Object Detection and Recognition

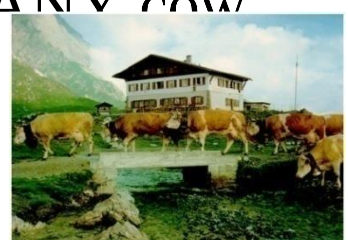
Object Categorization

- How to recognize ANY car





- How to recognize ANY cow

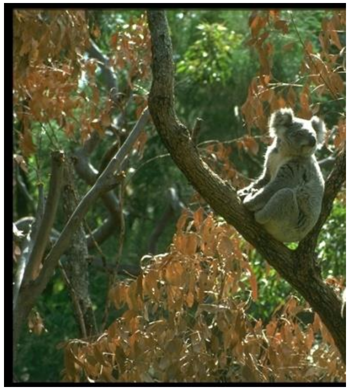



K. Grauman, B. Leibe

Challenges: robustness

Illumination

Object pose



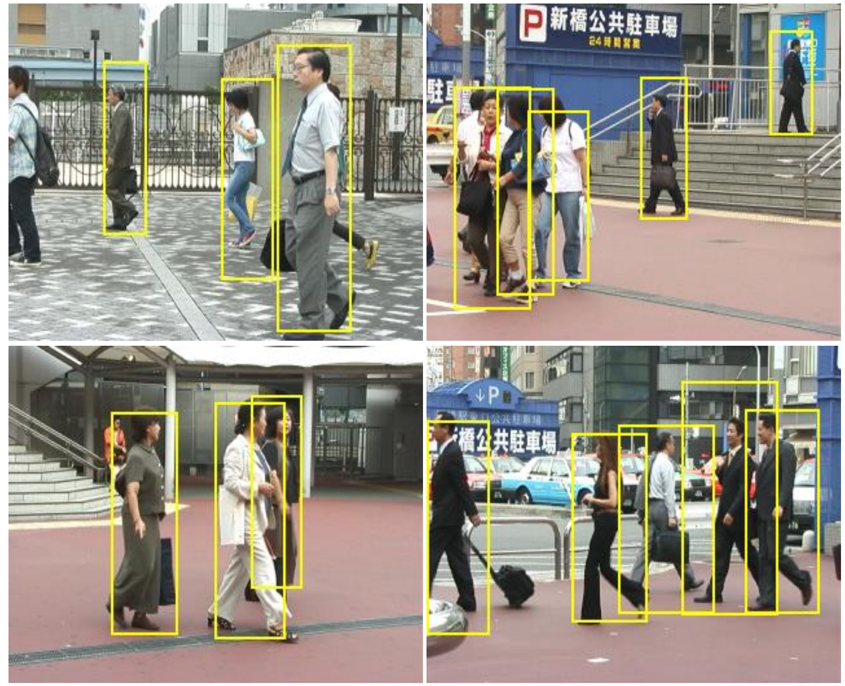
Clutter

Occlusions

Intra-class appearance

Viewpoint

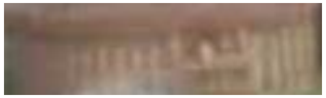
Challenges: robustness



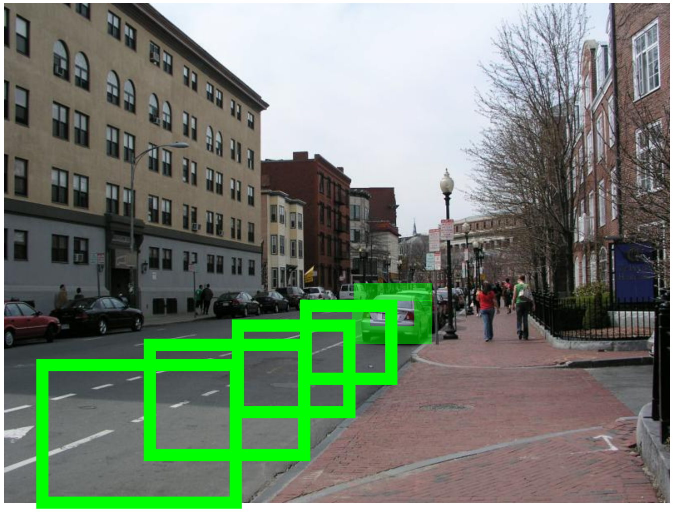
- Detection in Crowded Scenes
  - Learn object variability
    - Changes in appearance, scale, and articulation
  - Compensate for clutter, overlap, and occlusion

K. Grauman, B. Leibe

Challenges: context and human experience



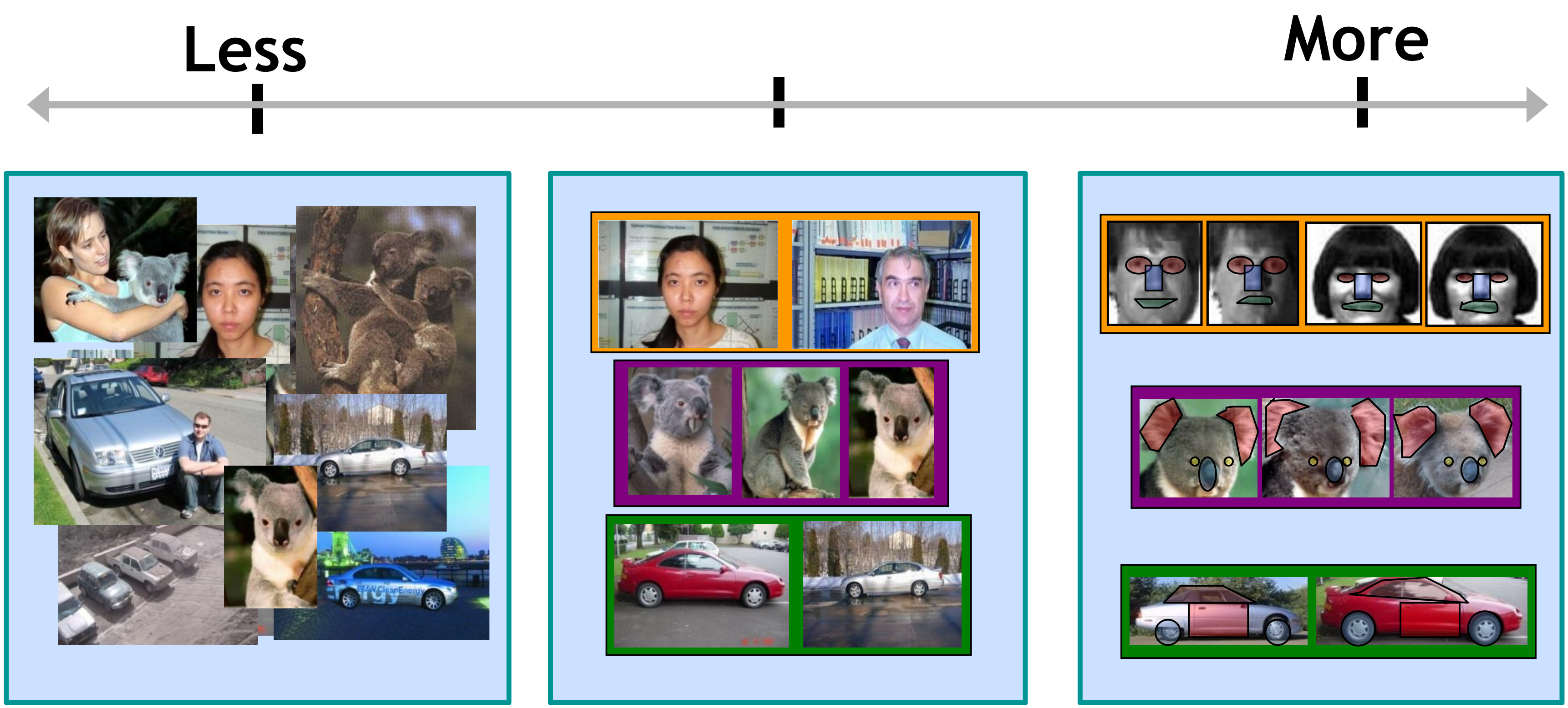
Challenges: context and human experience



Context cues

Image credit: D. Hoeim

Challenges: learning with minimal supervision



Unlabeled, multiple objects

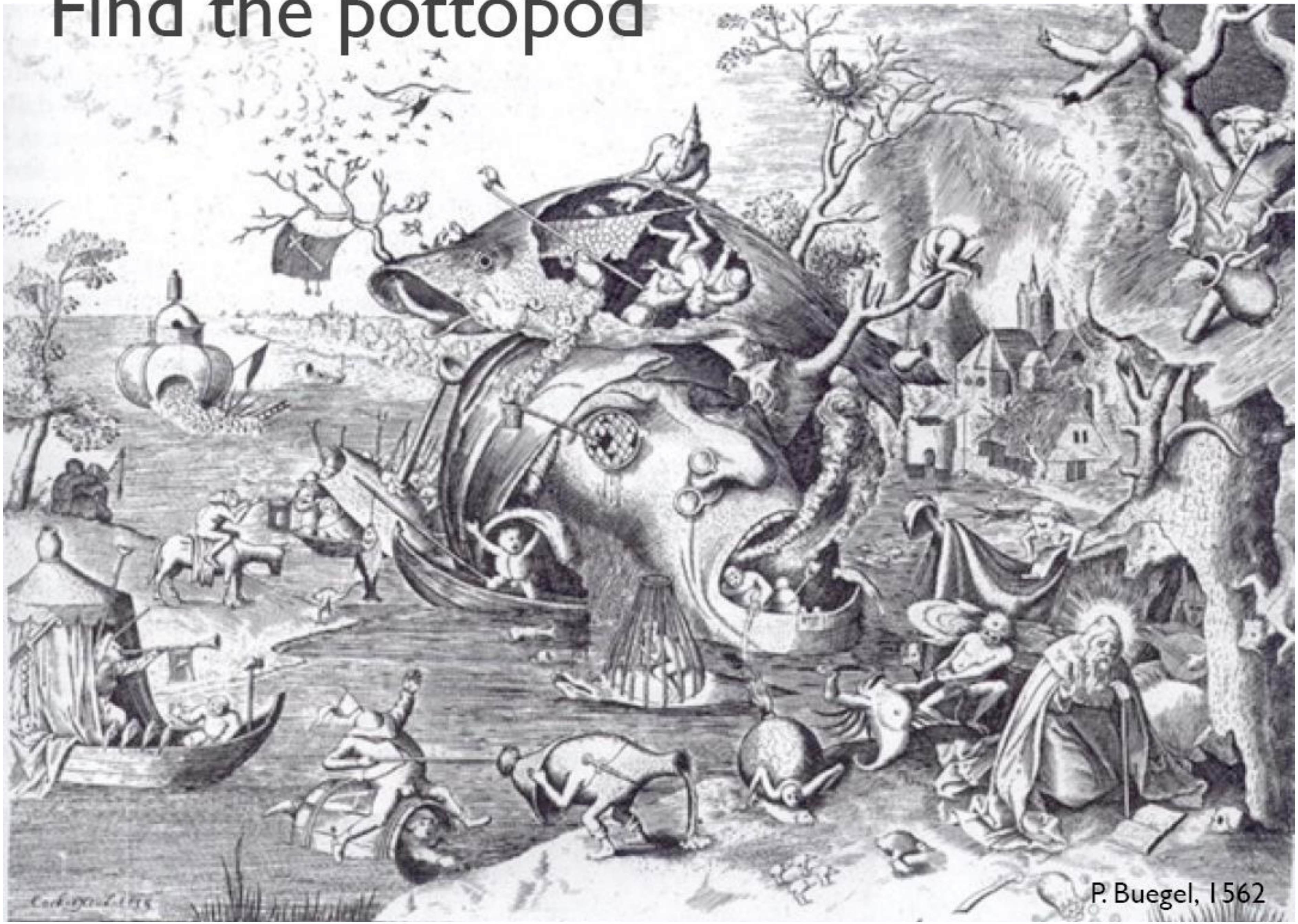
Classes labeled, some clutter

Cropped to object, parts and classes labeled

This is a pottopod

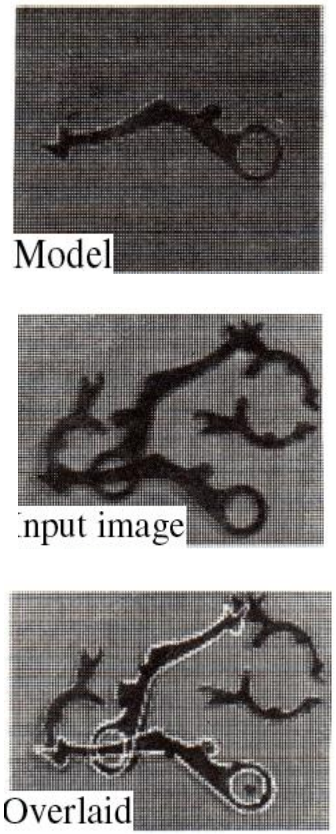
Slide from Pietro Perona, 2004 Object Recognition workshop

Find the pottopod



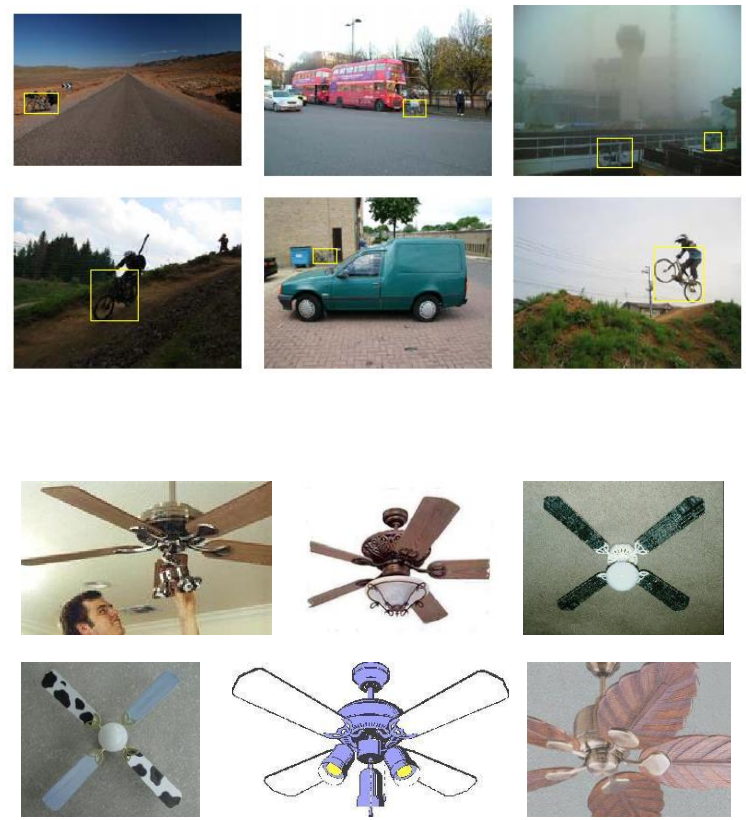
Slide from Pietro Perona, 2004 Object Recognition workshop

Rough evolution of focus in recognition research



1980s

1990s to early 2000s



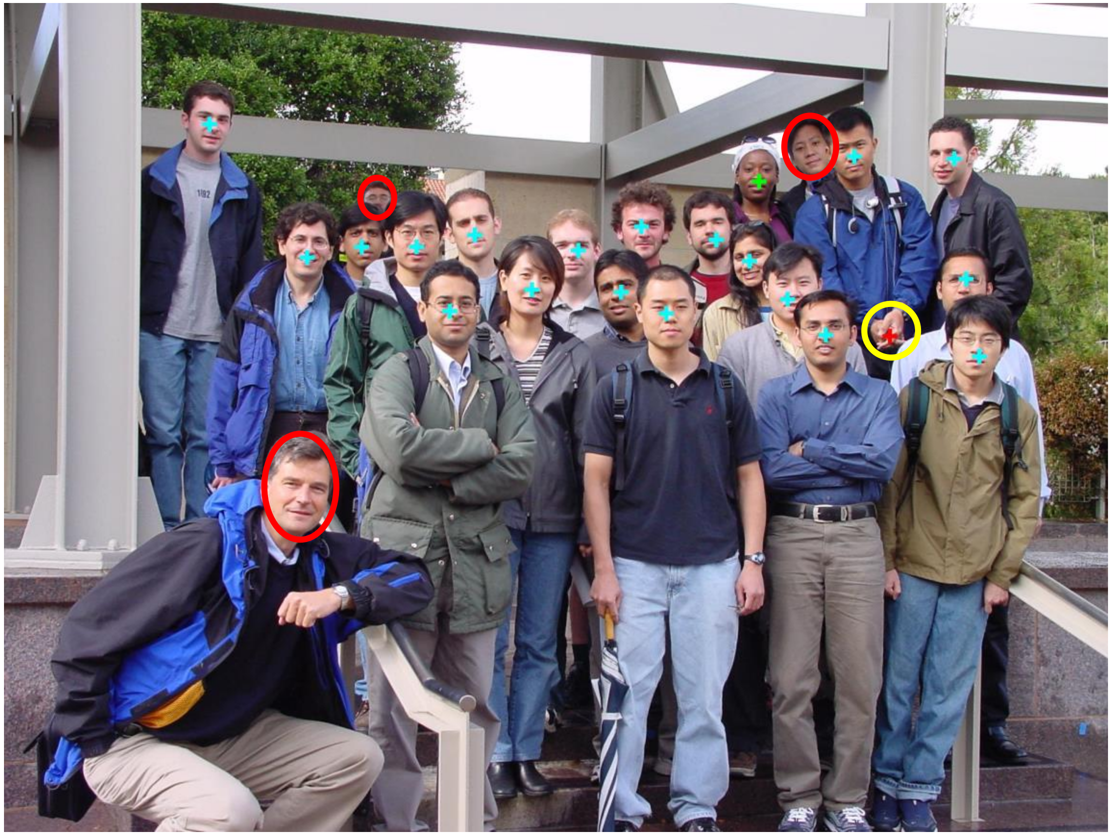
2000-2010…

Detection, recognition, and classification

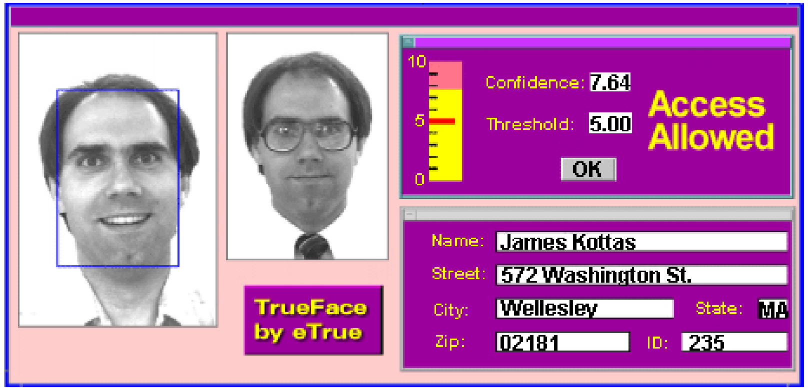
- Detection = 2-class classification problem
  - Object/class or not object/class
  - E.g., detect all the faces in this image
- Recognition of identity = within-class classification problem
  - Within a given class of objects (e.g., faces, logos), identify the object as one particular member of the class (e.g., Joe's face, Nike logo)
- Recognition of class = among-class classification
  - Which class of things is this: sky, cloud, forest, face, …

Example: Face detection

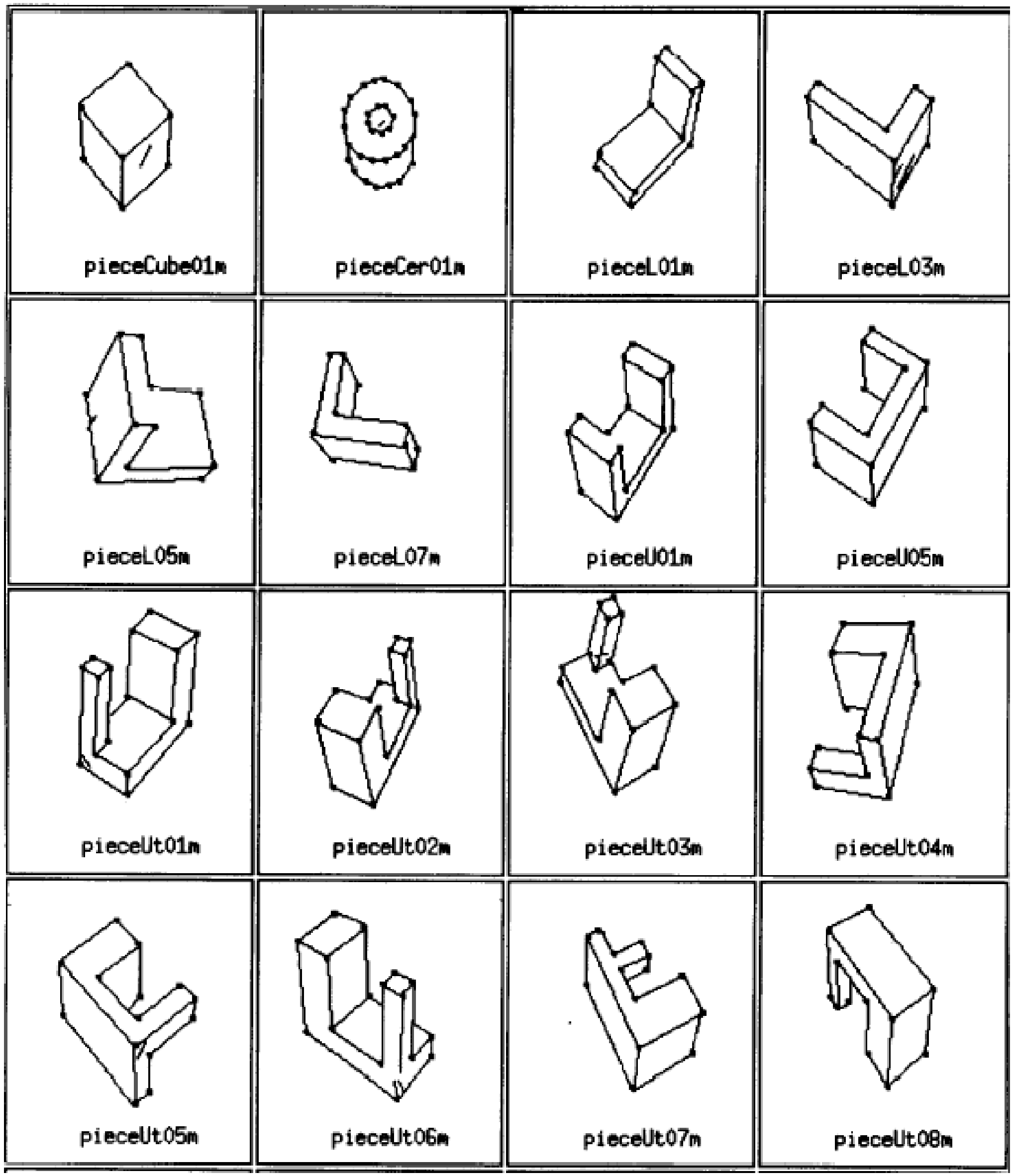
Found Face at [x=108, y=80]
Found Face at [x=76, y=73]
Found Face at [x=257, y=99]
Found Face at [x=154, y=44]
Found Face at [x=211, y=100]
Found Face at [x=147, y=97]

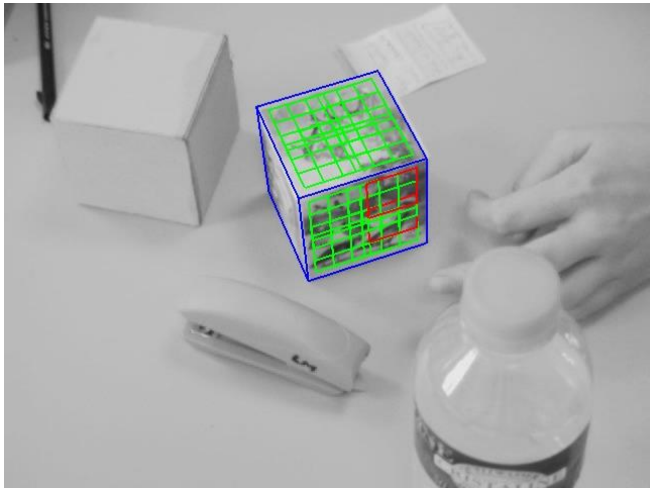


Example: Face recognition



Example: Polyhedral object recognition





Approaches to detection and classification

- There are many approaches to object detection and recognition, depending on how the object is modeled
  - Template-based: Match an image template (or a family of image templates) of the desired object
  - Feature-based: Derive image features and then match with feature model of object
    - Colors, texture, edges, corners, …
  - Shape (2D or 3D): Describe (parameterize) the object contour or full shape, and look for that shape in the image
  - And many more…
- In some sense, all these can be viewed as three steps:
  - Modeling the object(s) ("training")
  - Preprocessing the image (computing features, shape, ...)
  - Classify based on a comparison or match between model and image data

Template matching and classification

- If we want to detect and recognize (classify) objects in images, one simple technique is to use <u>normalized correlation</u>
  - Provides a measure of how well the correlation template matches the image region
  - I.e., "template matching"
- But in general there is not just one template to match
  - E.g., in face recognition – possibly many example templates (different people, expressions, lighting, rotation, scale, …)
  - A ***classifier*** takes an input feature set and produces an output class label

Classifiers

- A classifier assigns a label to any new example
  - E.g., the object name
  - Classes: {Joe, Bob, Mary, Fred, Lisa, unknown}
- A two-class classifier is a detector
  - Classes: {face, no face}
- The classifier is trained from a ***training set***
  - Training set: $(\boldsymbol{x}_\mathbf{1}, y_1)$, $(\boldsymbol{x}_\mathbf{2}, y_2)$, $(\boldsymbol{x}_\mathbf{3}, y_3)$, ….
  - x_i – measurements (image, features, histogram, …)
  - y_i – labels
  - This is typically framed as a **learning** problem
- Outcome: $(i \rightarrow j)$ means outcome i is labeled as j
  - $(Matthew \rightarrow Ralph)$ – error
  - $(Matthew \rightarrow Matthew)$ – correct

Training set

- Training set examples:
  - (image$_1$, “Joe”), (image$_2$, “Fred”), (image$_3$, “Sue”), …
  - (color$_1$, “Face”), (color$_2$, “Hair”), (color$_3$, “Lips”), …
  - (template$_1$, “eye”), (template$_2$, “eye”), (template$_3$, “eye”), …
  - Perhaps ***negative*** examples also: (image$_i$, “Not a face”)…
- We want a rule (function) that does
  - *F*(new measurement) = label

    …with a low error rate
- Errors
  - False positives: ***Yes*** when the true answer is ***No***
  - False negatives: ***No*** when the true answer is ***Yes***
  - Misclassifications: ***A*** when the true answer is ***B***

Classification errors

- For detection (two-class)

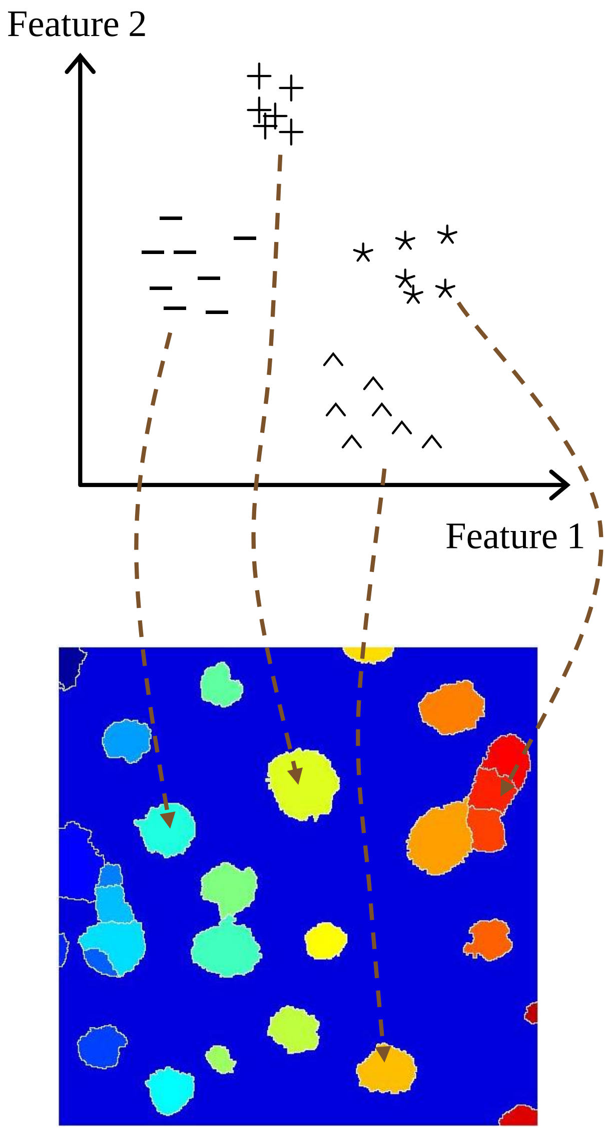
	Absent (0)	Present (1)
Not detected (0)	True negative ($0 \rightarrow 0$)	False negative ($1 \rightarrow 0$)
Detected (1)	False positive ($0 \rightarrow 1$)	True positive ($1 \rightarrow 1$)

Misclassifications: ($i \rightarrow j$), where $i \neq j$

$$\textit{Misclassifications} = \textit{False negatives} + \textit{False positives}$$

Segmentation and clustering

- Segmentation is about labeling similar pixels as belonging to the same group or segment
  - Pixels that *belong together* = pixels that *cluster*
- Clustering can be done along many dimensions (intensity, color, depth, motion, texture, ...)
  - Individually or combined
- There are some basic clustering methods that do well in certain cases
  - E.g., “k-means clustering”



Clustering/segmenting by *k*-means

- The "*k-means*" algorithm is a fast, simple way to cluster *N*-dimensional data
  - Given a bunch of data points, group them into *k* different clusters
  - Each data point is typically a feature vector
    - But could even be RGB values

- We would like to minimize the *objective function*

$$\Theta(\underset{c_i}{\text{clusters}}, \underset{x_j}{\text{data}}) = \sum_{i \in clusters} \sum_{j \in cluster(i)} (x_j - c_i)^T (x_j - c_i)$$

...but this is too expensive to do for lots of data points!

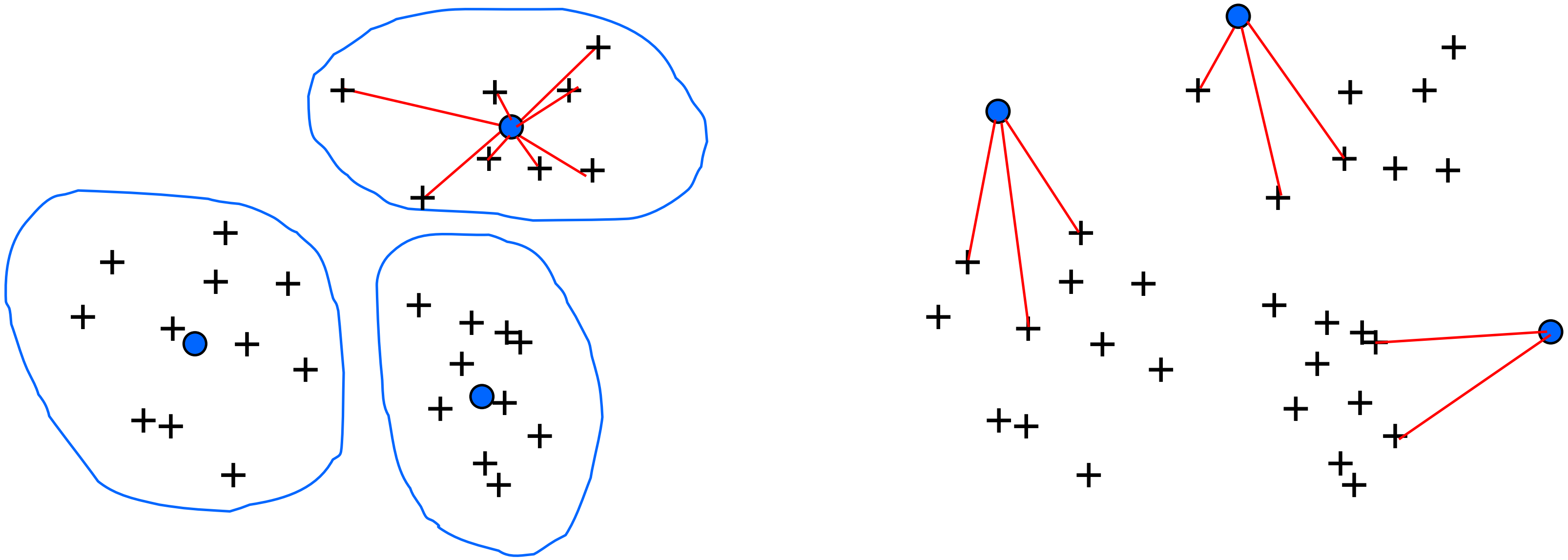
k-means clustering

Data points

Cluster centers

$$\Theta(\text{clusters}, \text{data}) = \sum_{i \in clusters} \sum_{j \in cluster(i)} (x_j - c_i)^T (x_j - c_i)$$

= the sum of the squares of the distances to cluster centers (means)

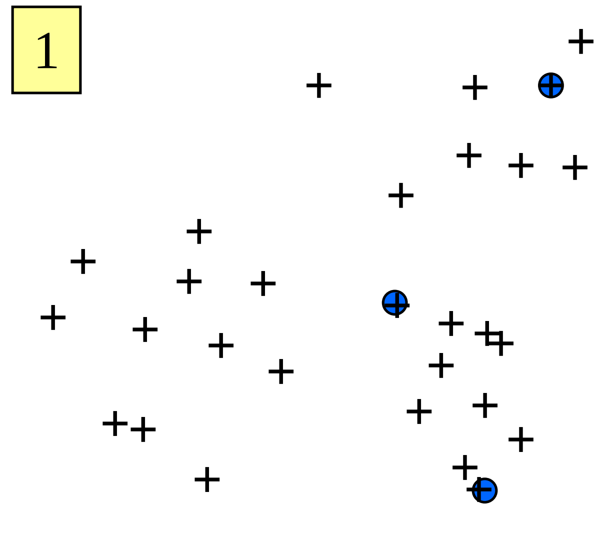


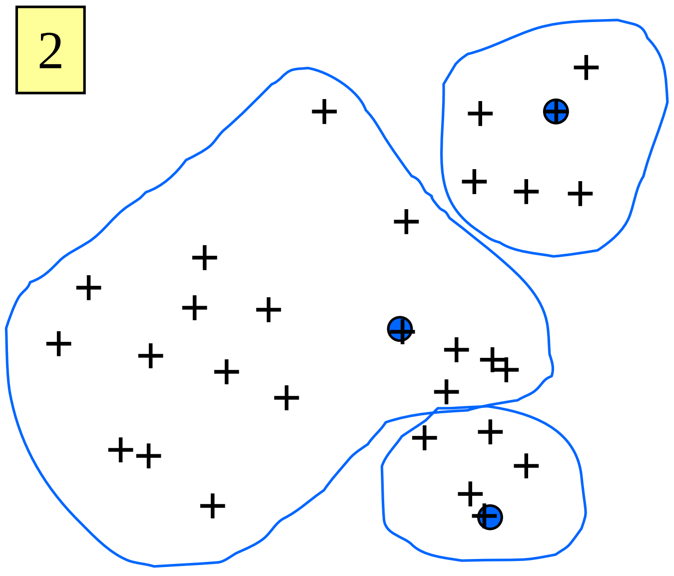
The objective function would be larger in this case

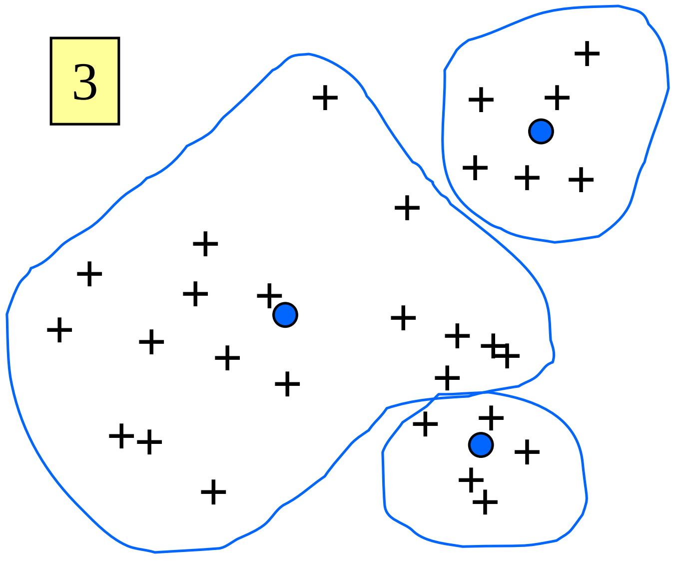
k-means clustering

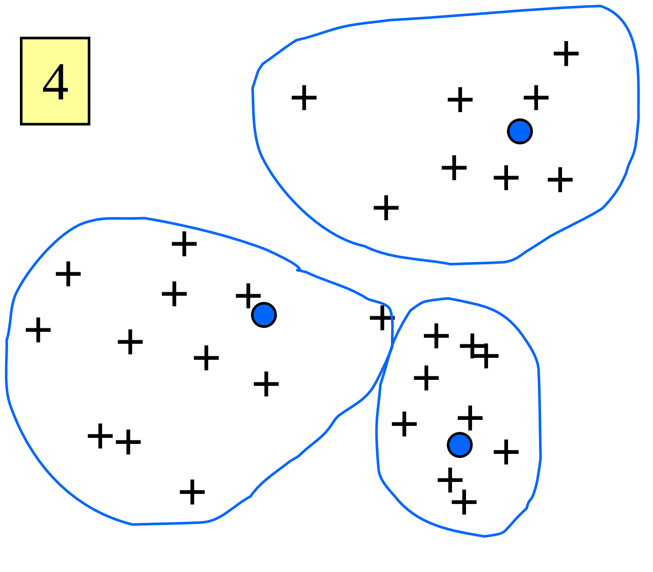
Algorithm

- Randomly choose *k* data points to be the initial cluster centers
- Iterate until centers are stable:
  - Assign each point to the nearest cluster
  - Recalculate the cluster center (mean)


