

# part 3: object recognition and detection

# Object Recognition

- Object is represented with a single model or prototype:
  1. Find consistent configurations of features for rigid objects:
    - ✓ Alignment to model or prototype (hypothesize & test)
    - ✓ Model selection by voting
    - ✓ Indexing with invariants - geometrical & photometrical
    - ✓ correspondence
  2. Part-based descriptions for articulated objects
  
- Common assumption for many effective methods - object is planar:
  - Valid for many structures on buildings
  - Sufficient for small viewpoint variations on 3D objects

⇒ 3-D objects are represented by multiple (distinct) viewpoints<sub>2</sub>

# Recognition by Hypothesize & Test

- General idea
  - Hypothesize object identity and pose
  - Render object in camera
  - Compare to image
- Issues
  - where do the hypotheses come from?
  - How do we compare to image (verification)?

# Alignment (late 1980's)

- Strategy:
  - Generate hypotheses using a small number of corresponding features (e.g., triples of points for a calibrated perspective camera)
  - Compute pose hypothesis
  - Backproject and verify pose consistency
- Appropriate groups are “frame groups”

# Example



Figure from “Object recognition using alignment,” D.P. Huttenlocher and S. Ullman, Proc. Int. Conf. Computer Vision, 1986, copyright IEEE, 1986

# More modern approach

- Representation: prototype (a typical image)
- Extract local invariant interest points in image
- Find corresponding points in stored images using cross-correlation
- Verify remaining points with the fundamental matrix

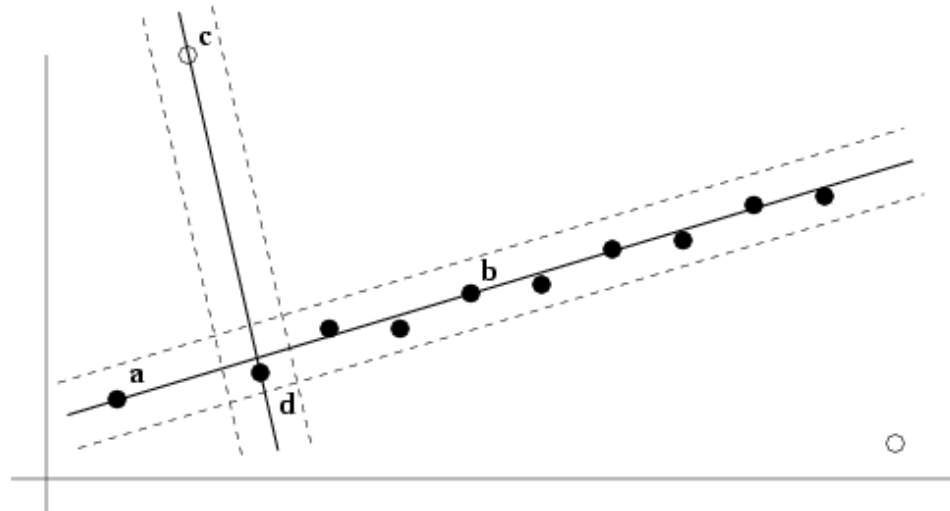
# Problem of Complexity

- Number of hypotheses is very large, each should be compared with every model in the database
- For each model, *there is much overlap (redundancy) between the hypotheses*

**⇒ solution: random sampling**

## RANSAC (RANDOM SAMPLE CONSENSUS) [Fischler81]

- **Create a hypothesis:** randomly choose a minimal subset of data points necessary to fit a model (a *sample*)
- **Test:** points within some distance threshold  $t$  of model are a *consensus set*; size of consensus set is model's *support*.
- Repeat for  $N$  samples; model with biggest support is most robust fit
  - Points within distance  $t$  of best model are inliers
  - Fit final model to all inliers



# RANSAC: How many samples?

- How many samples are needed?
    - Suppose  $w$  is fraction of inliers (points from line).
    - $n$  points needed to define hypothesis (2 for lines)
    - $k$  samples chosen.
  - Prob. that a single sample of  $n$  points is correct:  $w^n$
  - Prob. that all samples fail is:  $(1 - w^n)^k$
- ⇒ Choose  $k$  high enough to keep this below desired failure rate.

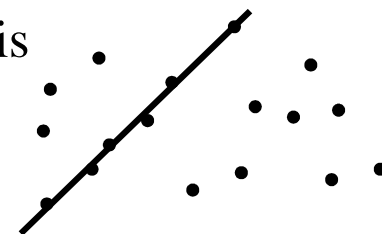
# Problem of Complexity

- Number of hypotheses is very large, each should be compared with every model in the database
- For each model, *there is much overlap (redundancy) between the hypotheses*

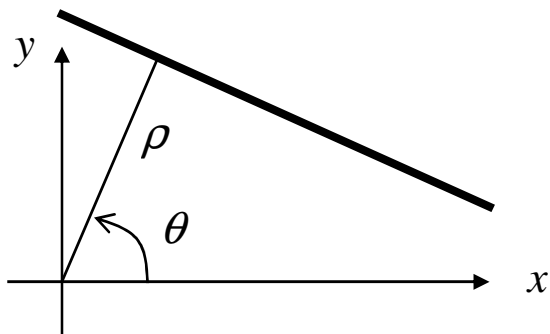
**⇒ solution: voting in some parameter space**

# Model Voting (Hough Transform)

- Origin: Detection of straight lines in clutter
  - Basic idea: each candidate point votes for all lines that it is consistent with.
  - Votes are accumulated in quantized array (hash table)
  - Local maxima correspond to candidate lines



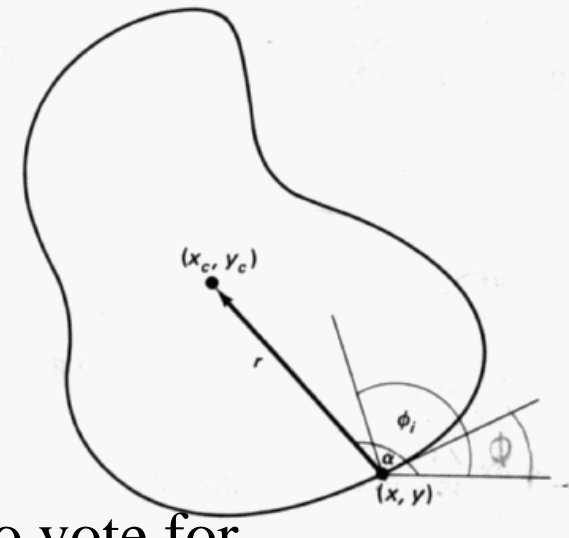
- Representation of a line
  - Usual form  $y = a x + b$  has a singularity around  $90^\circ$ .
  - Better parameterization:  $x \cos(\theta) + y \sin(\theta) = \rho$



# Generalized Hough Transform [Ballard81]

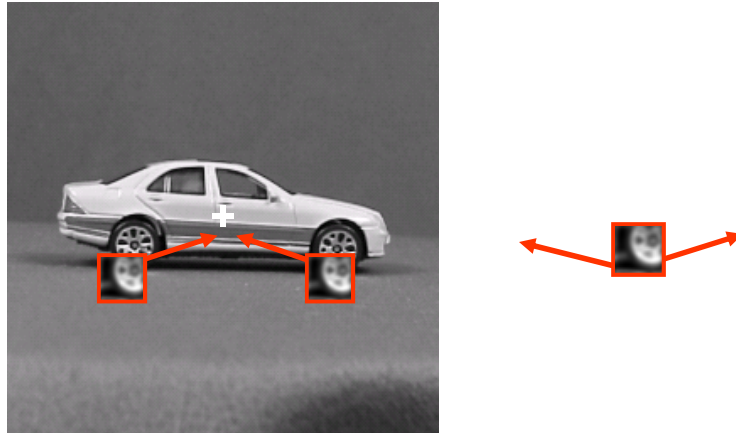
Generalization for an arbitrary contour or shape

- Choose reference point for the contour (e.g. center)
  - Parameters of reference points define parameter space:
    - ✓ For each point on the contour remember where it is located w.r.t. to the reference point
    - ✓ Remember radius  $r$  and angle  $\phi$  relative to the contour tangent
    - ✓ Recognition: whenever you find a contour point, calculate the tangent angle and ‘vote’ for all possible reference points
  - Instead of reference point, one can also vote for transformation (as in geometric hashing)
- ⇒ The same idea can be used with local features

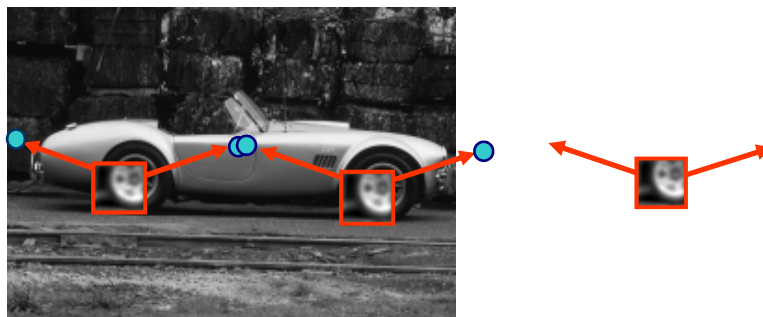


# Modern Approach: Gen. Hough Transform with Local Features

- For every feature, store possible “occurrences”



