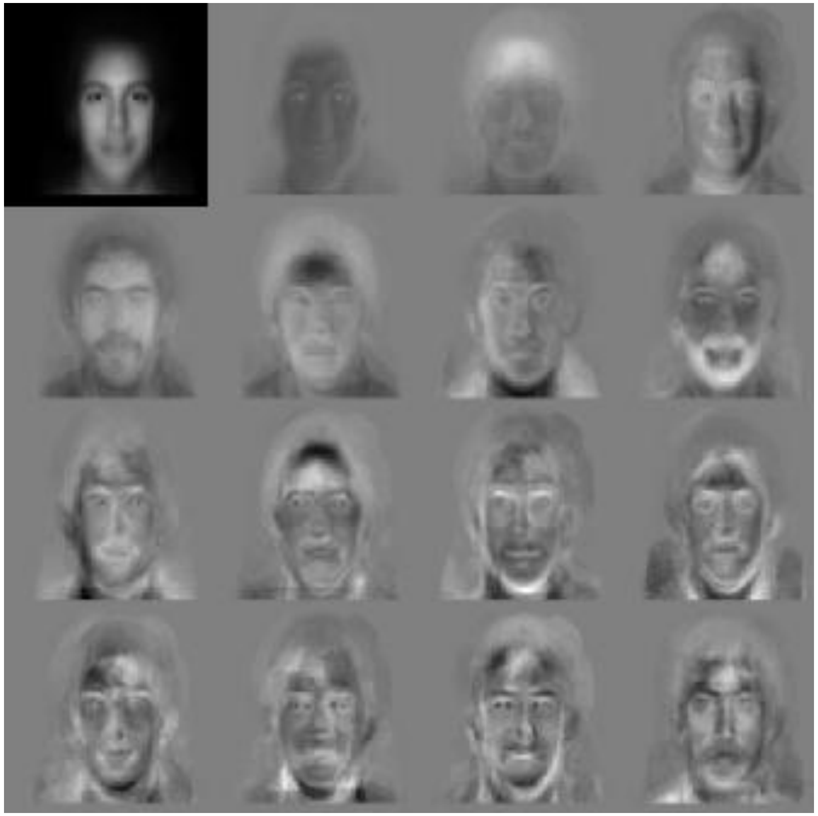
CS6670: Computer Vision

Noah Snavely

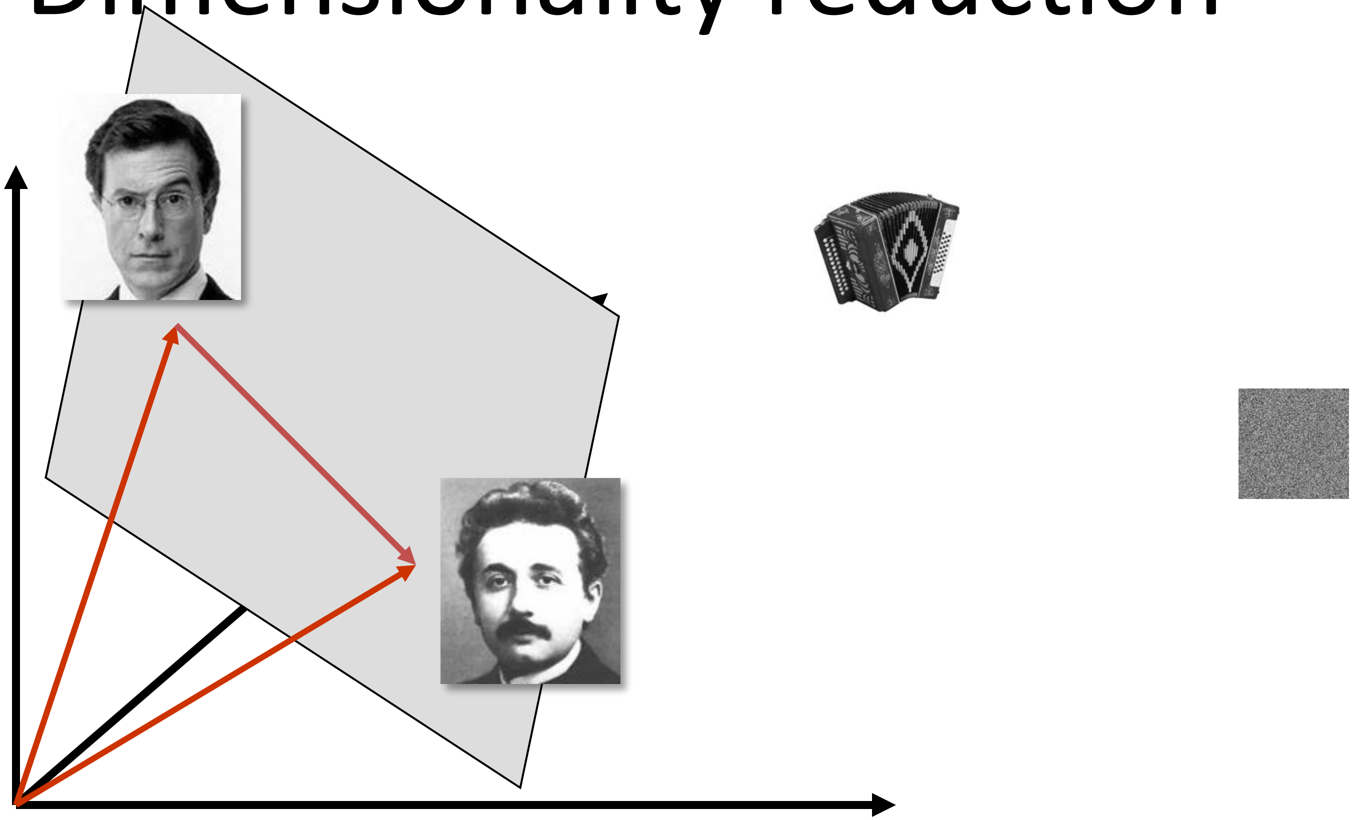
Lecture 15: Eigenfaces



Announcements

- Wednesday’s class is cancelled
- My office hours moved to tomorrow (Tuesday) 1:30-3:00

Dimensionality reduction

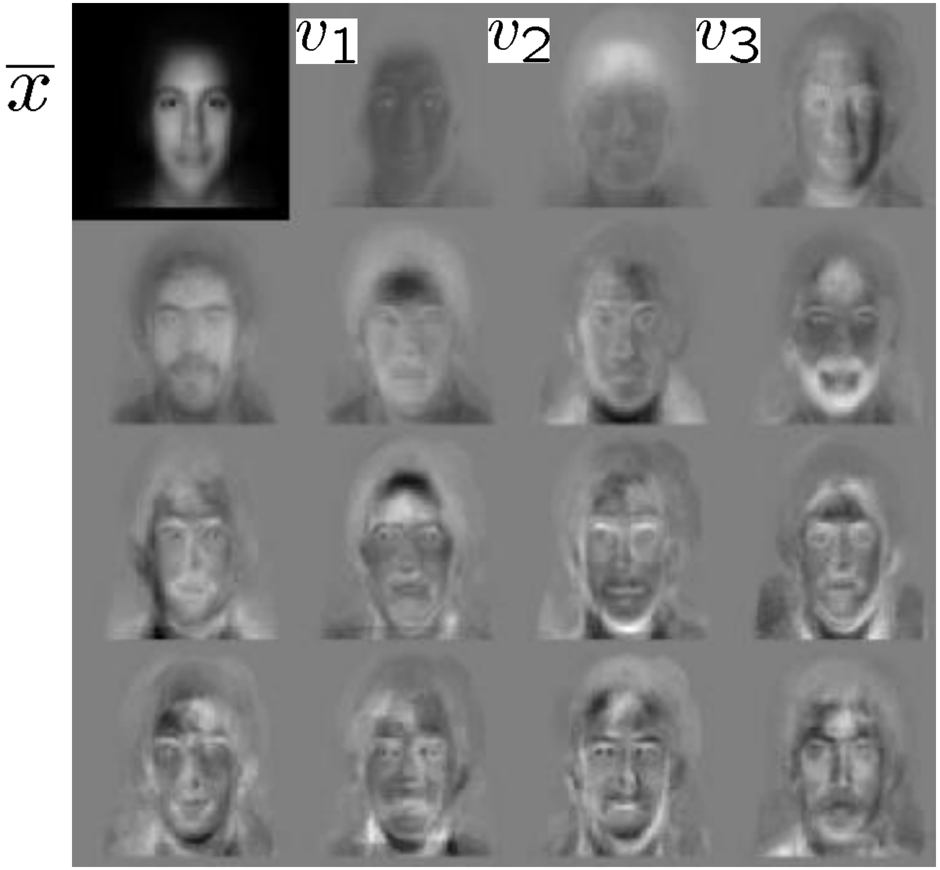


- The set of faces is a "subspace" of the set of images
  - Suppose it is K dimensional
  - We can find the best subspace using PCA
  - This is like fitting a "hyper-plane" to the set of faces
    - spanned by vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_K}$
    - any face $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

Eigenfaces

PCA extracts the eigenvectors of **A**

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



Projecting onto the eigenfaces

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

- A face is converted to eigenface coordinates by

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

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

$\mathbf{x}$

$a_1\mathbf{v_1}$ $a_2\mathbf{v_2}$ $a_3\mathbf{v_3}$ $a_4\mathbf{v_4}$ $a_5\mathbf{v_5}$ $a_6\mathbf{v_6}$ $a_7\mathbf{v_7}$ $a_8\mathbf{v_8}$

