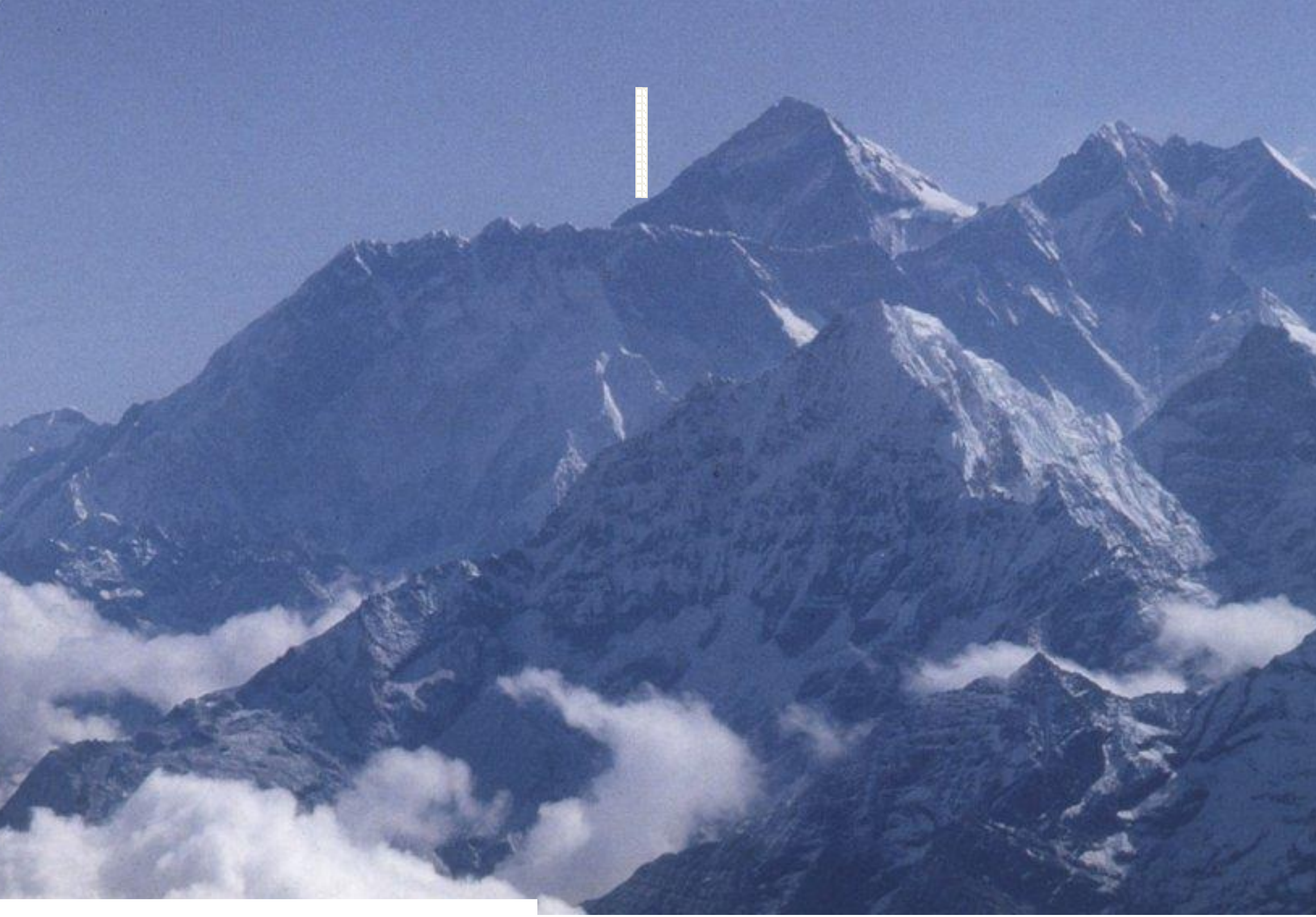

note:
black & white

Recap: Bag of Words for Large Scale Retrieval



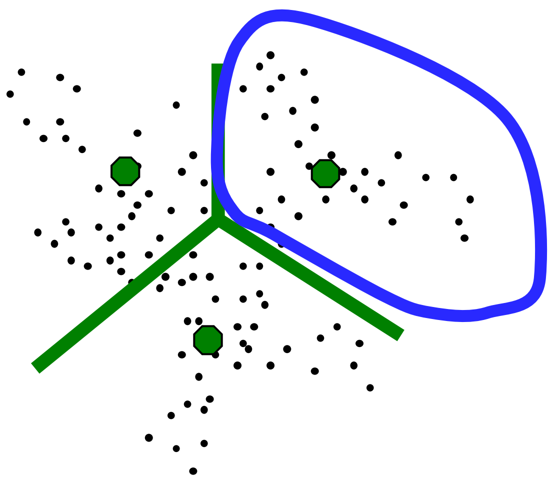
Slide Credit: Nister

Summary – large scale retrieval

- We want to do feature matching (project 2) with a billion images
- Problem: the all-pairs local feature matching is slow!
  - Solution: quantize features and build bag of feature representation. Lossy! But spatial verification can help.

Visual words

- Example: each group of patches belongs to the same visual word



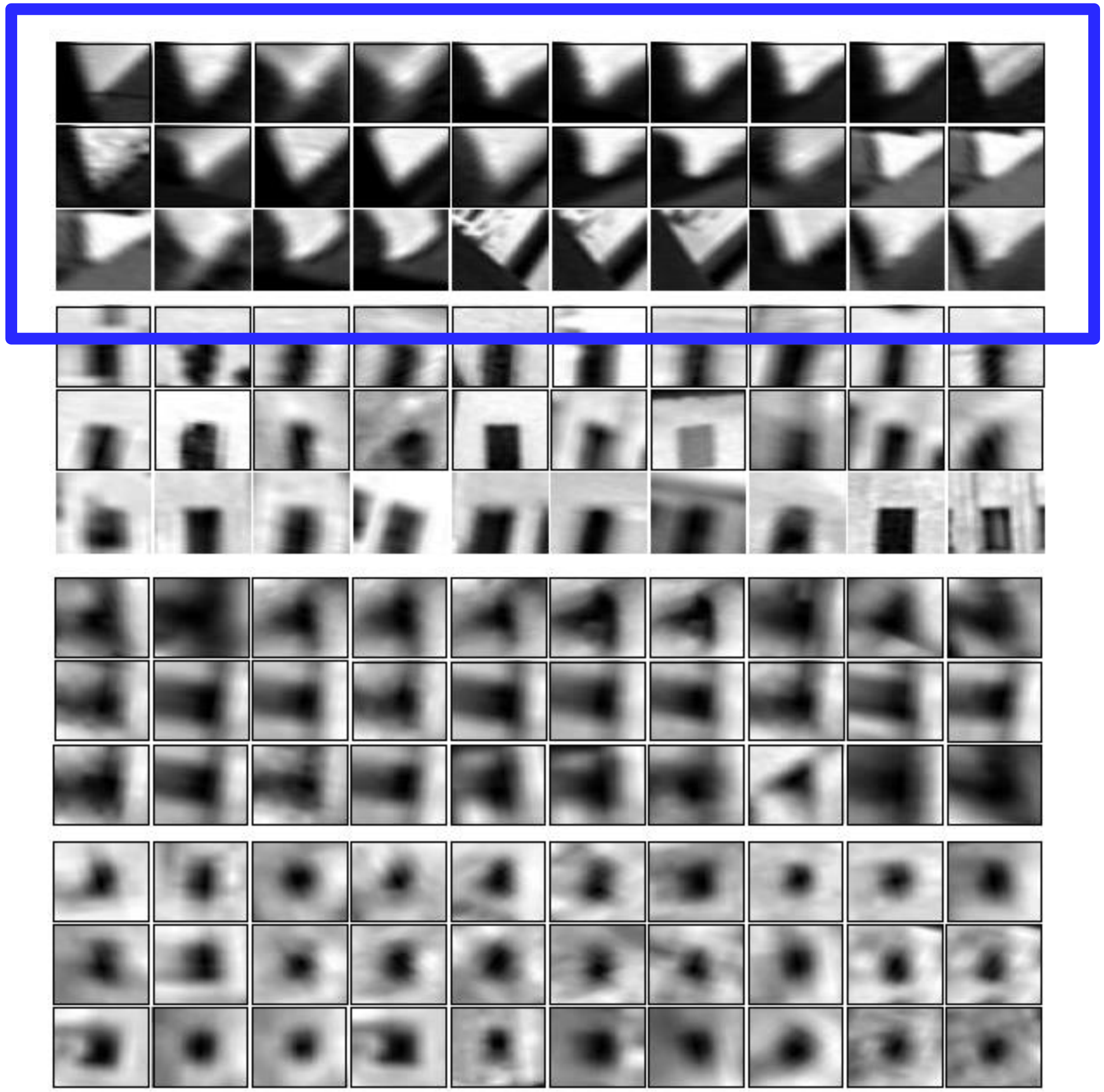


Figure from Sivic & Zisserman, ICCV 2003

Kristen Grauman

Summary – large scale retrieval

- We want to do feature matching (project 2) with a billion images
- Problem: the all-pairs local feature matching is slow!
  - Solution: quantize features and build bag of feature representation. Lossy! But spatial verification can help.
- Problem: Finding the overlap in visual words based on the Bags of Features is still too slow!
  - Solution: inverted file index, one lookup per word.

Inverted file index

New query image

Word #	Image #
1	3
2 …	
7	1, 2
8	3
9	
10 …	
91	2

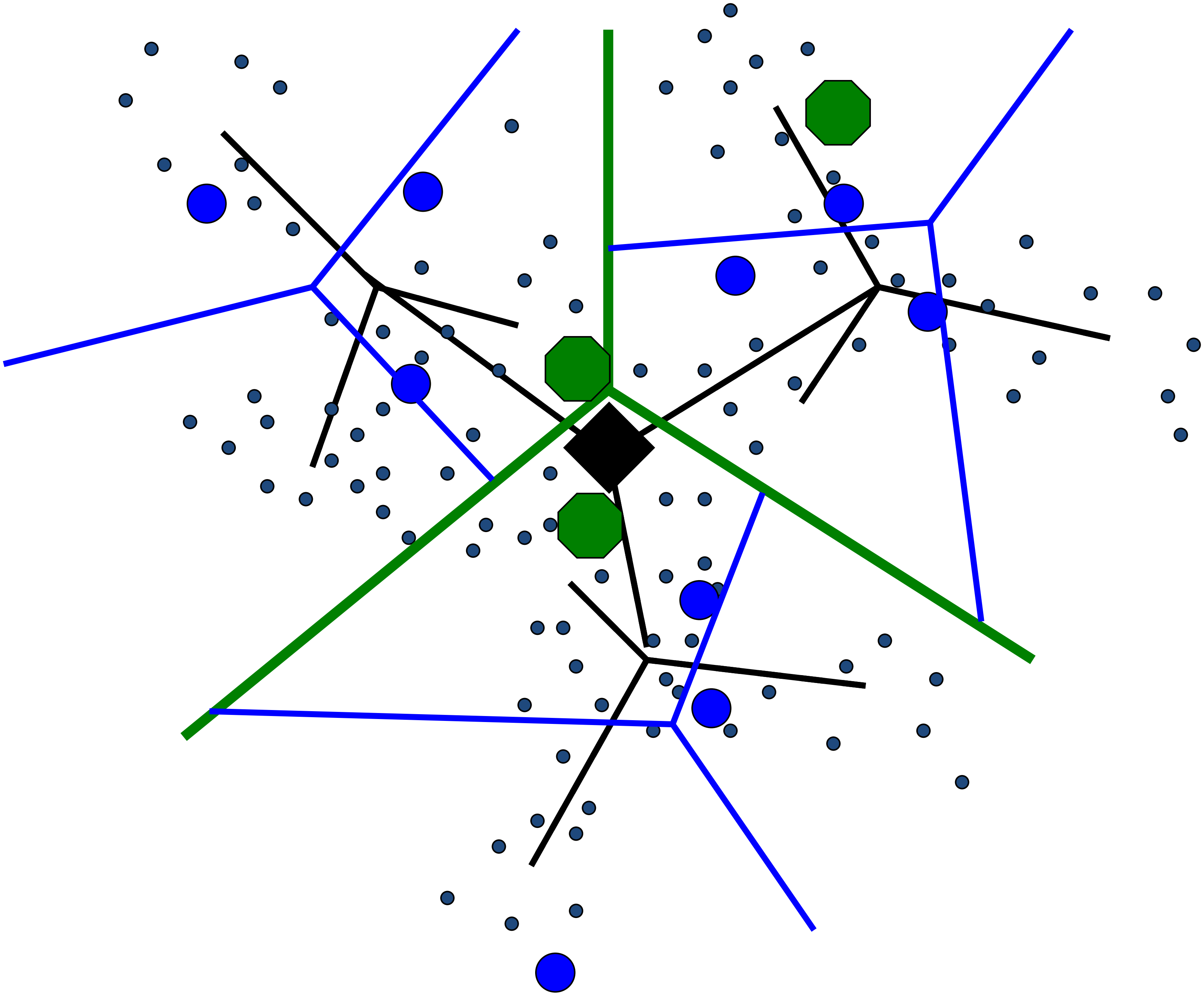
…

- New query image is mapped to indices of database images that share a word.