- For new image, let the matched features vote for possible object positions



# 3D Object Recognition

Visual Object Recognition - 67777



Figure credit: David Lowe

# Object Recognition

- Object is represented with a single model or prototype:
  1. Find consistent configurations of features for rigid objects:
    - ✓ Alignment to model or prototype (hypothesize & test)
    - ✓ Model selection by voting
    - ✓ **Indexing with invariants - geometrical & photometrical**
    - ✓ correspondence

# Problem of Complexity

- Number of hypotheses is very large, *each should be compared with every model in the database*
- For each model, there is much overlap (redundancy) between the hypotheses

**⇒ solution: indexing with invariants**

# Geometric invariants

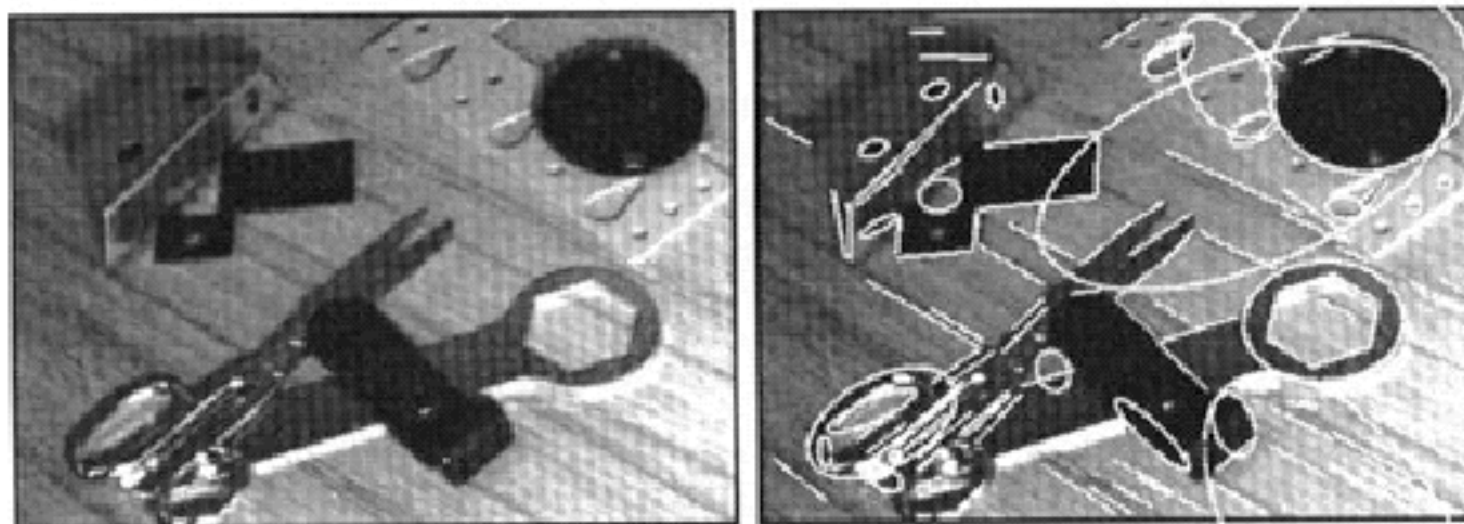
- Definition: functions with a value independent of the imaging transformation

$$f(x', y') = f(x, y) \quad \text{where} \quad (x', y')^t = T(x, y)^t$$

gives a way to compute a fixed value in every image of the object

- .
- Invariance to image rotation : distance between points
- Invariance to 2-D linear transformation: angles
- Invariance to planar homography : cross-ratio
- But: no invariants exist for the general imaging transformation: projection from 3D to 2D

# Examples



a

b

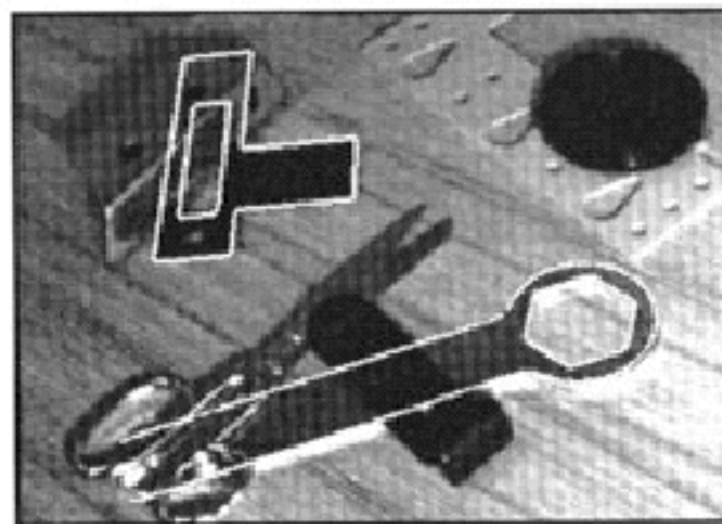


Figure from “Efficient model library access by projectively invariant indexing” C.A. Rothwell et al., CVPR 1992

# Object Recognition

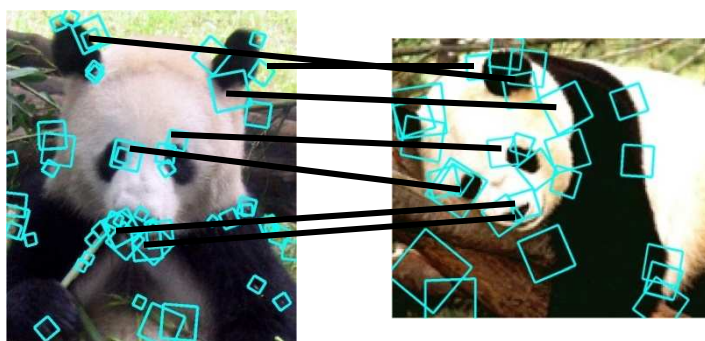
- Object is represented with a single model or prototype:
  1. Find consistent configurations of features for rigid objects:
    - ✓ Alignment to model or prototype (hypothesize & test)
    - ✓ Model selection by voting
    - ✓ Indexing with invariants - geometrical & photometrical
    - ✓ **correspondence**

# Local feature correspondences

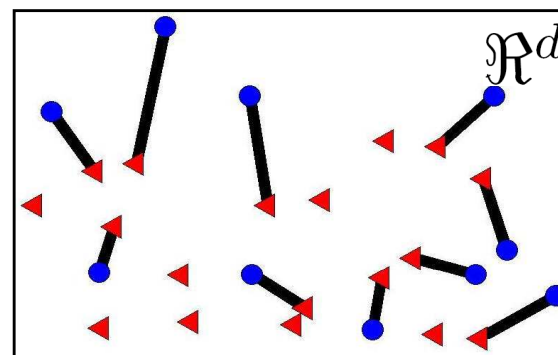
- The matching between sets of local features helps to establish overall similarity between objects or shapes.
- Assigned matches also useful for localization

# Local feature correspondences

- Least cost match: minimize total cost between matched points



$$\mathbf{X} = \{\vec{x}_1, \dots, \vec{x}_m\} \quad \mathbf{Y} = \{\vec{y}_1, \dots, \vec{y}_n\}$$



$$\min_{\pi: X \rightarrow Y} \sum_{x_i \in X} \|x_i - \pi(x_i)\|$$

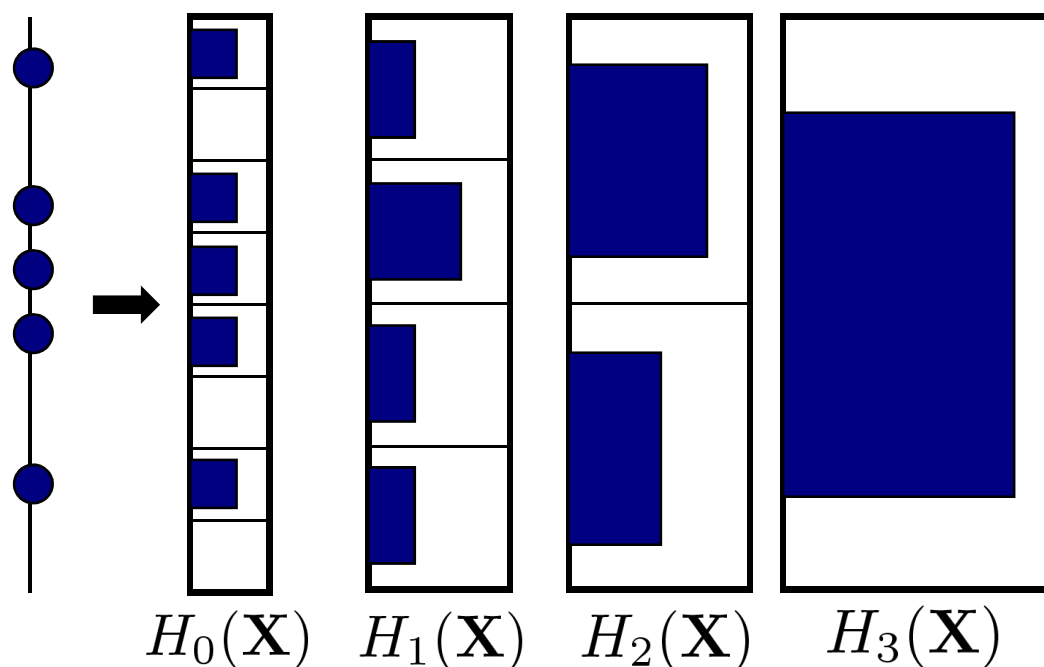
- Least cost *partial* match: match all of smaller set to some portion of larger set.