Detection and 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

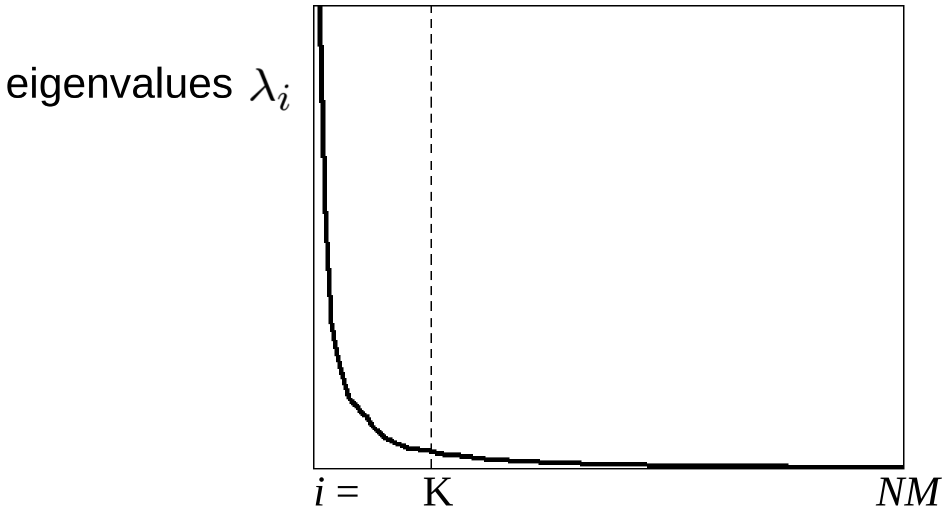
$$\mathbf{x} \rightarrow (a_1, a_2, \ldots, a_K)$$

3. Detect if x is a face

$$\|\mathbf{x} - (\overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K})\| < \text{threshold}$$

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

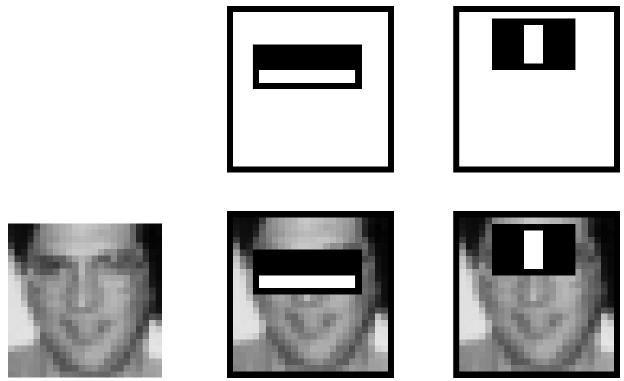
- the eigenvalue tells you the amount of variance "in the direction" of that eigenface
- ignore eigenfaces with low variance

Issues: metrics

What's the best way to compare images?

- need to define appropriate features
- depends on goal of recognition task

exact matching
complex features work well
(SIFT, MOPS, etc.)



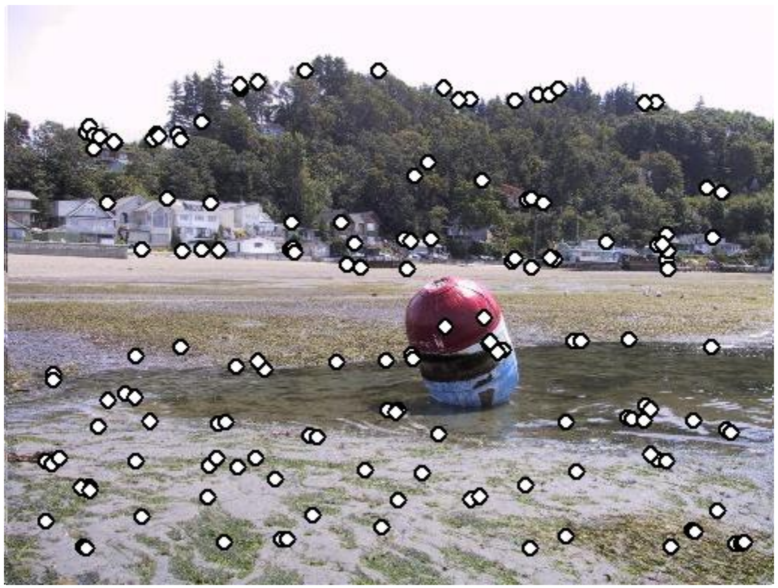
classification/detection
simple features work well
(Viola/Jones, etc.)

Metrics

Lots more feature types that we haven’t mentioned

- moments, statistics
  - metrics: Earth mover’s distance, ...
- edges, curves
  - metrics: Hausdorff, shape context, ...
- 3D: surfaces, spin images
  - metrics: chamfer (ICP)
- ...

Issues: feature selection



If all you have is one image:
non-maximum suppression, etc.

If you have a training set of images:
AdaBoost, etc.

Issues: data modeling

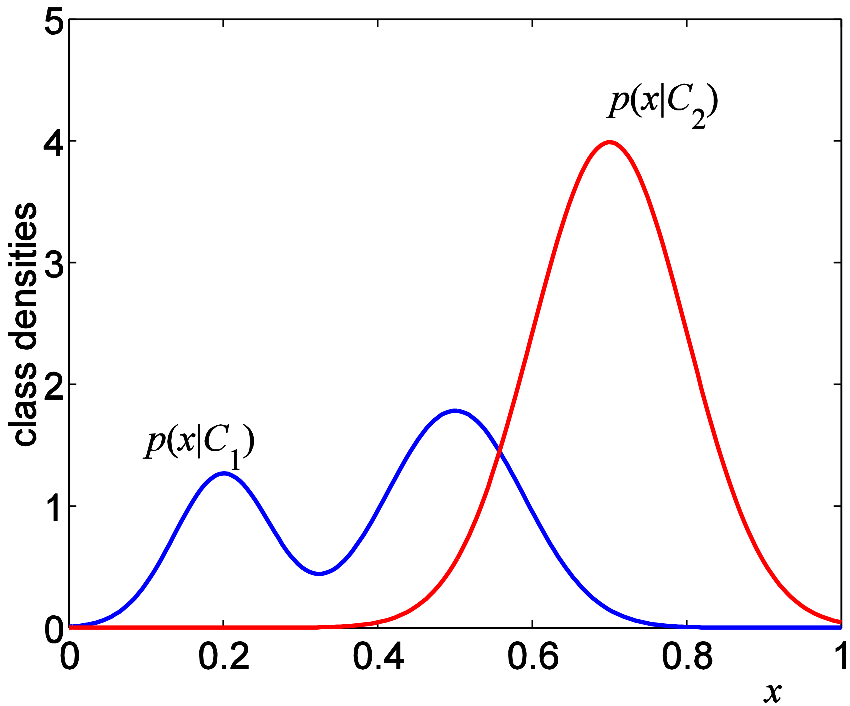
Generative methods

- model the “shape” of each class
  - histograms, PCA, mixtures of Gaussians
  - graphical models (HMM’s, belief networks, etc.)
  - ...

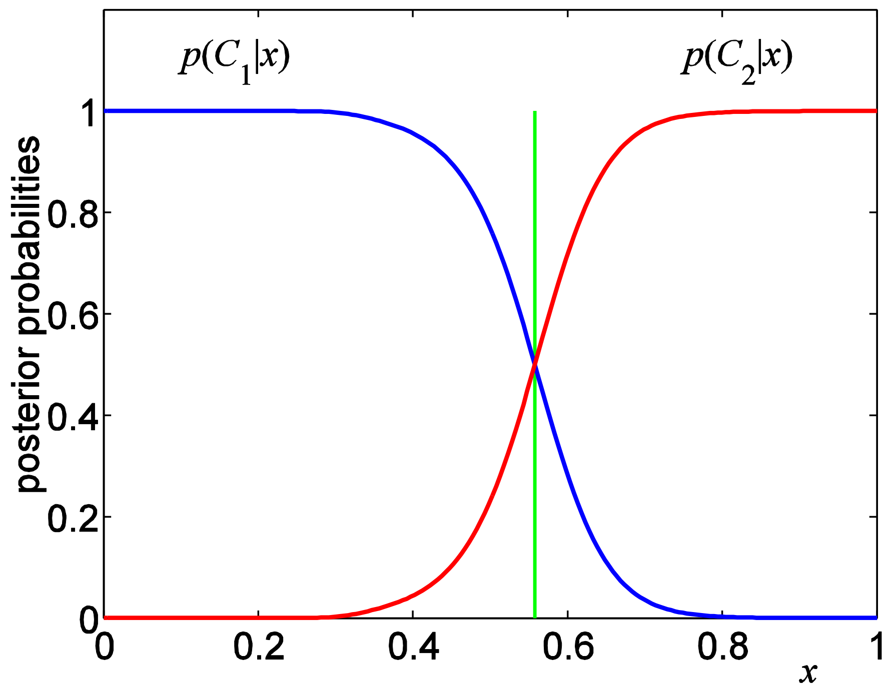
Discriminative methods

- model boundaries between classes
  - perceptrons, neural networks
  - support vector machines (SVM’s)

Generative vs. Discriminative



Generative Approach
model individual classes, priors



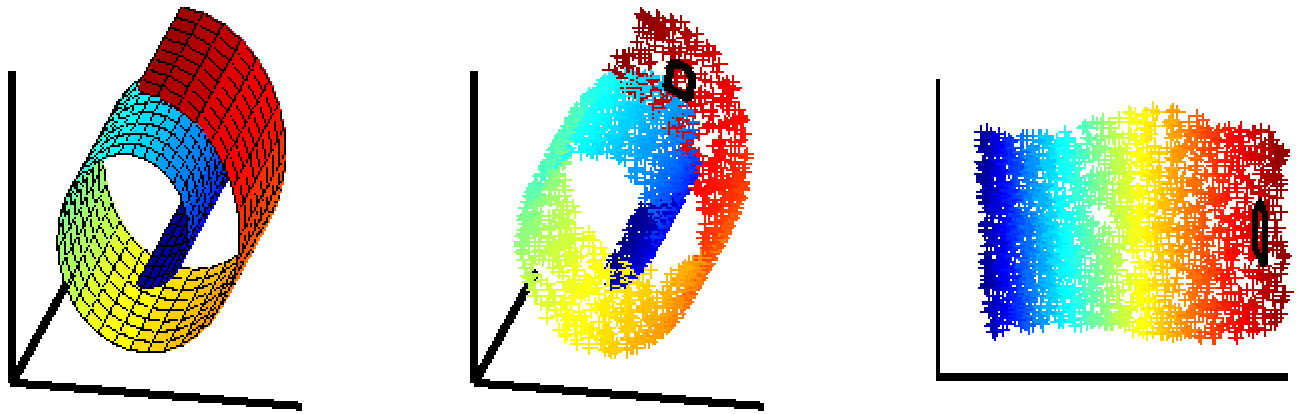
Discriminative Approach
model posterior directly

from Chris Bishop

Issues: dimensionality

What if your space isn't *flat*?

