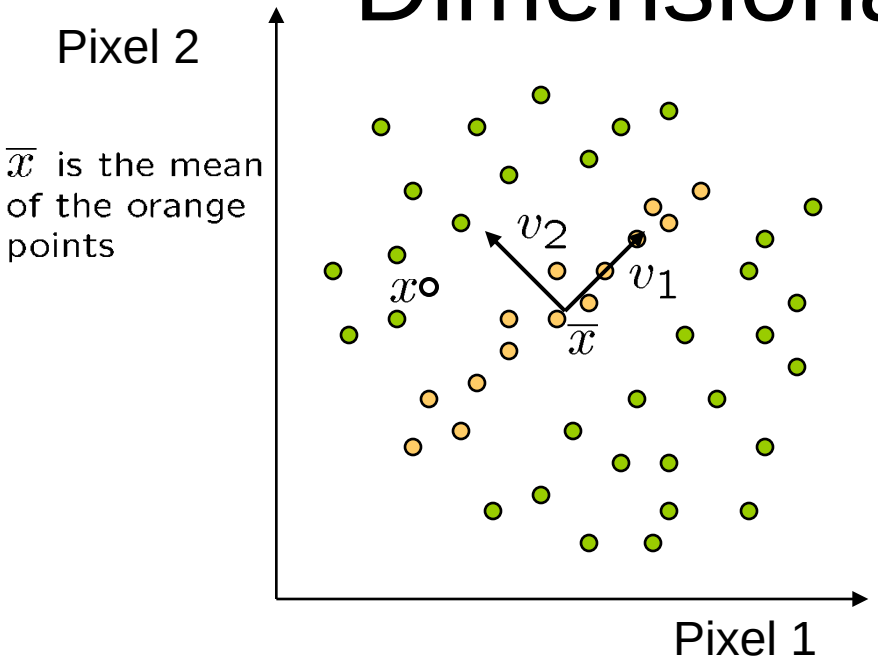
- Classification (to what class does x belong) can be expensive
  - Big search problem

