Face Recognition and Detection

The "Margaret Thatcher Illusion", by Peter Thompson

Computer Vision

CSE576, Spring 2008

Richard Szeliski

Recognition problems

What is it?
- Object and scene recognition

Who is it?
- Identity recognition

Where is it?
- Object detection

What are they doing?
- Activities

All of these are **classification** problems
- Choose one class from a list of possible candidates

What is recognition?

A different taxonomy from [Csurka *et al.* 2006]:
- Recognition
  - Where is *this* particular object?
- Categorization
  - What *kind* of object(s) is(are) present?
- Content-based image retrieval
  - Find me something that looks similar
- Detection
  - Locate *all* instances of a given class

Readings

- C. Bishop, "Neural Networks for Pattern Recognition", Oxford University Press, 1998, Chapter 1.
- Forsyth and Ponce, Chap 22.3 (through 22.3.2--eigenfaces)
- Turk, M. and Pentland, A. *Eigenfaces for recognition*. Journal of Cognitive Neuroscience, 1991
- Viola, P. A. and Jones, M. J. (2004). Robust real-time face detection. *IJCV*, 57(2), 137–154.

Sources

- Steve Seitz, CSE 455/576, previous quarters
- Fei-Fei, Fergus, Torralba, CVPR'2007 course
- Efros, CMU 16-721 Learning in Vision
- Freeman, MIT 6.869 Computer Vision: Learning
- Linda Shapiro, CSE 576, Spring 2007

Today's lecture

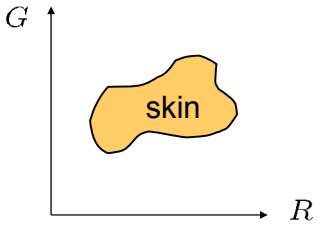
Face recognition and detection

- color-based skin detection
- recognition: eigenfaces [Turk & Pentland] and parts [Moghaddan & Pentland]
- detection: boosting [Viola & Jones]

Face detection

How to tell if a face is present?

Skin detection



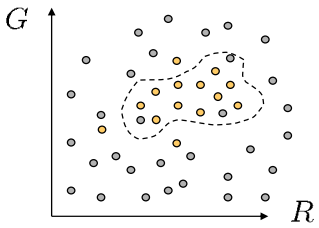
Skin pixels have a distinctive range of colors

- Corresponds to region(s) in RGB color space

Skin classifier

- A pixel X = (R,G,B) is skin if it is in the skin (color) region
- How to find this region?

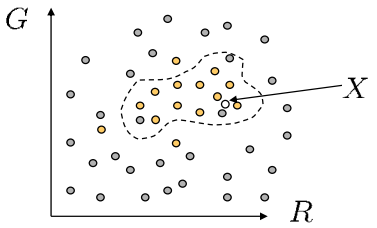
Skin detection



Learn the skin region from examples

- Manually label skin/non pixels in one or more "training images"
- Plot the training data in RGB space
  - skin pixels shown in orange, non-skin pixels shown in gray
  - some skin pixels may be outside the region, non-skin pixels inside.

Skin classifier

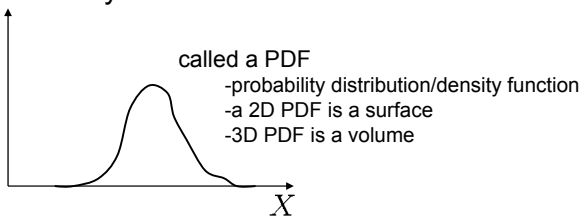


Given X = (R,G,B): how to determine if it is skin or not?

- Nearest neighbor
  - find labeled pixel closest to X
- Find plane/curve that separates the two classes
  - popular approach: Support Vector Machines (SVM)
- Data modeling
  - fit a probability density/distribution model to each class

Probability

- X is a random variable
- P(X) is the probability that X achieves a certain value