# Pyramid match kernel (PMK)

- Optimal matching expensive relative to number of features per image ( $m$ ):
  - Optimal match:  $O(m^3)$
  - Pyramid match:  $O(m)$
- PMK [Grauman & Darrell, ICCV 2005] gives approximate partial match

# Pyramid match kernel: pyramid extraction

$$\mathbf{X} = \{\vec{\mathbf{x}}_1, \dots, \vec{\mathbf{x}}_m\}, \quad \vec{\mathbf{x}}_i \in \mathbb{R}^d$$



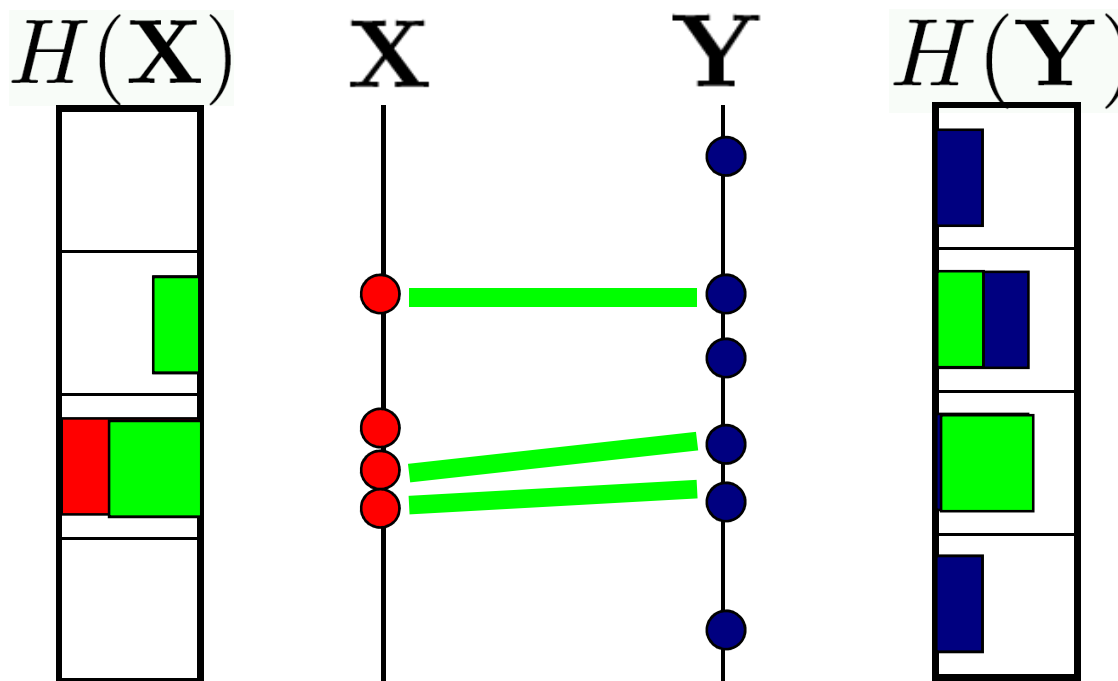
**Histogram  
pyramid:  
level  $i$  has  
bins of size  $2^i$**

$$\Psi(\mathbf{X}) = [H_0(\mathbf{X}), \dots, H_L(\mathbf{X})]$$

# Pyramid match kernel: counting matches

Histogram intersection

$$\mathcal{I}(H(\mathbf{X}), H(\mathbf{Y})) = \sum_{j=1}^r \min(H(\mathbf{X})_j, H(\mathbf{Y})_j)$$



$$\mathcal{I}(H(\mathbf{X}), H(\mathbf{Y})) = 3$$

# Pyramid match kernel: counting new matches

Histogram intersection  $\mathcal{I}(H(\mathbf{X}), H(\mathbf{Y})) = \sum_{j=1}^r \min(H(\mathbf{X})_j, H(\mathbf{Y})_j)$

<b>matches at this level</b>	<b>matches at previous level</b>
$N_i = \mathcal{I}(H_i(\mathbf{X}), H_i(\mathbf{Y}))$	$- \mathcal{I}(H_{i-1}(\mathbf{X}), H_{i-1}(\mathbf{Y}))$

Difference in histogram intersections across levels counts *number of new pairs* matched

# Pyramid match kernel

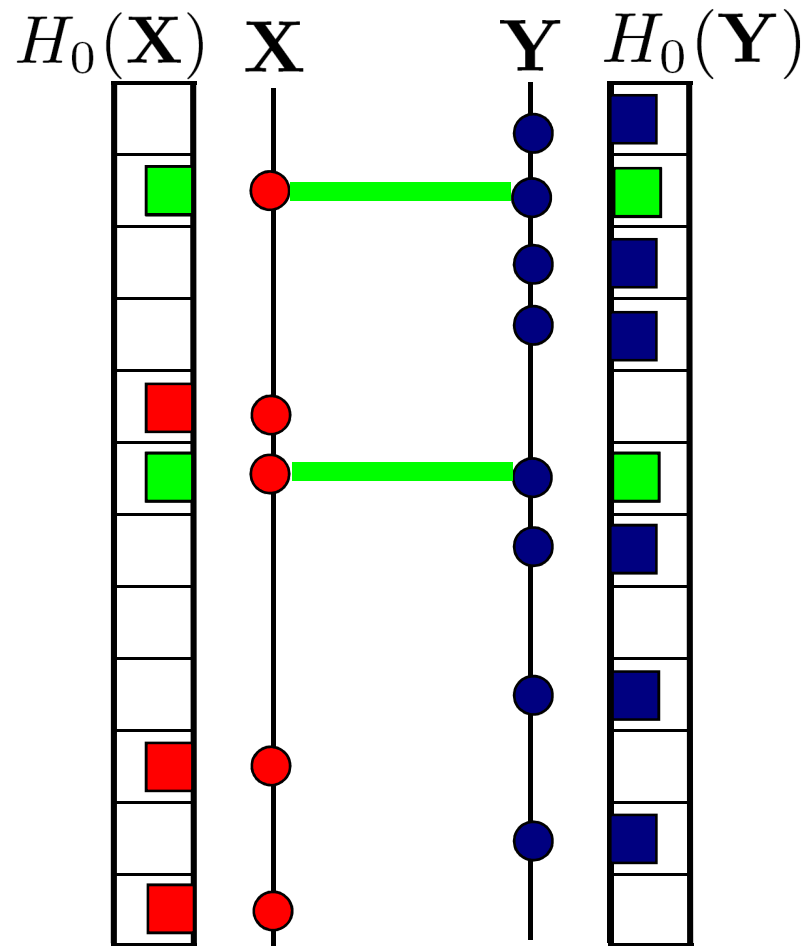
$$K_{\Delta} \left( \overbrace{\Psi(\mathbf{X}), \Psi(\mathbf{Y})}^{\text{histogram pyramids}} \right) = \sum_{i=0}^L \frac{1}{2^i} \left( \underbrace{\mathcal{I}(H_i(\mathbf{X}), H_i(\mathbf{Y})) - \mathcal{I}(H_{i-1}(\mathbf{X}), H_{i-1}(\mathbf{Y}))}_{\text{number of newly matched pairs at level } i} \right)$$

↑  
measure of difficulty of a match at level  $i$

- For similarity, weights inversely proportional to bin size (or may be learned discriminatively)
- Normalize kernel values to avoid favoring large sets

