Face/Flesh Detection and Face Recognition

Linda Shapiro

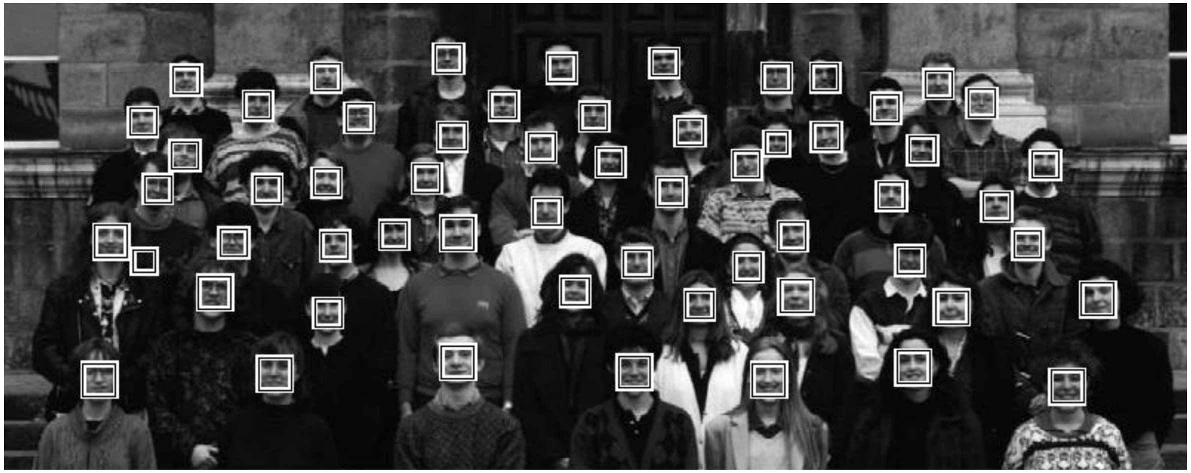
ECE/CSE 576

What’s Coming

1. Review of Bakic flesh detector
2. Fleck and Forsyth flesh detector
3. Review of Rowley face detector
4. The Viola Jones face detector with Adaboost
5. Face recognition with PCA

Person Detection

- Example: Face Detection



(Rowley, Baluja & Kanade, 1998)

- Example: Skin Detection

(Jones & Rehg, 1999)

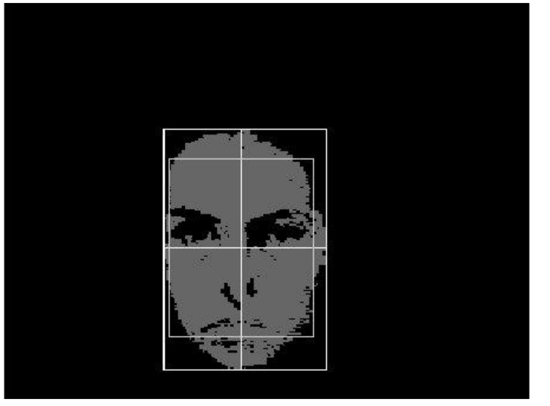
Review: Bakic Flesh Finder

- 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

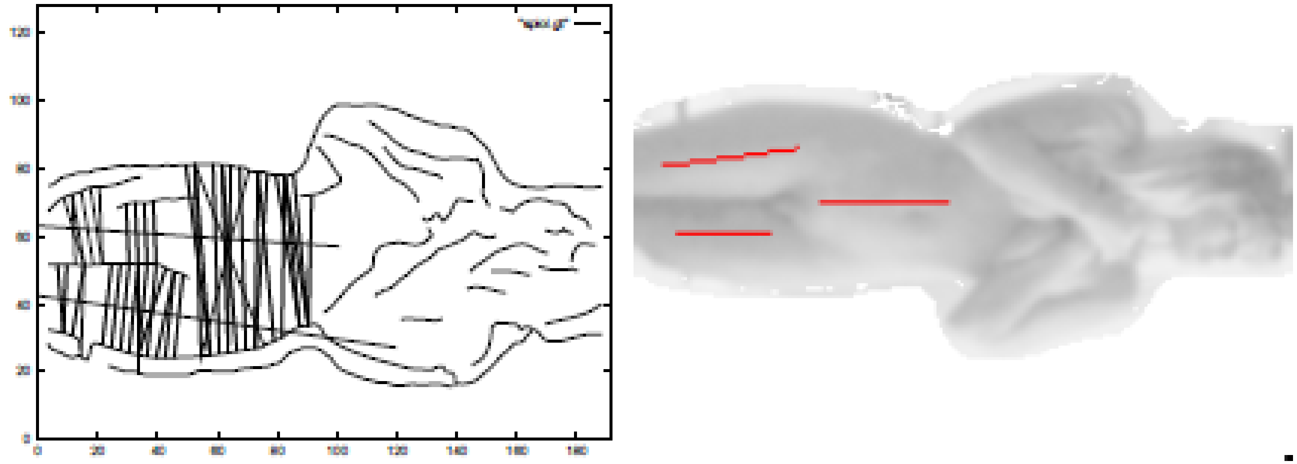
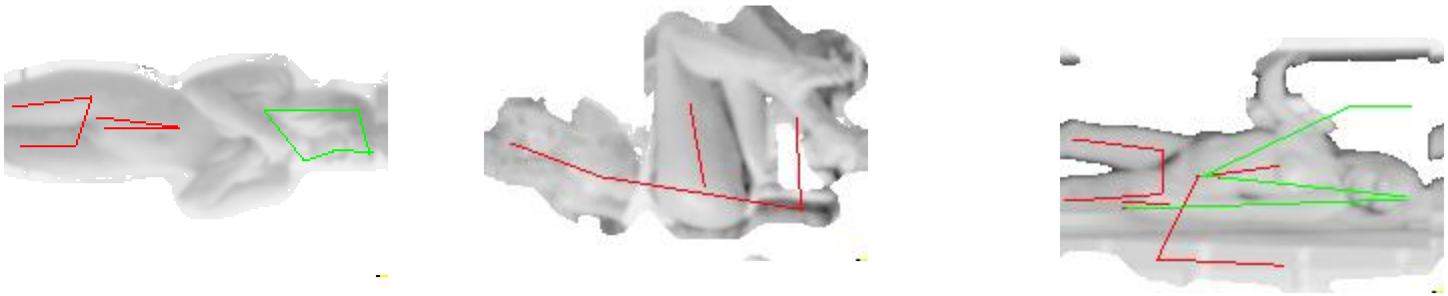


Fig. 2. Grouping a spine and two thighs: *Top left* the segment axes that will be grouped into a spine-thigh group, overlaid on the edges, showing the upper bounds on segment length and the their associated symmetries; *Top right* the spine and thigh group assembled from these segments, overlaid on the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

Face detection

State-of-the-art face detection demo
(Courtesy [Boris Babenko])

Face detection

Where are the faces?

Face Detection

What kind of features?

- Rowley: 32 x 32 subimages at different levels of a pyramid
- Viola/Jones: rectangular features

What kind of classifiers?

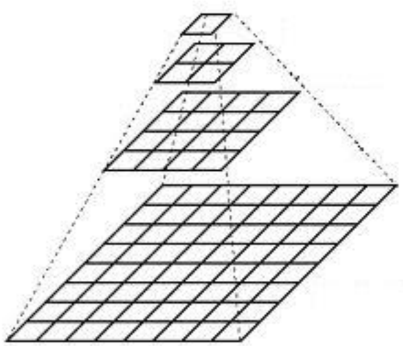
- Rowley: neural nets
- Viola/Jones: Adaboost over simple one-node decision trees (stumps)

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

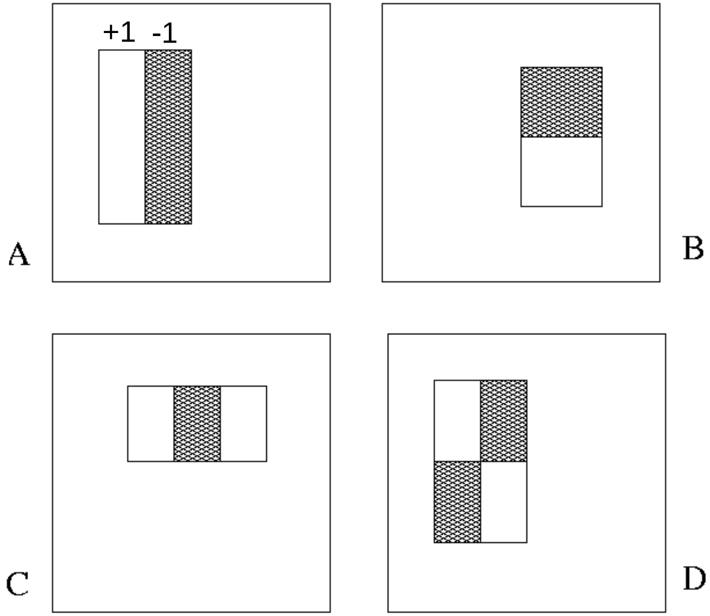
What data is fed to the classifier?

20 x 20 windows in a pyramid structure



Viola/Jones: Image Features

"Rectangle filters"



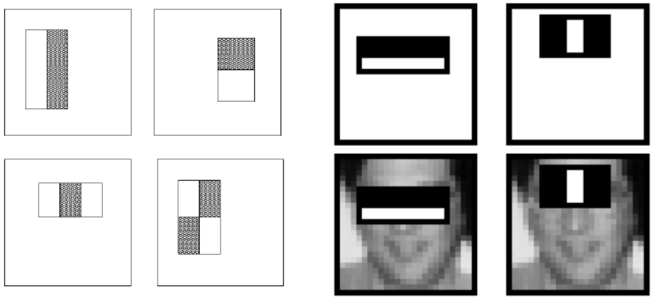
People call them Haar-like features, since similar to 2D Haar wavelets.