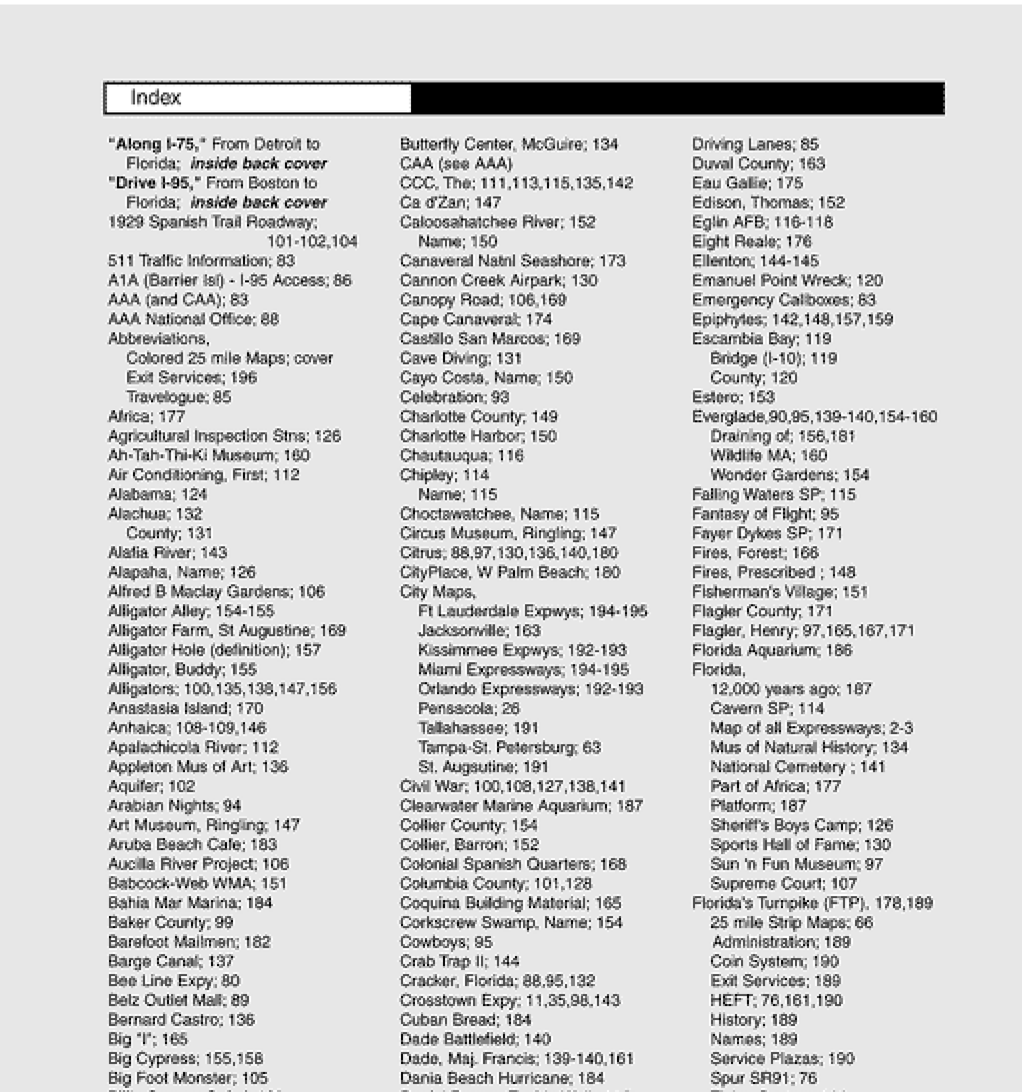
Kristen Grauman

Summary – large scale retrieval

- We want to do feature matching (project 2) with a billion images
- Problem: the all-pairs local feature matching is slow!
  - Solution: quantize features and build bag of feature representation. Lossy! But spatial verification can help.
- Problem: Finding the overlap in visual words based on the Bags of Features is still too slow!
  - Solution: inverted file index, one lookup per word.
- Problem: Even quantizing the local features into a visual word is too slow!
  - Solution: vocabulary tree. Lossy!



What else can we borrow from text retrieval?


Index

tf-idf weighting

- **T**erm **f**requency – **i**nverse **d**ocument **f**requency
- Describe frame by frequency of each word within it, downweight words that appear often in the database
- (Standard weighting for text retrieval)

Number of occurrences of word i in document d

Number of words in document d

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

Total number of documents in database

Number of documents word i occurs in, in whole database

Kristen Grauman

Query expansion

Query: ***golf green***

Results:

- How can the grass on the ***greens*** at a ***golf*** course be so perfect?
- For example, a skilled ***golf***er expects to reach the ***green*** on a par-four hole in **...**
- Manufactures and sells synthetic ***golf*** putting ***green****s* and mats.

Irrelevant result can cause a \`topic drift':

- Volkswagen ***Golf***, 1999, ***Green***, 2000cc, petrol, manual, , hatchback, 94000miles, 2.0 GTi, 2 Registered Keepers, HPI Checked, Air-Conditioning, Front and Rear Parking Sensors, ABS, Alarm, Alloy

Slide credit: Ondrej Chum

Query Expansion

Chum, Philbin, Sivic, Isard, Zisserman: Total Recall…, ICCV 2007

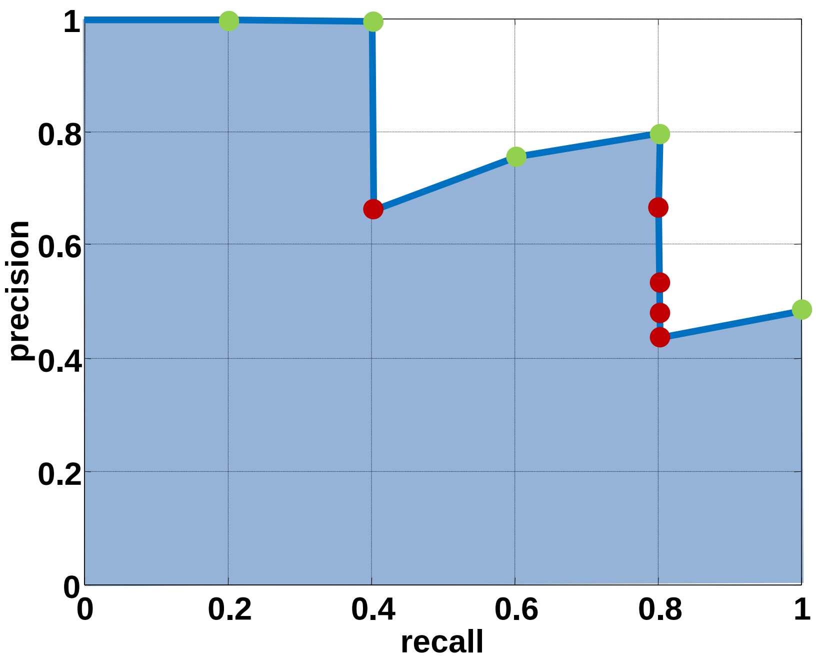
Slide credit: Ondrej Chum

Scoring retrieval quality

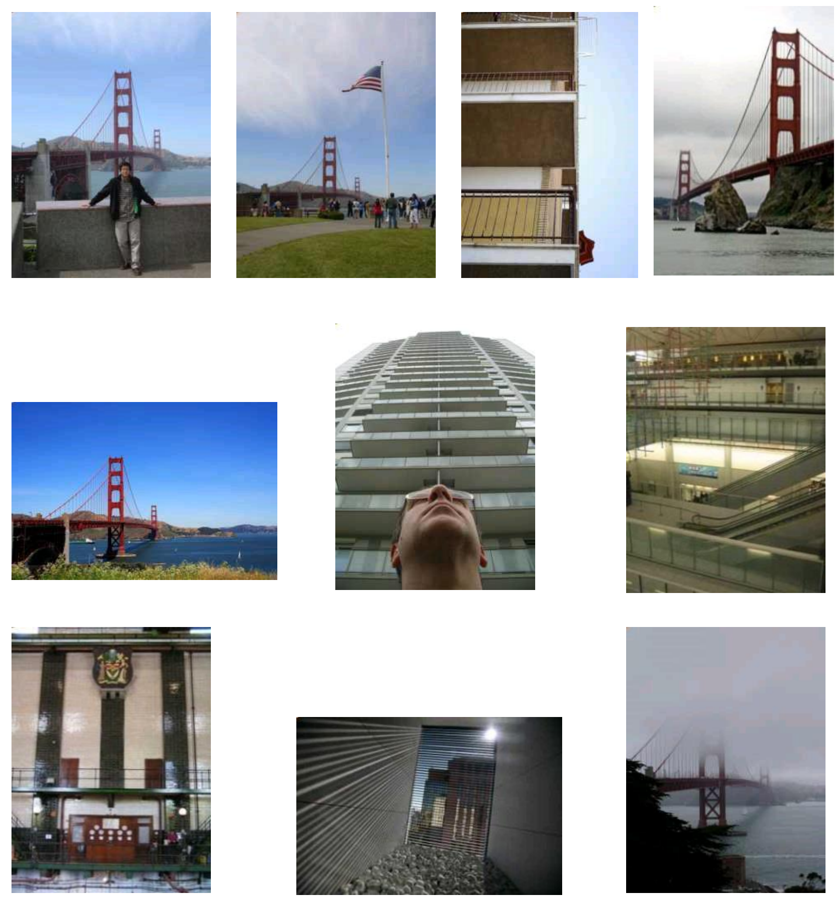
Query

Database size: 10 images
Relevant (total): 5 images

precision = #relevant / #returned
recall = #relevant / #total relevant



Results (ordered):



Slide credit: Ondrej Chum

- Sliding window detector must evaluate tens of thousands of location/scale combinations
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - A megapixel image has ~10^6 pixels and a comparable number of candidate face locations
  - To avoid having a false positive in every image image, our false positive rate has to be less than 10^{-6}

Sliding Window Face Detection with Viola-Jones

Computer Vision

CS 143, Brown

James Hays

Many Slides from Lana Lazebnik

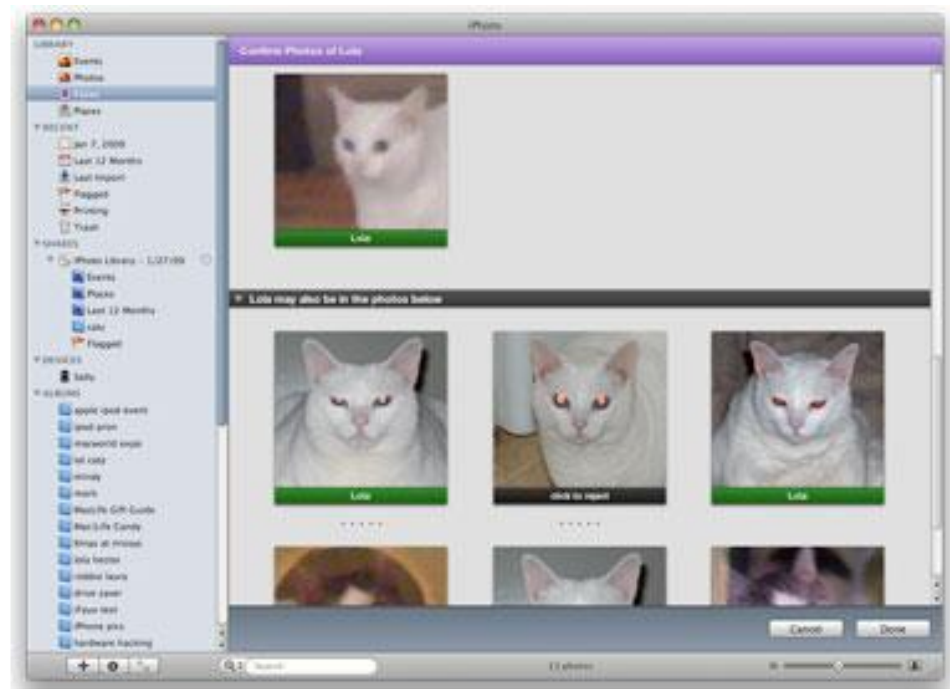
Face detection and recognition

Consumer application: Apple iPhoto

http://www.apple.com/ilife/iphoto/

Consumer application: Apple iPhoto

Can be trained to recognize pets!