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

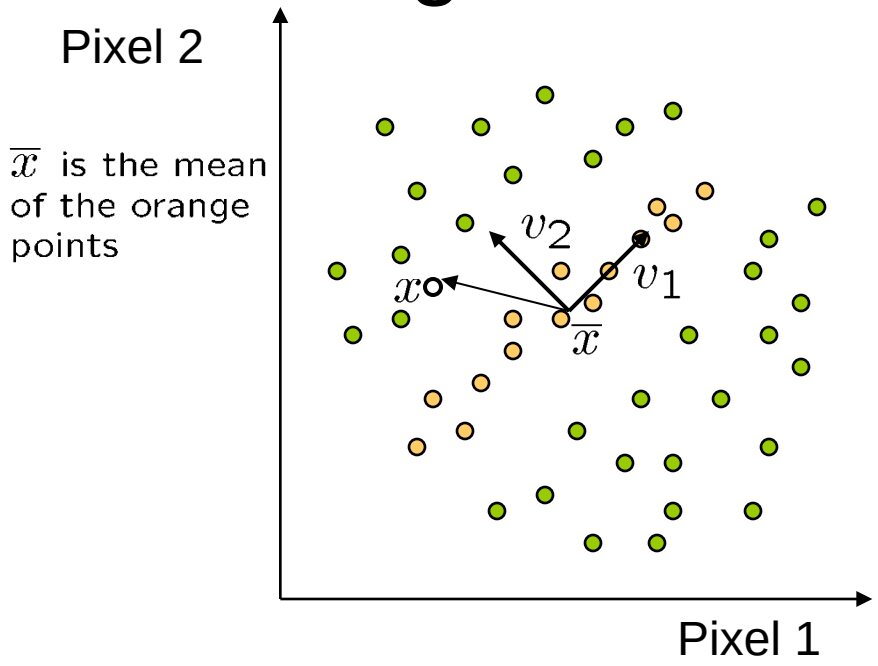
# Dimensionality reduction




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

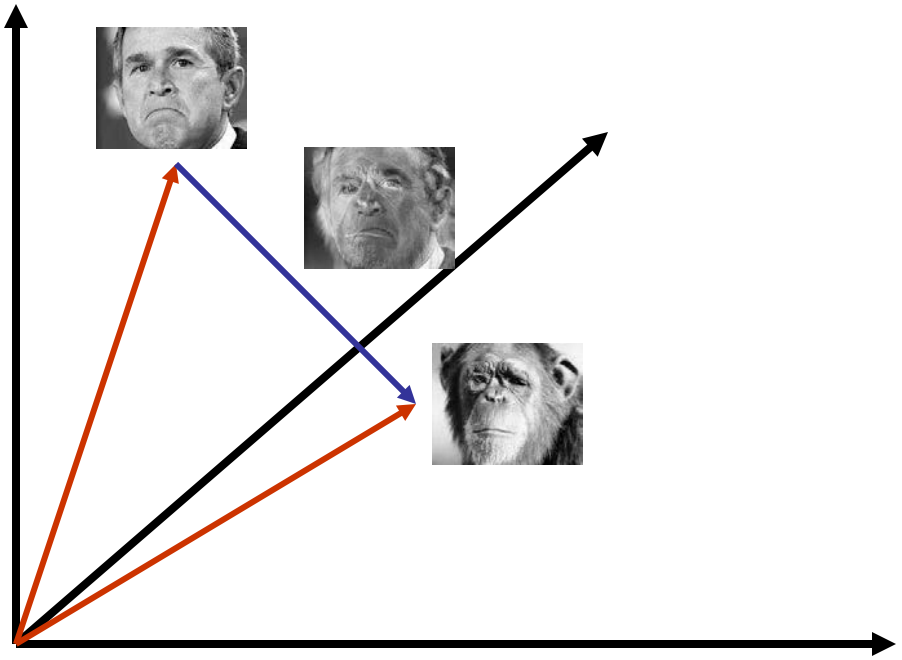
- Suppose each data point is N-dimensional
  - Same procedure applies:

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

  - 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

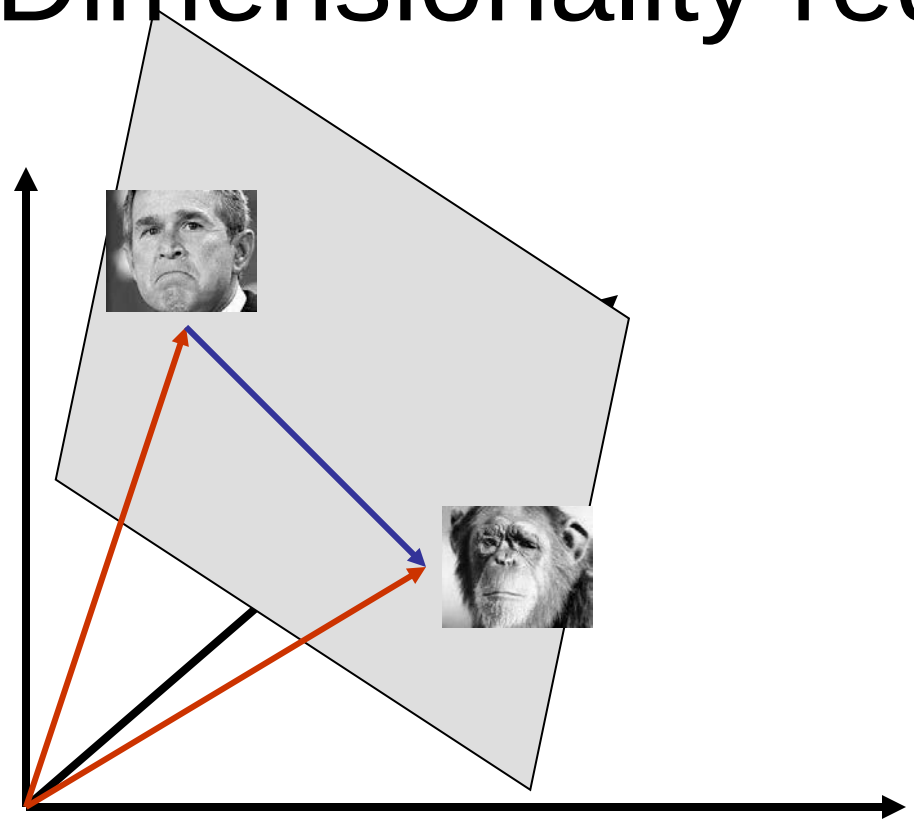
  - We can compress the data by only using the **top few eigenvectors**
    - corresponds to choosing a “linear subspace”
      - represent points on a line, plane, or “hyper-plane”
    - these eigenvectors are known as ***principal component vectors***
    - procedure is known as ***Principal Component Analysis (PCA)***

# 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 space of all faces is a “subspace” of the space of all 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 $\mathbf{x} \approx \bar{\mathbf{x}} + a_1\mathbf{v}_1 + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

# Turk and Pentland's Eigenfaces: Training

- Let $F_1, F_2, \ldots, F_M$ be a set of training face images.
  Let F be their mean and $\Phi_i = F_i - F$.