1


2

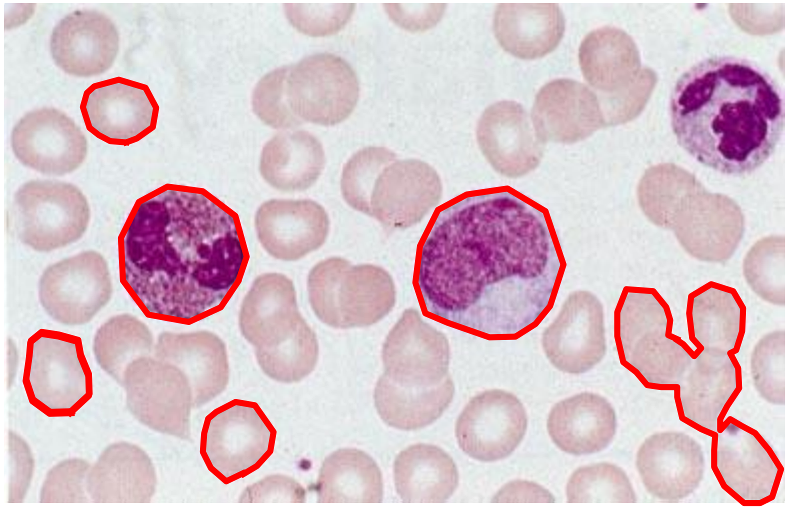

3


4

Classification/detection example

Task: Automatically detect abnormal white blood cells

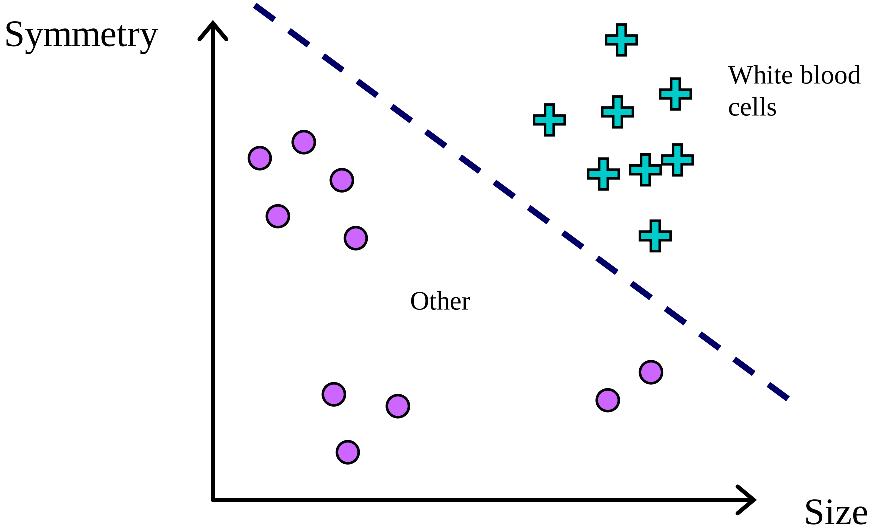
1. Process images to find outlines
2. Count white blood cells
3. Classify abnormal white blood cells

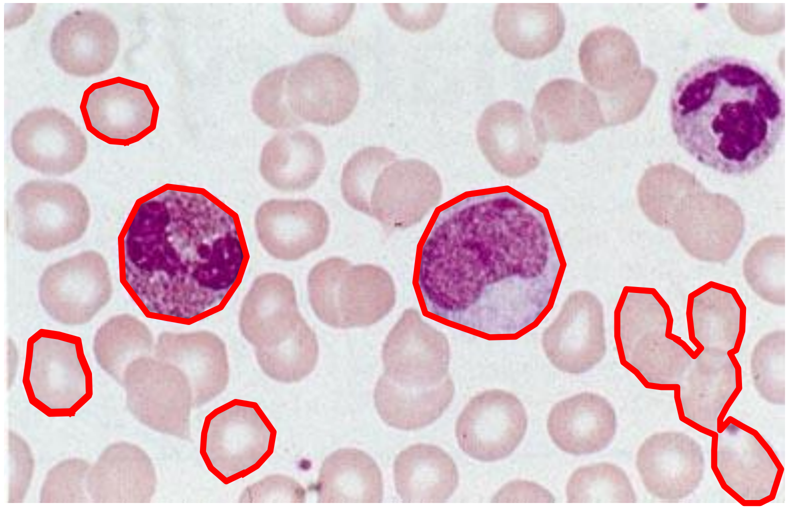


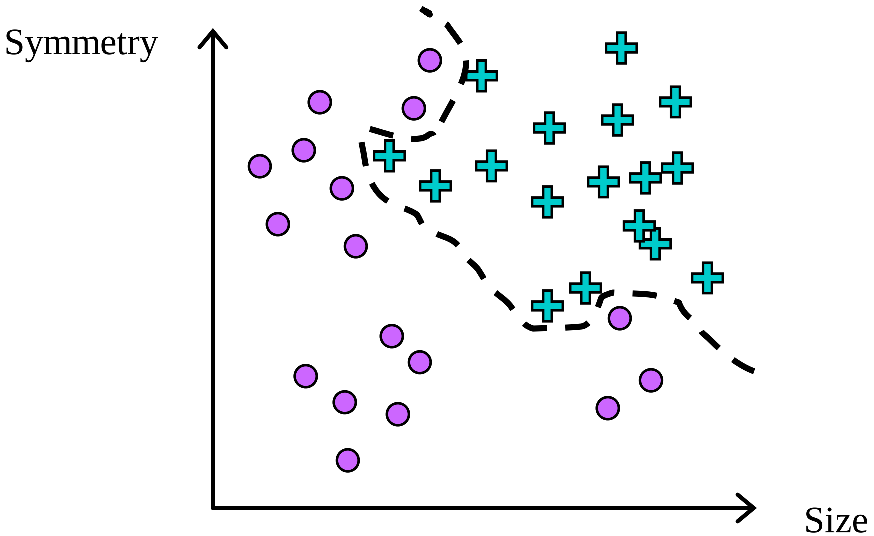
Steps 2 and 3 require *training* – teaching the system how to distinguish between white blood cells and others, and between normal and abnormal white blood cells

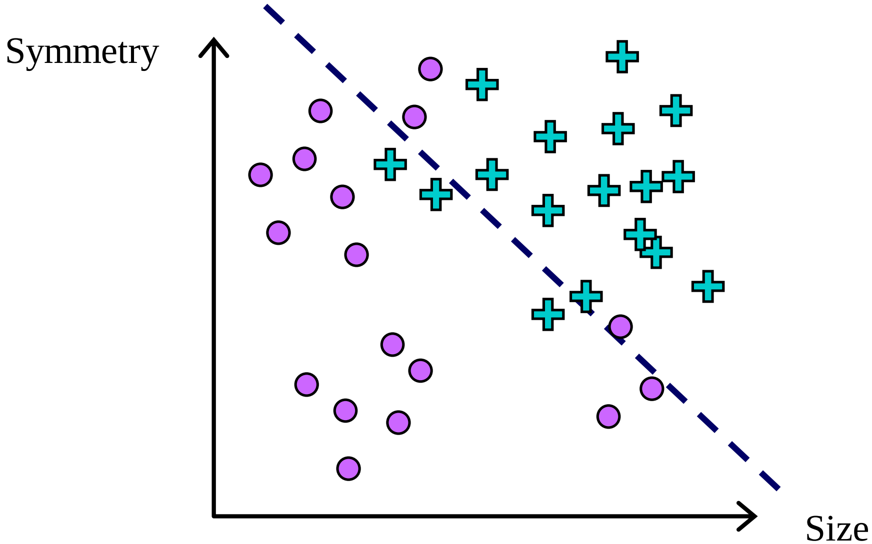
It’s very important to choose good, discriminating features

Classification/detection example

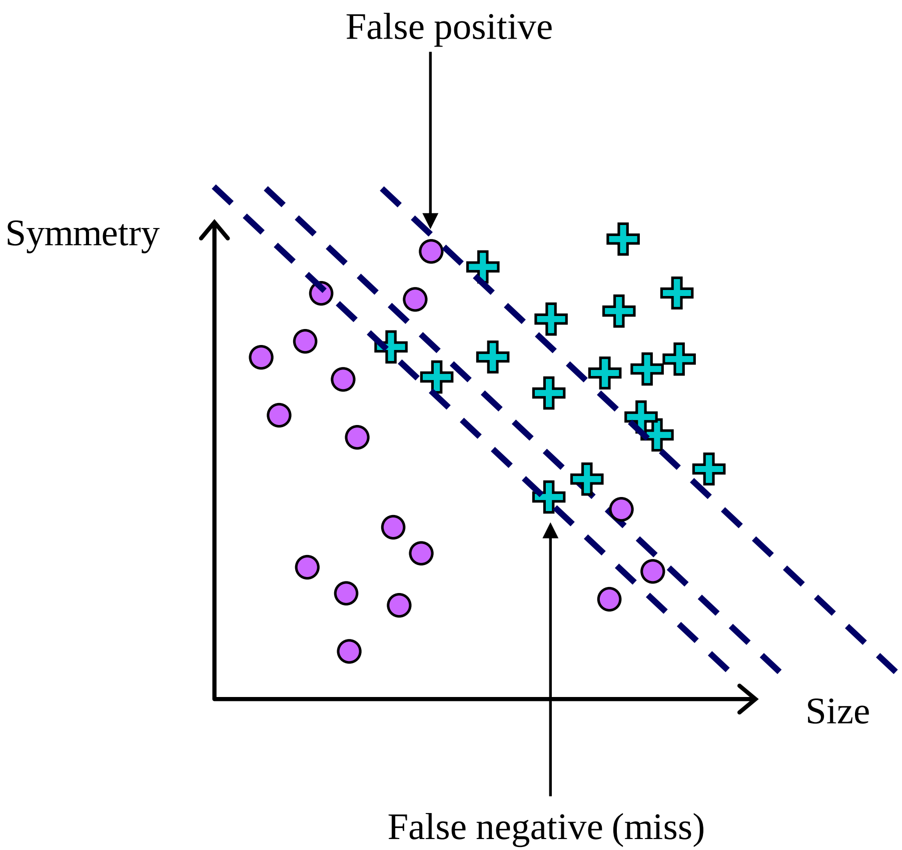








Where to place the boundary?



We wish to minimize false positives and false negatives

Training vs. Testing

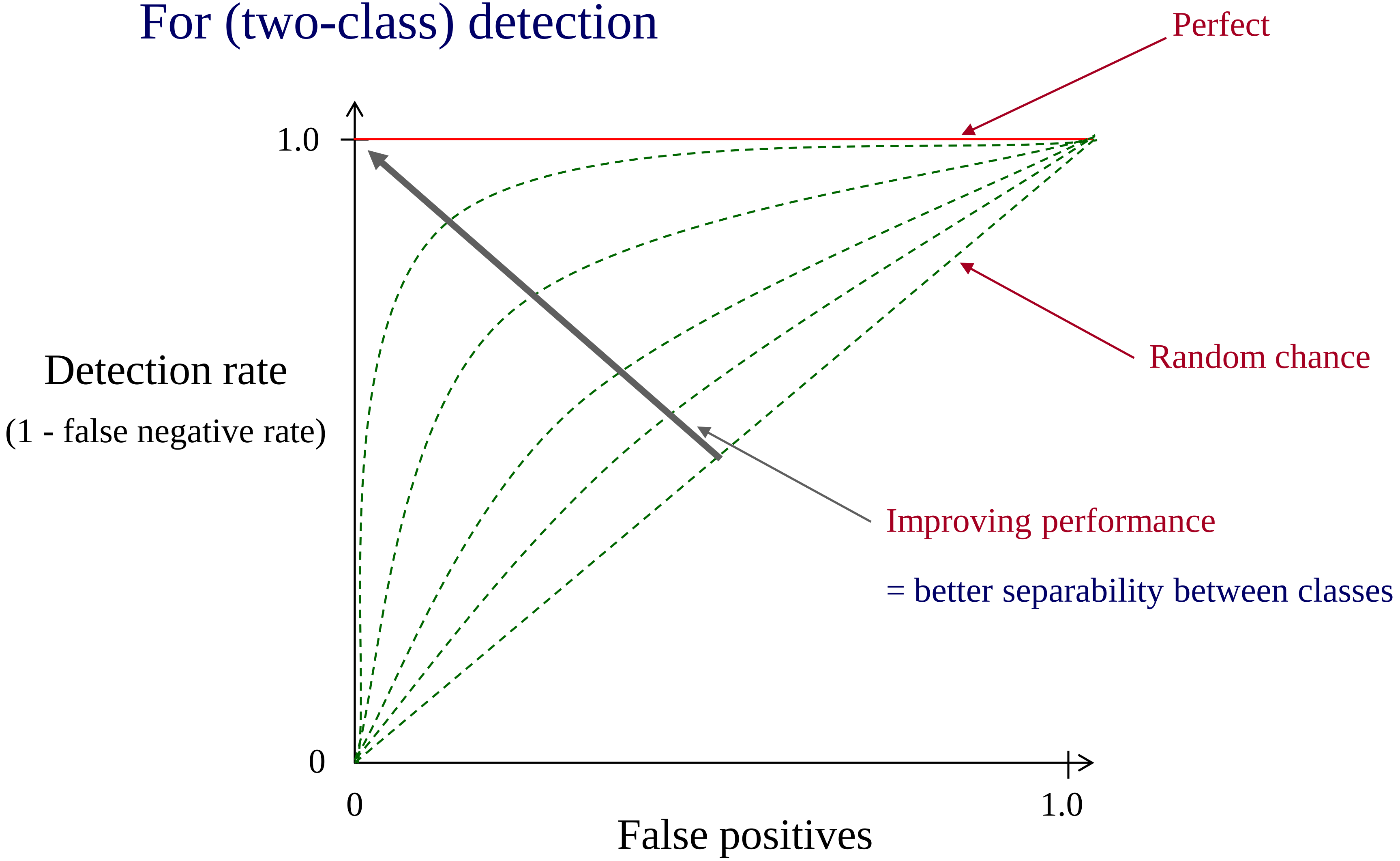
The *training set* (known examples) should be representative of the *testing set* (real data).

Good performance on your training set alone is meaningless...!

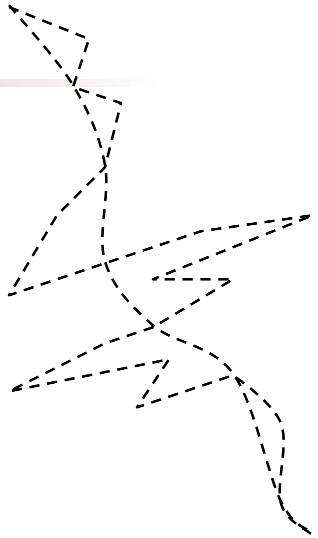
In every experiment, keep the *training* and *testing* data separate.

Evaluating performance – the ROC curve

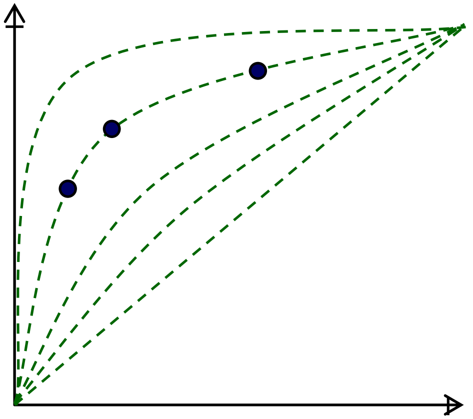
For (two-class) detection



Improving classification/detection

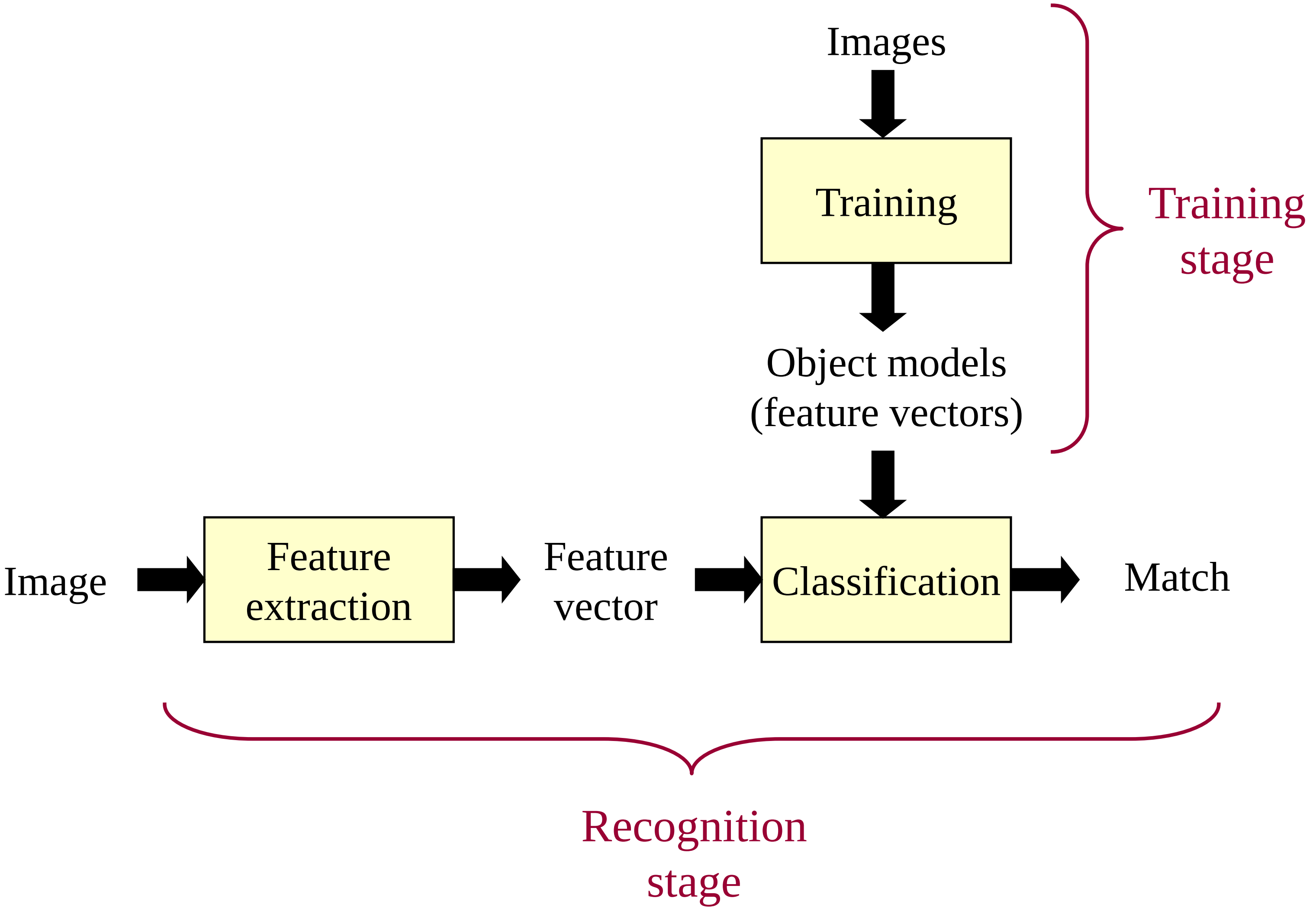


- More training samples
  - So classification strategy is more general
- Don’t “overtrain”
  - Don’t want to “learn the noise” – keep it simple
- Use better features
  - Good features lead to good class separation
- Don’t confuse movement *along* the ROC curve with *improving* the ROC curve



Extra

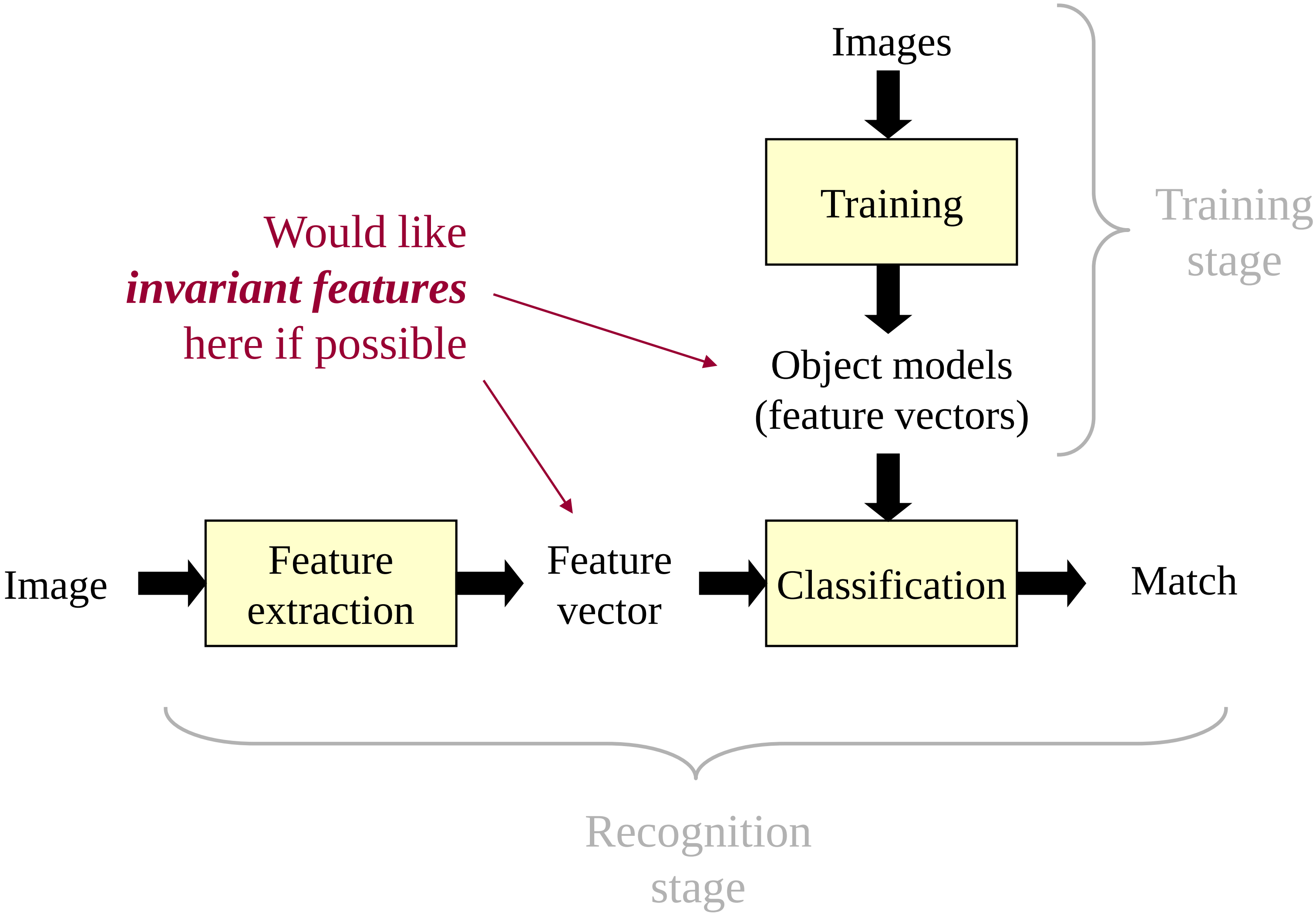
General approach to recognition



Invariant features for recognition

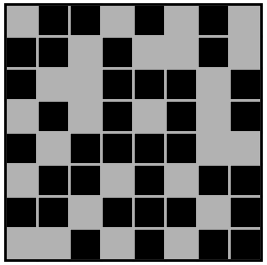
- An ***invariant*** feature is one that does not change under a certain class of transformations
  - Lengths and angles are invariant under rigid motion
  - Normalized correlation is invariant under scaling of image intensities
  - Brightness/color of a Lambertian surface is invariant under rotation
  - Length and angles *in the image* are **not** invariant under out-of-plane rotation and translation
  - Etc....
- Invariants can be geometric (location, shape) or radiometric (image values)
  - Geometric invariants tend to be much more common and useful

General approach to recognition



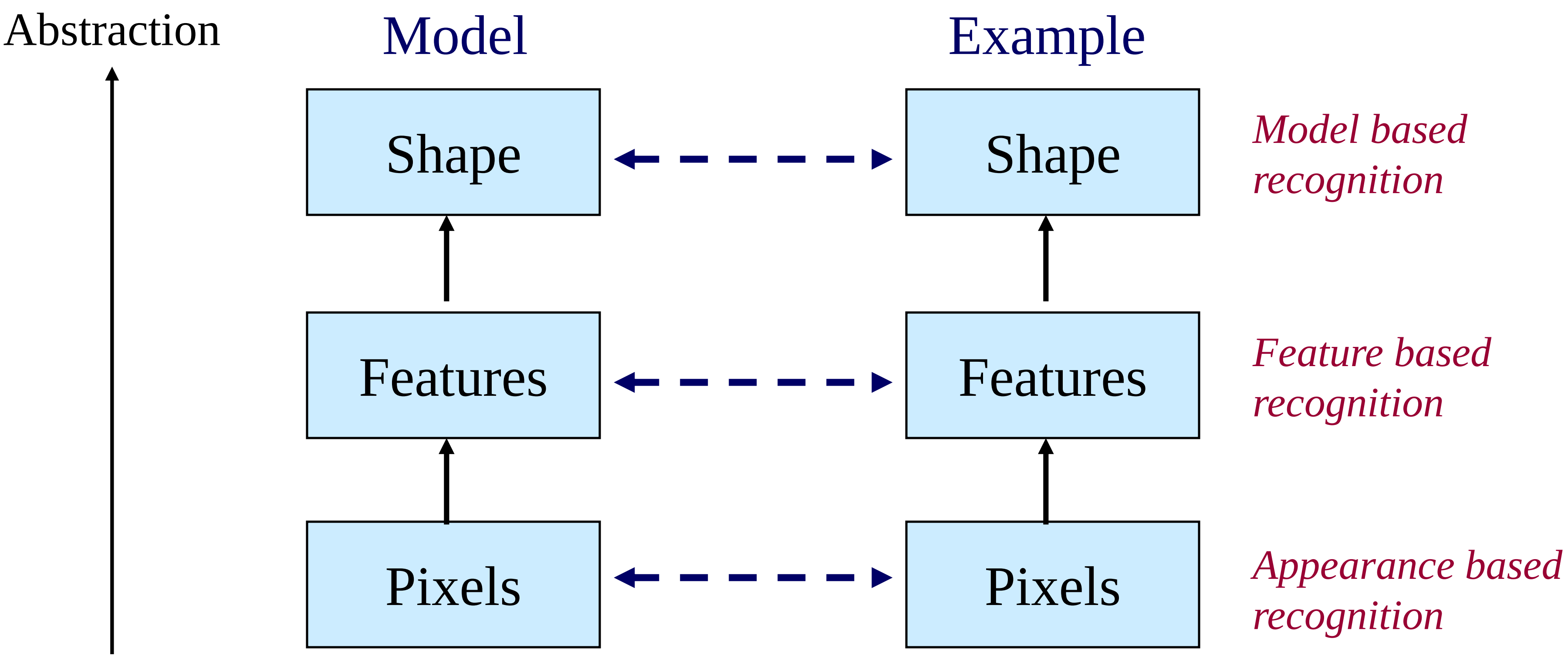
Why not this for face recognition/detection?

1. Ahead of time, search over all possible images to see which ones look like my face, and save these
2. During recognition, see if the input image is one of these



- Image space is vastly large
  - 8x8 binary image $\rightarrow$ 2^{64} image points (distinct images)
  - 1 billion images per second $\rightarrow$ **600 years**
- Step #1 would never finish!
- Not to mention, we'd have to do this for every possible view of my face
  - Range of facial expressions, lighting conditions, poses, etc.

Levels of Recognition/Matching



Reminder: Why computer vision is so hard!

These are all images of Simon's face!

In general, object recognition is difficult because of the immense variability of object appearance. With faces, this is even worse!

How can we reliably detect/recognize Simon???

Scanning windows…

Detection via classification: Main idea

Basic component: a binary classifier

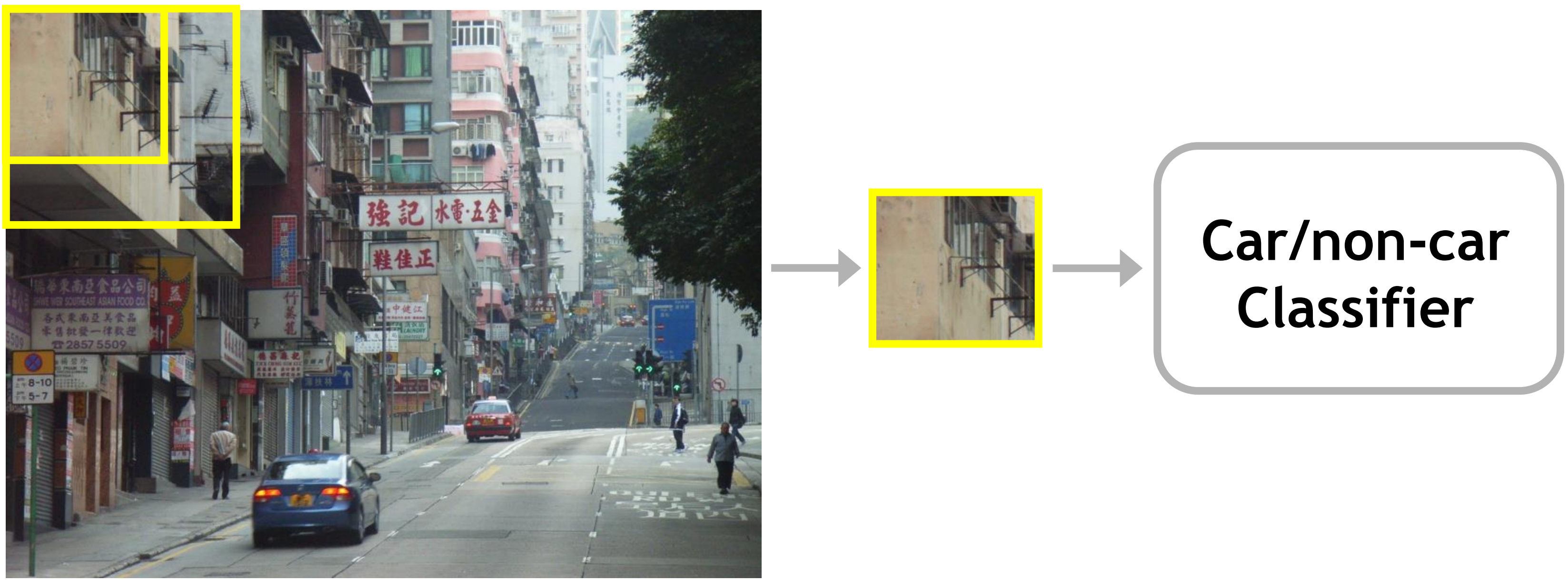
Car/non-car
Classifier

No, not a car.

Yes, car.

Detection via classification: Main idea

If object may be in a cluttered scene, slide a window around looking for it.



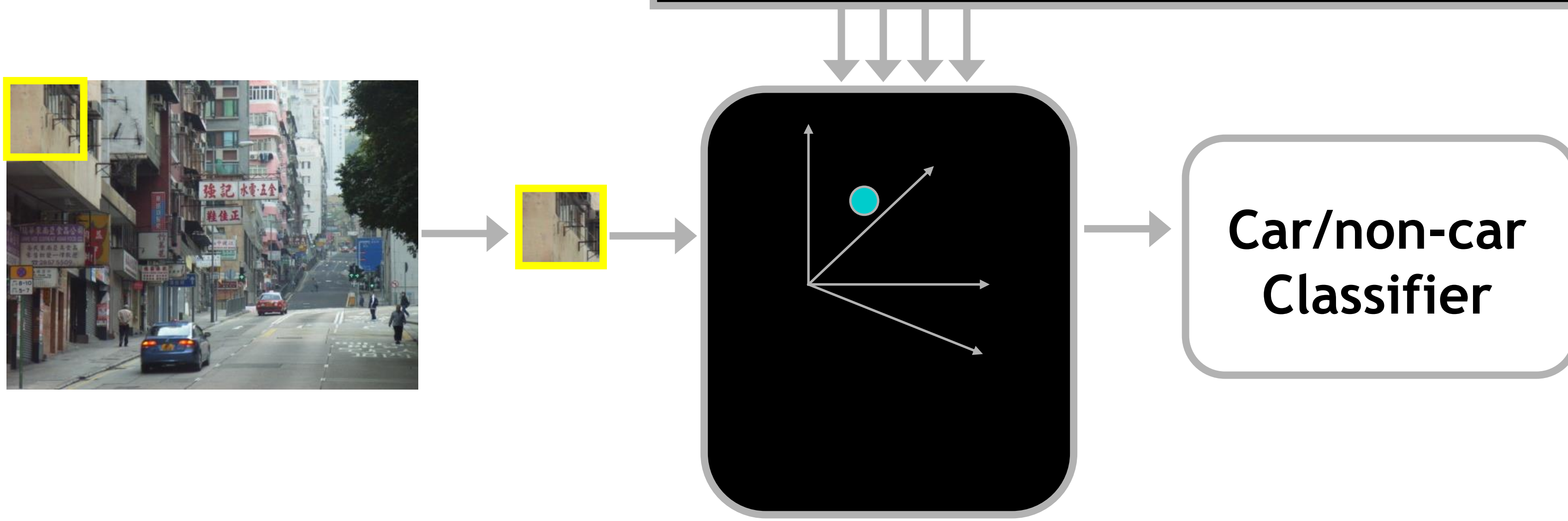
(Essentially, our skin detector was doing this, with a window that was one pixel big.)

K. Grauman, B. Leibe

Detection via classification: Main idea

Fleshing out this pipeline a bit more, we need to:

1. **Obtain training data**
2. **Define features**
3. **Define classifier**

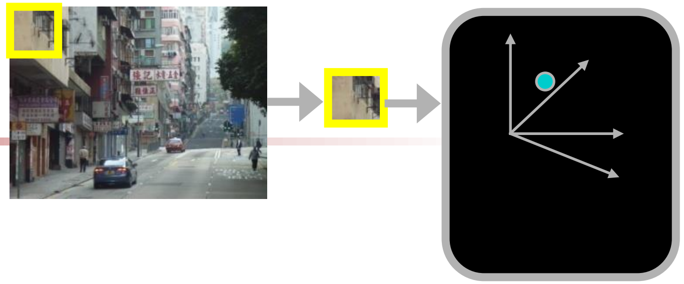


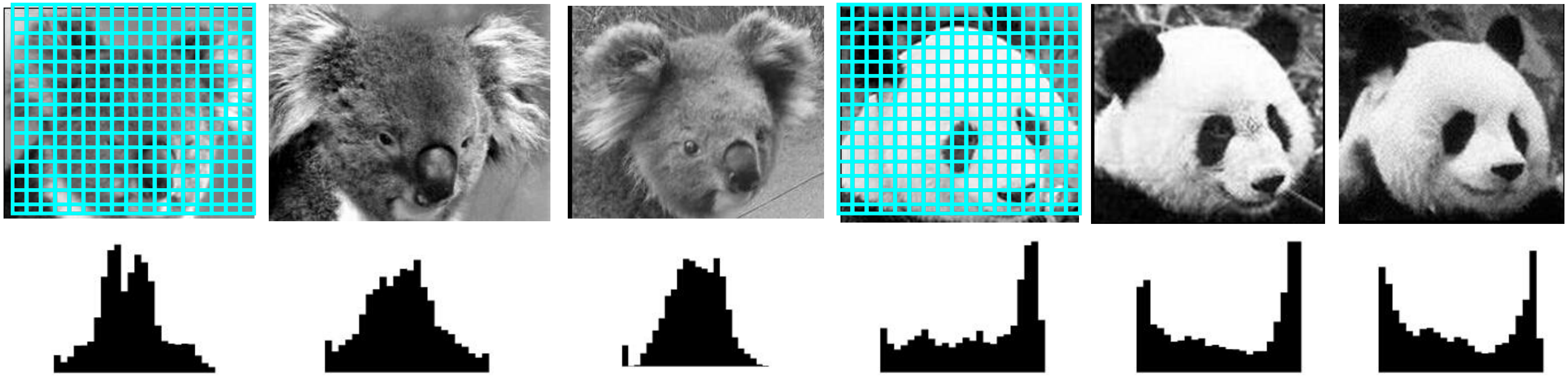
Detection via classification: Main idea

- Consider all subwindows in an image
  - Sample at multiple scales and positions (and orientations)

- Make a decision per window:
  - "Does this contain object category X or not?"