- PCA may not help



Nonlinear methods
LLE, MDS, etc.

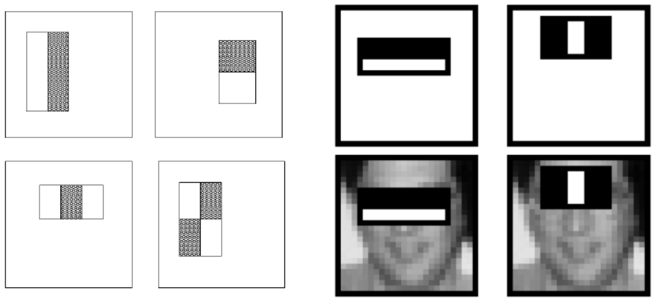
Issues: speed

- Case study: Viola Jones face detector
- Exploits two key strategies:
  - simple, super-efficient features
  - pruning (cascaded classifiers)

- Next few slides adapted Grauman & Liebe's tutorial
  - http://www.vision.ee.ethz.ch/~bleibe/teaching/tutorial-aaai08/
- Also see Paul Viola's talk (video)
  - http://www.cs.washington.edu/education/courses/577/04sp/contents.html#DM

Feature extraction

"Rectangular" filters

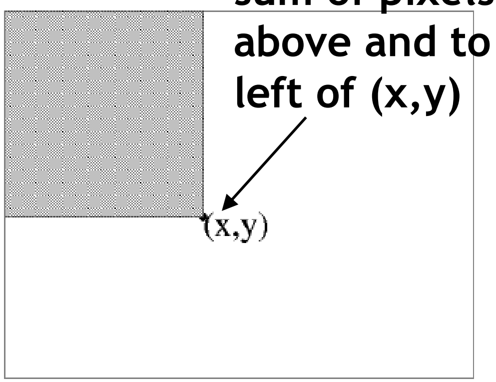


Feature output is difference between adjacent regions

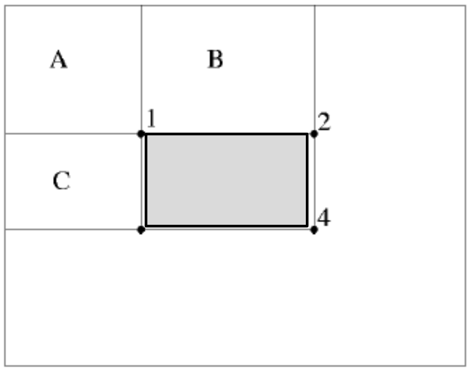
Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images → scale features directly for same cost

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



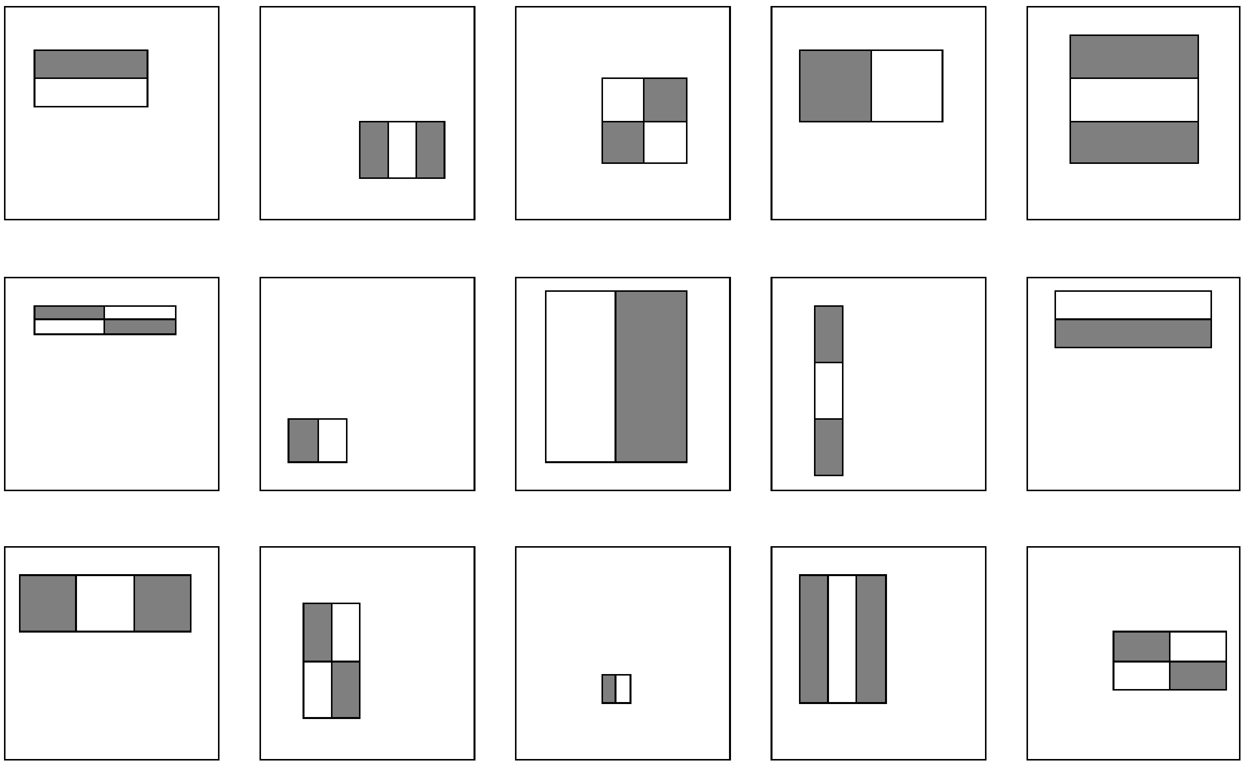
Integral image



$$
\begin{aligned}
D &= 1+4-(2+3) \\
&= A+(A+B+C+D)-(A+C+A+B) \\
&= D
\end{aligned}
$$

Viola & Jones, CVPR 2001

Large library of filters



Considering all possible filter parameters: position, scale, and type:

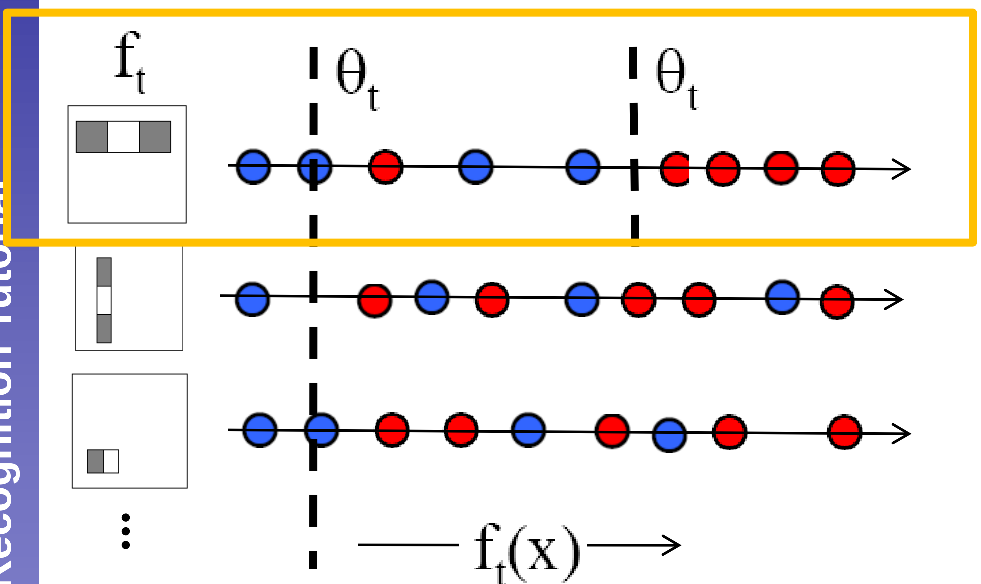
180,000+ possible features associated with each 24 x 24 window

Use AdaBoost both to select the informative features and to form the classifier

Viola & Jones, CVPR 2001

AdaBoost for feature+classifier selection

- Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of *weighted* error.



Outputs of a possible rectangle feature on faces and non-faces.

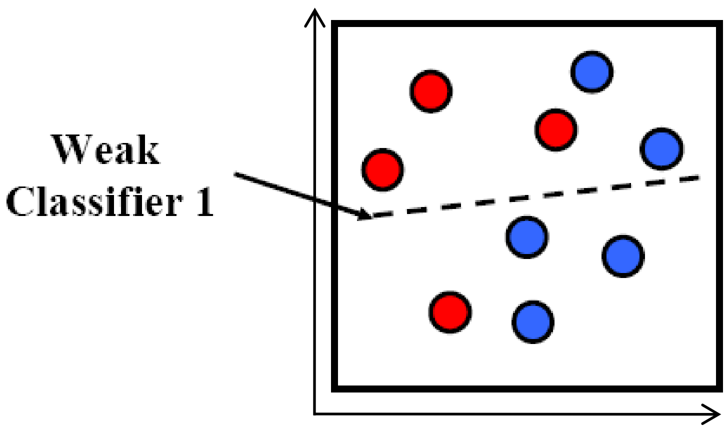
Resulting weak classifier:

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

For next round, reweight the examples according to errors, choose another filter/threshold combo.

