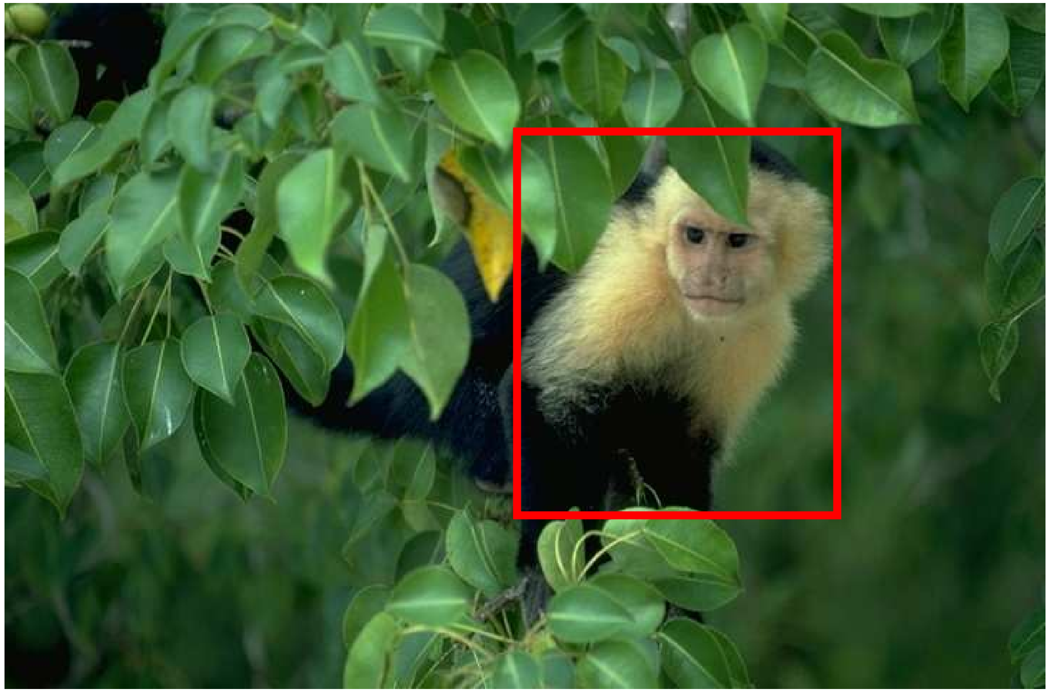
Category-level localization

Cordelia Schmid

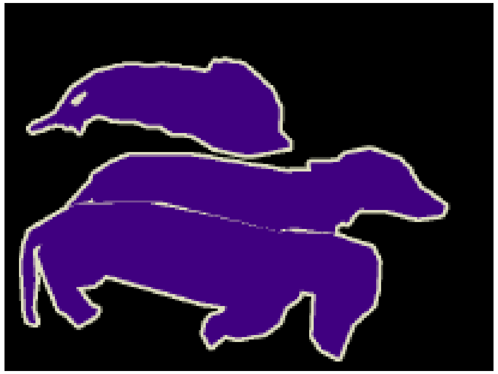
Recognition

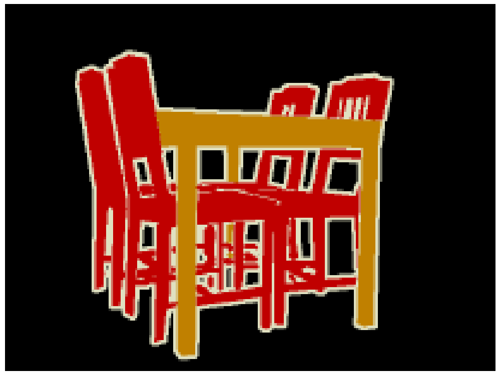
- Classification
  - Object present/absent in an image
  - Often presence of a significant amount of background clutter

- Localization / Detection
  - Localize object within the frame
  - Bounding box or pixel-level segmentation



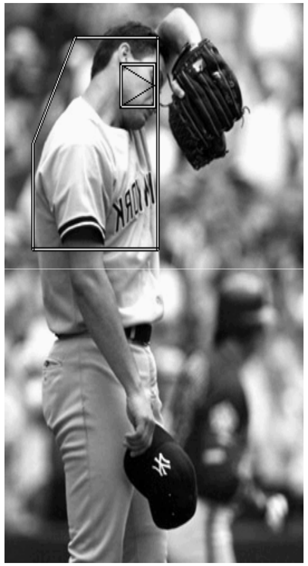
Pixel-level object classification

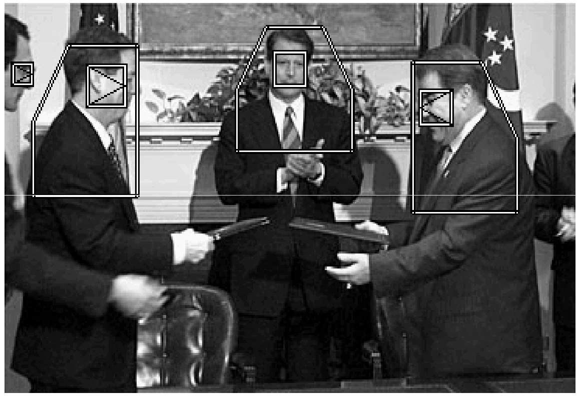



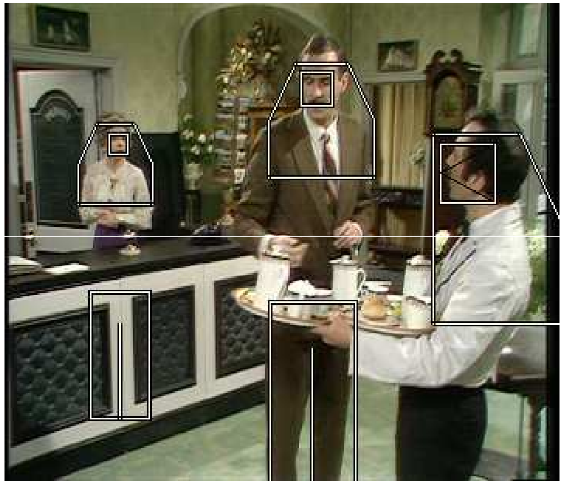



Difficulties

- Intra-class variations







- Scale and viewpoint change
- Multiple aspects of categories

Approaches

- Intra-class variation

  => Modeling of the variations, mainly by learning from a large dataset, for example by SVMs

- Scale + limited viewpoints changes

  => multi-scale approach or invariant local features

- Multiple aspects of categories

  => separate detectors for each aspect, front/profile face, build an approximate 3D “category” model

Approaches

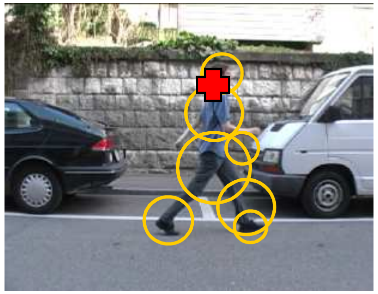
- Localization (bounding box)
  - Hough transform
  - Sliding window approach
- Localization (segmentation)
  - Shape based
  - Pixel-based +MRF
  - Segmented regions + classification

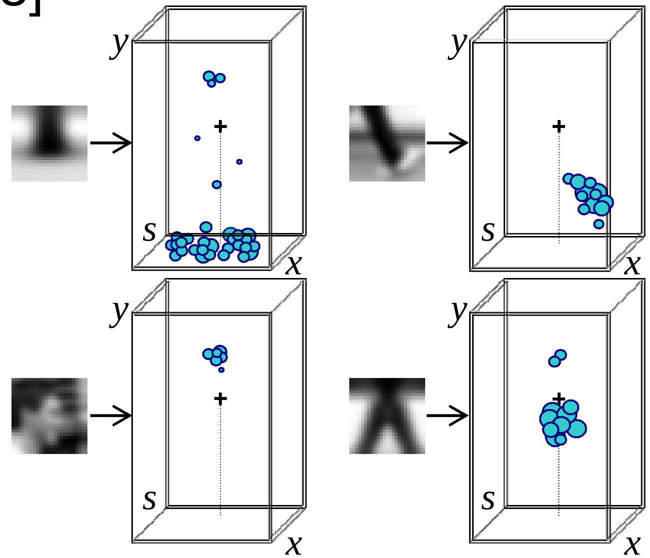
Hough voting

- Use Hough space voting to find objects of a class
- Implicit shape model [Leibe and Schiele ’03,’05]

Learning

- Learn appearance codebook
  - Cluster over interest points on training images
- Learn spatial distributions
  - Match codebook to training images
  - Record matching positions on object
  - Centroid + scale is given