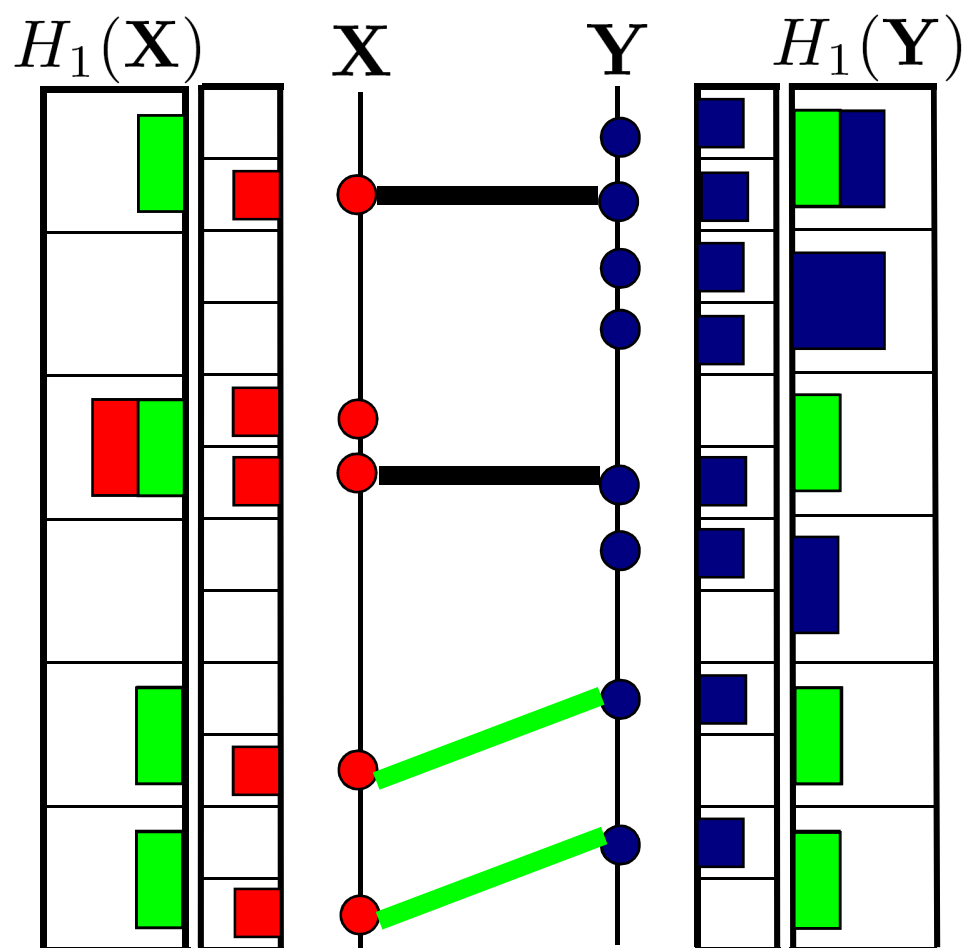
# Example pyramid match

$$\mathcal{I}(H_0(\mathbf{X}), H_0(\mathbf{Y})) = 2 \longrightarrow \begin{matrix} N_0 = 2 \\ w_0 = 1 \end{matrix}$$



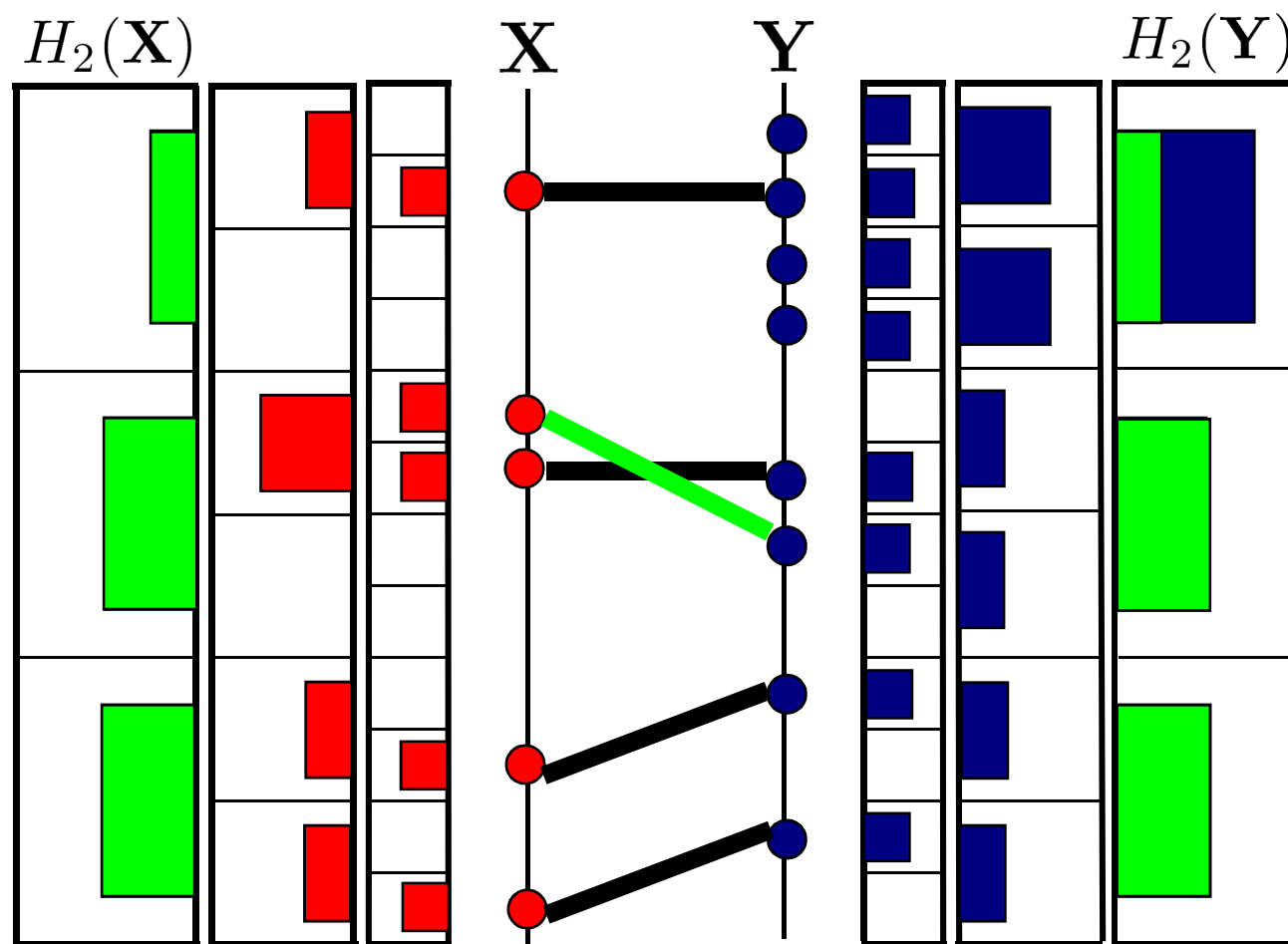
# Example pyramid match

$$\mathcal{I}(H_1(\mathbf{X}), H_1(\mathbf{Y})) = 4 \longrightarrow \begin{aligned} N_1 &= 4 - 2 = 2 \\ w_1 &= \frac{1}{2} \end{aligned}$$



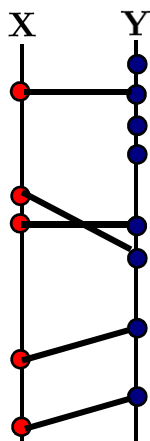
# Example pyramid match

$$\mathcal{I}(H_2(\mathbf{X}), H_2(\mathbf{Y})) = 5 \longrightarrow \begin{aligned} N_2 &= 5 - 4 = 1 \\ w_2 &= \frac{1}{4} \end{aligned}$$



# Example pyramid match

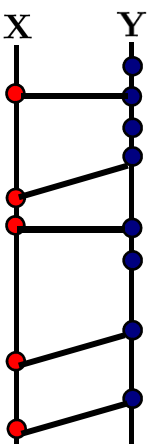
pyramid match



$$K_{\Delta} = \sum_{i=0}^L w_i N_i$$

$$= 1(2) + \frac{1}{2}(2) + \frac{1}{4}(1) = 3.25$$

optimal match

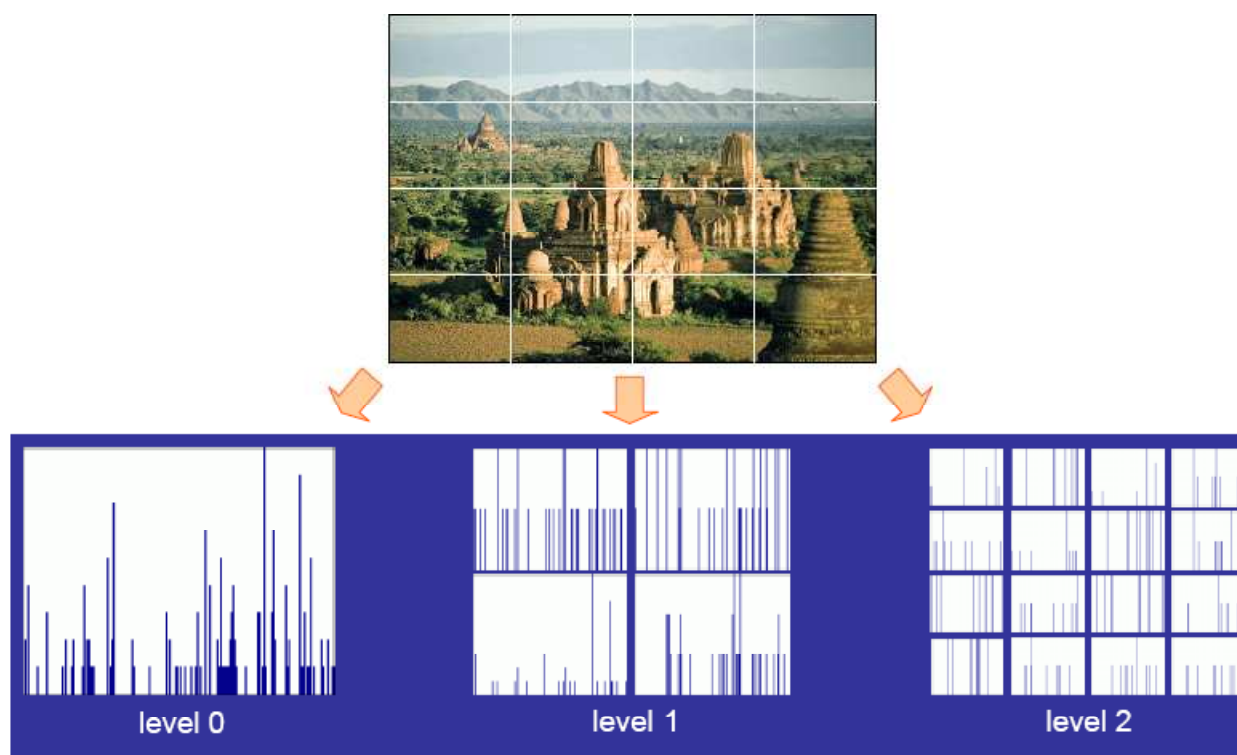


$$K = \max_{\pi: \mathbf{X} \rightarrow \mathbf{Y}} \sum_{\mathbf{x}_i \in \mathbf{X}} \mathcal{S}(\mathbf{x}_i, \pi(\mathbf{x}_i))$$

$$= 1(2) + \frac{1}{2}(3) = 3.5$$

# Spatial pyramid match kernel

- First quantize descriptors into words, then do one pyramid match *per word* in image coordinate space.