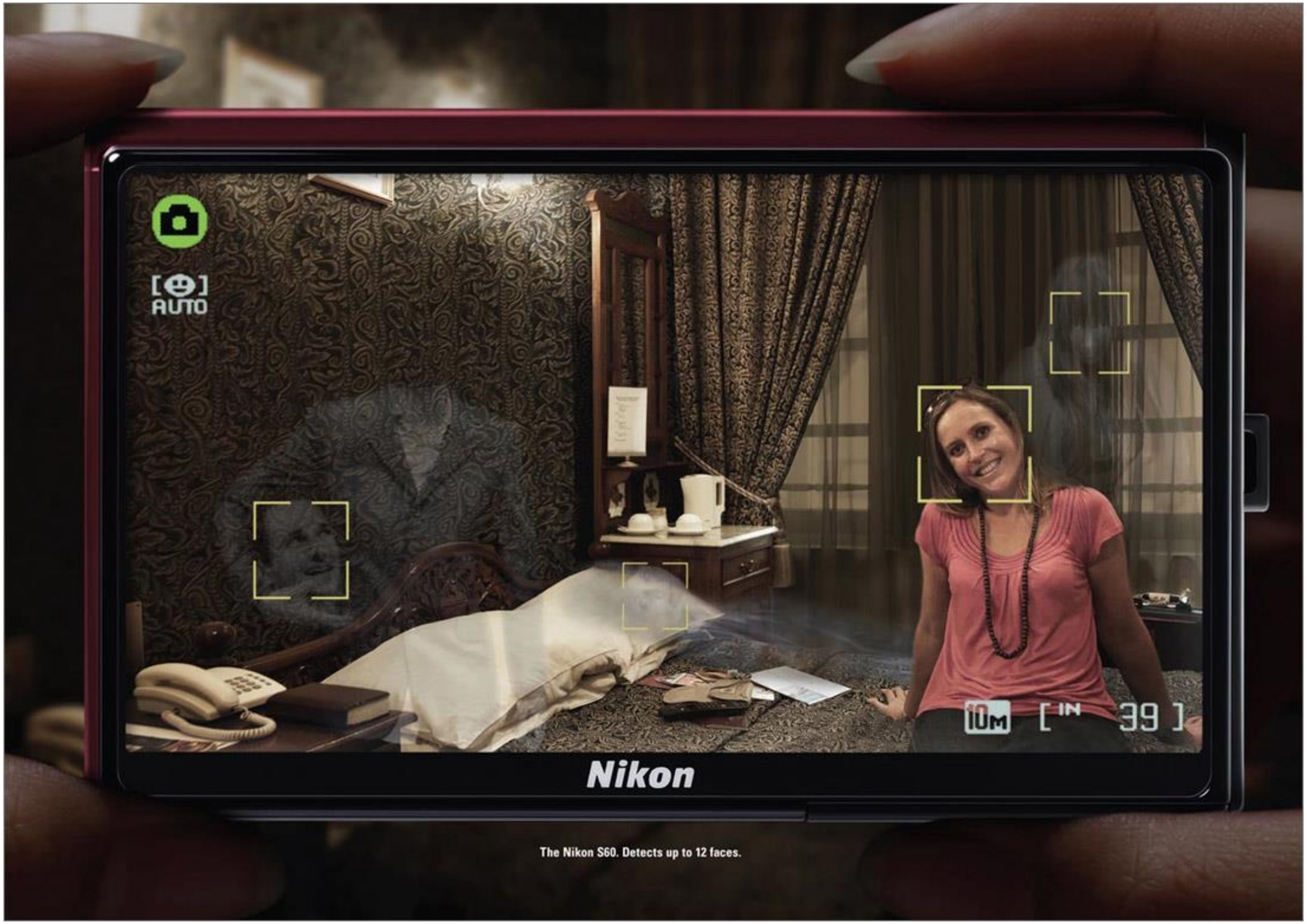
http://www.maclife.com/article/news/iphotos_faces_recognizes_cats

Consumer application: Apple iPhoto

[Things iPhoto thinks are faces]

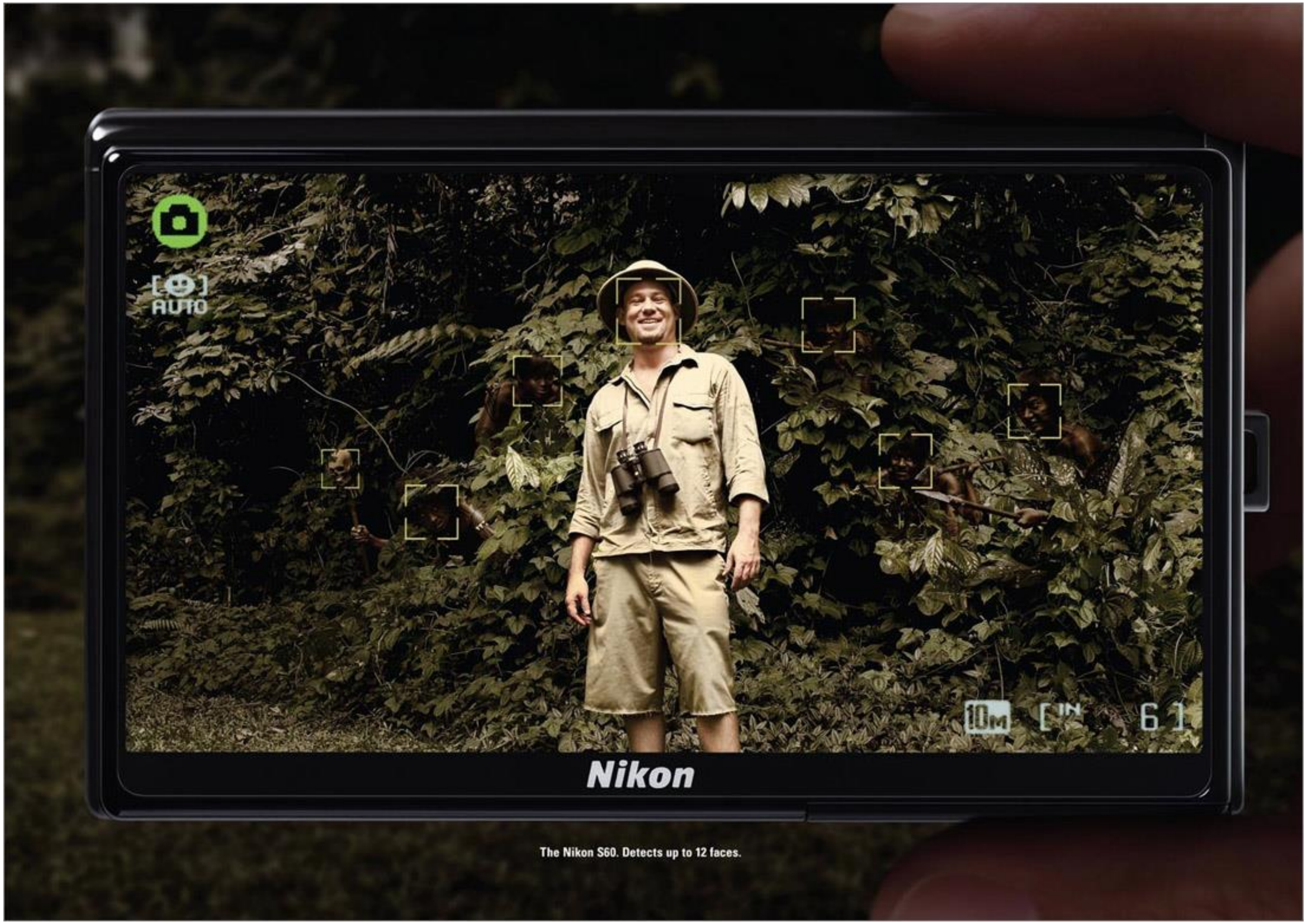
Funny Nikon ads

"The Nikon S60 detects up to 12 faces."



Funny Nikon ads

"The Nikon S60 detects up to 12 faces."



Challenges of face detection

- Sliding window detector must evaluate tens of thousands of location/scale combinations
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - A megapixel image has ~10^6 pixels and a comparable number of candidate face locations
  - To avoid having a false positive in every image image, our false positive rate has to be less than 10^{-6}

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

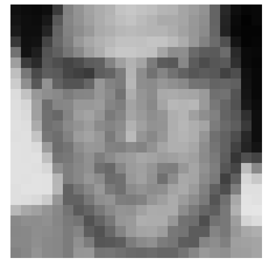
P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

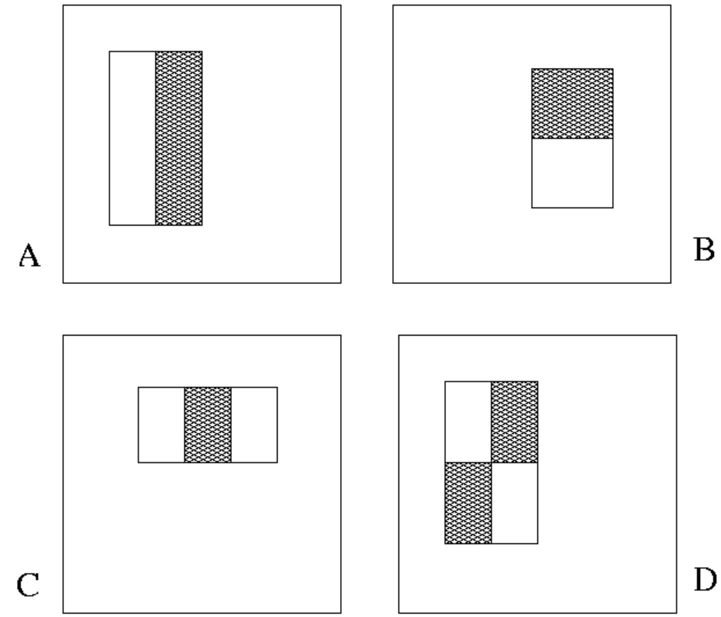
P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

~8000 citations!

Image Features

"Rectangle filters"

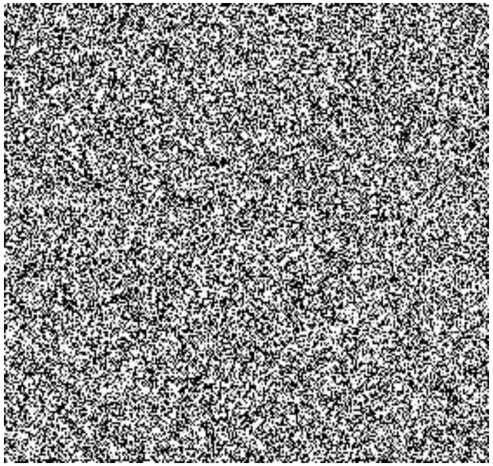




Value =

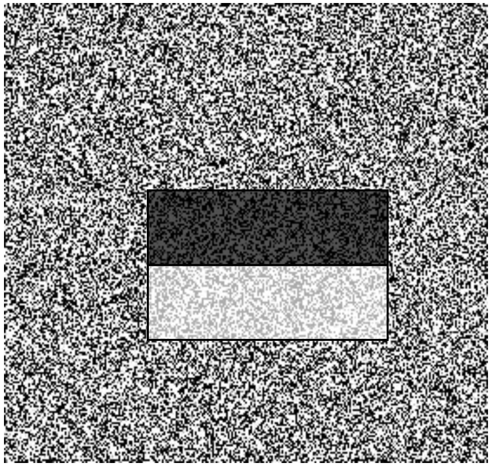
∑ *(pixels in white area)* –
∑ *(pixels in black area)*