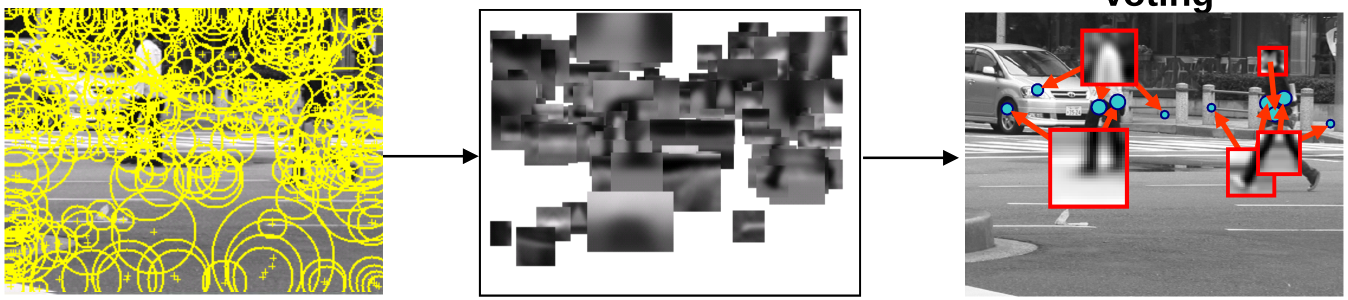




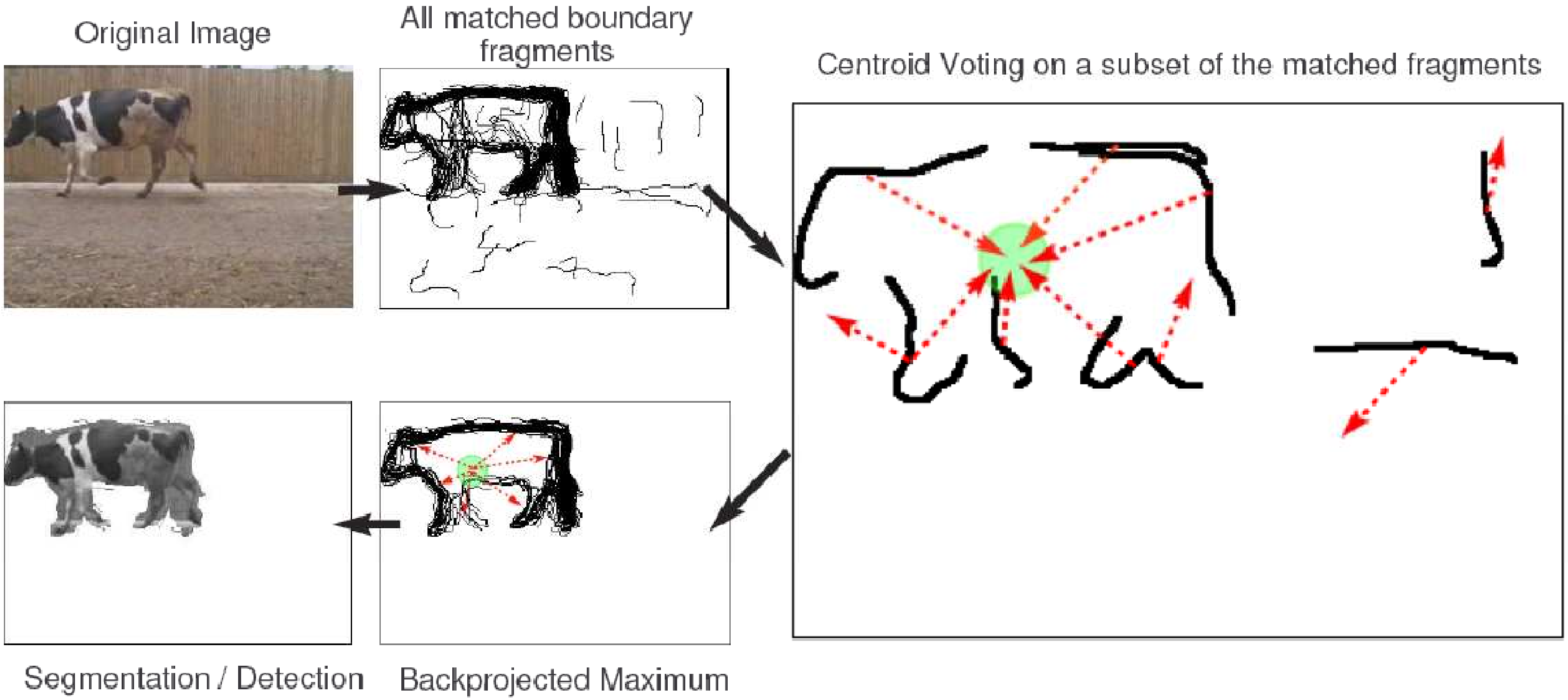
Spatial occurrence distributions

Recognition

Interest Points | **Matched Codebook Entries** | **Probabilistic Voting**



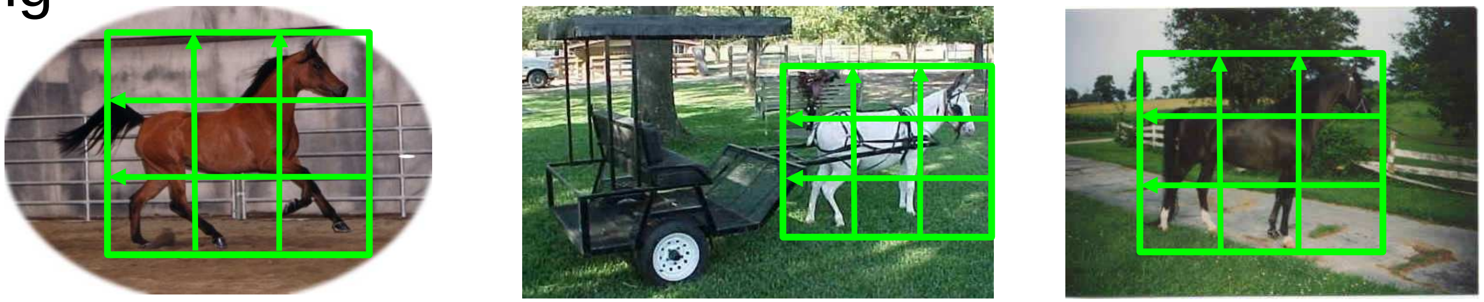
Hough voting



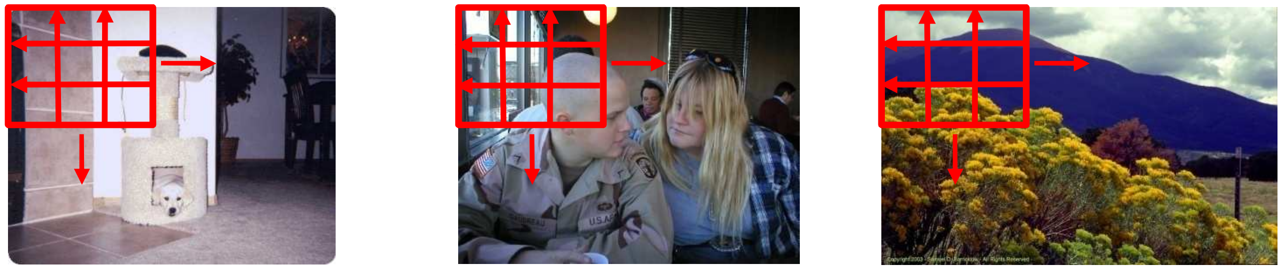
[Opelt, Pinz,Zisserman, ECCV 2006]