Value =

∑ (pixels in white area) –
∑ (pixels in black area)

Feature extraction

"Rectangular" filters



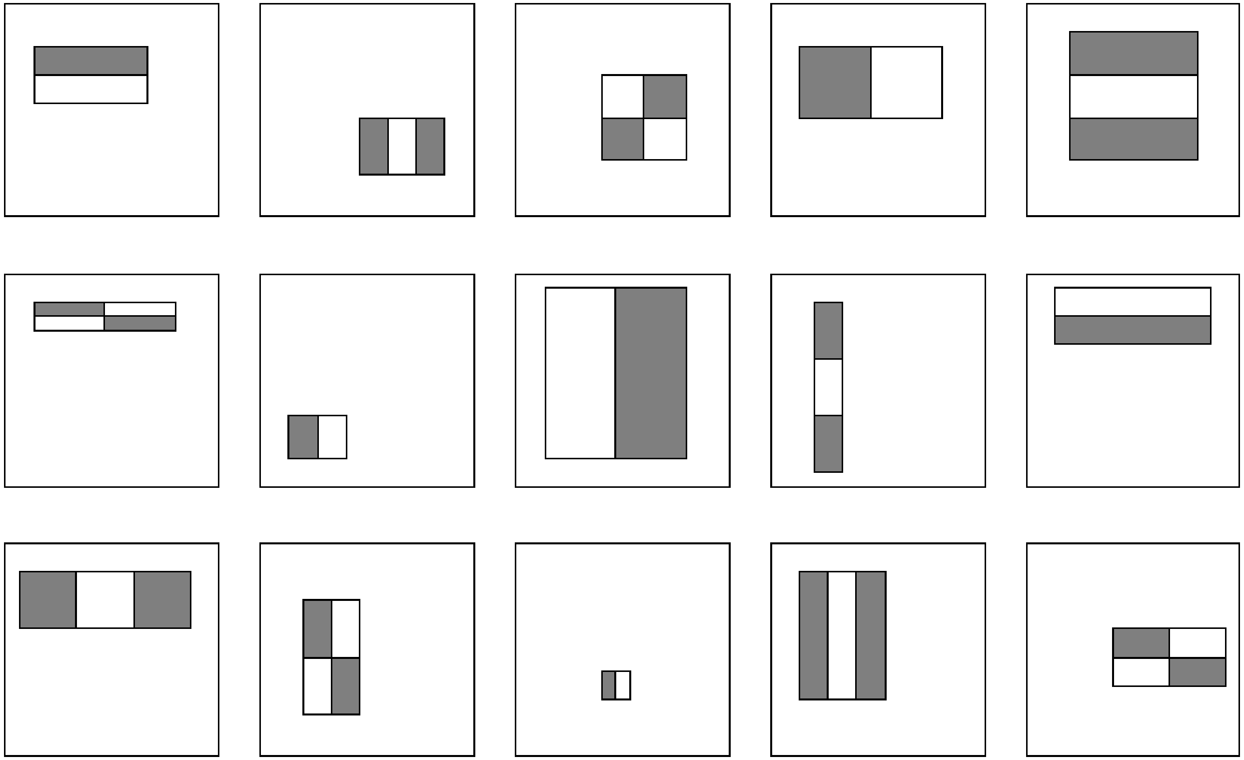
- **Filters can be different sizes.**
- **Filters can be anywhere in the box being analyzed.**
- **Feature output is very simple convolution.**
- **Requires sums of large boxes.**

Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images scale features directly for same cost

Viola & Jones, CVPR 2001

Large library of filters



Considering all possible filter parameters: position, scale, and type:

160,000+ possible features associated with each 24 x 24 window

Use AdaBoost both to select the informative features and to form the classifier

Viola & Jones, CVPR 2001

Feature selection

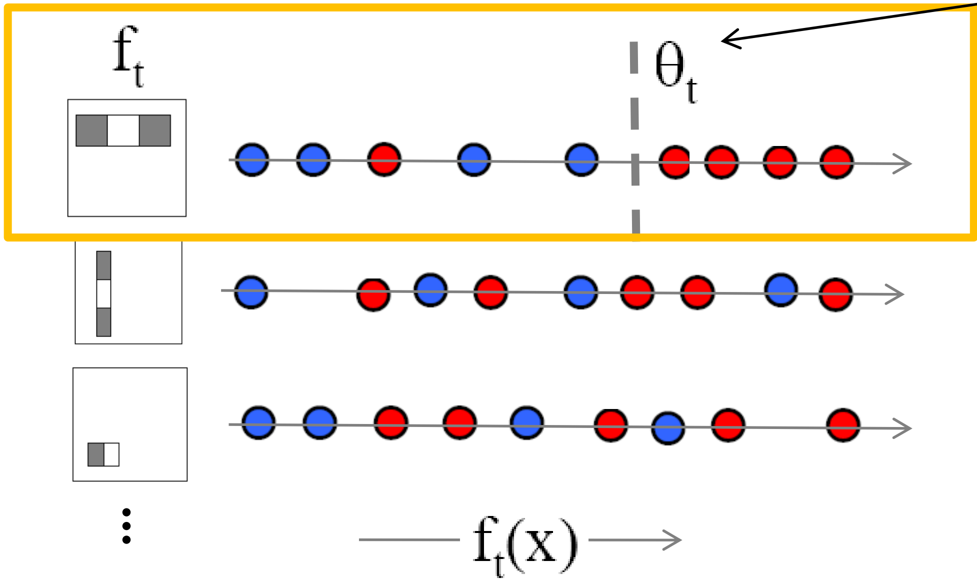
- For a 24x24 detection region, the number of possible rectangle features is ~160,000!
- At test time, it is impractical to evaluate the entire feature set
- Can we create a good classifier using just a small subset of all possible features?
- How to select such a subset?

Basic AdaBoost Review

- Input is a set of training examples (X_i, y_i) i = 1 to m.
- We train a sequence of weak classifiers, such as decision trees, neural nets or SVMs. Weak because not as strong as the final classifier.
- The training examples will have weights, initially all equal.
- At each step, we use the current weights, train a new classifier, and use its performance on the training data to produce new weights for the next step (normalized).
- But we keep ALL the weak classifiers.
- When it's time for testing on a new feature vector, we will combine the results from all of the weak classifiers.

AdaBoost for feature+classifier selection

- Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of *weighted* error.



θ_t is a threshold for classifier h_t

Outputs of a possible rectangle feature on training examples x (faces and non faces)

Resulting weak classifier:

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ \boxed{0} & \text{otherwise} \end{cases}$$

For next round, reweight the examples according to errors, choose another filter/threshold combo.

Viola & Jones, CVPR 2001

Weak Classifiers

- Each weak classifier works on exactly one rectangle feature.
- Each weak classifier has 3 associated variables
  1. its threshold θ
  2. its polarity p
  3. its weight α
- The polarity can be 0 or 1 (in our code)
- The weak classifier computes its one feature f
  - When the polarity is 1, we want $f > \theta$ for face
  - When the polarity is 0, we want $f < \theta$ for face
- The weight will be used in the final classification by AdaBoost.

- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

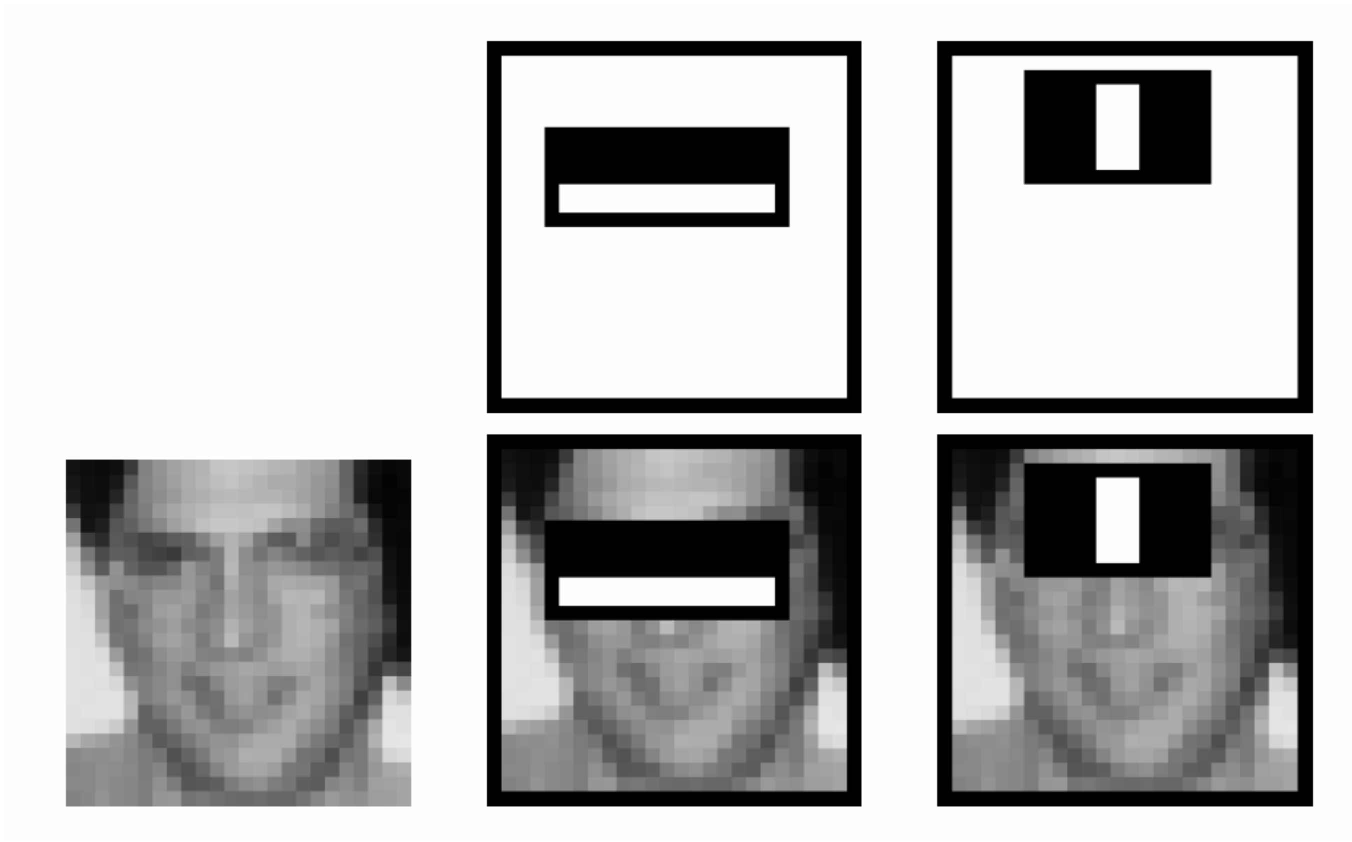
where $\alpha_t = \log \frac{1}{\beta_t}$

$\beta_t = \varepsilon_t / (1- \varepsilon_t)$: the training error of the classifier h_t

- Final classifier is combination of the weak ones, weighted according to error they had.
- E.g. $\beta_t = .25/.75 = .3333$. $1/.3333 = 3$; $\log_2 3 = 1.58$,
- $\beta_t = .1/.9$; $\log_2 9 = 3.16$
- If the error were 0, $\beta_t = 0/1 = 0$, and 1/0 is infinite, so code has to handle that but the weight should be large!

