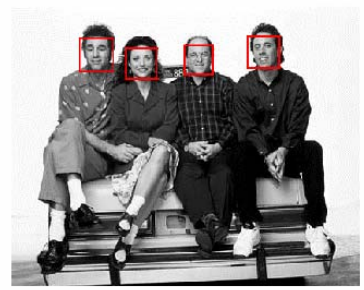
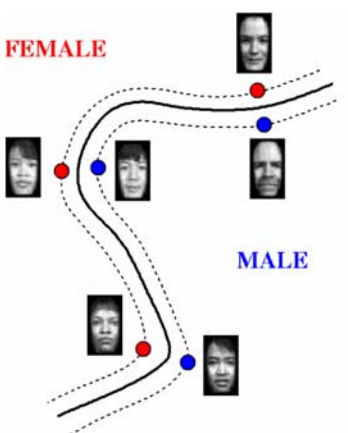


# Lecture 17: Recognition III



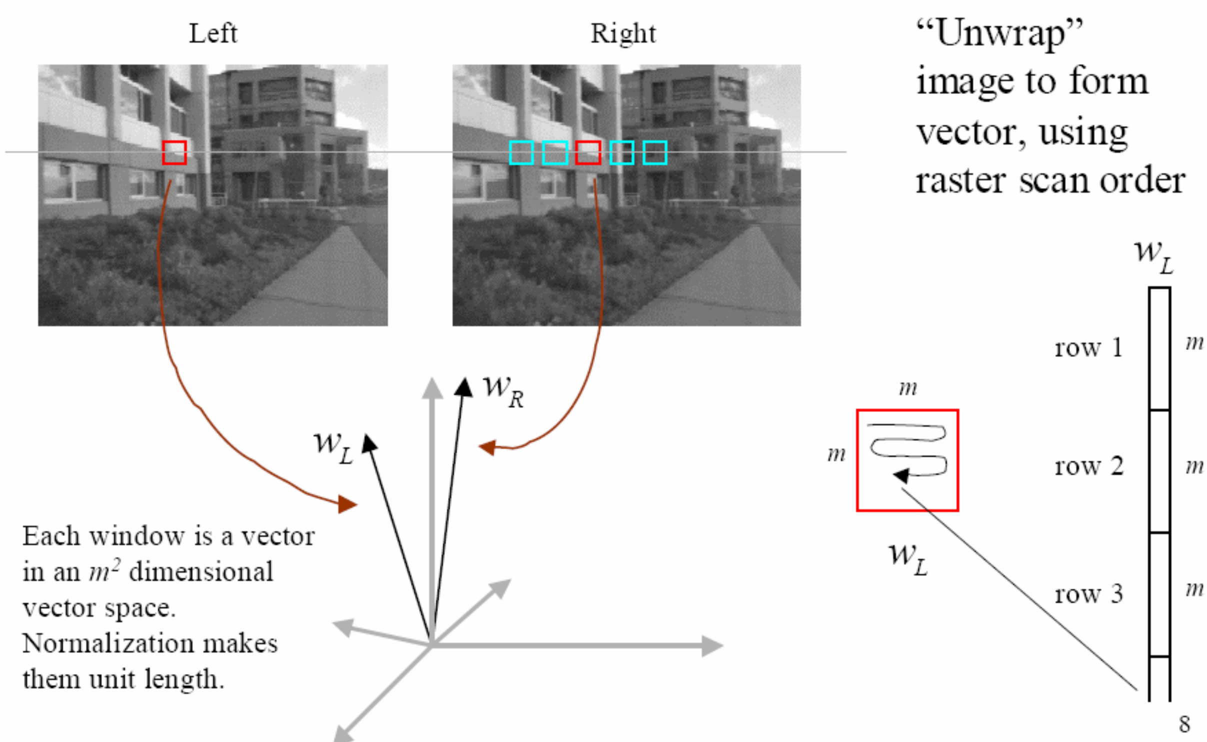

Tuesday, Nov 13

Prof. Kristen Grauman

# Outline

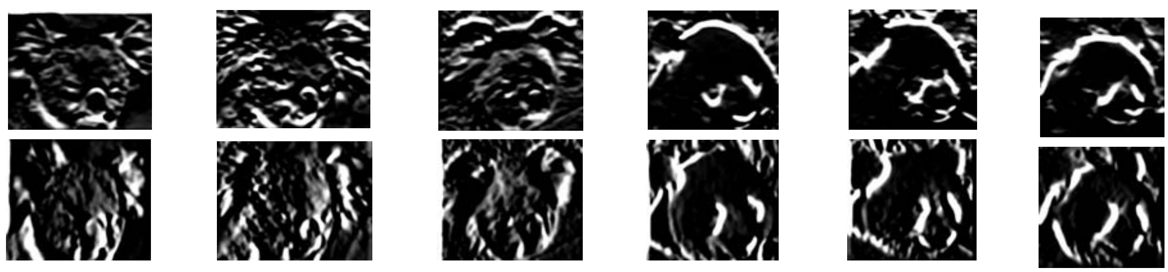
- Last time:
  - Model-based recognition wrap-up
  - Classifiers: templates and appearance models
    - Histogram-based classifier
    - Eigenface approach, nearest neighbors
- Today:
  - Limitations of Eigenfaces, PCA
  - Discriminative classifiers
    - Viola & Jones face detector (boosting)
    - SVMs

# Images (patches) as vectors



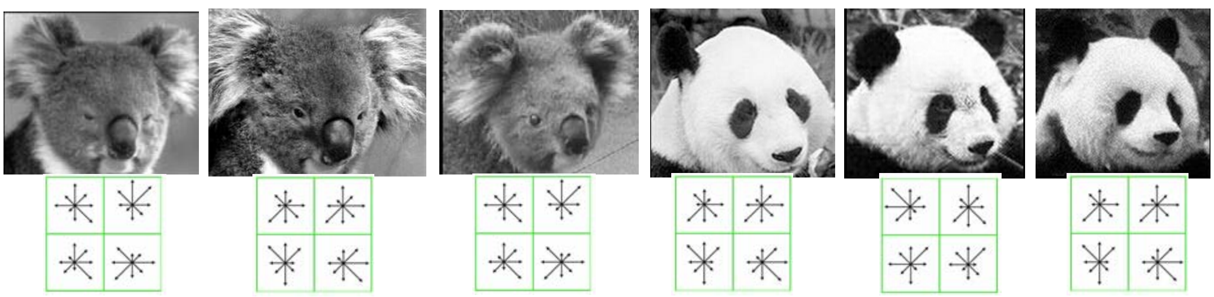

Slide by Trevor Darrell, MIT

# Other image features



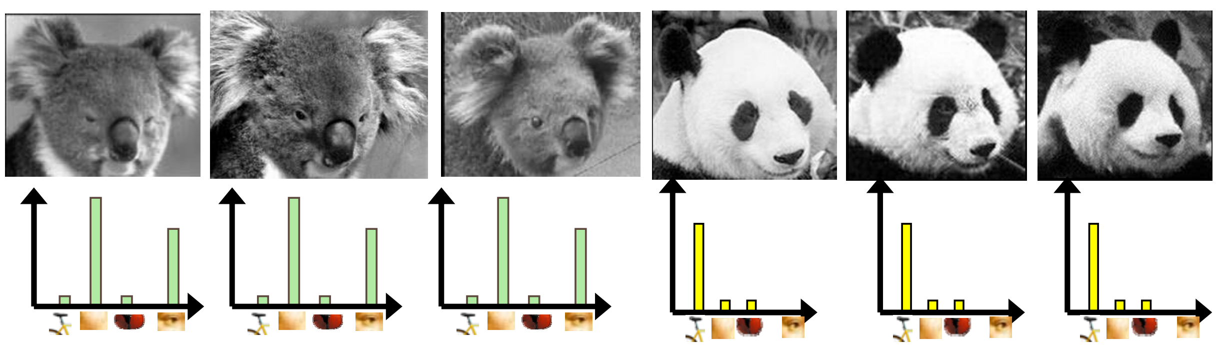
– vector of pixel intensities
– grayscale / color histogram
– bank of filter responses