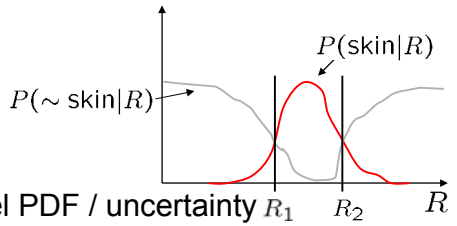


$0 \leq P(X) \leq 1$

$\int_{-\infty}^{\infty} P(X)dX = 1$ continuous X

$\sum P(X) = 1$ discrete X

Probabilistic skin classification



Model PDF / uncertainty

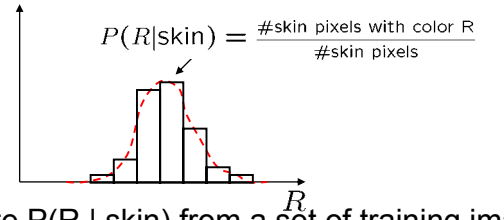
- Each pixel has a probability of being skin or not skin
  $P(\sim \text{skin}|R) = 1 - P(\text{skin}|R)$

Skin classifier

- Given X = (R,G,B): how to determine if it is skin or not?
- Choose interpretation of highest probability

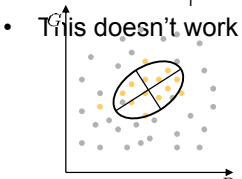
Where do we get $P(\text{skin}|R)$ and $P(\sim \text{skin}|R)$?

Learning conditional PDF's



We can calculate P(R | skin) from a set of training images

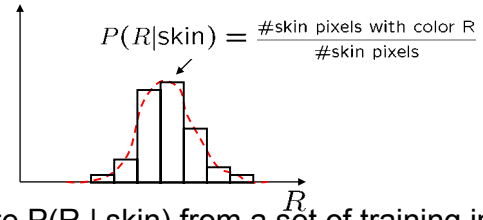
- It is simply a histogram over the pixels in the training images
  - each bin R_i contains the proportion of skin pixels with color R_i
- This doesn't work as well in higher-dimensional spaces. Why not?



Approach: fit parametric PDF functions

- common choice is rotated Gaussian
  - center $\mathbf{c} = \overline{X}$
  - covariance $\sum (X - \overline{X})(X - \overline{X})^T$

Learning conditional PDF's



We can calculate P(R | skin) from a set of training images

But this isn't quite what we want

- Why not? How to determine if a pixel is skin?
- We want P(skin | R) not P(R | skin)
- How can we get it?

Bayes rule

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

what we measure (**likelihood**) — domain knowledge (**prior**)

In terms of our problem:

$$P(\text{skin}|R) = \frac{P(R|\text{skin})\ P(\text{skin})}{P(R)}$$

what we want (**posterior**)

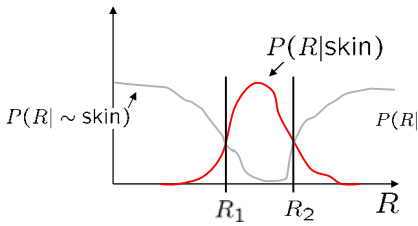
normalization term

$$P(R) = P(R|\text{skin})P(\text{skin}) + P(R|\sim\text{skin})P(\sim\text{skin})$$

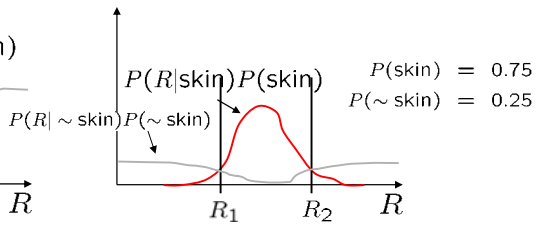
What can we use for the prior P(skin)?

- Domain knowledge:
  - P(skin) may be larger if we know the image contains a person
  - For a portrait, P(skin) may be higher for pixels in the center
- Learn the prior from the training set. How?
  - P(skin) is proportion of skin pixels in training set

Bayesian estimation



likelihood



posterior (unnormalized)

Bayesian estimation

- Goal is to choose the label (skin or ~skin) that maximizes the posterior ↔ minimizes probability of misclassification
  - this is called **Maximum A Posteriori (MAP) estimation**