Viola & Jones, CVPR 2001

AdaBoost: Intuition



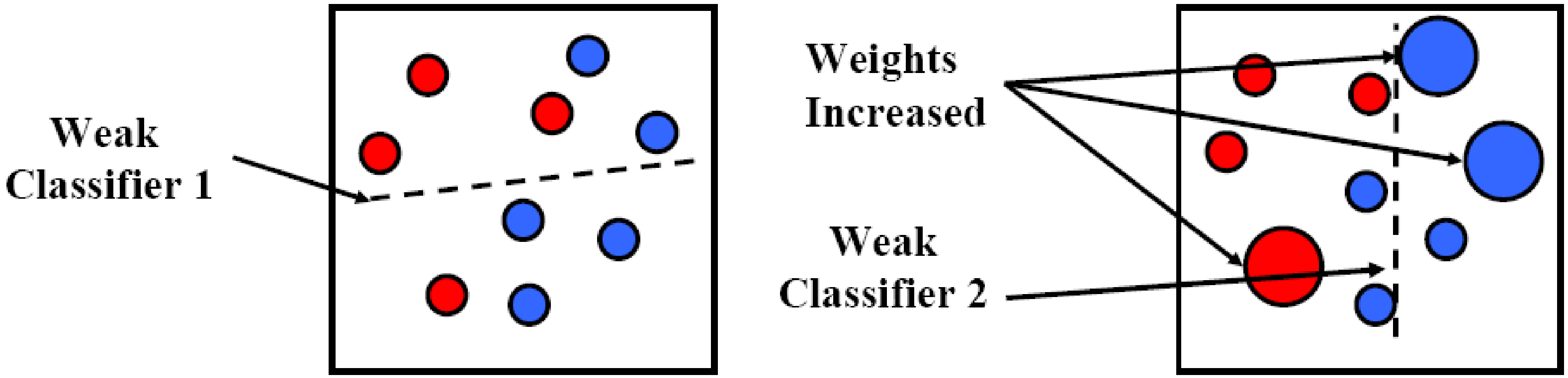
Consider a 2-d feature space with positive and negative examples.

Each weak classifier splits the training examples with at least 50% accuracy.

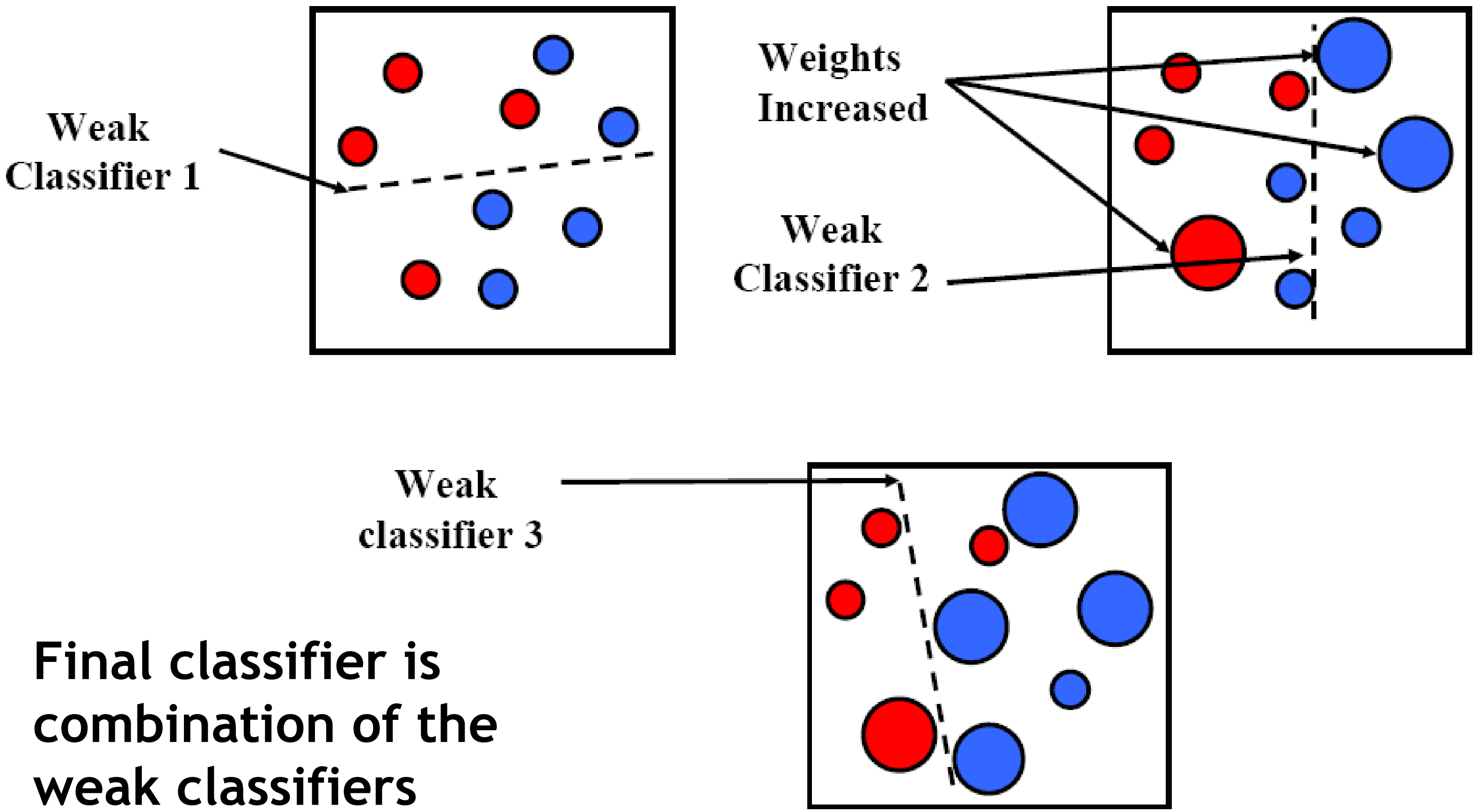
Examples misclassified by a previous weak learner are given more emphasis at future rounds.

Figure adapted from Freund and Schapire

AdaBoost: Intuition



AdaBoost: Intuition

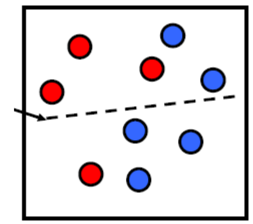


Final classifier is combination of the weak classifiers

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ 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 $t = 1, \ldots, T$:
  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 ϵ_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 $\alpha_t = \log \frac{1}{\beta_t}$

AdaBoost Algorithm

Start with uniform weights on training examples



$\{x_1, \ldots x_n\}$

For T rounds

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

Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

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

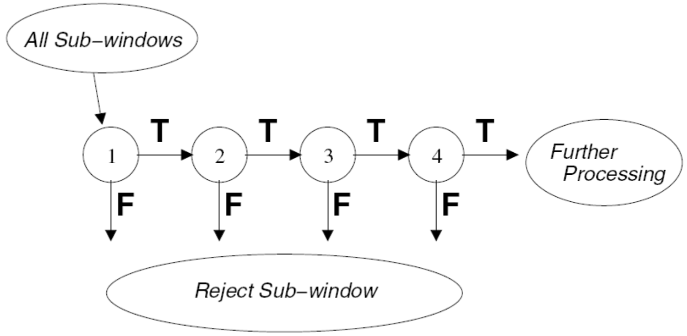
an, B. Leibe

Freund & Schapire 1995

Cascading classifiers for detection

For efficiency, apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative; e.g.,

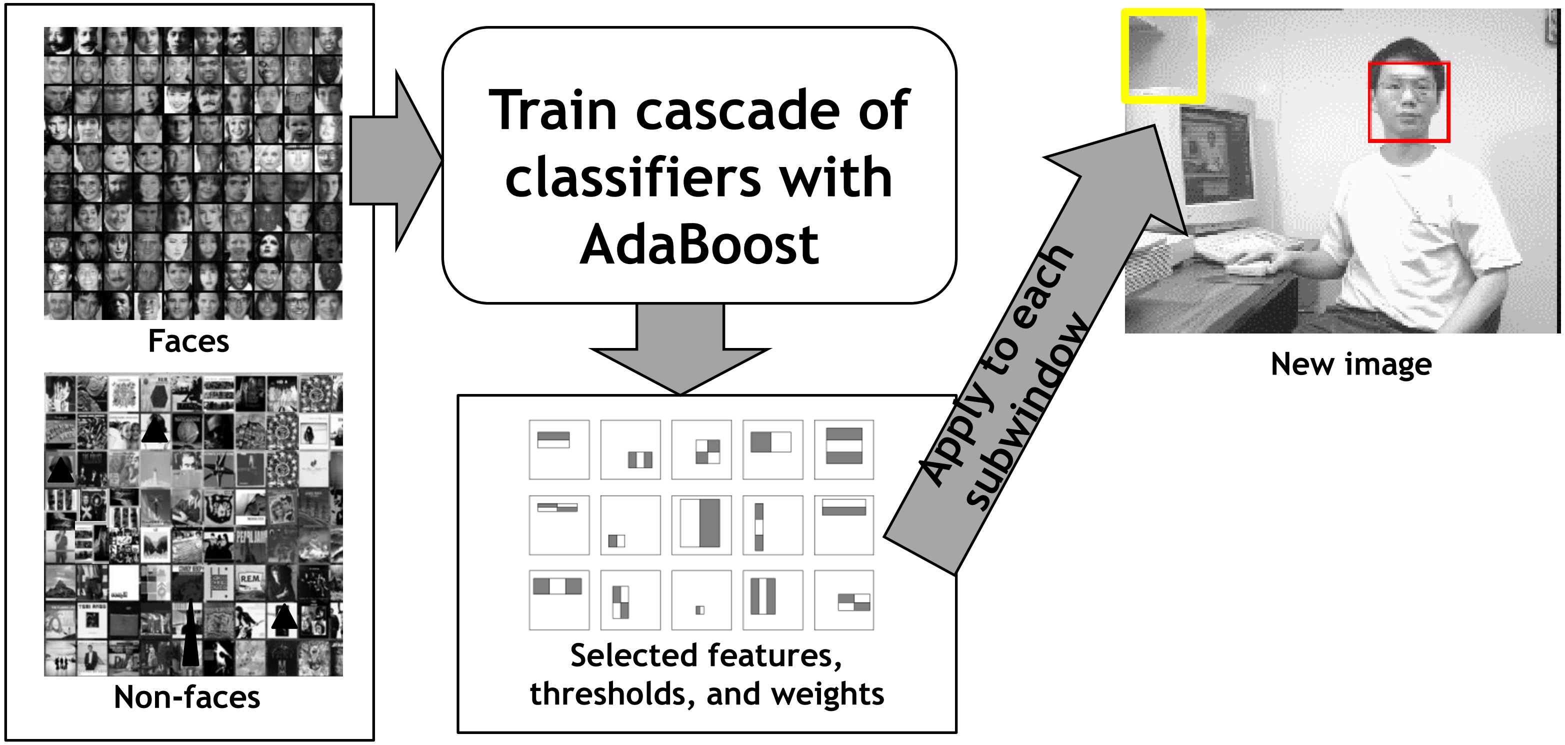
- Filter for promising regions with an initial inexpensive classifier
- Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain



Fleuret & Geman, IJCV 2001
Rowley et al., PAMI 1998
Viola & Jones, CVPR 2001

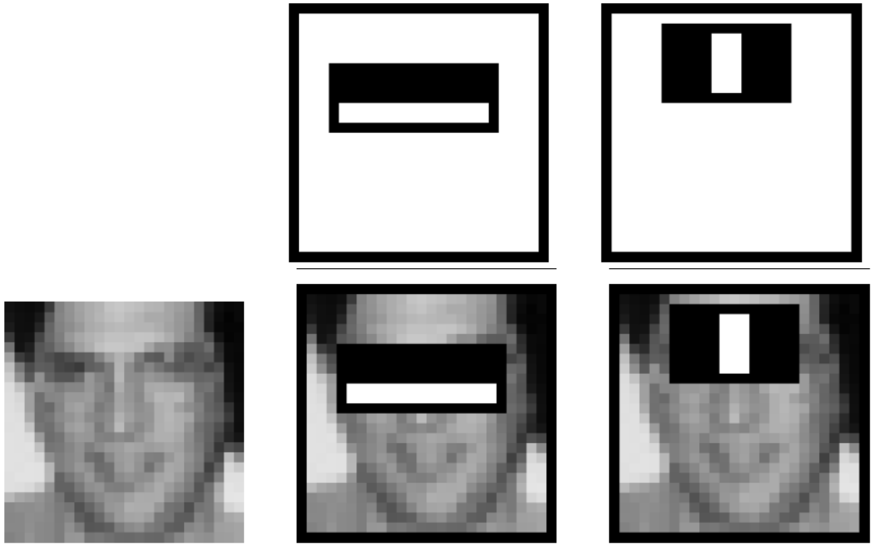
Figure from Viola & Jones CVPR 2001

Viola-Jones Face Detector: Summary



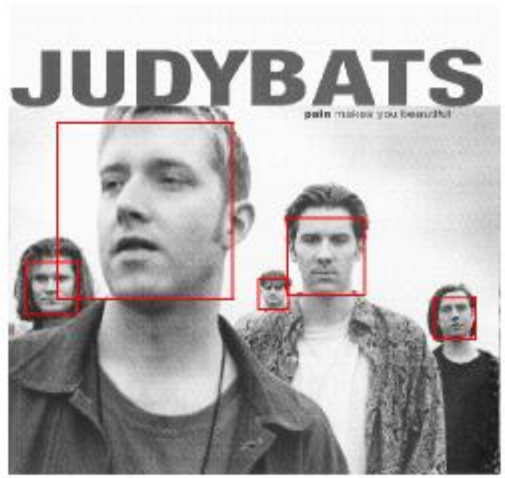
- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]