- Use principal components to compute the eigenvectors and eigenvalues of the covariance matrix.

$$C = (1/M) \sum_{i=1}^{M} \Phi_i \Phi_i^T$$

- Choose the vector u of most significant M eigenvectors to use as the basis.

- Each face is represented as a linear combination of eigenfaces

$u = (u_1, u_2, u_3, u_4, u_5)$; $F_{27} = a_1 * u_1 + a_2 * u_2 + \ldots + a_5 * u_5$

# Matching

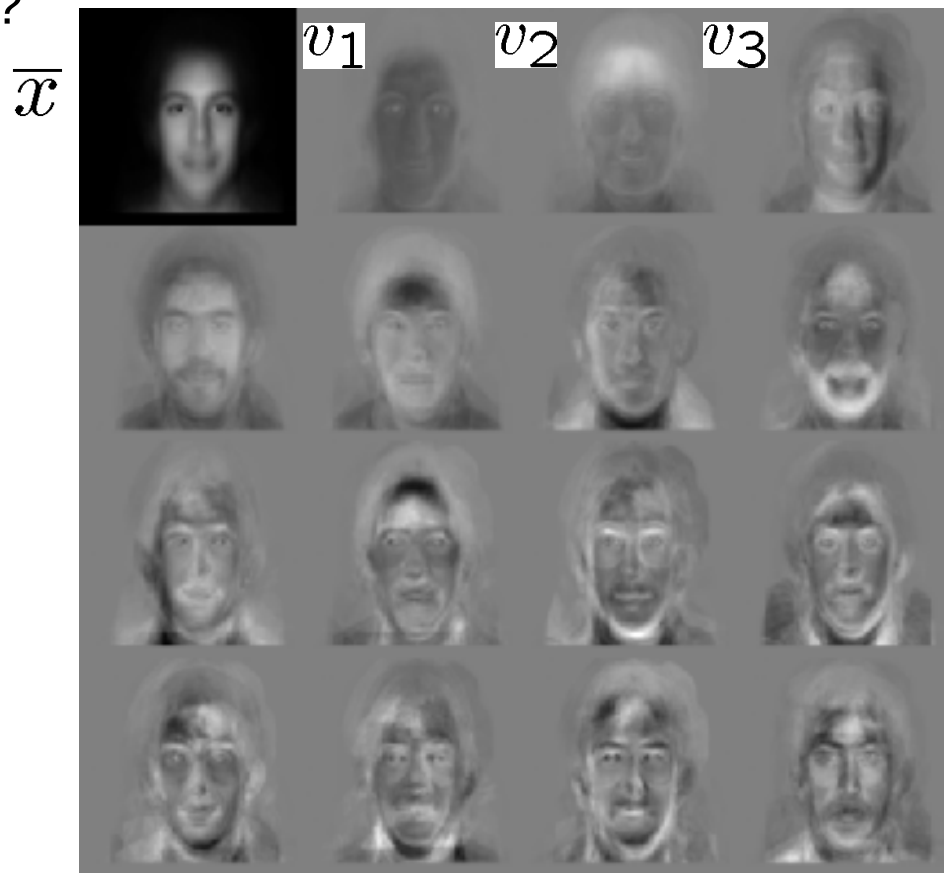
unknown face image I → convert to its eigenface representation → $\Omega = (\Omega_1, \Omega_2, \ldots, \Omega_m)$

Find the face class k that minimizes

$$\varepsilon_k = \| \Omega - \Omega_k \|$$

# Eigenfaces

- PCA extracts the eigenvectors of covariance matrix **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?