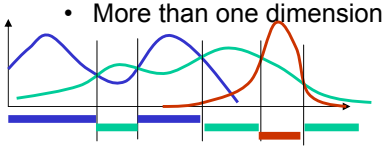
Skin detection results

Figure 25.3. The figure shows a variety of images together with the output of the skin detector of Jones and Rehg applied to the image. Pixels marked black are skin pixels, and white are background. Notice that this process is relatively effective, and could certainly be used to focus attention on, say, faces and hands. *Figure from "Statistical color models with application to skin detection," M.J. Jones and J. Rehg, Proc. Computer Vision and Pattern Recognition, 1999* © *1999, IEEE*

General classification

This same procedure applies in more general circumstances

- More than two classes
- More than one dimension



Example: face detection

- Here, X is an image region
  - dimension = # pixels
  - each face can be thought of as a point in a high dimensional space

H. Schneiderman, T. Kanade. "A Statistical Method for 3D Object Detection Applied to Faces and Cars". CVPR 2000

Today's lecture

Face recognition and detection

- color-based skin detection
- recognition: eigenfaces [Turk & Pentland] and parts [Moghaddan & Pentland]
- detection: boosting [Viola & Jones]

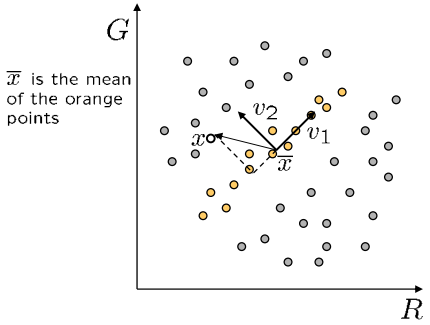
Eigenfaces for recognition

Matthew Turk and Alex Pentland

J. Cognitive Neuroscience

1991

Linear subspaces



convert **x** into $\mathbf{v_1}$, $\mathbf{v_2}$ coordinates

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

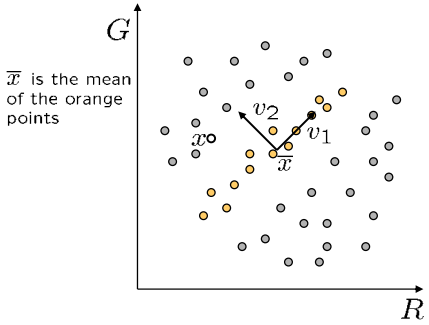
What does the $\mathbf{v_2}$ coordinate measure?
- distance to line
- use it for classification—near 0 for orange pts

What does the $\mathbf{v_1}$ coordinate measure?
- position along line
- use it to specify which orange point it is

Classification can be expensive:
- Big search prob (e.g., nearest neighbors) or store large PDF's

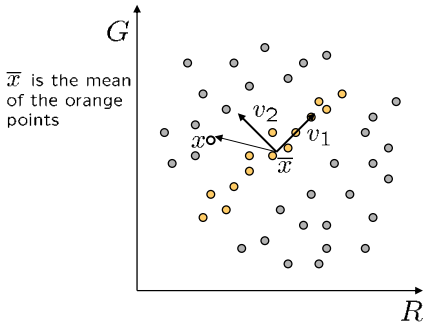
Suppose the data points are arranged as above
- Idea—fit a line, classifier measures distance to line

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

Linear subspaces



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

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

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

$$\mathbf{v_2} = min_{\mathbf{v}}\ \{var(\mathbf{v})\}$$

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

$$\mathbf{v_1} = max_{\mathbf{v}}\ \{var(\mathbf{v})\}$$

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

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

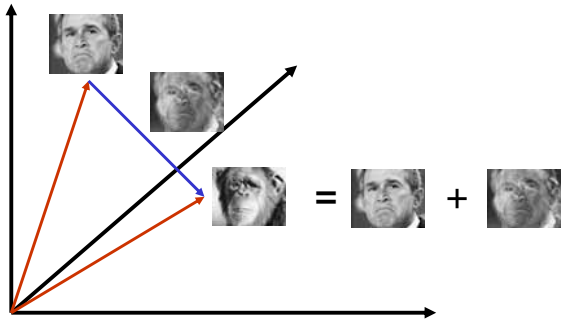
Principal component analysis

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 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***