Boosting for face detection

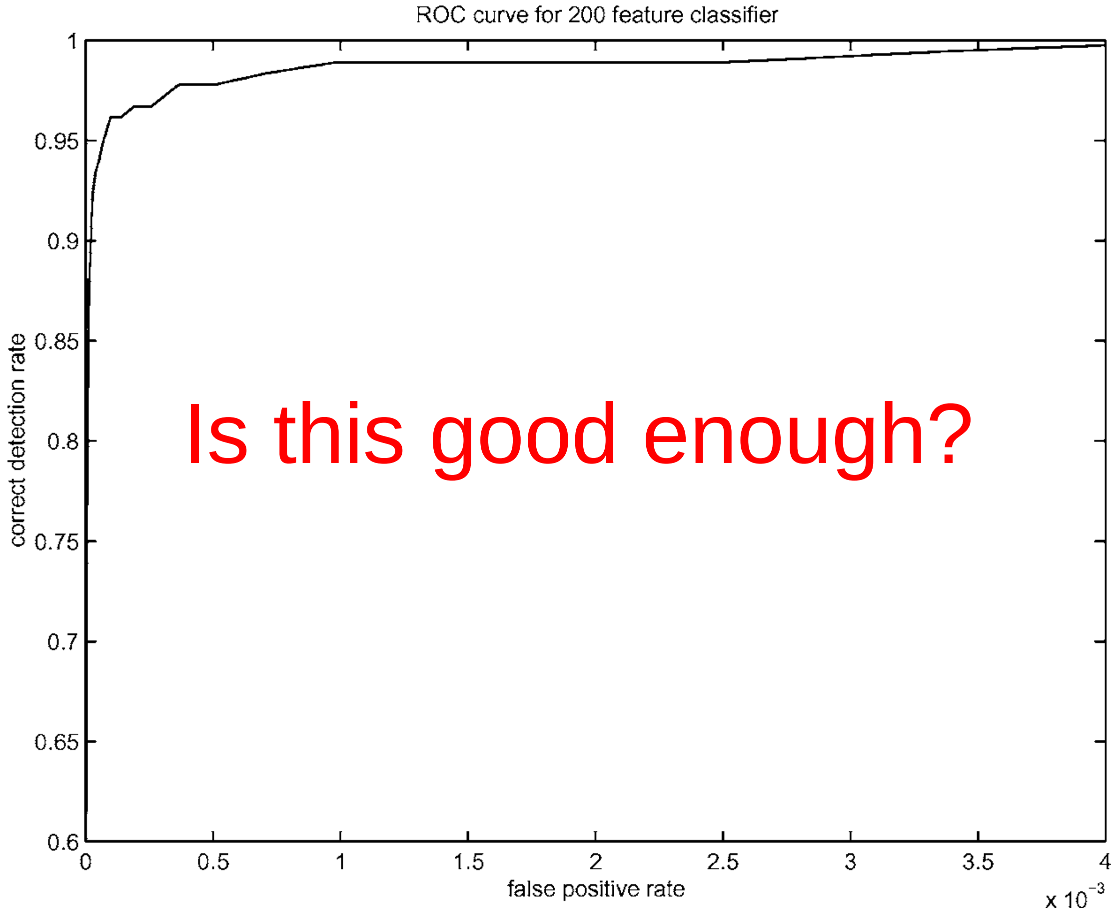
- First two features selected by boosting:



This feature combination can yield 100% detection rate and 50% false positive rate

Boosting for face detection

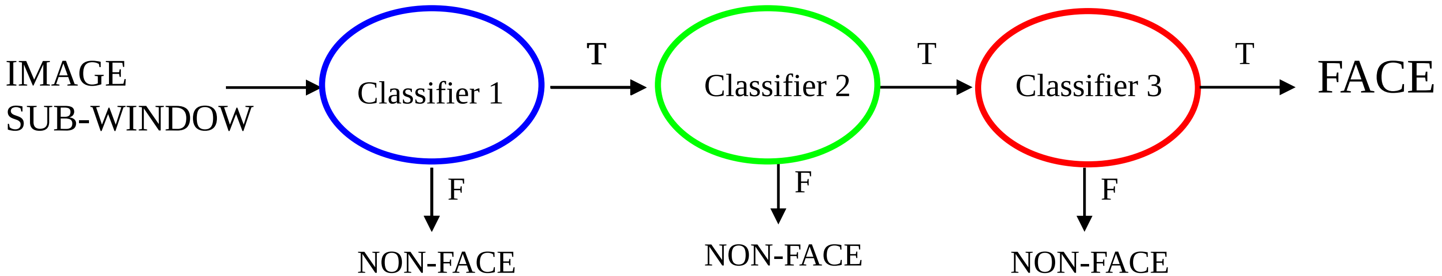
- A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084



Receiver operating characteristic (ROC) curve

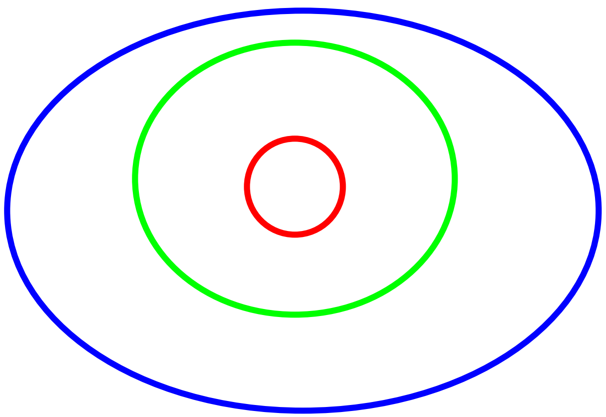
Attentional cascade (from Viola-Jones)

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window

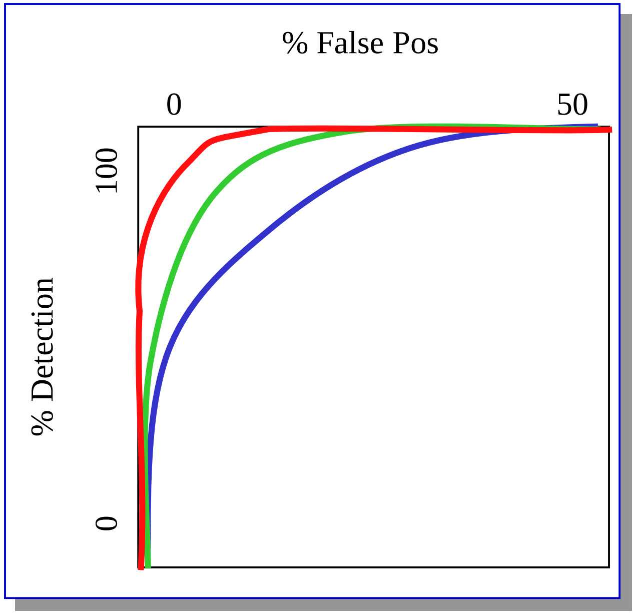


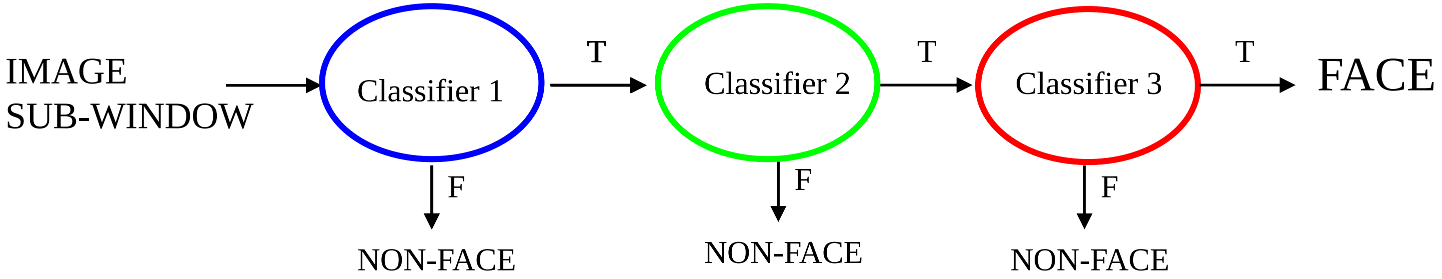
Attentional cascade

- Chain of classifiers that are progressively more complex and have lower false positive rates:



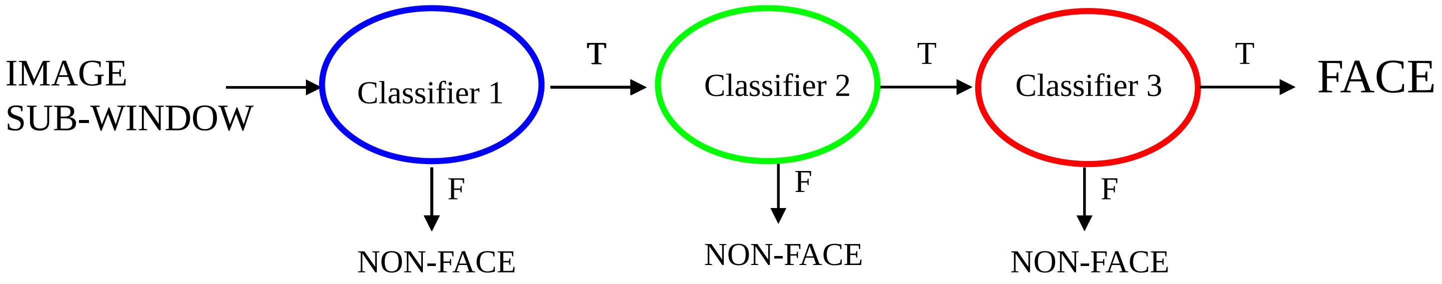
Receiver operating characteristic





Attentional cascade

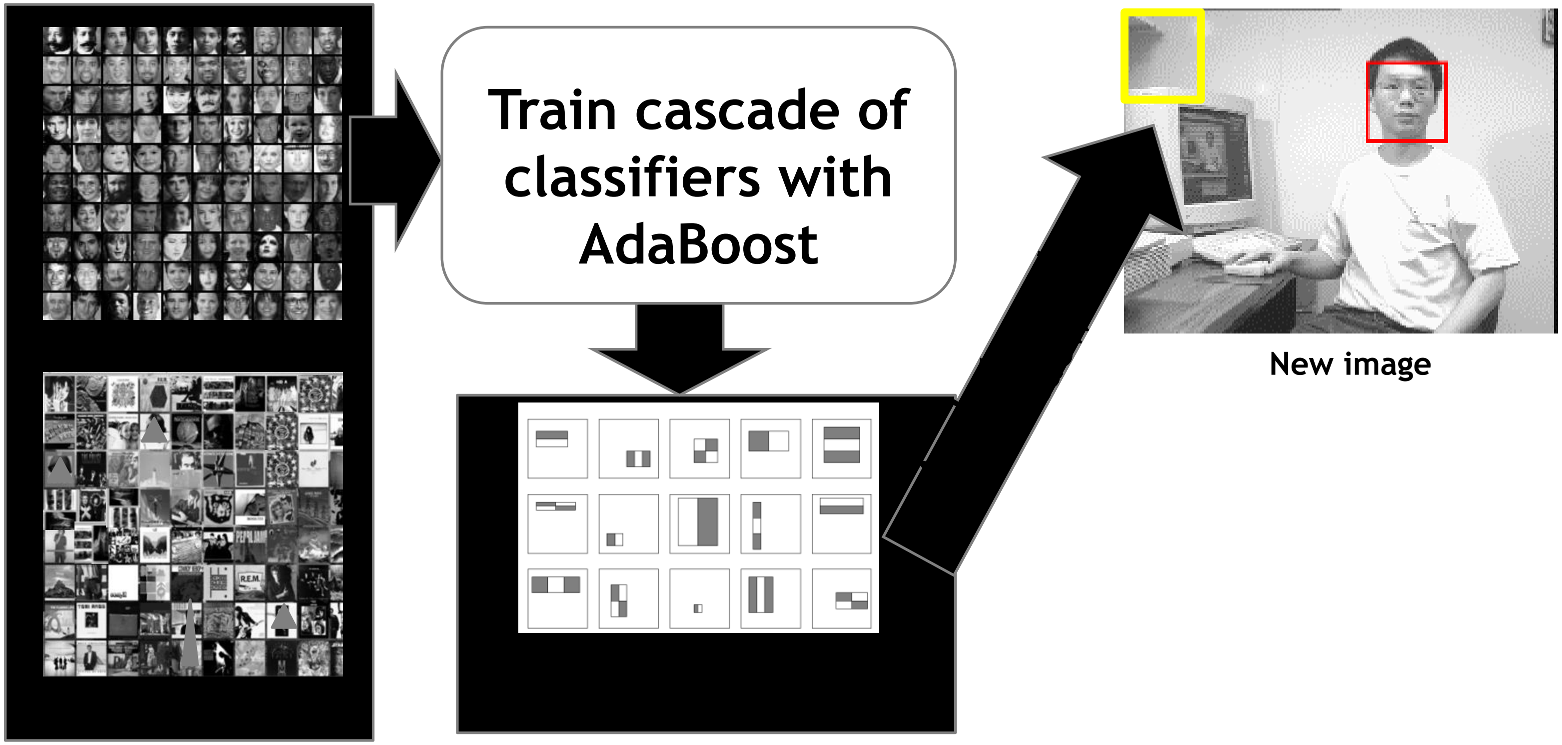
- The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages
- A detection rate of 0.9 and a false positive rate on the order of 10^{-6} can be achieved by a 10-stage cascade if each stage has a detection rate of 0.99 ($0.99^{10} \approx 0.9$) and a false positive rate of about 0.30 ($0.3^{10} \approx 6\times10^{-6}$)



Training the cascade

- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
  - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
  - Test on a *validation set*
- If the overall false positive rate is not low enough, then add another stage
- Use false positives from current stage as the negative training examples for the next stage

Viola-Jones Face Detector: Summary



Train with 5K positives, 350M negatives

Real-time detector using 38 layer cascade

6061 features in final layer

[Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]

The implemented system

- Training Data
  - 5000 faces
    - – All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - – 9500 non-face images
  - Faces are normalized
    - – Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

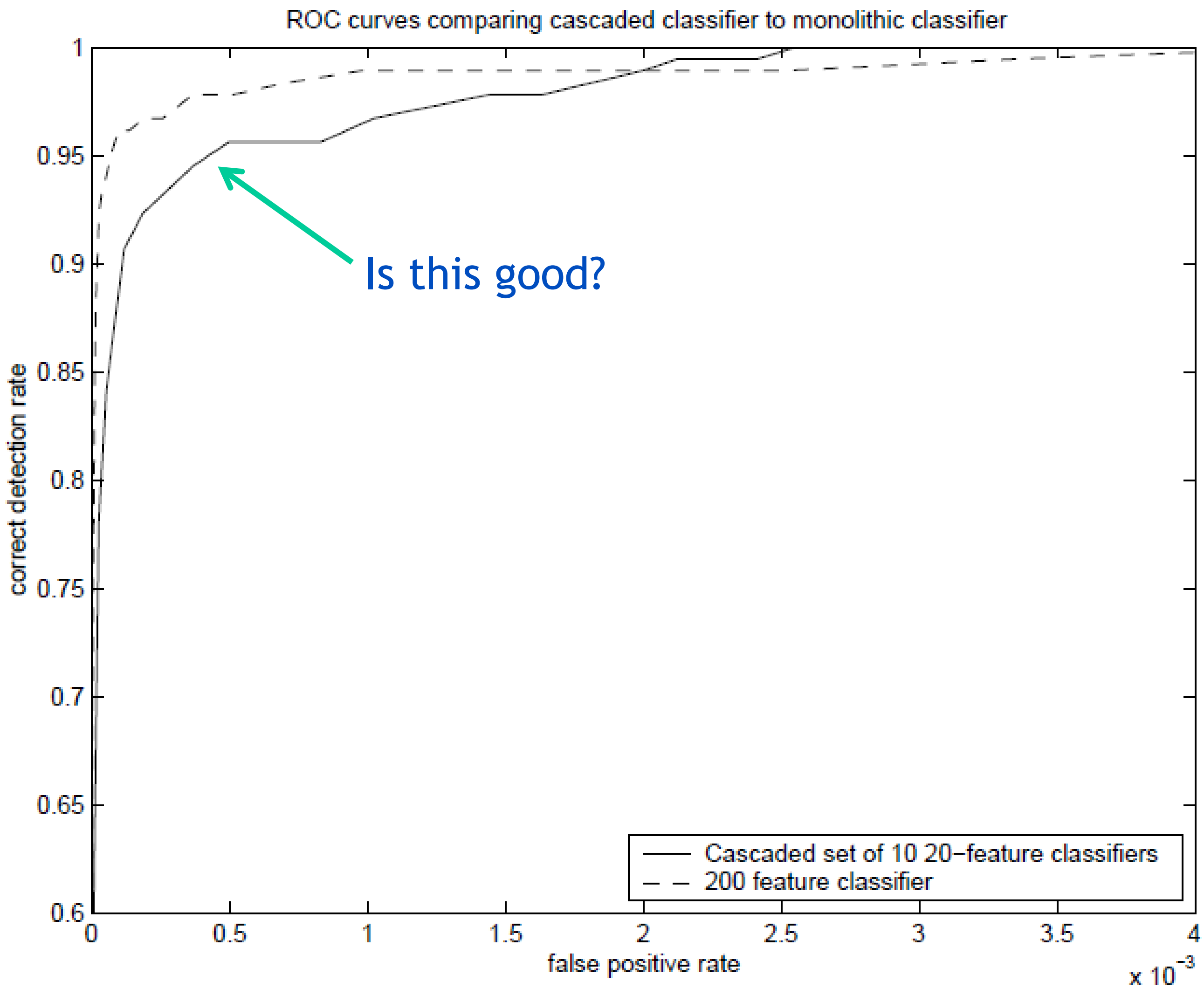
System performance

- Training time: “weeks” on 466 MHz Sun workstation
- 38 layers, total of 6061 features
- Average of 10 features evaluated per window on test set
- “On a 700 Mhz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

Non-maximal suppression (NMS)

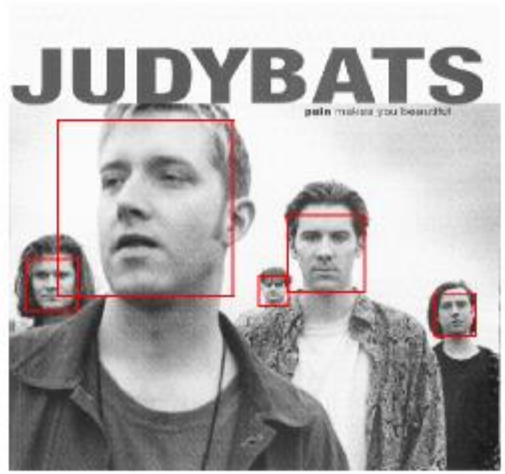
Many detections above threshold.

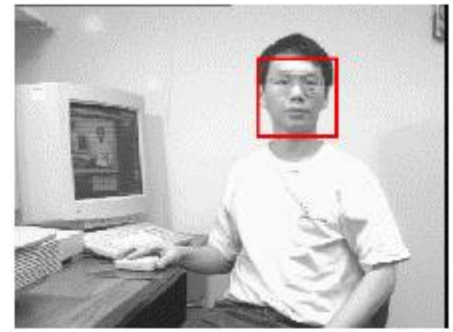
Non-maximal suppression (NMS)