Localization with sliding window

Training



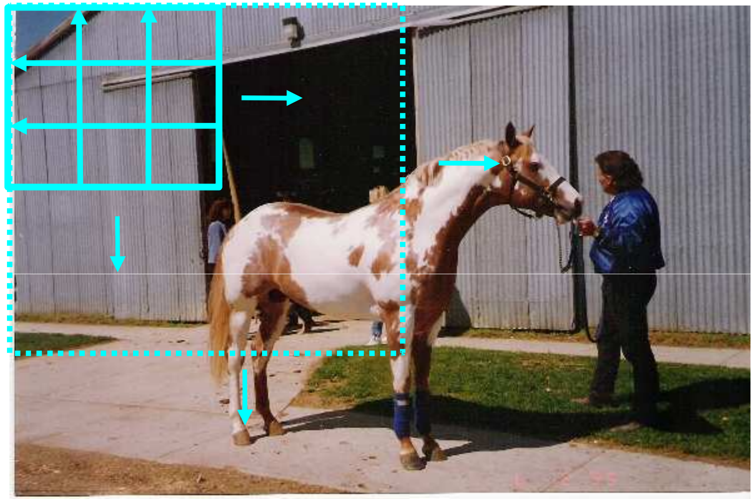
Positive examples



Negative examples

Description + Learn a classifier

Localization with sliding window



Testing at multiple locations and scales

Find local maxima, non-maxima suppression

Sliding Window Detectors

Detection Phase

Scan image(s) at all scales and locations

↓

Extract features over windows

↓

Run window classifier at all locations

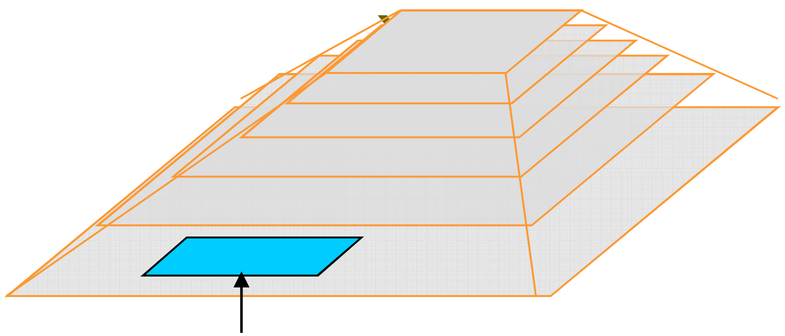
↓

Fuse multiple detections in 3-D position & scale space

↓

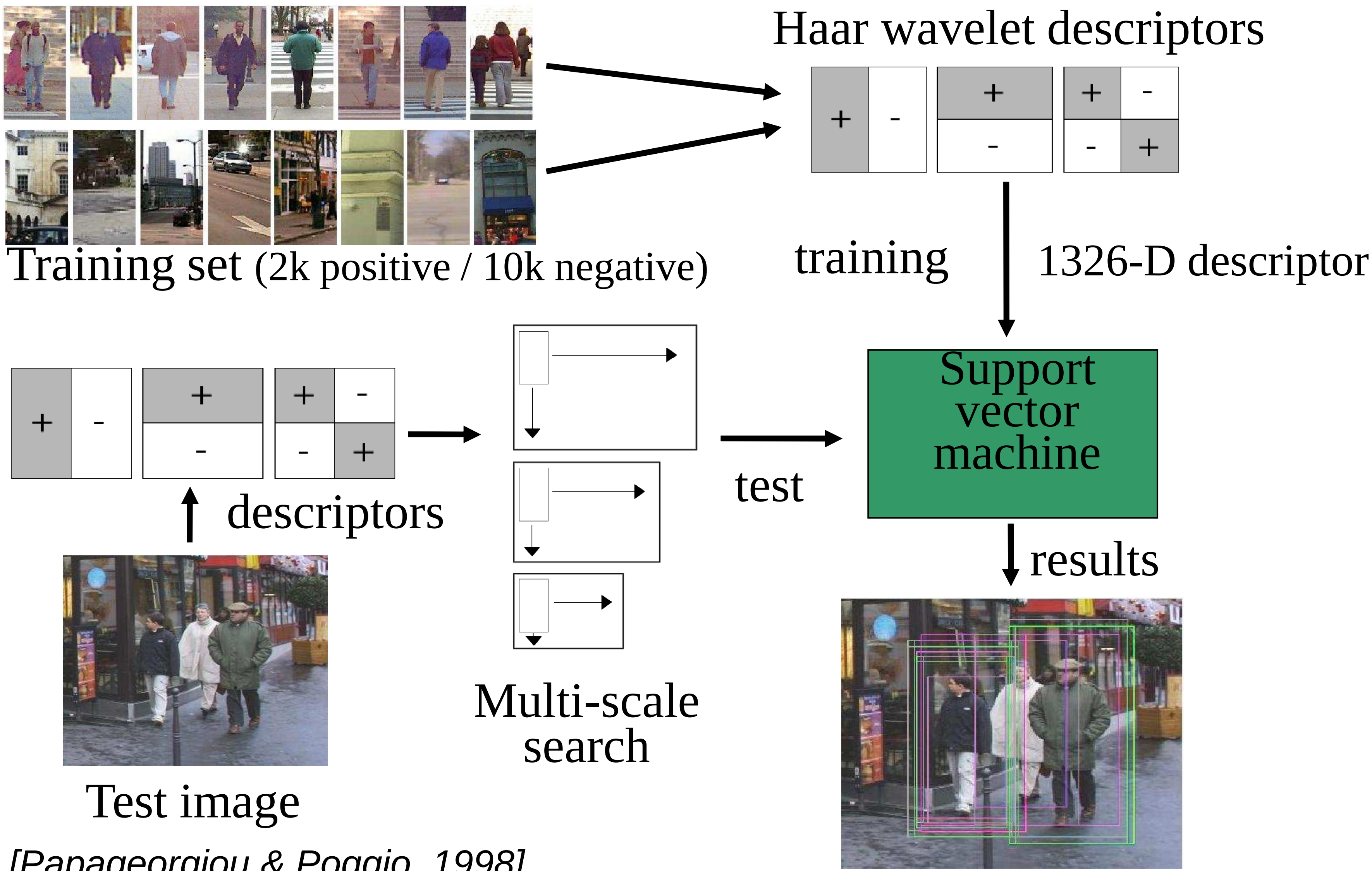
Object detections with bounding boxes

Scale-space pyramid



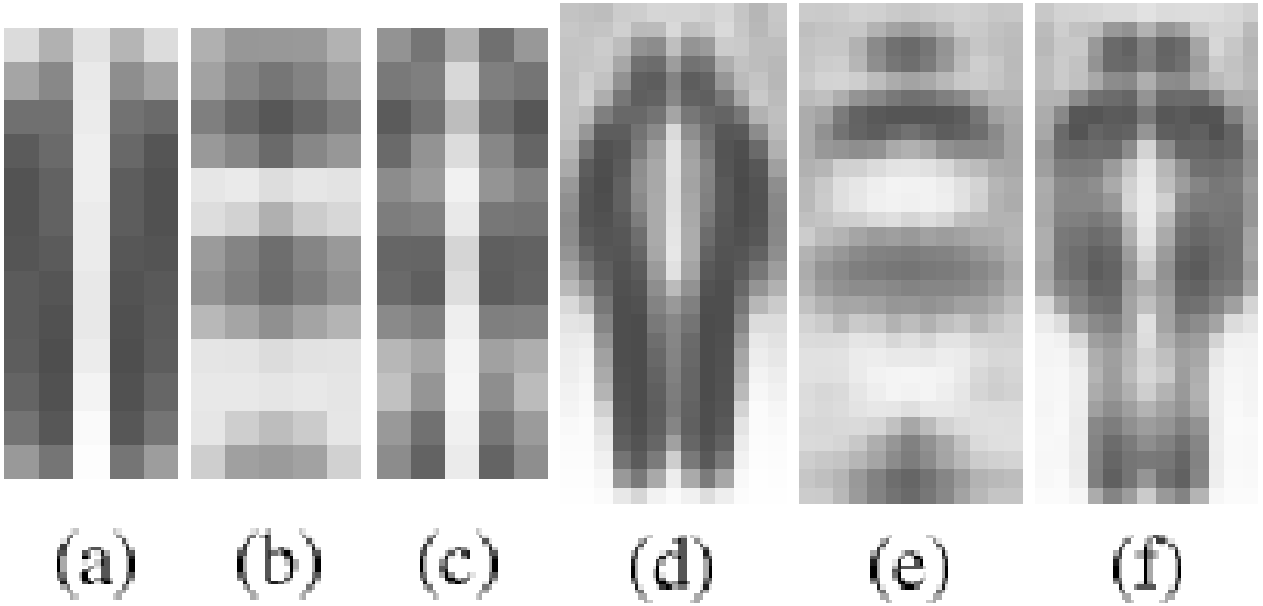
Detection window

Haar Wavelet / SVM Human Detector



[Papageorgiou & Poggio, 1998]

Which Descriptors are Important?

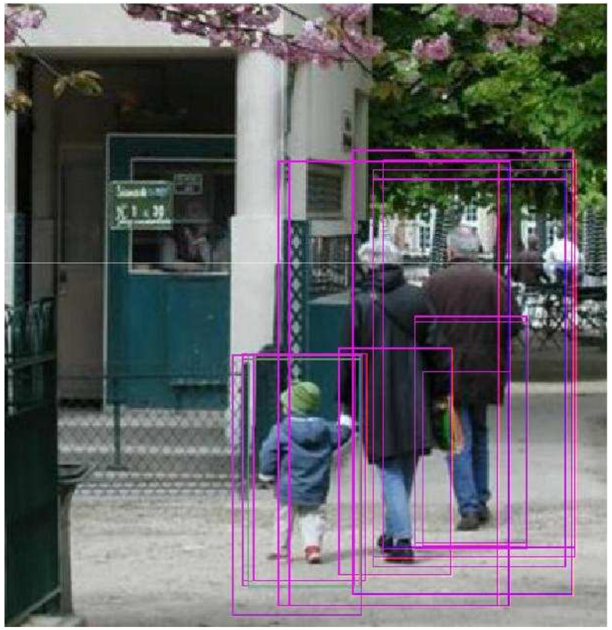


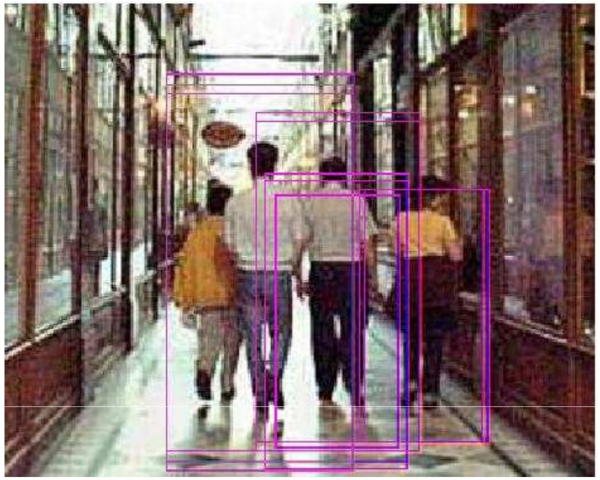
32x32 descriptors *16x16 descriptors*

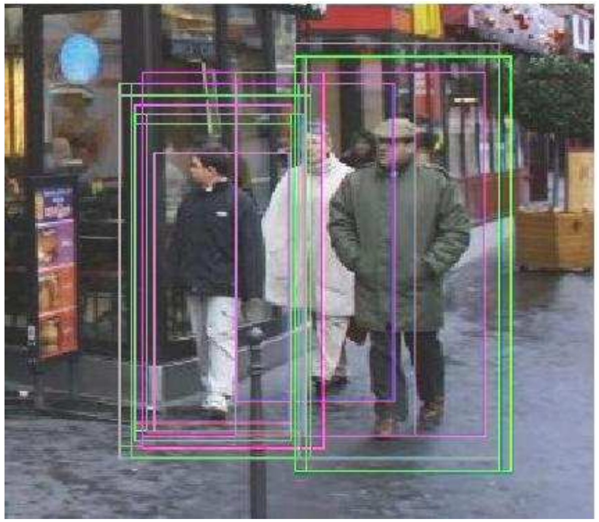
Mean response difference between positive & negative training examples

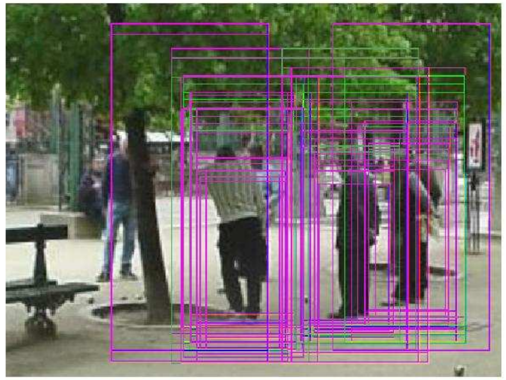
Essentially just a coarse-scale human silhouette template!

Some Detection Results





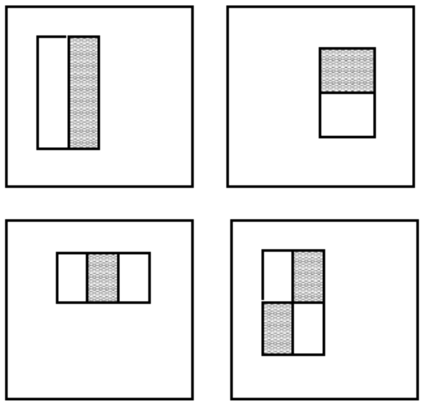




The Viola/Jones Face Detector

- A seminal approach to real-time object detection
  - Training is slow, but detection is very fast

- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows



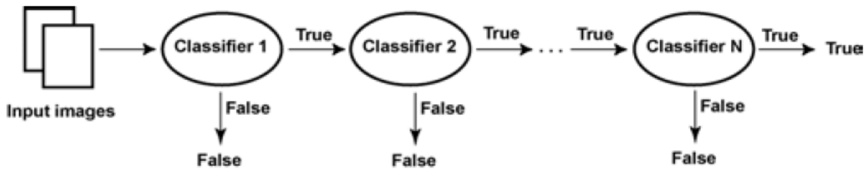
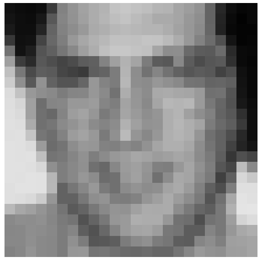
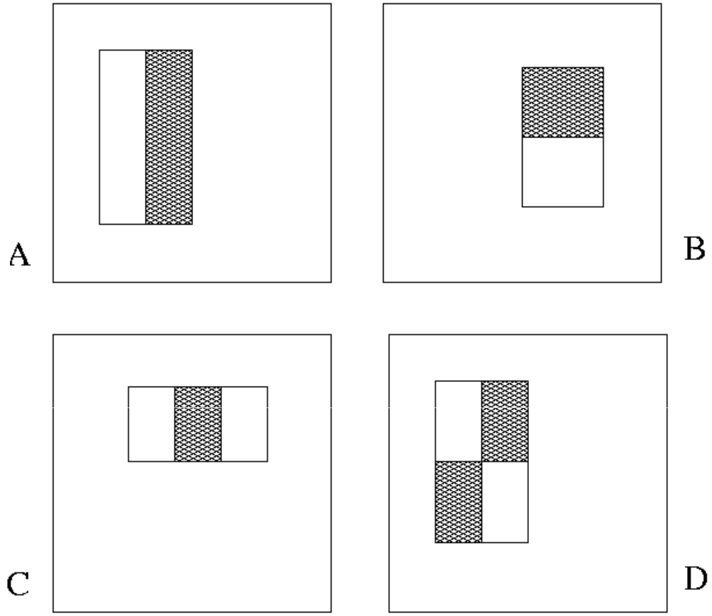


Image Features

"Rectangle filters"