# Other image features



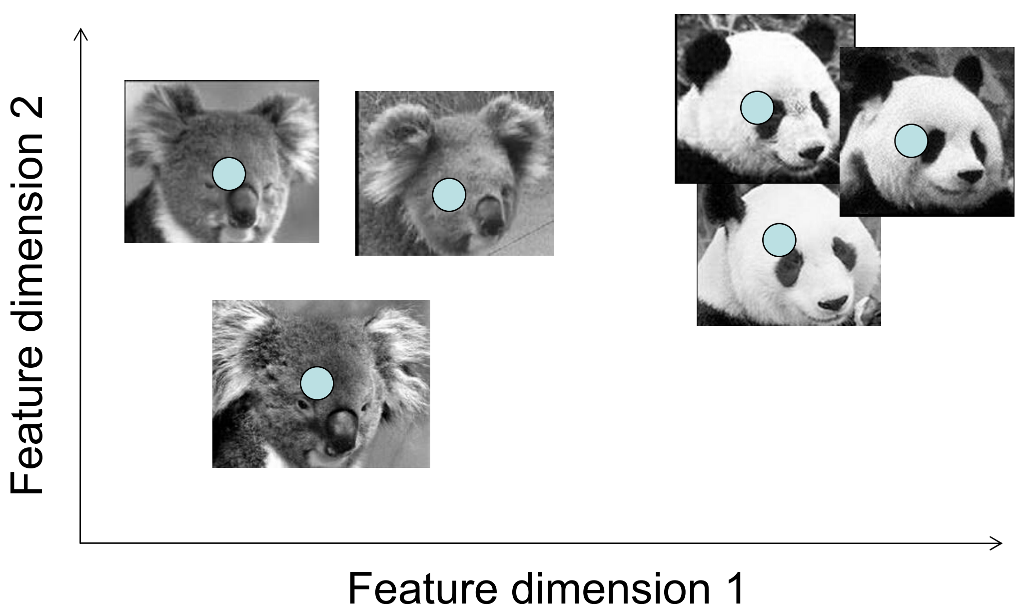
– vector of pixel intensities
– grayscale / color histogram
– bank of filter responses
– SIFT descriptor

# Other image features



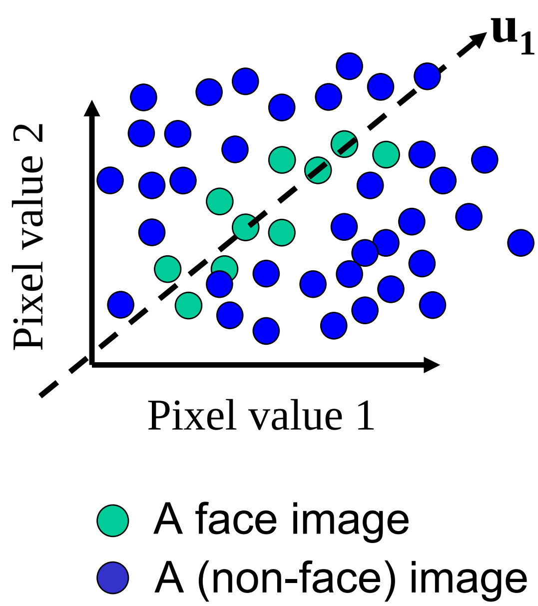
– vector of pixel intensities
– grayscale / color histogram
– bank of filter responses
– SIFT descriptor
– bag of words…

# Feature space / Representation



# Last time: Eigenfaces

- Construct lower dimensional linear subspace that best explains variation of the training examples



# Last time: Eigenfaces

- Premise: set of faces lie in a subspace of set of all images
- Use PCA to determine the $k$ ($k<d$) vectors $\mathbf{u}_1,\ldots\mathbf{u}_k$ that span that subspace:

  $$\mathbf{x} =\sim \boldsymbol{\mu} + w_1\mathbf{u}_1 + \ldots + w_k\mathbf{u}_k$$

- Then use nearest neighbors in “face space” coordinates $(w_1,\ldots w_k)$ to do recognition

d = num rows * num cols in training images

# Last time: Eigenfaces

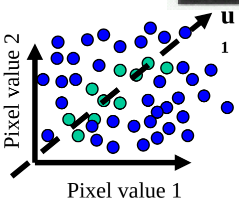
Training images:

$\mathbf{x}_1,\ldots,\mathbf{x}_N$

# Last time: Eigenfaces

Top eigenvectors of the covariance matrix: $\mathbf{u}_1, \ldots \mathbf{u}_k$

Mean: μ



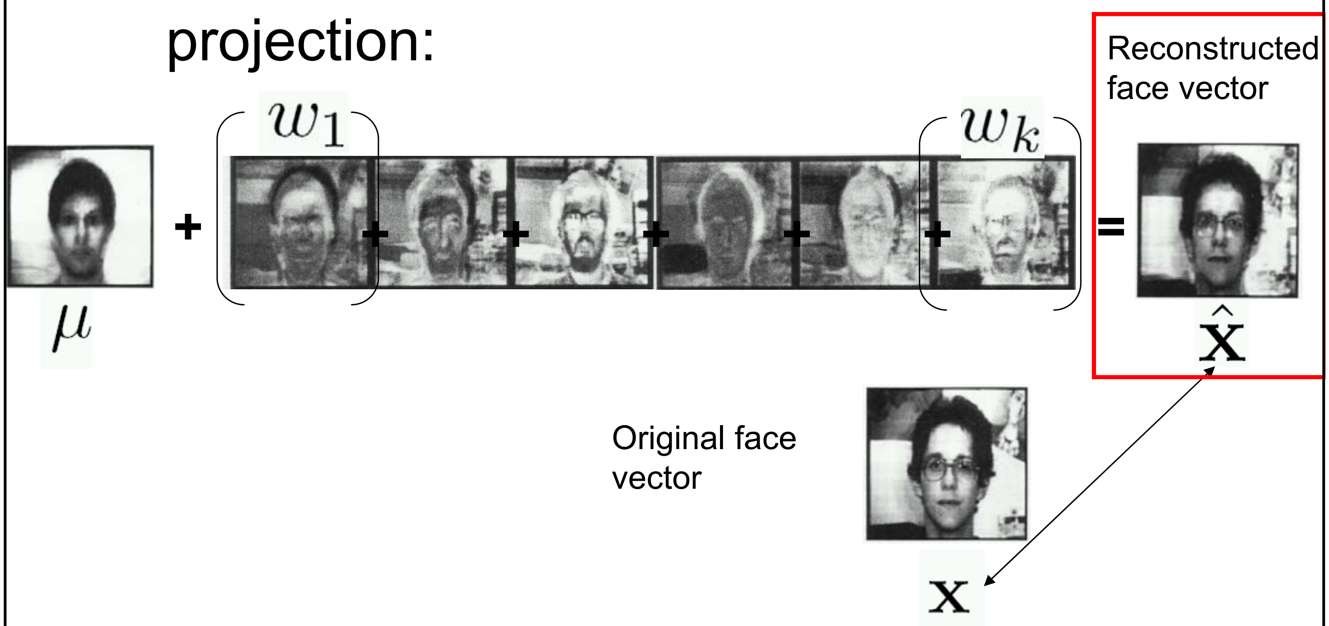
# Last time: Eigenfaces

Face **x** in “face space” coordinates [$w_1$,…,$w_k$]: project the vector of pixel intensities onto each eigenvector.

$$\mathbf{x} \rightarrow \underbrace{[\mathbf{u}_1^{\mathrm{T}}(\mathbf{x}-\mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x}-\mu)]}_{w_1, \ldots, w_k}$$

# Last time: Eigenfaces

Reconstruction from low-dimensional projection:



# Last time: Eigenface recognition

- Process labeled training images:
  - Unwrap the training face images into vectors to form a matrix
  - Perform principal components analysis (PCA): compute eigenvalues and eigenvectors of the covariance matrix
  - Project each training image onto subspace
- Given novel image:
  - Project onto subspace
  - If $||\hat{\mathbf{x}} - \mathbf{x}|| > \theta$