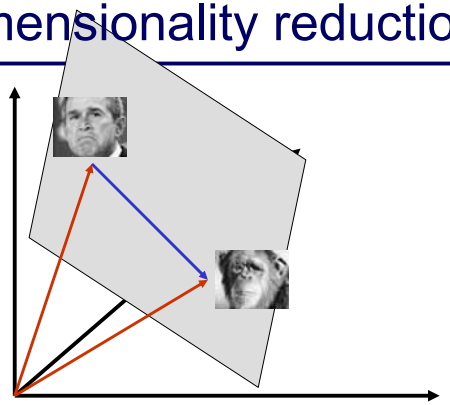
The space of faces



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



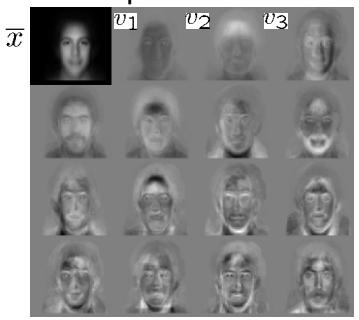
The set of faces is a "subspace" of the set of images

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

Eigenfaces

PCA extracts the eigenvectors of **A**

- Gives a set of vectors $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}, \ldots$
- Each vector is a direction in face space
  - what do these look like?



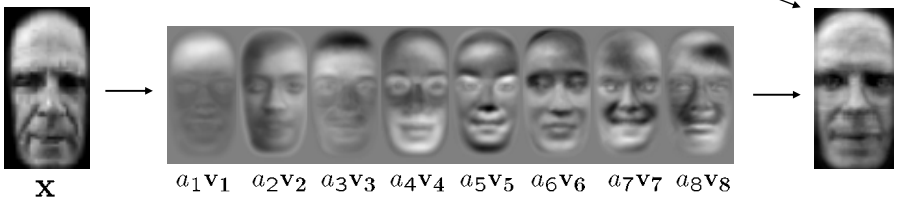
Projecting onto the eigenfaces

The eigenfaces $\mathbf{v_1}, \ldots, \mathbf{v_K}$ span the space of faces

- A face is converted to eigenface coordinates by

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

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



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

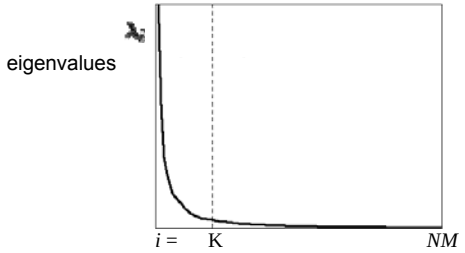
$$\mathbf{x} \rightarrow (a_1, a_2, \ldots, a_K)$$

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

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

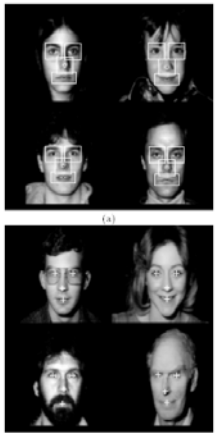
View-Based and Modular Eigenspaces for Face Recognition

Alex Pentland, Baback Moghaddam and Thad Starner
CVPR'94

Part-based eigenfeatures

Learn a separate eigenspace for each face feature

Boosts performance of regular eigenfaces



Bayesian Face Recognition

Baback Moghaddam, Tony Jebara
and Alex Pentland

Pattern Recognition

33(11), 1771-1782, November 2000

(slides from Bill Freeman, MIT 6.869, April 2005)

Bayesian Face Recognition

Intrapersonal Ω_I

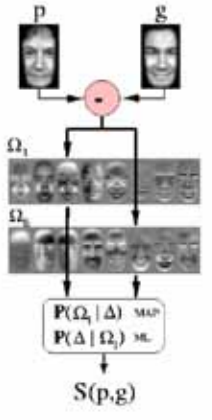
Extrapersonal Ω_E

$$\Omega_I \equiv \{\Delta = x_i - x_j : L(x_i) = L(x_j)\}$$

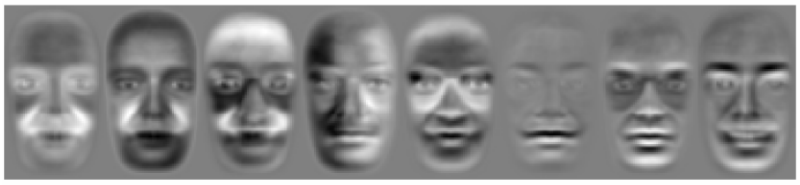
$$\Omega_E \equiv \{\Delta = x_i - x_j : L(x_i) \neq L(x_j)\}$$

$$S = \frac{P(\Delta \mid \Omega_I)P(\Omega_I)}{P(\Delta \mid \Omega_I)P(\Omega_I) + P(\Delta \mid \Omega_E)P(\Omega_E)}$$