Example



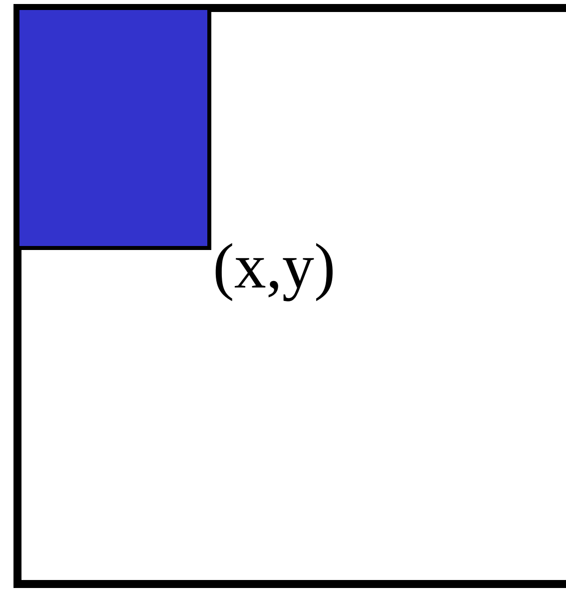
Source

Result

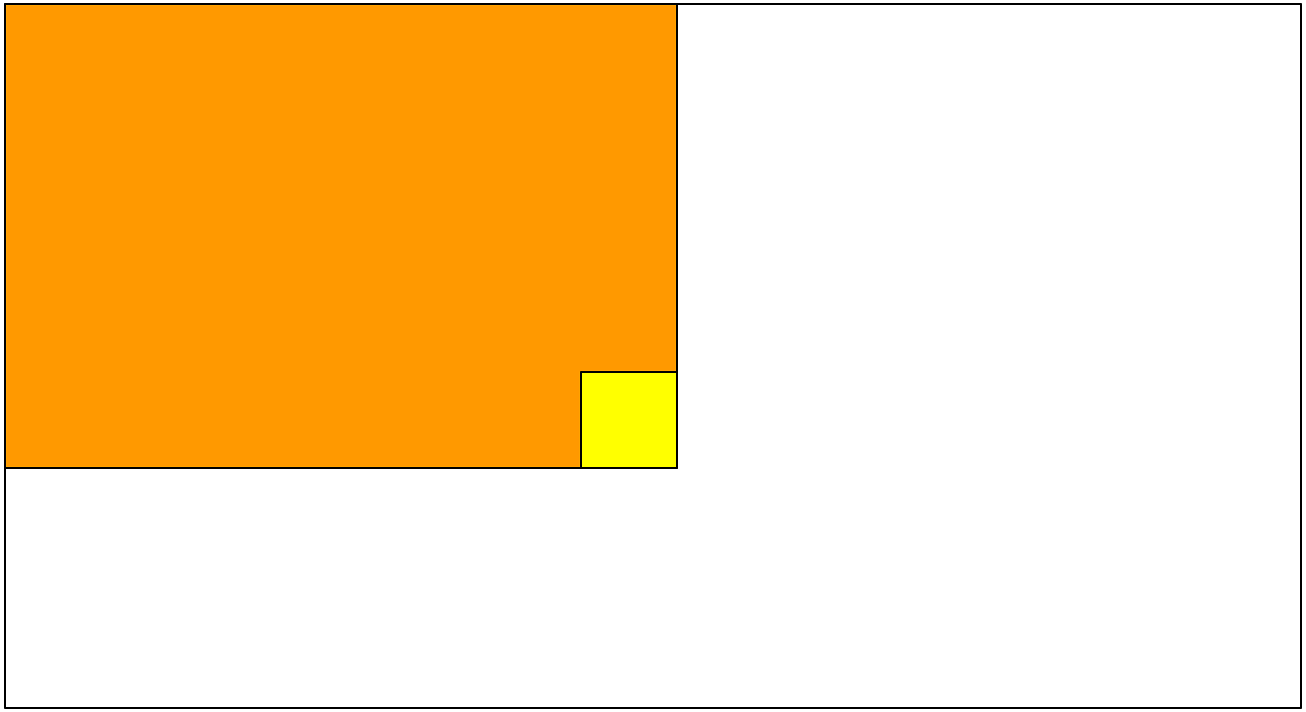


Fast computation with integral images

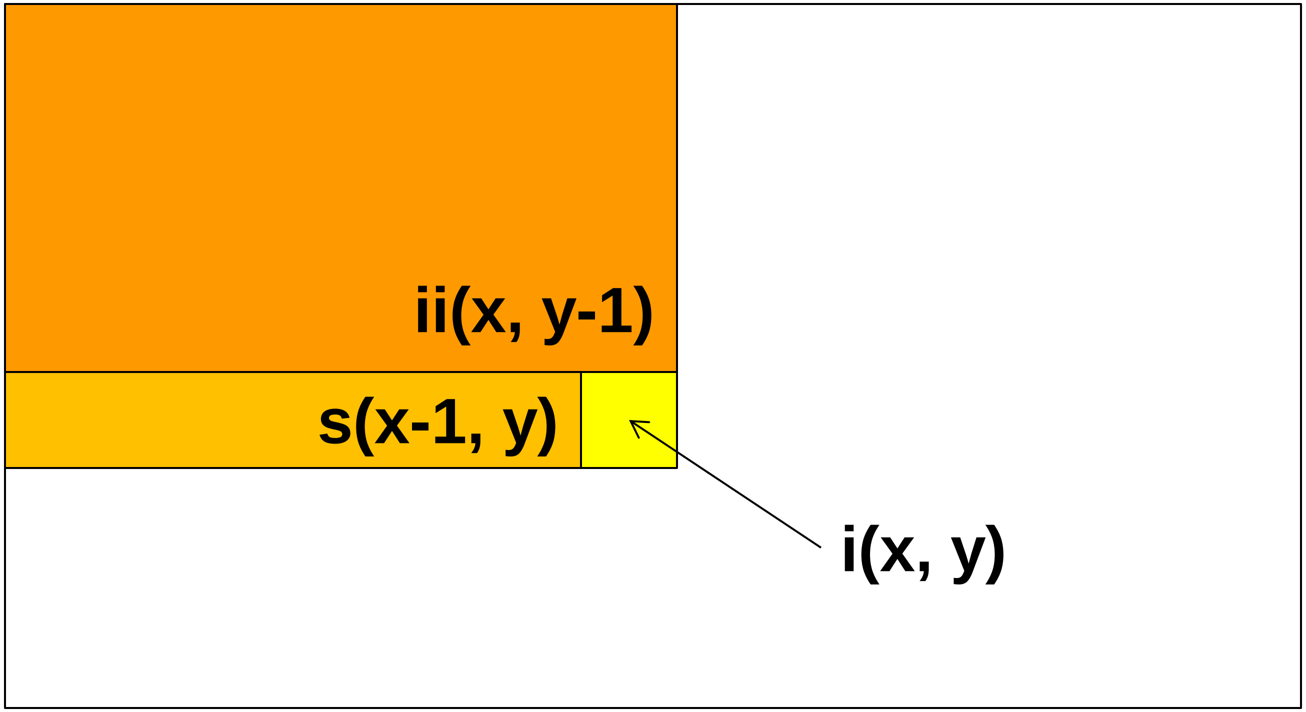
- The *integral image* computes a value at each pixel (x,y) that is the sum of the pixel values above and to the left of (x,y), inclusive
- This can quickly be computed in one pass through the image



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)

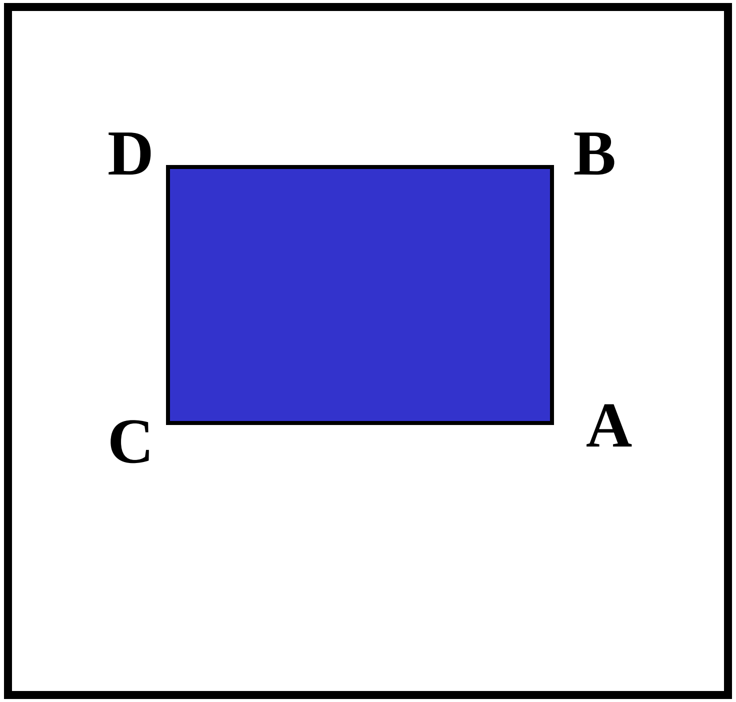
MATLAB: `ii = cumsum(cumsum(double(i)), 2);`

Computing sum within a rectangle

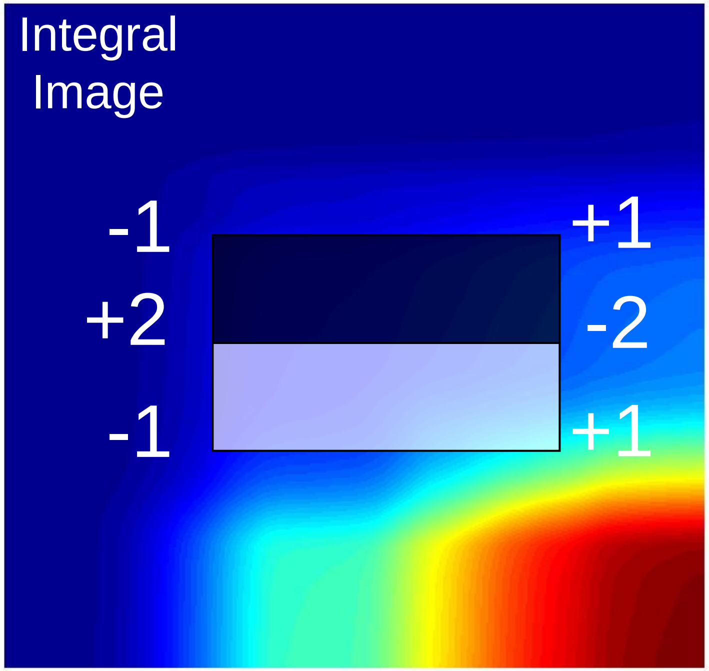
- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

  sum = A – B – C + D

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

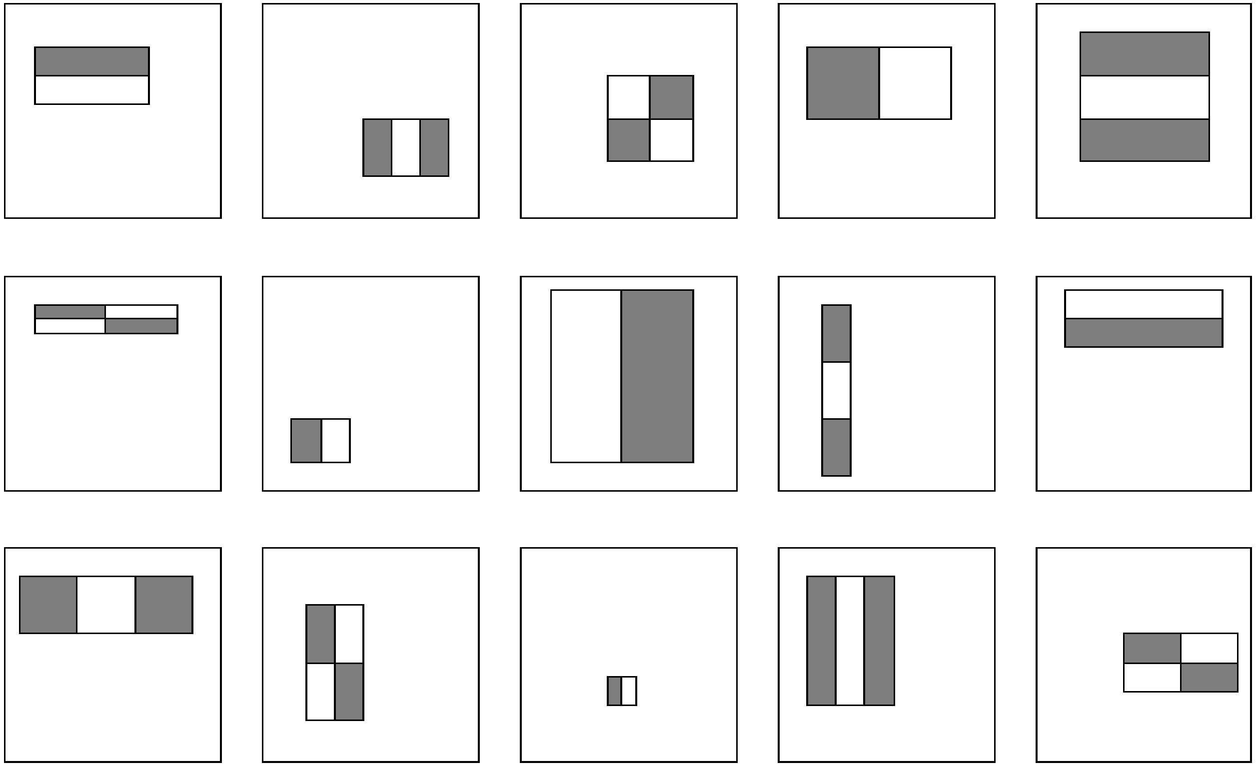


Computing a rectangle feature



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 combines *weak learners* into a more accurate *ensemble classifier*
- Weak learners based on rectangle filters:

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

- Ensemble classification function:

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

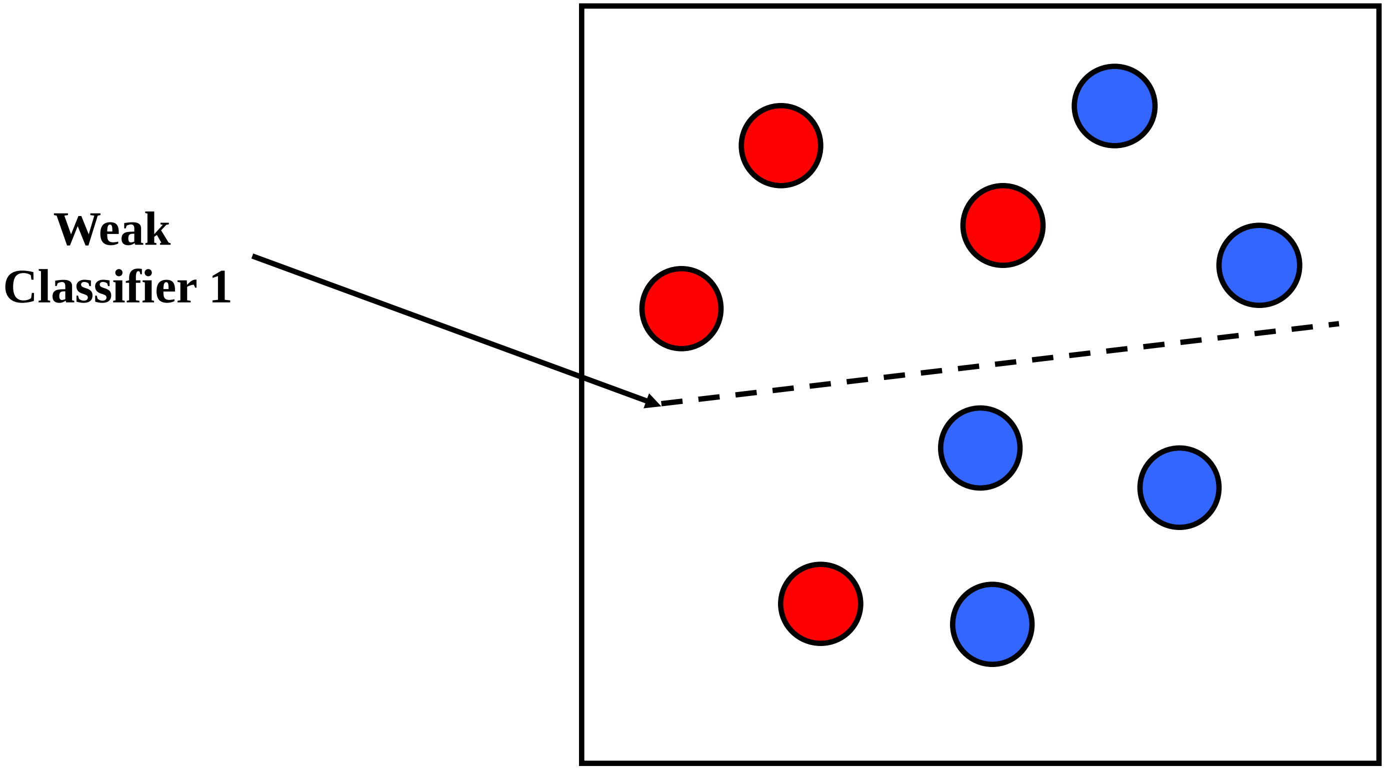
learned weights

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)

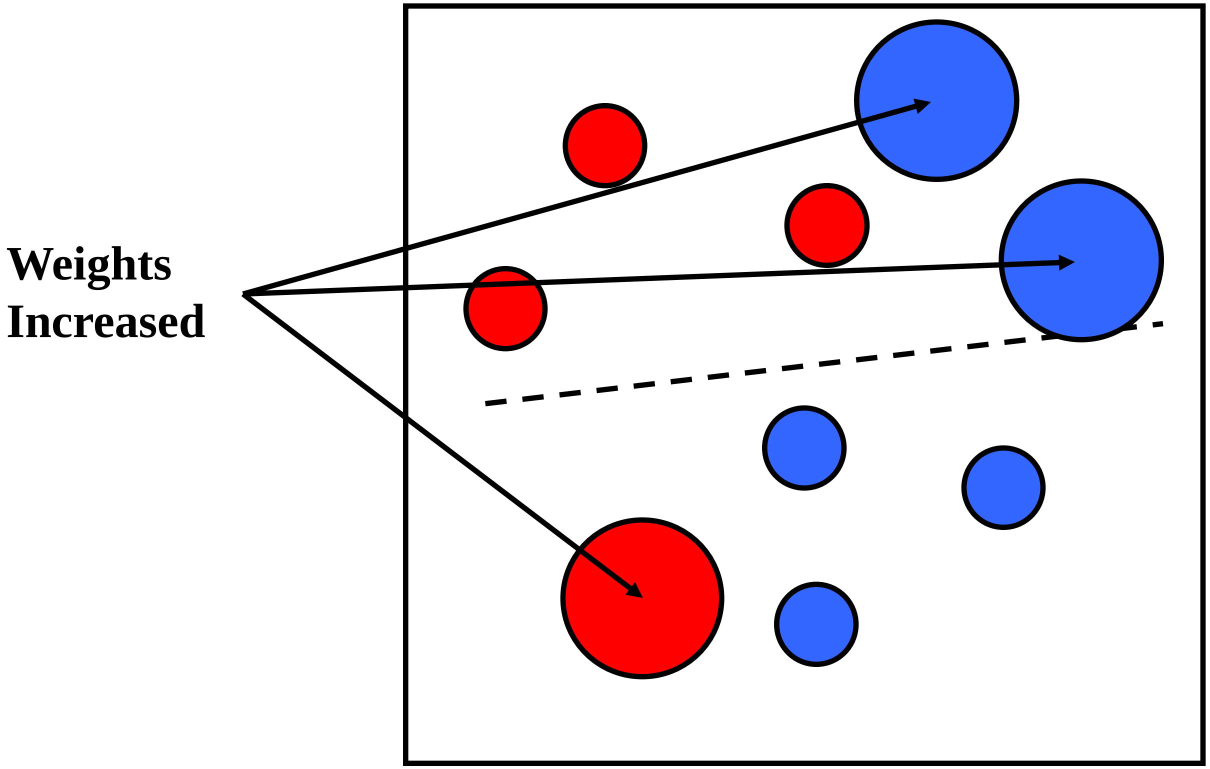
Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