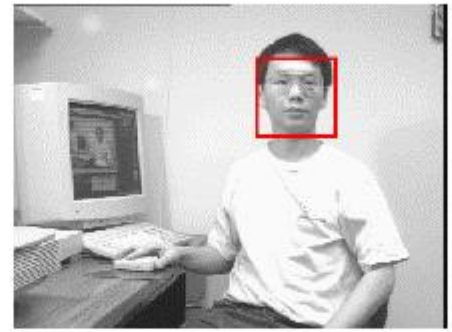
Viola-Jones Face Detector: Results

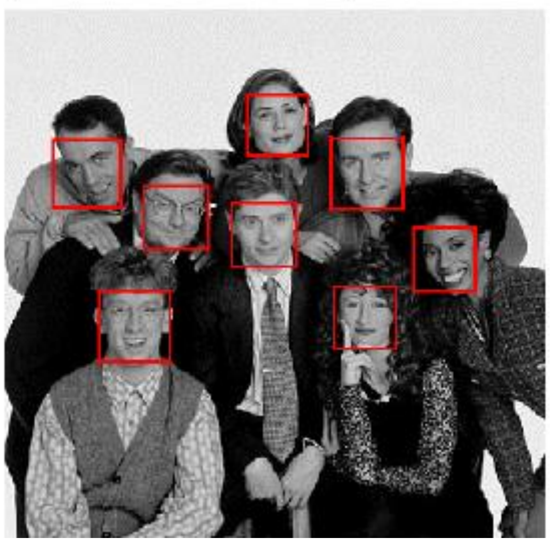


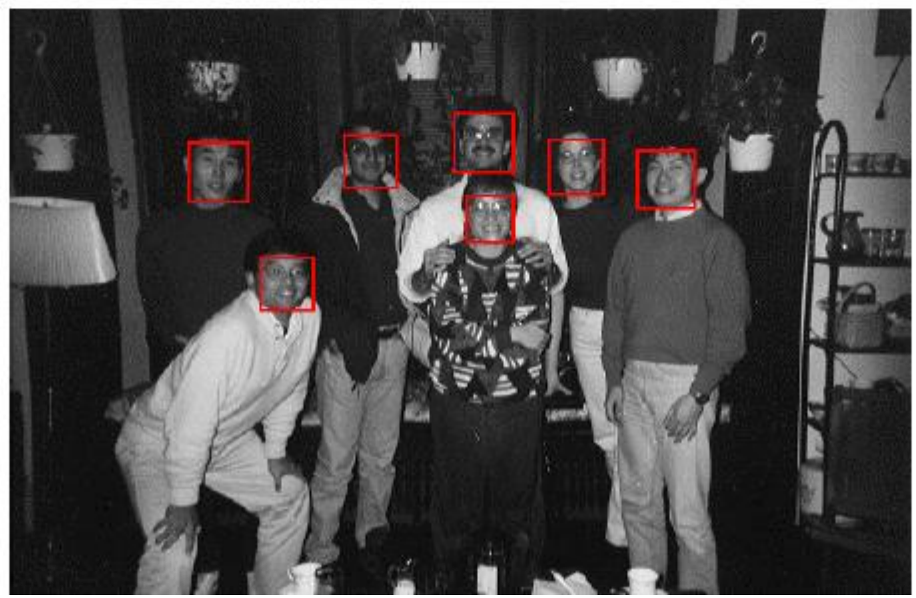
First two features selected

Viola-Jones Face Detector: Results

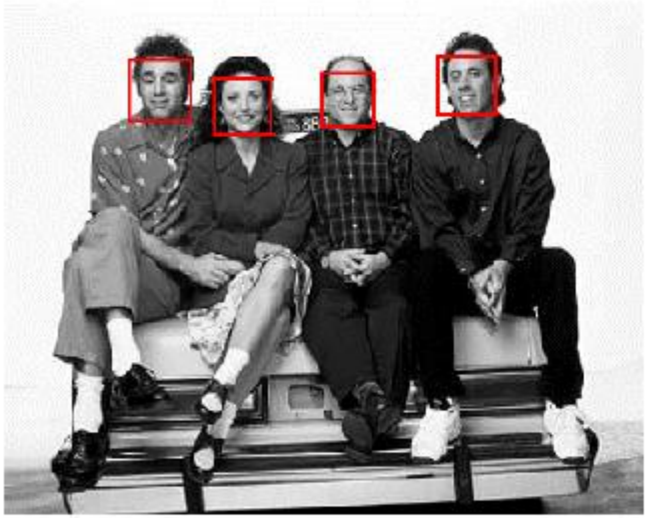


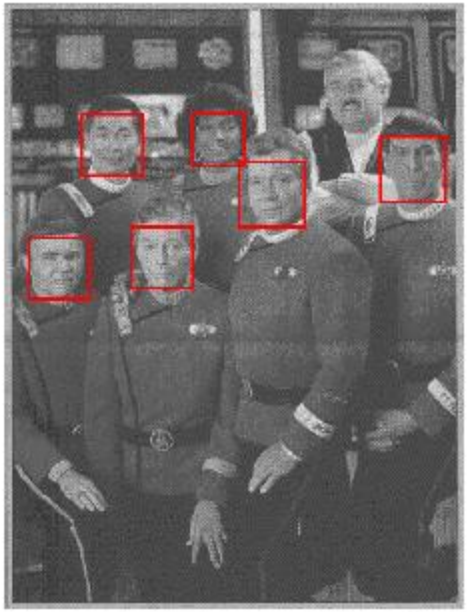


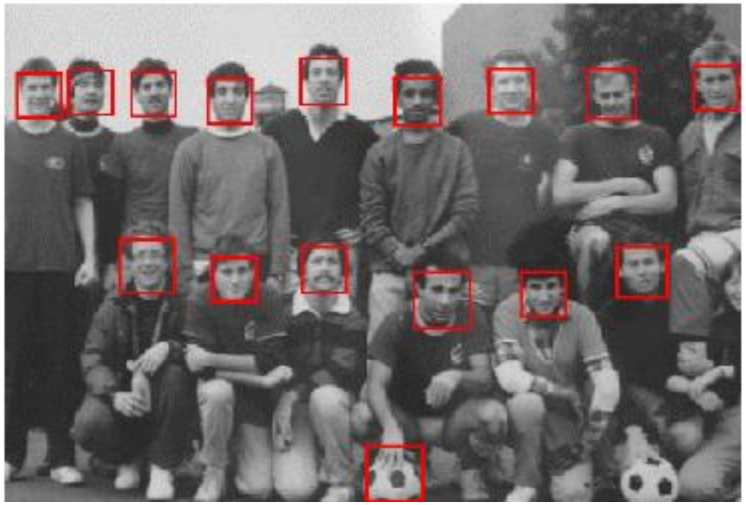


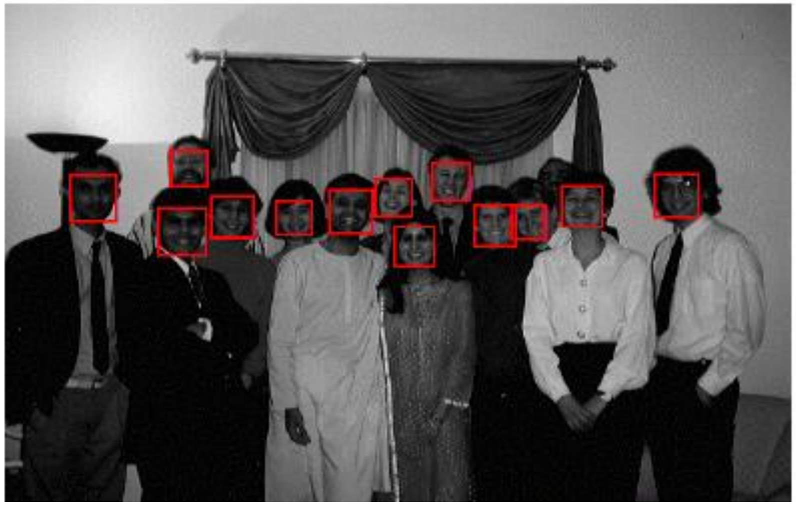


Viola-Jones Face Detector: Results





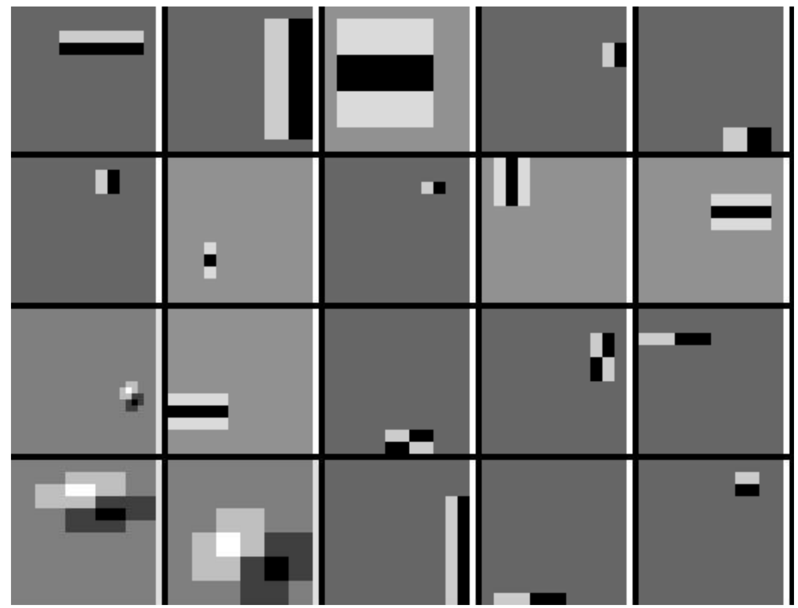




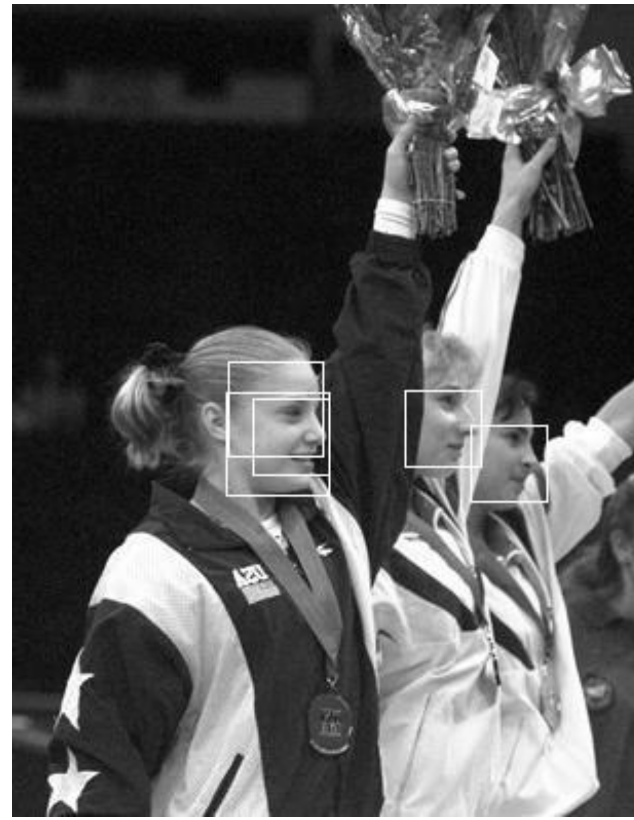
Viola-Jones Face Detector: Results

Detecting profile faces?

Detecting profile faces requires training separate detector with profile examples.



Viola-Jones Face Detector: Results



Questions?

- 3-minute break

Moving forward

- Faces are pretty well-behaved
  - Mostly the same basic shape
  - Lie close to a subspace of the set of images

- Not all objects are as nice

Different appearance, similar parts

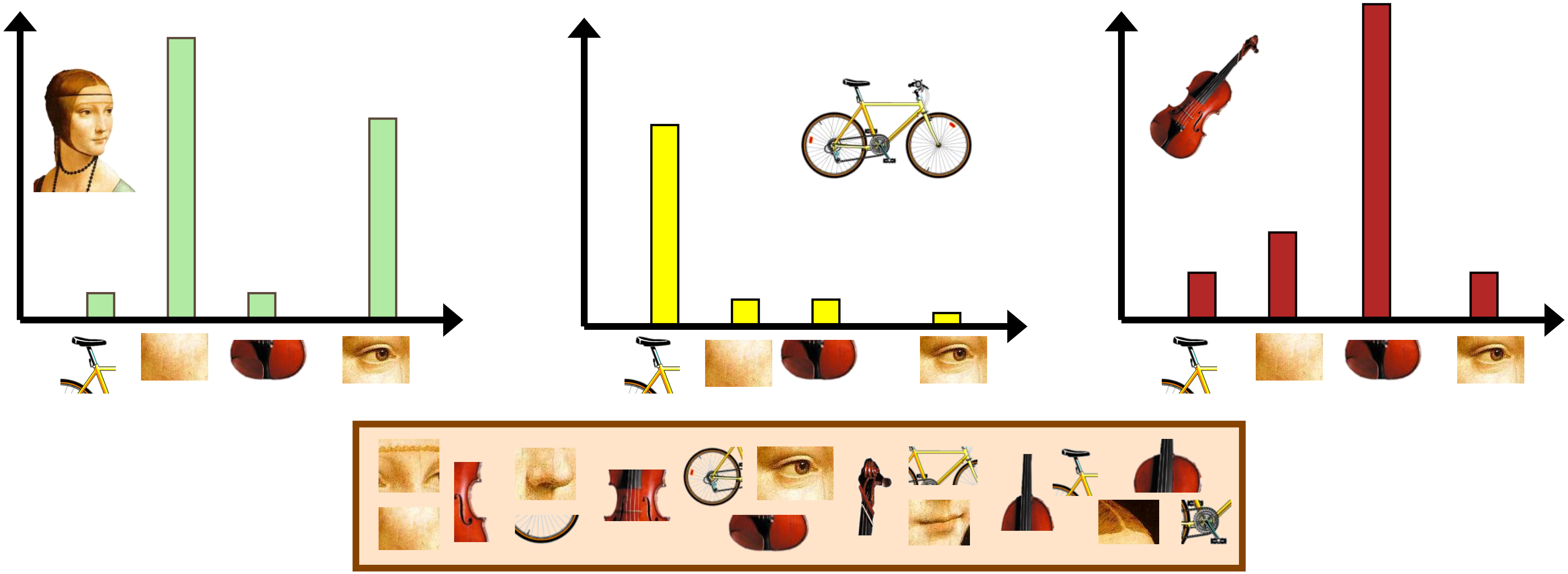
Bag of Words Models

Adapted from slides by Rob Fergus

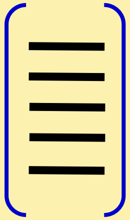
Object → **Bag of 'words'**

Bag of Words

- Independent features
- Histogram representation

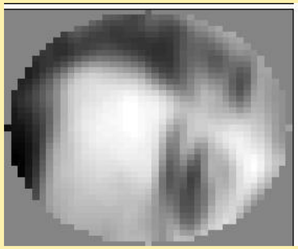


1.Feature detection and representation

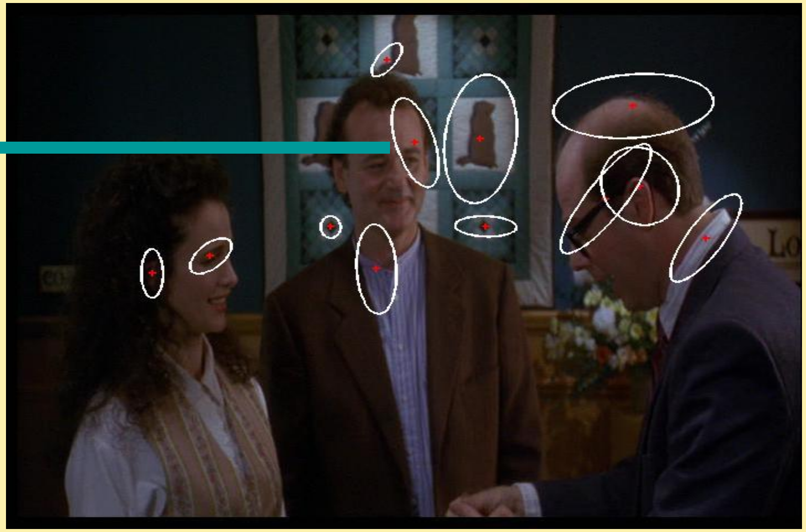


Compute descriptor

e.g. **SIFT** [Lowe'99]



Normalize patch



Detect patches

[Mikojaczyk and Schmid '02]