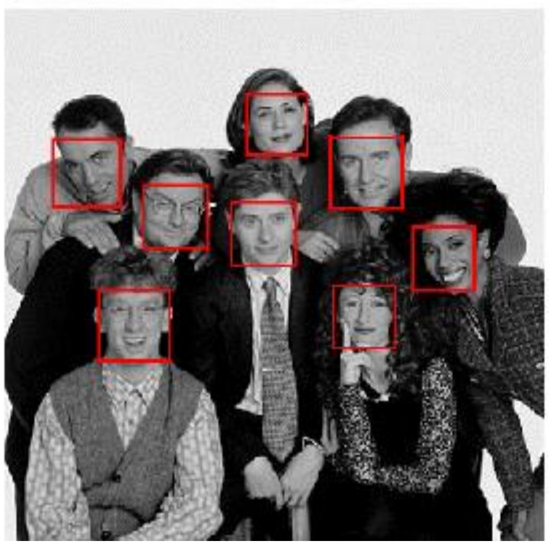


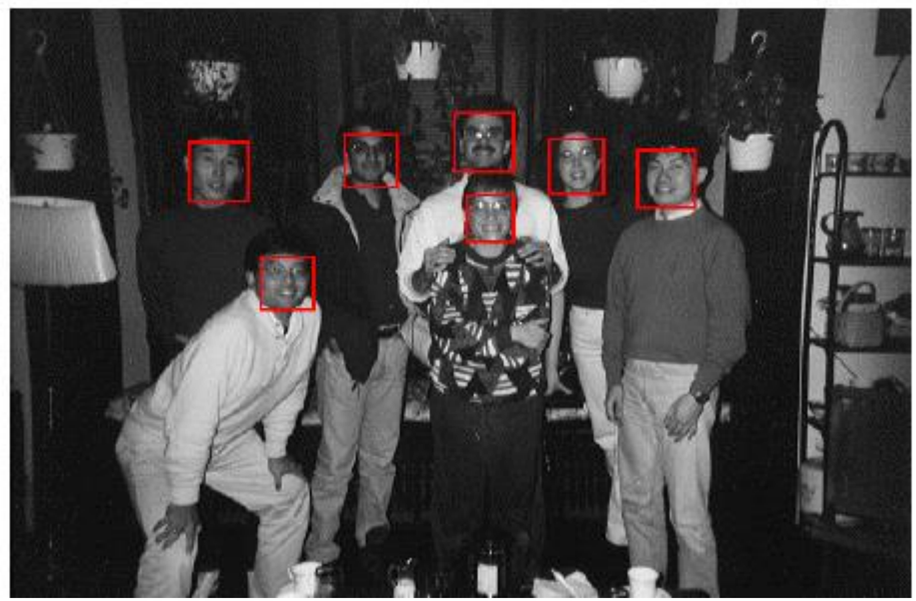
Similar accuracy, but 10x faster

Viola-Jones Face Detector: Results

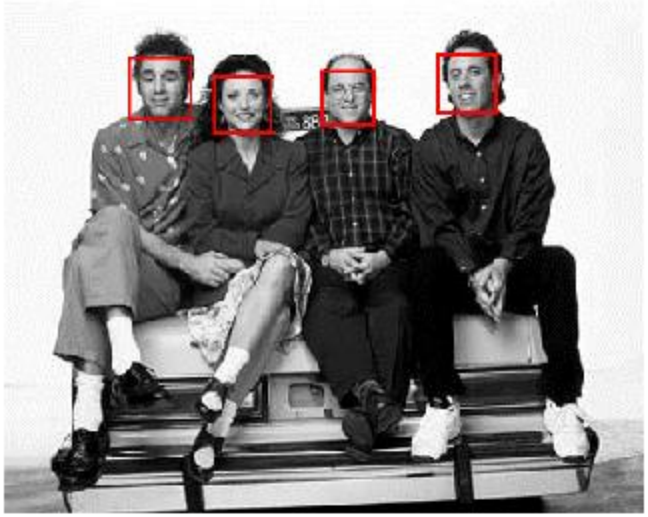


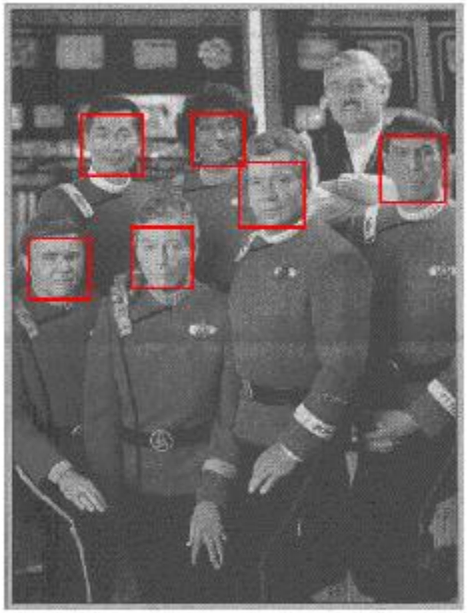


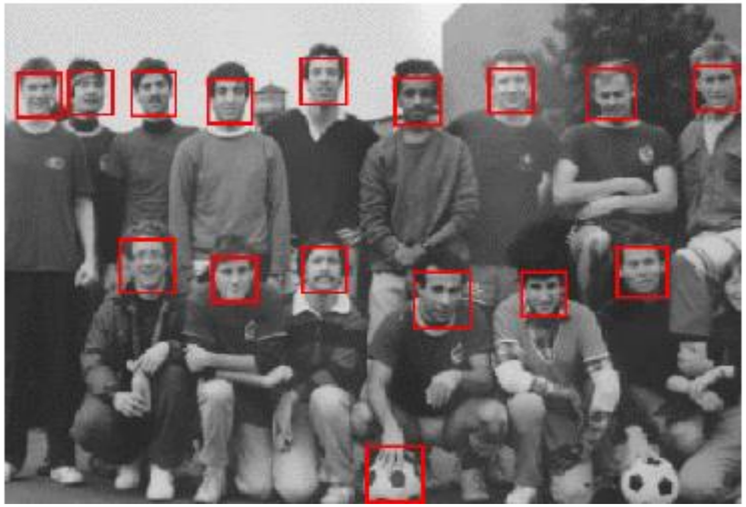


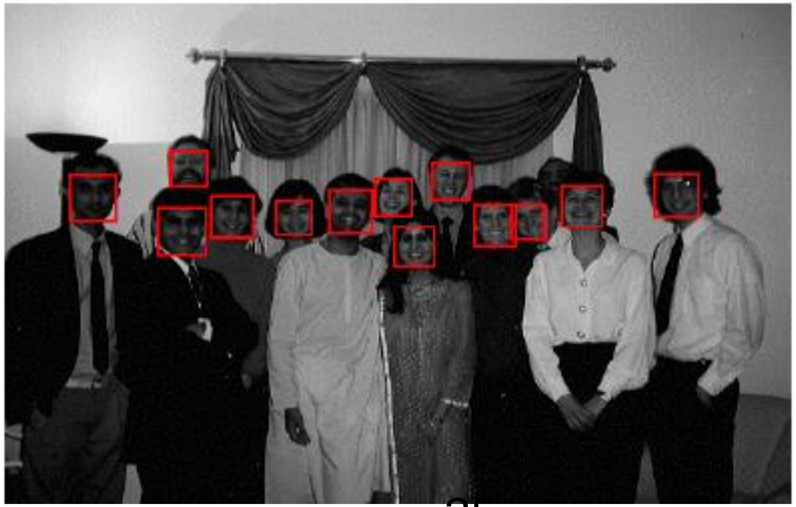


Viola-Jones Face Detector: Results





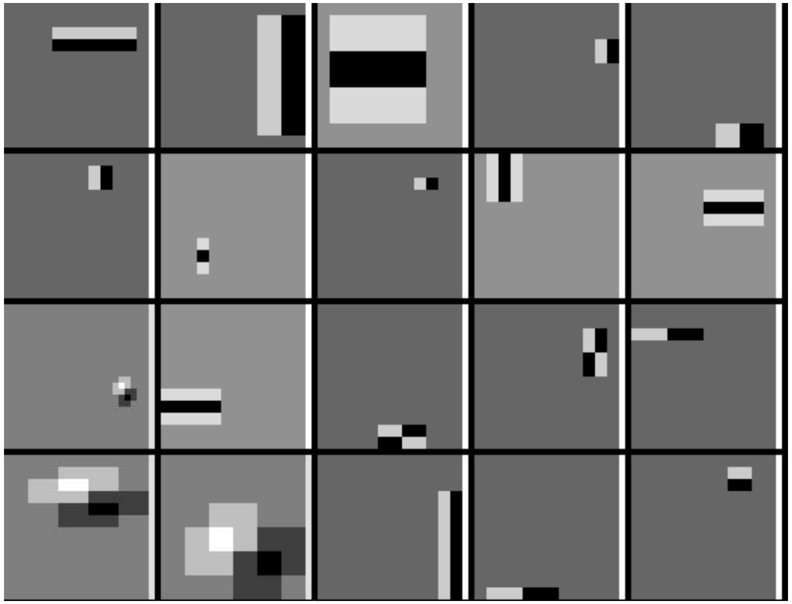




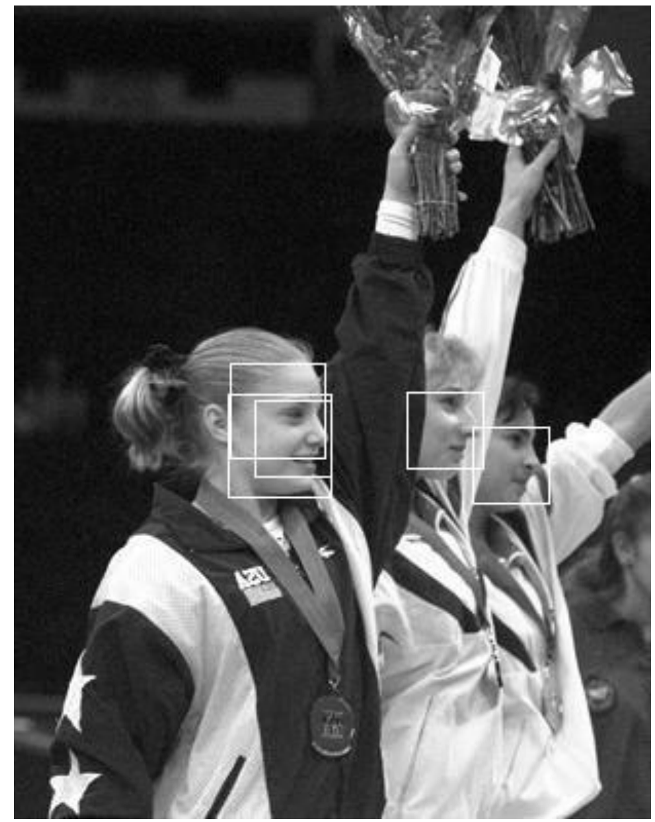
Viola-Jones Face Detector: Results

Detecting profile faces?

Detecting profile faces requires training separate detector with profile examples.