Feature extraction: global appearance



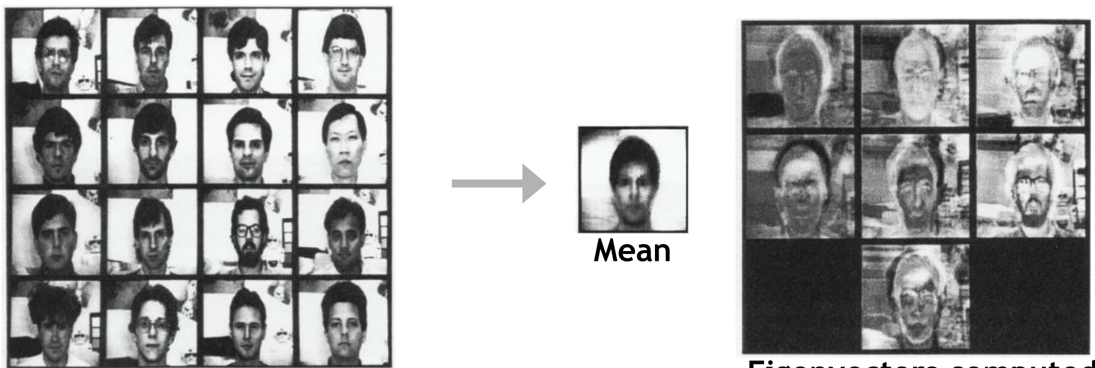


Simple holistic descriptions of image content

- **grayscale / color histogram**
- **vector of pixel intensities**

Eigenfaces: global appearance description

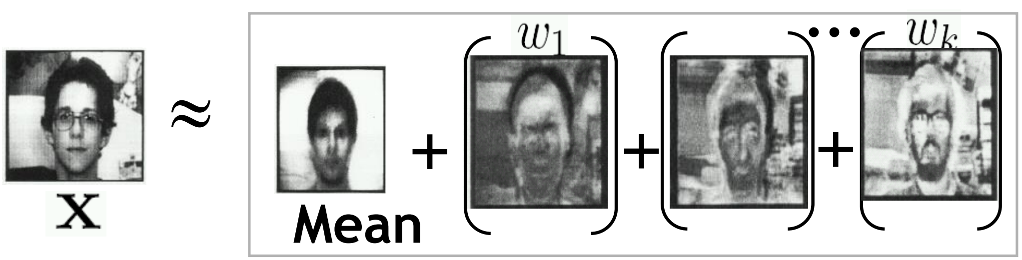
An early appearance-based approach to face recognition



Training images

Eigenvectors computed from covariance matrix

Generate low-dimensional representation of appearance with a linear subspace.



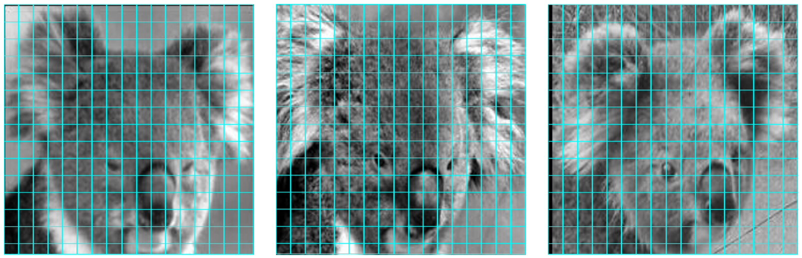
Project new images to "face space".

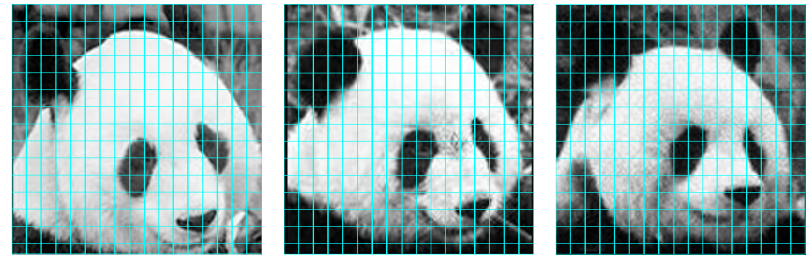
Recognition via nearest neighbors in face space

Turk & Pentland, 1991

Feature extraction: global appearance

- Pixel-based representations sensitive to small shifts





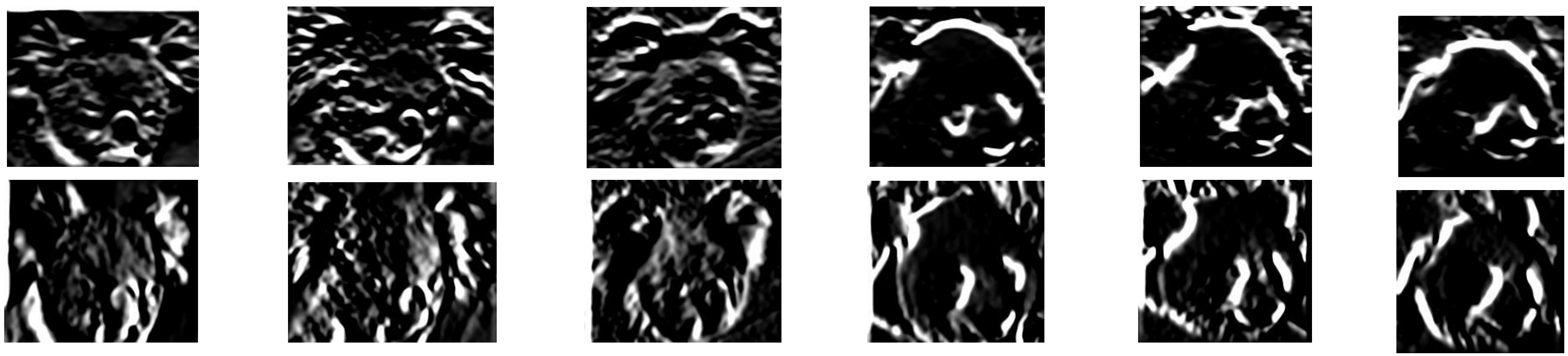
- Color or grayscale-based appearance description can be sensitive to illumination and intra-class appearance variation

Cartoon example: an albino koala

K. Grauman, B. Leibe

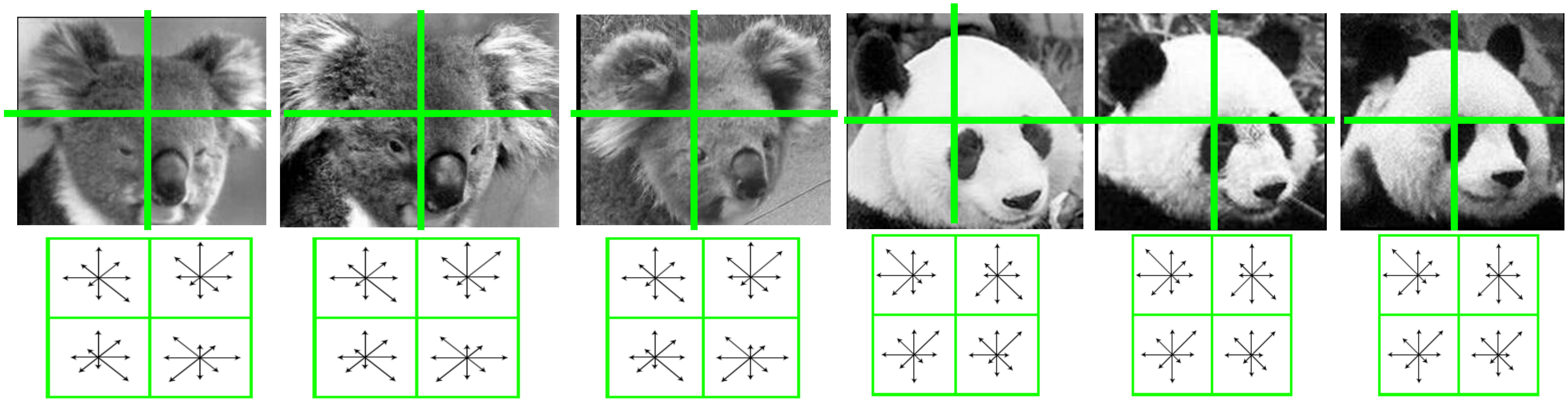
Gradient-based representations

- Consider edges, contours, and (oriented) intensity gradients



Gradient-based representations

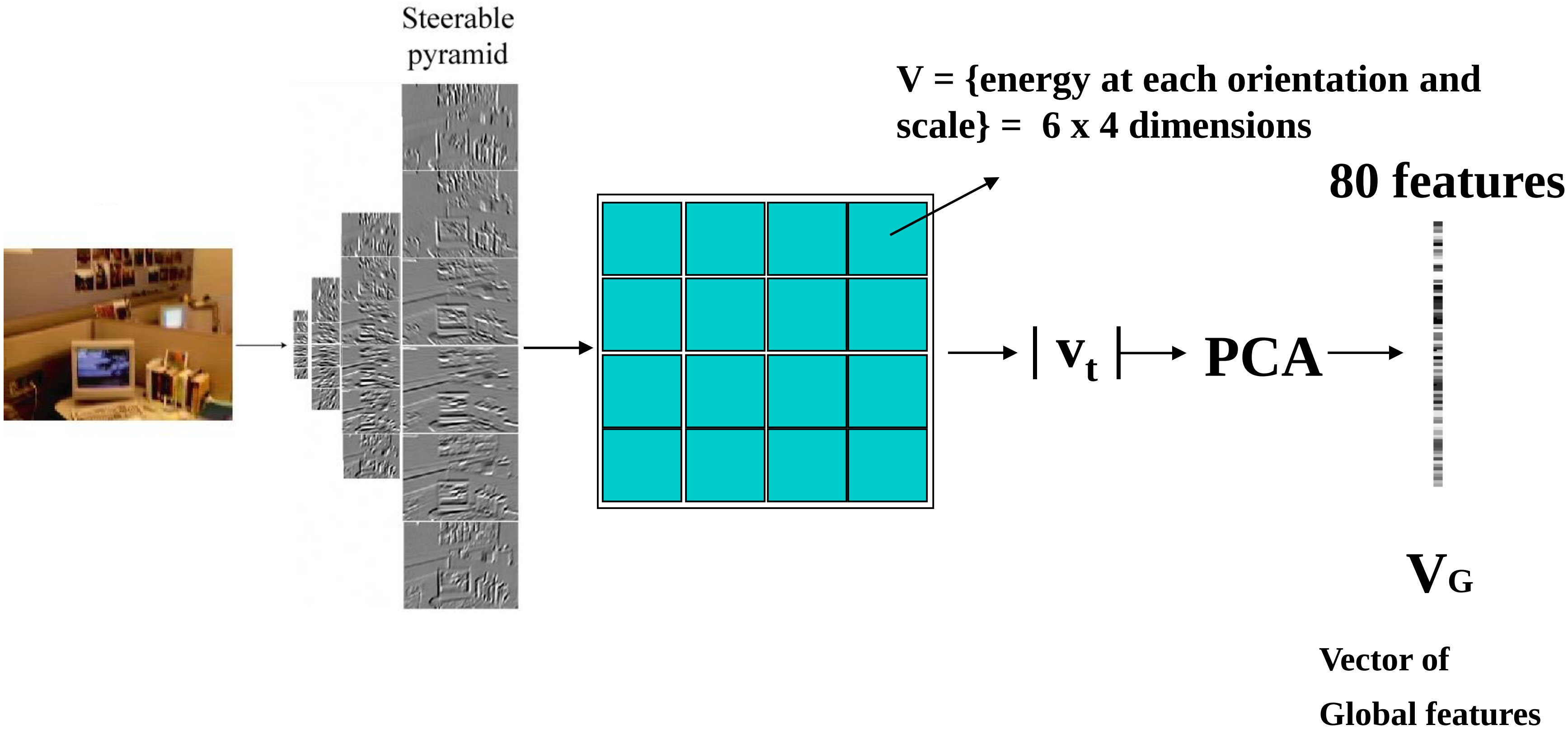
- Consider edges, contours, and (oriented) intensity gradients



- Summarize local distribution of gradients with histogram
  - Locally orderless: offers invariance to small shifts and rotations
  - Contrast-normalization: try to correct for variable illumination

GIST

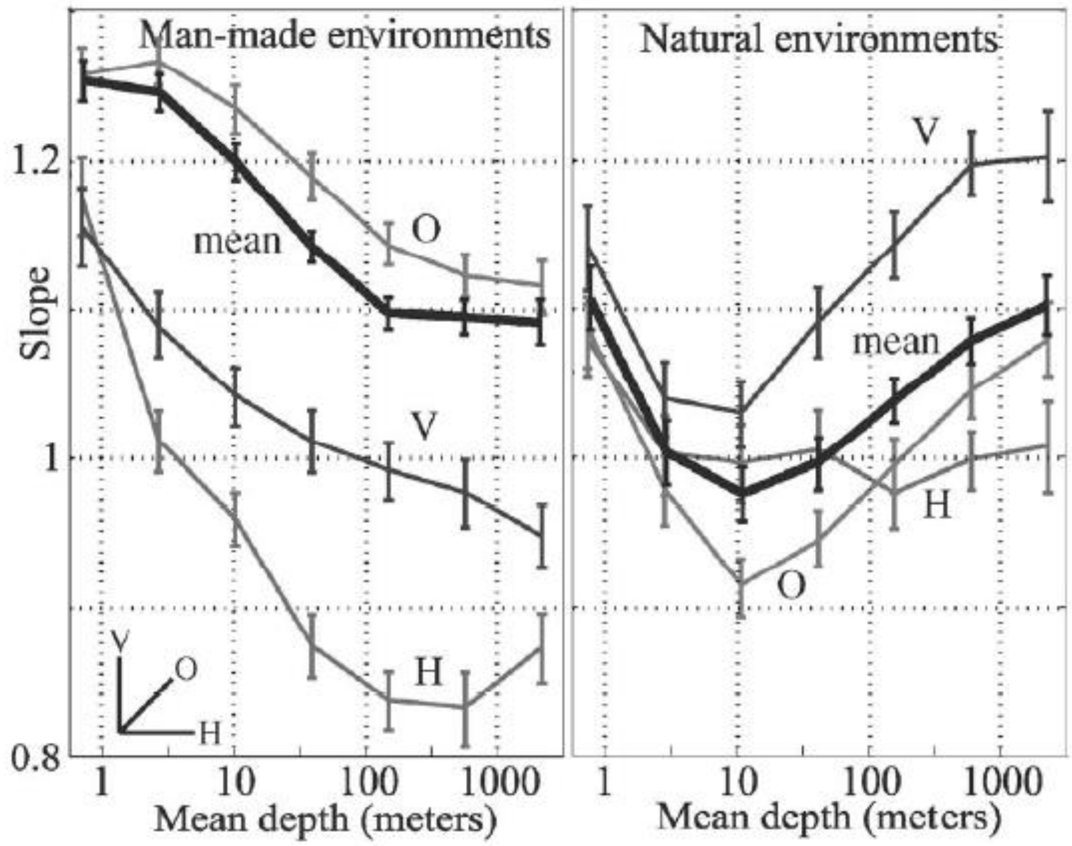
Representing Image Structure with "GIST"



Oliva & Torralba (2001,2002, 2006)

Slide Credit: Olivia

What do Images Statistics say about Depth?



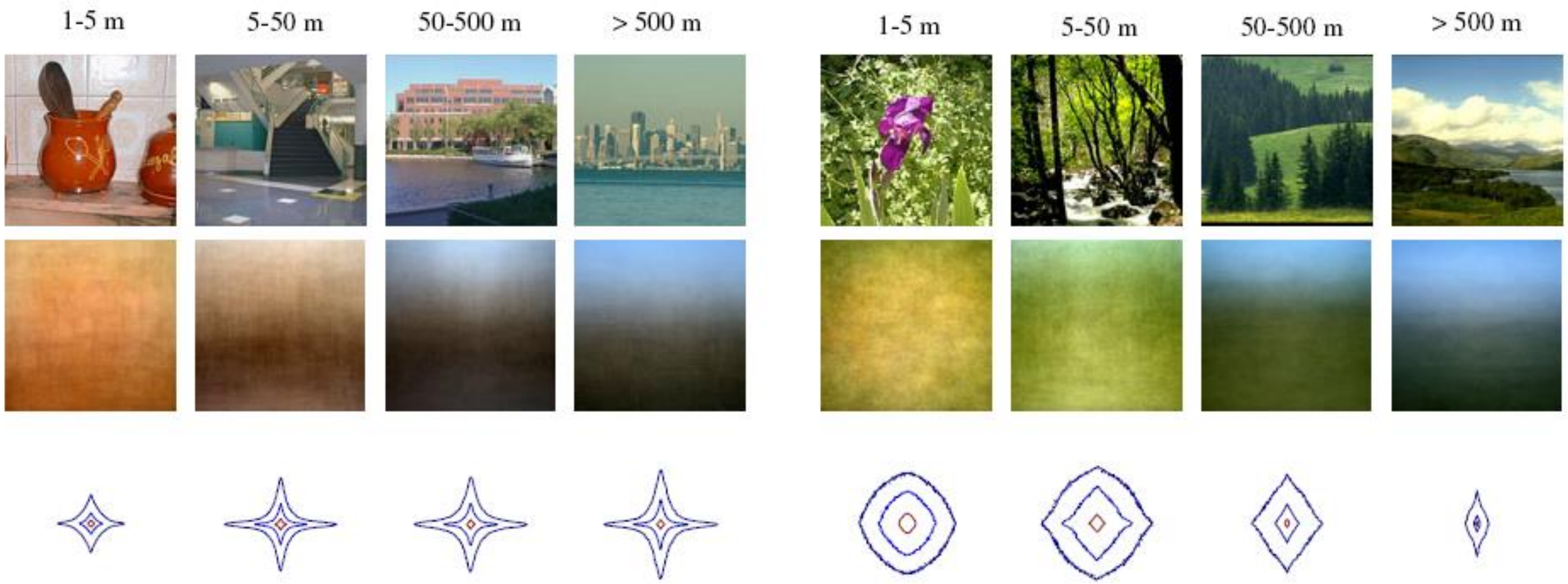
V: Vertical

H: Horizontal

O: Oblique

Slide Credit: Torralba, Olivia, J. Huang

Scene Scale



- "The point of view that any given observer adopts on a specific scene is constrained by the volume of the scene."
- How does the amount of clutter vary against scene scale in man-made environments? In natural environments?

Slide Credit: Torralba, Olivia, J. Huang

Categorization of Natural Scenes

Confusion Matrix (in % using Layout template) :

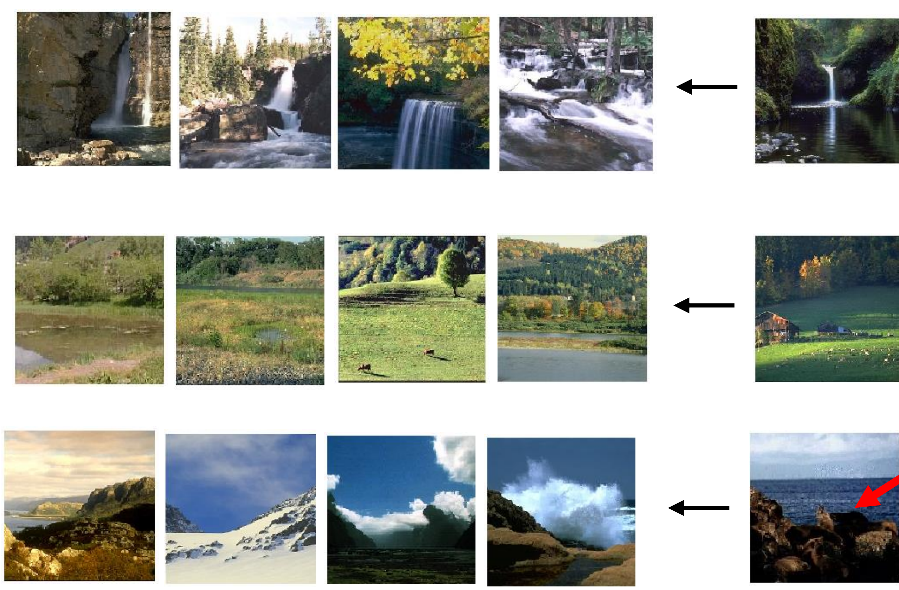
Classification of prototypical scenes (400 / category)

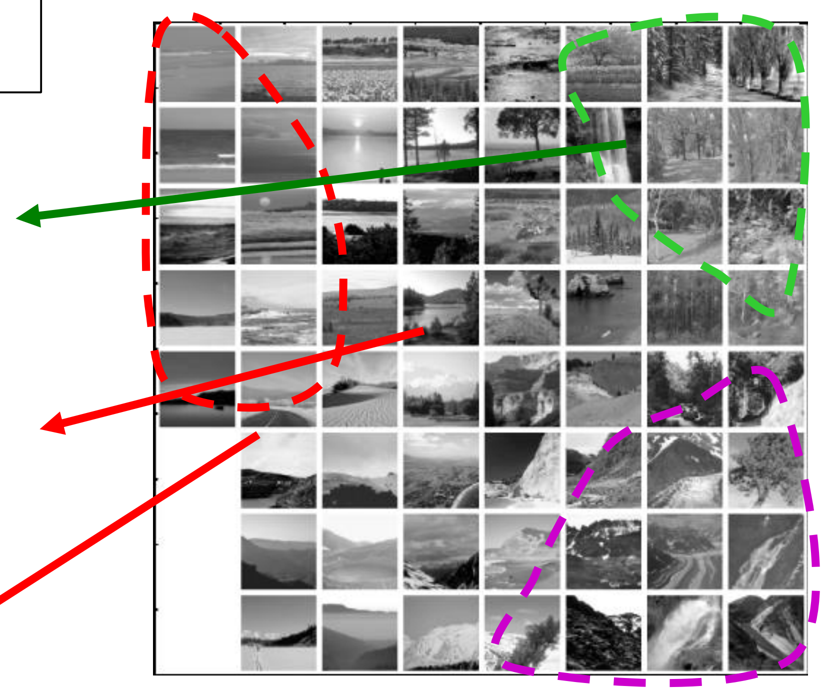
	Coast	Countryside	Forest	Mountain
Coast	88.6	8.9	1.2	1.3
Countryside	9.8	85.2	3.7	1.3
Forest	0.4	3.6	91.5	4.5
Mountain	0.4	4.6	3.8	91.2

Local organization:

correct for 92 % images

(4 similar images on 7 K-NN)

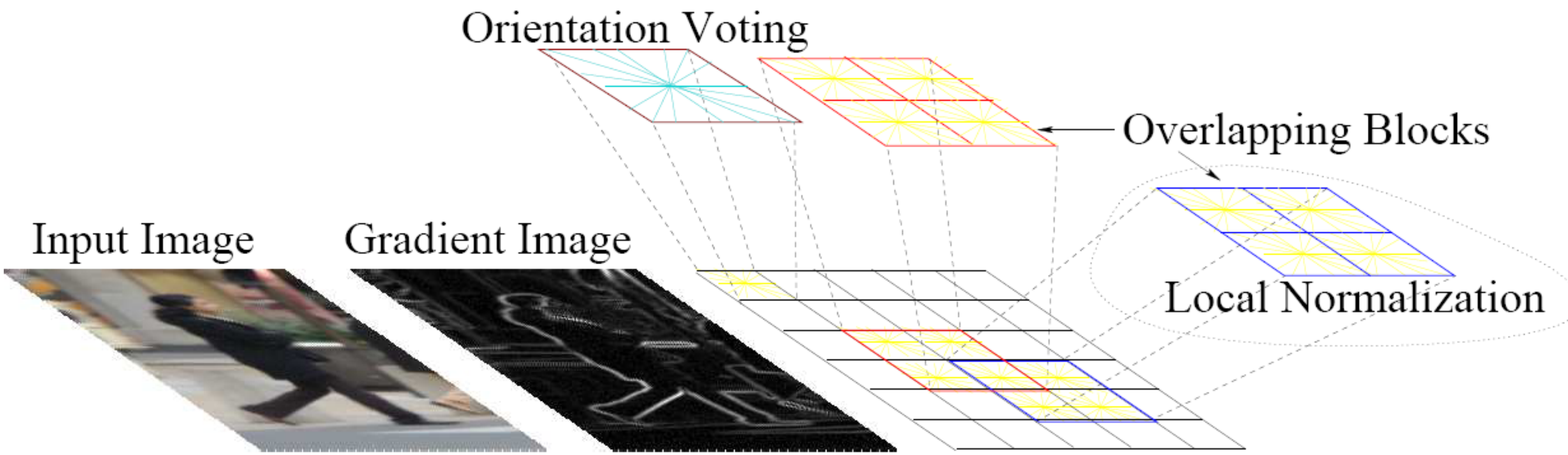




Slide Credit: Olivia

HOG

Gradient-based representations: Histograms of oriented gradients (HoG)



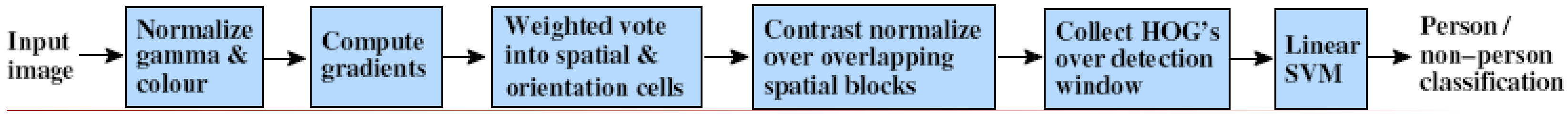
Map each grid cell in the input window to a histogram counting the gradients per orientation.

Code available:
http://pascal.inrialpes.fr/soft/olt/

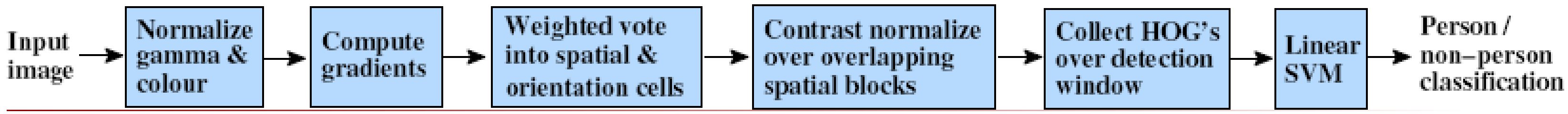
Dalal & Triggs, CVPR 2005

K. Grauman, B. Leibe

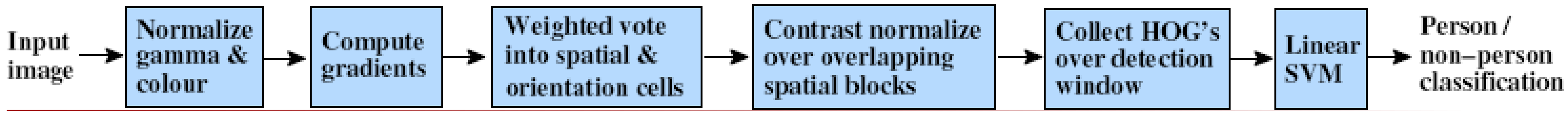
Slide credit: Dalal, Triggs, P. Barnum



Slide credit: Dalal, Triggs, P. Barnum



- Tested with
  - RGB
  - LAB
  - Grayscale
- Gamma Normalization and Compression
  - Square root
  - Log



-1	0	1

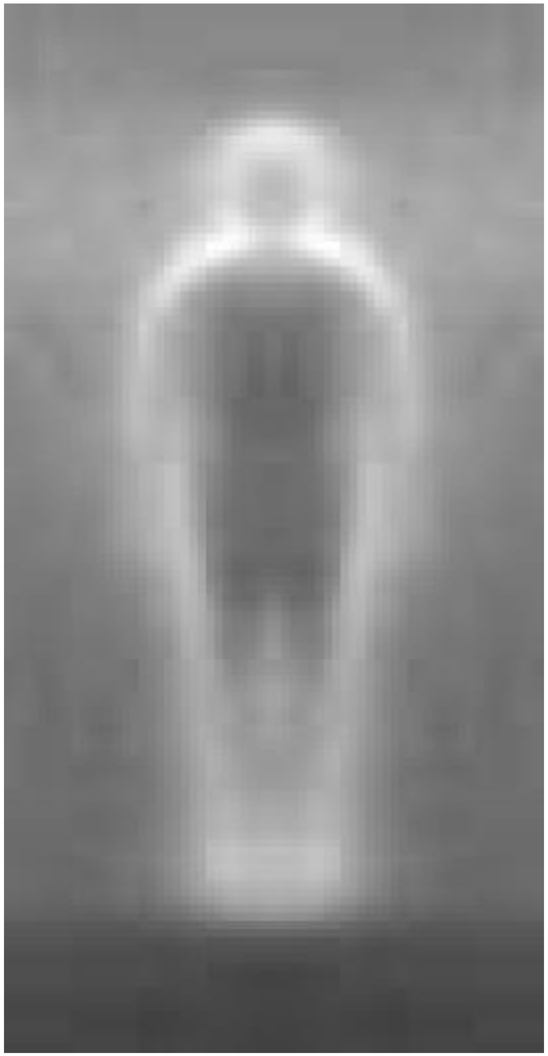
centered

-1	1

uncentered

1	-8	0	8	-1

cubic-corrected



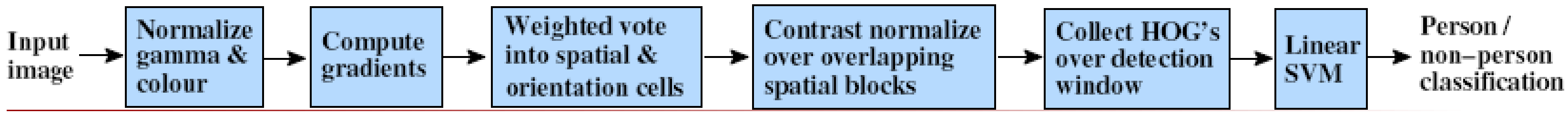
0	1
-1	0

diagonal

-1	0	1
-2	0	2
-1	0	1

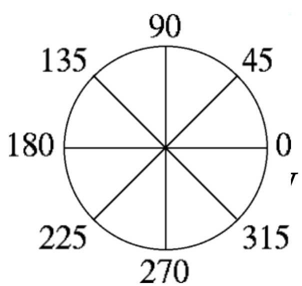
Sobel

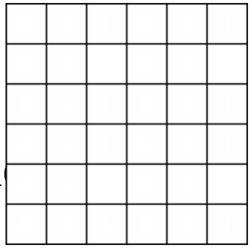
Slide credit: Dalal, Triggs, P. Barnum

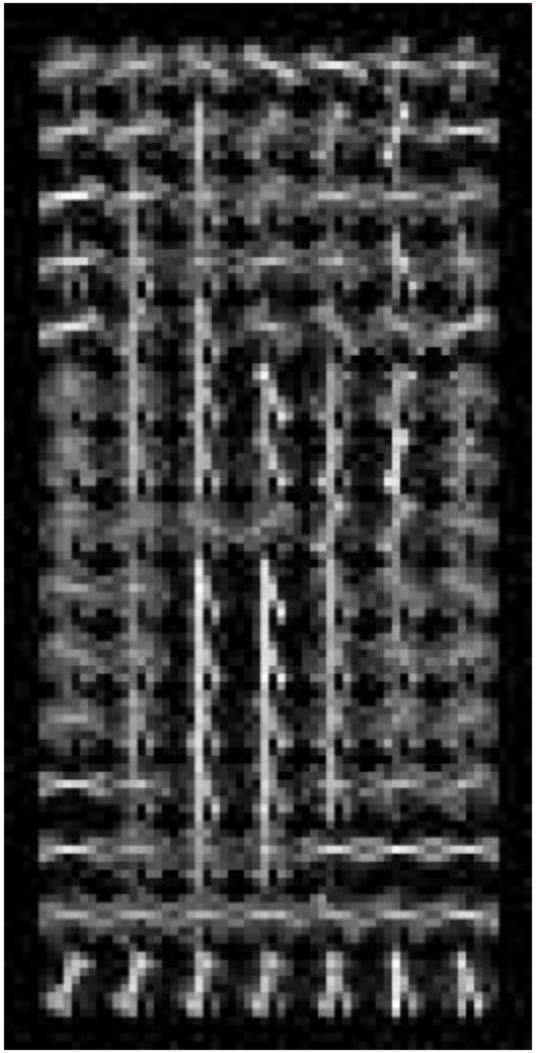


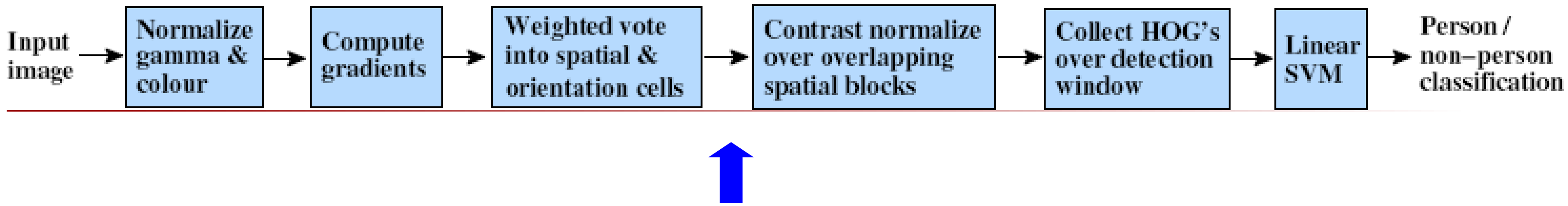
- Histogram of gradient orientations
  - -Orientation -Position

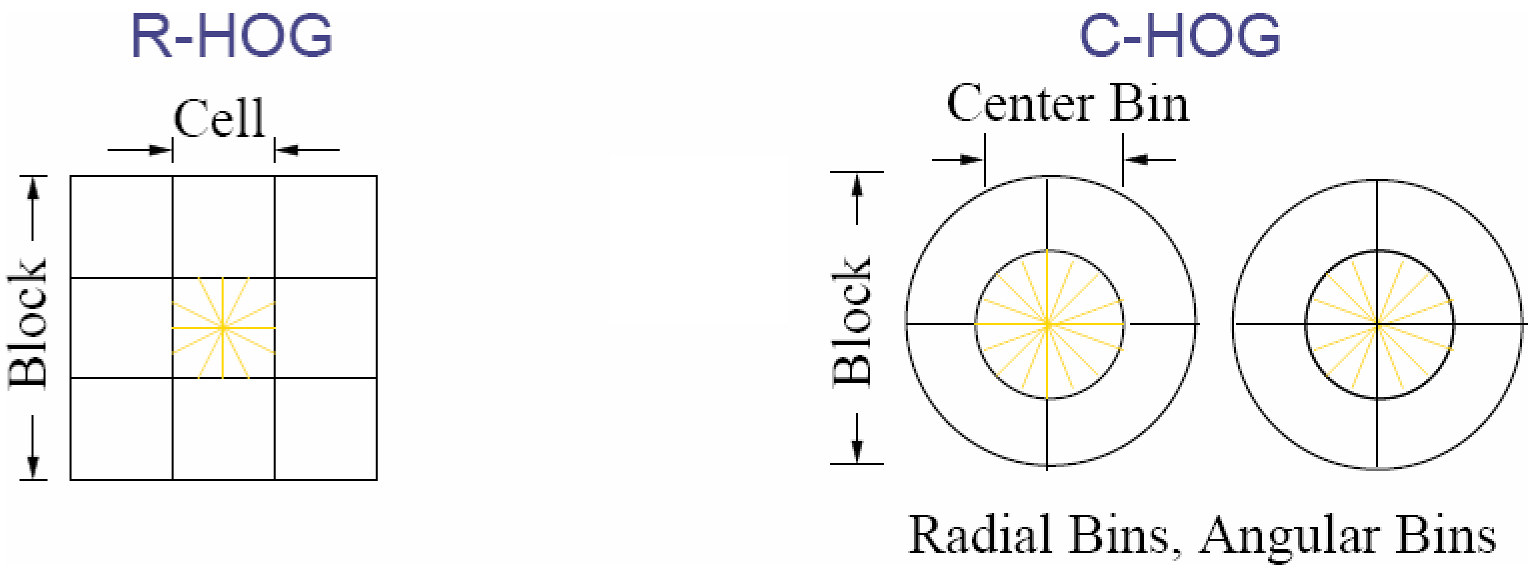
– magnitu



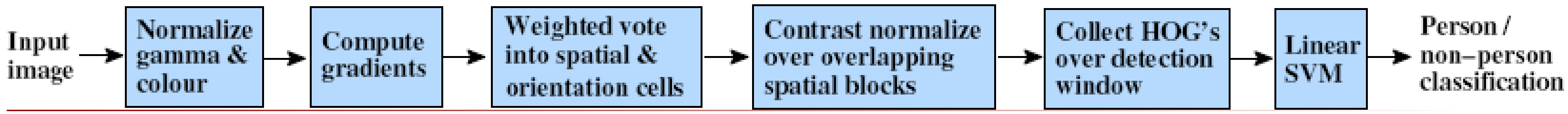




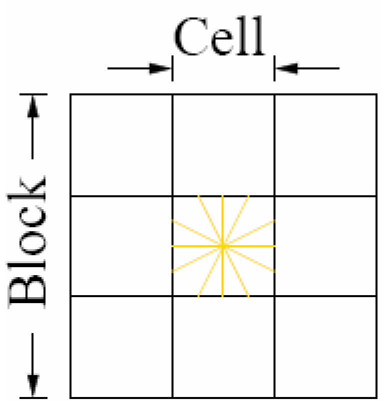




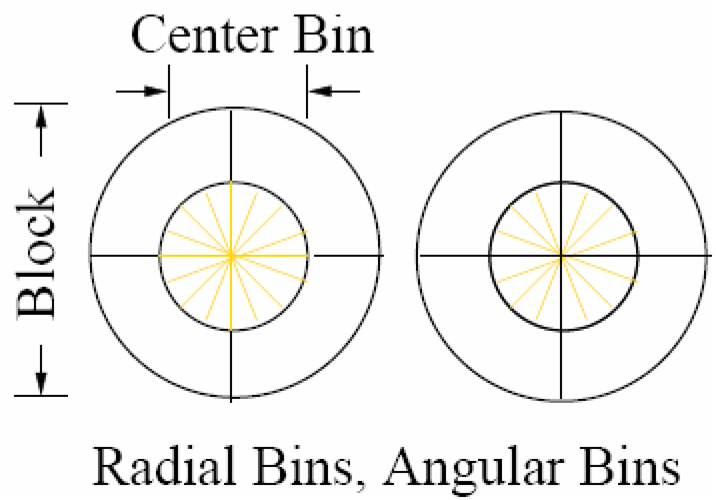
Slide credit: Dalal, Triggs, P. Barnum



R-HOG



C-HOG

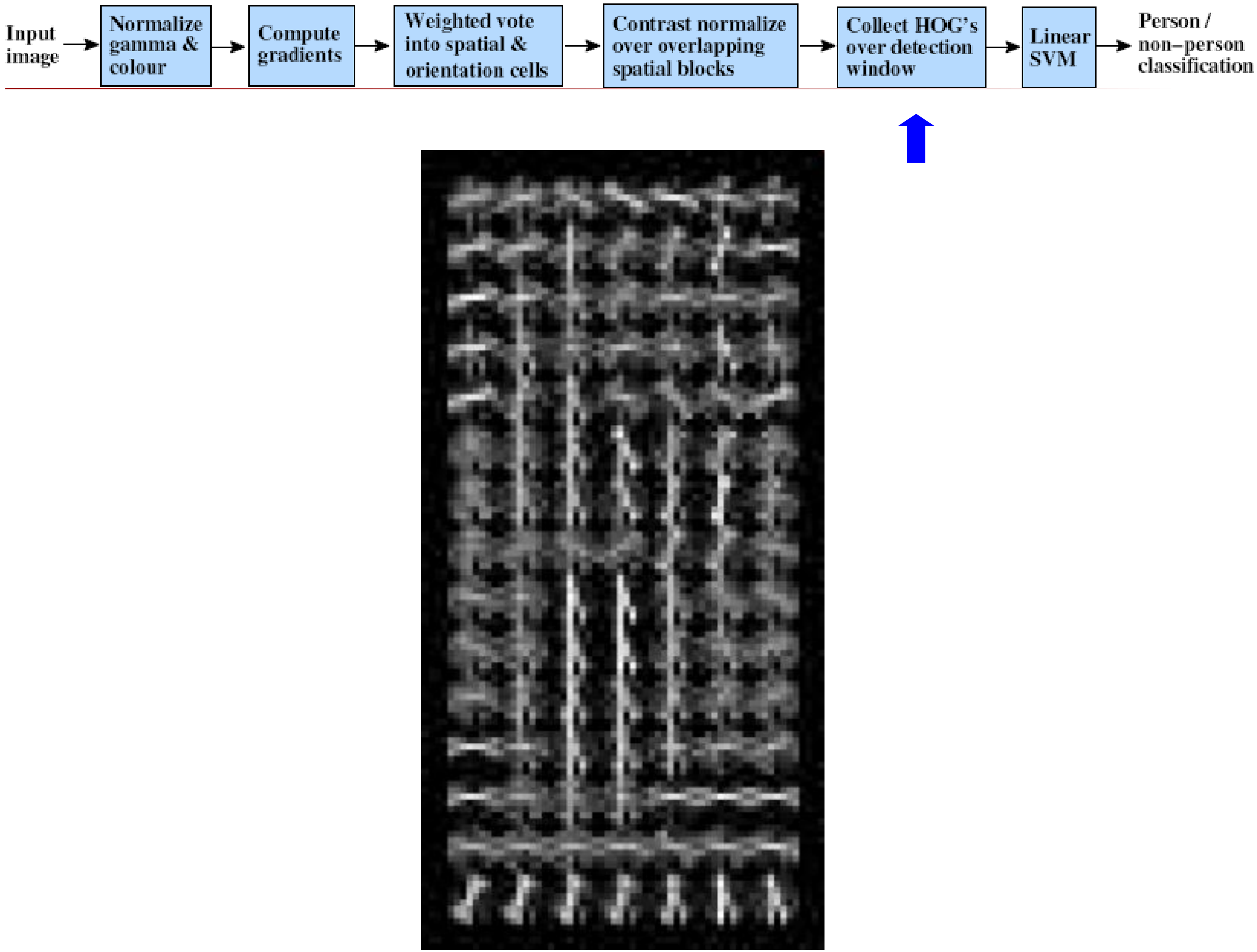


$$L1-norm : v \longrightarrow v/(||v||_1 + \epsilon)$$

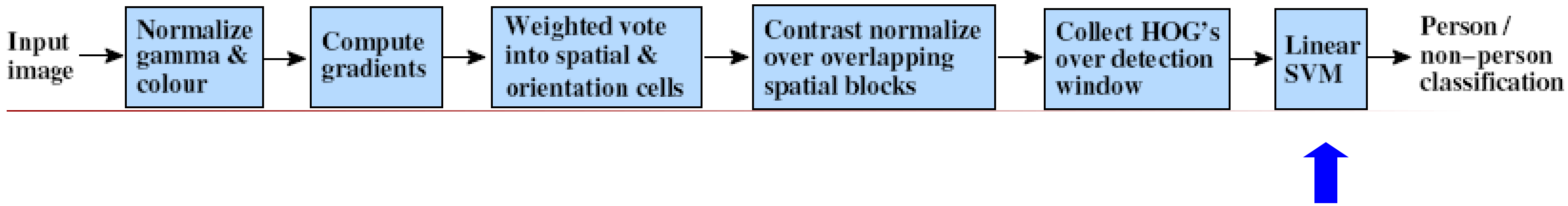
$$L1-sqrt : v \longrightarrow \sqrt{v/(||v||_1 + \epsilon)}$$