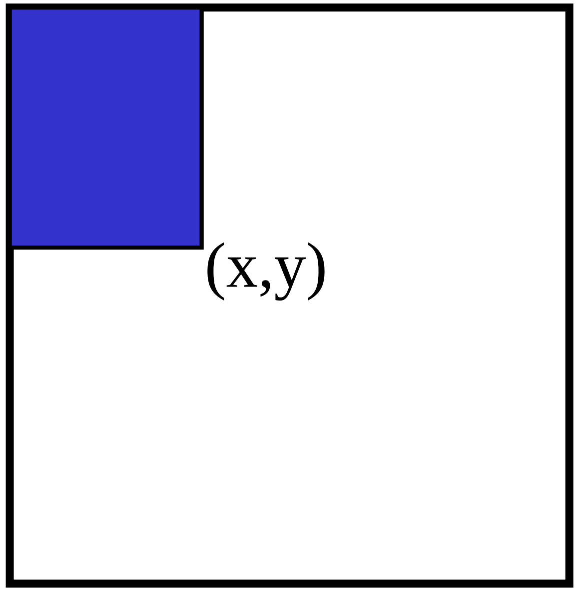


Value =

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

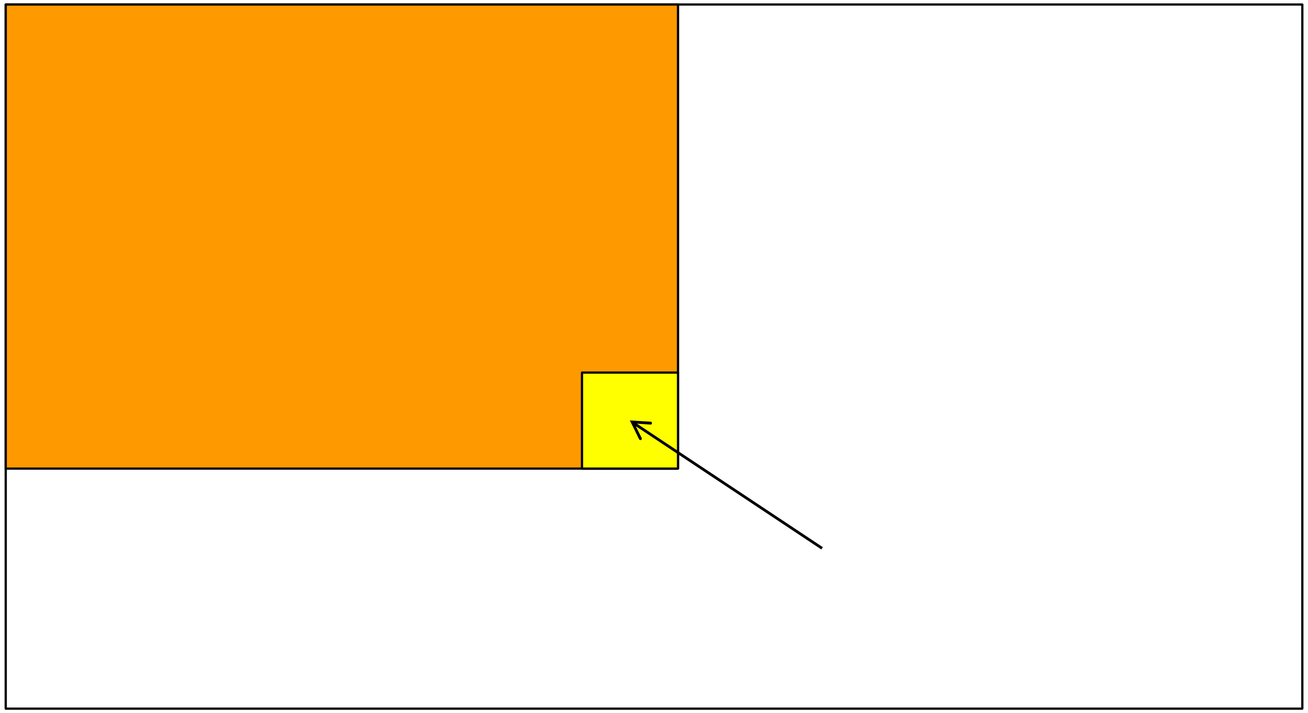
Fast computation with integral images

- The *integral image* computes a value at each pixel (x,y) that is the sum of the pixel values above and to the left of (x,y), inclusive

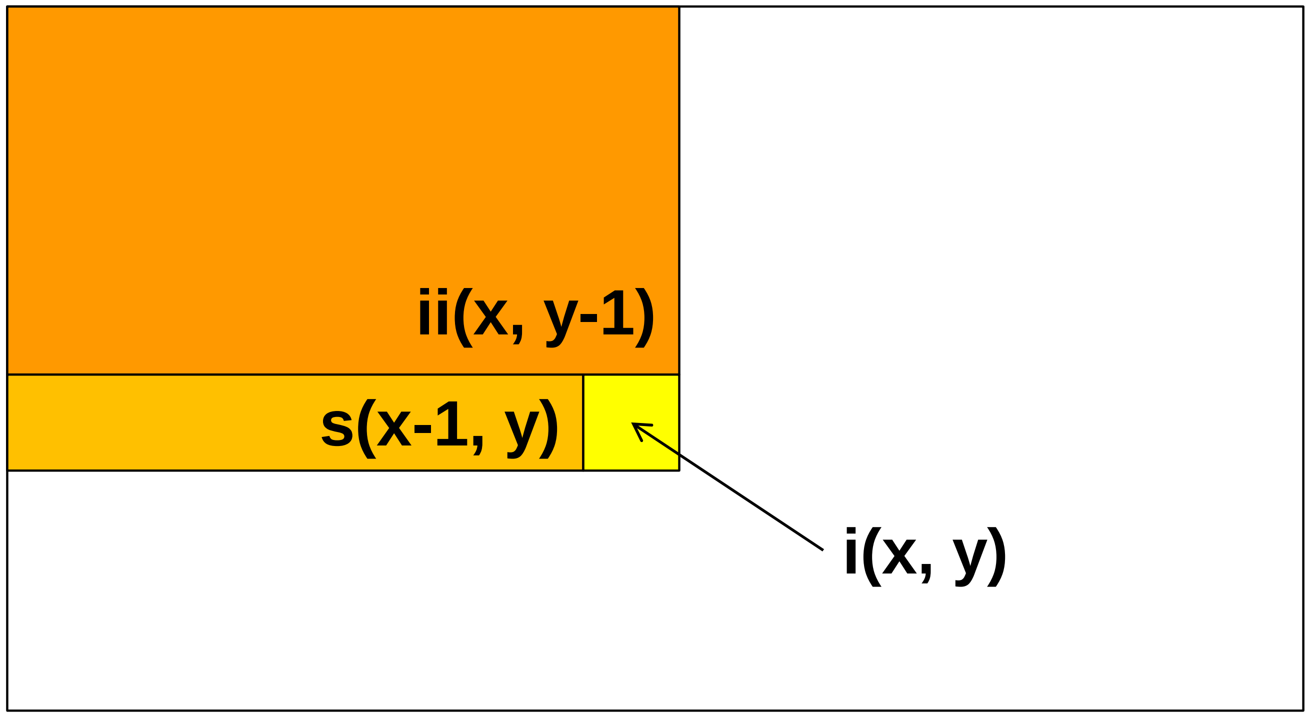


- This can quickly be computed in one pass through the image

Computing the integral image



Computing the integral image



Cumulative row sum: $s(x, y) = s(x–1, y) + i(x, y)$

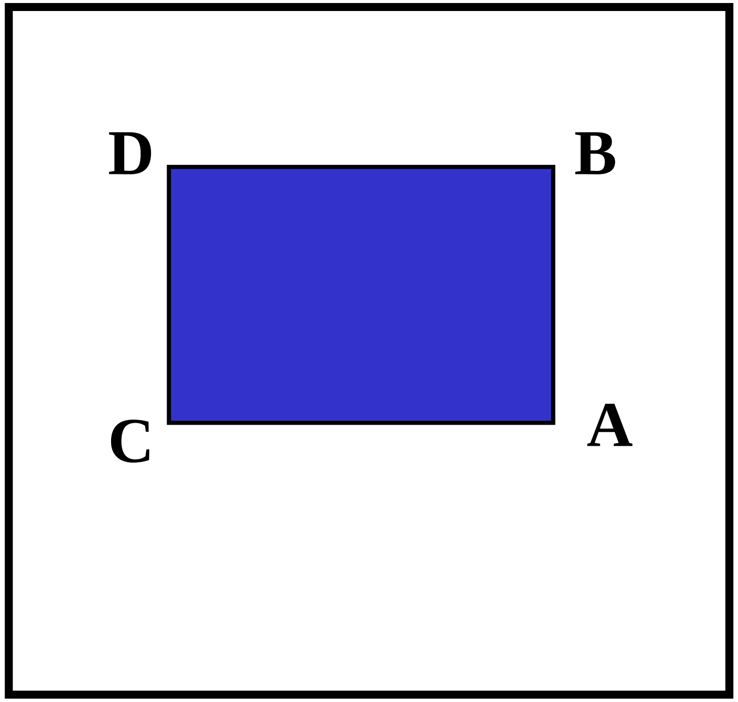
Integral image: $ii(x, y) = ii(x, y−1) + s(x, y)$

Computing sum within a rectangle

- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

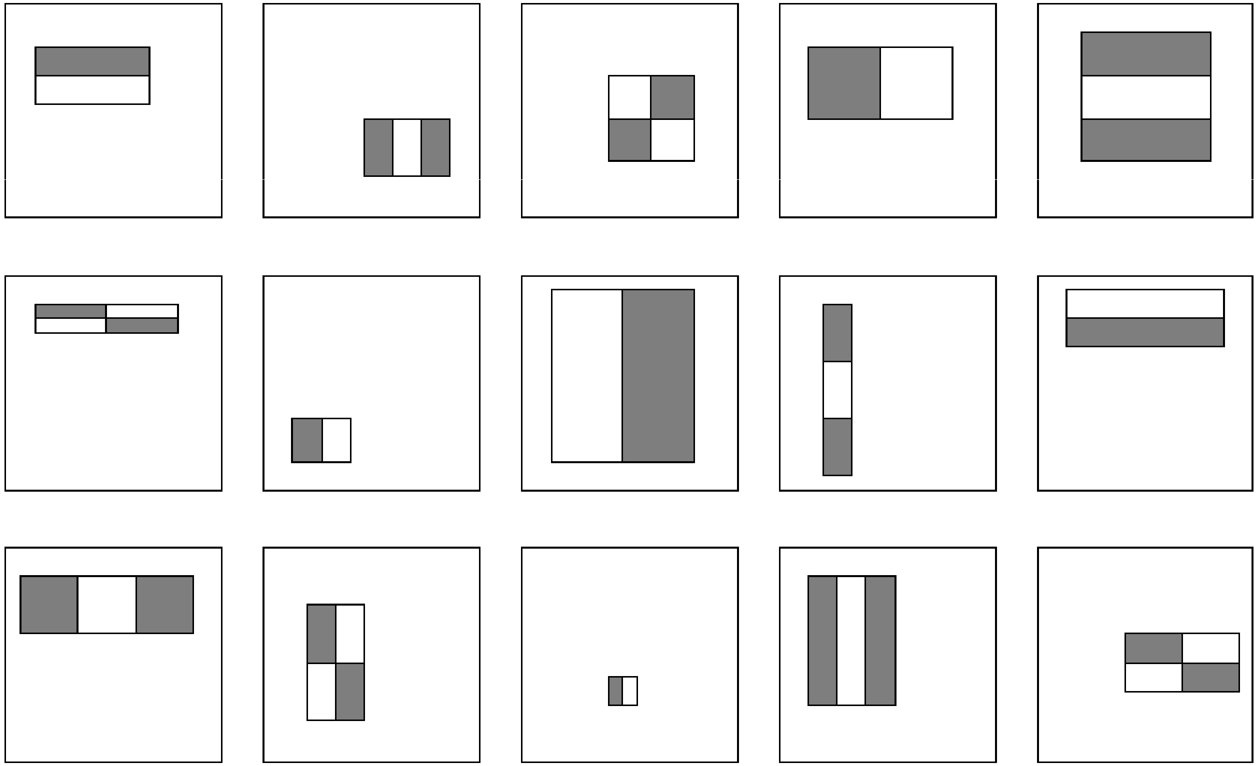
  sum = A – B – C + D

- Only 3 additions are required for any size of rectangle!



Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!



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?