Boosting intuition

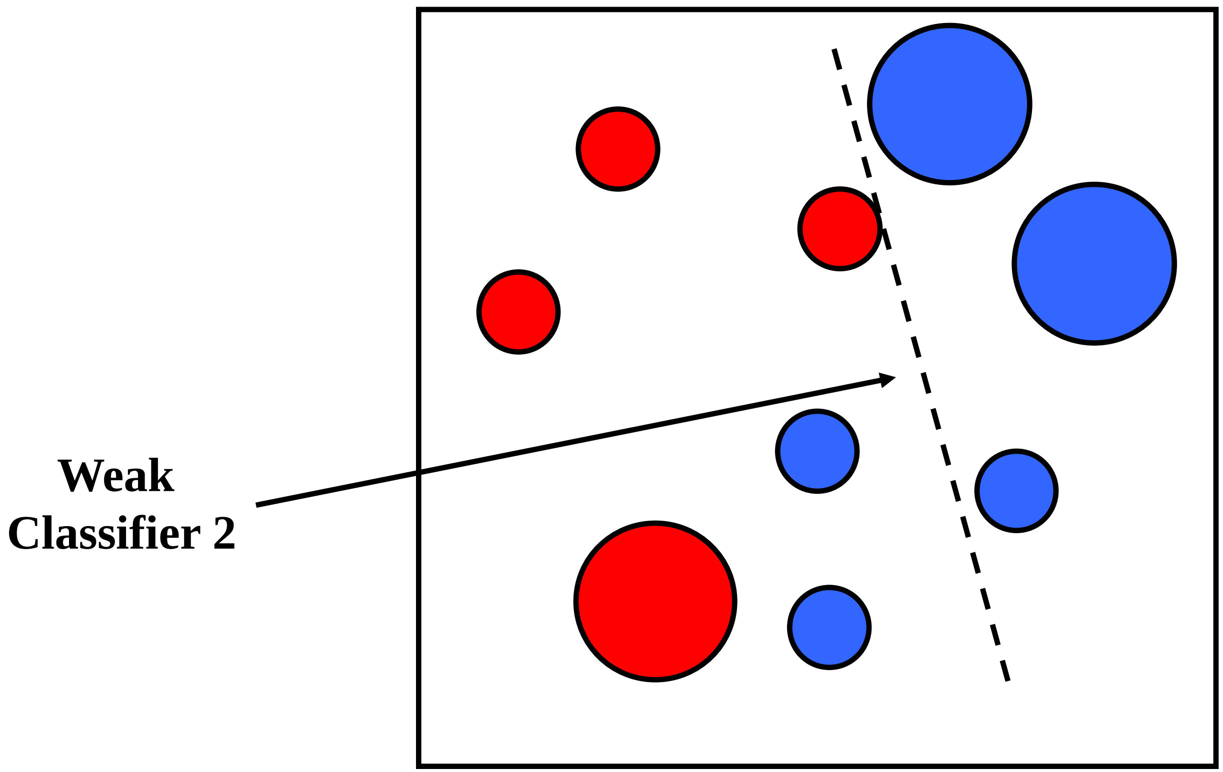


Slide credit: Paul Viola

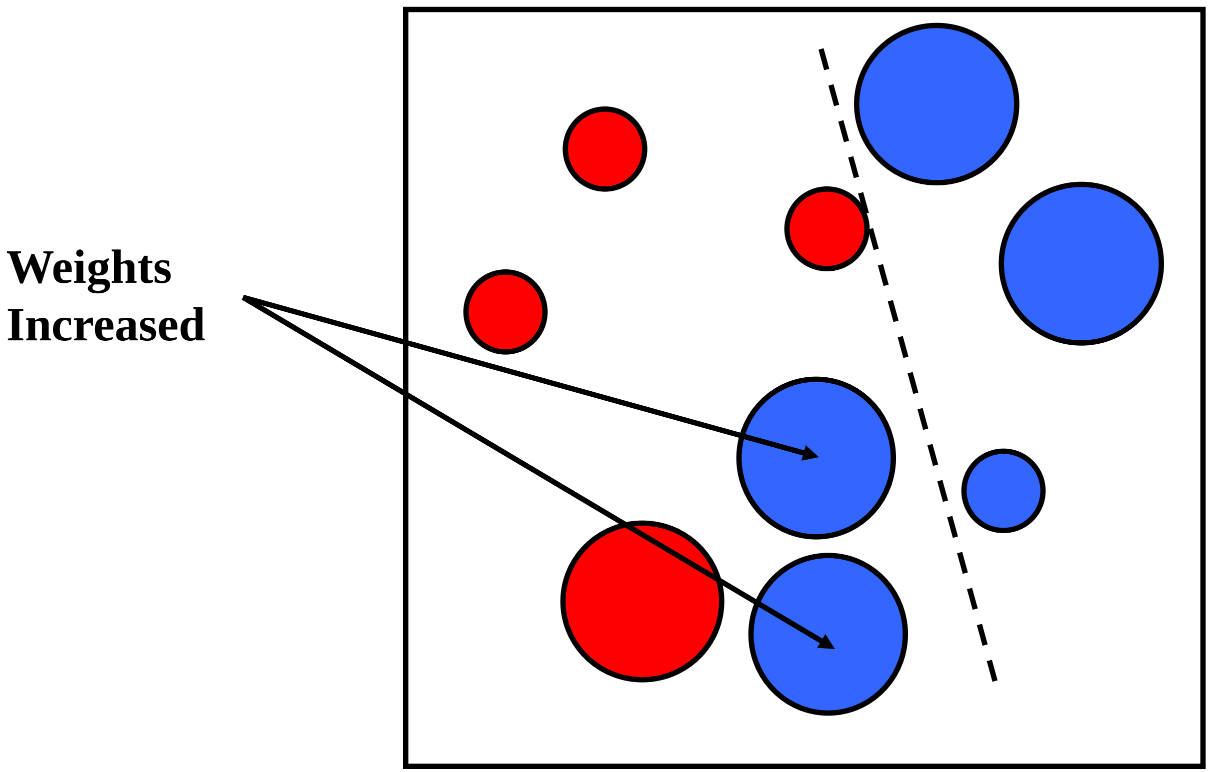
Boosting illustration



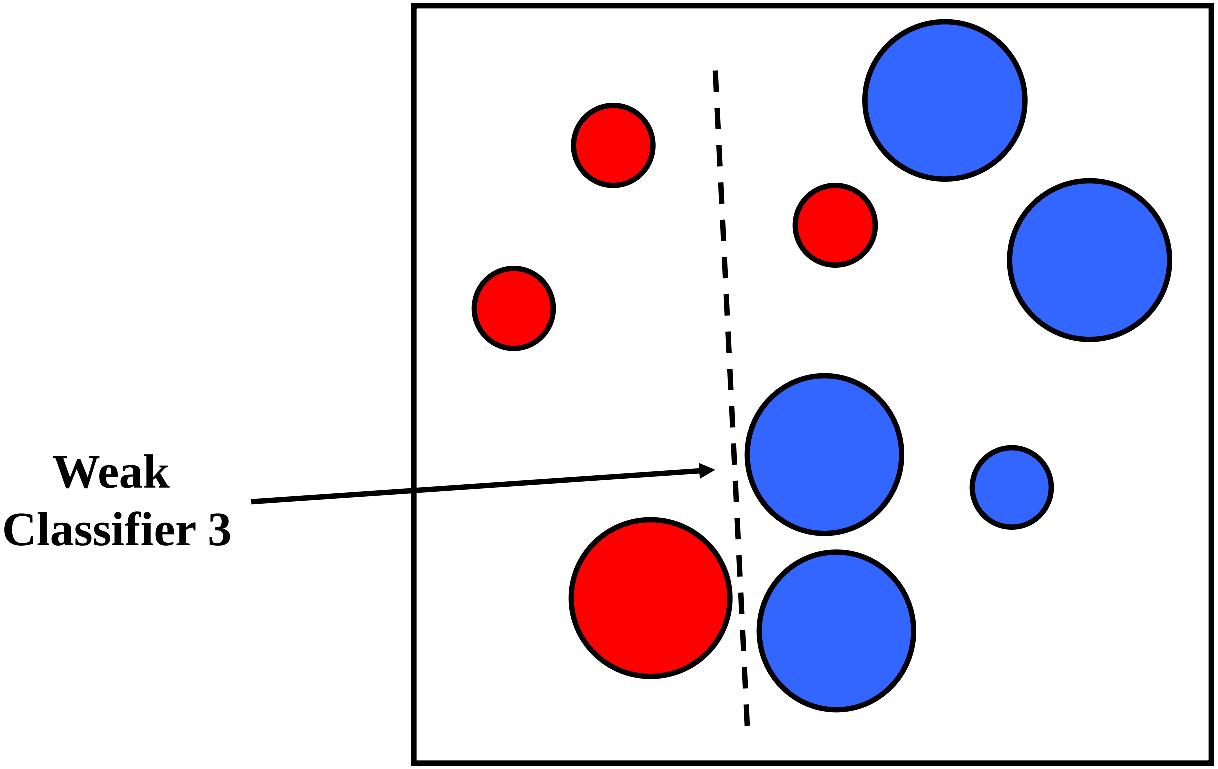
Boosting illustration



Boosting illustration

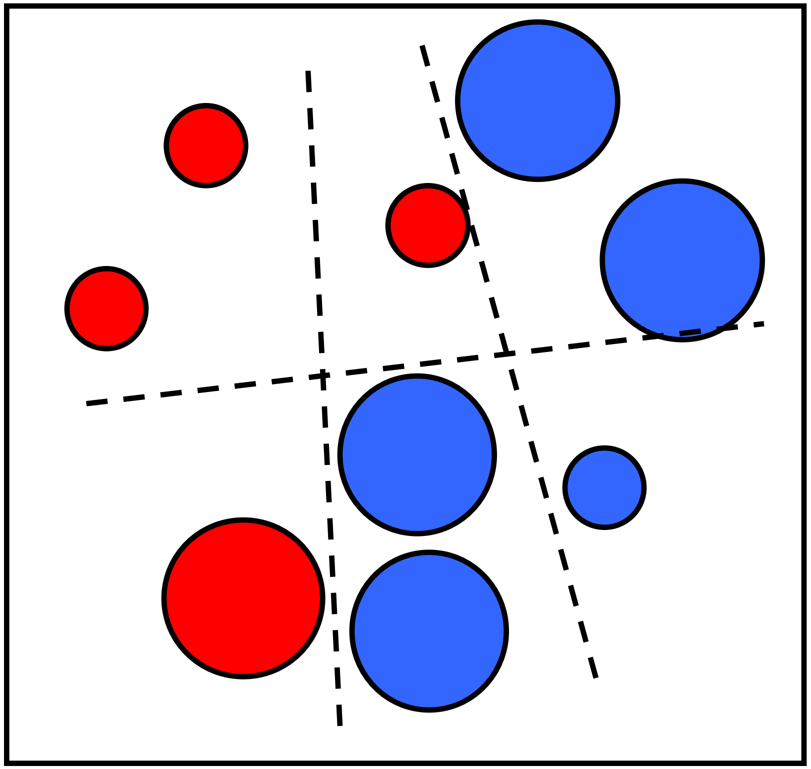


Boosting illustration



Boosting illustration

Final classifier is a combination of weak classifiers



Boosting for face detection

- First two features selected by boosting:

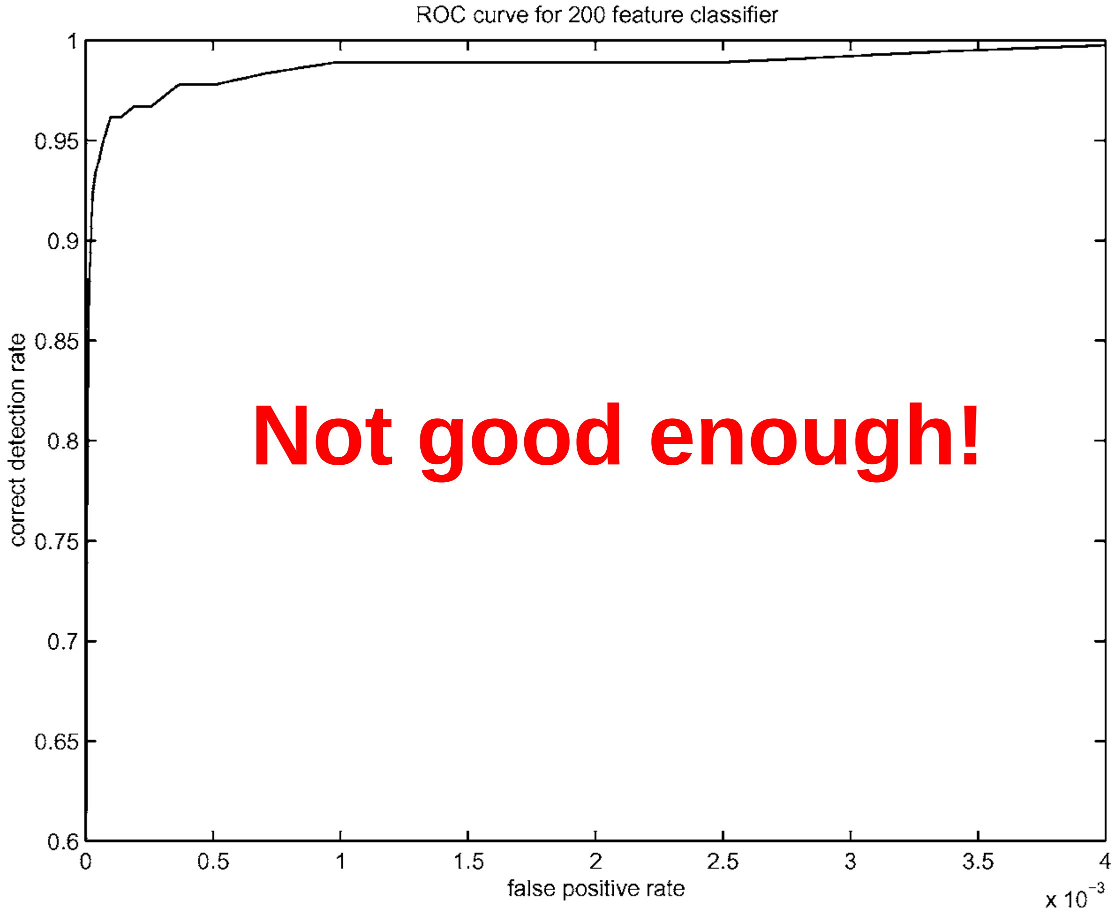
This feature combination can yield 100% detection rate and 50% false positive rate

Boosting vs. SVM

- Advantages of boosting
  - Integrates classifier training with feature selection
  - Complexity of training is linear instead of quadratic in the number of training examples
  - Flexibility in the choice of weak learners, boosting scheme
  - Testing is fast
  - Easy to implement
- 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

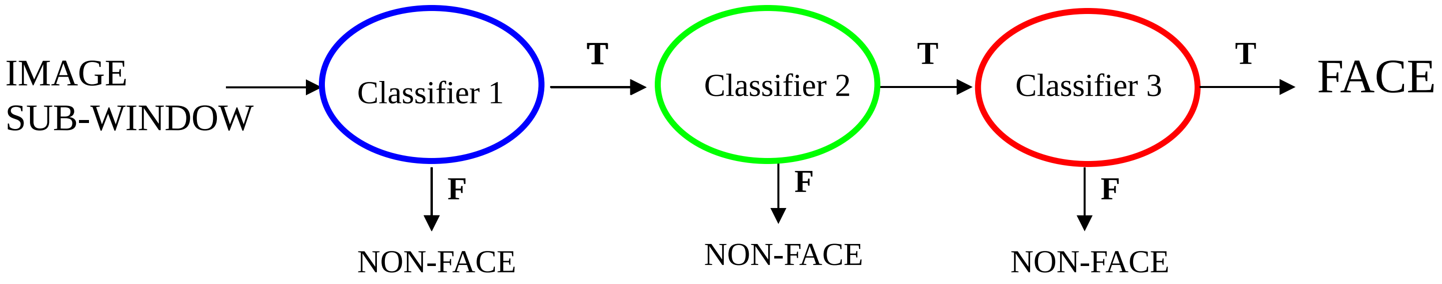
- A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084



Receiver operating characteristic (ROC) curve

Attentional cascade

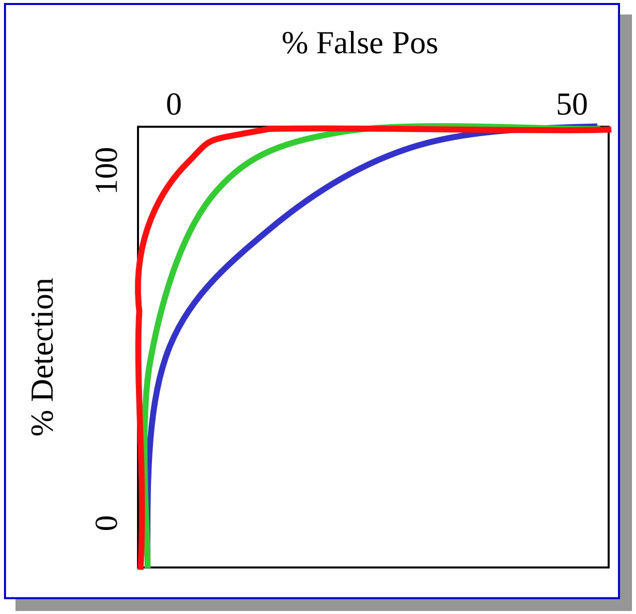
- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate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