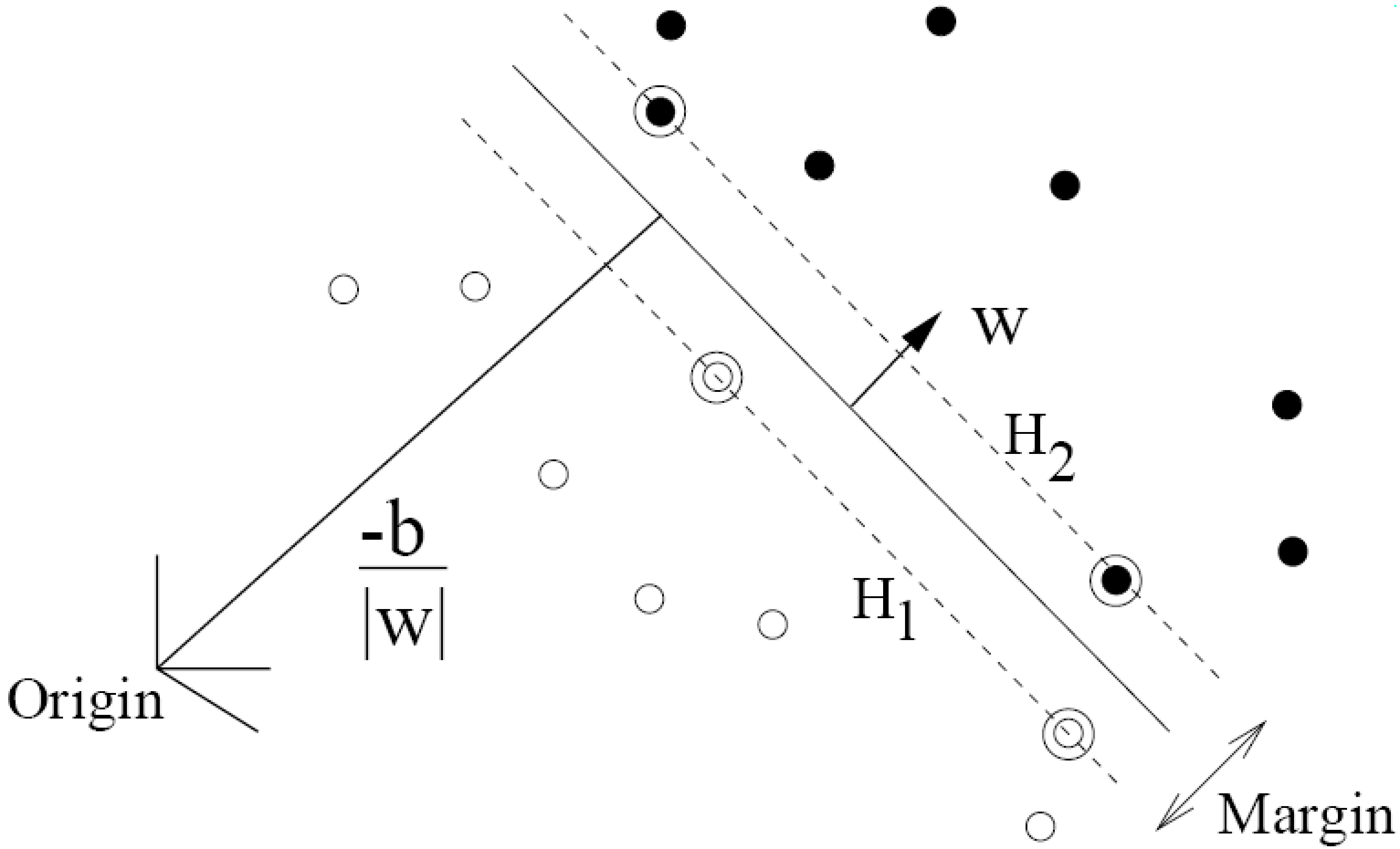
$$L2-norm : v \longrightarrow v/\sqrt{||v||_2^2 + \epsilon^2}$$

$L2-hys$: L2-norm, plus clipping at .2 and renomalizing

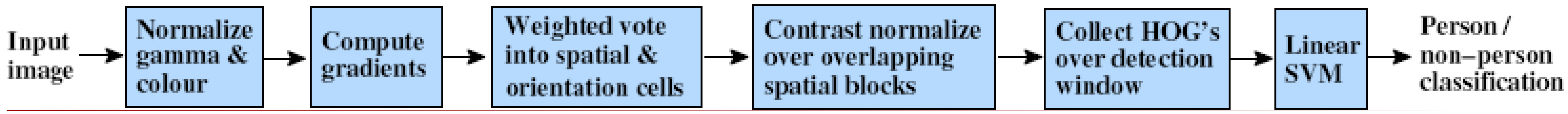


Slide credit: Dalal, Triggs, P. Barnum





Slide credit: Dalal, Triggs, P. Barnum

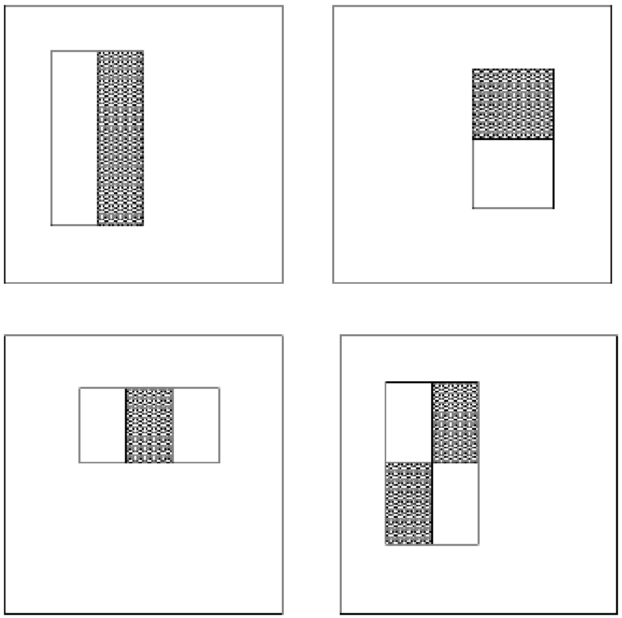


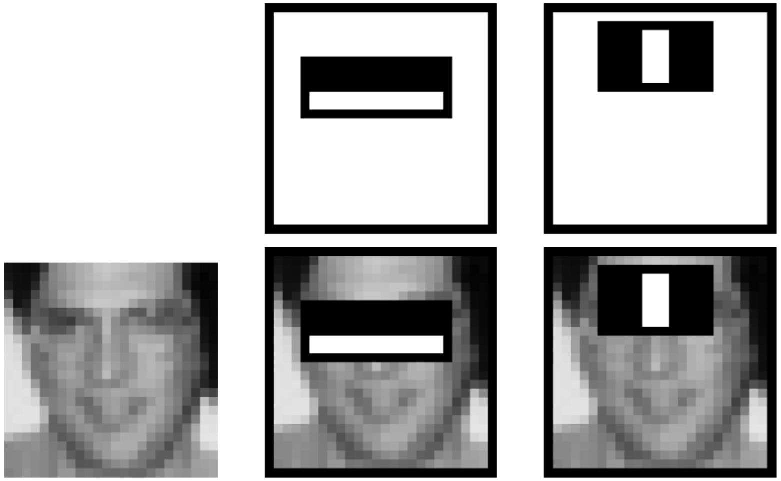
Slide credit: Dalal, Triggs, P. Barnum

Slide credit: Dalal, Triggs, P. Barnum

Boosted Face Detection with Gradient Features

Gradient-based representations: Rectangular features





Compute differences between sums of pixels in rectangles

Captures contrast in adjacent spatial regions, efficient to compute

Each feature parameterized by scale, position, type.

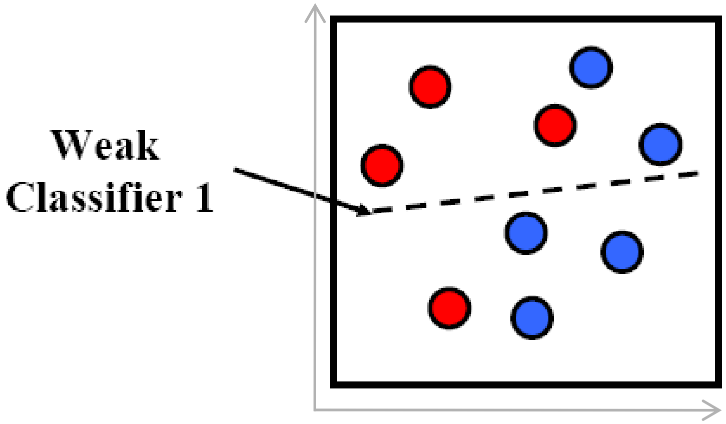
Viola & Jones, CVPR 2001

Boosting

- Build a strong classifier by combining number of "weak classifiers", which need only be better than chance
- Sequential learning process: at each iteration, add a weak classifier
- Flexible to choice of weak learner
  - including fast simple classifiers that alone may be inaccurate
- We'll look at Freund & Schapire's AdaBoost algorithm
  - Easy to implement
  - Base learning algorithm for Viola-Jones face detector

K. Grauman, B. Leibe

AdaBoost: Intuition



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

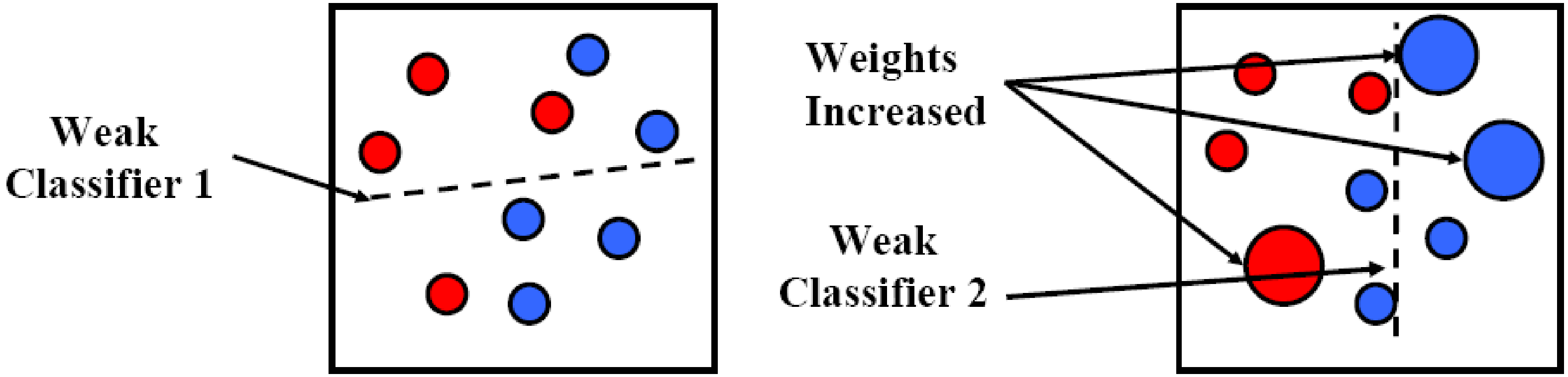
Each weak classifier splits the training examples with at least 50% accuracy.

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

Figure adapted from Freund and Schapire

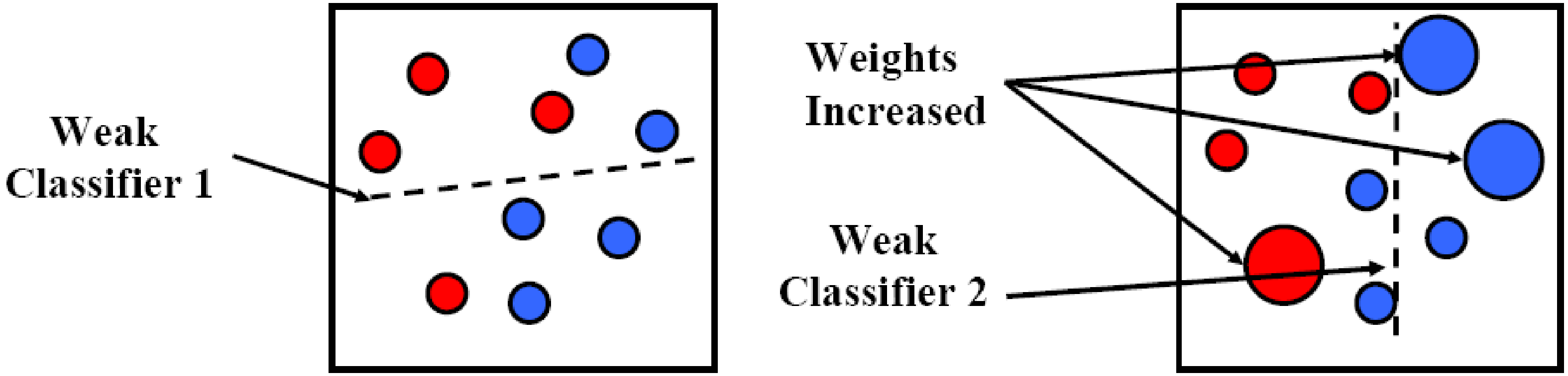
K. Grauman, B. Leibe

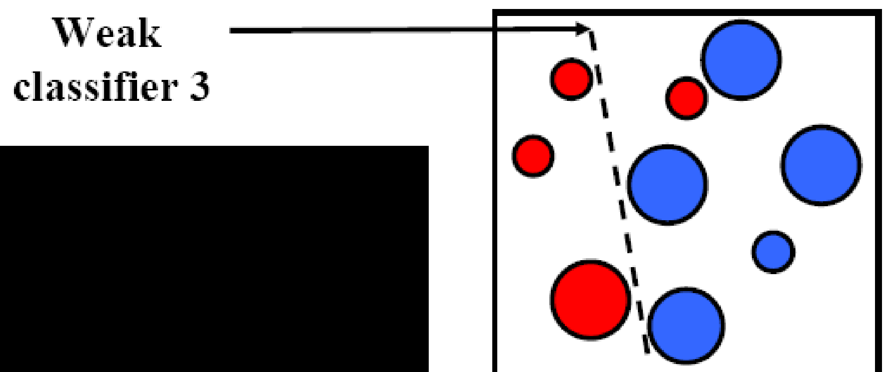
AdaBoost: Intuition



K. Grauman, B. Leibe

AdaBoost: Intuition





Final [redacted]
comb[redacted]
weak[redacted]

K. Grauman, B. Leibe

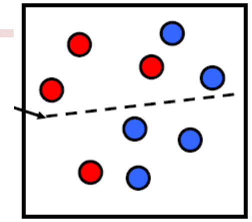
- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:

  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.

Freund & Schapire 199

Example: Face detection

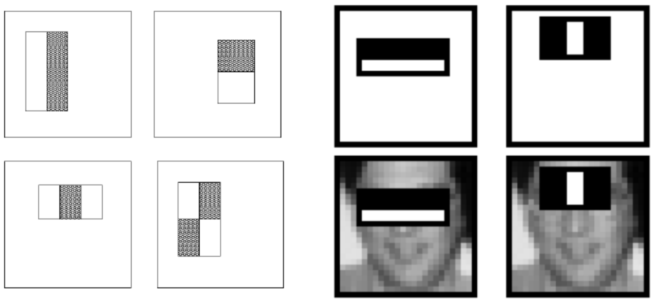
- Frontal faces are a good example of a class where global appearance models + a sliding window detection approach fit well:
  - Regular 2D structure
  - Center of face almost shaped like a “patch”/window

- Now we’ll take AdaBoost and see how the Viola-Jones face detector works

K. Grauman, B. Leibe

Feature extraction

“Rectangular” filters

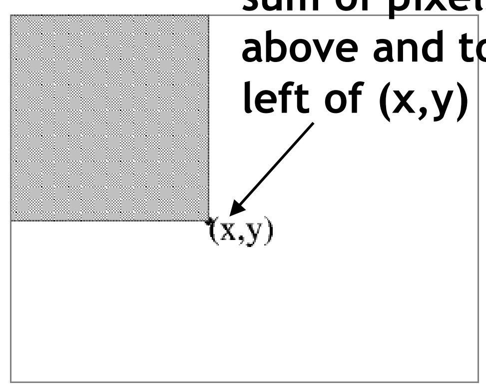


Feature output is difference between adjacent regions

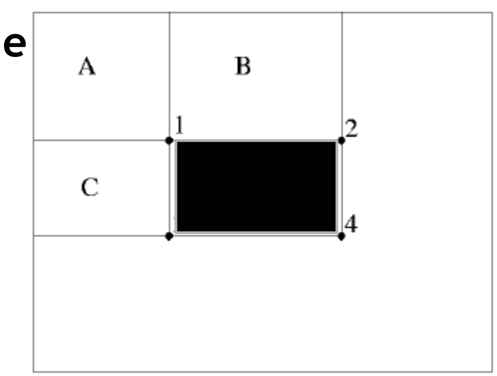
Efficiently computable with integral image: any sum can be computed in constant time

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

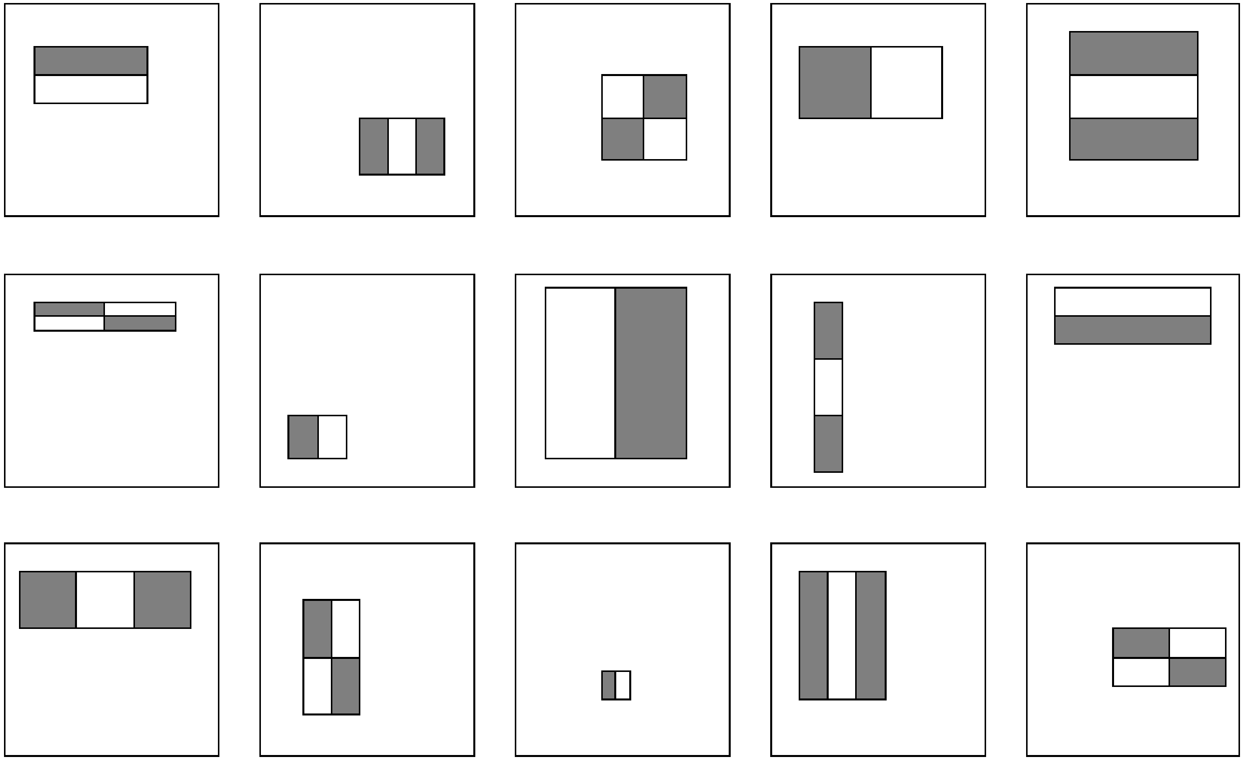


$$
\begin{aligned}
D &= 1 + 4 - (2 + 3) \\
&= A + (A + B + C + D) - (A + C + A + B) \\
&= D
\end{aligned}
$$

Viola & Jones, CVPR 2001

K. Grauman, B. Leibe

Large library of filters



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

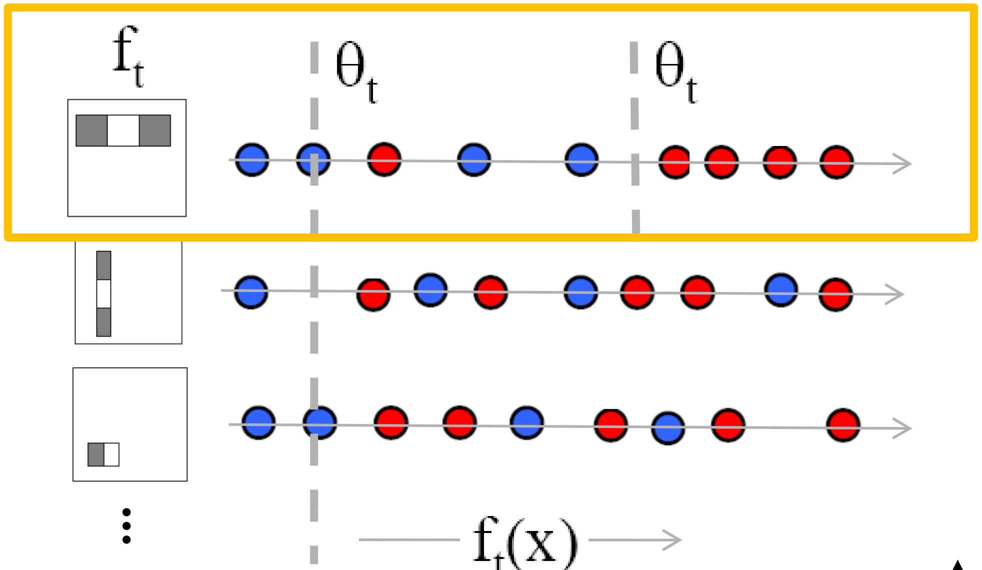
180,000+ possible features associated with each 24 x 24 window

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 (non-faces) training examples, in terms of *weighted* error.



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

Resulting weak classifier:

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

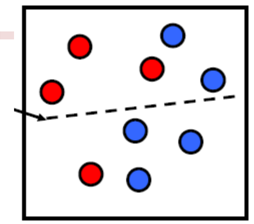
For next round, reweight the examples according to errors, choose another filter/threshold combo.

Viola & Jones, CVPR 2001

AdaBoost Algorithm

- 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 $\beta_t = \frac{\epsilon_t}{}$.
- 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



$\{x_1, \ldots x_n\}$

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.

Freund & Schapire 199

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

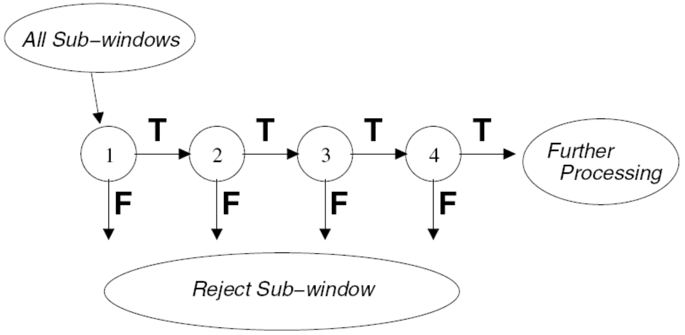
- Even if the filters are fast to compute, each new image has a lot of possible windows to search.

- How to make the detection more efficient?

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



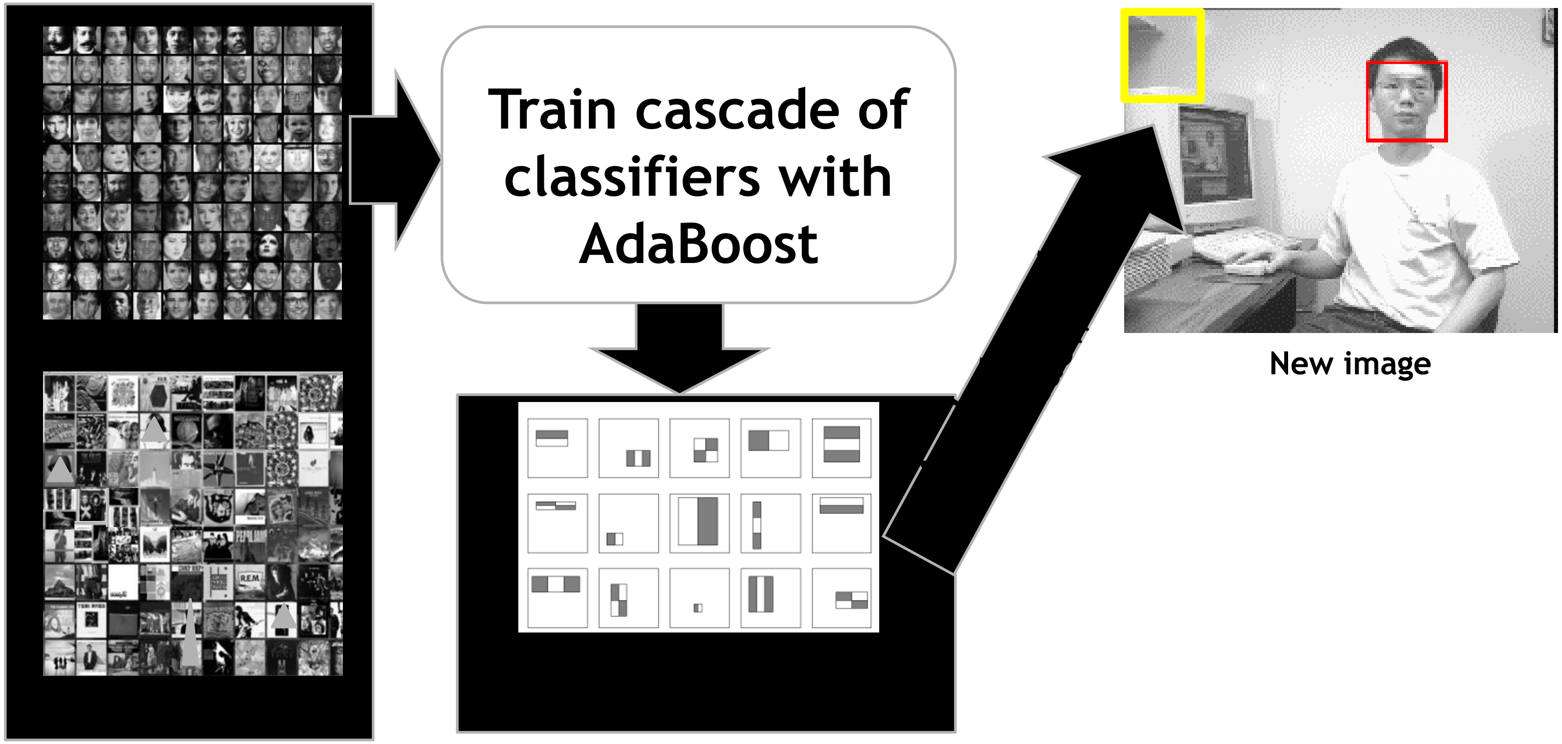
Fleuret & Geman, IJCV 2001

Rowley et al., PAMI 1998

Viola & Jones, CVPR 2001

Figure from Viola & Jones CVPR 2001

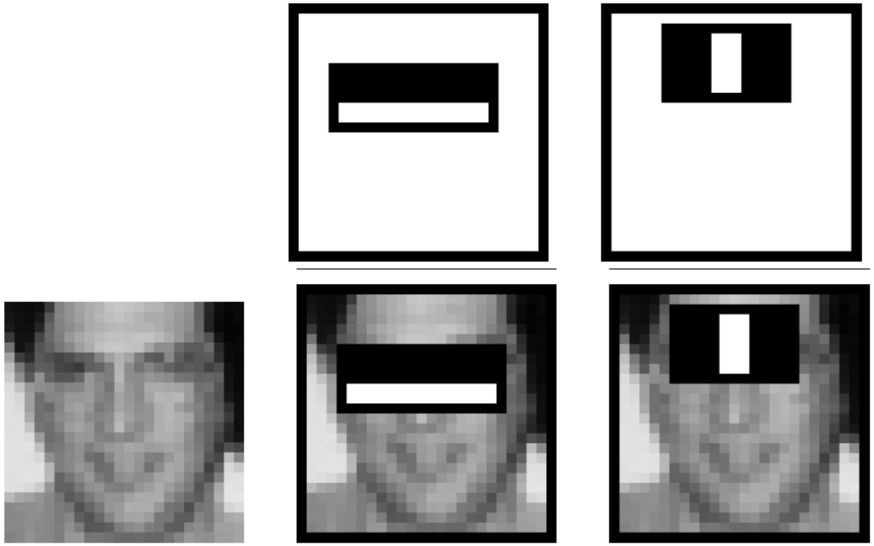
Viola-Jones Face Detector: Summary



- 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/]