Boosting

- Boosting is a classification scheme that works by combining *weak learners* into a more accurate ensemble classifier

- Training consists of multiple *boosting rounds*
  - During each boosting round, we select a weak learner that does well on examples that were hard for the previous weak learners
  - “Hardness” is captured by weights attached to training examples

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

Training procedure

- Initially, weight each training example equally

- In each boosting round:
  - Find the weak learner that achieves the lowest *weighted* training error
  - Raise the weights of training examples misclassified by current weak learner

- Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)
  - Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost)

Boosting vs. SVM

- Advantages of boosting
  - Integrates classifier training with feature selection
  - Flexibility in the choice of weak learners, boosting scheme
  - Testing is very fast
- Disadvantages
  - Needs many training examples
  - Training is slow
  - Often doesn't work as well as SVM (especially for many-class problems)

Boosting for face detection

- Define weak learners based on rectangle features

value of rectangle feature

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

window

parity

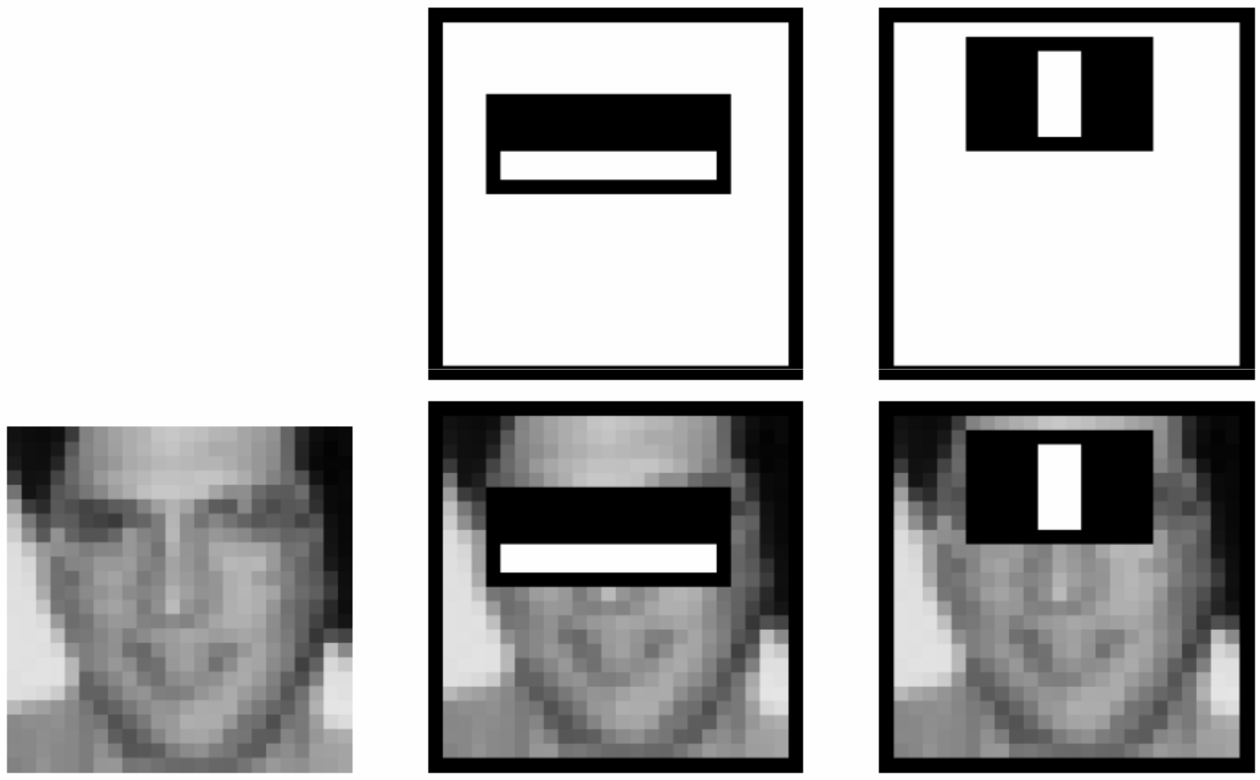
threshold

Boosting for face detection

- Define weak learners based on rectangle features
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Select best filter/threshold combination based on weighted training error
  - Reweight examples

Boosting for face detection

- First two features selected by boosting:



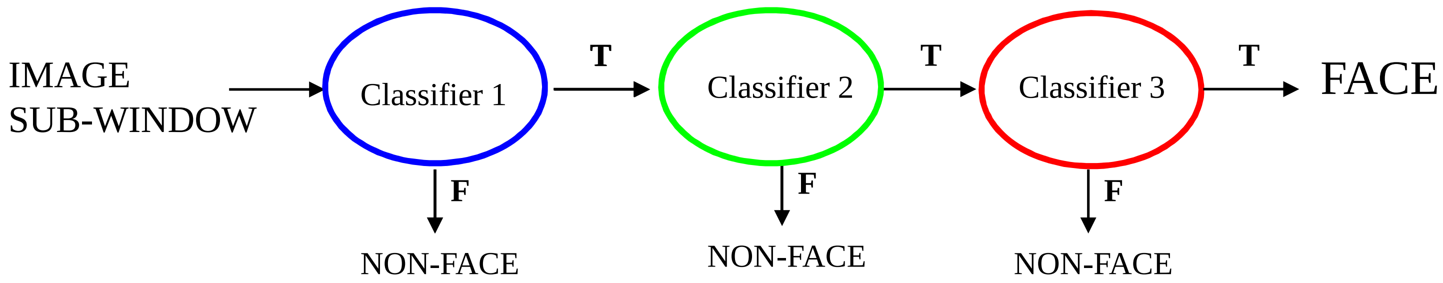
This feature combination can yield 100% detection rate and 50% false positive rate

Attentional cascade

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows

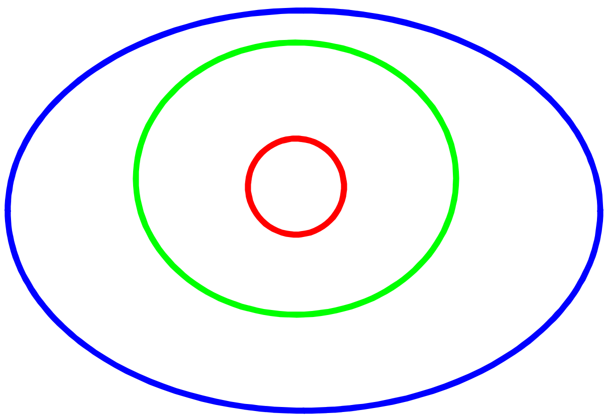
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on

- A negative outcome at any point leads to the immediate rejection of the sub-window

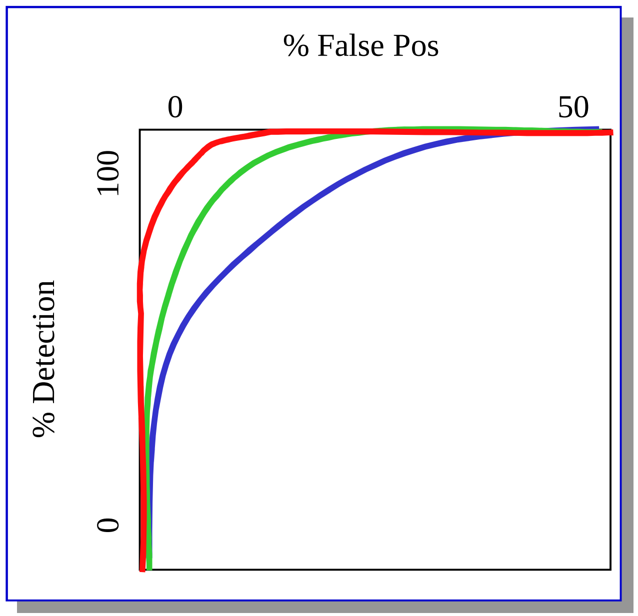


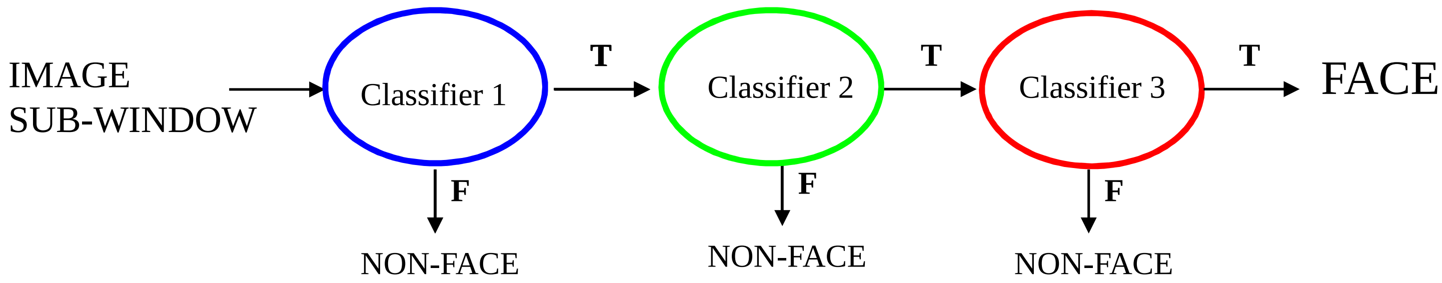
Attentional cascade

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



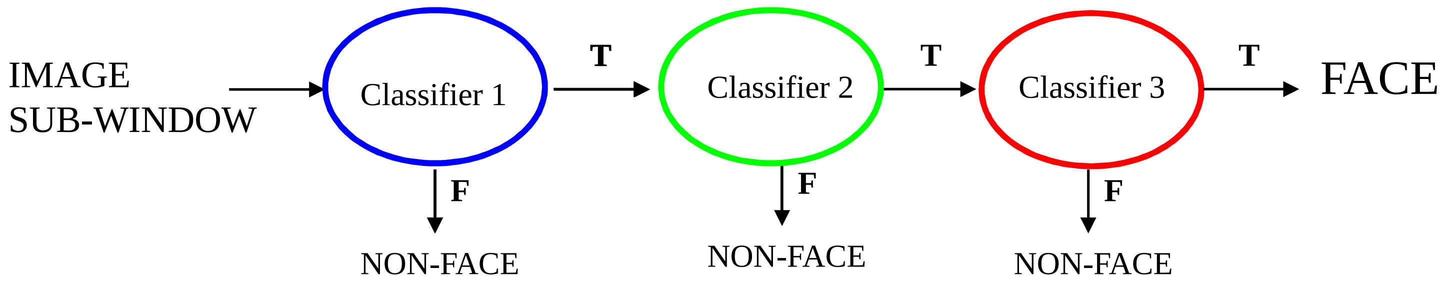
Receiver operating characteristic





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 ($0.3^{10} \approx 6\times10^{-6}$)



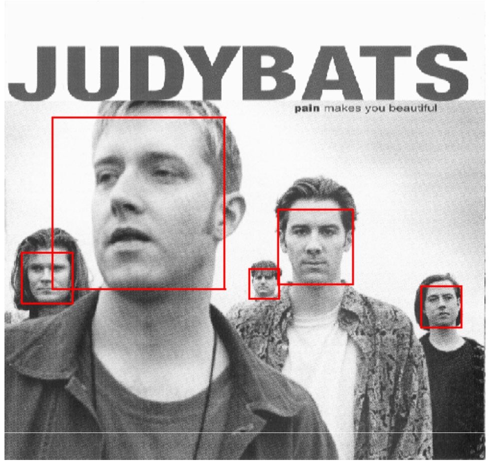
Training the cascade

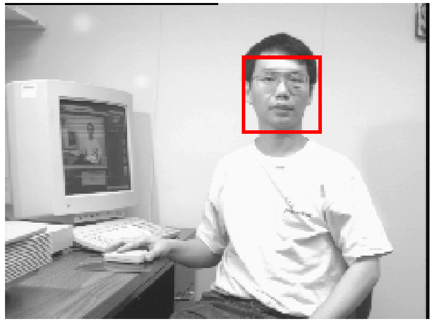
- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
  - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
  - Test on a *validation set*
- If the overall 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