    Unknown, not face
  - Else

    Classify as closest training face in k-dimensional subspace

# Benefits

- Form of automatic feature selection
- Can sometimes remove lighting variations
- Computational efficiency:
  - Reducing storage from d to k
  - Distances computed in k dimensions

# Limitations

- PCA useful to *represent* data, but directions of most variance not necessarily useful for classification

# Alternative: Fisherfaces

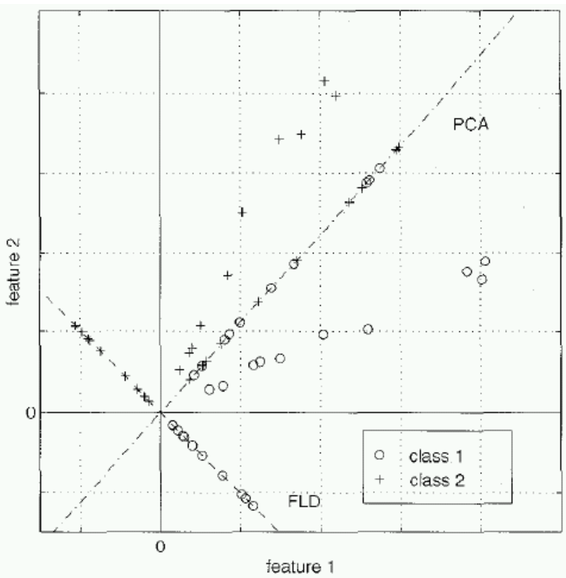


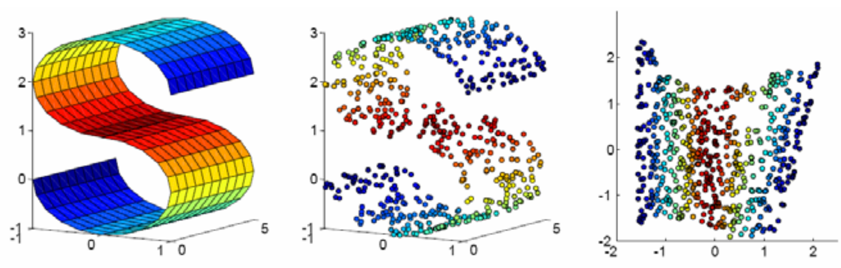

Fig. 2. A comparison of principal component analysis (PCA) and Fisher's linear discriminant (FLD) for a two class problem where data for each class lies near a linear subspace.

Belhumeur et al. PAMI 1997

Rather than maximize scatter of projected classes as in PCA, maximize ratio of between-class scatter to within-class scatter by using Fisher's Linear Discriminant

# Limitations

- PCA useful to *represent* data, but directions of most variance not necessarily useful for classification
- Not appropriate for all data: PCA is fitting Gaussian where Σ is covariance matrix



There may be non-linear structure in high-dimensional data.
Figure from Saul & Roweis

# Limitations

- PCA useful to *represent* data, but directions of most variance not necessarily useful for classification
- Not appropriate for all data: PCA is fitting Gaussian where $\Sigma$ is covariance matrix
- Assumptions about pre-processing may be unrealistic, or demands good detector

# Prototype faces

- Mean face as average of intensities: ok for well-aligned images…

Mean: $\mu$

# Prototype faces

…but unaligned shapes are a problem.

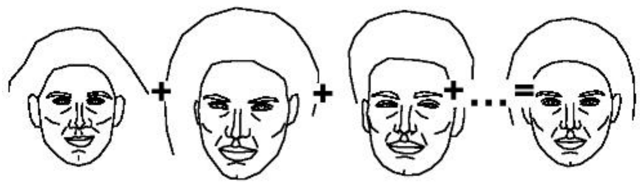
We must include appearance AND shape to construct a prototype.

# Prototype faces in shape *and* appearance

1

Mark coordinates of standard features

2



Compute average *shape* for a group of faces

3

Warp faces to mean shape. Blend images to provide image with average appearance of the group, normalized for shape.

Compare to faces that are blended without changing shape.

University of St. Andrews, Perception Laboratory Figures from http://perception.st-and.ac.uk/Prototyping/prototyping.htm

# Using prototype faces: aging

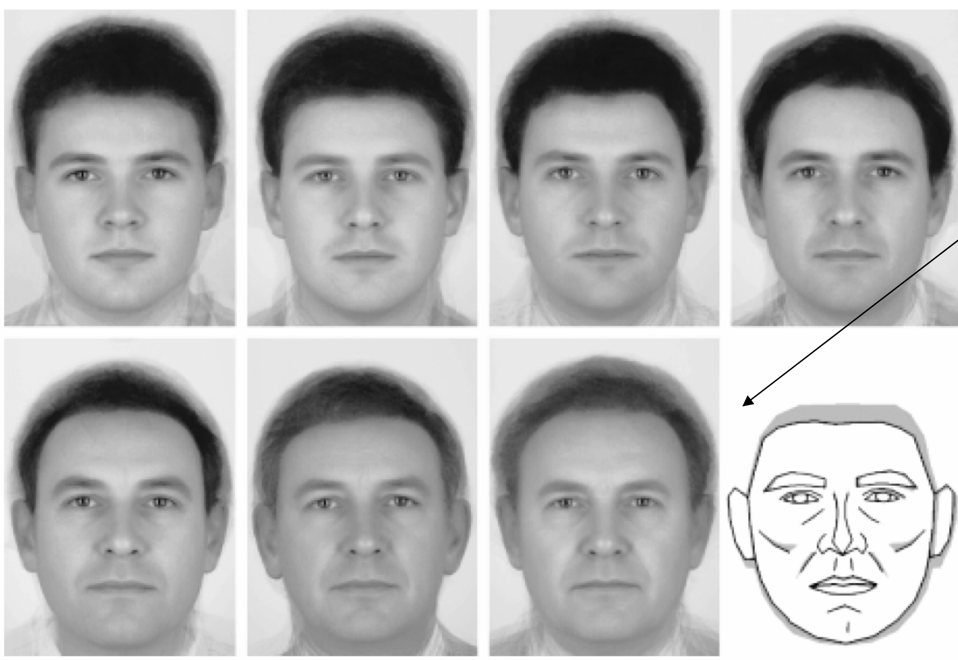


Figure 1. Face blends. Each blend contains the average colour and shape information from individual faces within the same 5 year age bracket *(a)* 20-24, *(b)* 25-29, *(c)* 30-34, *(d)* 35-39, *(e)* 40-44, *(f)* 45-49 and *(g)* 50-54. *(h)* The shaded area illustrates the difference between average face shapes of the 25-29 (dark lines) and 50-54 age brackets.

Average appearance and shape for different age groups.

Shape differences for 25-29 yr olds and 50-54 yr olds

*Burt D.M. & Perrett D.I. (1995) Perception of age in adult Caucasian male faces: computer graphic manipulation of shape and colour information. Proc. R. Soc. 259, 137-143.*

# Using prototype faces: aging

Enhance their differences to
form caricature

Caricature

Figure 2. Enhancing colour cues to age. *(a)* Image caricaturing colour differences between *(b)* the blend of 50-54 year old male faces and a shape matched blend of all age groups (age 20-54). *(c)* Contrast and colour enhanced image made by amplifying RGB pixel differences between original blend and a uniform grey image.