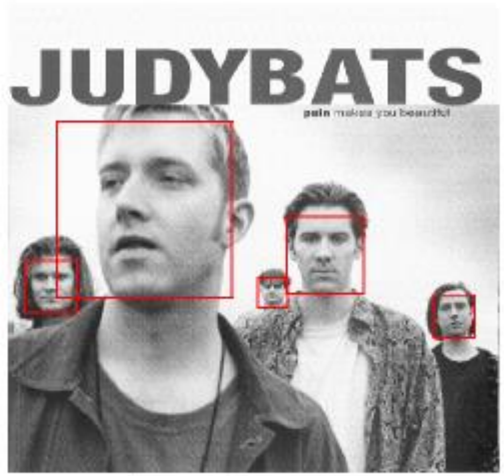
K. Grauman, B. Leibe

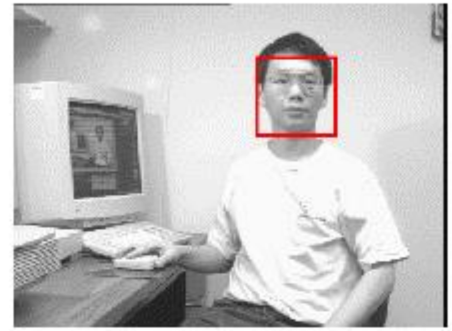
Viola-Jones Face Detector: Results

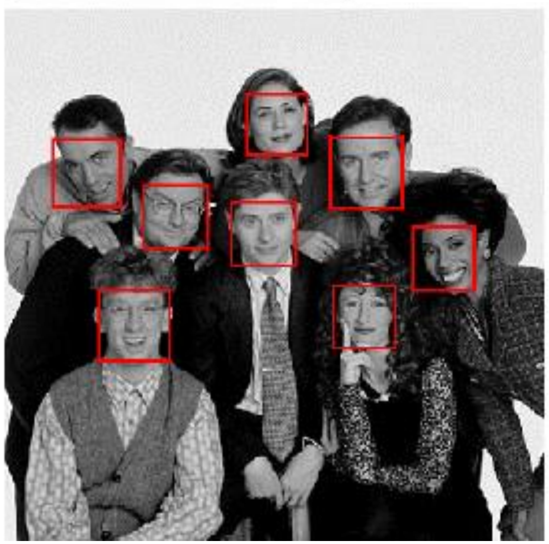


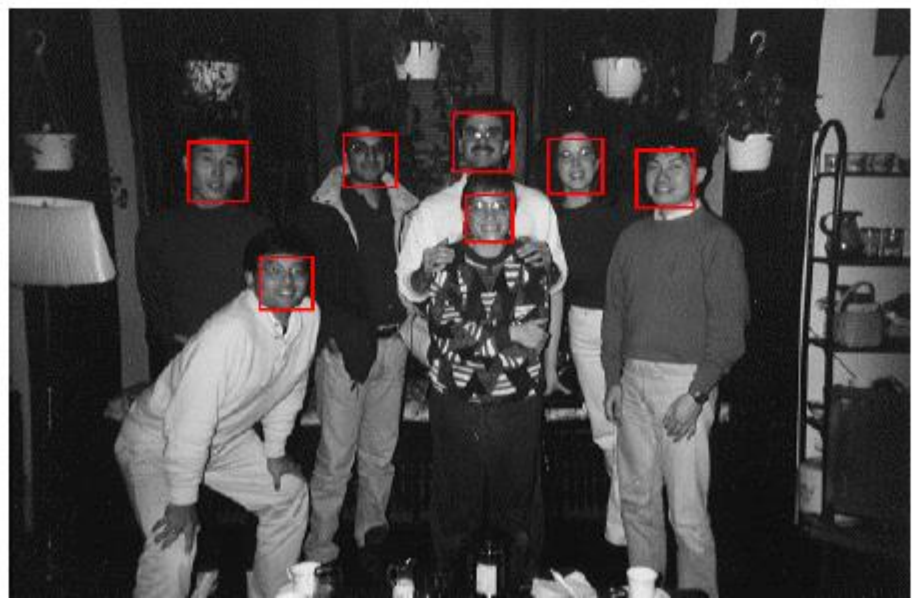
First two features selected

Viola-Jones Face Detector: Results

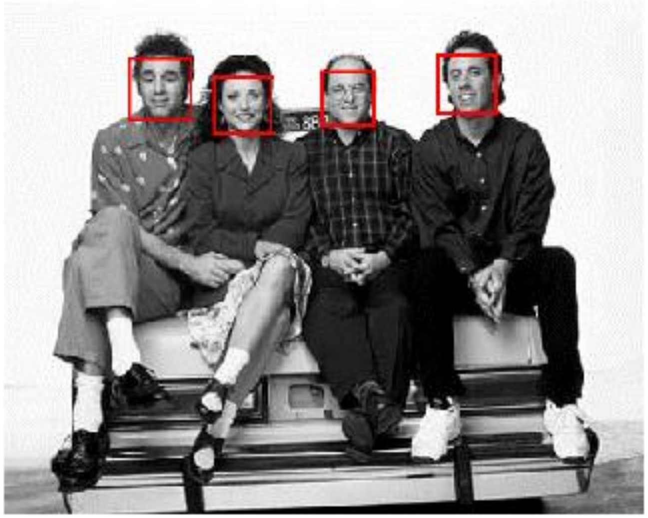


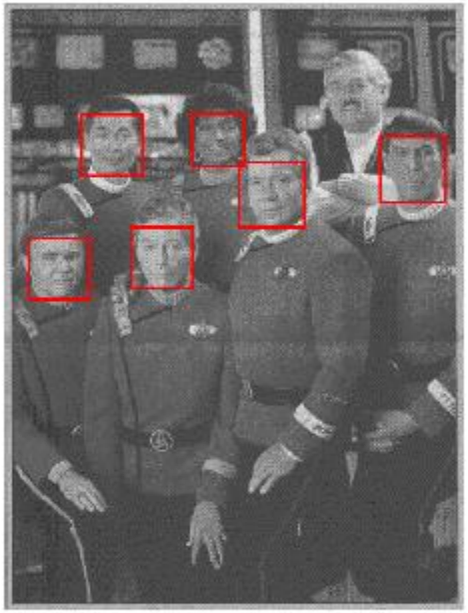


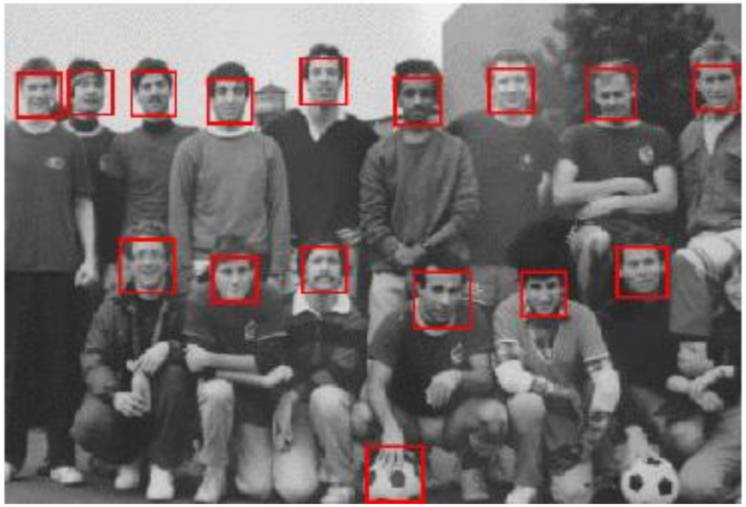


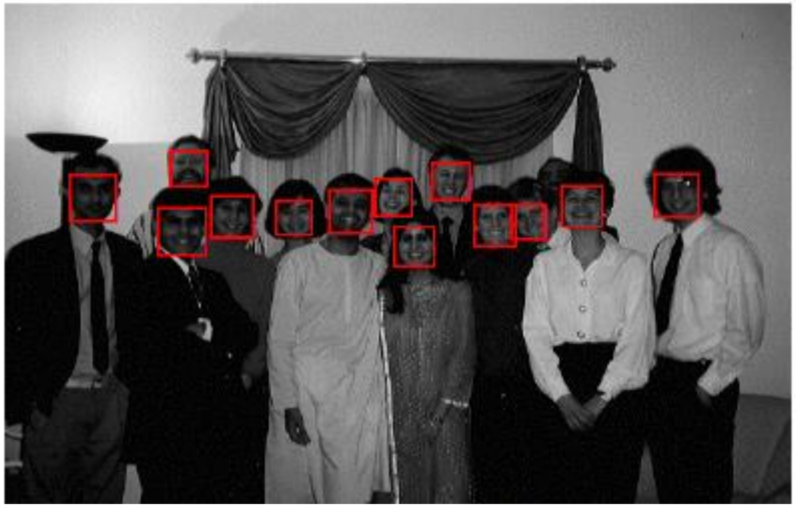


Viola-Jones Face Detector: Results





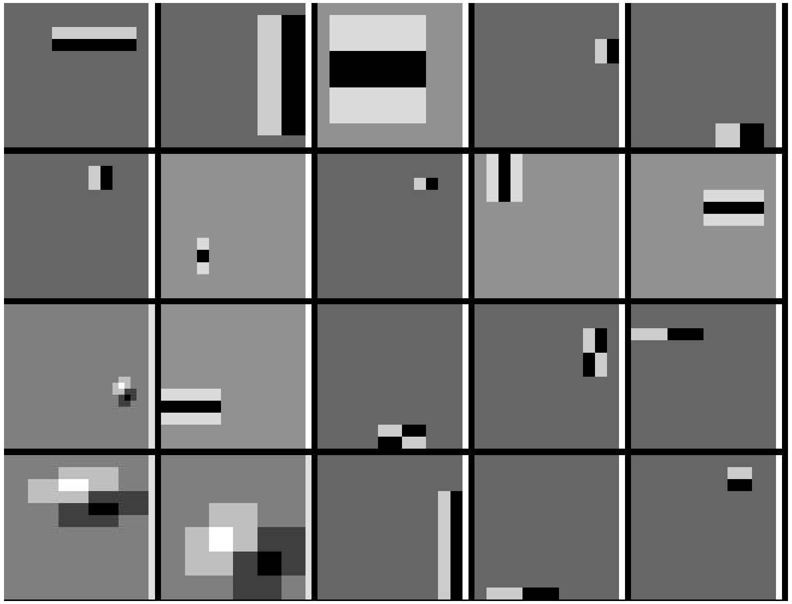




Viola-Jones Face Detector: Results

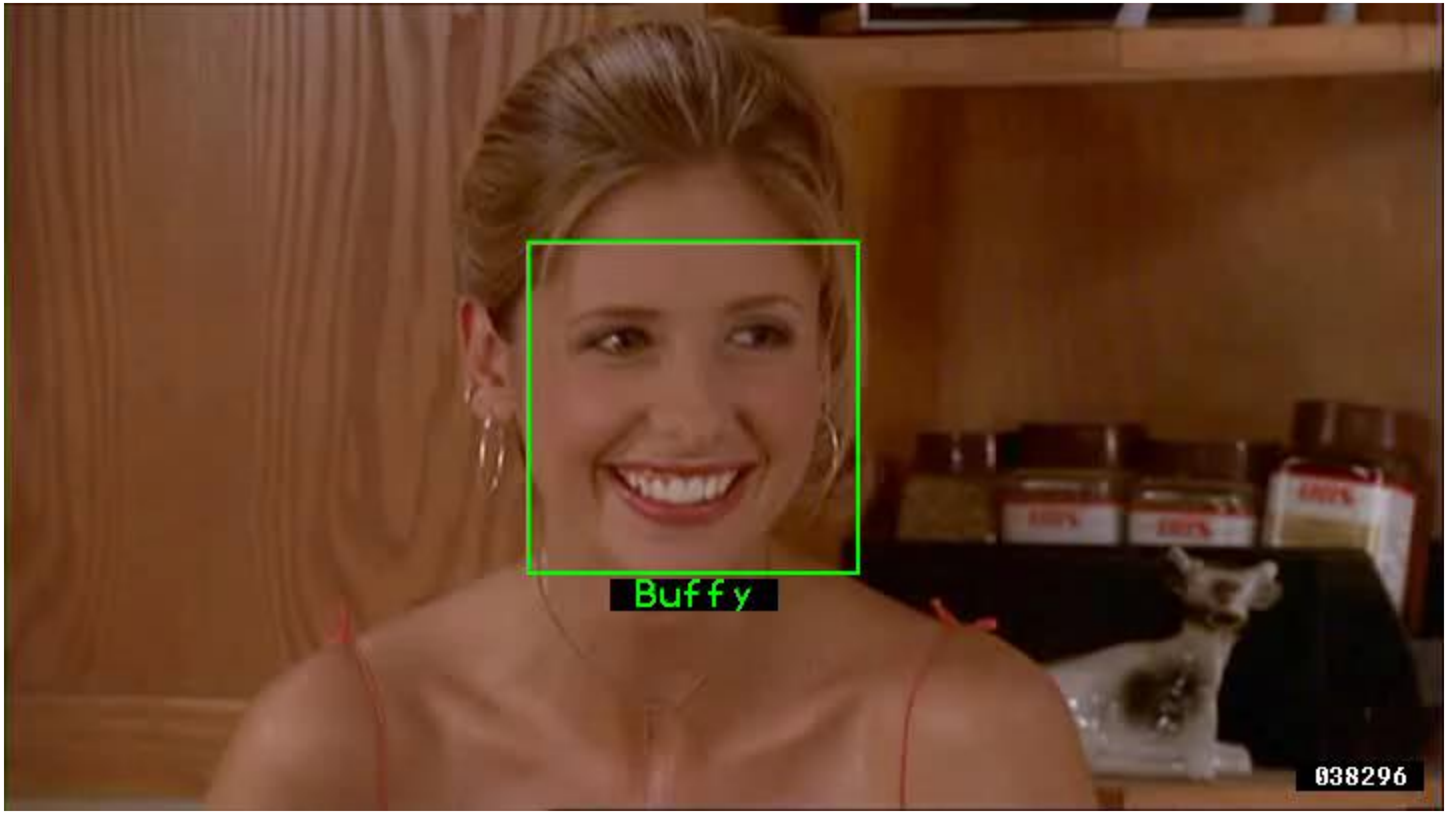
Detecting profile faces?

Detecting profile faces requires training separate detector with profile examples.



Viola-Jones Face Detector: Results

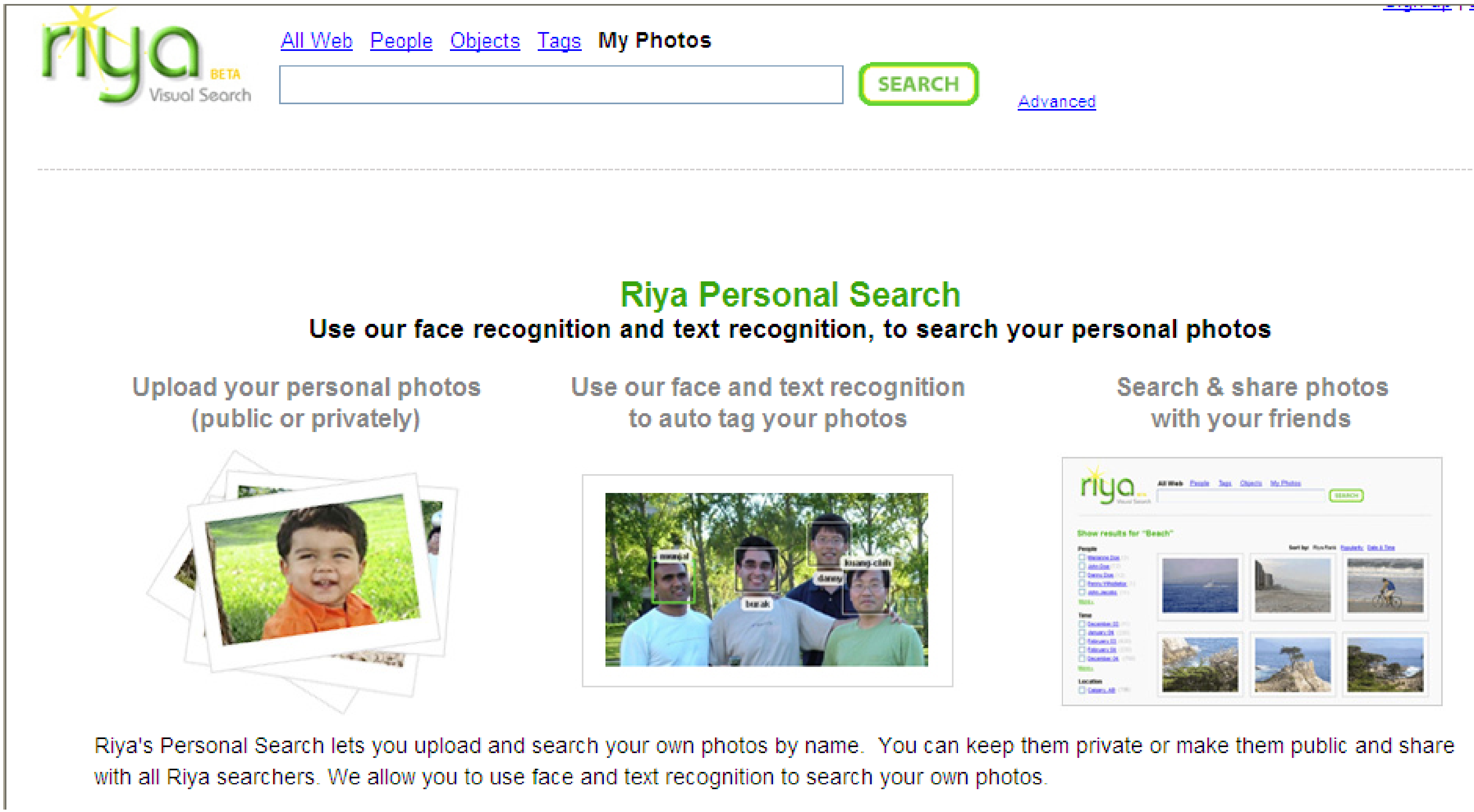
Example application



Frontal faces detected and then tracked, character names inferred with alignment of script and subtitles.

Everingham, M., Sivic, J. and Zisserman, A.
"Hello! My name is... Buffy" - Automatic naming of characters in TV vide
BMVC 2006.
http://www.robots.ox.ac.uk/~vgg/research/nface/index.html

Example application: faces in photos



Highlights

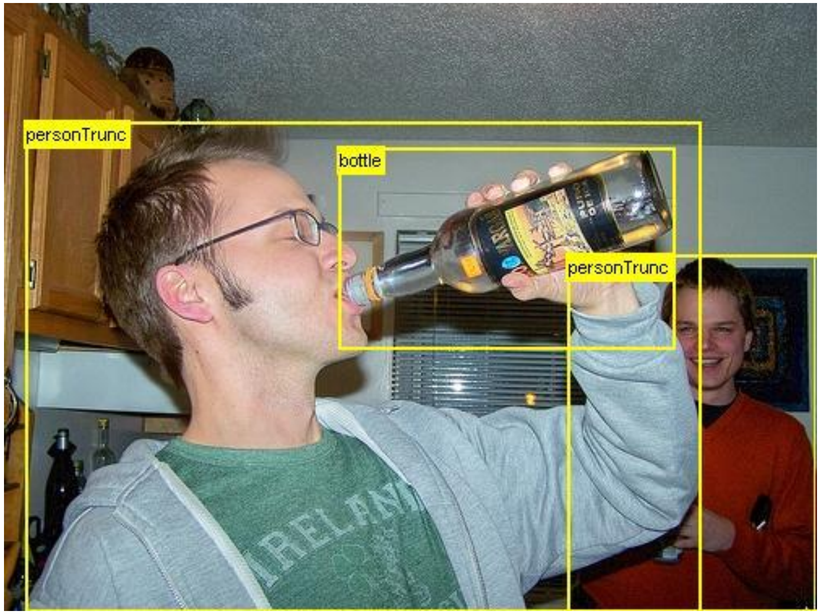
- Sliding window detection and global appearance descriptors:
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes

Limitations

- High computational complexity
  - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
  - If training binary detectors independently, means cost increases linearly with number of classes
- With so many windows, false positive rate better be low

Limitations (continued)

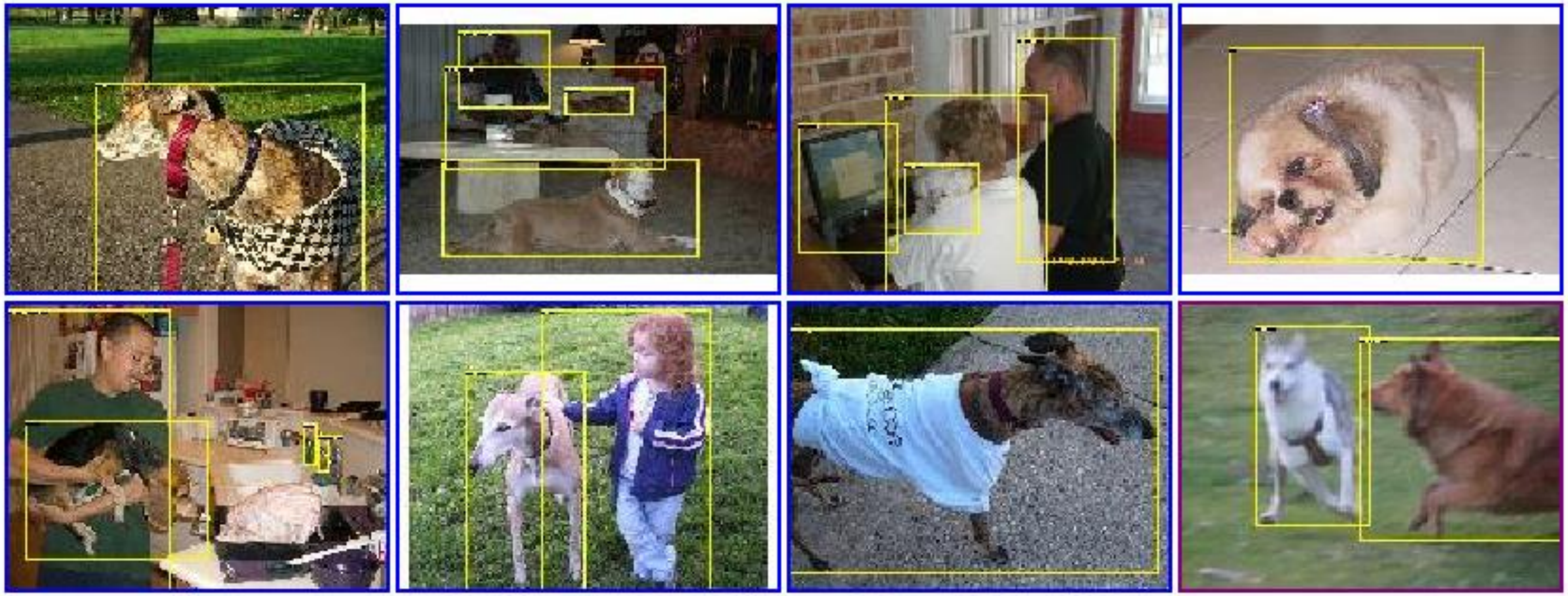
- Not all objects are "box" shaped



K. Grauman, B. Leibe

Limitations (continued)

- Non-rigid, deformable objects not captured well with representations assuming a fixed 2d structure; or must assume fixed viewpoint
- Objects with less-regular textures not captured well with holistic appearance-based descriptions

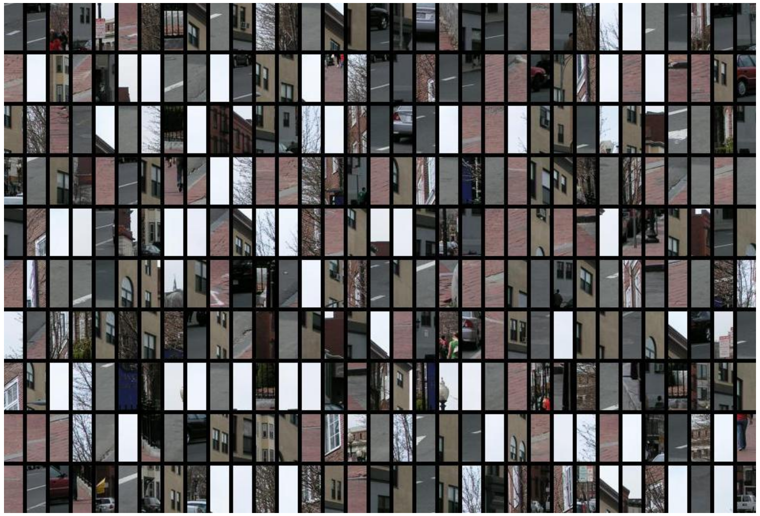


K. Grauman, B. Leibe

Limitations (continued)

- If considering windows in isolation, context is lost

Sliding window

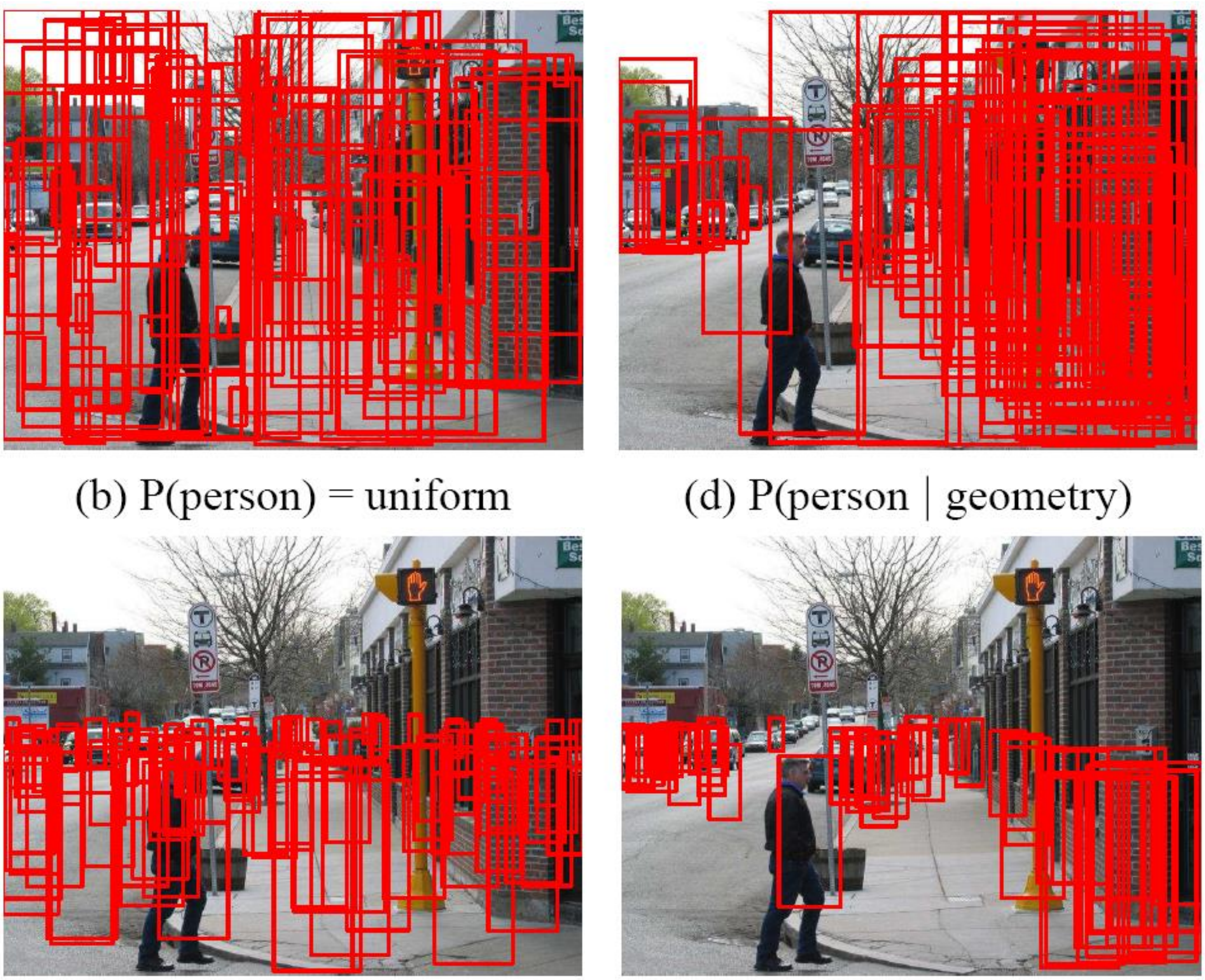


Detector's view

Figure credit: Derek Hoiem

K. Grauman, B. Leibe

Context can constrain a sliding window search



(b) P(person) = uniform

(d) P(person | geometry)

(f) P(person | viewpoint)

(g) P(person|viewpoint,geometry)

Hoiem, Efros, Herbert, 2006

Limitations (continued)

- In practice, often entails large, cropped training set (expensive)
- Requiring good match to a global appearance description can lead to sensitivity to partial occlusions

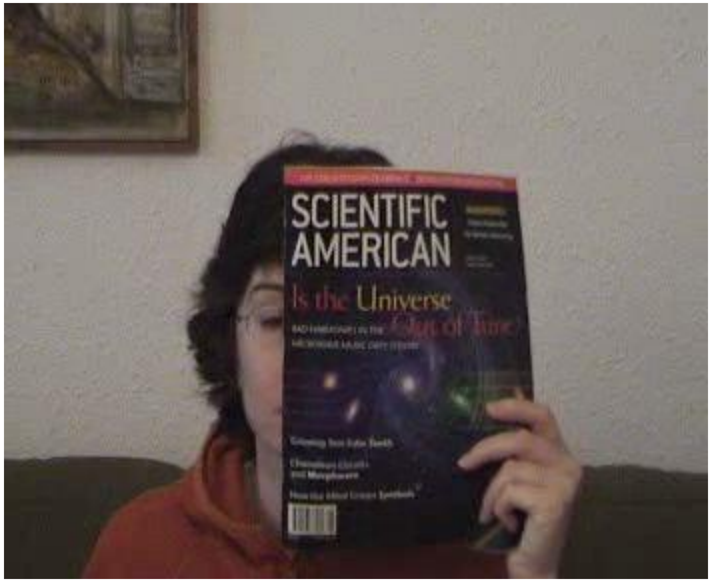
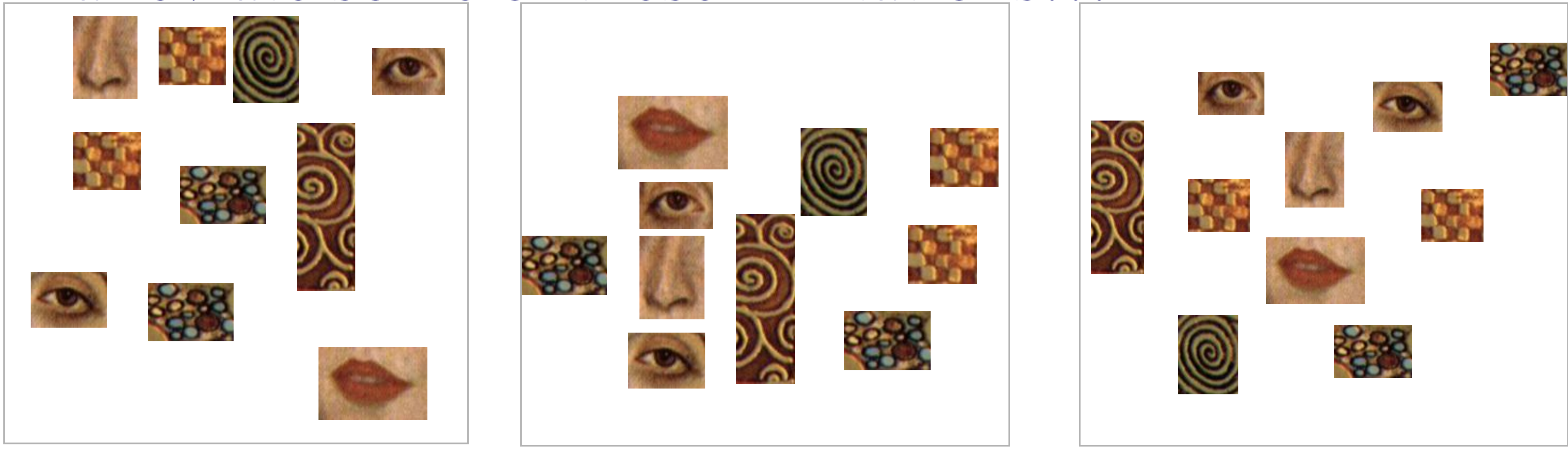


Image credit: Adam, Rivlin, & Shimshoni

Models based on local features will alleviate some of these limitations…

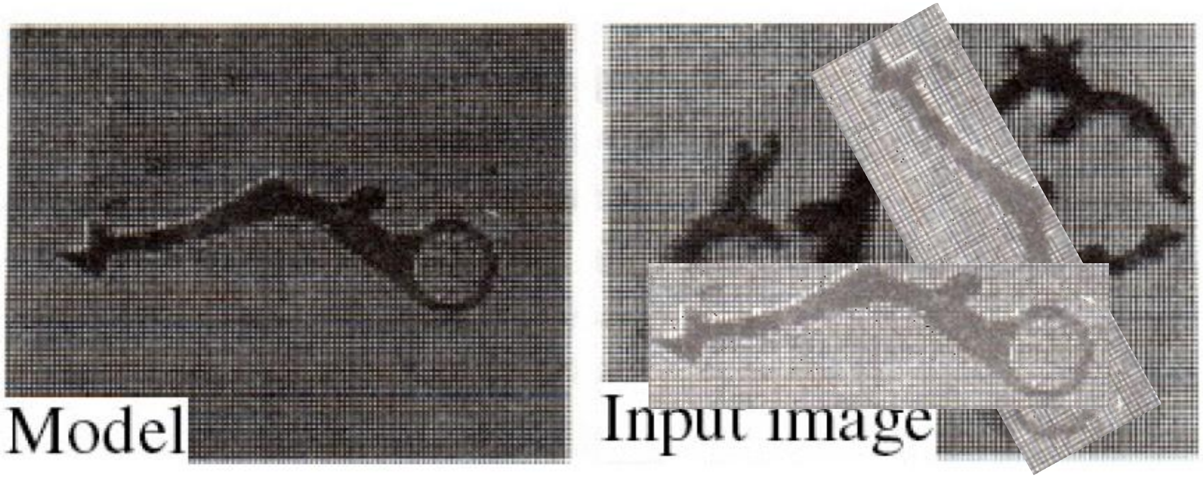


K. Grauman, B. Leibe

Local-feature Alignment

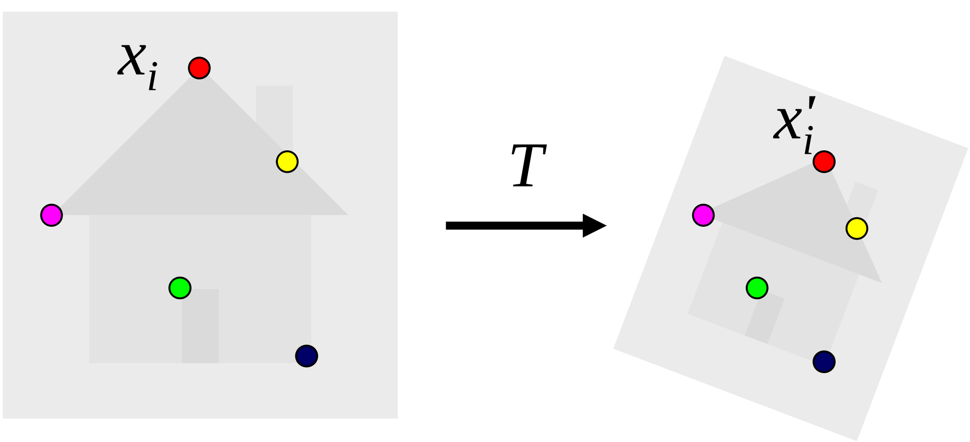
Hypothesize and test: main idea

- Given model of object
- New image: hypothesize object identity and pose
- Render object in camera
- Compare rendering to actual image: if close, good hypothesis.



Recall: Alignment

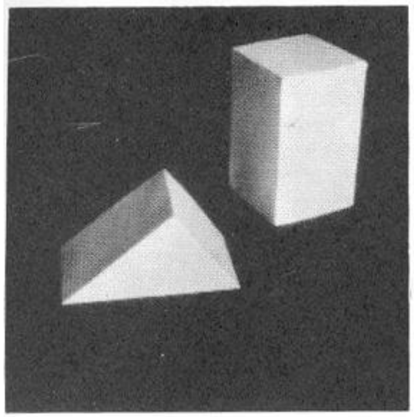
- Alignment: fitting a model to a transformation between pairs of features (*matches*) in two images



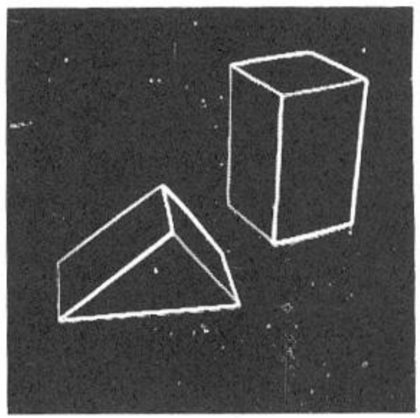
Find transformation T that minimizes

$$\sum_i \text{residual}(T(x_i), x_i')$$

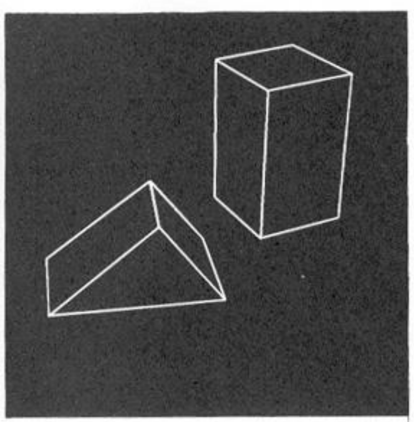
Alignment-based



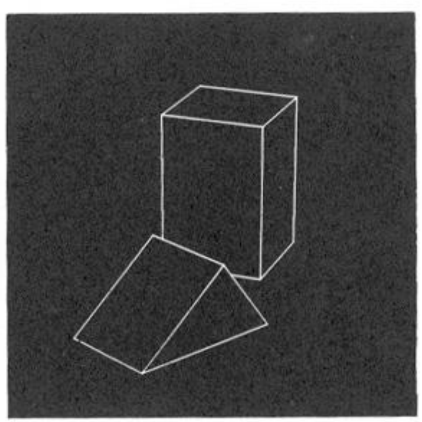
(a) Original picture.



(b) Differentiated picture.



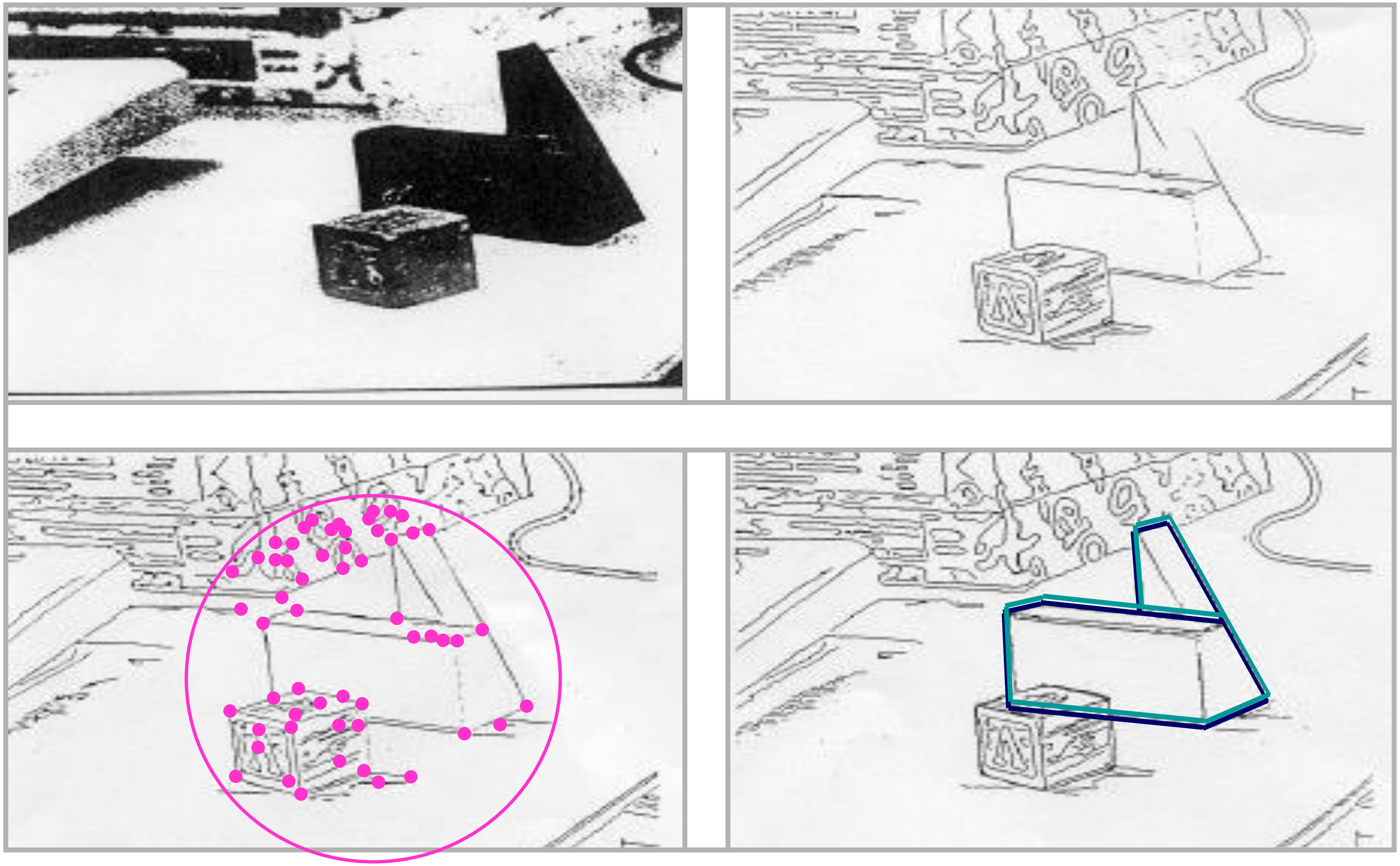
(c) Line drawing.