Lazebnik, Schmid & Ponce, CVPR 2006

# Object Recognition

- Object is represented with a single model or prototype:
  1. Find consistent configurations of features for rigid objects:
    - ✓ Alignment to model or prototype (hypothesize & test)
    - ✓ Model selection by voting
    - ✓ Indexing with invariants - geometrical & photometrical
    - ✓ correspondence
  2. **Part-based descriptions for articulated objects**

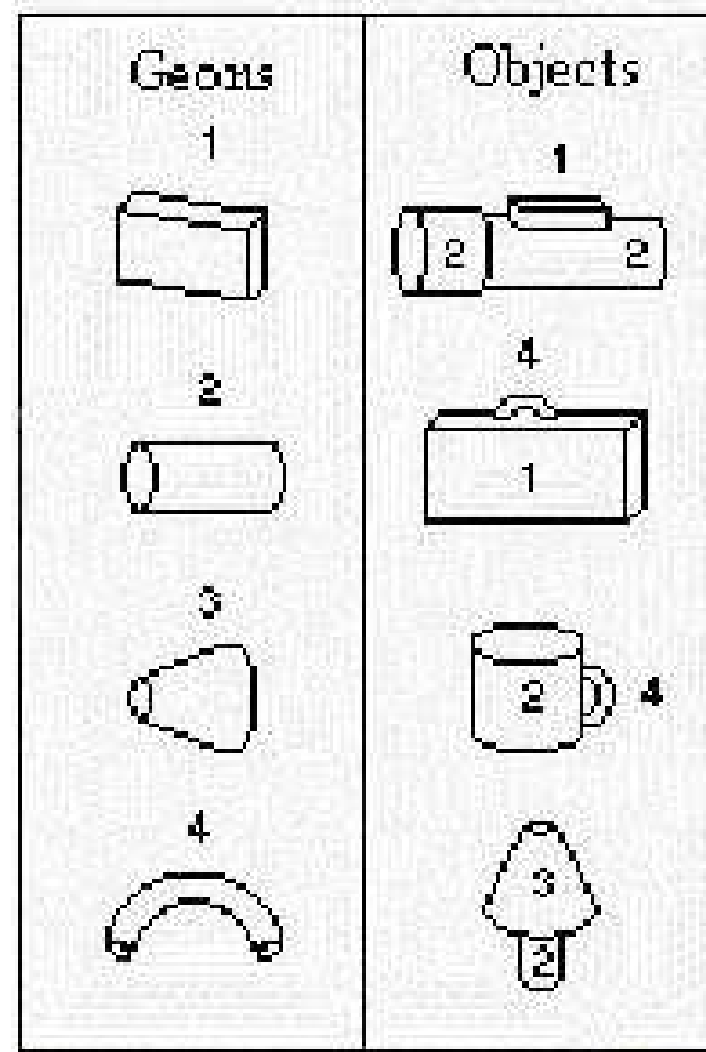
## 2. Part-based Object Recognition

- Finding consistent recognition is useful for rigid objects
- How about articulated objects, those that change their internal configuration?

# Recognition by Components

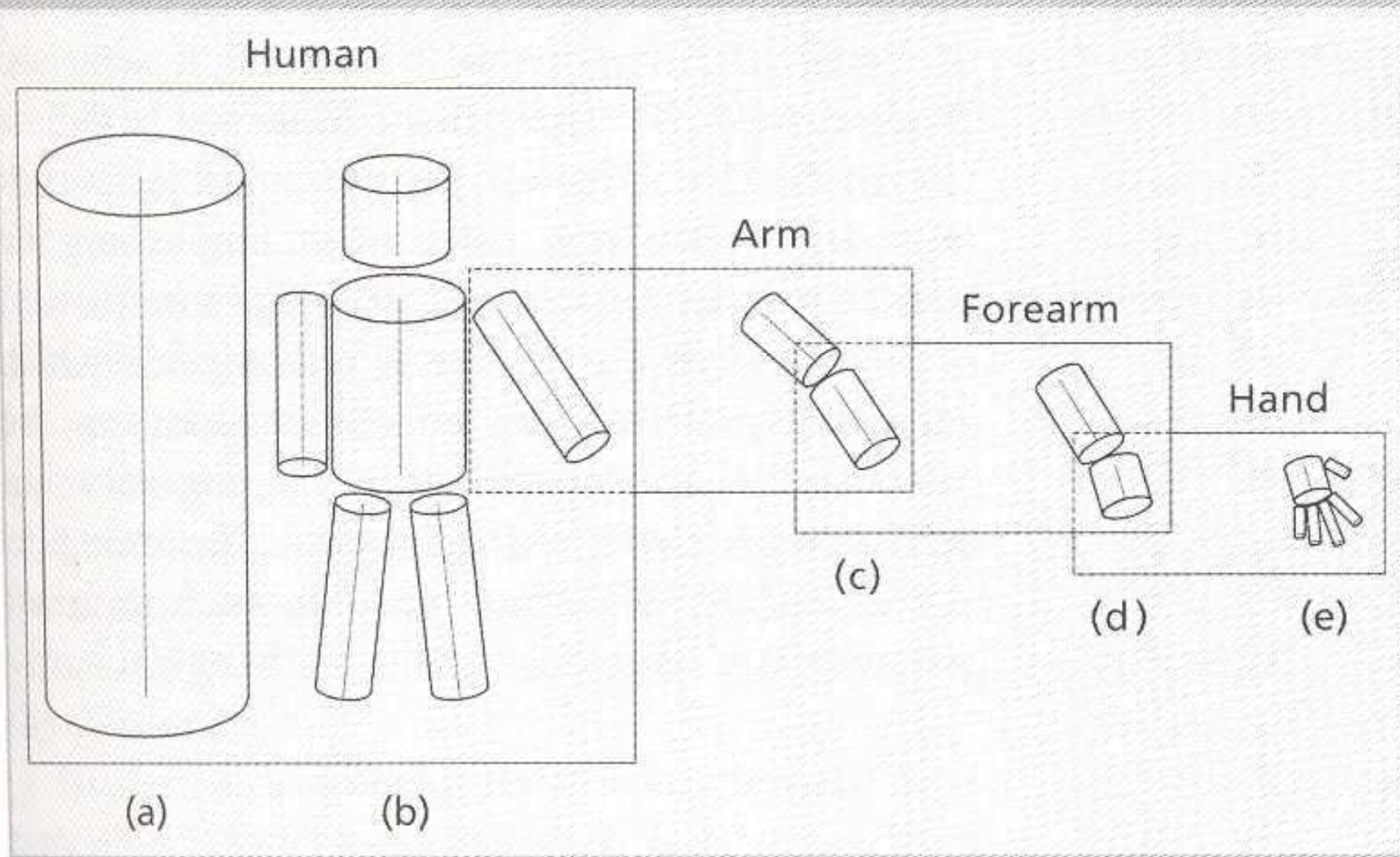
(Irving Biederman)

- Complex objects are constructed from pieces (primitives) called *geons*:
  - View invariant – look the same from most angles
  - Discriminative – not confusable with each other
  - Noise resistant
  - Parsimonious: few geons can create complex representations
  