Viola-Jones Face Detector: Results



Summary: Viola/Jones detector

- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows

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 (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

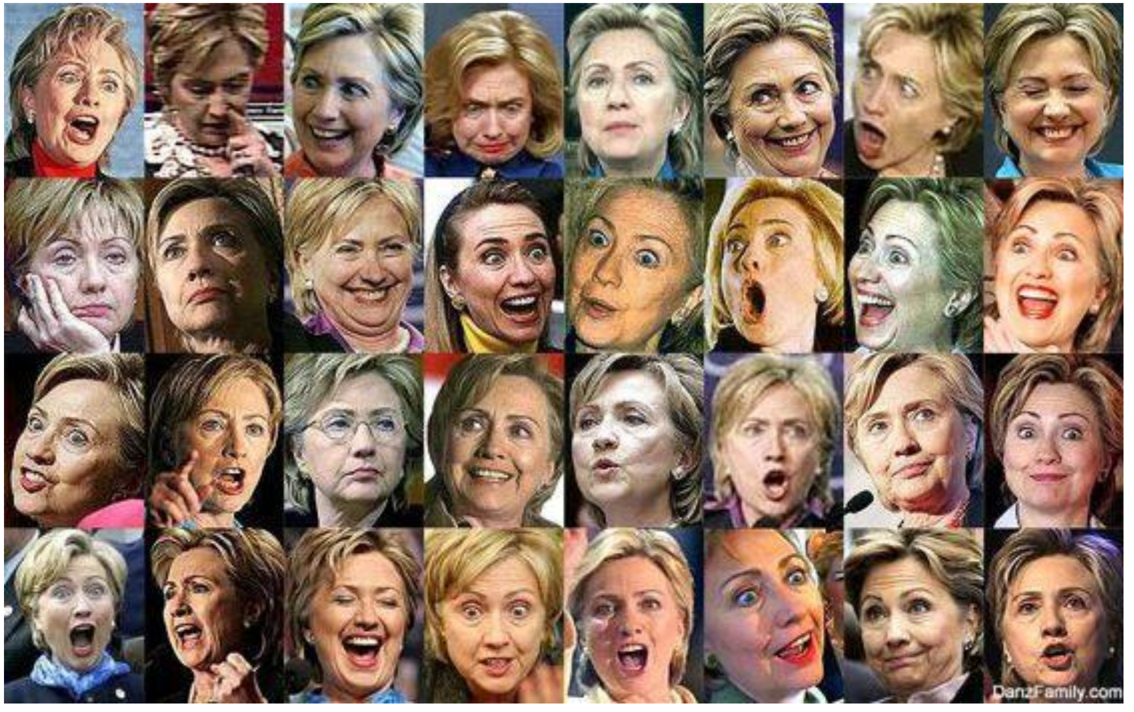
- E.g., security

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

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

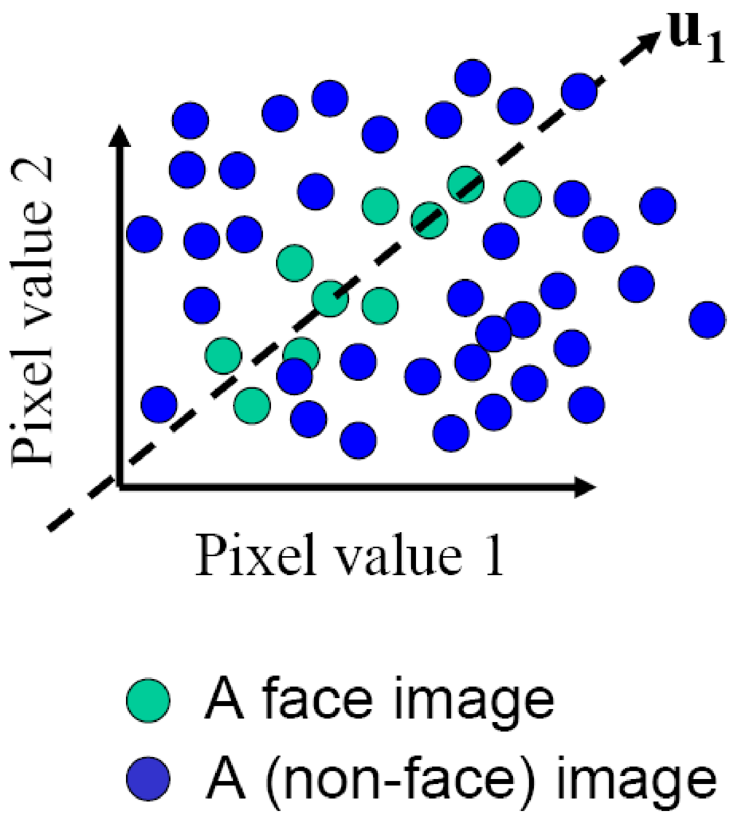
$$k = \underset{k}{\operatorname{argmin}} \left\| \mathbf{y}_k - \mathbf{x} \right\|$$

The space of all face images

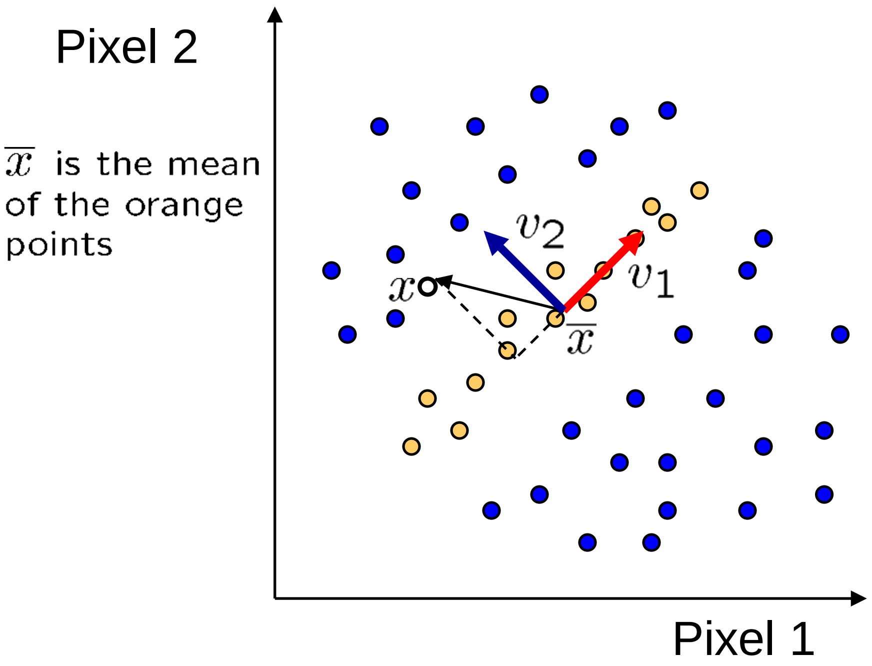
- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 100x100 image = 10,000 dimensions
  - Slow and lots of storage
- But very few 10,000-dimensional vectors are valid face images
- We want to effectively model the subspace of face images

The space of all face images

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



Linear subspaces



v_1 is the major direction of the orange points and **v_2** is perpendicular to **v_1**.

Convert **x** into **v_1**, **v_2** coordinates

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

What does the **v_2** coordinate measure?

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

What does the **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

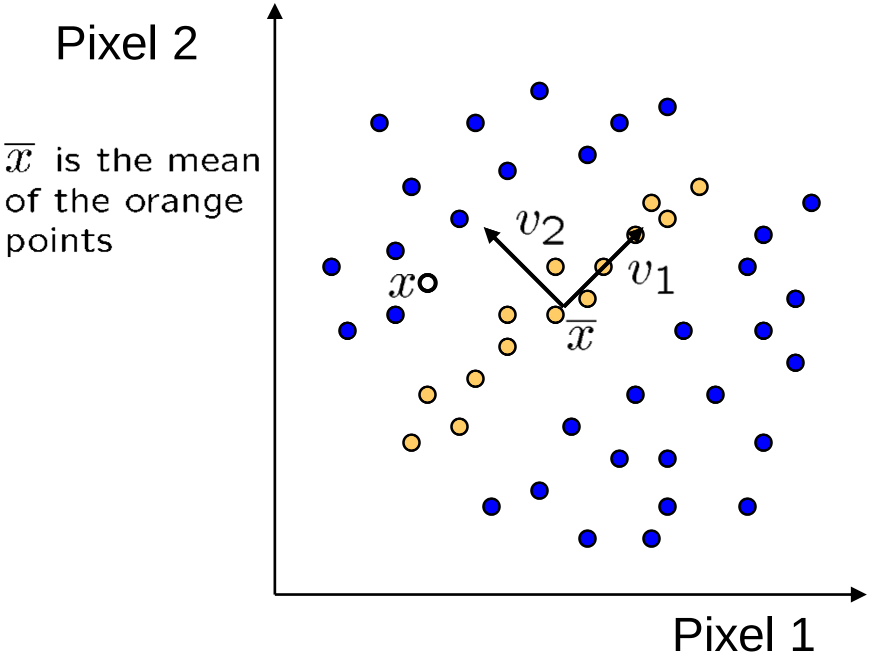
- Big search problem

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

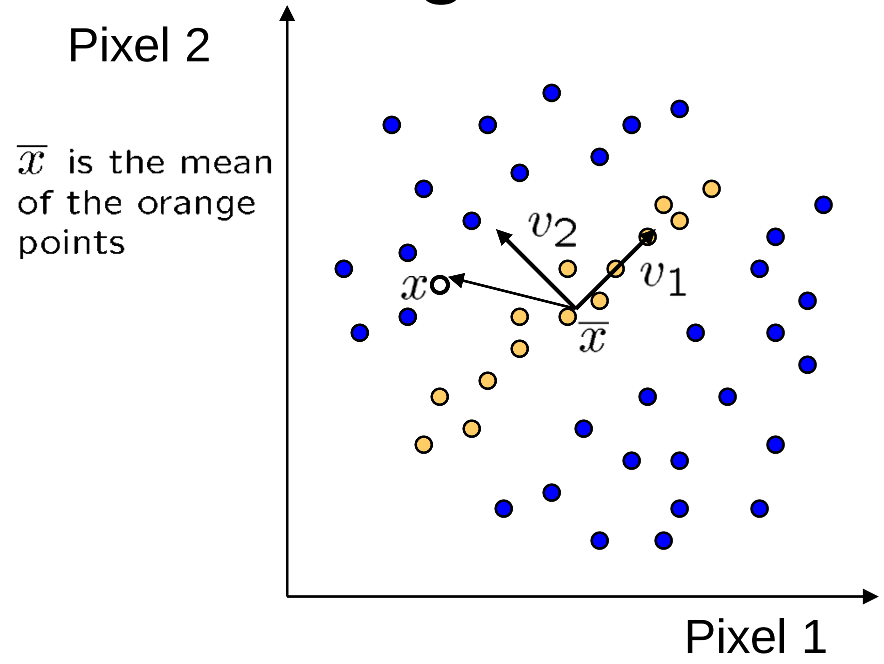
Dimensionality reduction



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:

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

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

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

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

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

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

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)