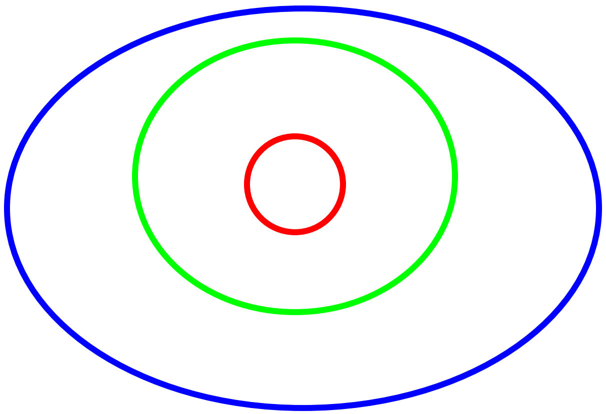


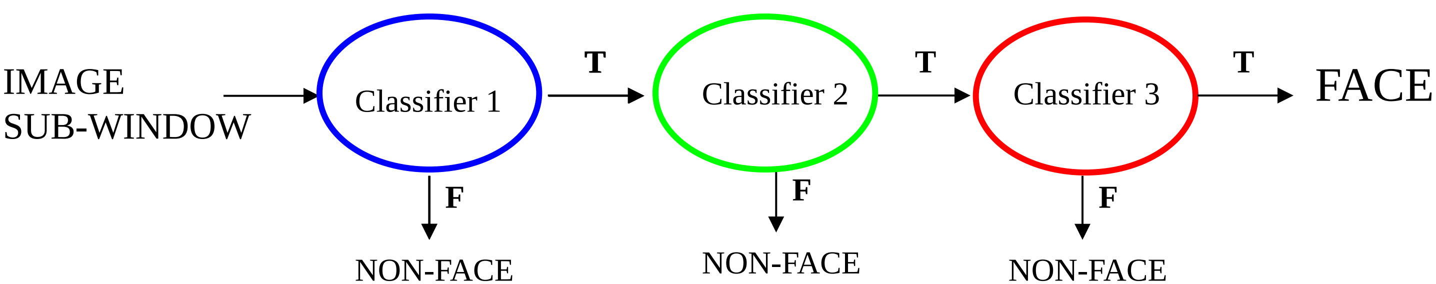
Attentional cascade

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

Receiver operating characteristic

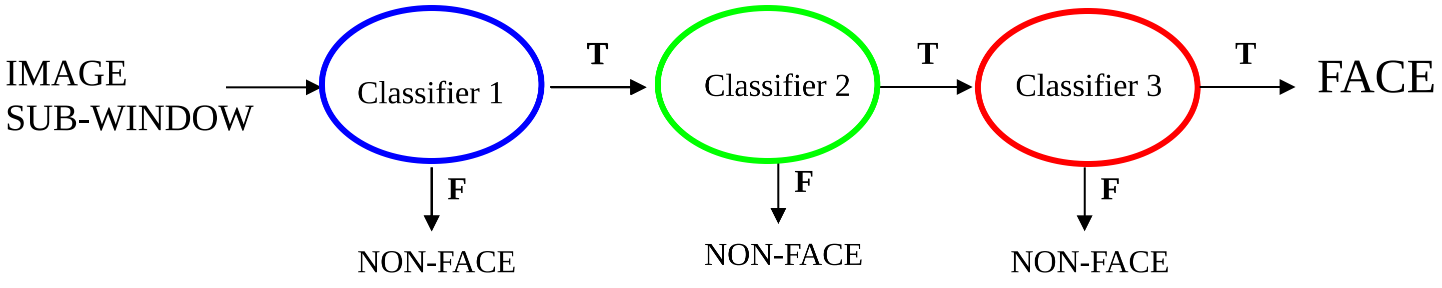






Attentional cascade

- 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

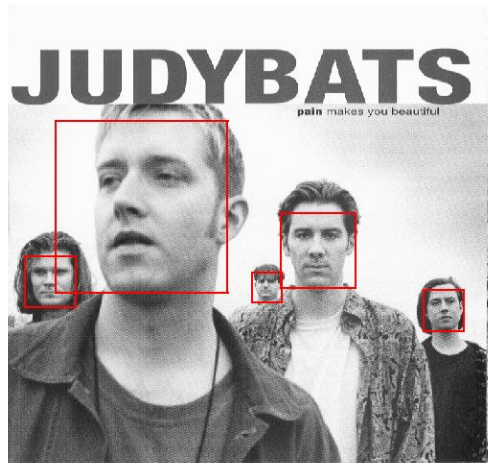
The implemented system

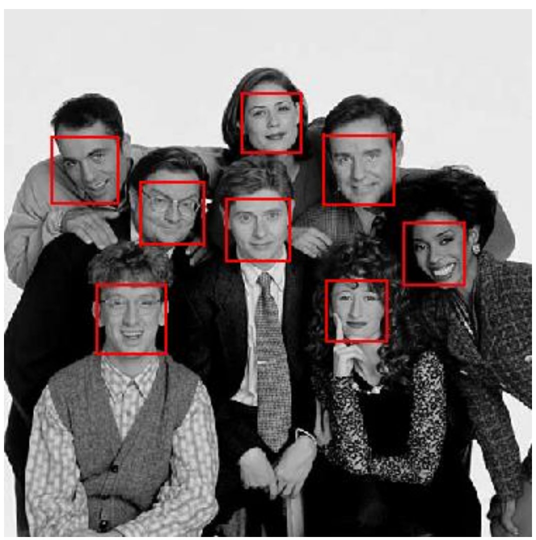
- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