- We recognize an unfamiliar object by parsing its component pieces



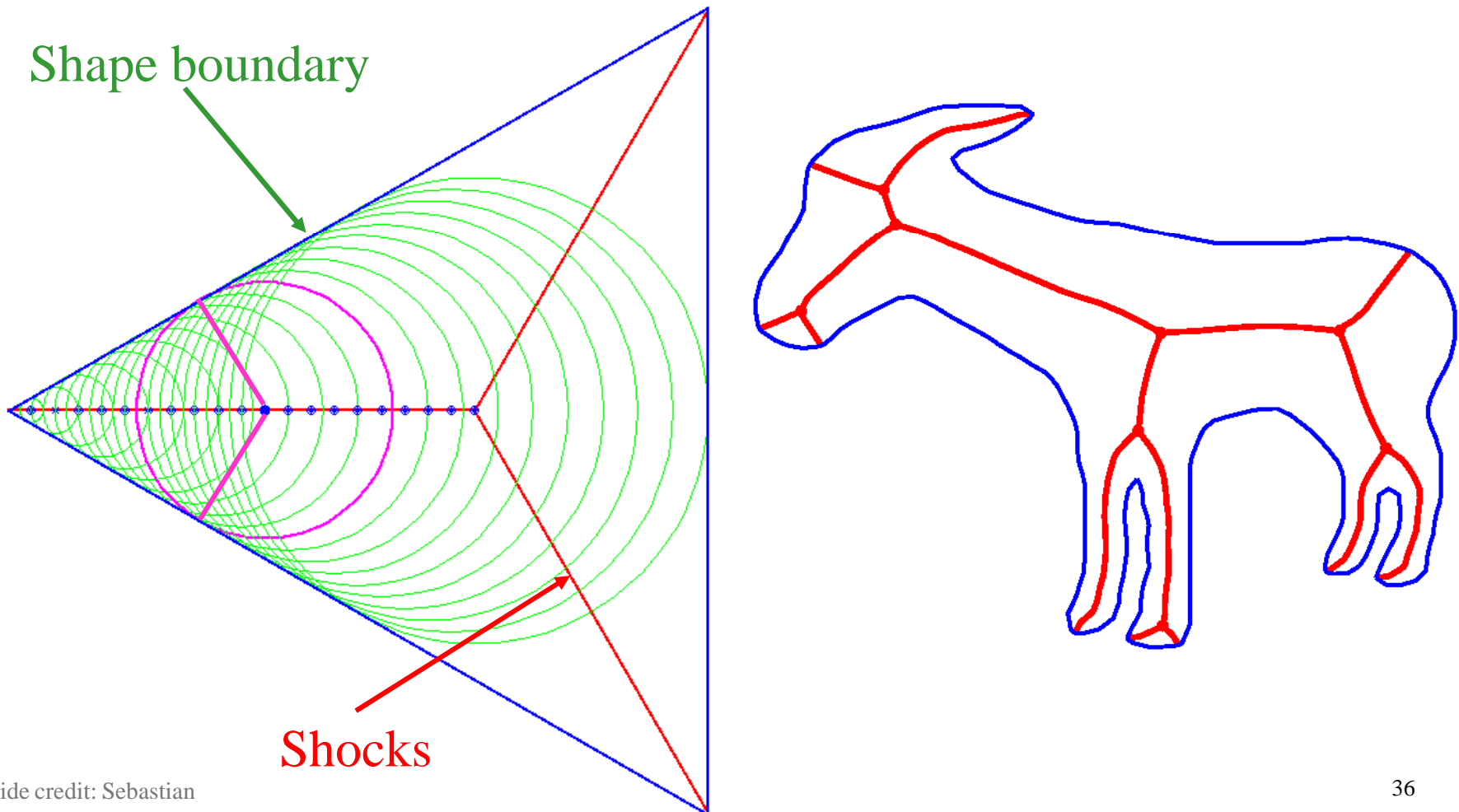
# Binford '78

FIGURE 4.4

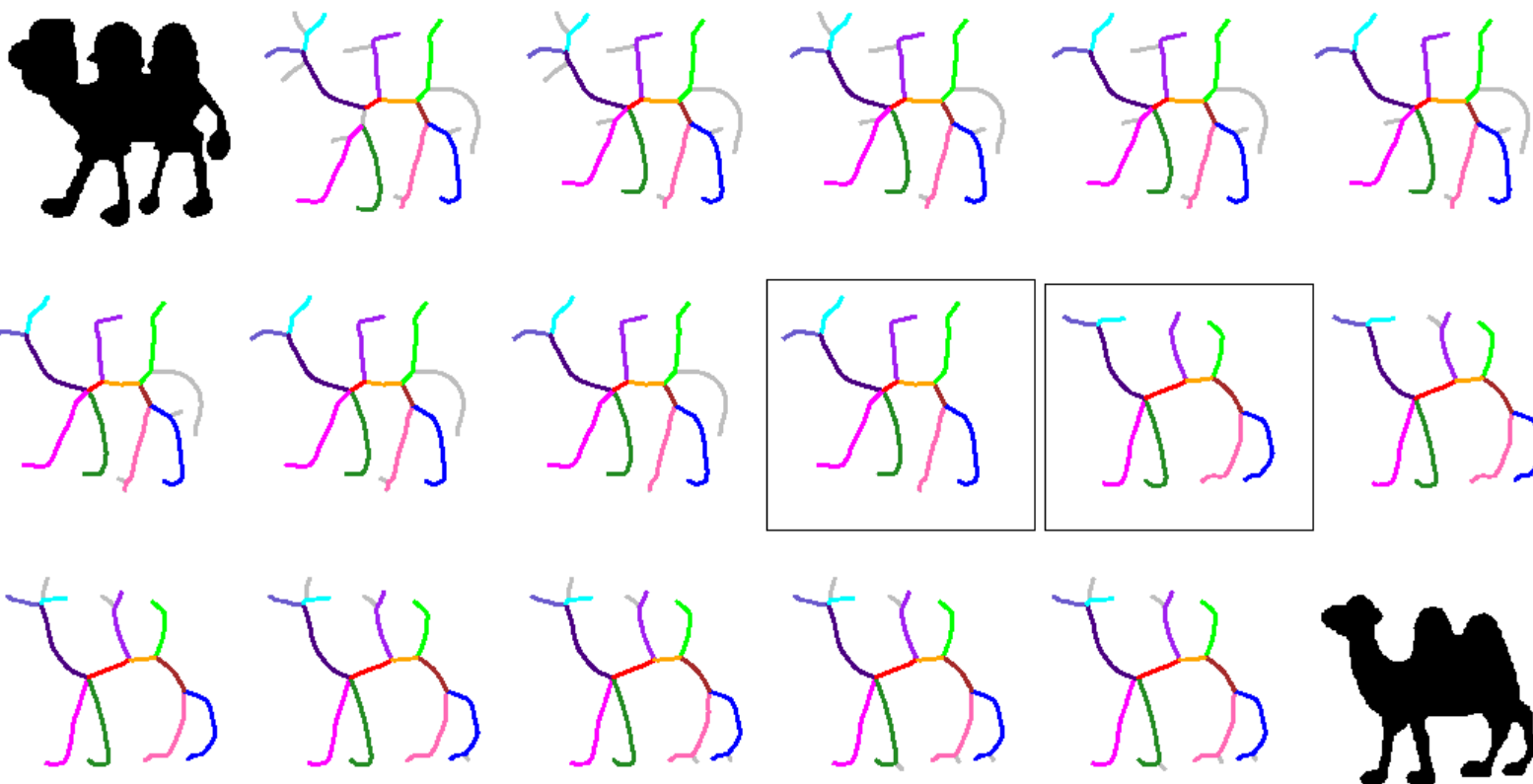


# Computing part-decomposition

**Shocks** (or medial axis or skeleton) are locus of centers of maximal circles that are bitangent to shape boundary



# Matching two shapes



# Shock graphs represent both object parts and part hierarchy



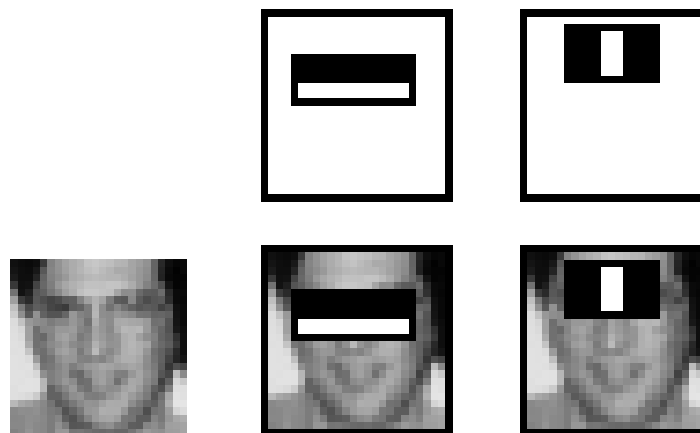
# Object detection

- Up to now we talked about identification and verification
- Now to face detection via a totally different idea
  - Use many very simple features
  - Learn cascade of tests for target object
  - Efficient if:
    - ✓ features easy to compute
    - ✓ cascade short