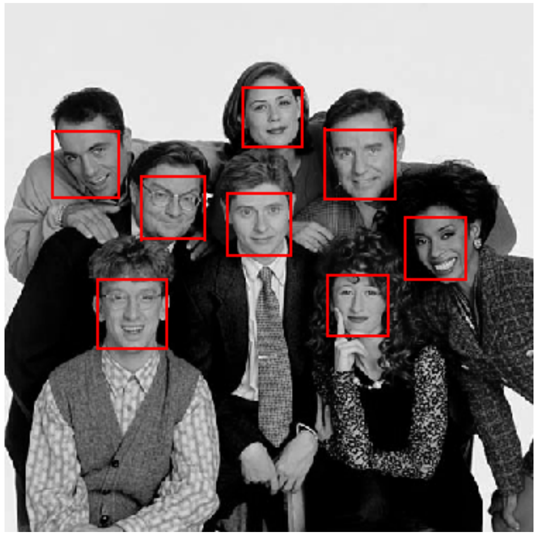
The implemented system

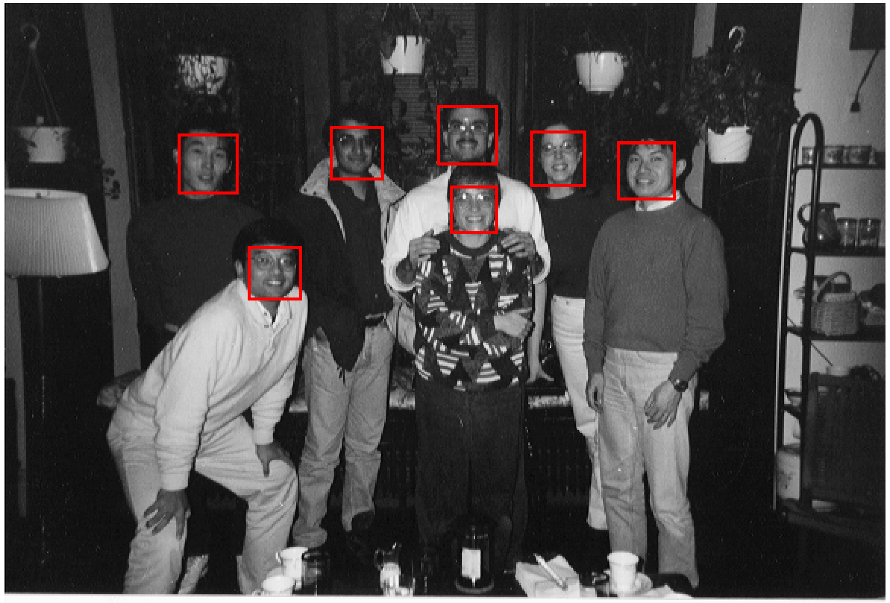
- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

Result of Face Detector on Test Images

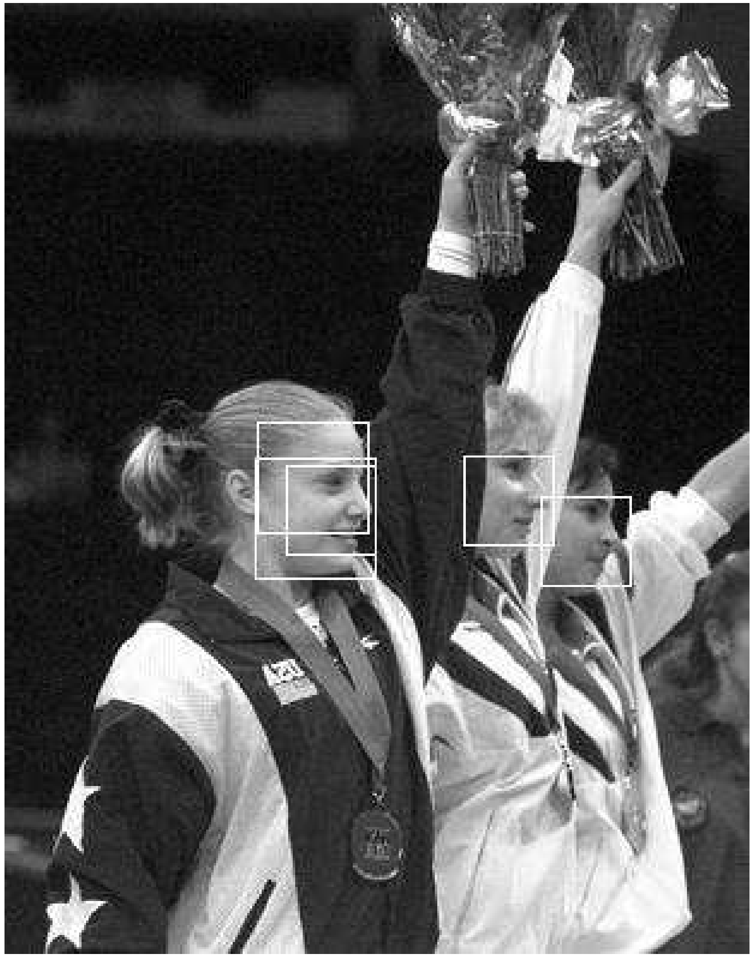




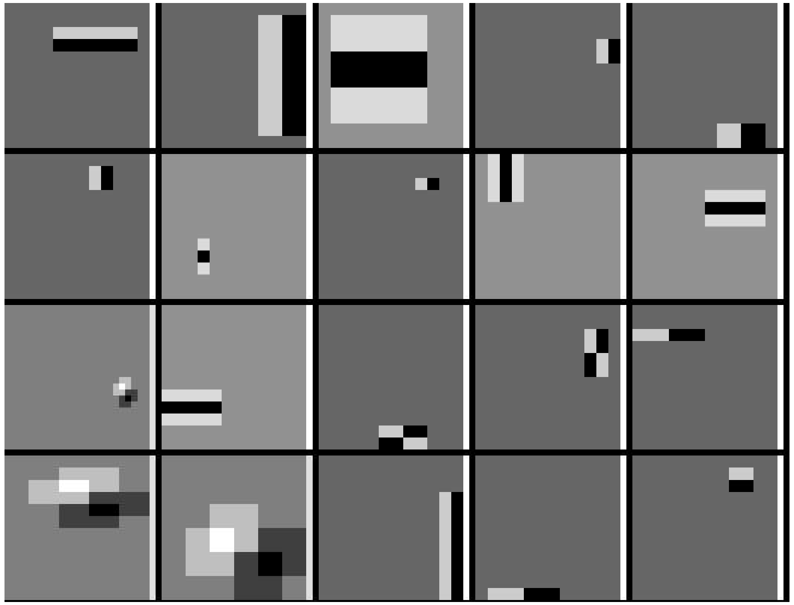




Profile Detection



Profile Features



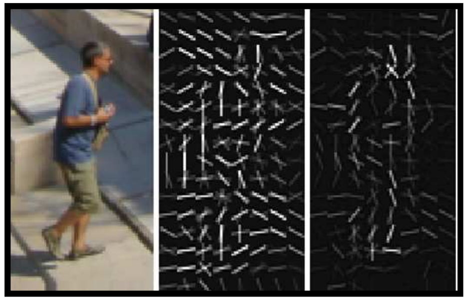
Summary: Viola/Jones detector

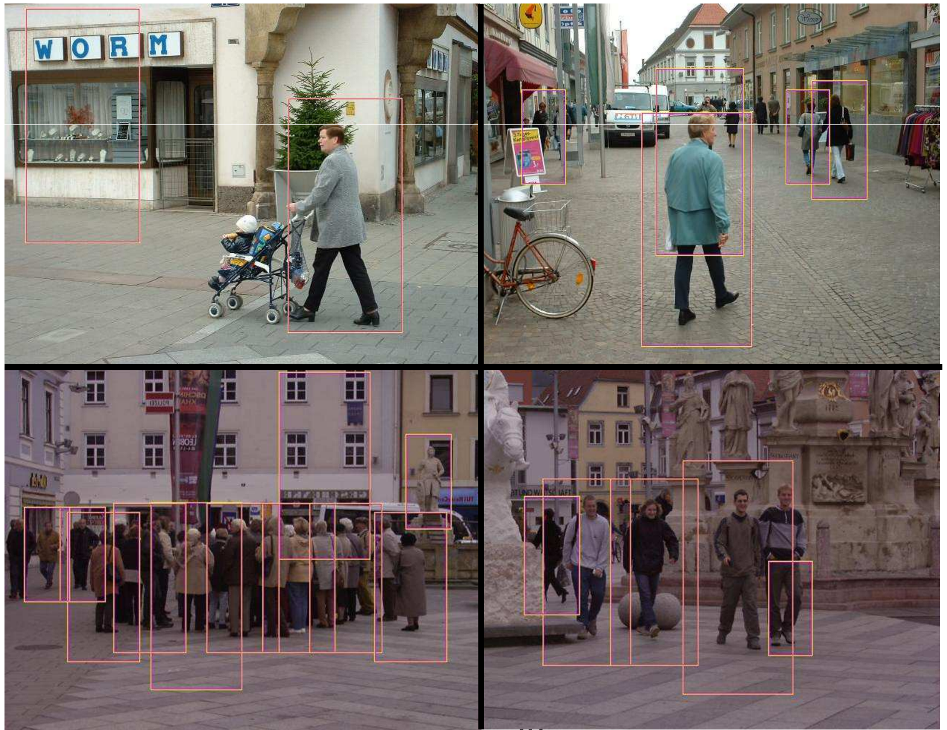
- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows
- Available in open CV

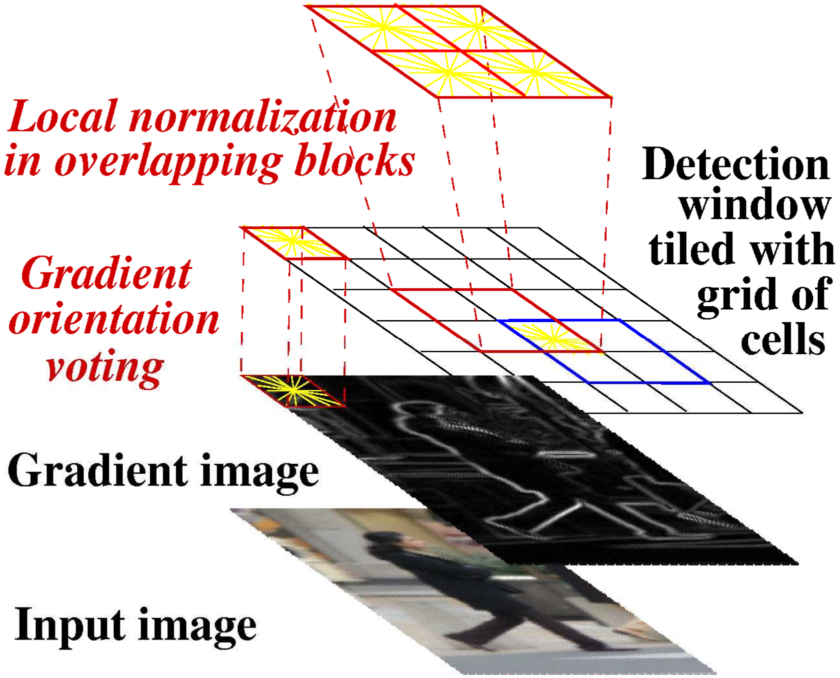
Histogram of Oriented Gradient Human Detector

- Descriptors are a grid of local Histograms of Oriented Gradients (HOG)
- Linear SVM for runtime efficiency
- Tolerates different poses, clothing, lighting and background
- Assumes upright fully visible people

Importance weighted responses







[Dalal & Triggs, CVPR 2005]

Human detection

Two layer detection [Harzallah et al. 2009]

- Combination of a linear with a non-linear SVM classifier
  - Linear classifier is used to preselection
  - Non-linear one for scoring