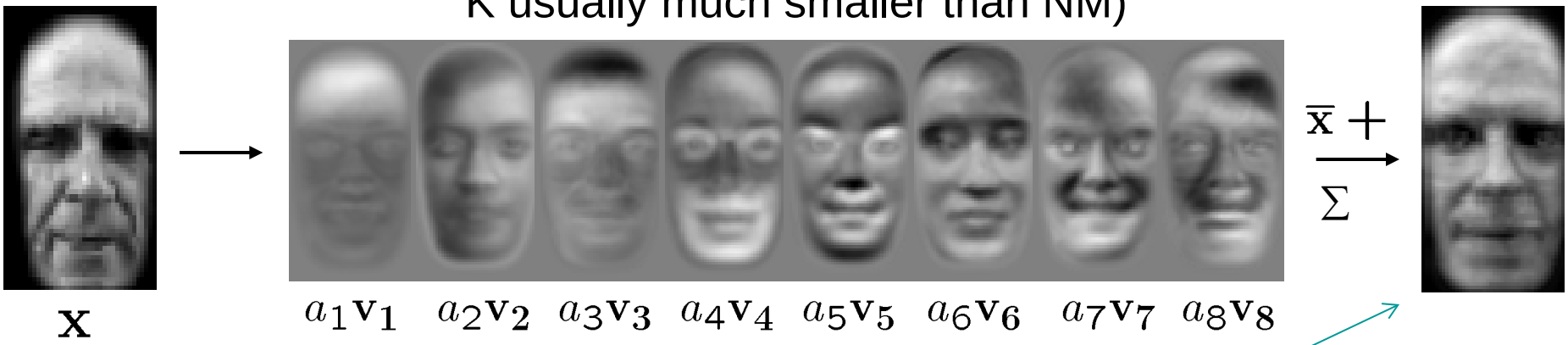


# Projecting onto the eigenfaces

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

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

(Compressed representation of face,
K usually much smaller than NM)