# Combining “good” features:

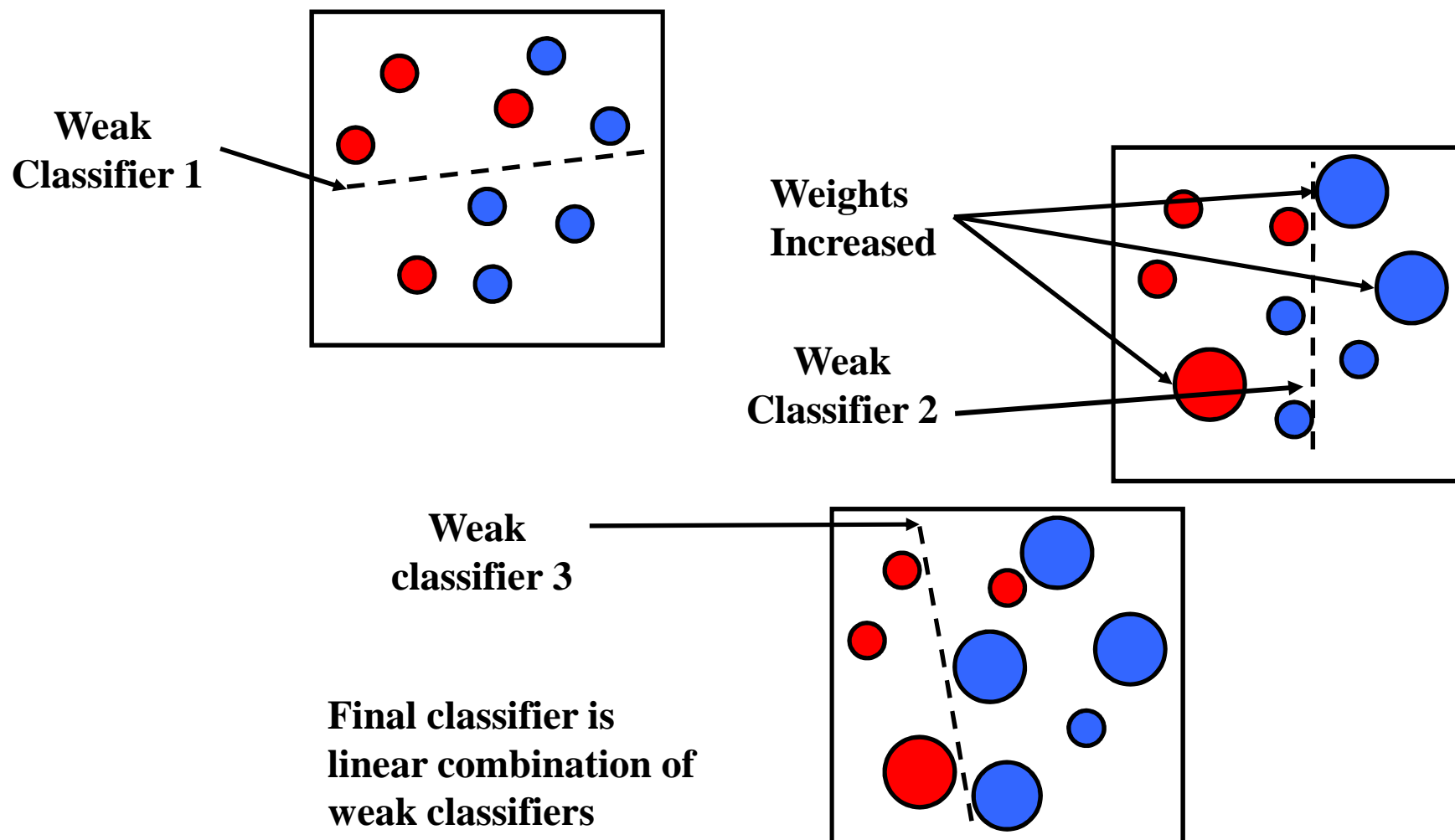
[Basic tool: AdaBoost (Viola & Jones 2004)]

- Construct a “weak” classifier from each generalized Haar feature
- Given this set of classifiers:
  - Pick best one
  - Reweight training examples, so that misclassified images have larger weight
  - Reiterate; then linearly combine resulting classifiers



# AdaBoost

*Freund & Shapire*



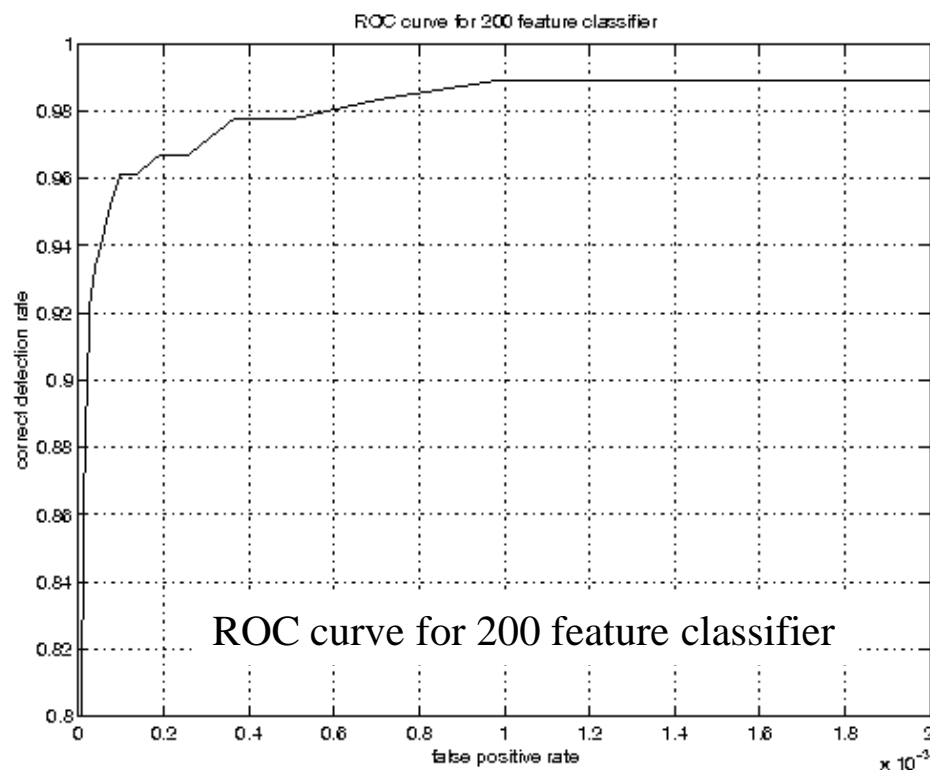
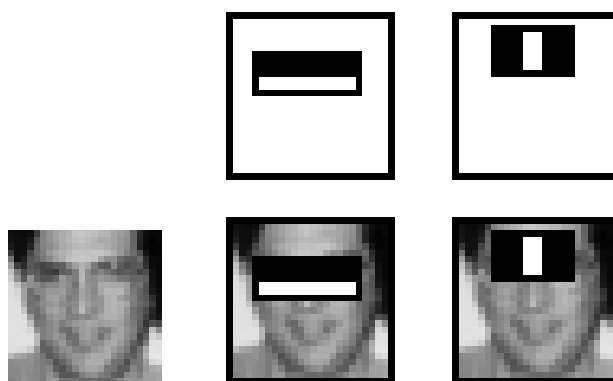
# AdaBoost gives efficient classifier:

- Features  $\Leftrightarrow$  Weak Classifiers
- Each round selects the optimal feature given:
  - Previous selected features
  - Exponential Loss

# Example Classifier for Face Detection

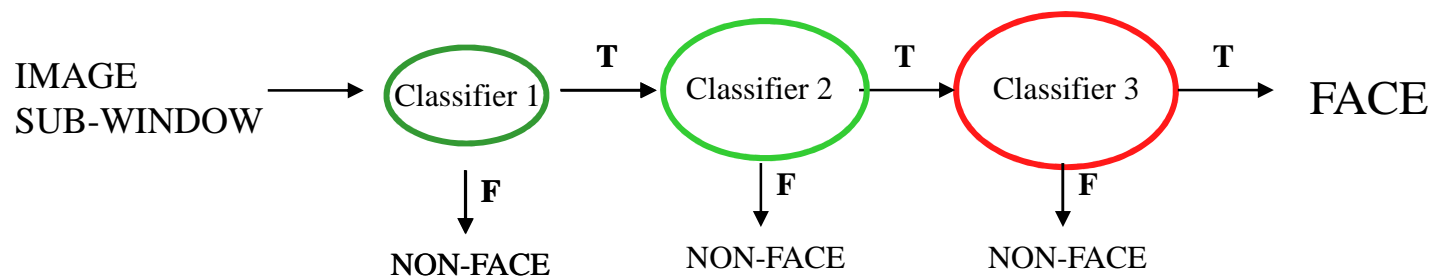
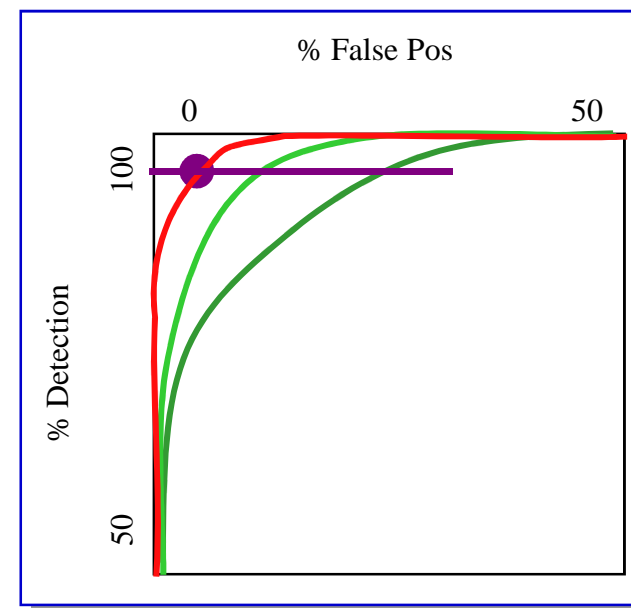
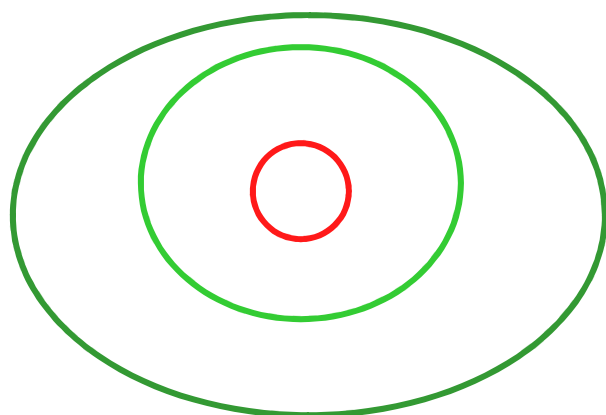
A classifier with 200 rectangle features was learned using AdaBoost

95% correct detection on test set with 1 in 14084 false positives.