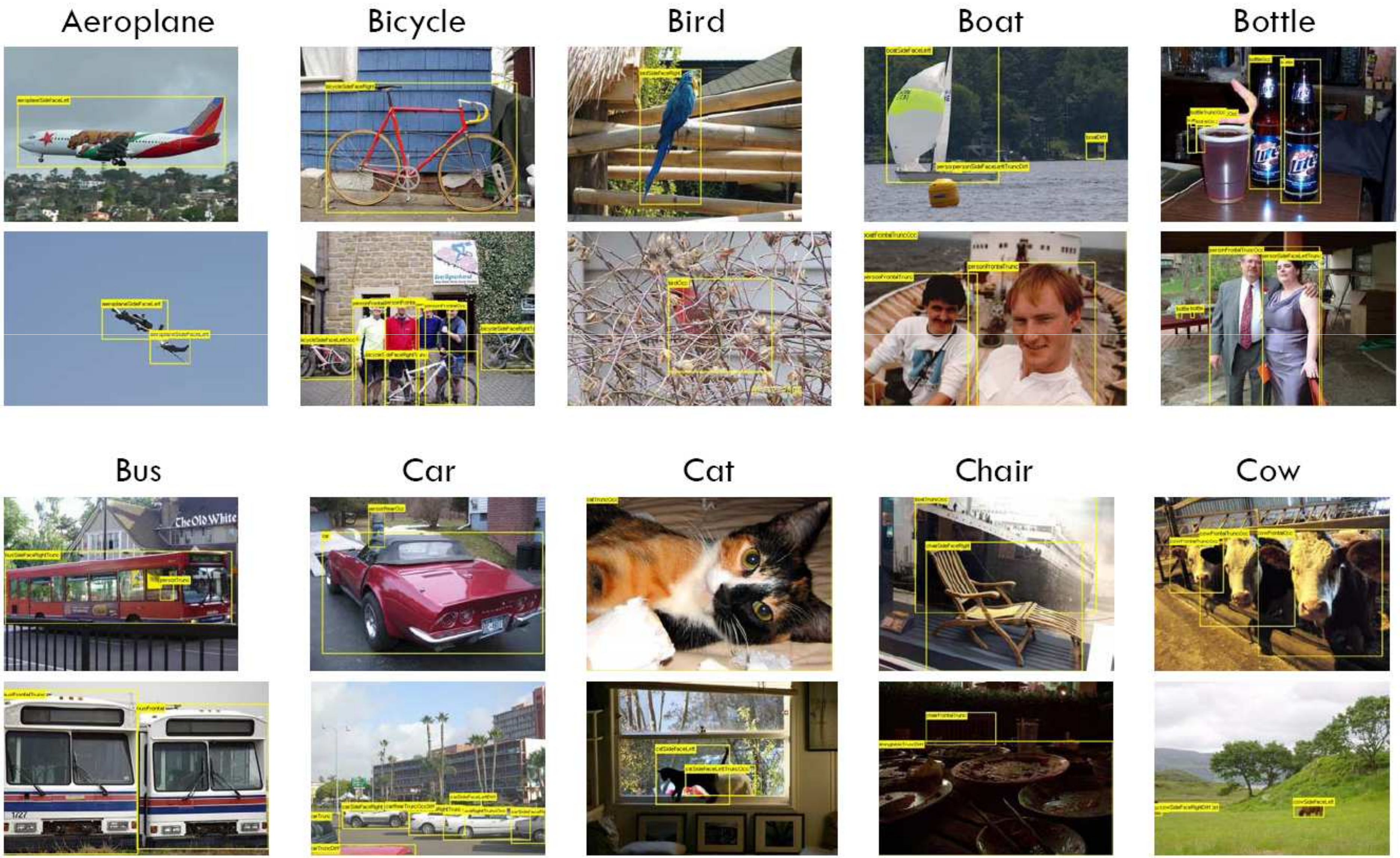
- Use of image classification for context information

- Winner of 11/20 classes in the PASCAL Visual Object Classes Challenge 2008 (VOC 2008)

PASCAL VOC 2008 dataset

- 8465 image (4332 training and 4133 test) downloaded from Flickr, manually annotated
- 20 object classes (aeroplane, bicycle, bird, etc.)
- Between 130 and 832 images per class (except person 3828)
- On average 2-3 objects per image
- Viewpoint information : front, rear, left, right, unspecified
- Other information : truncated, occluded, difficult

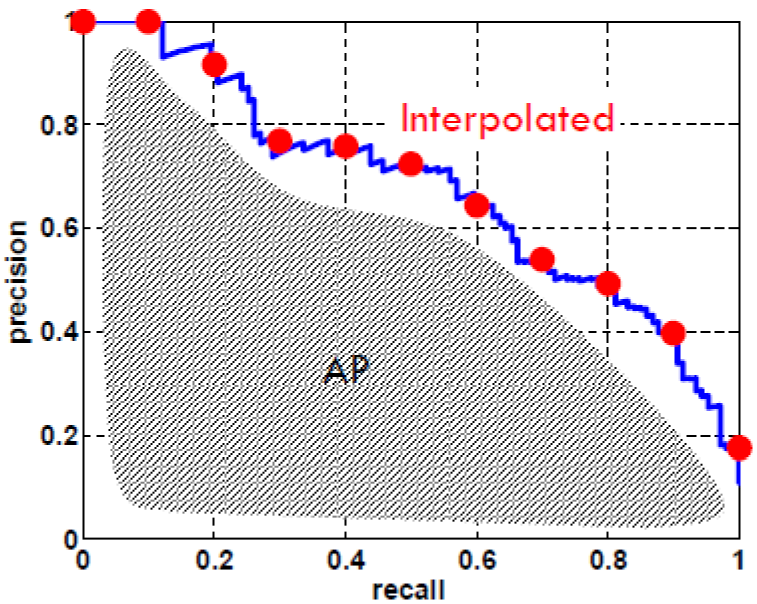
PASCAL 2008 dataset



PASCAL 2008 dataset

Evaluation

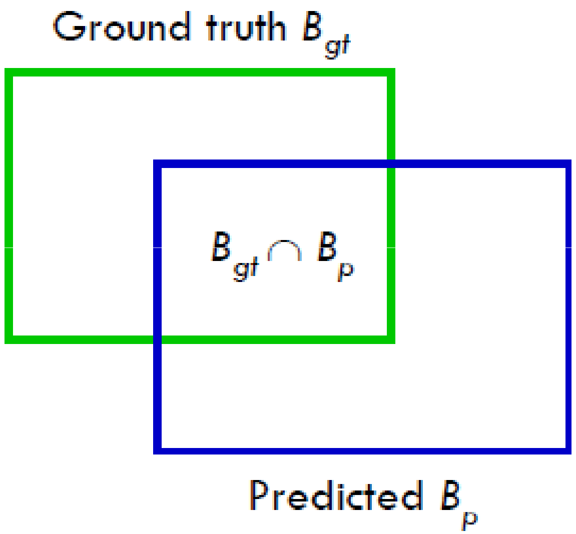
- Average Precision [TREC] averages precision over the entire range of recall
  - Curve interpolated to reduce influence of “outliers”



- A good score requires both high recall and high precision
- Application-independent
- Penalizes methods giving high precision but low recall

Evaluating bounding boxes

- Area of Overlap (AO) Measure



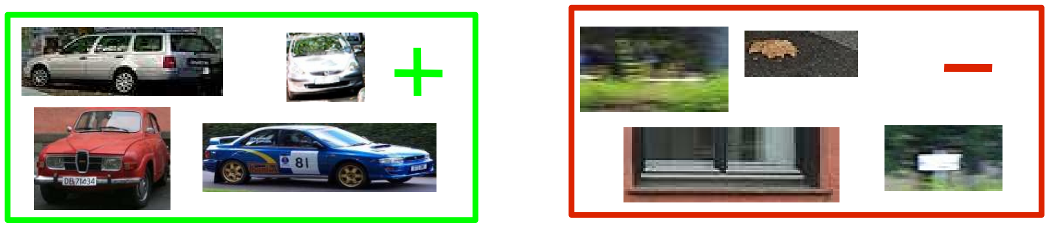
$$AO(B_{gt}, B_p) = \frac{|B_{gt} \bigcap B_p|}{|B_{gt} \bigcup B_p|}$$

- Need to define a threshold t such that $AO(B_{gt}, B_p)$ implies a correct detection: 50%

Introduction [Harzallah et al. 2000]

- Method with sliding windows (Each window is classified as containing or not the targeted object)

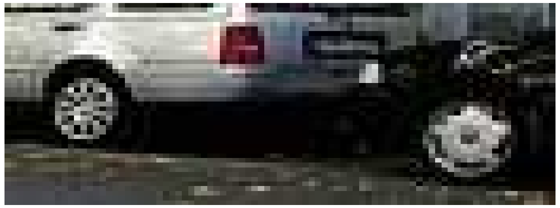
- Learn a classifier by providing positive and negative examples



Generating training windows

- Adding positive training examples by shifting and scaling the original annotations [Laptev06]

- Initial negative examples randomly extracted from background
- Training an initial classifier
- Retraining 4 times by adding false positives

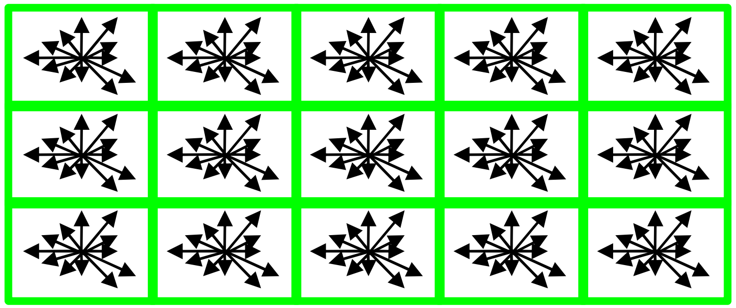



Examples of false positives

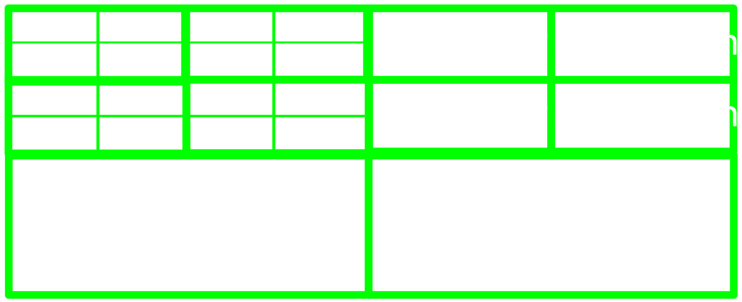
Image representation

- Combination of 2 image representations

- Histogram Oriented Gradient
  - Gradient based features
  - Integral Histograms



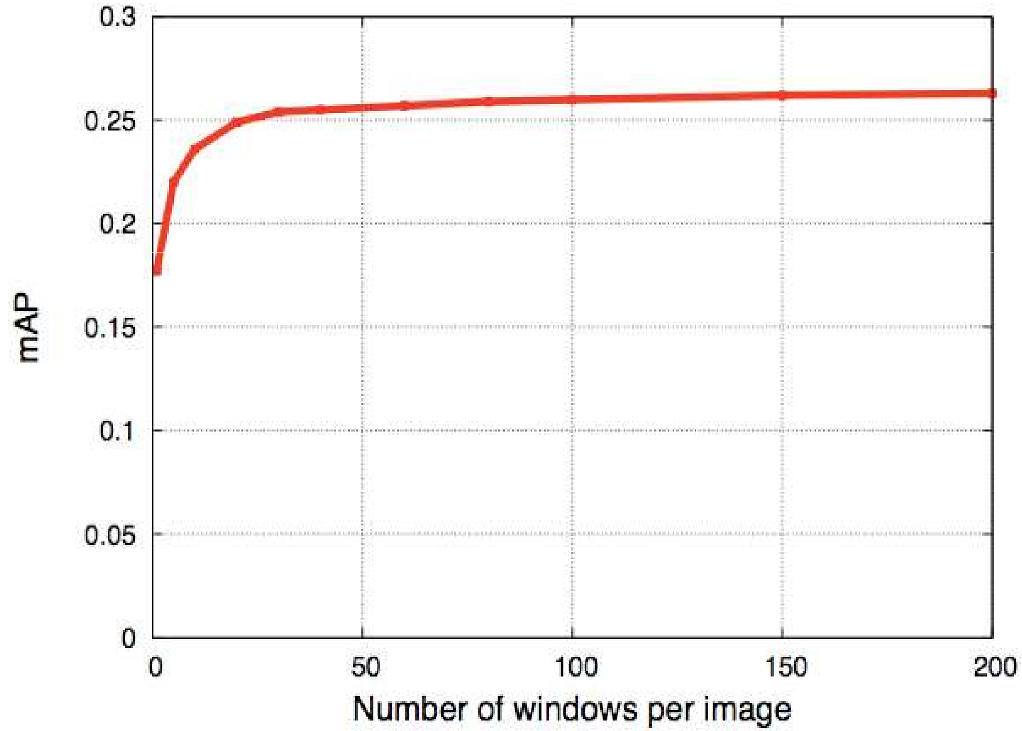
- Bag of Features
  - SIFT features extracted densely + k-means clustering
  - Pyramidal representation of the sliding windows
  - One histogram per tile