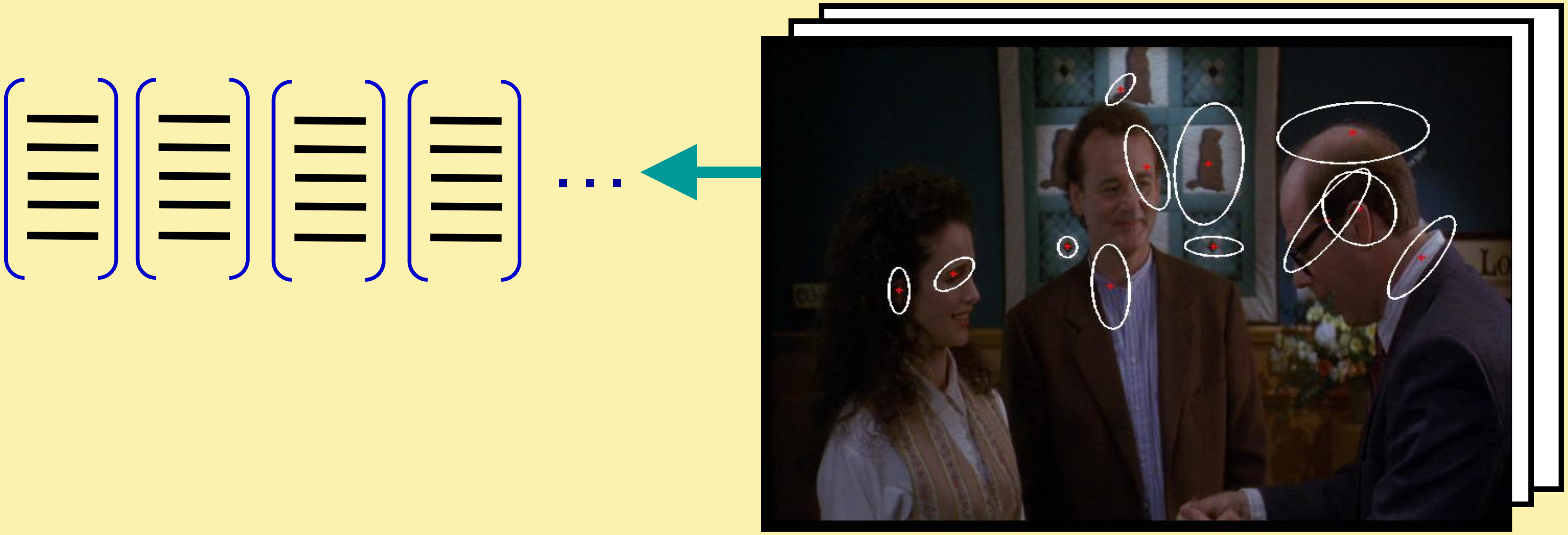
[Mata, Chum, Urban & Pajdla, '02]

[Sivic & Zisserman, '03]