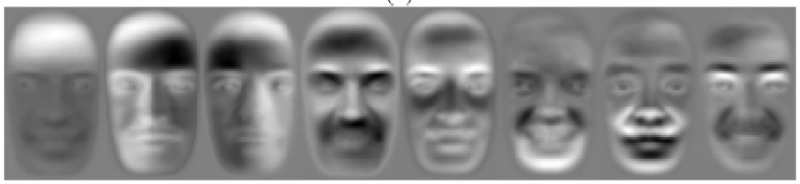
$P(\Delta \mid \Omega)$ ⟶ [Moghaddam ICCV'95]



Bayesian Face Recognition



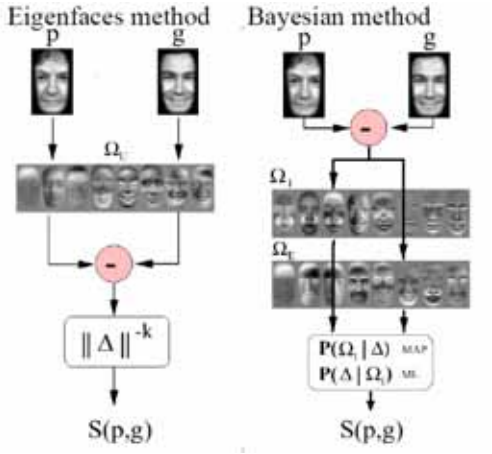
(a)



(b)

Figure 6: "Dual" Eigenfaces: (a) Intrapersonal, (b) Extrapersonal

Bayesian Face Recognition

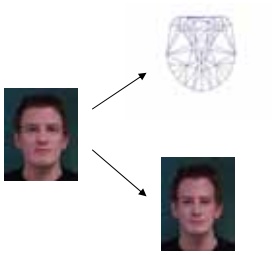


Morphable Face Models

Rowland and Perrett ’95
Lanitis, Cootes, and Taylor ’95, ’97
Blanz and Vetter ’99
Matthews and Baker ’04, ‘07

Morphable Face Model

Use subspace to model elastic 2D or 3D *shape* variation (vertex positions), in addition to *appearance* variation

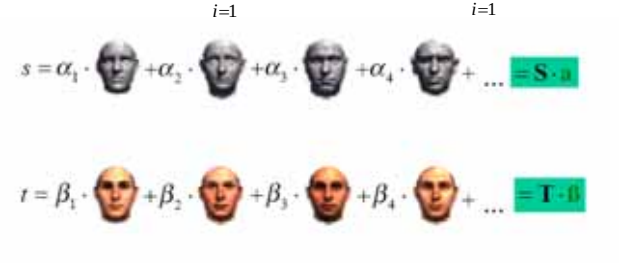


Shape S

Appearance T

Morphable Face Model

$$\mathbf{S}_{model} = \sum_{i=1}^{m} a_i \mathbf{S}_i \qquad \mathbf{T}_{model} = \sum_{i=1}^{m} b_i \mathbf{T}_i$$



3D models from Blanz and Vetter ‘99

Face Recognition Resources

Face Recognition Home Page:

- http://www.cs.rug.nl/~peterkr/FACE/face.html

PAMI Special Issue on Face & Gesture (July ‘97)

FERET

- http://www.dodcounterdrug.com/facialrecognition/Feret/feret.htm

Face-Recognition Vendor Test (FRVT 2000)

- http://www.dodcounterdrug.com/facialrecognition/FRVT2000/frvt2000.htm

Biometrics Consortium

- http://www.biometrics.org

Today's lecture

Face recognition and detection

- color-based skin detection
- recognition: eigenfaces [Turk & Pentland] and parts [Moghaddan & Pentland]
- detection: boosting [Viola & Jones]