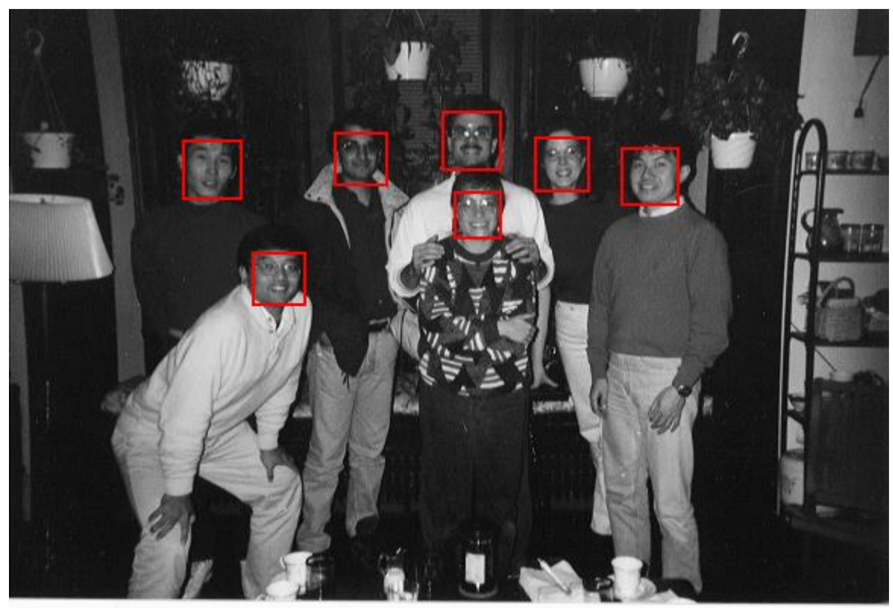
System performance

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

Output of Face Detector on Test Images

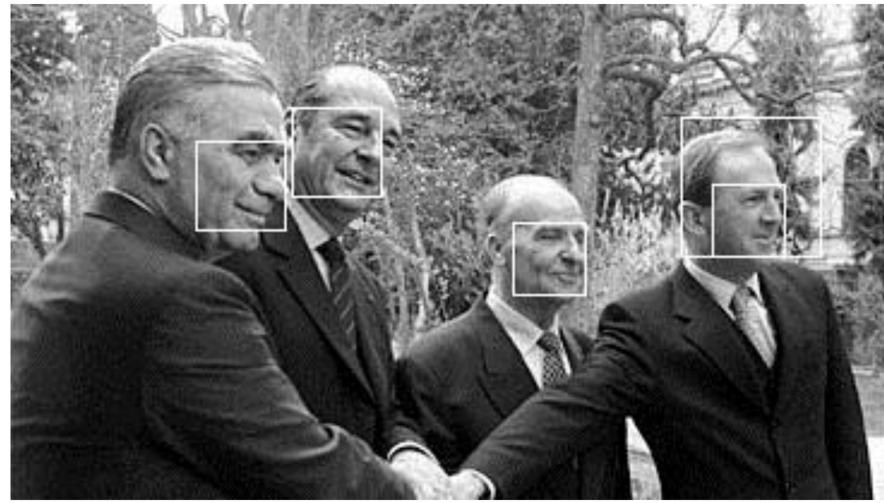






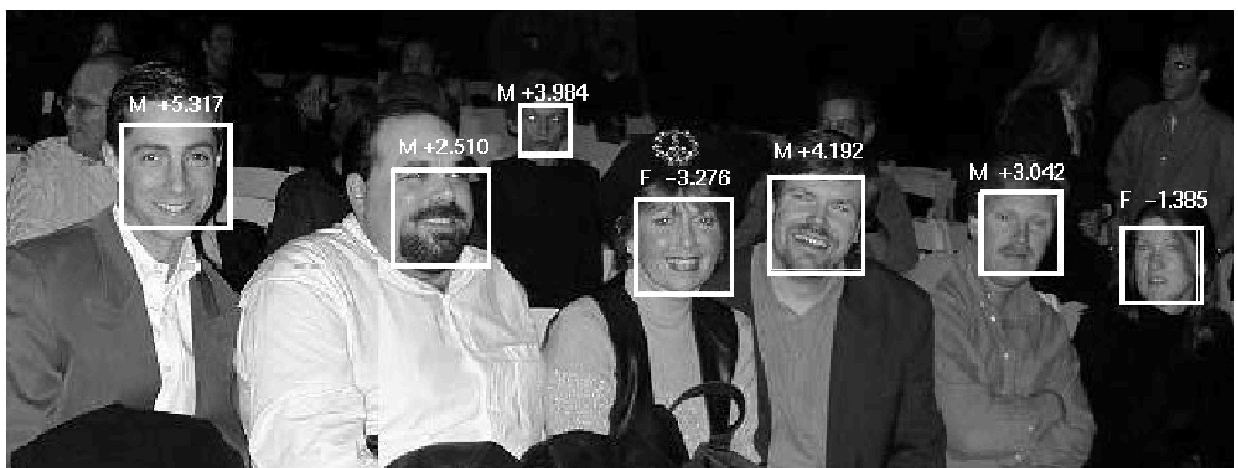
Other detection tasks

Facial Feature Localization

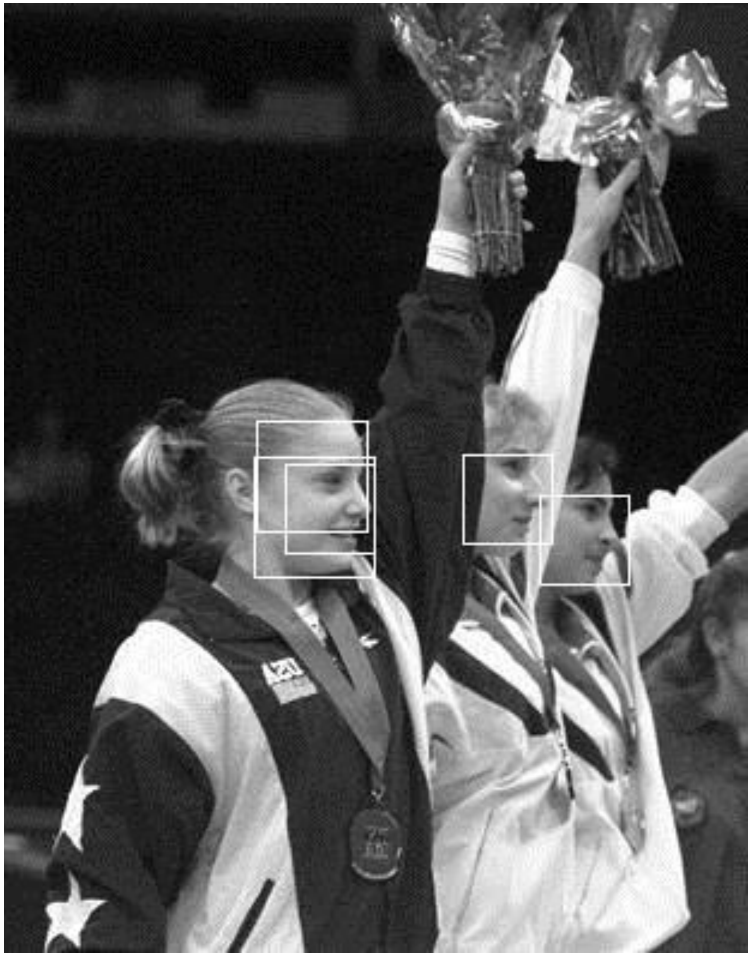


Profile Detection

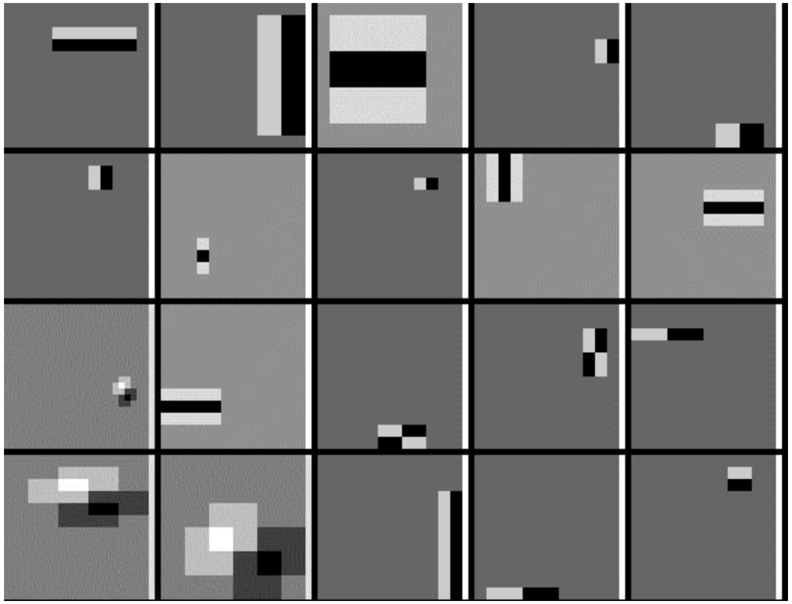
Male vs. female



Profile Detection



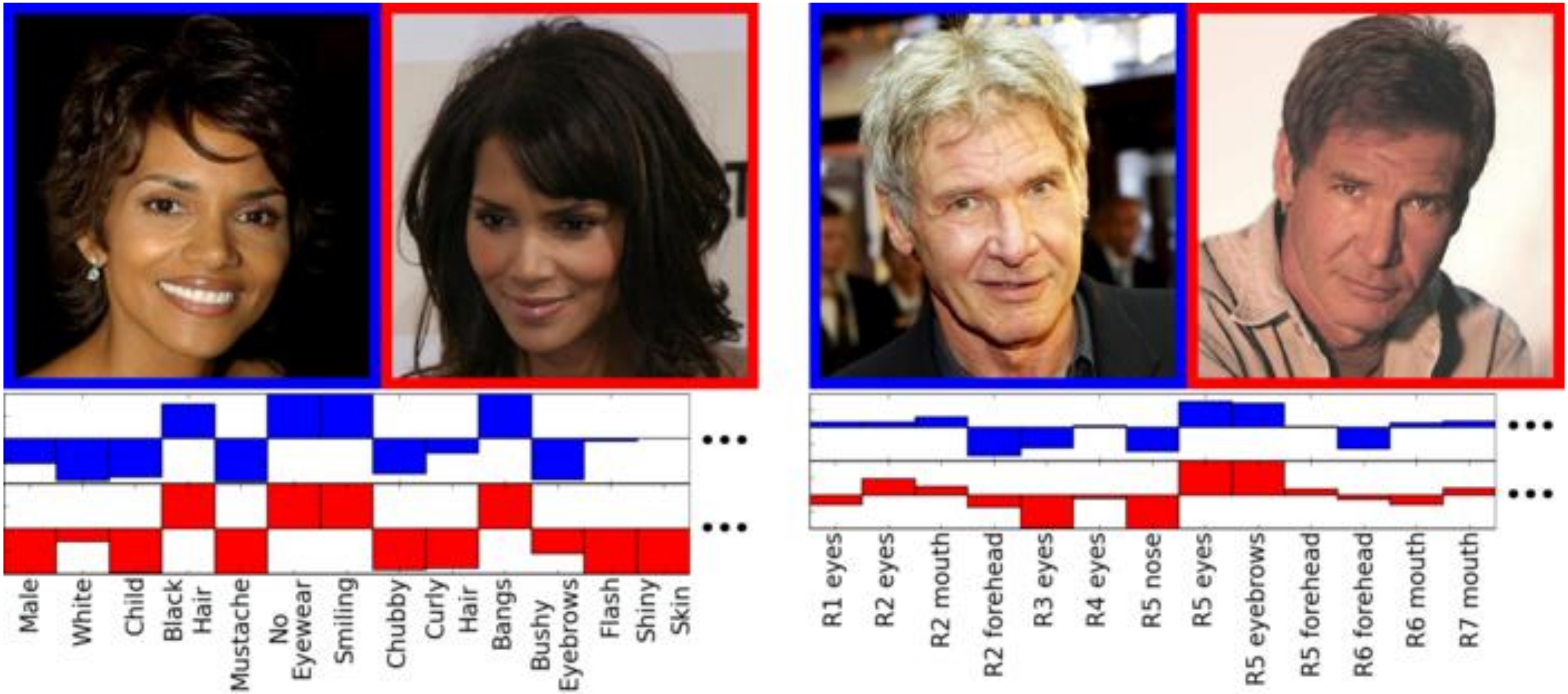
Profile Features



Summary: Viola/Jones detector

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

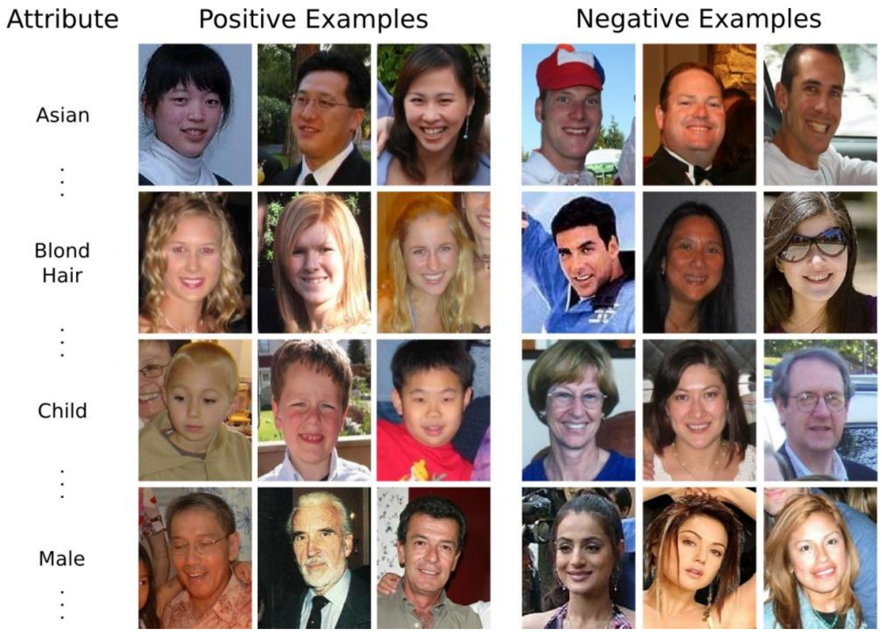
Face Recognition



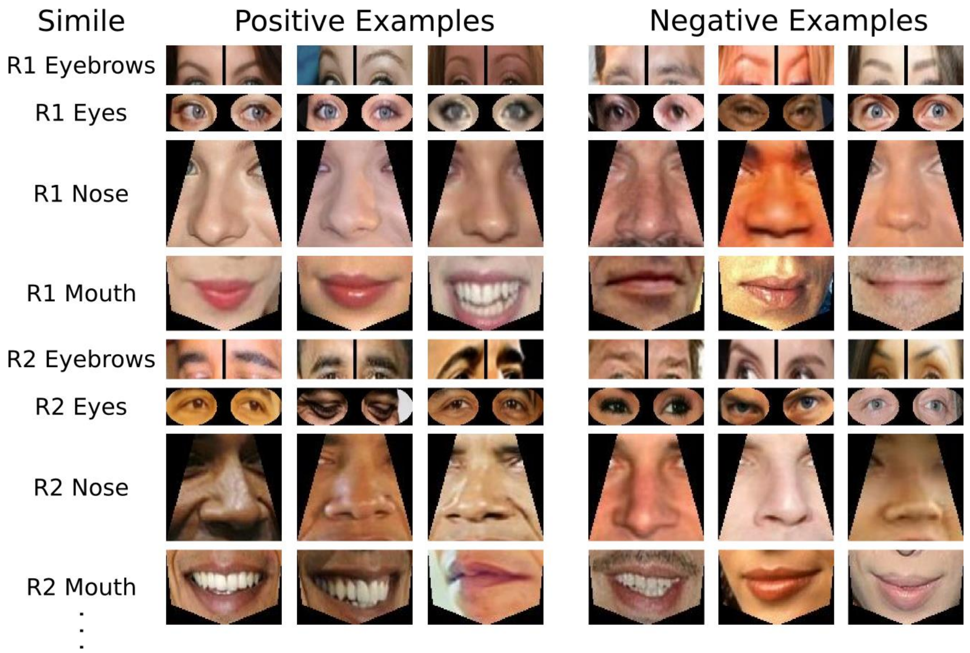
N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, **"Attribute and Simile Classifiers for Face Verification,"** ICCV 2009.

Face Recognition

Attributes for training



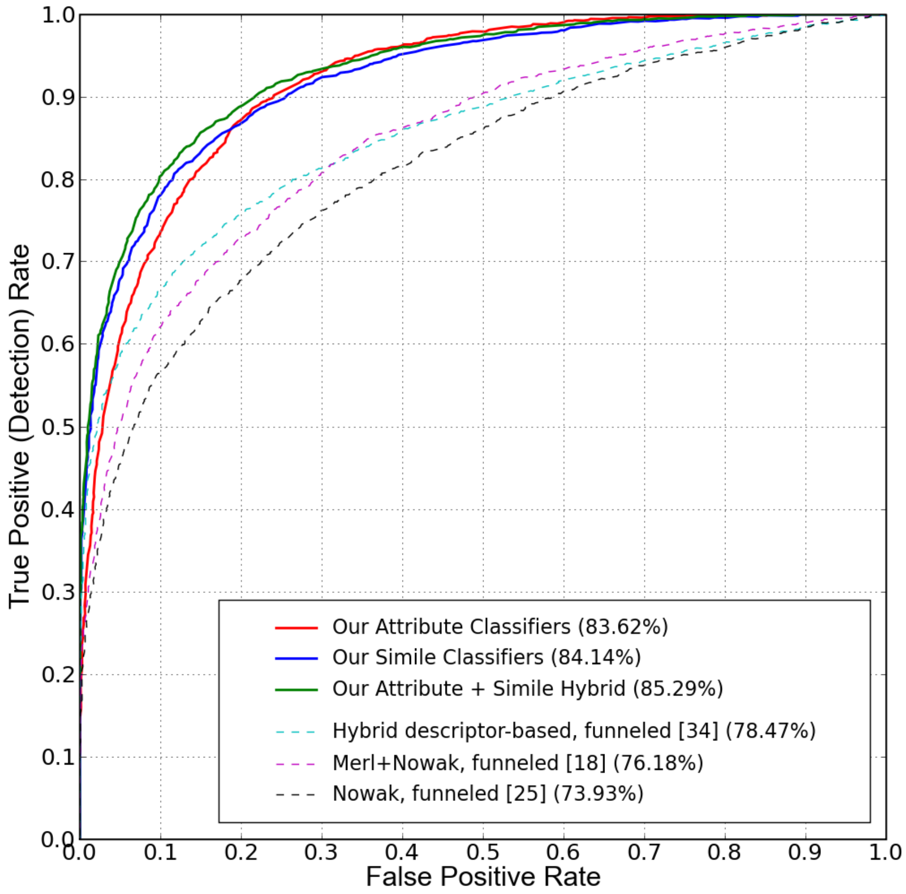
Similes for training



N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, **["Attribute and Simile Classifiers for Face Verification,"]()** ICCV 2009.

Face Recognition

Results on [Labeled Faces in the Wild](#) Dataset



N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, **"Attribute and Simile Classifiers for Face Verification,"** ICCV 2009.