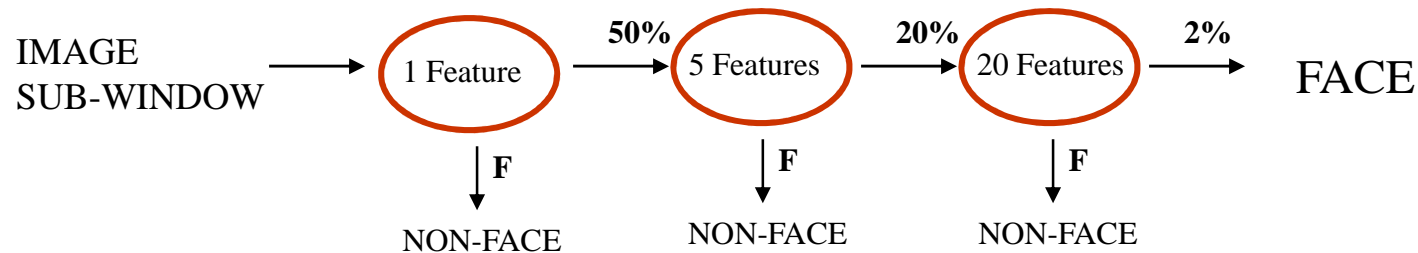


# Efficiency via nested classifiers

- Given a nested set of classifier hypothesis classes

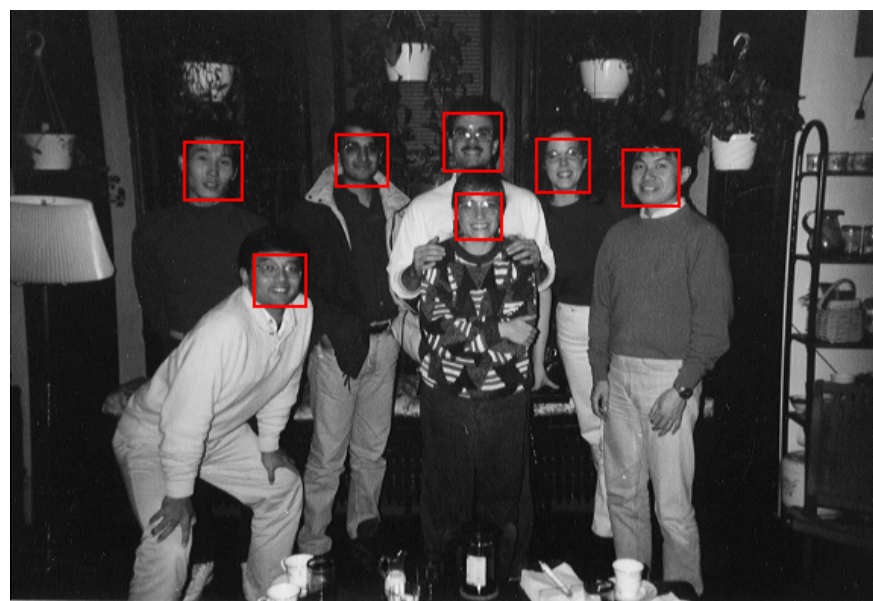
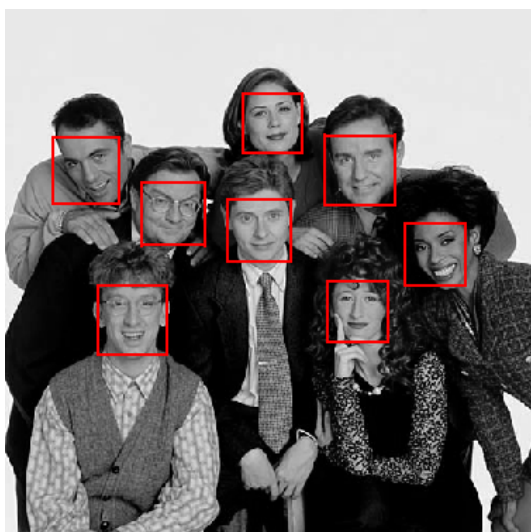
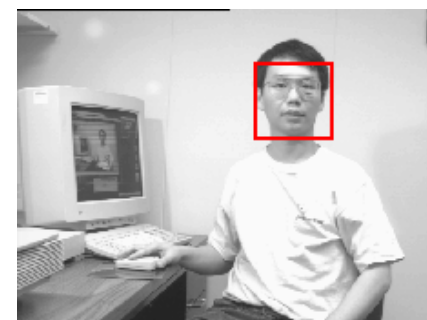
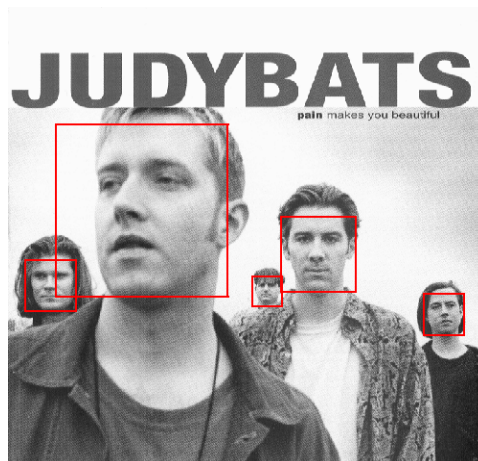


# Cascaded Classifier



- 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, and assuming independence*
- A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

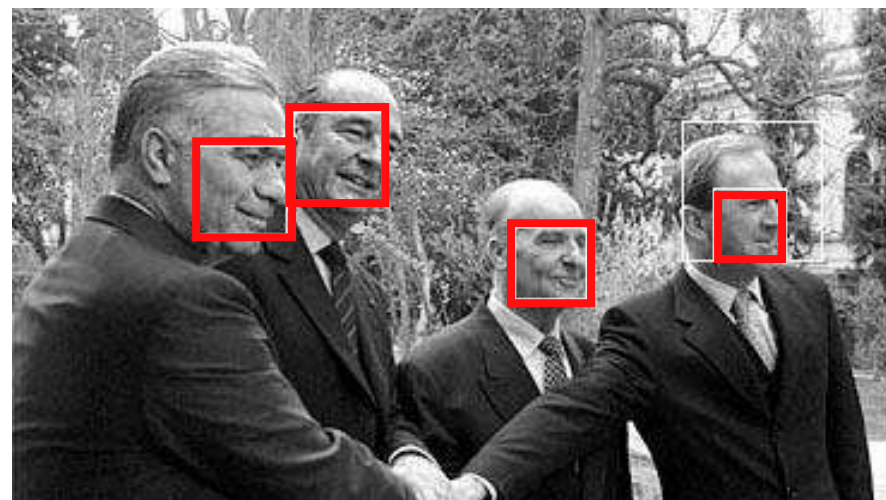
# Output of Face Detector on Test Images



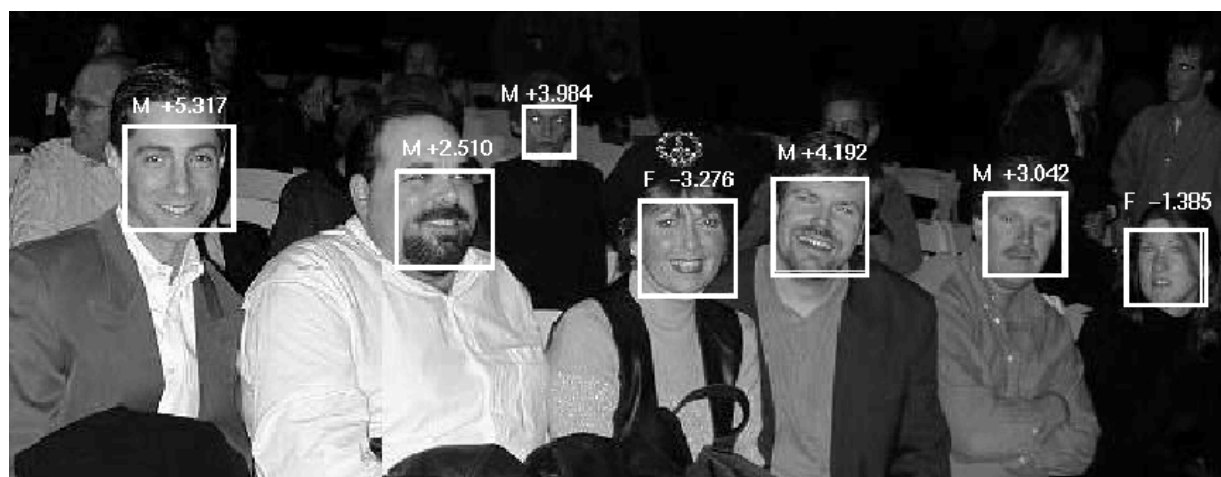
# Solving other "Face" Tasks



Facial Feature Localization



Profile Detection



Demographic Analysis

# Face Profile Detection



# Summary

- Rigid object recognition: find consistent configurations
  - Identification: hypothesis sampling, indexing via invariants
  - Verification: re-projection, voting
- Articulated object recognition: part based representations
- Object detection: via the learning of a cascade of classifiers (defined by efficient features)