Robust real-time face detection

Paul A. Viola and Michael J. Jones

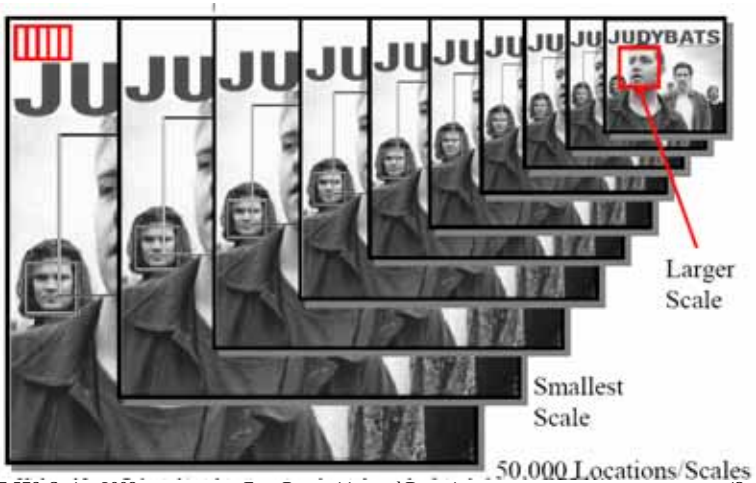
Intl. J. Computer Vision

57(2), 137–154, 2004

(originally in CVPR'2001)

(slides adapted from Bill Freeman, MIT 6.869, April 2005)

Scan classifier over locs. & scales



"Learn" classifier from data

Training Data

- 5000 faces (frontal)
- 10^8 non faces
- Faces are normalized
  - Scale, translation

Many variations

- Across individuals
- Illumination
- Pose (rotation both in plane and out)

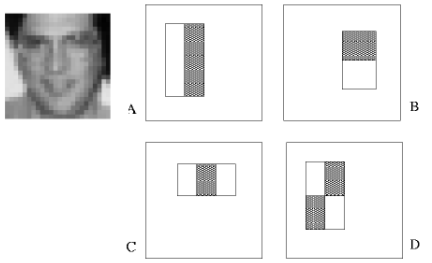
Characteristics of algorithm

- Feature set (…is huge about 16M features)
- Efficient feature selection using AdaBoost
- New image representation: Integral Image
- Cascaded Classifier for rapid detection

➢ Fastest known face detector for gray scale images

Image features

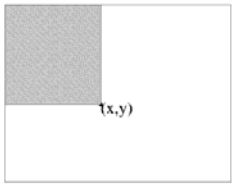
- "Rectangle filters"
  - Similar to Haar wavelet
- Differences between sums of pixels in adjacent rectangles



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

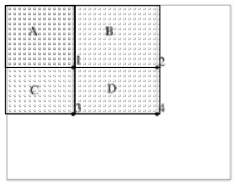
Integral Image

Partial sum $I'(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} I(x',y')$



Any rectangle is

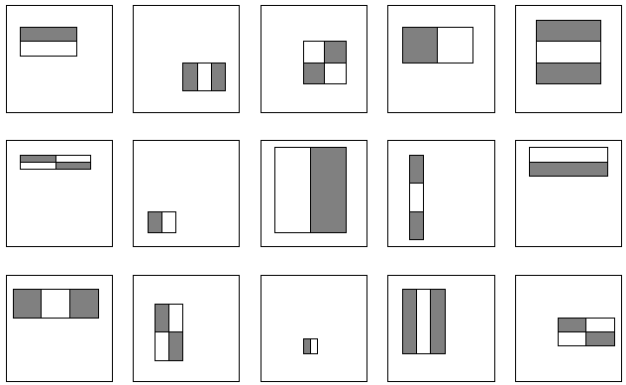
D = 1+4-(2+3)



Also known as:

- *summed area tables* [Crow84]
- *boxlets* [Simard98]