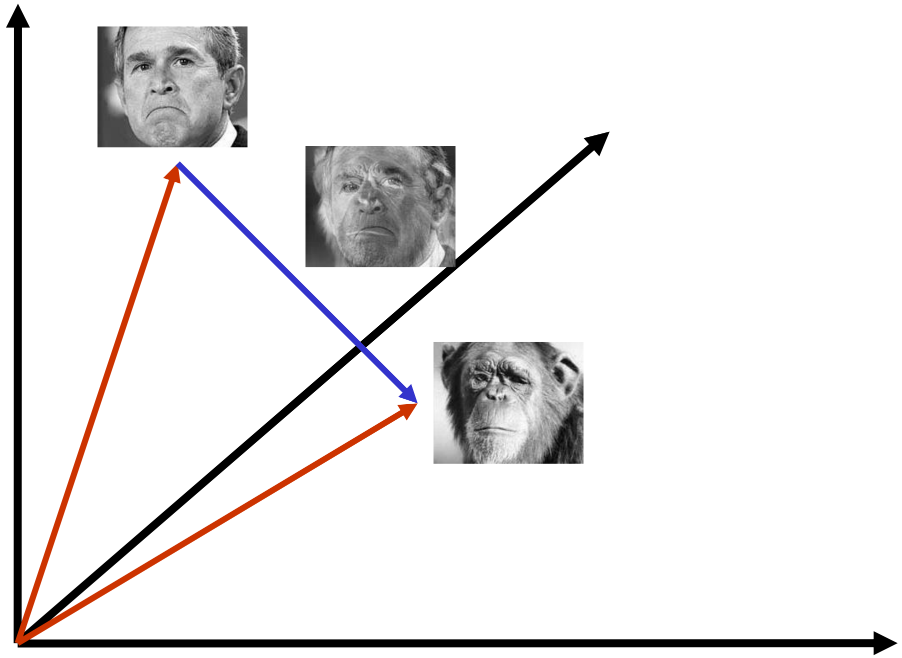
Suppose each data point is N-dimensional

- Same procedure applies:

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

- 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***

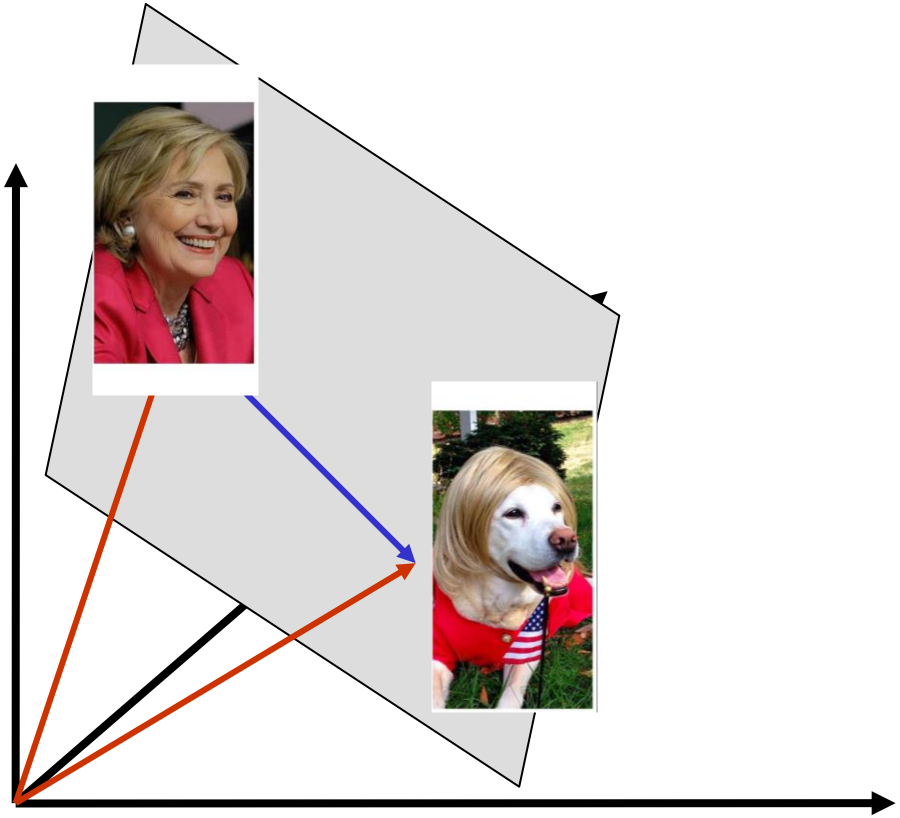
The space of faces



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

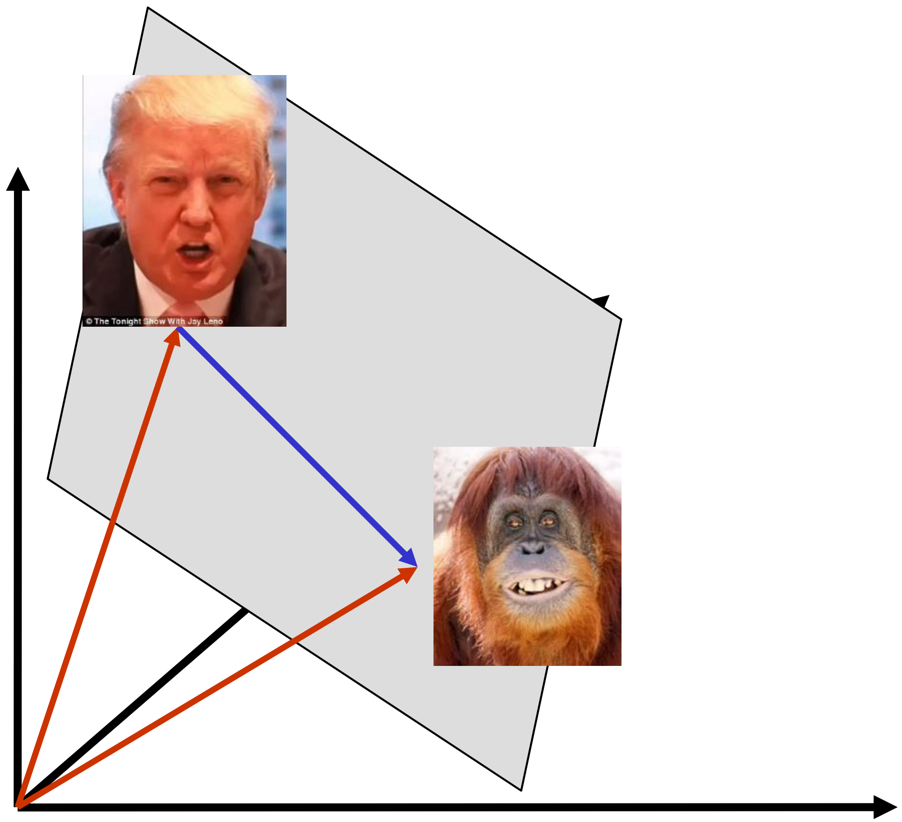
Dimensionality reduction



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 $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_K}$
  - any face $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

Dimensionality reduction



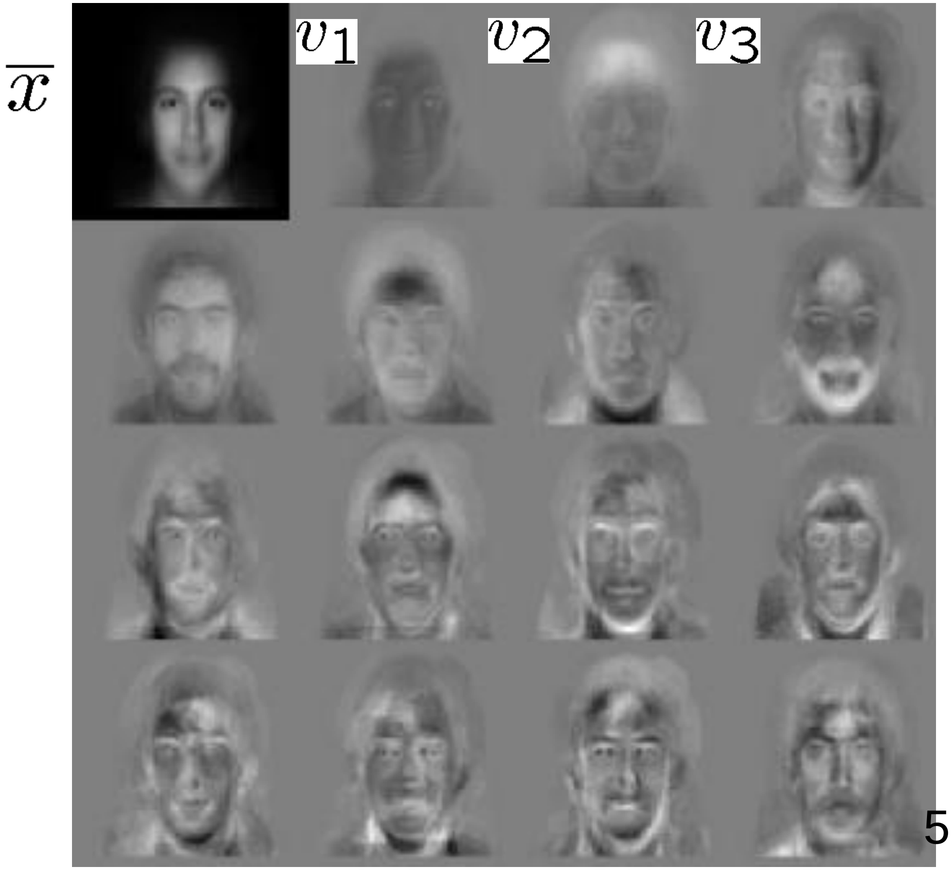
The set of faces is a "subspace" of the set of images