*Burt D.M. & Perrett D.I. (1995) Perception of age in adult Caucasian male faces: computer graphic manipulation of shape and colour information. Proc. R. Soc. 259, 137-143.*

# Using prototype faces: aging

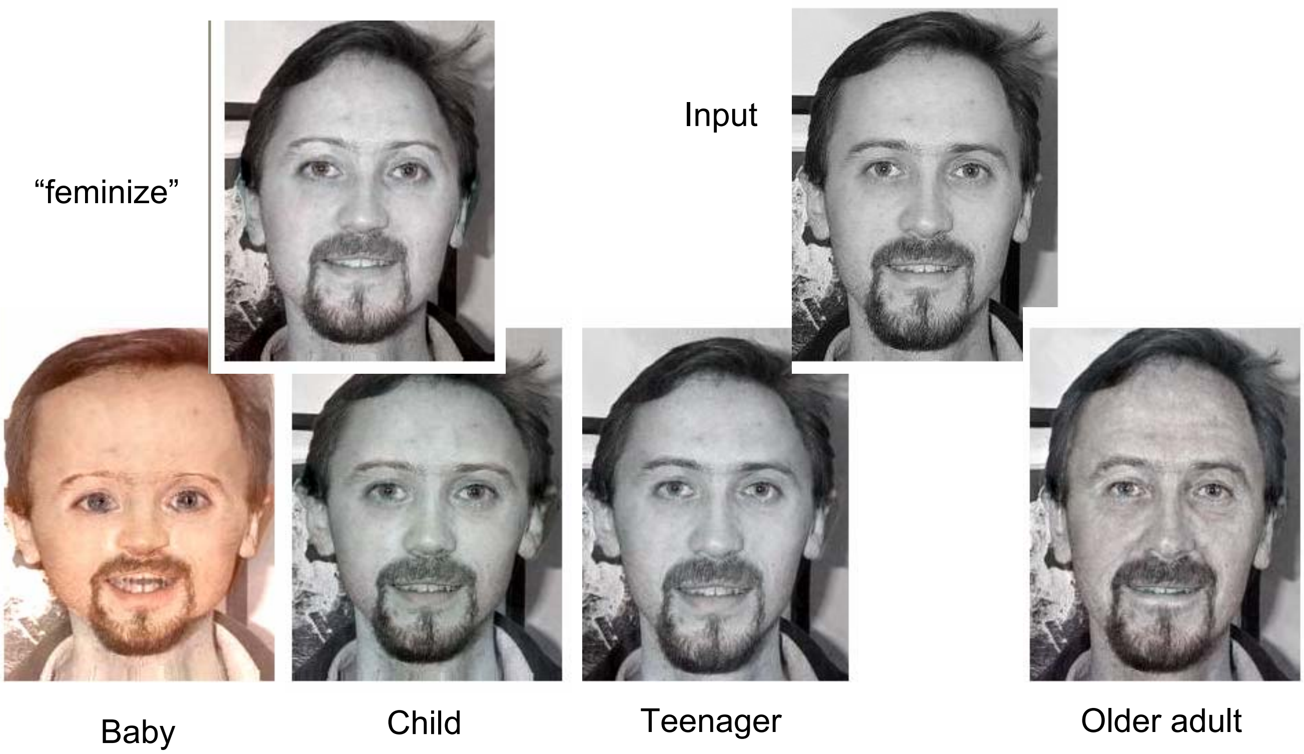
"Facial aging": get facial prototypes from different age groups, consider the difference to get function that maps one age group to another.

University of St. Andrews, Perception Laboratory



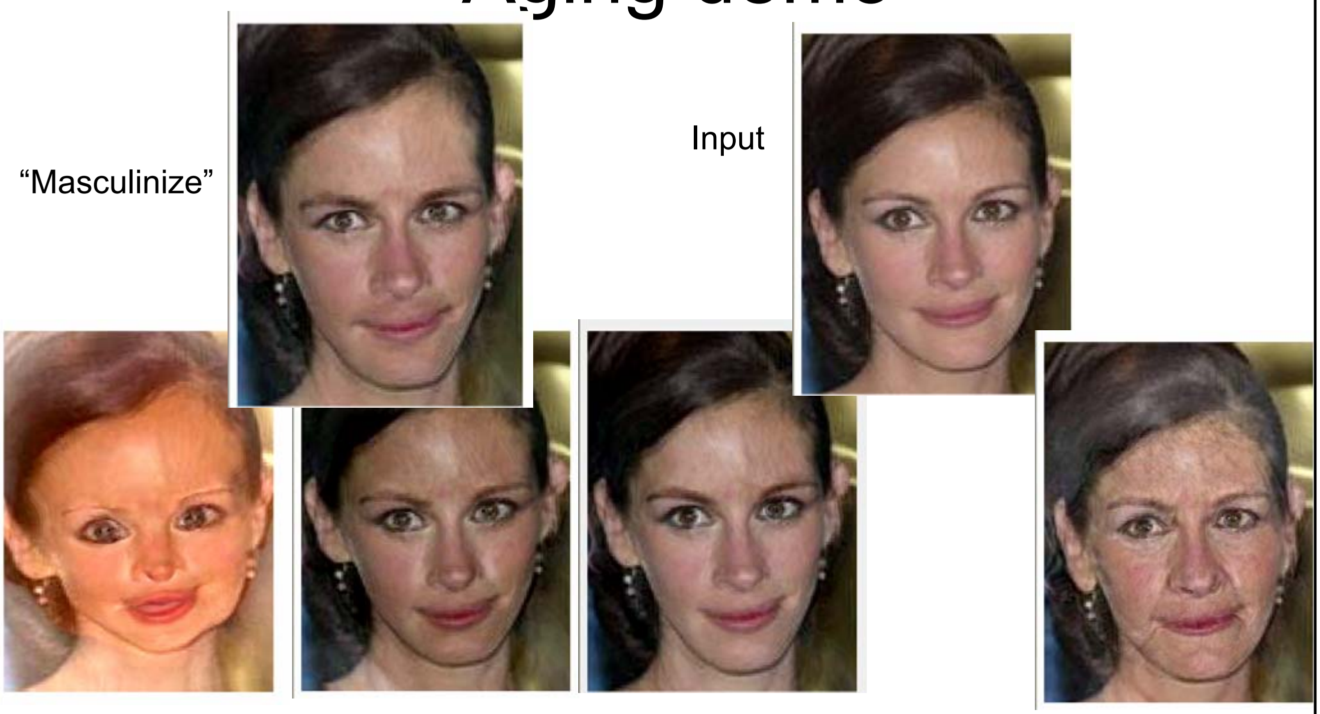
*Burt D.M. & Perrett D.I. (1995) Perception of age in adult Caucasian male faces: computer graphic manipulation of shape and colour information. Proc. R. Soc. 259, 137-143.*