Huge library of filters



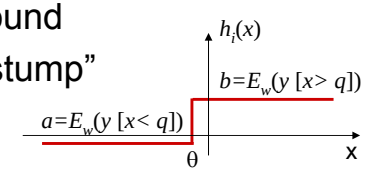
Constructing the classifier

Perceptron yields a sufficiently powerful classifier

$$C(x) = \theta\left(\sum_i \alpha_i h_i(x) + b\right)$$

Use AdaBoost to efficiently choose best features

- add a new $h_i(x)$ at each round
- each $h_i(x_k)$ is a "decision stump"



Constructing the classifier

For each round of boosting:

- Evaluate each rectangle filter on each example
- Sort examples by filter values
- Select best threshold for each filter (min error)
  - Use sorting to quickly scan for optimal threshold
- Select best filter/threshold combination
- Weight is a simple function of error rate
- Reweight examples
  - (There are many tricks to make this more efficient.)

Good reference on boosting

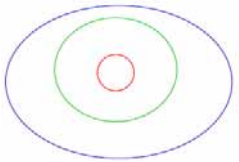
Friedman, J., Hastie, T. and Tibshirani, R.
Additive Logistic Regression: a Statistical View of Boosting

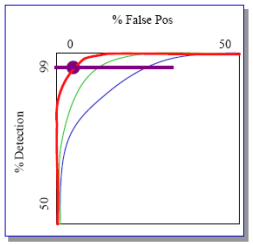
http://www-stat.stanford.edu/~hastie/Papers/boost.ps

"We show that boosting fits an additive logistic regression model by stagewise optimization of a criterion very similar to the log-likelihood, and present likelihood based alternatives. We also propose a multi-logit boosting procedure which appears to have advantages over other methods proposed so far."

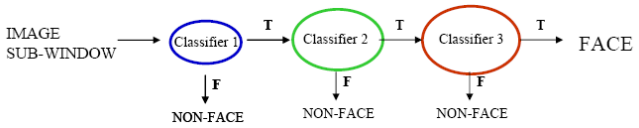
Trading speed for accuracy

Given a nested set of classifier hypothesis classes





Computational Risk Minimization



Speed of face detector (2001)

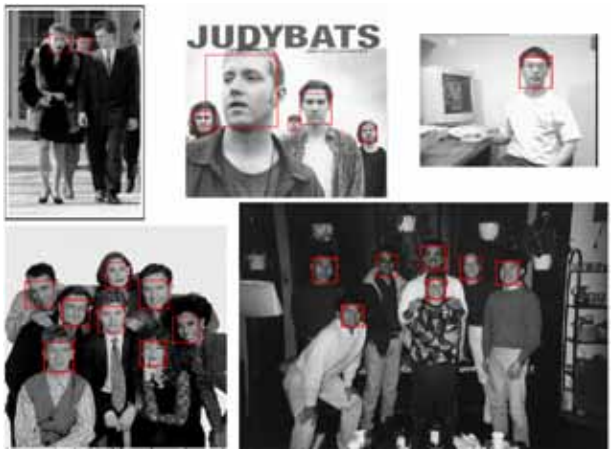
Speed is proportional to the average number of features computed per sub-window.

On the MIT+CMU test set, an average of 9 features (/ 6061) are computed per sub-window.

On a 700 Mhz Pentium III, a 384x288 pixel image takes about 0.067 seconds to process (15 fps).

Roughly 15 times faster than Rowley-Baluja-Kanade and 600 times faster than Schneiderman-Kanade.

Sample results



Summary (Viola-Jones)

- Fastest known face detector for gray images
- Three contributions with broad applicability:
  - ❖Cascaded classifier yields rapid classification
  - ❖AdaBoost as an extremely efficient feature selector
  - ❖Rectangle Features + Integral Image can be used for rapid image analysis

Face detector comparison

Informal study by Andrew Gallagher, CMU, for CMU 16-721 Learning-Based Methods in Vision, Spring 2007

- The Viola Jones algorithm OpenCV implementation was used. (<2 sec per image).
- For Schneiderman and Kanade, Object Detection Using the Statistics of Parts [IJCV'04], the www.pittpatt.com demo was used. (~10-15 seconds per image, including web transmission).

Viola
Jones

Schneiderman
Kanade

Today's lecture

Face recognition and detection

- color-based skin detection
- recognition: eigenfaces [Turk & Pentland] and parts [Moghaddan & Pentland]
- detection: boosting [Viola & Jones]

Questions?