(d) Rotated view.

L. G. Roberts, *Machine Perception of Three Dimensional Solids*, Ph.D. thesis, MIT Department of Electrical Engineering, 1963.

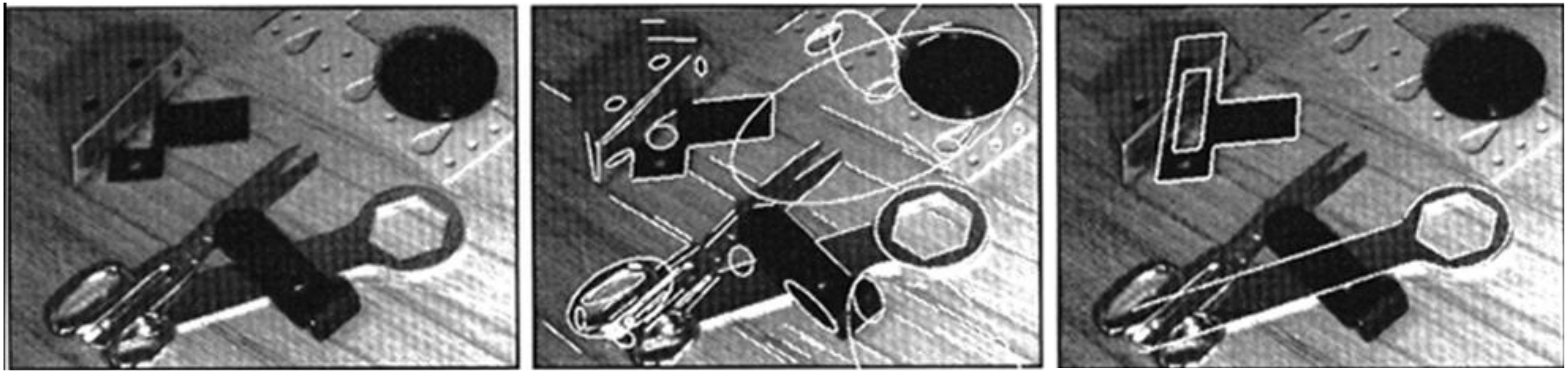
Alignment-based

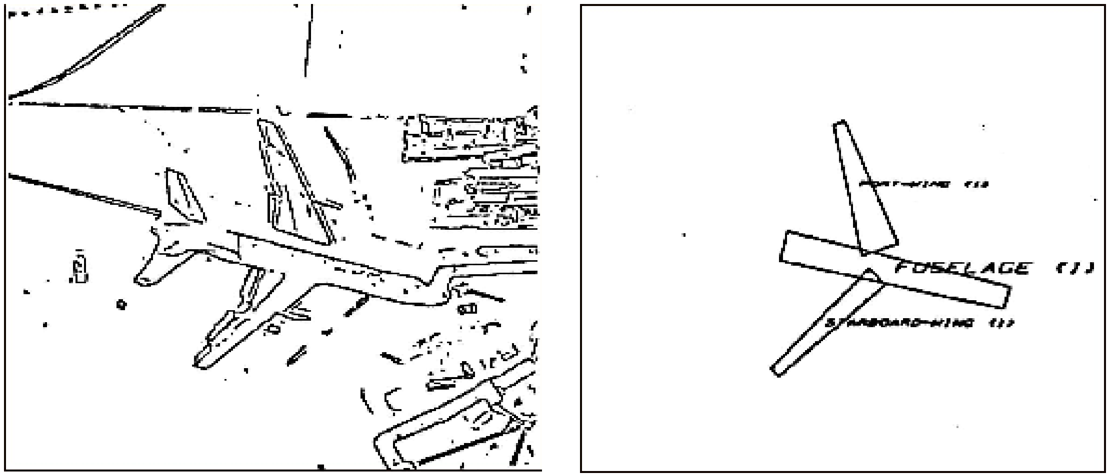


Huttenlocher & Ullman (1987)

Source: Lana Lazebnik

Alignment-based





ACRONYM (Brooks and Binford, 1981)

How to form a hypothesis?

Given a particular model object, we can estimate the *correspondences* between image and model features

Use correspondence to estimate model pose relative to object coordinate frame

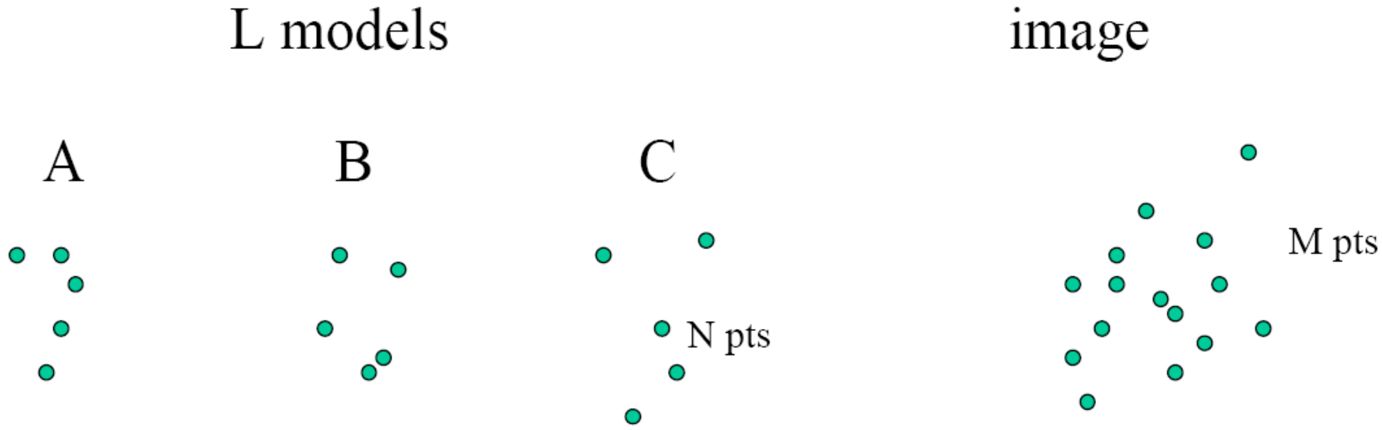
Generating hypotheses

We want a good correspondence between model features and image features.

- Brute force?

Brute force hypothesis generation

- For every possible model, try every possible subset of image points as matches for that model’s points.
- Say we have L objects with N features, M features in image



Generating hypotheses

We want a good correspondence between model features and image features.

- Brute force?

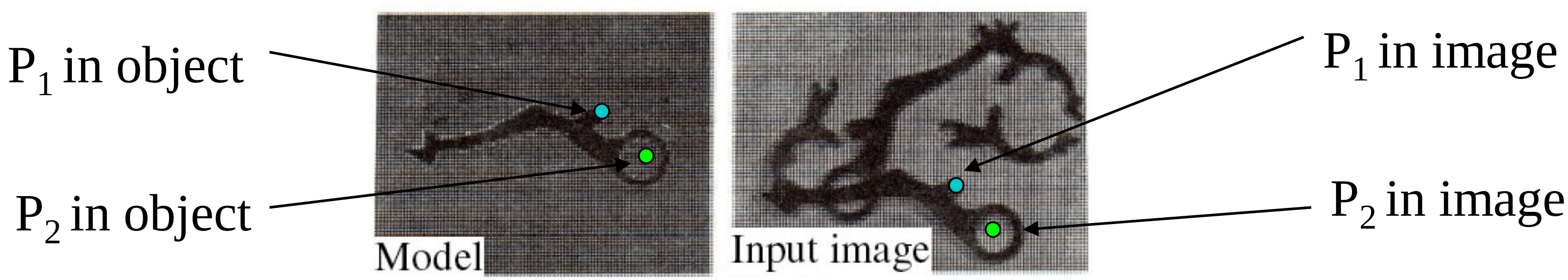
- **Pose consistency**, alignment: use subsets of features to estimate larger correspondence
- **Voting**, pose clustering

Pose consistency / alignment

- Key idea:
  - If we find good correspondences for a small set of features, it is easy to obtain correspondences for a much larger set.
- Strategy:
  - Generate hypotheses using small numbers of correspondences
  - Backproject: transform *all* model features to image features
  - Verify

Example: 2d affine mappings

- Say camera is looking down perpendicularly on planar surface



- We have two coordinate systems (object and image), and they are related by some affine mapping (rotation, scale, translation, shear).

Alignment: verification

- Given the back-projected model in the image:
  - Check if image edges coincide with predicted model edges
  - May be more robust if also require edges to have the same orientation
  - Consider texture in corresponding regions

- Possible issues?

Alignment: verification

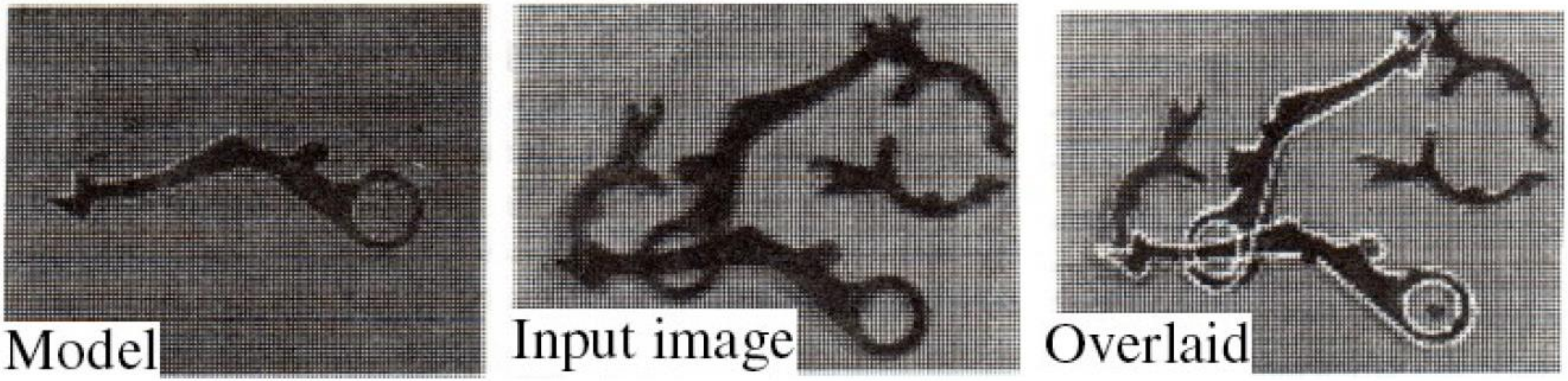
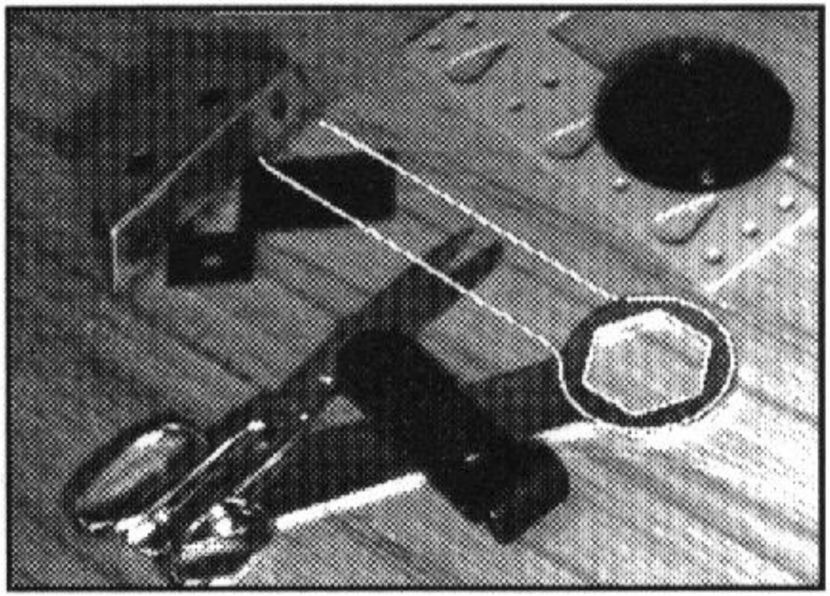


Figure from "Object recognition using alignment," D.P. Huttenlocher and S. Ullman, Proc. Int. Conf. Computer Vision, 1986, copyright IEEE, 1986

Alignment: verification



Issue with hypothesis & test approach

- May have false matches
  - We want *reliable* features to form the matches
    - **Local invariant features** useful to find matches, and to verify hypothesis

      (SIFT, etc.)

- May be too many hypotheses to consider
  - We want to look at the *most likely* hypotheses first
    - **Pose clustering (voting):** Narrow down number of hypotheses to verify by letting features *vote* on model parameters.

Pose clustering (voting)

- Narrow down the number of hypotheses to verify: identify those model poses that a lot of features agree on.
  - Use each group’s correspondence to estimate pose
  - Vote for that object pose in accumulator array (one array per object if we have multiple models)

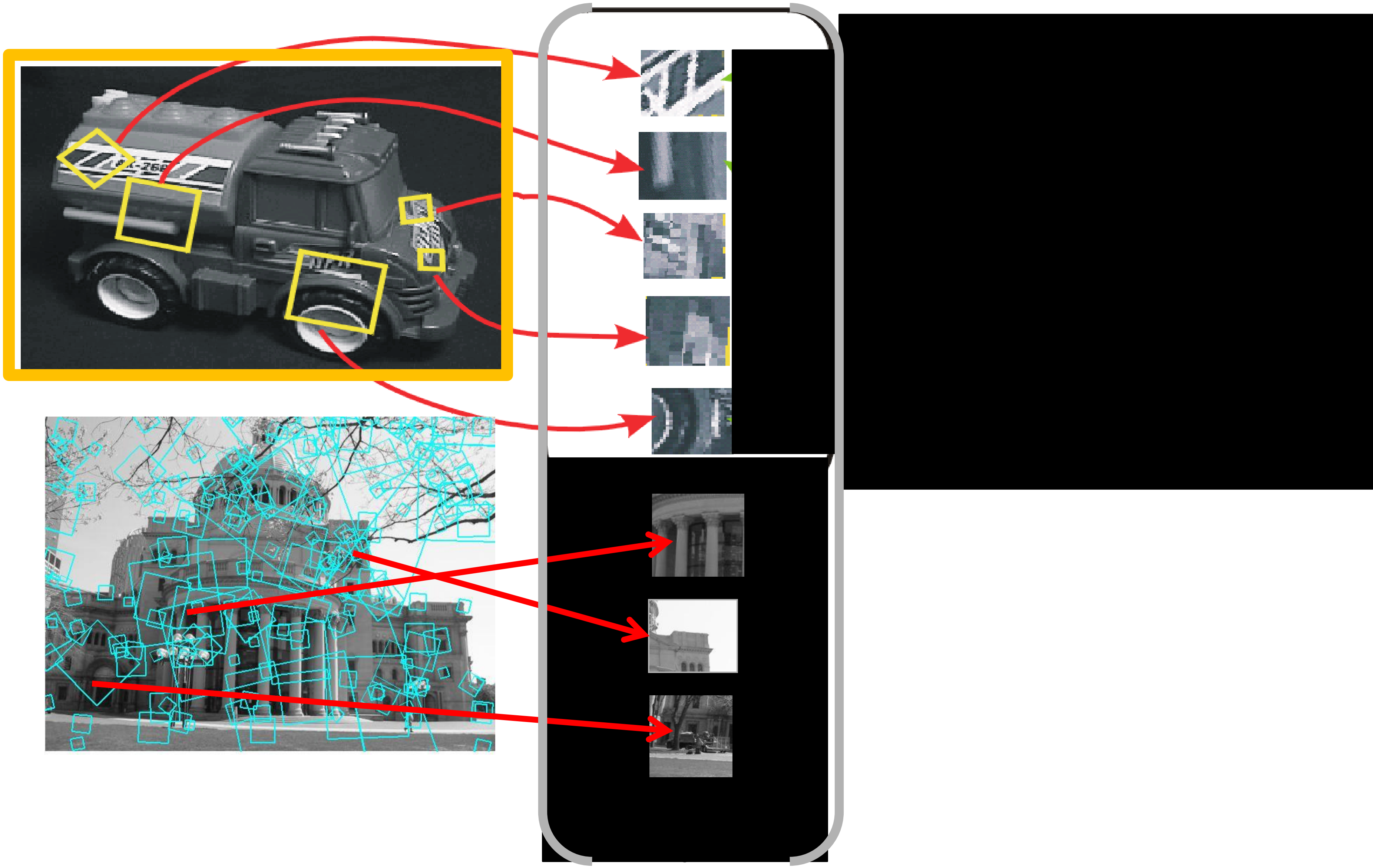
- Local invariant features can give more reliable matches and means of verification

Pose clustering and verification with SIFT [Lowe]

To detect **instances** of objects from a model base:

1) Index descriptors (distinctive features narrow possible matches)

Indexing local features



Pose clustering and verification with SIFT [Lowe]

To detect **instances** of objects from a model base:

1) Index descriptors (distinctive features narrow possible matches)

2) Generalized Hough transform to vote for poses (keypoints have record of parameters relative to model coordinate system)

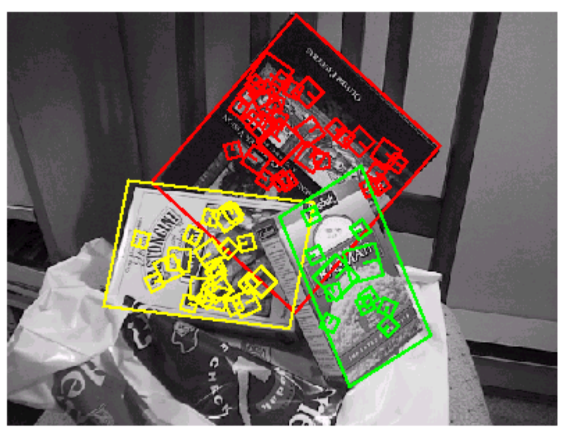
3) Affine fit to check for agreement between model and image features (approximates perspective projection for planar objects)

Planar objects

Model images and their SIFT keypoints

Input image

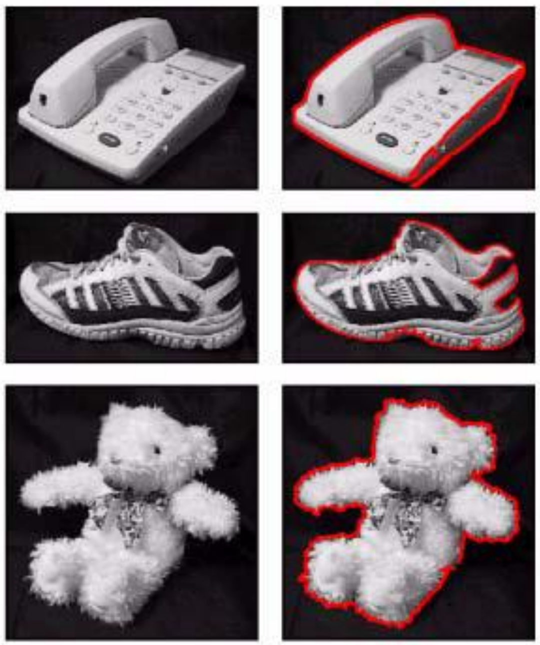
Model keypoints that were used to recognize, get least squares solution.



Recognition result

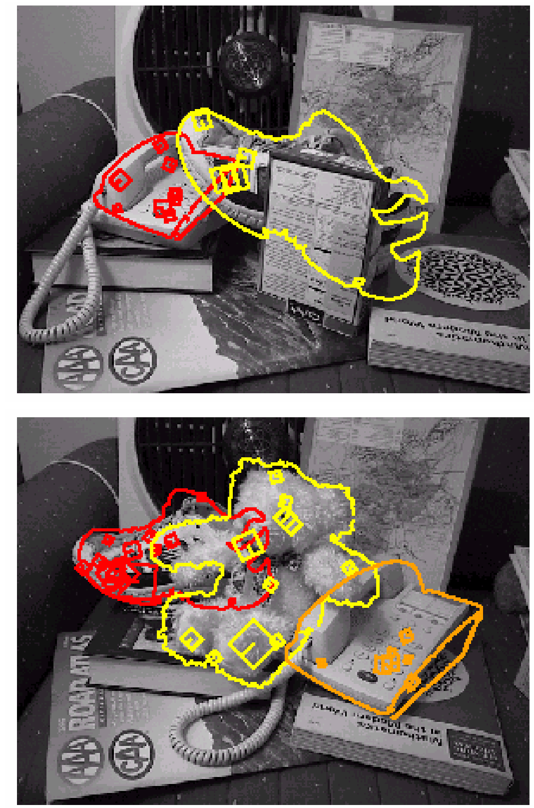
[Lowe]

3d objects



Background subtract for model boundaries

Objects recognized, though affine model not as accurate.



Recognition in spite of occlusion

[Lowe]

Recall: difficulties of voting

- Noise/clutter can lead to as many votes as true target
- Bin size for the accumulator array must be chosen carefully
- (Recall Hough Transform)

- In practice, good idea to make broad bins and spread votes to nearby bins, since verification stage can prune bad vote peaks.

A probabilistic interpretation (and re-tuning) of Lowe’s system:

P. Moreels and P. Perona, **"A probabilistic cascade of detectors for individual object recognition,"** European Conference on Computer Vision, 2008.

Coarse-to-Fine detection

- Progressively narrow down focus on correct region of hypothesis space
- Reject with little computation cost irrelevant regions of search space
- Use first information that is easy to obtain
- Simple building blocks organized in a cascade
- Probabilistic interpretation of each step

Score of an extended hypothesis

Features

assignments

observed features

Hypothesis:

model + position

geometry + appearance

database of models

$$P(H, V | F, M) = \frac{P(F, H, V | M)}{P(F | M)}$$

constant

Votes per model

Votes per model pose bin

(Hough transform)

$$P(F, H, V | M) = P(H | M) \cdot P(\bar{N} | H, M) \cdot P(\tilde{N} | \bar{N}, H, M)$$
$$\cdot P(V | \tilde{N}, \bar{N}, H, M) \cdot P(F | V, \tilde{N}, \bar{N}, H, M)$$

Prior on model

and poses

Prior on assignments

(before actual observations)

Consistency

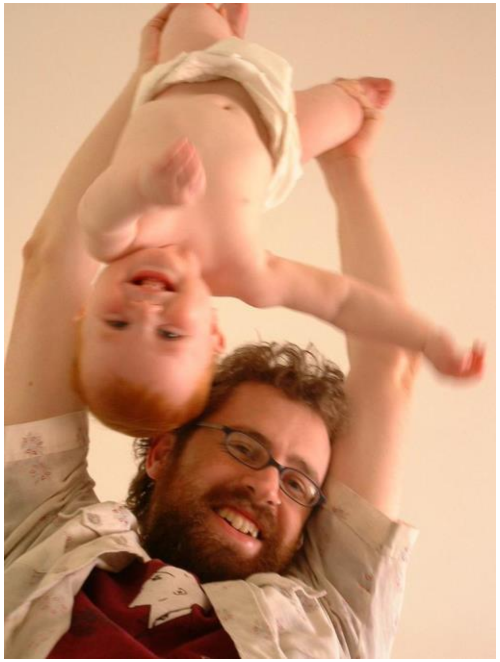
(after PROSAC)

Coarse data : prior knowledge

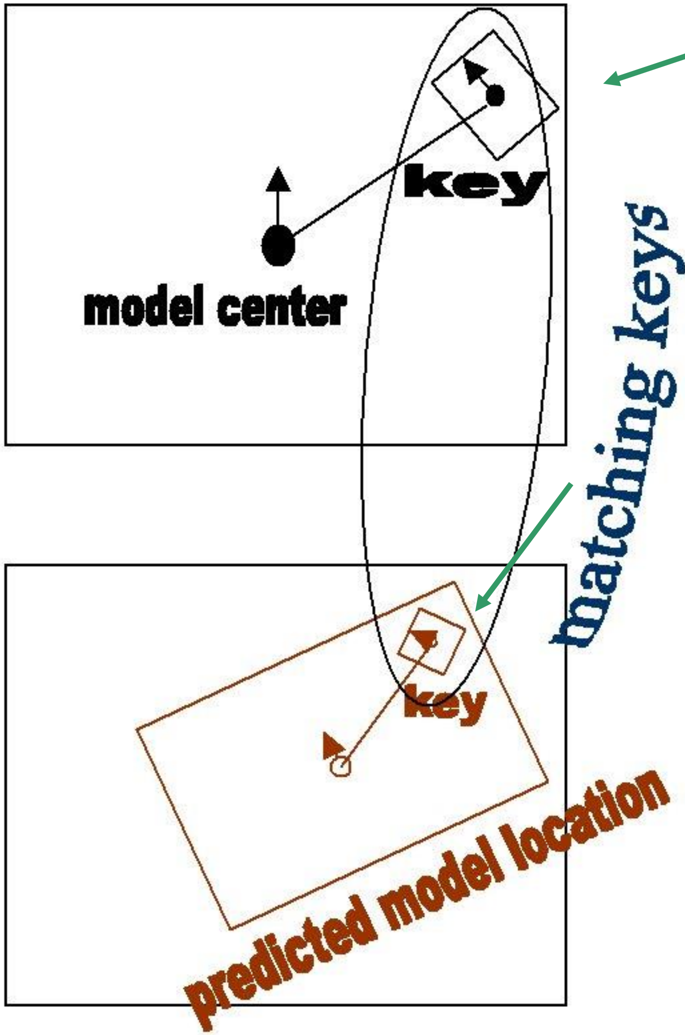
- Which objects are likely to be there, which pose are they likely to have ?

unlikely

situations



Coarse Hough transform



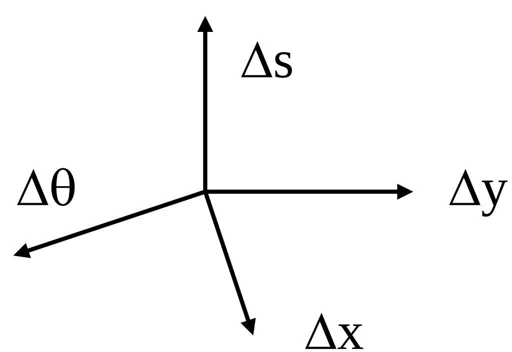
(x_1,y_1,s_1,θ_1)

(x_2,y_2,s_2,θ_2)

Transform predicted by this match:

- $\Delta x = x_2 - x_1$
- $\Delta y = y_2 - y_1$
- $\Delta s = s_2 / s_1$
- $\Delta\theta = \theta_2 - \theta_1$

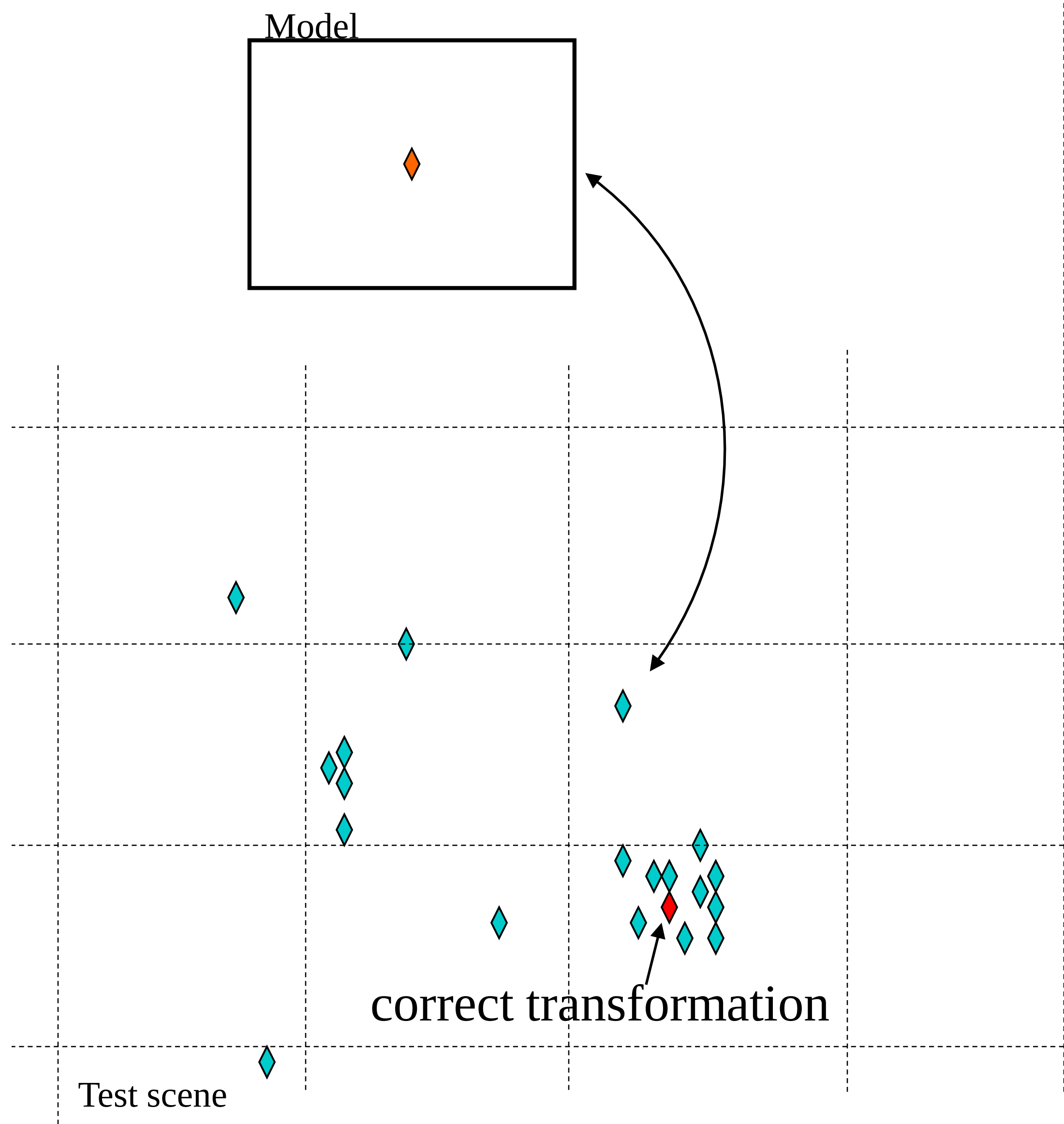
Each match is represented by a dot in the space of 2D similarities (Hough space)



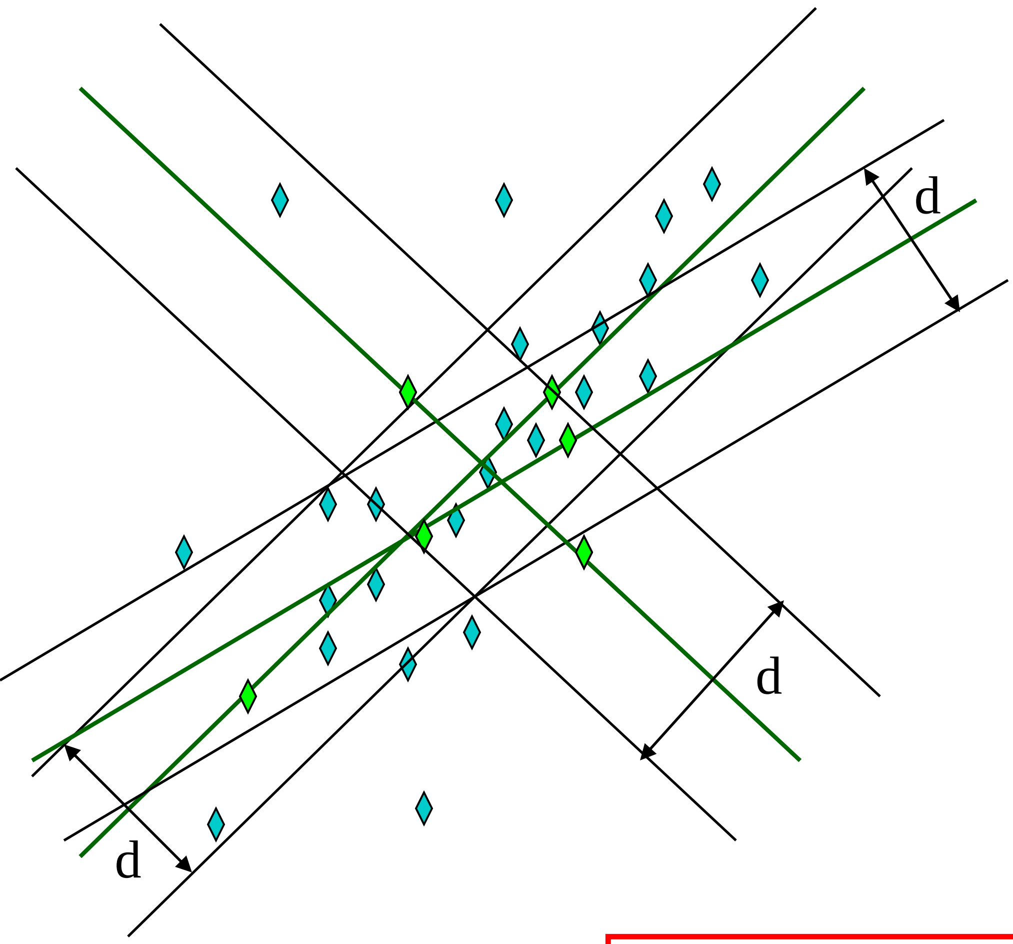
[Lowe1999,2004]

Coarse Hough transform

- Prediction of position of model center after transform
- The space of transform parameters is discretized into ‘bins’
- Coarse bins to limit boundary issues and have a low false-alarm rate for this stage
- We count the number $\tilde{N}$ of votes collected by each bin.



Correspondence or clutter ? PROSAC



- Similar to RANSAC – robust statistic for parameter estimation
- Priority to candidates with good **quality** of appearance match
- 2D affine transform : 6 parameters
  $\Rightarrow$ each sample contains 3 candidate correspondences.