# Aging demo




- **http://morph.cs.st-andrews.ac.uk//Transformer/**

# Aging demo



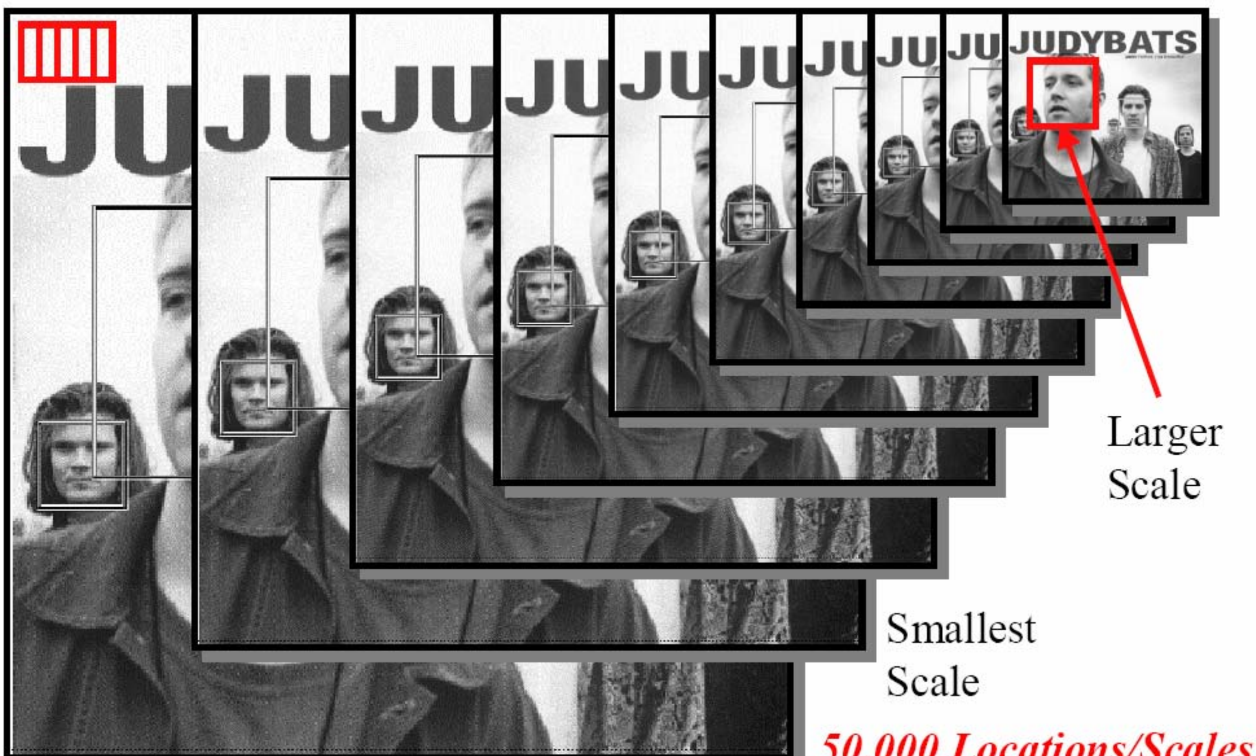

Baby Child Teenager Older adult

- **http://morph.cs.st-andrews.ac.uk//Transformer/**

# Outline

- Last time:
  - Model-based recognition wrap-up
  - Classifiers: templates and appearance models
    - Histogram-based classifier
    - Eigenface approach, nearest neighbors
- Today:
  - Limitations of Eigenfaces, PCA
  - Discriminative classifiers
    - Viola & Jones face detector (boosting)
    - SVMs

# The Classical Face Detection Process

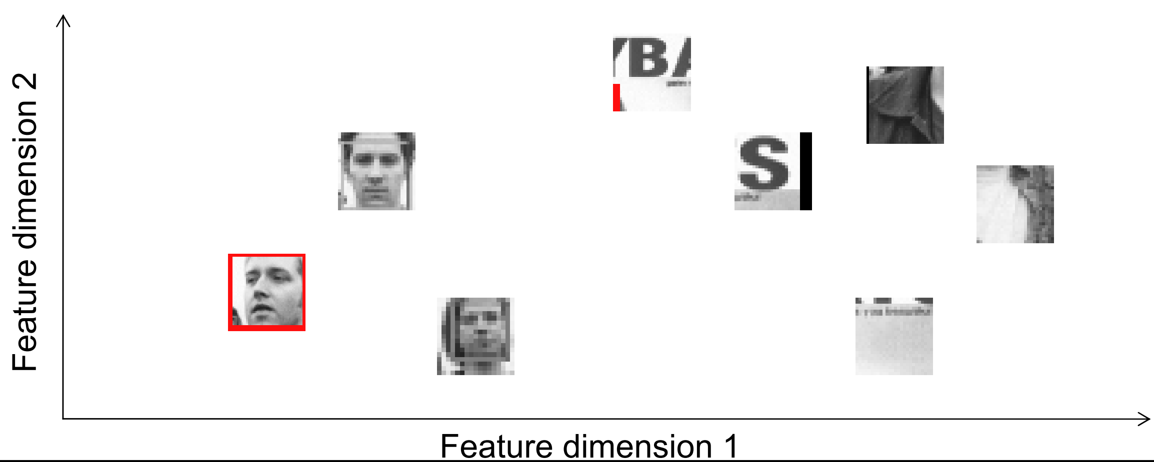





Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

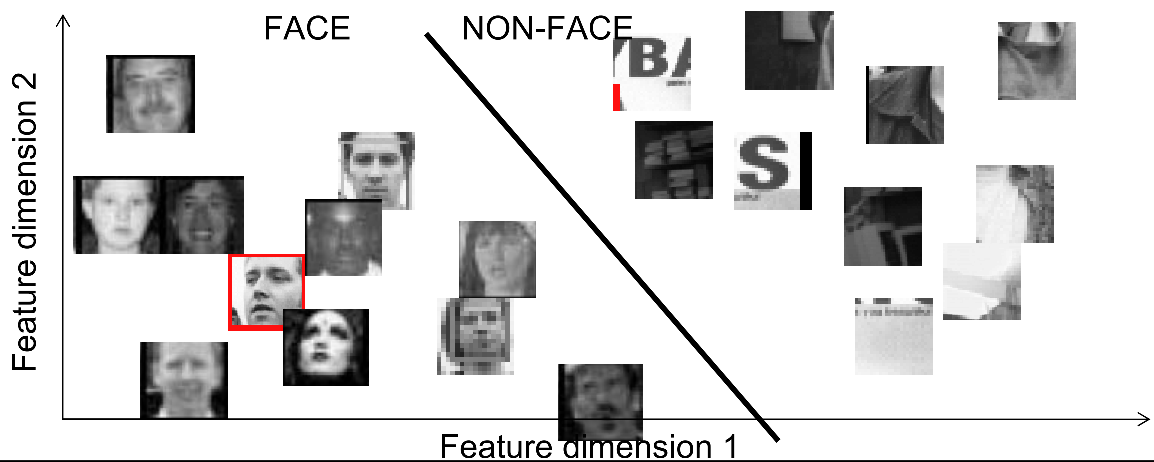
# Learning to distinguish faces and "non-faces"

- How should the decision be made at every sub-window?



# Learning to distinguish faces and "non-faces"

- How should the decision be made at every sub-window?
- Compute boundary that divides the training examples well…



# Questions

- How to discriminate faces and non-faces?
  - Representation choice
  - Classifier choice
- How to deal with the expense of such a windowed scan?
  - Efficient feature computation
  - Limit amount of computation required to make a decision per window

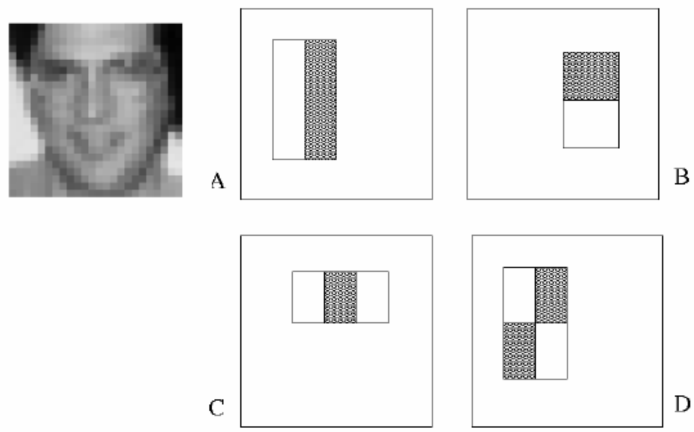
# *Rapid Object Detection Using a Boosted Cascade of Simple Features*

Paul Viola Michael J. Jones
Mitsubishi Electric Research Laboratories (MERL)
Cambridge, MA

[CVPR 2001]

# Image Features

"Rectangle filters"



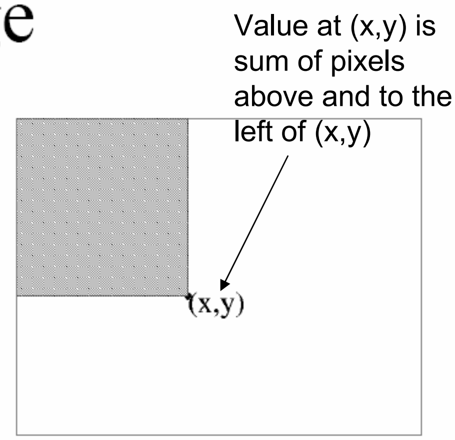

Differences between sums of pixels in adjacent rectangles

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

# Integral Image

- Defined as:

$$ii(x, y) = \sum_{x' \le x, y' \le y} i(x', y')$$




Can be computed in one pass over the original image:

$$s(x, y) = s(x, y-1) + i(x, y)$$
$$ii(x, y) = ii(x-1, y) + s(x, y)$$

# Integral Image

- Defined as:

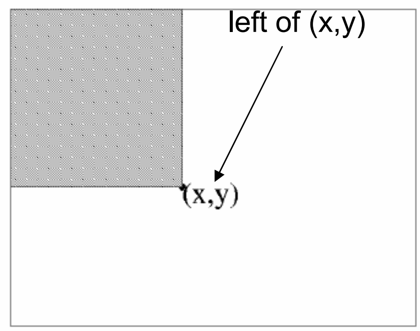
$$ii(x, y) = \sum_{x' \le x, y' \le y} i(x', y')$$

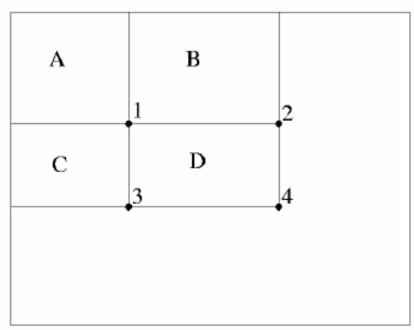
- Any rectangular sum can be computed in constant time:

$$D = 1 + 4 - (2 + 3)$$
$$= A + (A + B + C + D) - (A + C + A + B)$$
$$= D$$

- Rectangle features can be computed as differences between rectangles

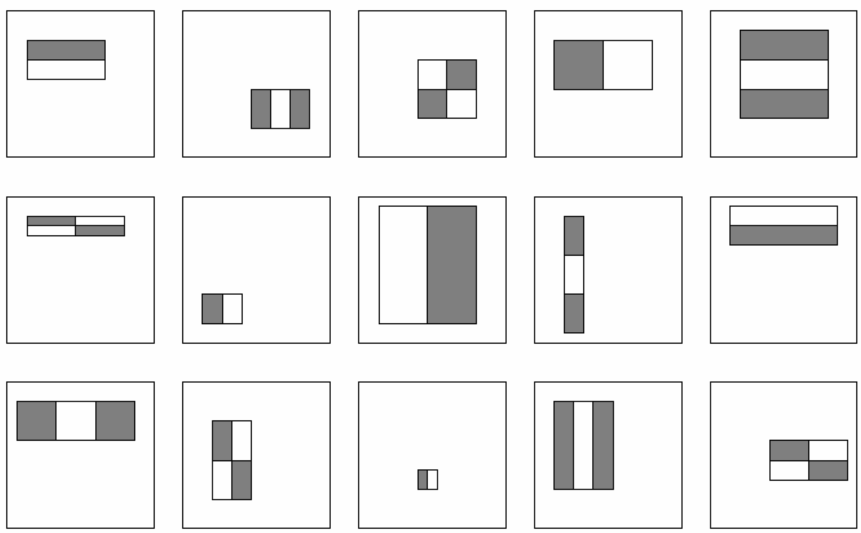
Value at (x,y) is sum of pixels above and to the left of (x,y)




Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

# Large library of filters



180,000+ possible features associated with each image subwindow…efficient, but still can't compute complete set at detection time.

# Boosting

- *Weak learner*: classifier with accuracy that need be only better than chance
  - Binary classification: error < 50%
- Boosting combines multiple weak classifiers to create accurate ensemble
- Can use fast simple classifiers without sacrificing accuracy.

# AdaBoost [Freund & Schapire]: Intuition

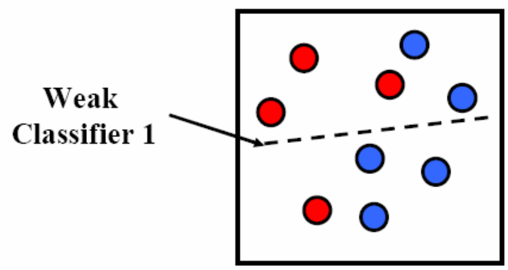



Figure from Freund and Schapire

# AdaBoost [Freund & Schapire]: Intuition

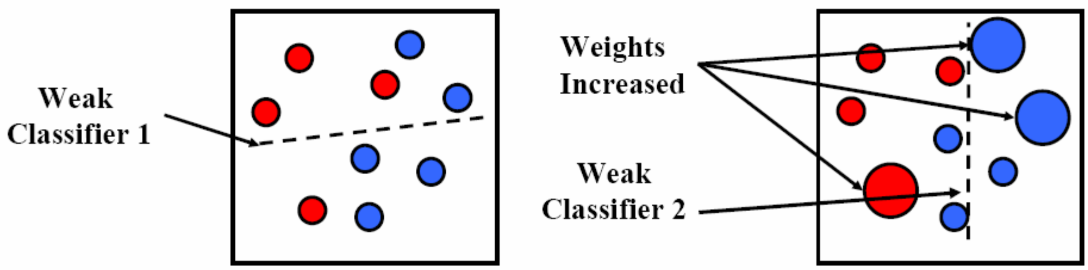
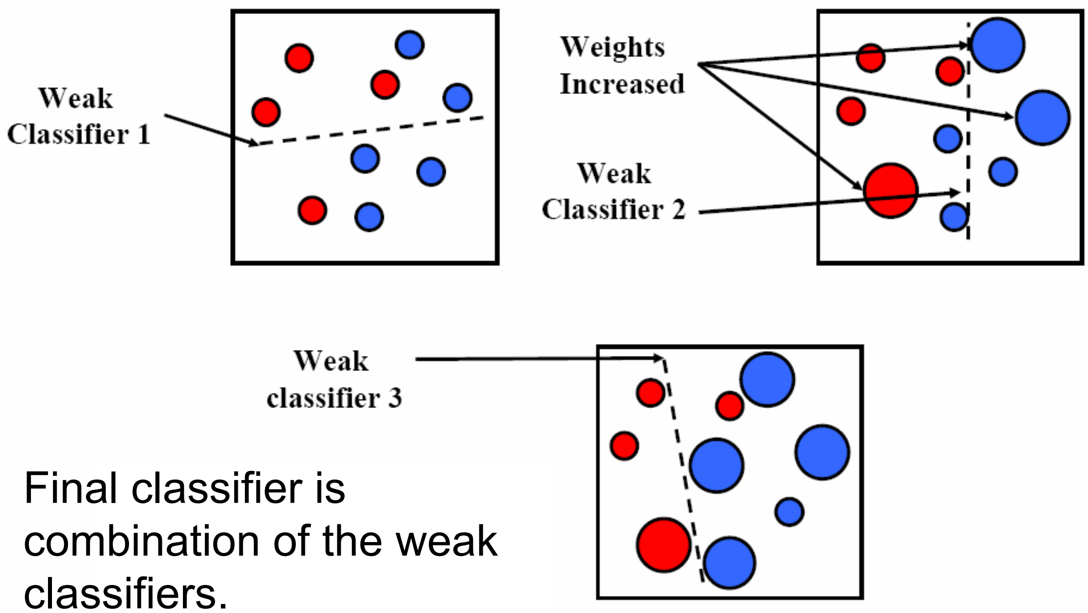



Figure from Freund and Schapire

# AdaBoost [Freund & Schapire]: Intuition




Final classifier is combination of the weak classifiers.

Figure from Freund and Schapire

# AdaBoost Algorithm [Freund & Schapire]:

- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where $m$ and $l$ are the number of negatives and positives respectively.
- For $t = 1, \dots, T$:
  1. Normalize the weights,
     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
     so that $w_t$ is a probability distribution.
  2. For each feature, $j$, train a classifier $h_j$ which is restricted to using a single feature. The error is evaluated with respect to $w_t$, $\epsilon_j = \sum_i w_i \,|h_j(x_i) - y_i|$.
  3. Choose the classifier, $h_t$, with the lowest error $\epsilon_t$.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
     where $e_i = 0$ if example $x_i$ is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- 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 $\alpha_t = \log \frac{1}{\beta_t}$

Start with uniform weights on training examples

Evaluate *weighted* error for each feature, pick best.