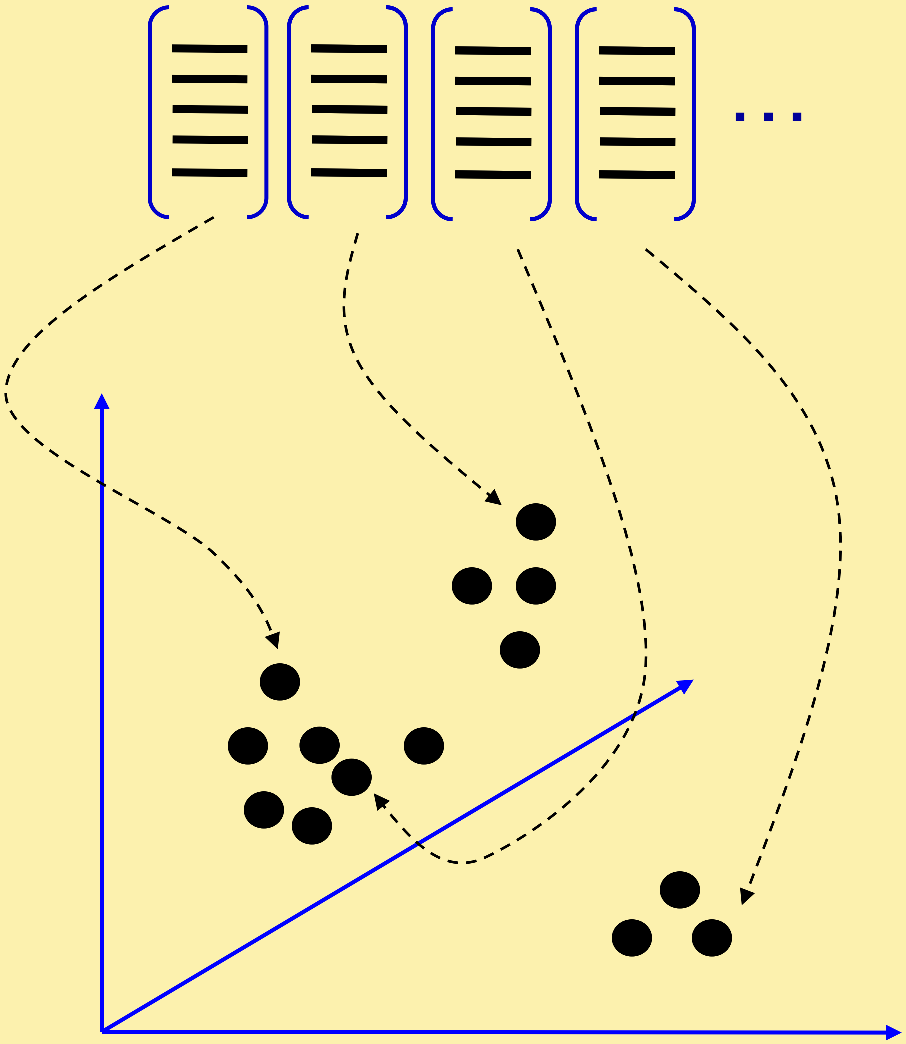
Local interest operator
or
Regular grid

Slide credit: Josef Sivic

1.Feature detection and representation

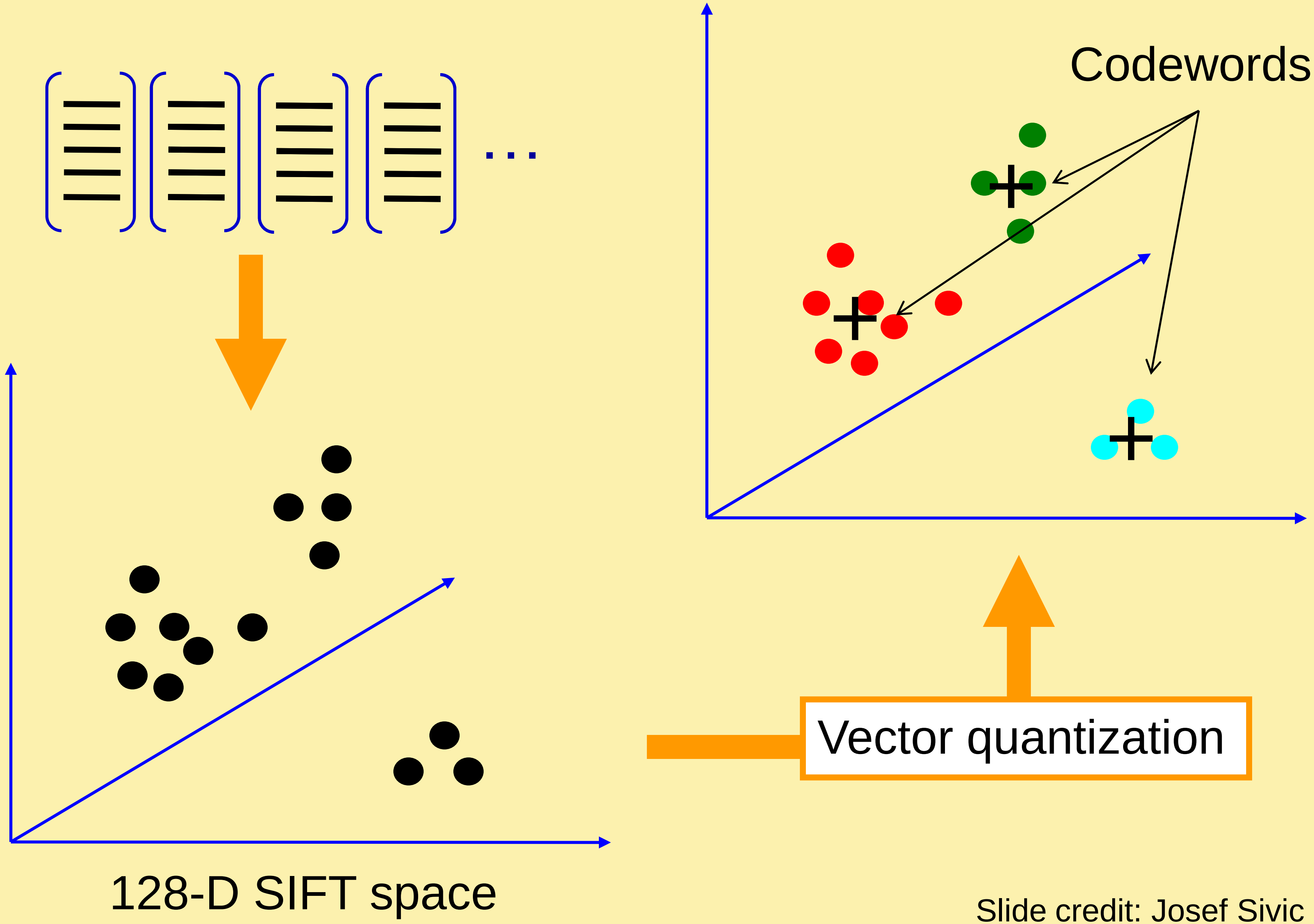


2. Codewords dictionary formation



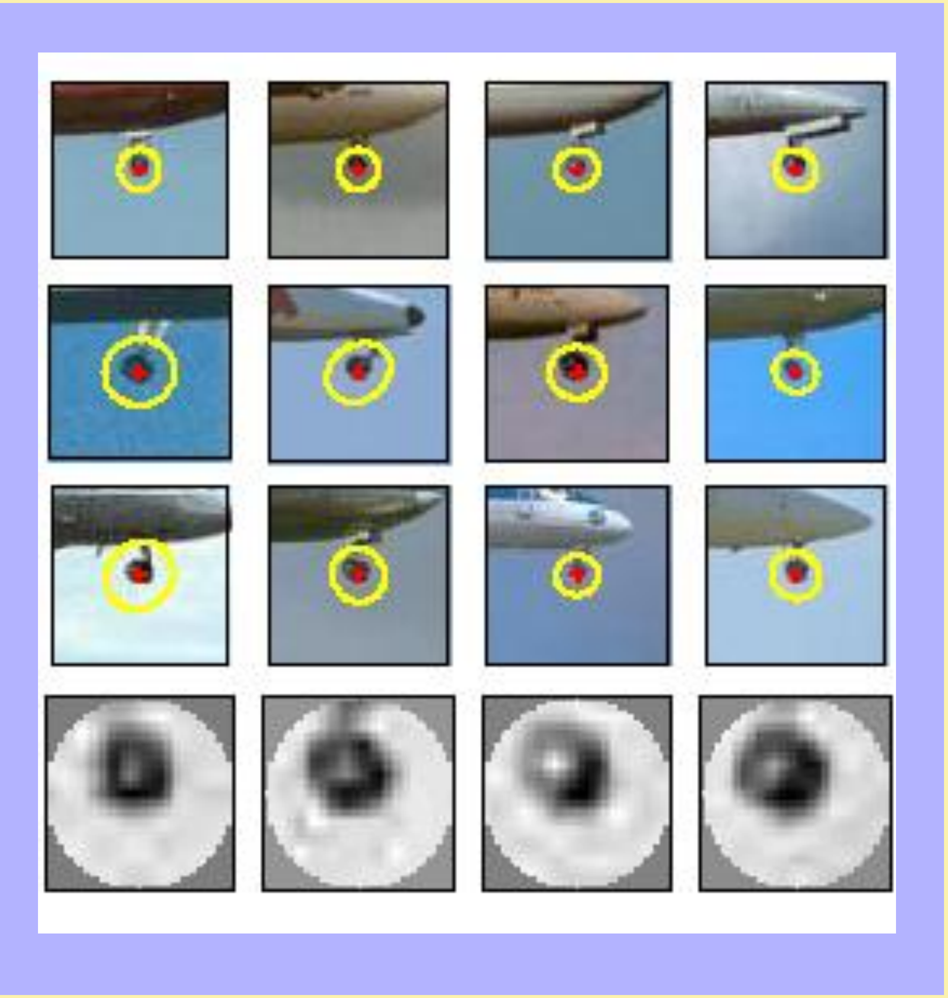
128-D SIFT space

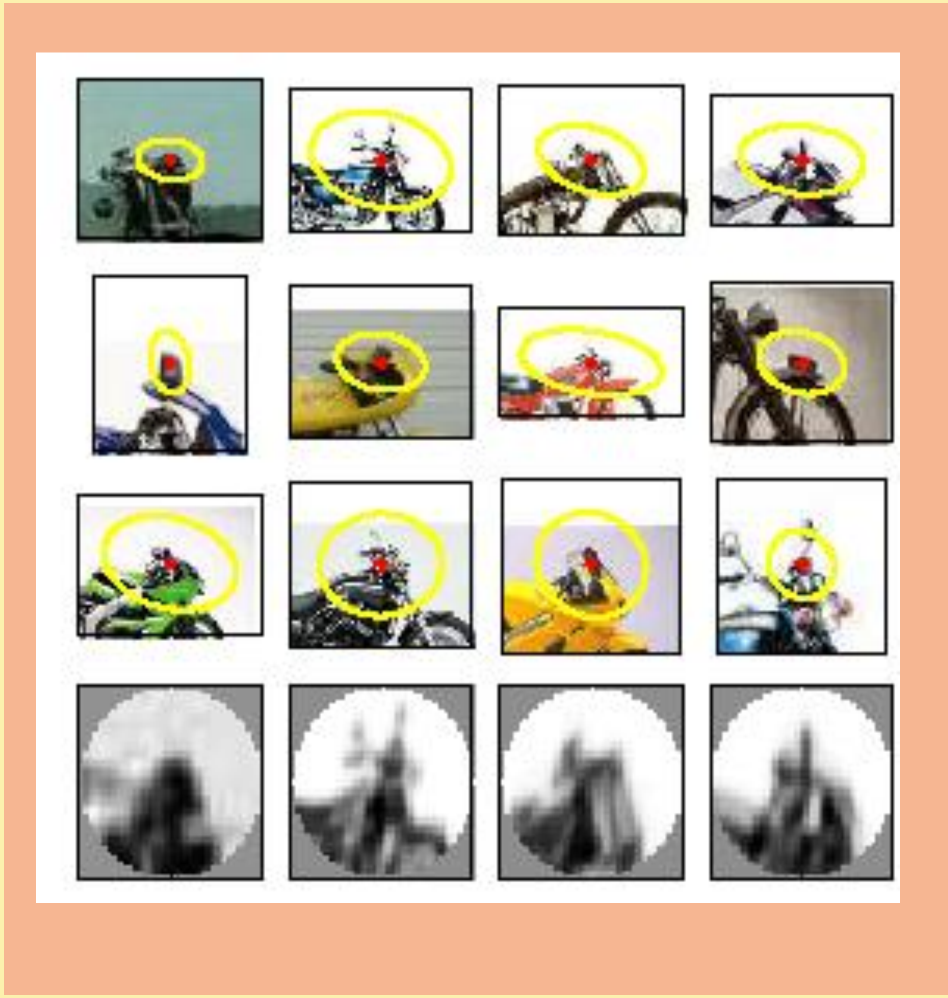
2. Codewords dictionary formation



Slide credit: Josef Sivic

Image patch examples of codewords

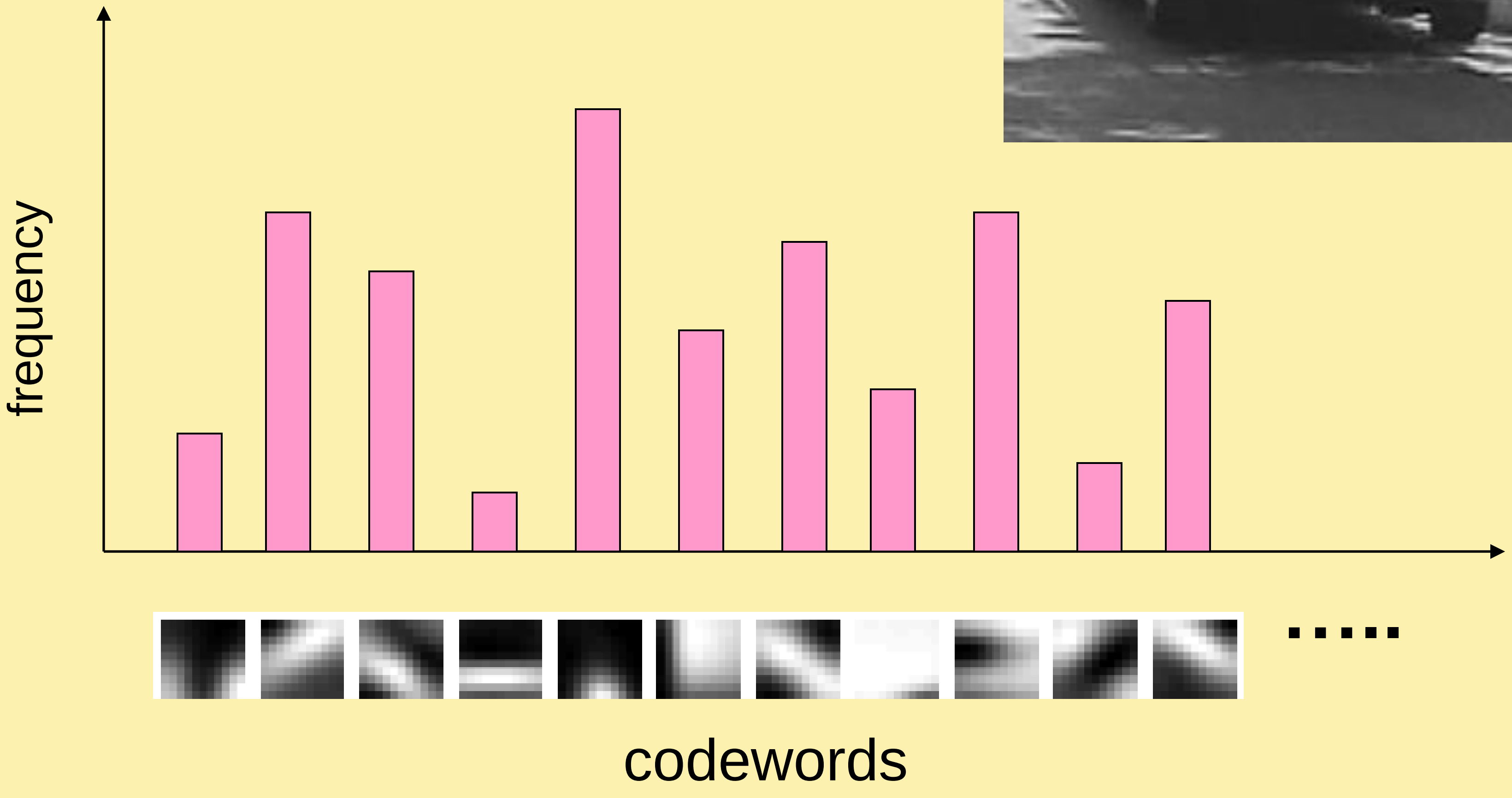




Sivic et al. 2005

Image representation

Histogram of features assigned to each cluster



Uses of BoW representation

- Treat as feature vector for standard classifier
  - e.g k-nearest neighbors, support vector machine
- Cluster BoW vectors over image collection
  - Discover visual themes

What about spatial info?