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

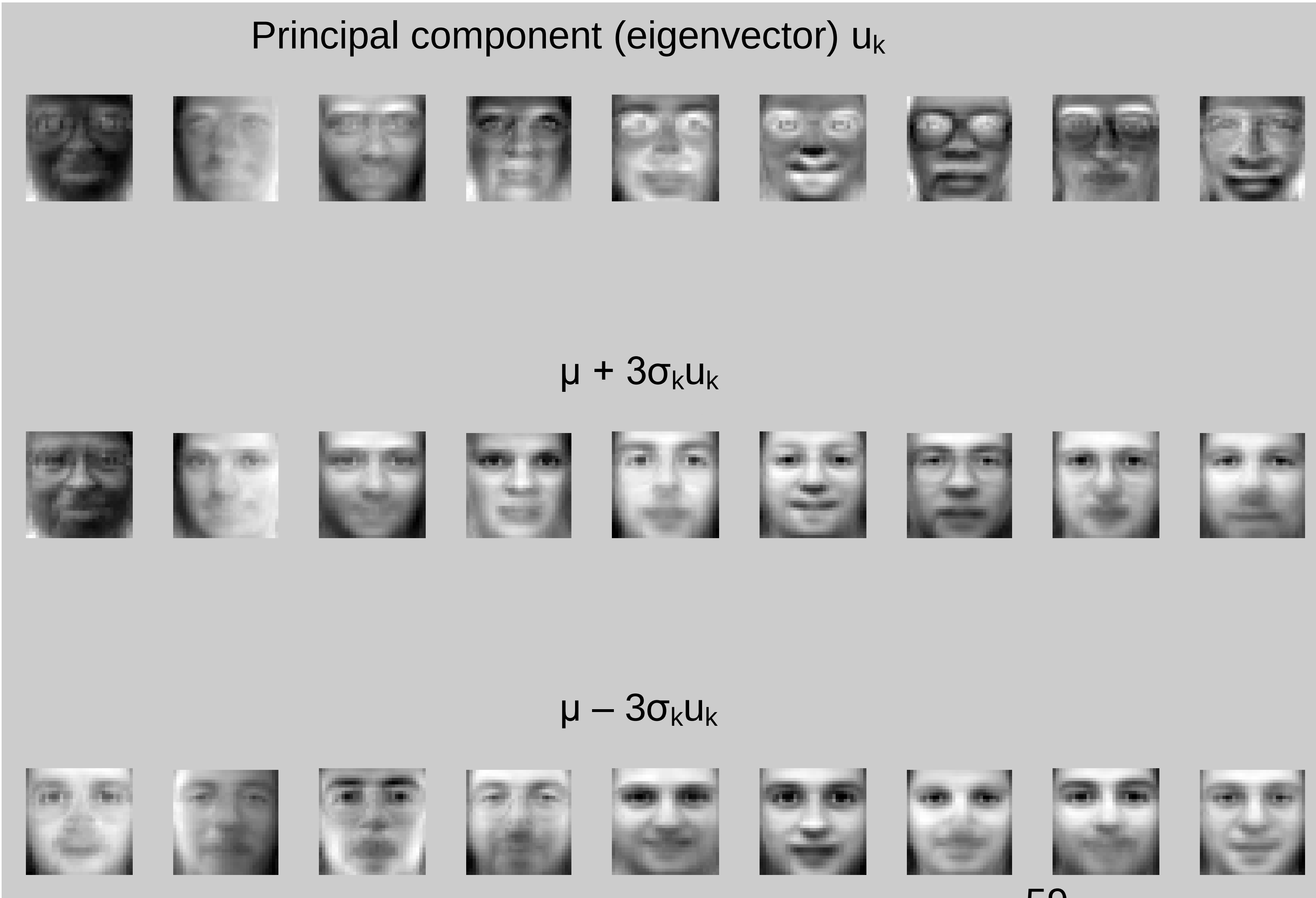
Eigenfaces

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

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

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

$\mathbf{x}$

$a_1\mathbf{v}_1$ $a_2\mathbf{v}_2$ $a_3\mathbf{v}_3$ $a_4\mathbf{v}_4$ $a_5\mathbf{v}_5$ $a_6\mathbf{v}_6$ $a_7\mathbf{v}_7$ $a_8\mathbf{v}_8$

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

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

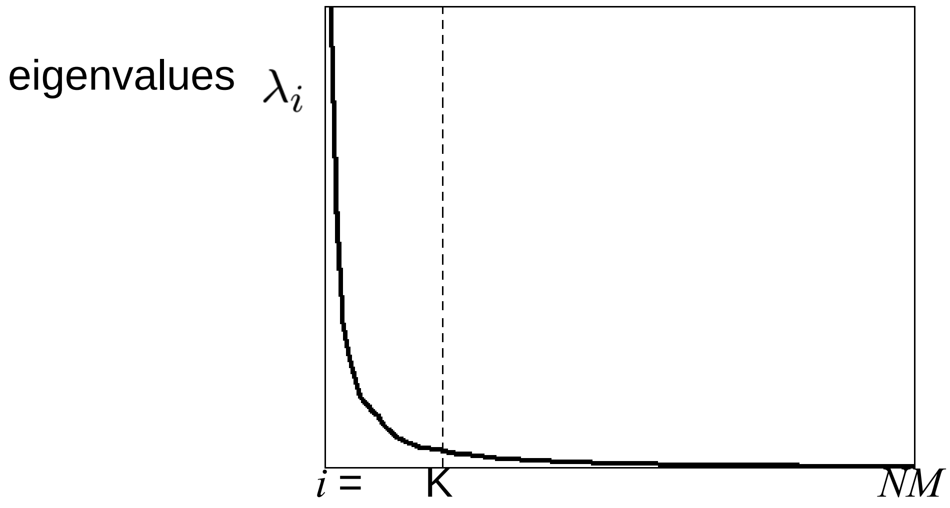
3. Detect if x is a face

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

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 oppos

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

shape

color

both

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.

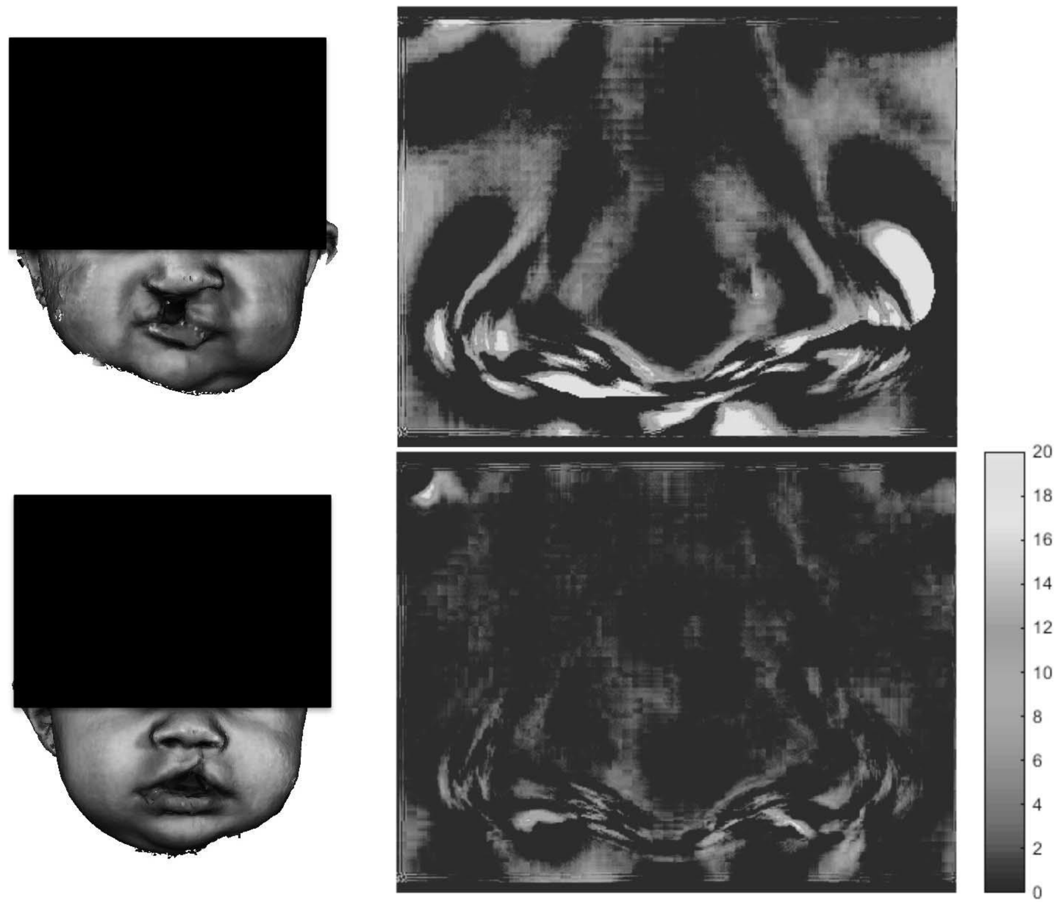
We construct their principal components.

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



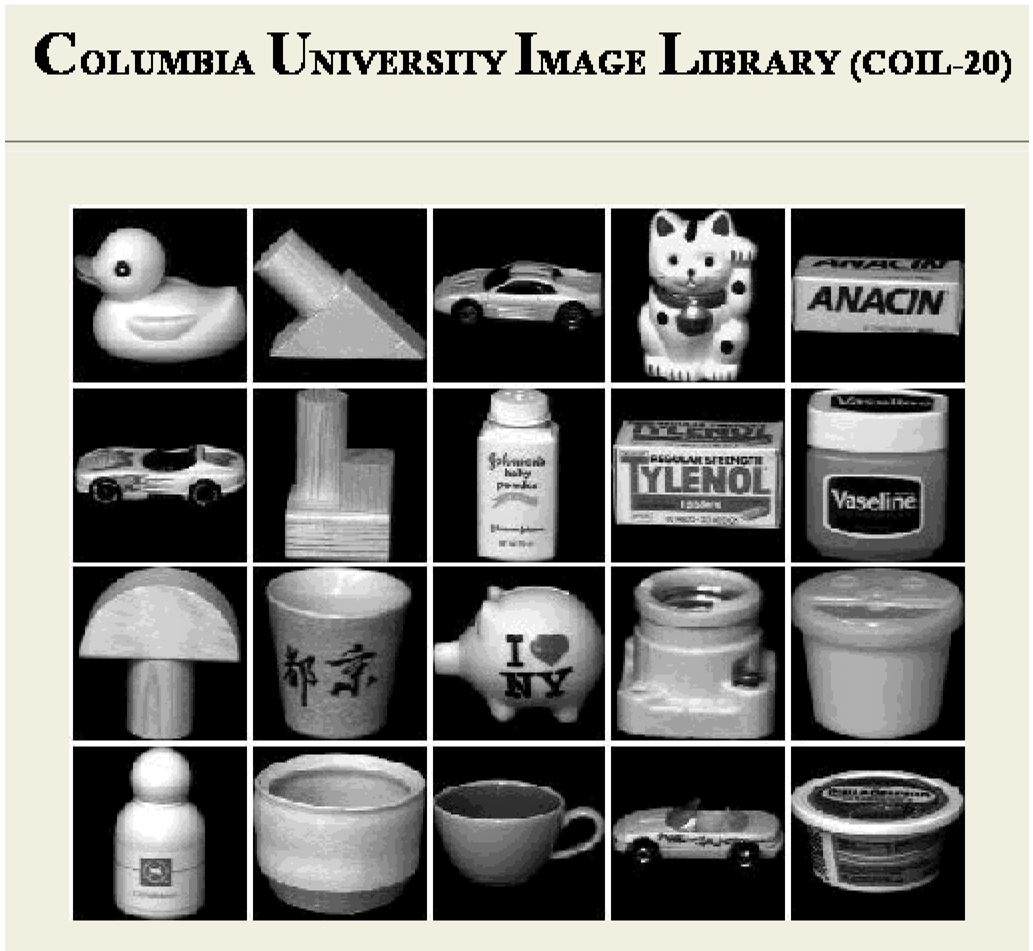
Aligned head mesh

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.