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

# 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

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

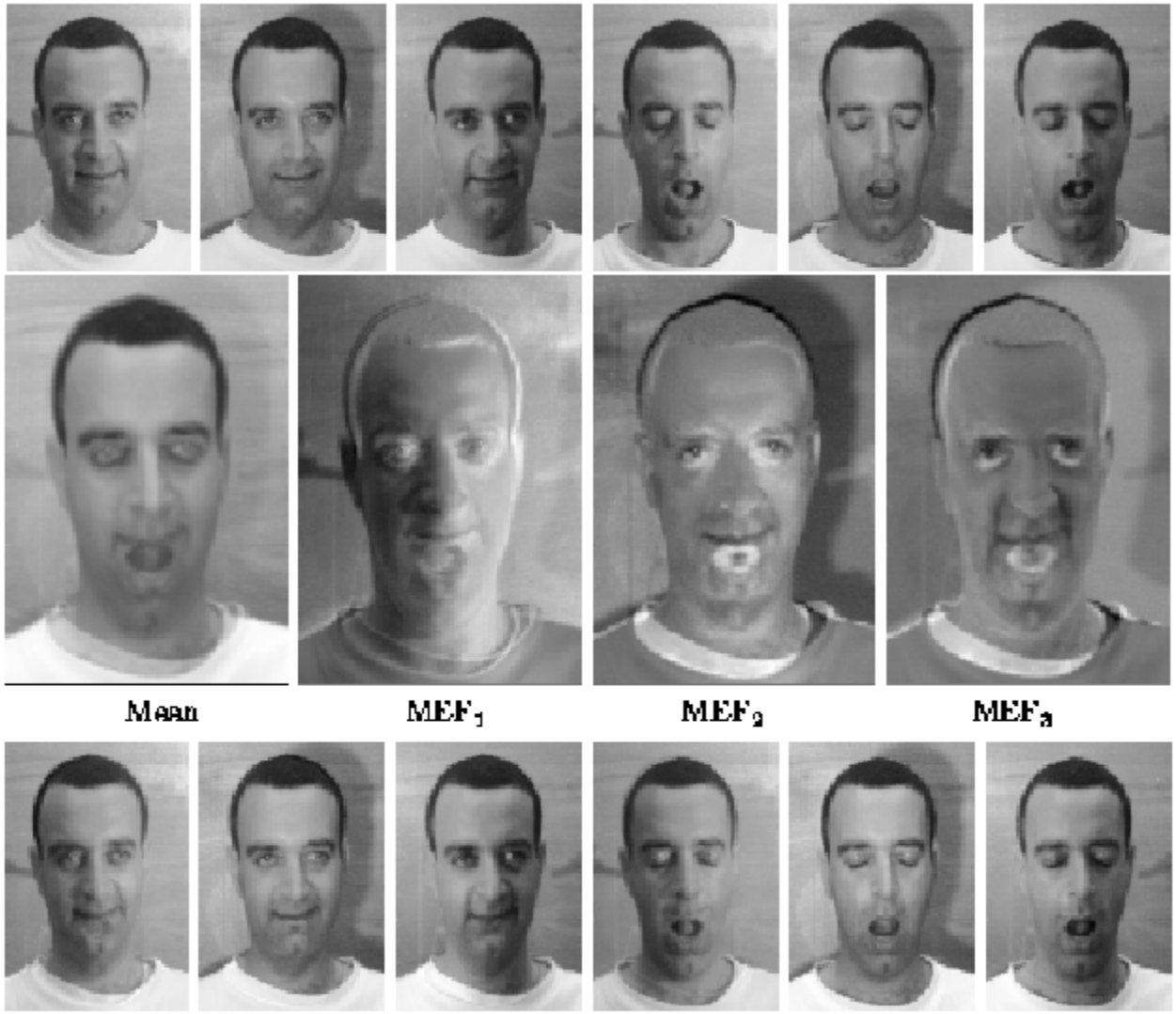
  3. Detect if x is a face

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

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

# Reconstruction Example

training images




mean image

3 eigen-images

linear approxi-mations

# Extension to 3D Objects

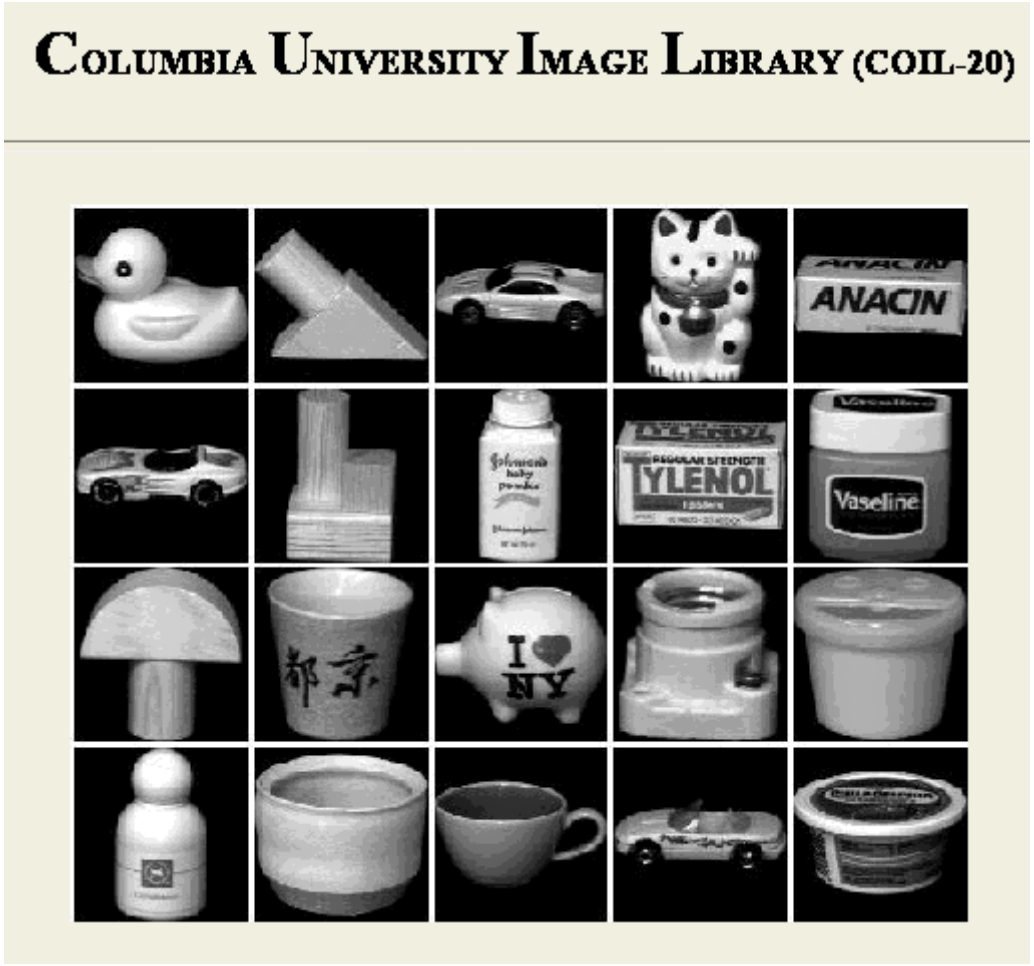
- Murase and Nayar (1994, 1995) extended this idea to 3D objects.

- 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.