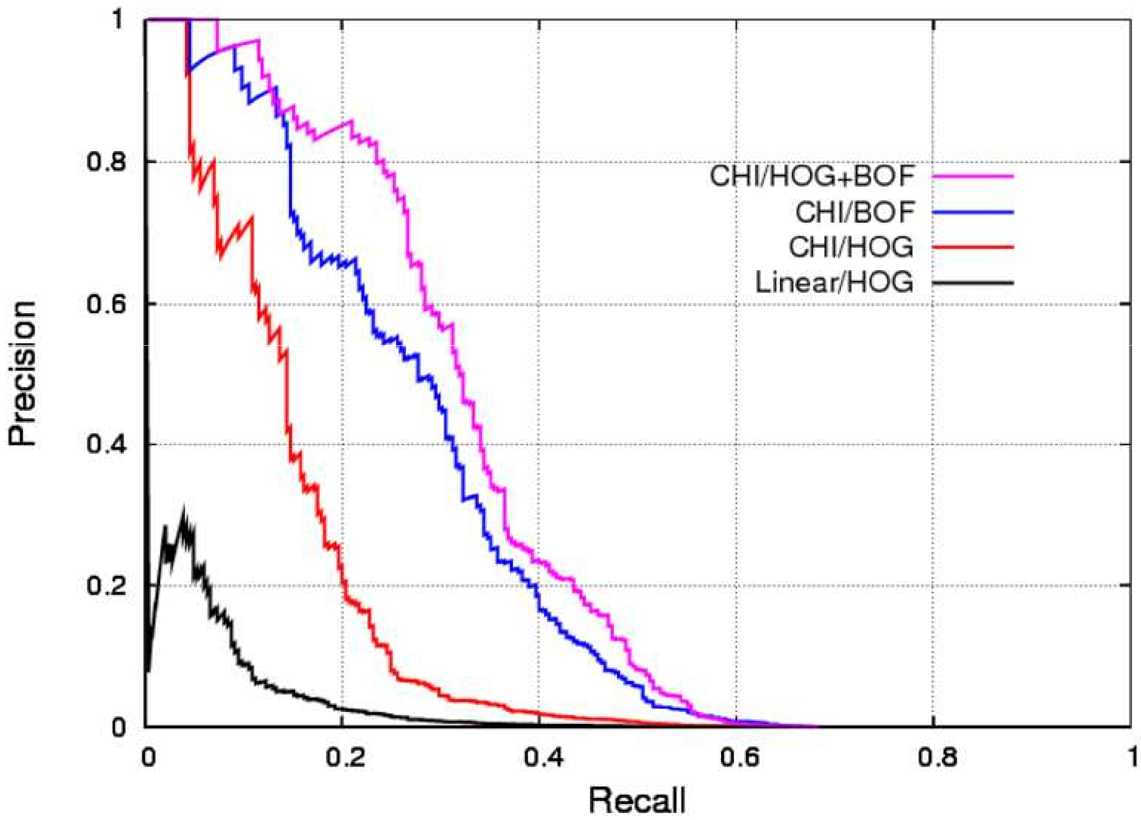
Efficient search strategy

- Reduce search complexity
  - Sliding windows: huge number of candidate windows
  - Cascade to reject windows quickly
- Two stage cascade:
  - Filtering classifier with a linear SVM
    - Low computational cost
    - Capacity of rejecting negative windows
  - Scoring classifier with a non-linear SVM
    - X^2 kernel with a channel combination [Zhang07]
    - Significant increase of performance

Efficiency of the 2 stage localization

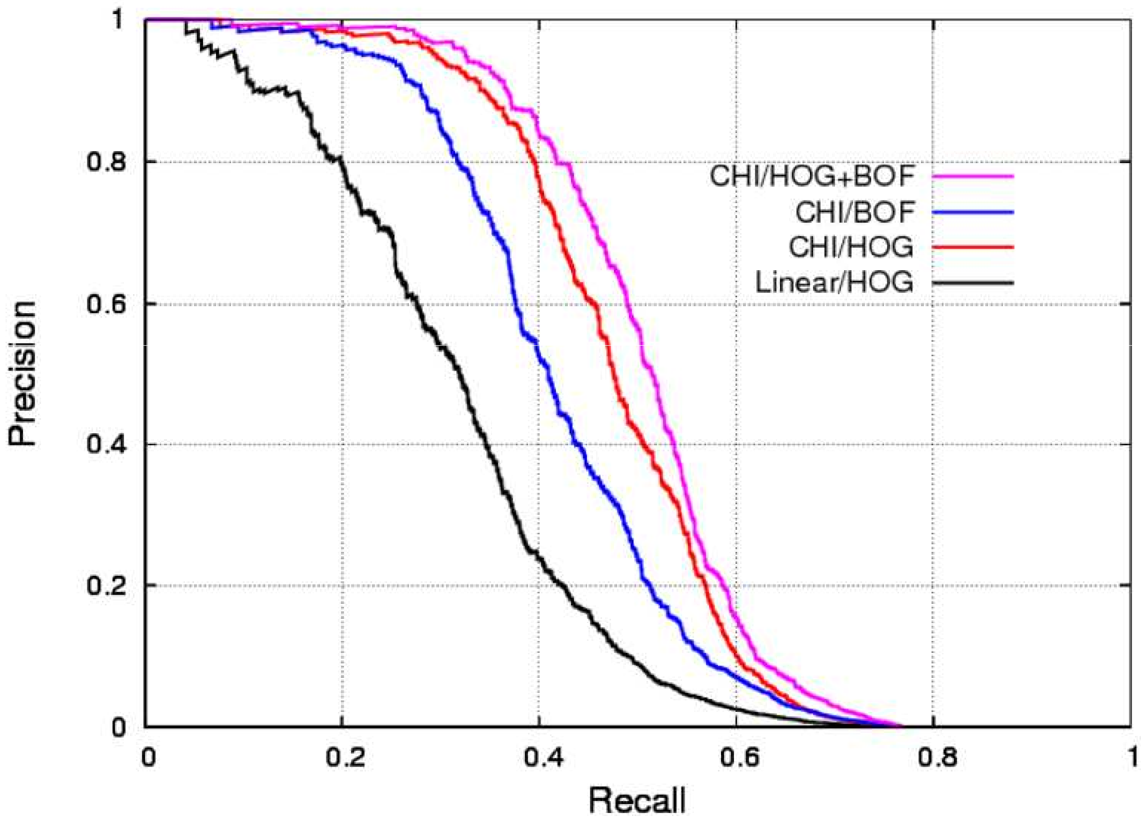


Localization performance: aeroplane



Method	AP
X^2, HOG+BOF	33.8
X^2, BOF	29.8
X^2, HOG	18.4
Linear, HOG	10.0

Localization performance: car



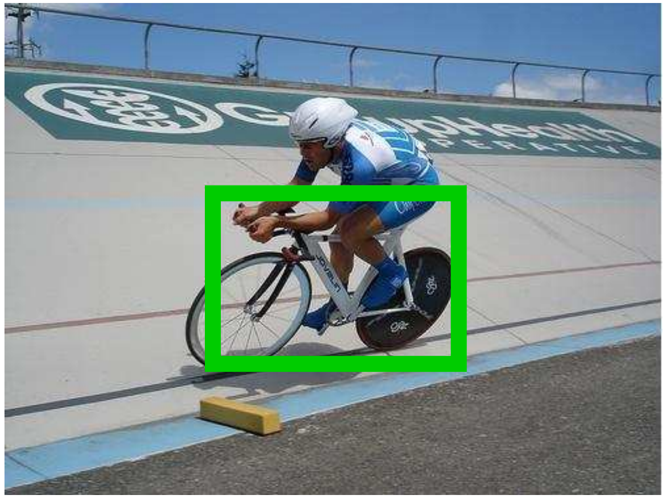
Method	AP
X^2, HOG+BOF	50.4
X^2, BOF	42.3
X^2, HOG	47.5
Linear, HOG	33.9

Localization performance

Mean Average Precision on all 20 classes, PASCAL 2007 dataset

Method	mAP
Linear, HOG	**14.6**
Linear, BOF	**15.0**
Linear, HOG+BOF	**17.6**
X^2, HOG	**21.9**
X^2, BOF	**23.1**
X^2, HOG+BOF	**26.3**

Localization examples: correct localizations



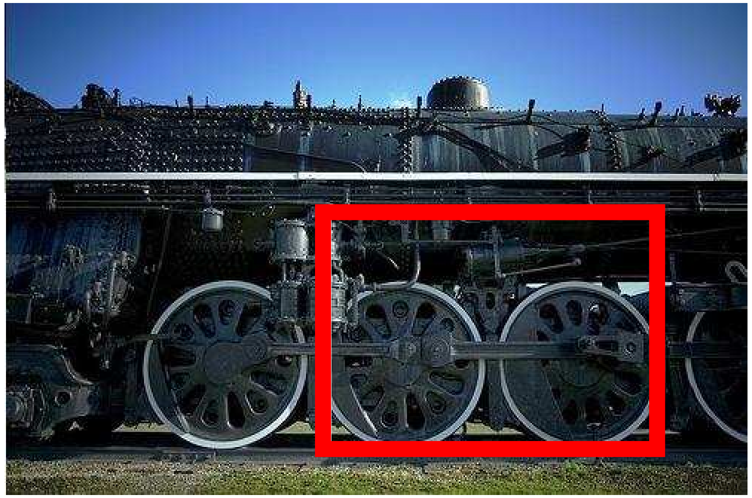
Bicycle

Car

Horse

Sofa

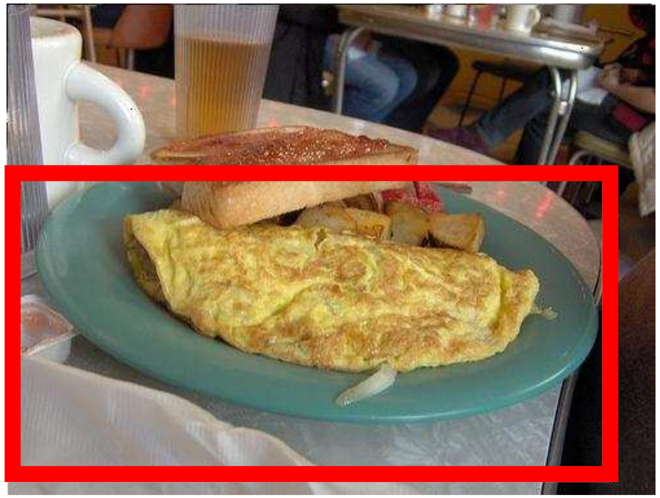
Localization examples: false positives



Bicycle

Car

Horse



Sofa

Localization examples: missed objects