[Fischler 1973]

[Chum&Matas 2005]

Output of PROSAC : pose transformation

+ set of features correspondences

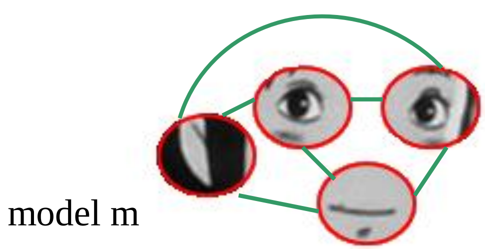
Consistency

Consistency between observations and predictions from hypothesis

$$P(F|V, \tilde{N}, \bar{N}, H, M) = \prod_{V(i) \neq 0} p_{fg}(f_i|H, f_{V(i)}) \cdot \prod_{V(i)=0} p_{bg}(f_i)$$

Common-frame approximation : parts are conditionally independent once reference position of the object is fixed. [Lowe1999,Huttenlocher90,Moreels04]

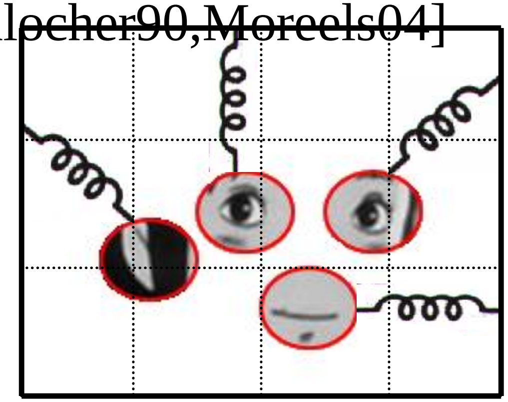
Constellation model



$$p_m(x_1 \ldots x_N)$$

$$D.O.F. = O(Parts^2)$$

Common-frame



$$\prod_i p_m(x_i|\Theta_m)$$

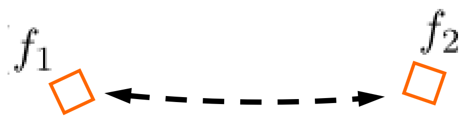
position of model m

$$D.O.F. = O(Parts)$$

Consistency

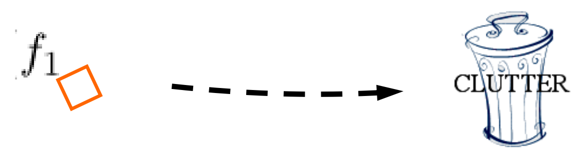
Consistency between observations and predictions from hypothesis

$$P(F|V,\tilde{N},\bar{N},H,M) = \prod_{V(i)\neq 0} p_{fg}(f_i|H,f_{V(i)}) \cdot \prod_{V(i)=0} p_{bg}(f_i)$$



foreground features

appearance geometry



'null' assignments

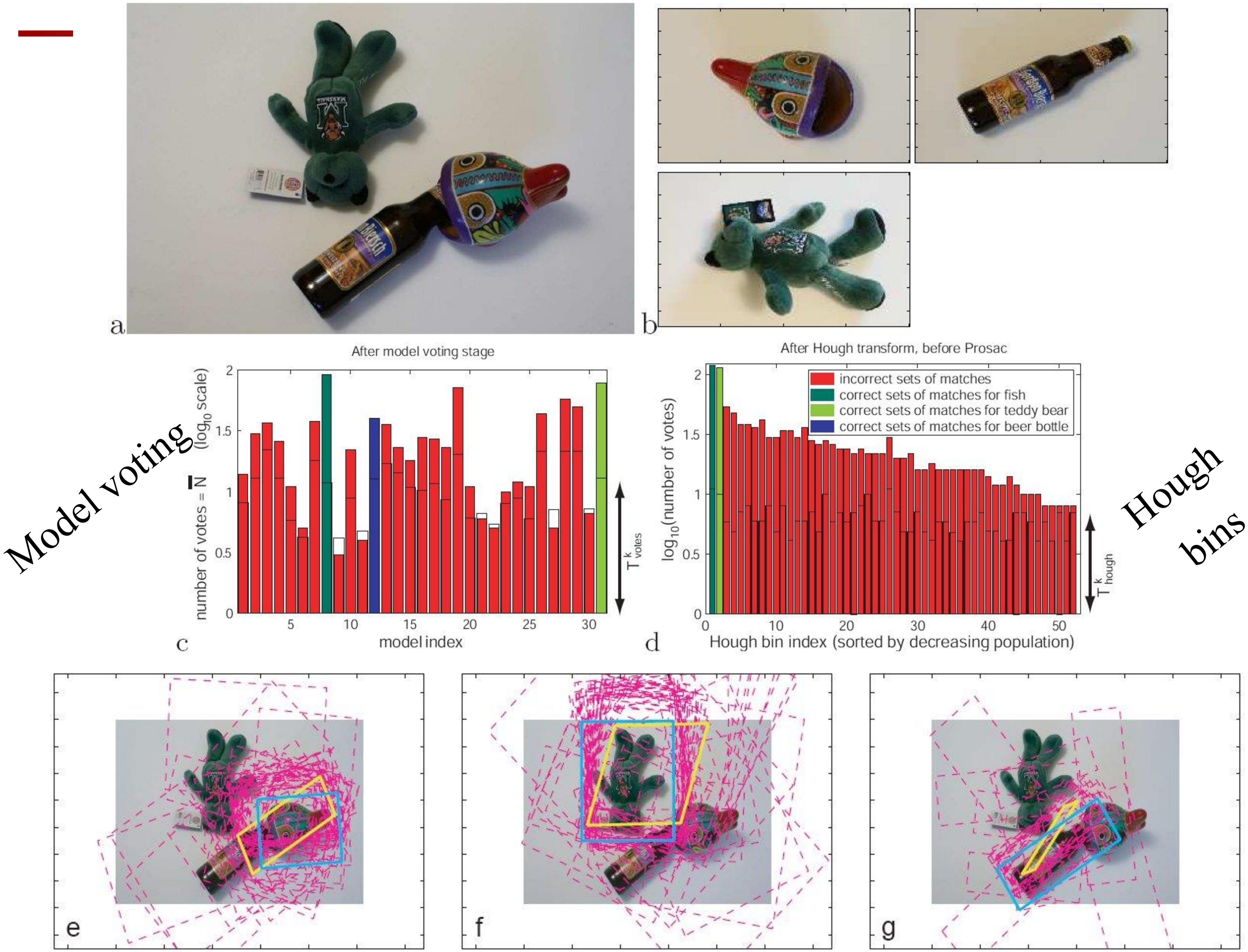
appearance geometry

$$p_{fg}(f_i|H,f_{V(i)}) = p_{fg,\mathcal{A}}(\mathcal{A}|H,\mathcal{A}_{V(i)}) \cdot p_{fg,\mathcal{X}}(\mathcal{X}|H,\mathcal{X}_{V(i)})$$

Consistency - appearance

Consistency - geometry

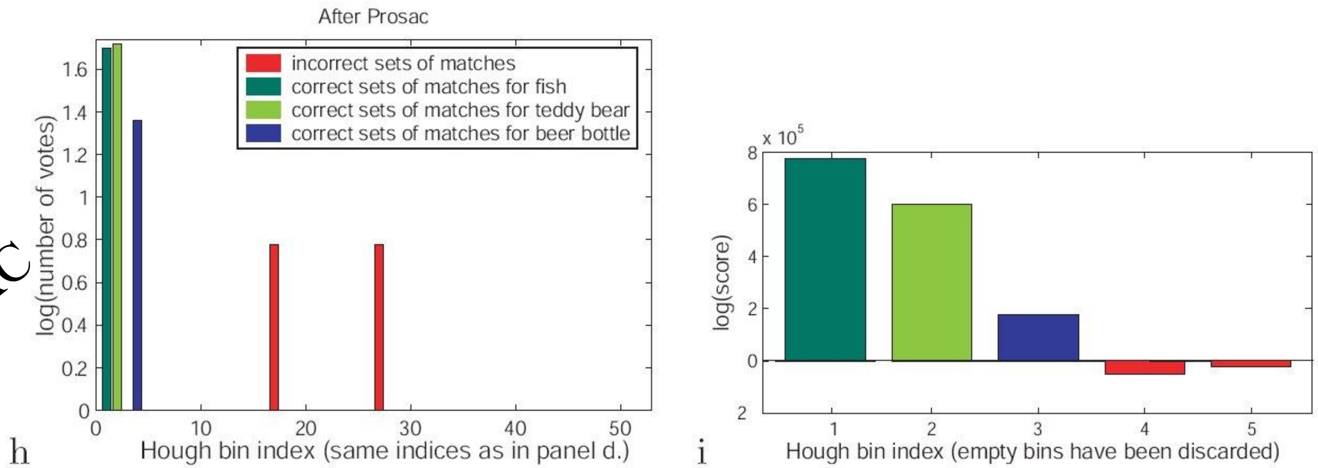
An example



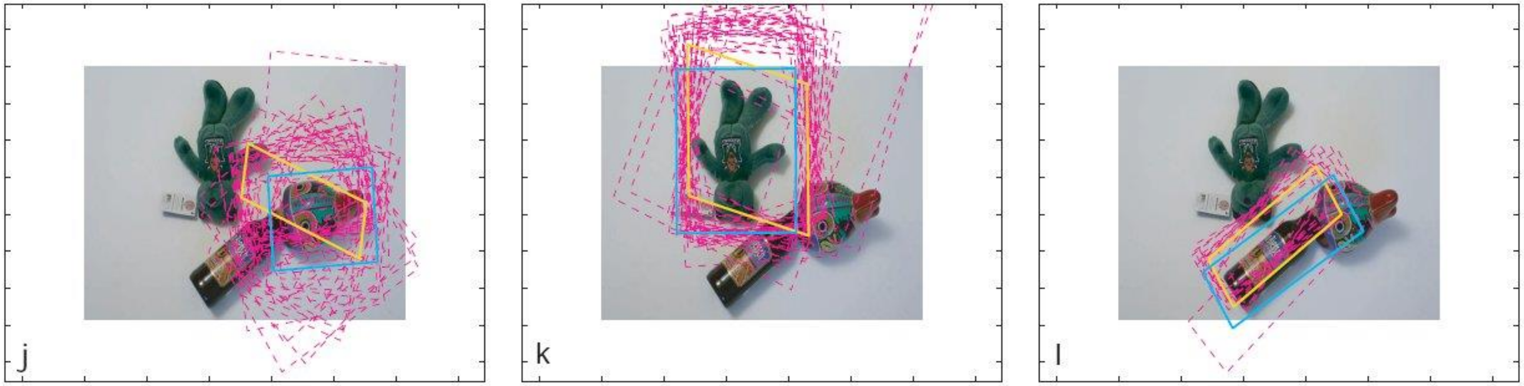
a
b
After model voting stage
number of votes = N̄ (log10 scale)
T^k_{votes}
model index
c
After Hough transform, before Prosac
$\log_{10}$(number of votes)
incorrect sets of matches
correct sets of matches for fish
correct sets of matches for teddy bear
correct sets of matches for beer bottle
T^k_{hough}
Hough bin index (sorted by decreasing population)
d
e
f
g
Model voting
Hough bins

An example

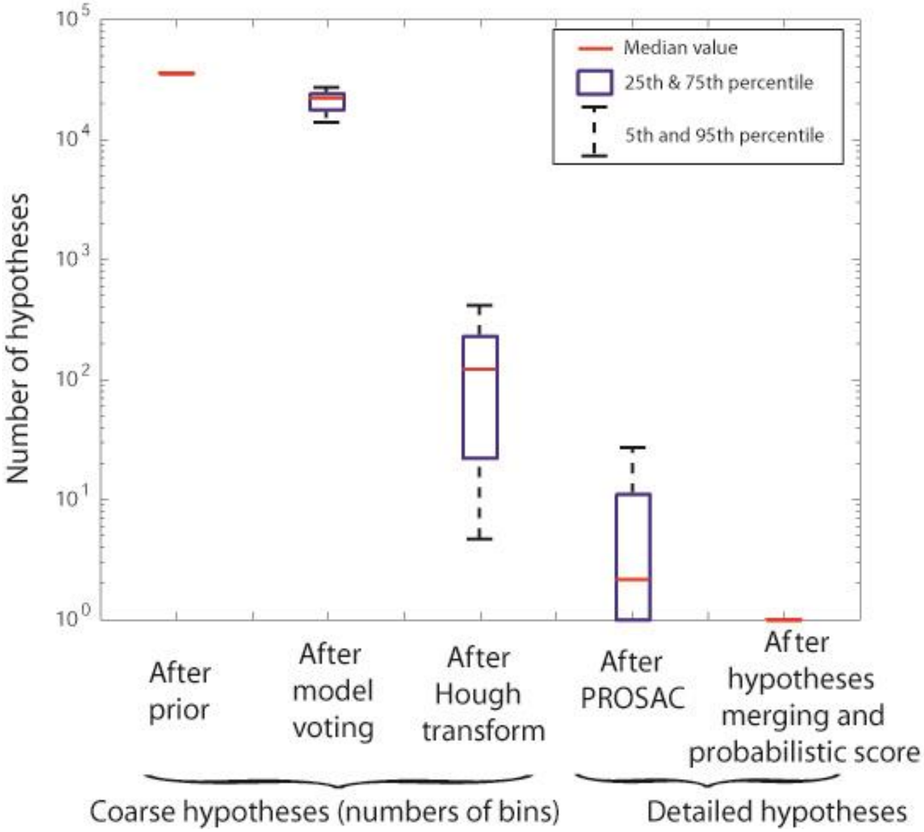
After PROSAC

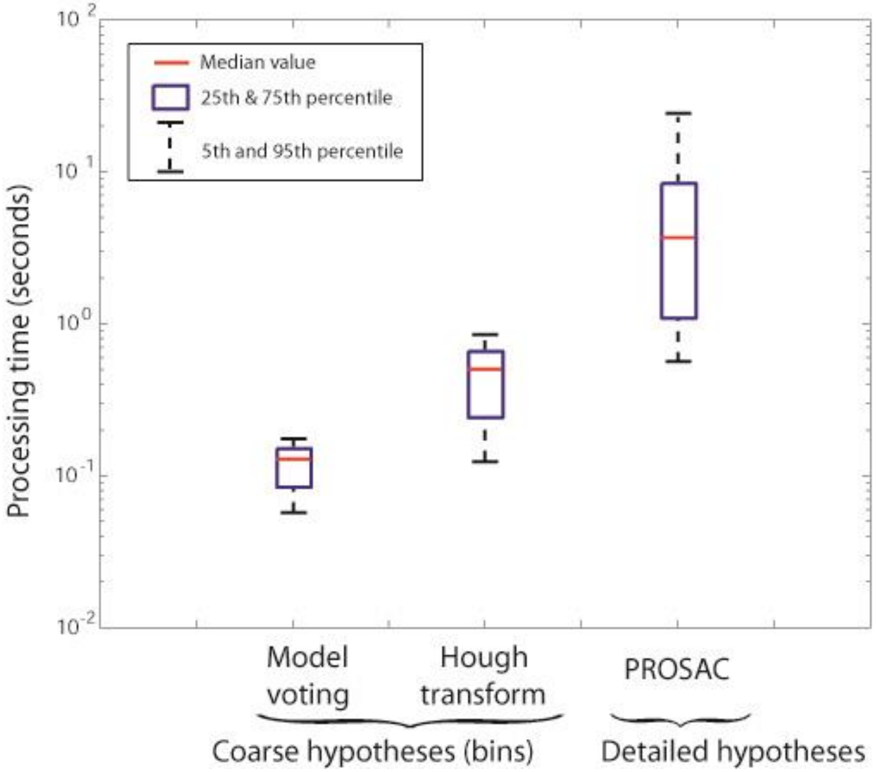


Probabilistic scores



Efficiency of coarse-to-fine processing





Giuseppe Toys database – Models

61 objects, 1-2 views/object

Giuseppe Toys database – Test scenes

141 test scenes

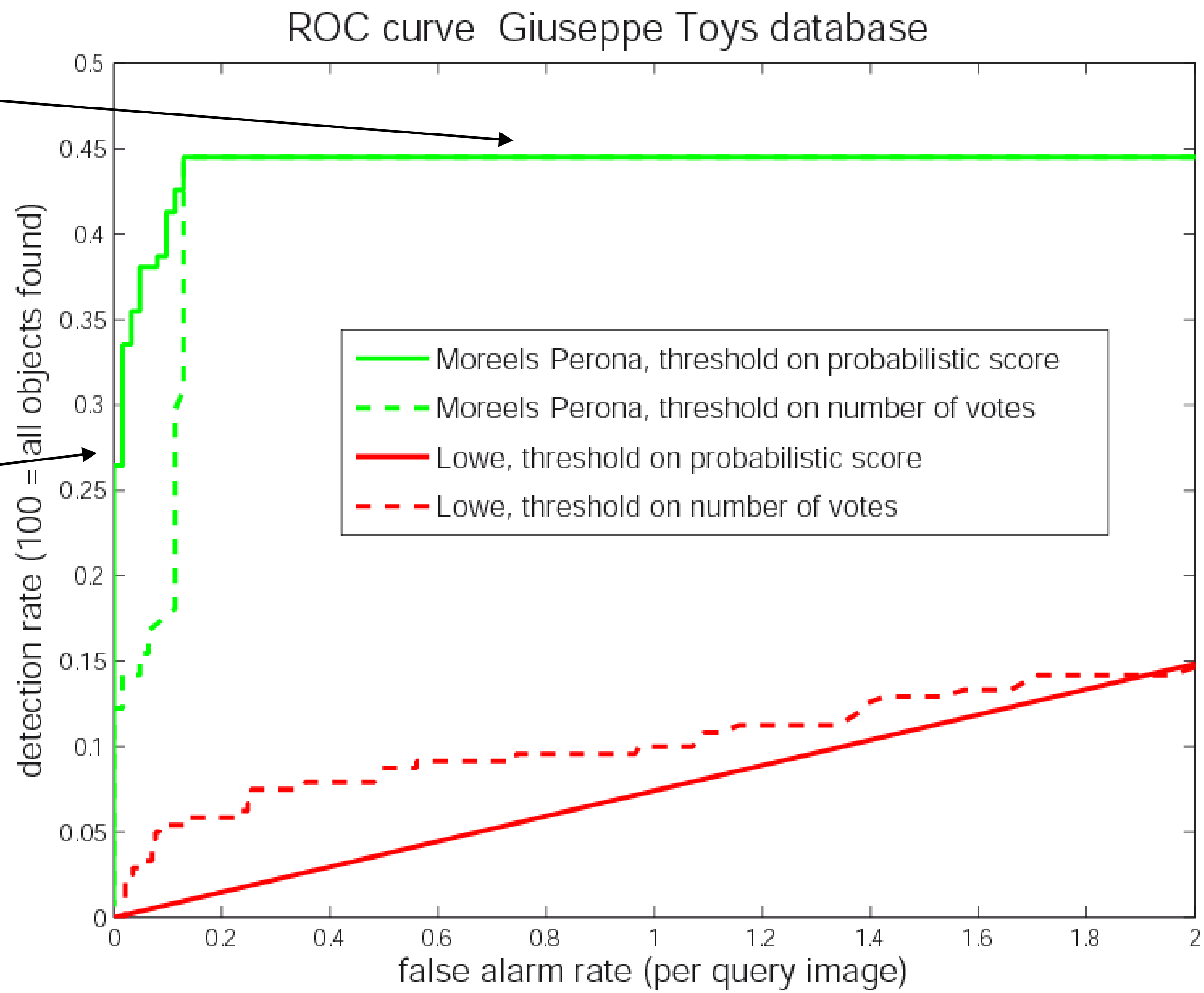
Results – Giuseppe Toys database

-

undetected objects:
features with poor appearance distinctiveness index to incorrect models

+

Lower false alarm rate
- more systematic verification of geometry consistency
- more consistent verification of geometric consistency



Lowe'99,'04

Conclusions – Moreels and Perona

- Coarse-to-fine strategy prunes irrelevant search branches at early stages.
- Probabilistic interpretation of each step.
- Higher performance than Lowe, especially in cluttered environment.
- Front end (features) needs more work for smooth or shiny surfaces.

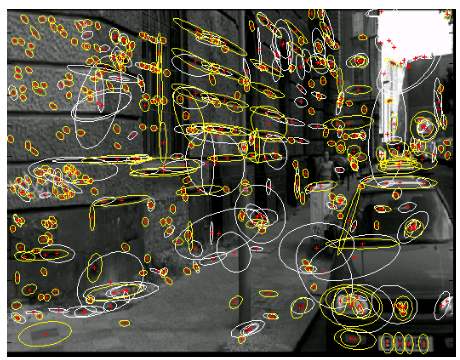
Scaling up: BOW Indexing

Outline of a large-scale retrieval strategy

regions

frames

invariant descriptor vectors

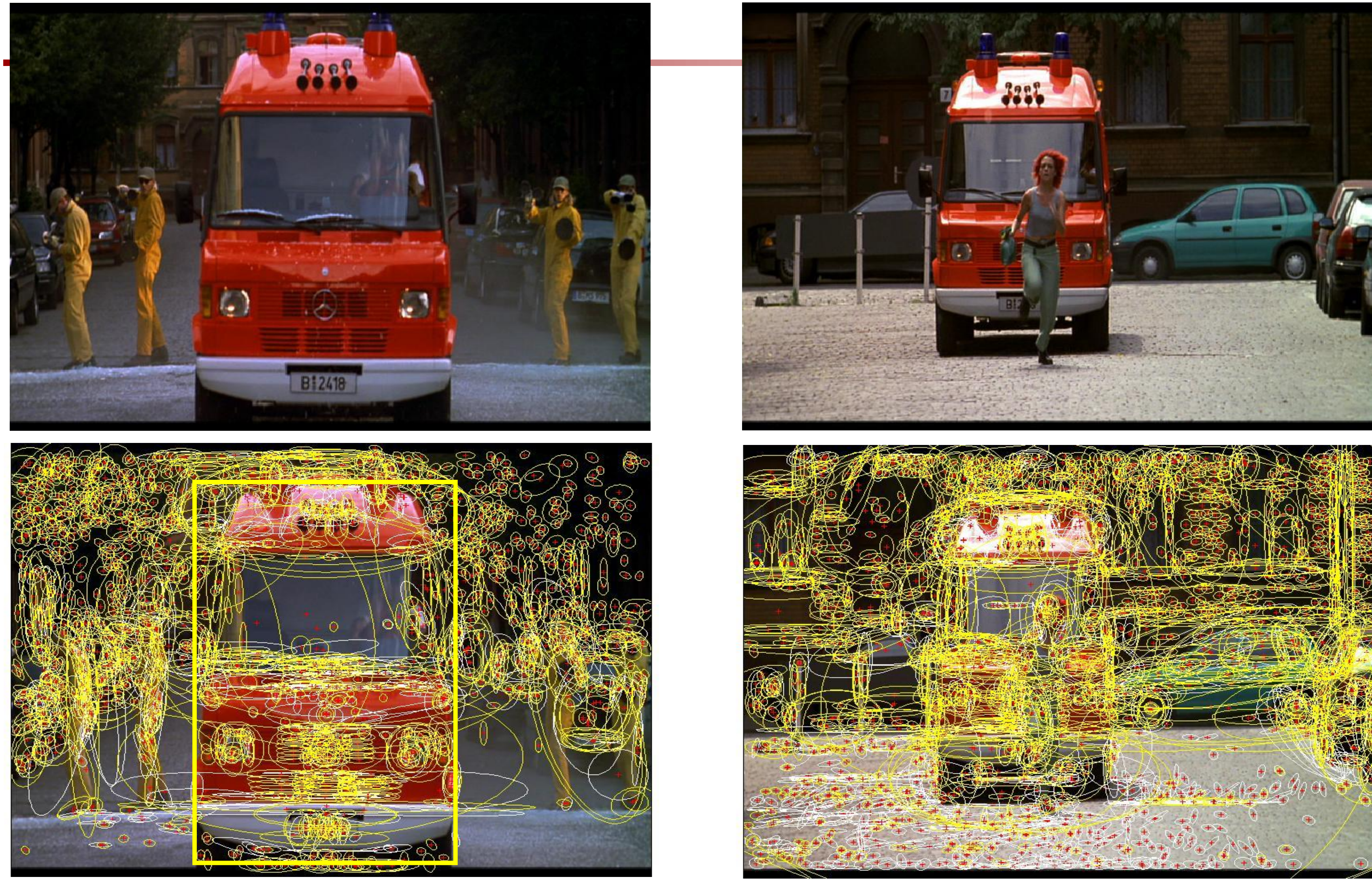


invariant descriptor vectors

1. Compute affine covariant regions in each frame independently
2. “Label” each region by a vector of descriptors based on its intensity
3. Finding corresponding regions is transformed to finding nearest neighbour vectors
4. Rank retrieved frames by number of corresponding regions
5. Verify retrieved frame based on spatial consistency

Slide credit: J. Sivic

Example of object recognition



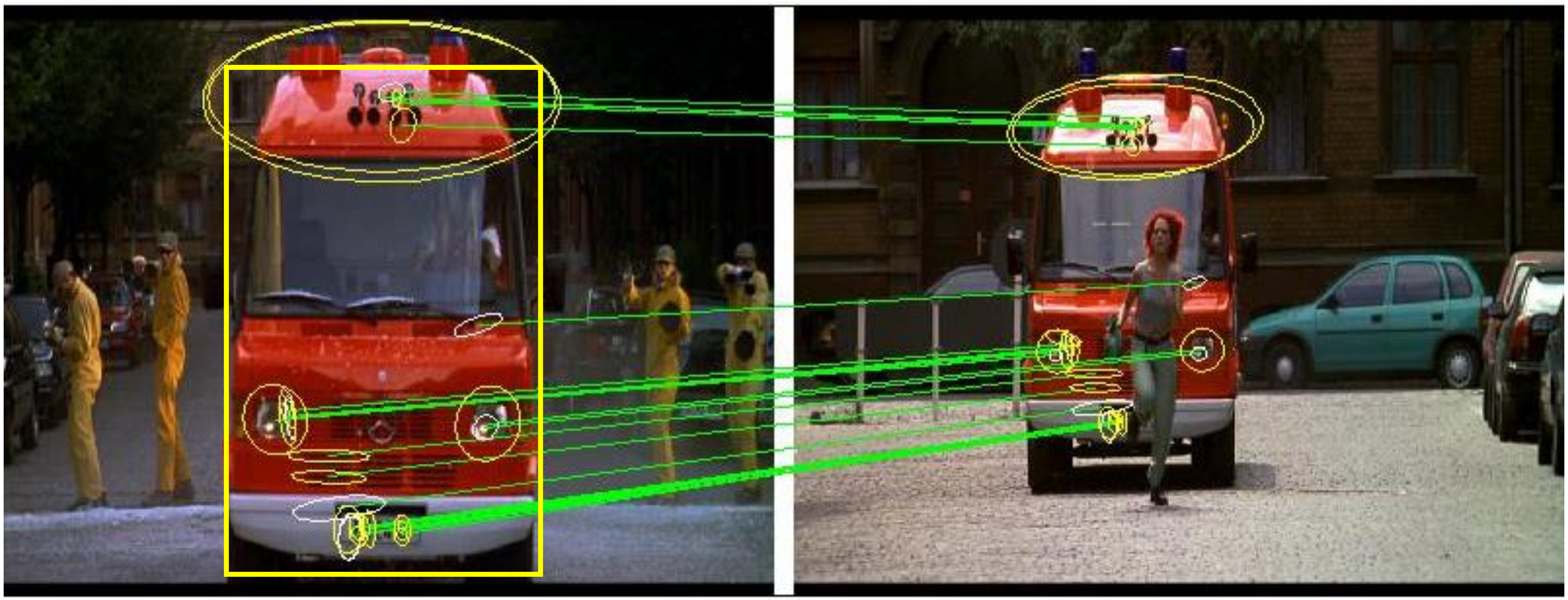
1000+ descriptors per frame

Shape adapted regions

Maximally stable regions

Slide credit: J. Sivic

Match regions between frames using SIFT descriptors and spatial consistency



Multiple regions overcome problem of partial occlusion

Shape adapted regions

Maximally stable regions

Slide credit: J. Sivic

Visual search using local regions

Schmid and Mohr '97 – 1k images

Sivic and Zisserman'03 – 5k images

Nister and Stewenius'06 – 50k images (1M)

Philbin et al.'07 – 100k images

Chum et al.'07 + Jegou and Schmid'07 – 1M images

Chum et al.'08 – 5M images

Index 1 billion (10^9) images

– 200 servers each indexing 5M images?

Slide credit: J. Sivic

Beyond Nearest Neighbors…
Indexing local features using inverted file index

Index

For text documents, an efficient way to find all *pages* on which a *word* occurs is to use an index…

We want to find all *images* in which a *feature* occurs.

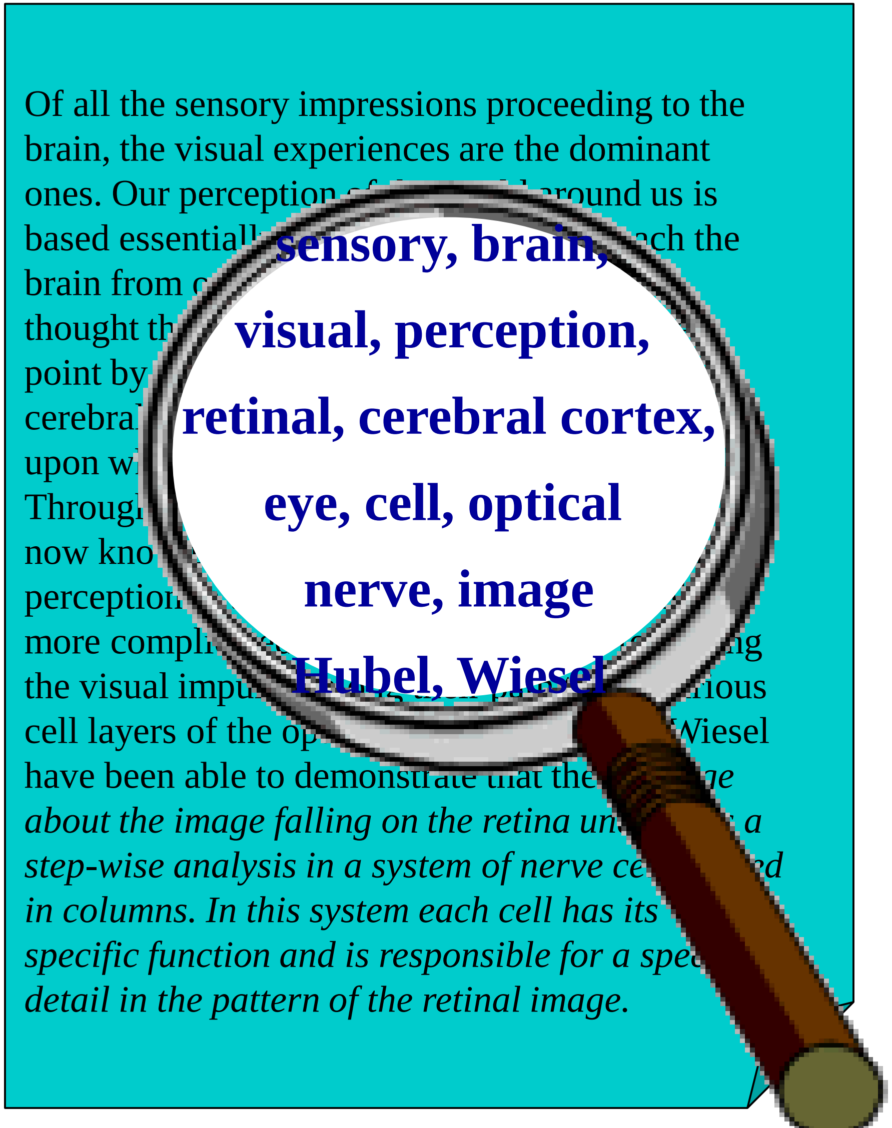
To use this idea, we'll need to map our features to "visual words".

Slide credit: J. Sivic

Object → Bag of 'words'

Slide credit L. Fei-Fei

Analogy to documents



Slide credit L. Fei-Fei

A clarification: definition of “BoW”

Looser definition

– Independent features

Slide credit L. Fei-Fei

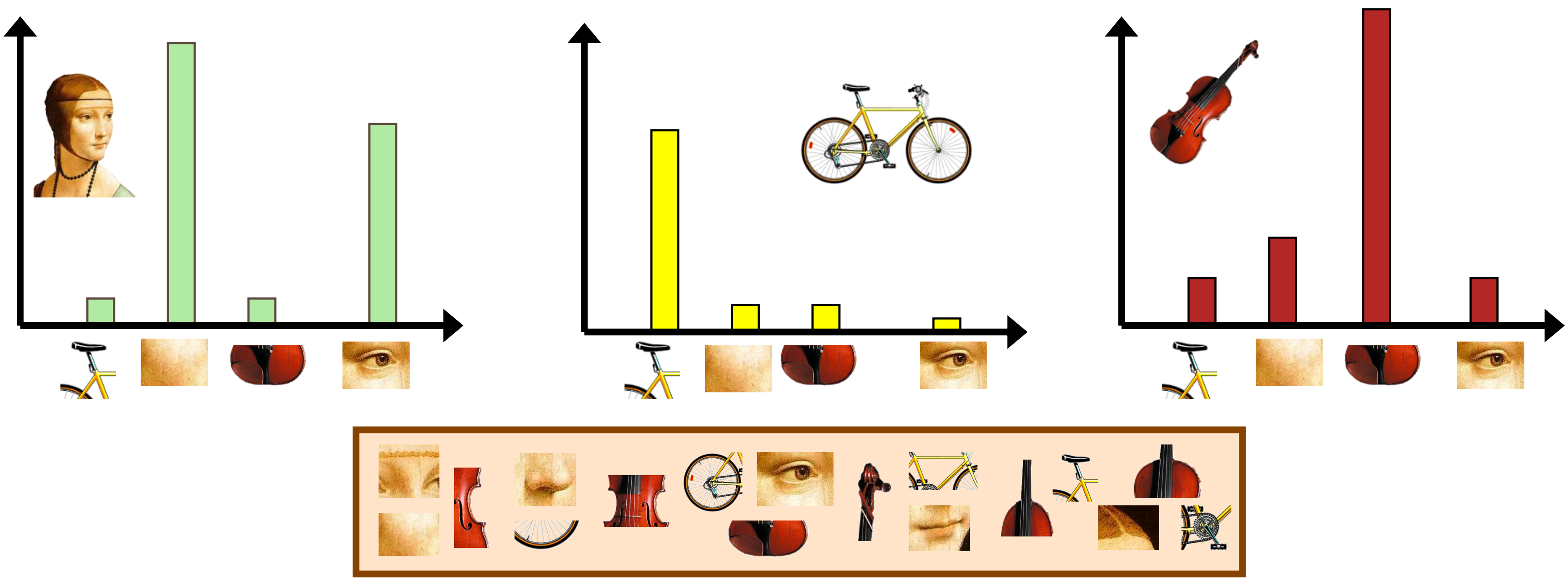
A clarification: definition of "BoW"

Looser definition

– Independent features

Stricter definition

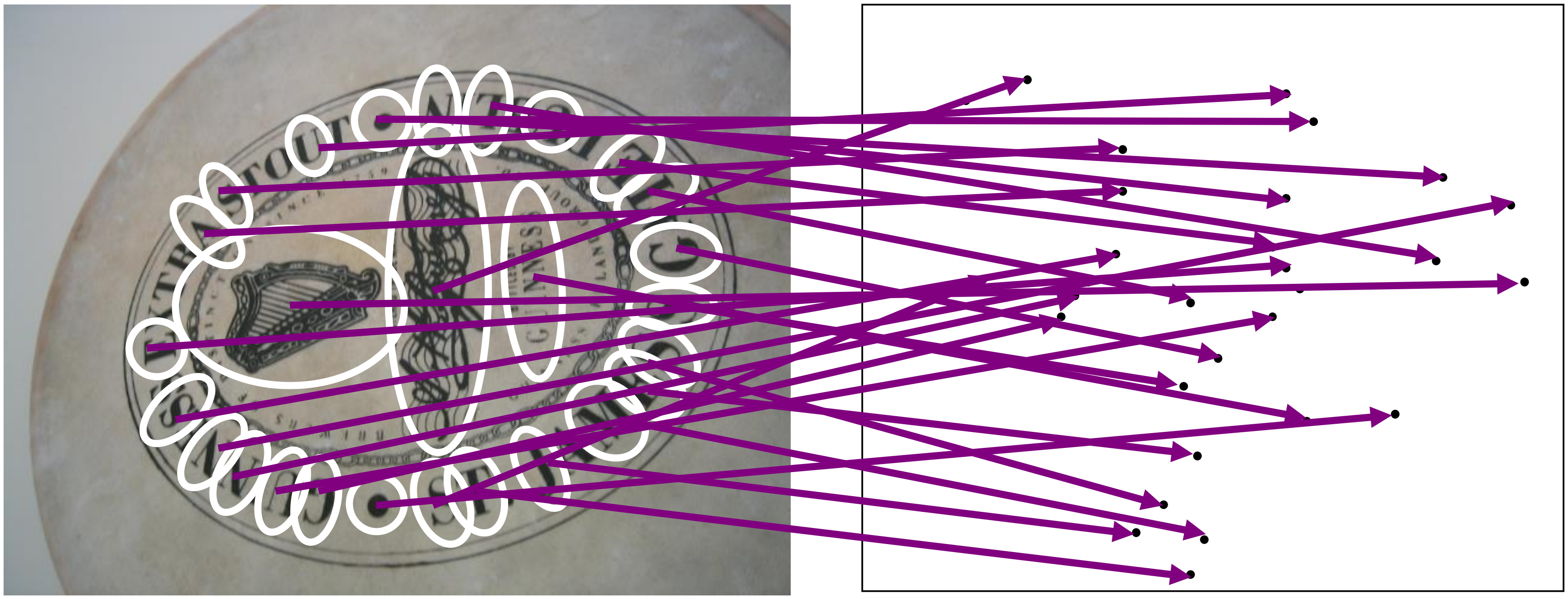
– Independent features
– histogram representation