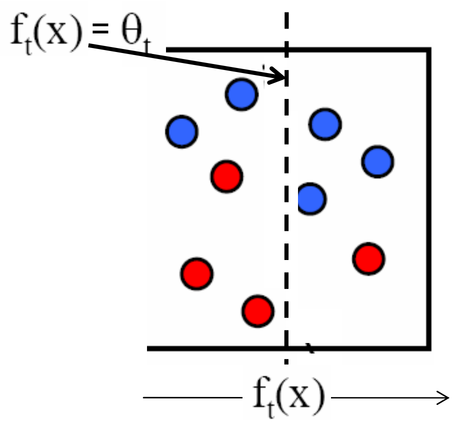
Incorrectly classified -> more weight

Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

# Boosting for feature selection

- Want to select the single rectangle feature that best separates positive and negative examples (in terms of weighted error).




This dimension: output of a possible rectangle feature on faces and non-faces.

Optimal threshold that results in minimal misclassifications

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

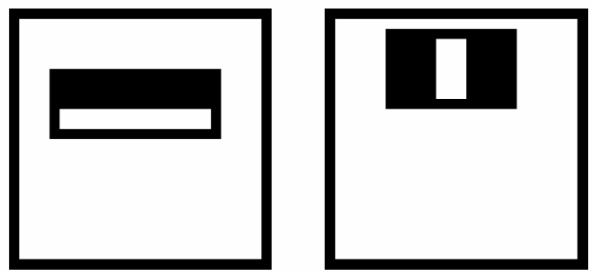
Image subwindow

# AdaBoost for Efficient Feature Selection

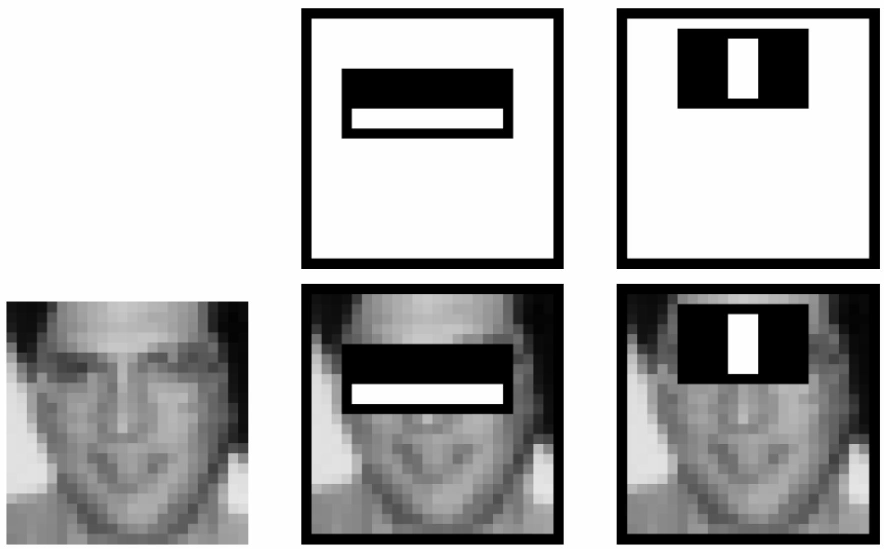
- Image Features = Weak Classifiers
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min error)
    - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this feature is a simple function of error rate
  - Reweight examples

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

First and second features selected by AdaBoost.



First and second features selected by AdaBoost.



# Questions

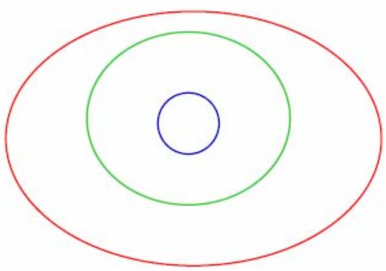
- How to discriminate faces and non-faces?
  - Representation choice
  - Classifier choice
- How to deal with the expense of such a windowed scan?
  - Efficient feature computation
  - Limit amount of computation required to make a decision per window

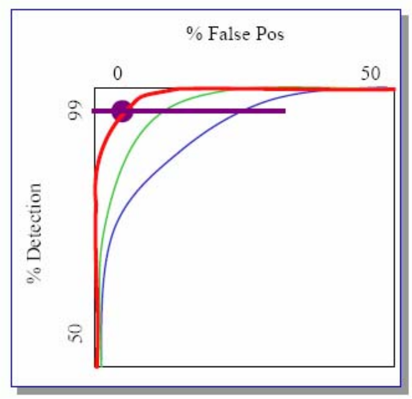
# Attentional cascade

- First apply smaller (fewer features, efficient) classifiers with very low *false negative* rates.
  - accomplish this by adjusting threshold on boosted classifier to get false negative rate near 0.
- This will reject many non-face windows early, but make sure most positives get through.
- Then, more complex classifiers are applied to get low *false positive* rates.
- Negative label at any point → reject sub-window

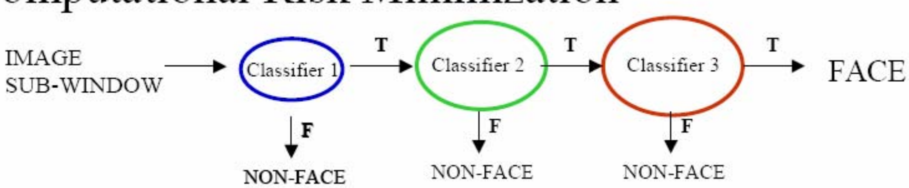
# Trading Speed for Accuracy

- Given a nested set of classifier hypothesis classes



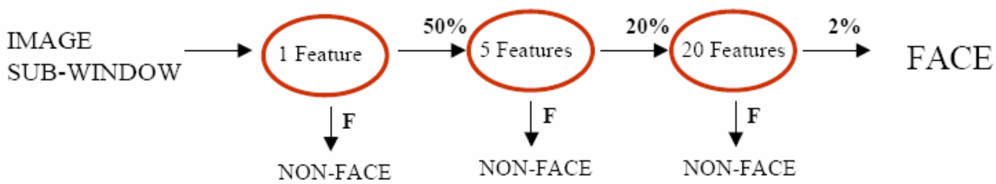



- Computational Risk Minimization




Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

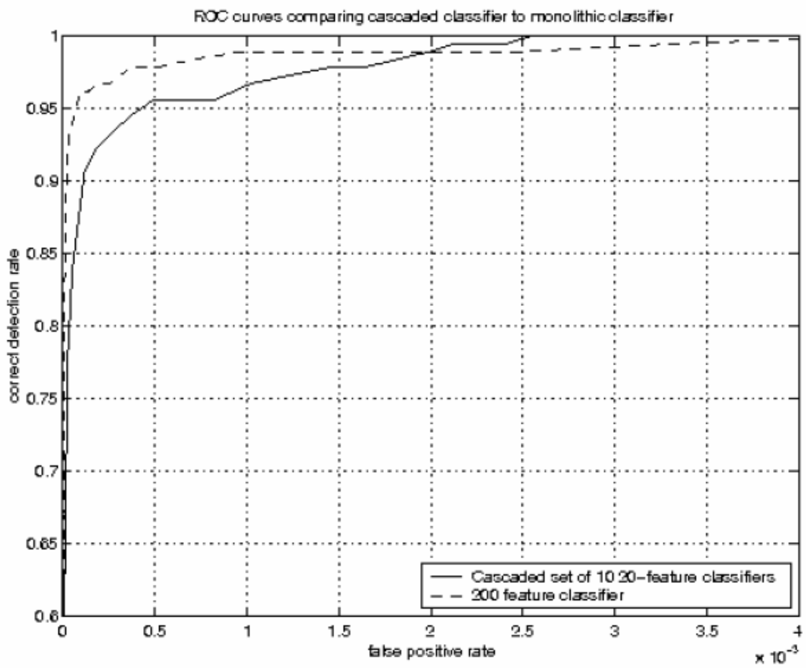
# Cascaded Classifier




- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - using data from previous stage.
- A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

# Experiment: Simple Cascaded Classifier




Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

# A Real-time Face Detection System

**Training faces**: 4916 face images (24 x 24 pixels) plus vertical flips for a total of 9832 faces

**Training non-faces**: 350 million sub-windows from 9500 non-face images

**Final detector**: 38 layer cascaded classifier
The number of features per layer was 1, 10, 25, 25, 50, 50, 50, 75, 100, …, 200, …

Final classifier contains 6061 features.

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

# Running the detector

- Scan across image at multiple scales and locations
- Scale the detector (features) rather than the input image
  - Note: does not change cost of feature computation

# Speed of Face Detector

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

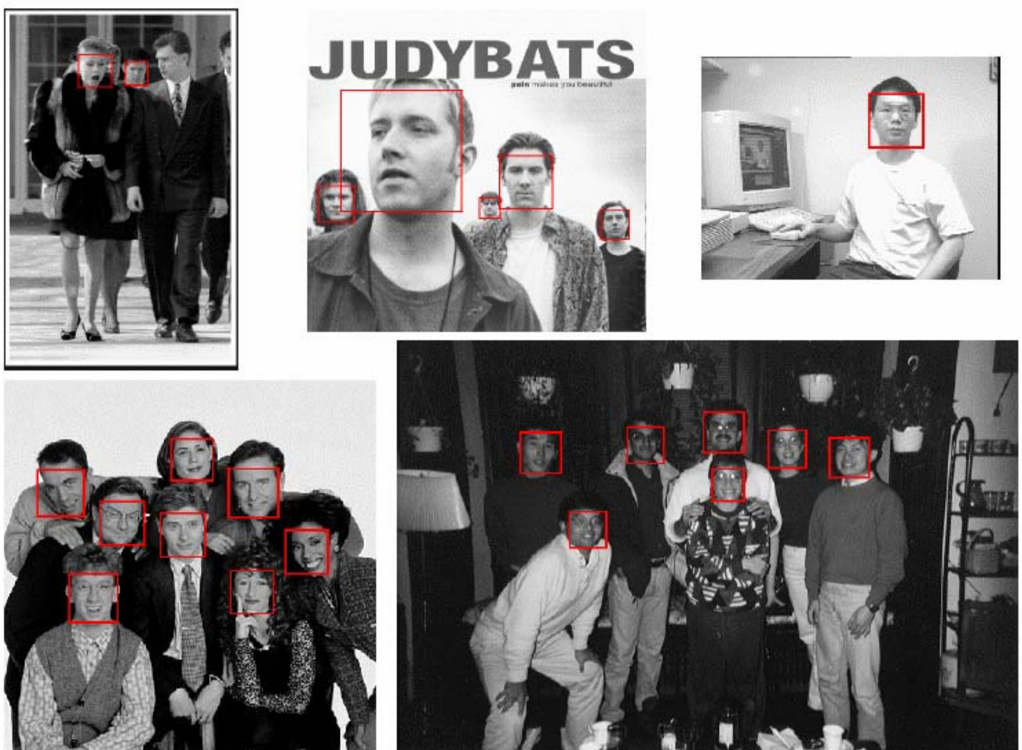
On the MIT+CMU test set, an average of 9 features out of a total of 6061 are computed per sub-window.

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.

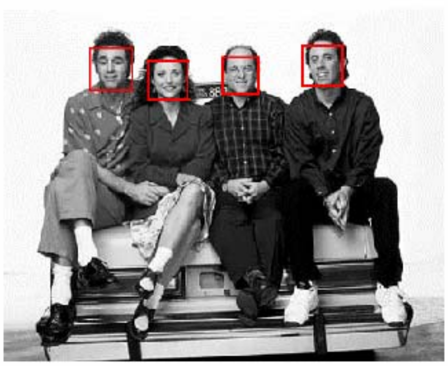
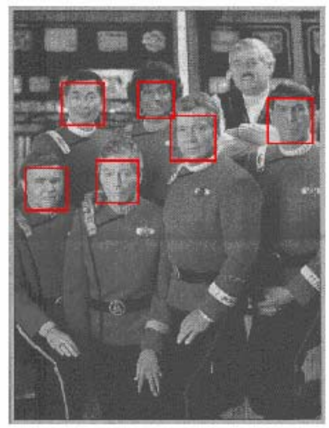
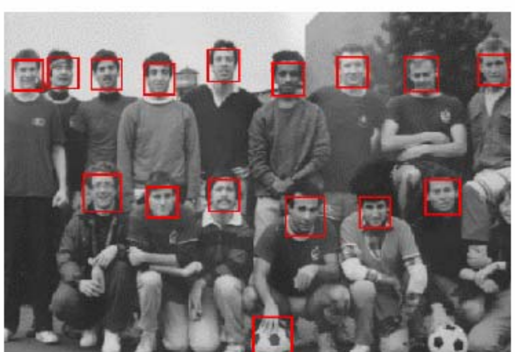
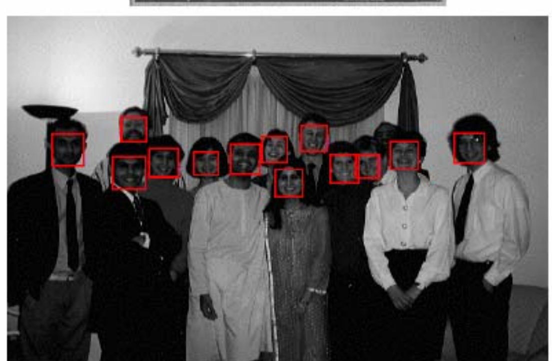
An implementation is available in Intel's OpenCV library.

# Output of Face Detector on Test Images




Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

# More Examples

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

# *Profile Detection*

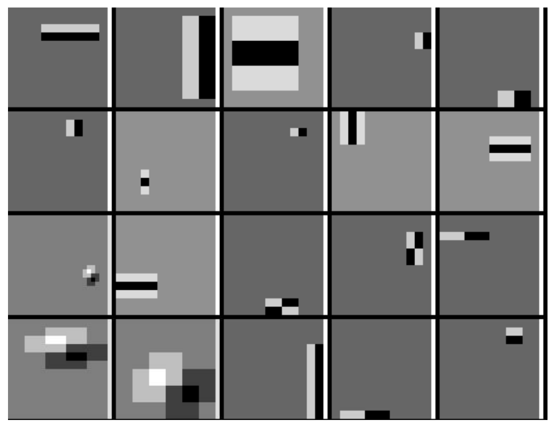
Train with profile views
instead of frontal



Viola 2003

# *More Results*

# *Profile Features*



# Fast detection: Viola & Jones

Key points:

- Huge library of features
- Integral image – efficiently computed
- AdaBoost to find best combo of features
- Cascade architecture for fast detection

# Local features vs. template matching

- Template matching
  - 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations
  - Partial occlusions and other variations not handled well without large increase in number of templates
  - **(Have to be careful about false positives!)**
- Local feature approach
  - Say 3000 points considered for evaluation
  - Features more invariant to illumination, 3d rotation, object variation
  - Use of many small sub-templates increases robustness to partial occlusion

Adapted from Bill Freeman, MIT

# General approaches to face recognition/detection

- Subspaces
  - e.g. Turk and Pentland, Belhumeur and Kreigman
- Shape and appearance models
  - e.g. Cootes and Taylor, Blanz and Vetter
- Boosting
  - e.g. Viola and Jones
- SVMs
  - e.g. Heisele et al., Guo et al.
- Neural networks
  - e.g. Rowley et al.
- HMMs
  - e.g. Nefian et al.

# Outline

- Last time:
  - Model-based recognition wrap-up
  - Classifiers: templates and appearance models
    - Histogram-based classifier
    - Eigenface approach, nearest neighbors
- Today:
  - Limitations of Eigenfaces, PCA
  - Discriminative classifiers
    - Viola & Jones face detector (boosting)
    - SVMs

# Next

Coming up:

- Problem set 4 out Thursday, due 11/29
- Read FP Ch 25