Slide credit L. Fei-Fei

Visual words: main idea

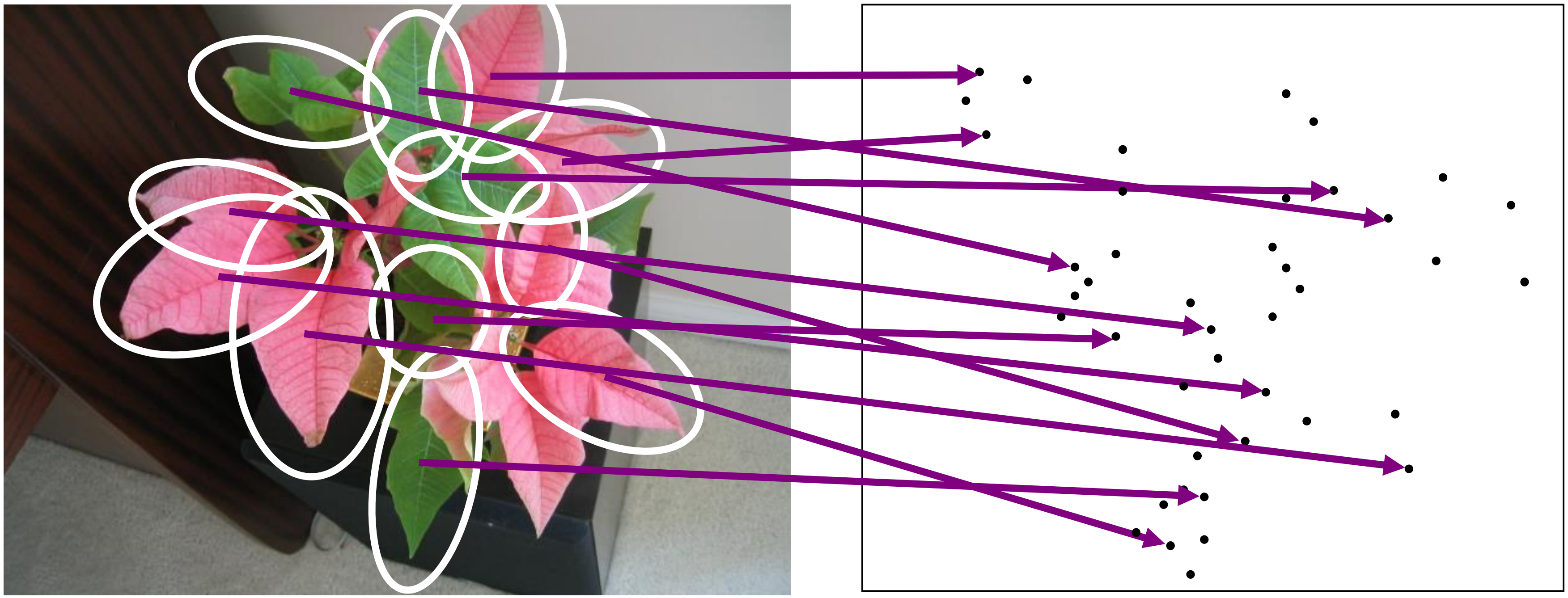
Extract some local features from a number of images …



e.g., SIFT descriptor space: each point is 128-dimensional

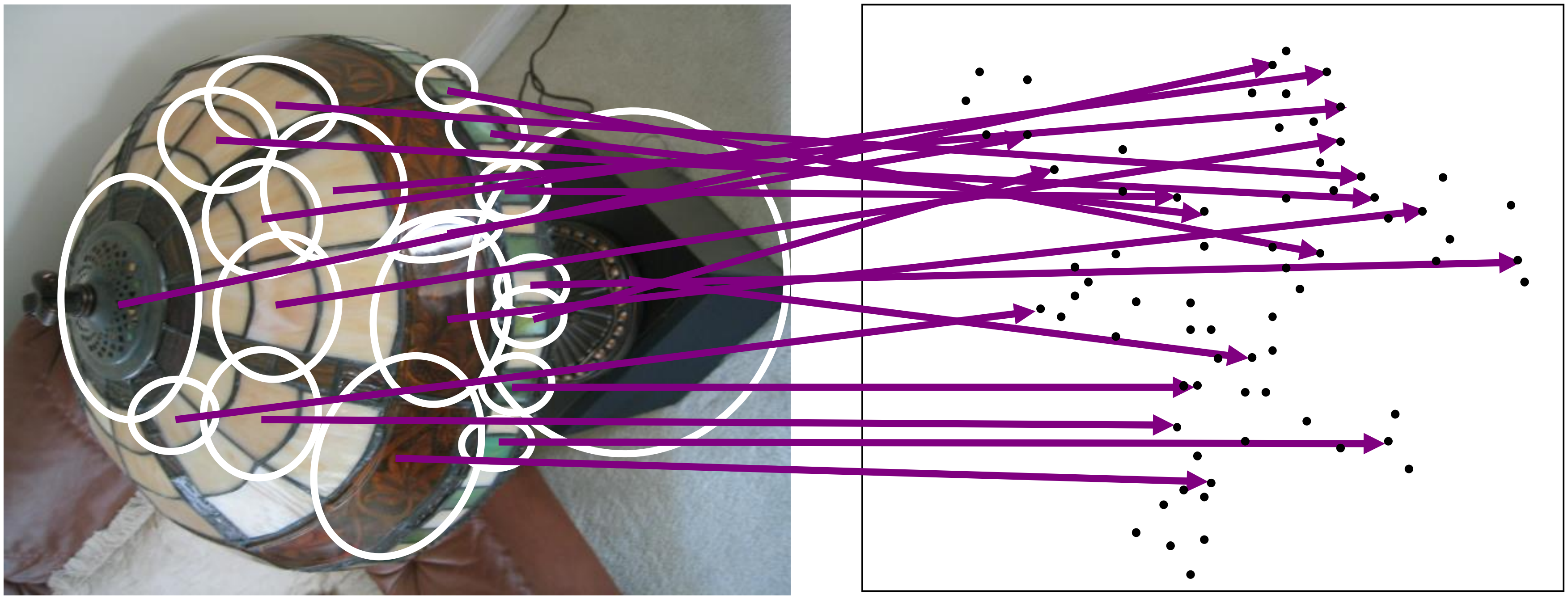
Slide credit: D. Nister

Visual words: main idea



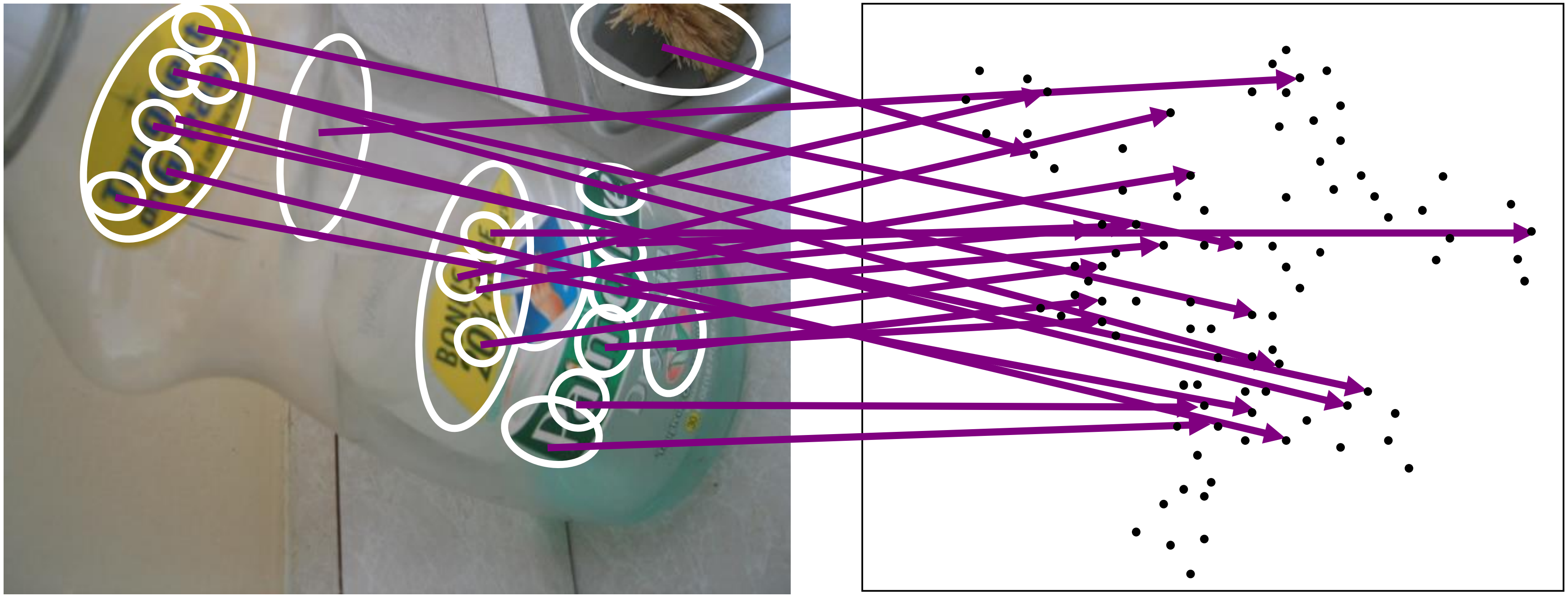
Slide credit: D. Nister

Visual words: main idea

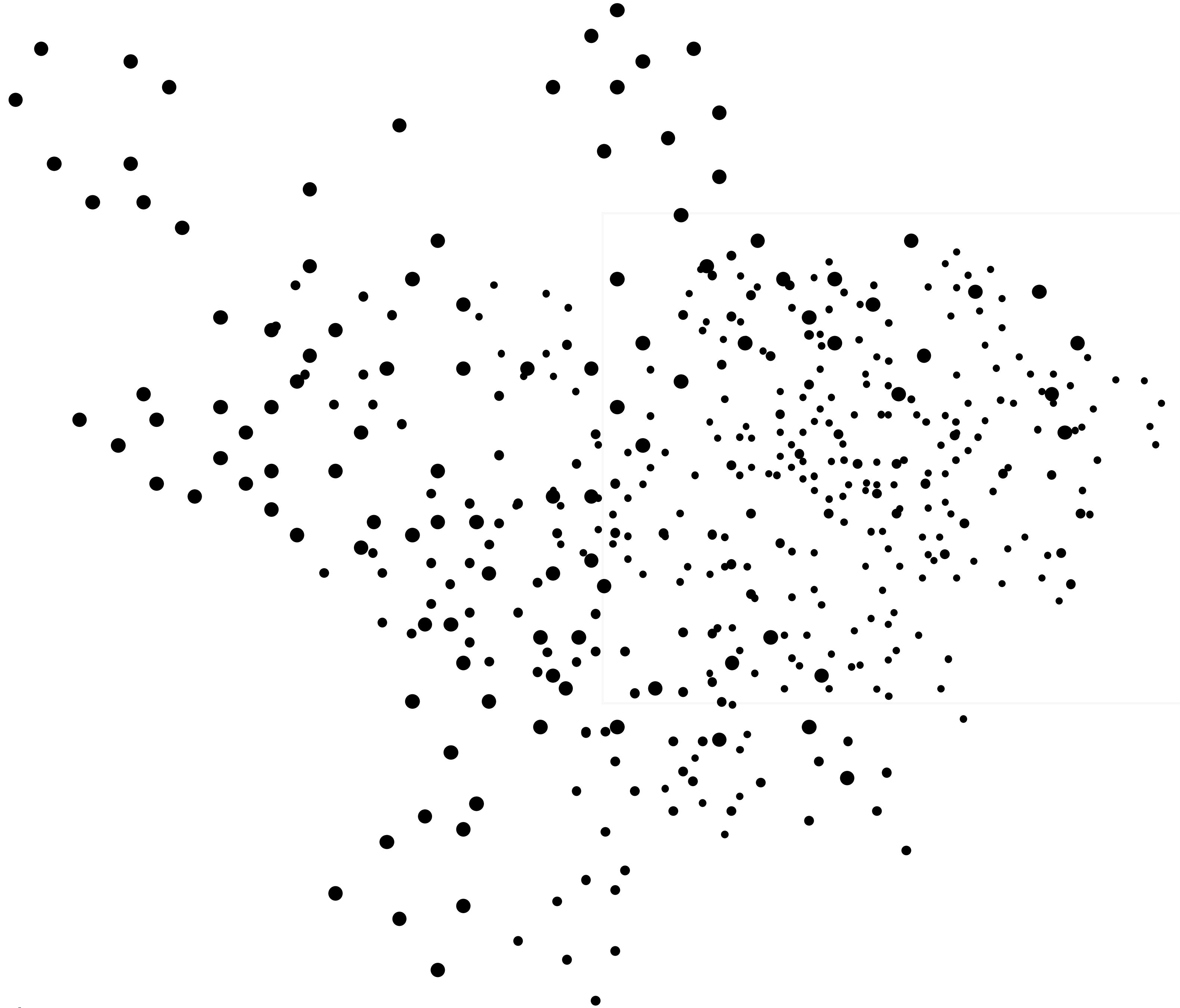


Slide credit: D. Nister

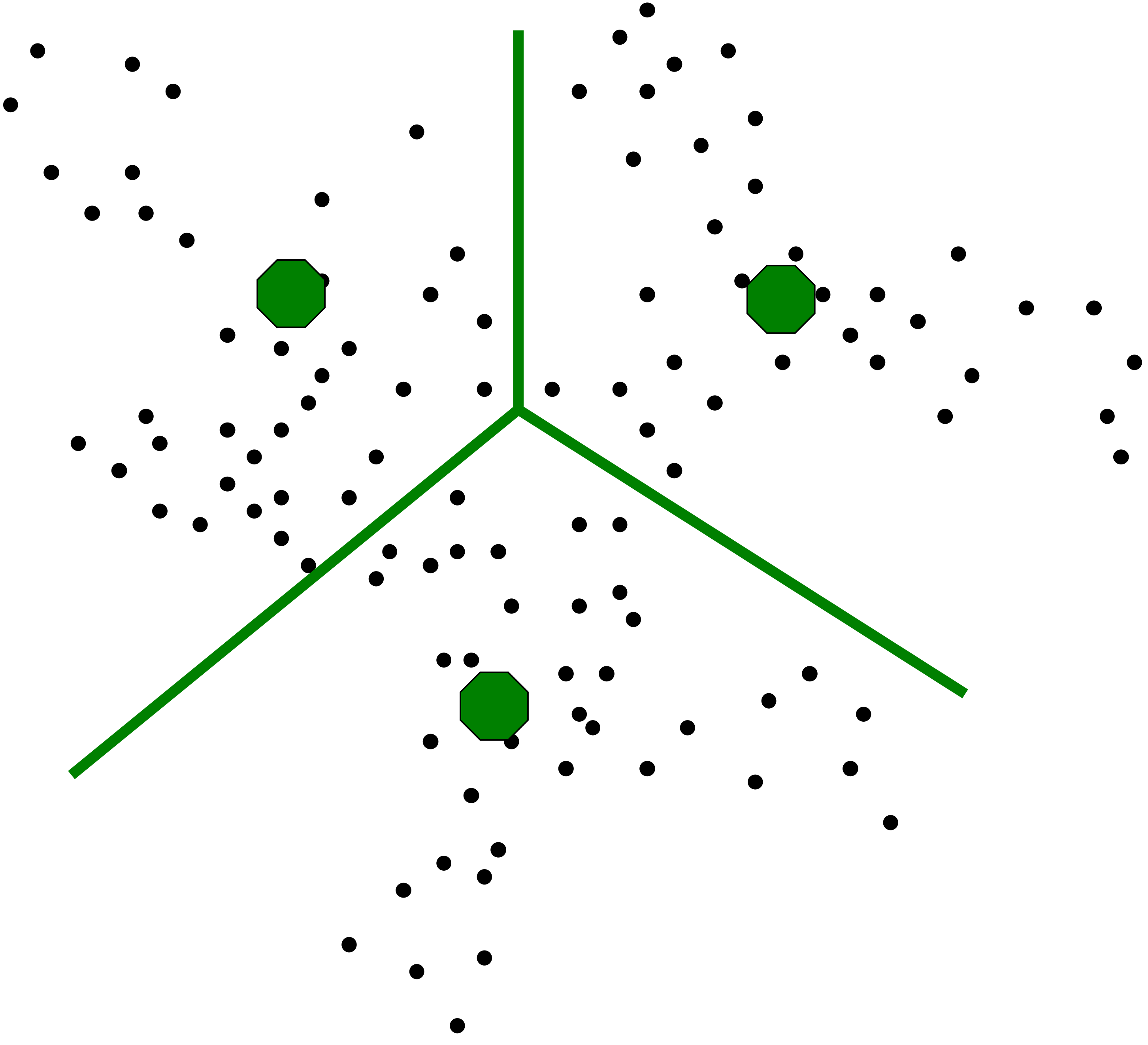
Visual words: main idea



Slide credit: D. Nister



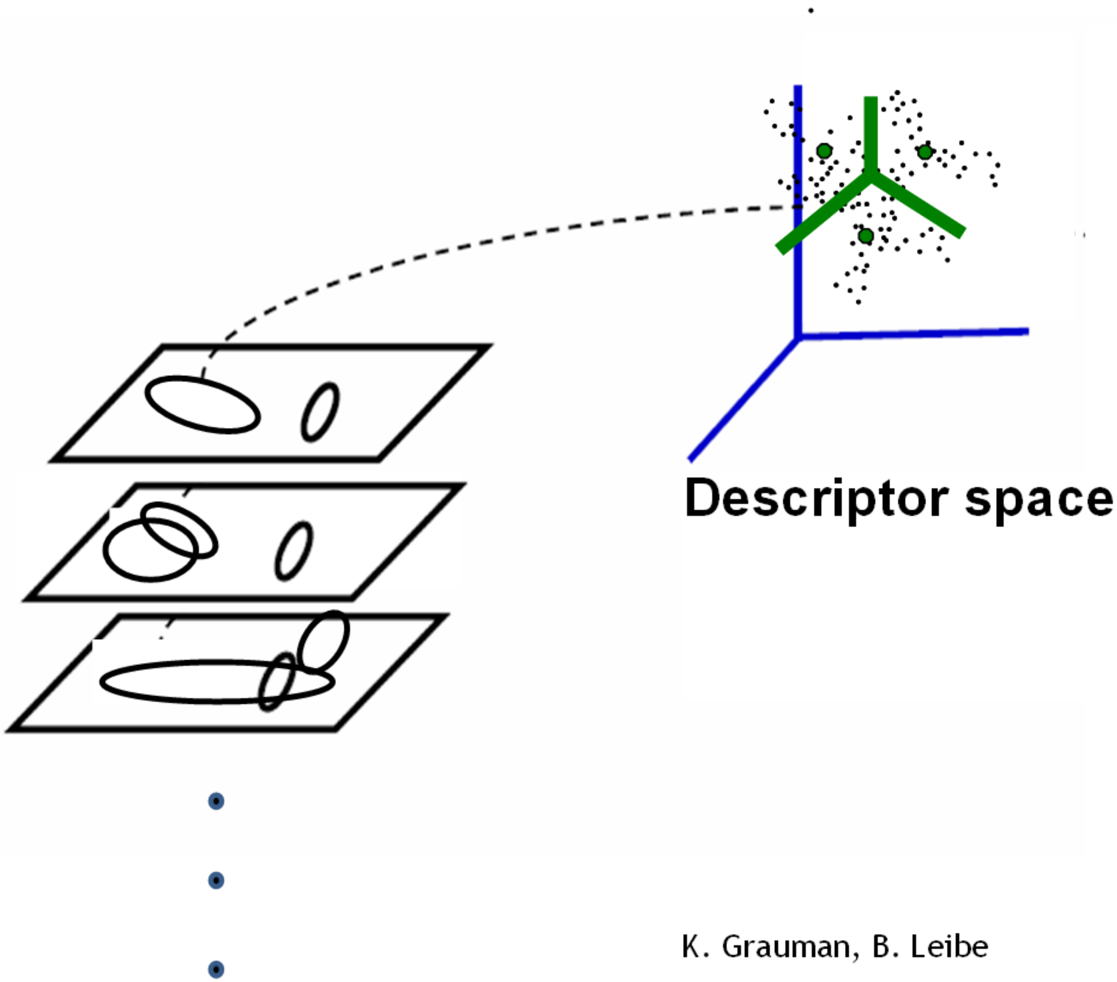
Slide credit: D. Nister



Slide credit: D. Nister

Visual words: main idea

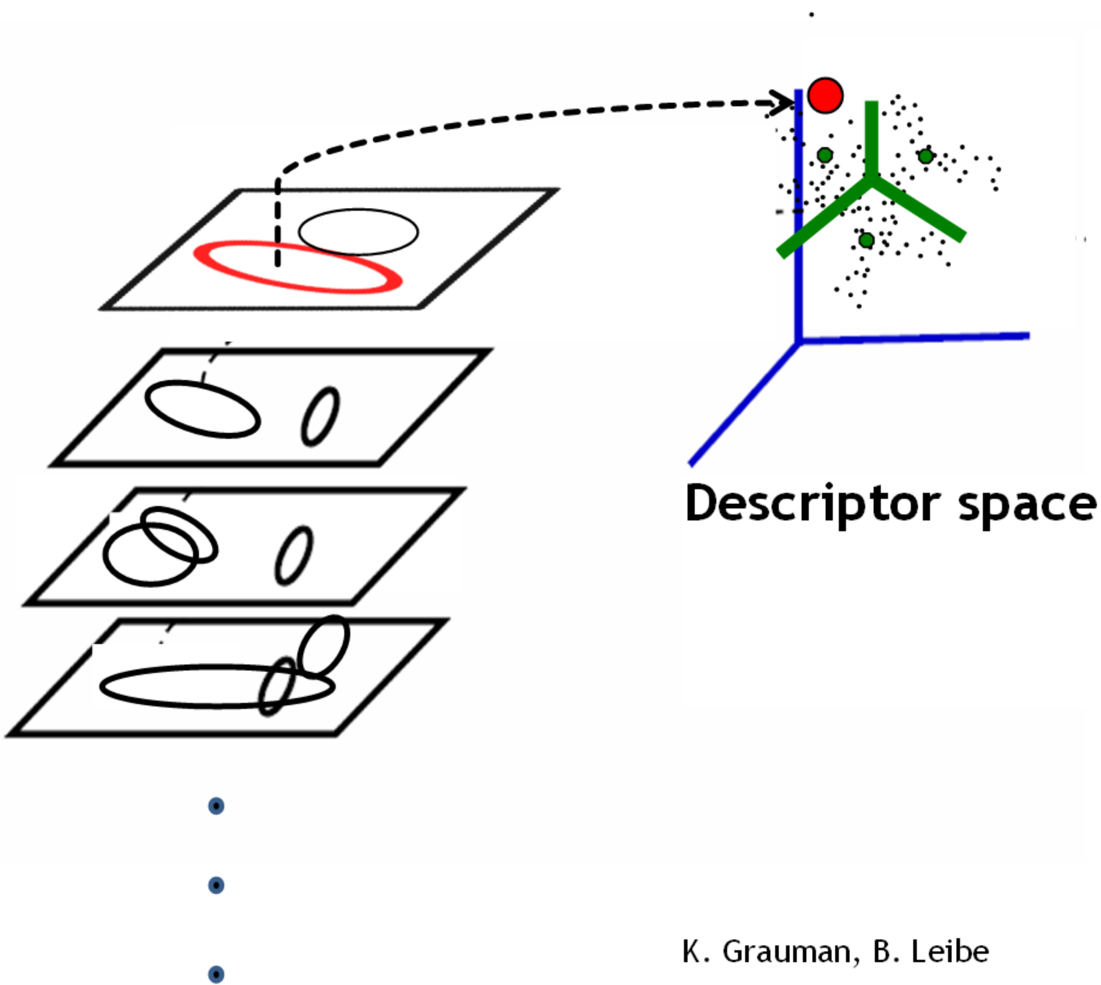
Map high-dimensional descriptors to tokens/words by quantizing the feature space



- Quantize via clustering, let cluster centers be the prototype “words”

Visual words: main idea

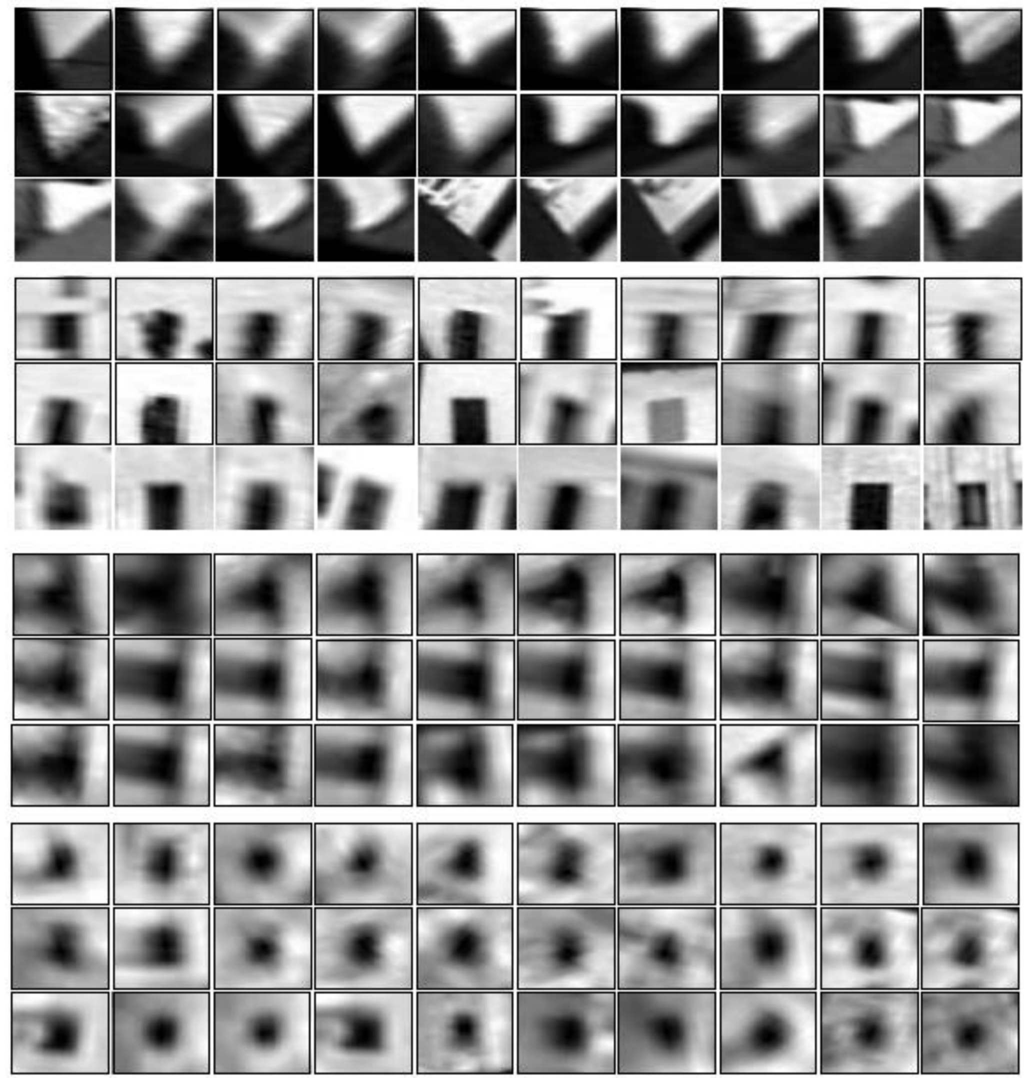
Map high-dimensional descriptors to tokens/words by quantizing the feature space



- Determine which word to assign to each new image region by finding the closest cluster center.

Visual words

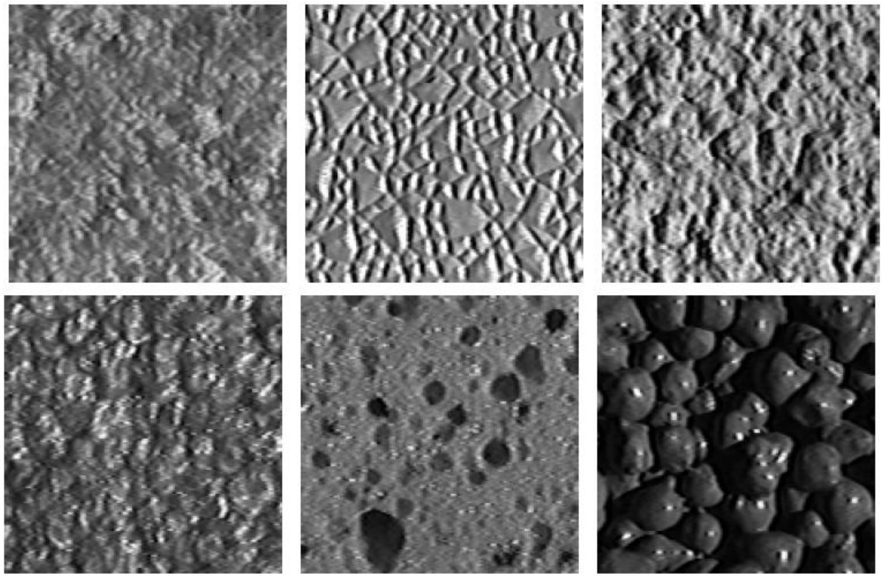
Example: each group of patches belongs to the same visual word



Slide credit: J. Sivic

Visual words

- First explored for texture and material representations
- *Texton* = cluster center of filter responses over collection of images
- Describe textures and materials based on distribution of prototypical texture elements.



Leung & Malik 1999; Varma & Zisserman, 2002; Lazebnik, Schmid & Ponce, 2003;

Slide credit: J. Sivic

Inverted file index for images comprised of visual words

frame #5

frame #10

Word number	List of image numbers
1	→ 5,10, ...
2	→ 10,...
...	...

- Score each image by the number of common visual words (tentative correspondences)
- But: does not take into account spatial layout of regions

Slide credit: J. Sivic

Clustering / quantization methods

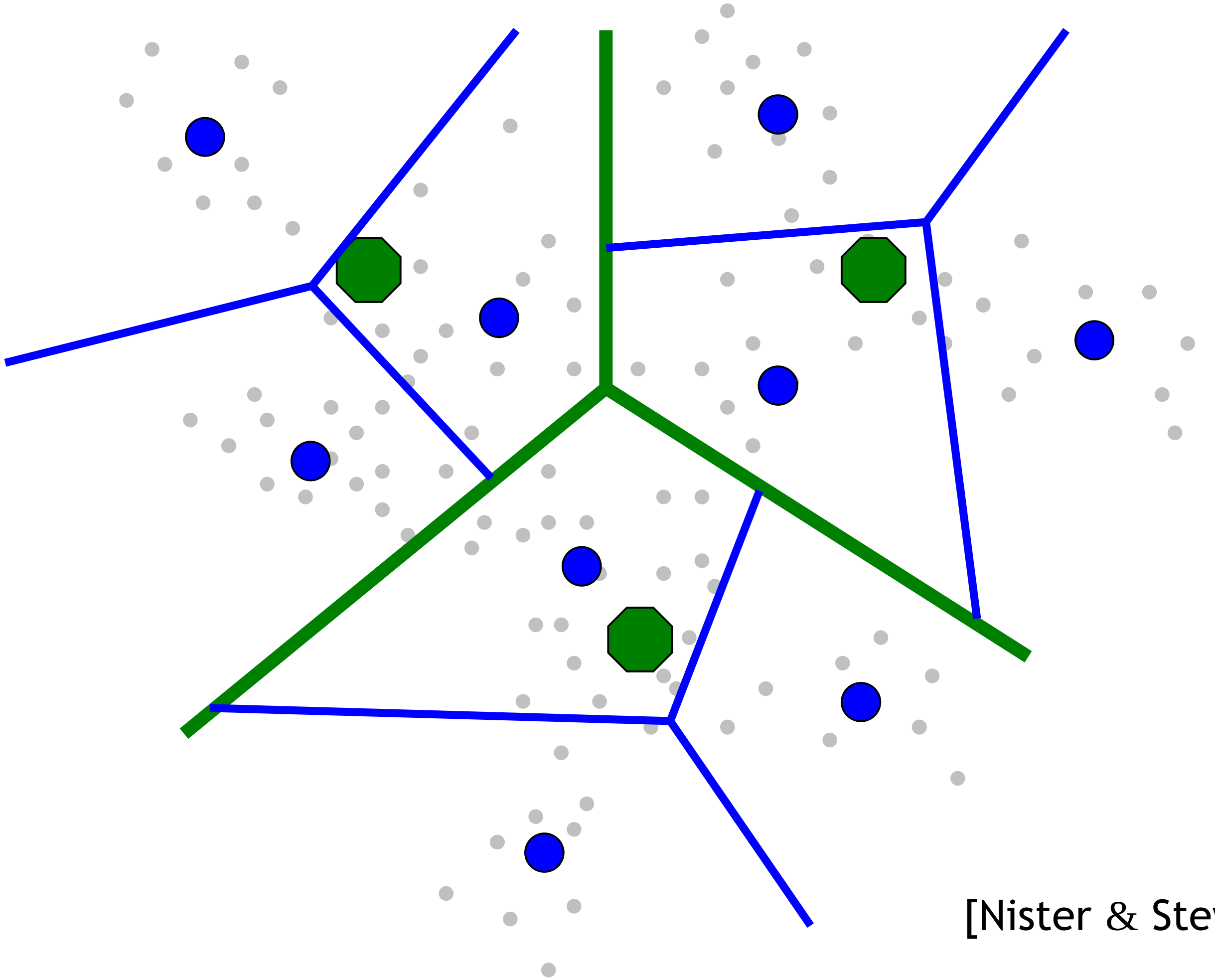
- k-means (typical choice), agglomerative clustering, mean-shift,…

- **Hierarchical clustering: allows faster insertion / word assignment while still allowing large vocabularies**
  - **Vocabulary tree [Nister & Stewenius, CVPR 2006]**

Slide credit: J. Sivic

Example: Recognition with Vocabulary Tree

Tree construction:

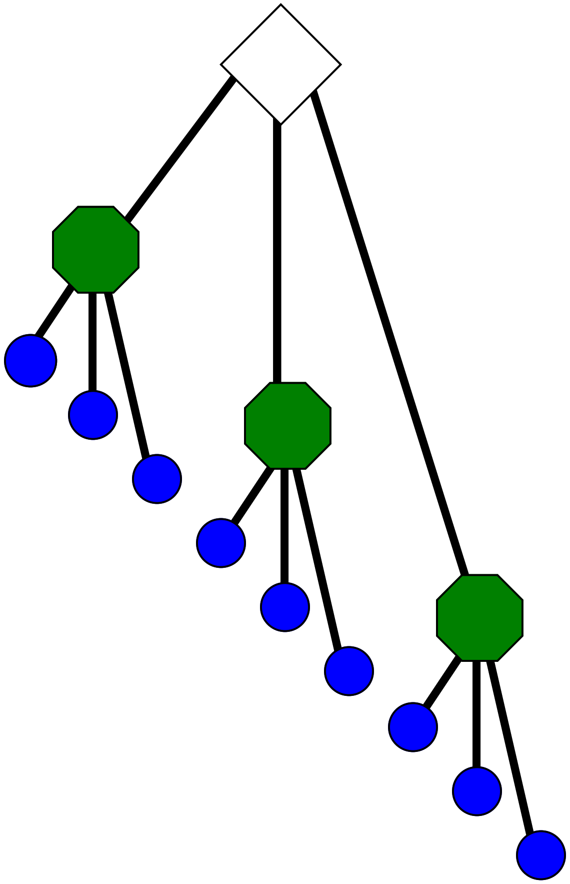


[Nister & Stewenius, CVPR'06]

Slide credit: David Nister

Vocabulary Tree

Training: Filling the tree

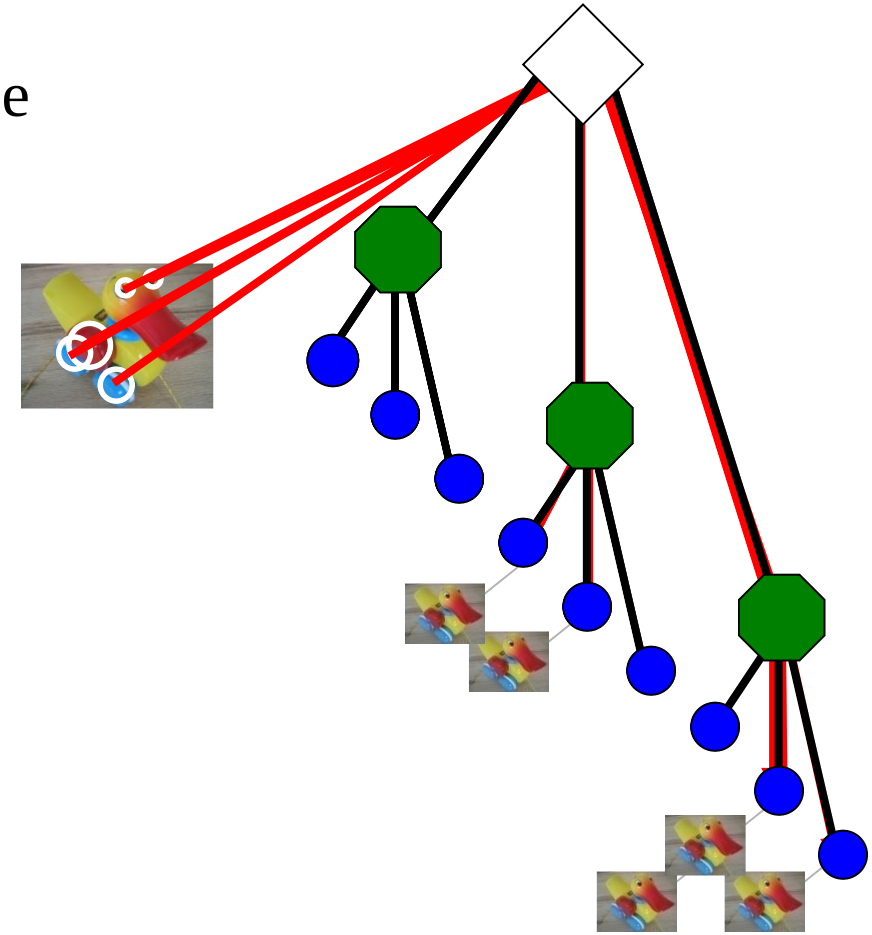


[Nister & Stewenius, CVPR'06]

Slide credit: David Nister

Vocabulary Tree

Training: Filling the tree

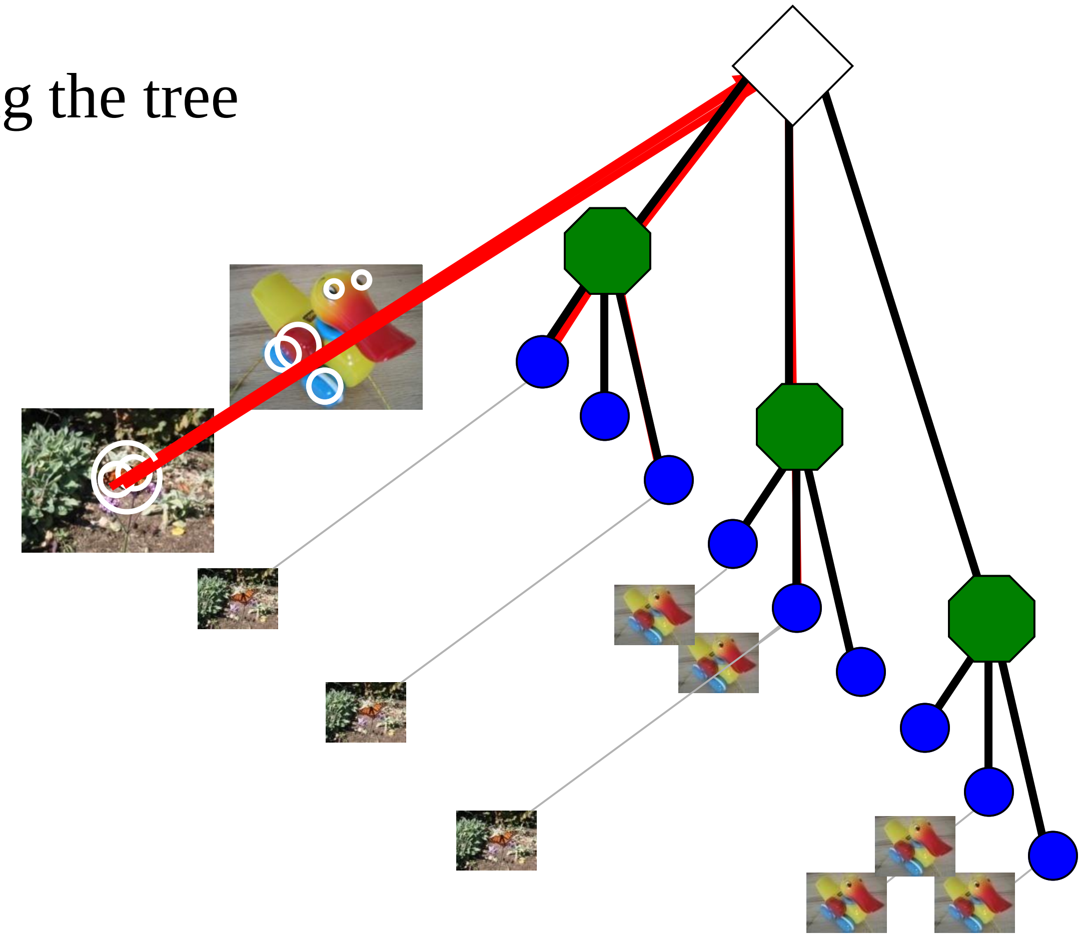


[Nister & Stewenius, CVPR'06]

Slide credit: David Nister

Vocabulary Tree

Training: Filling the tree

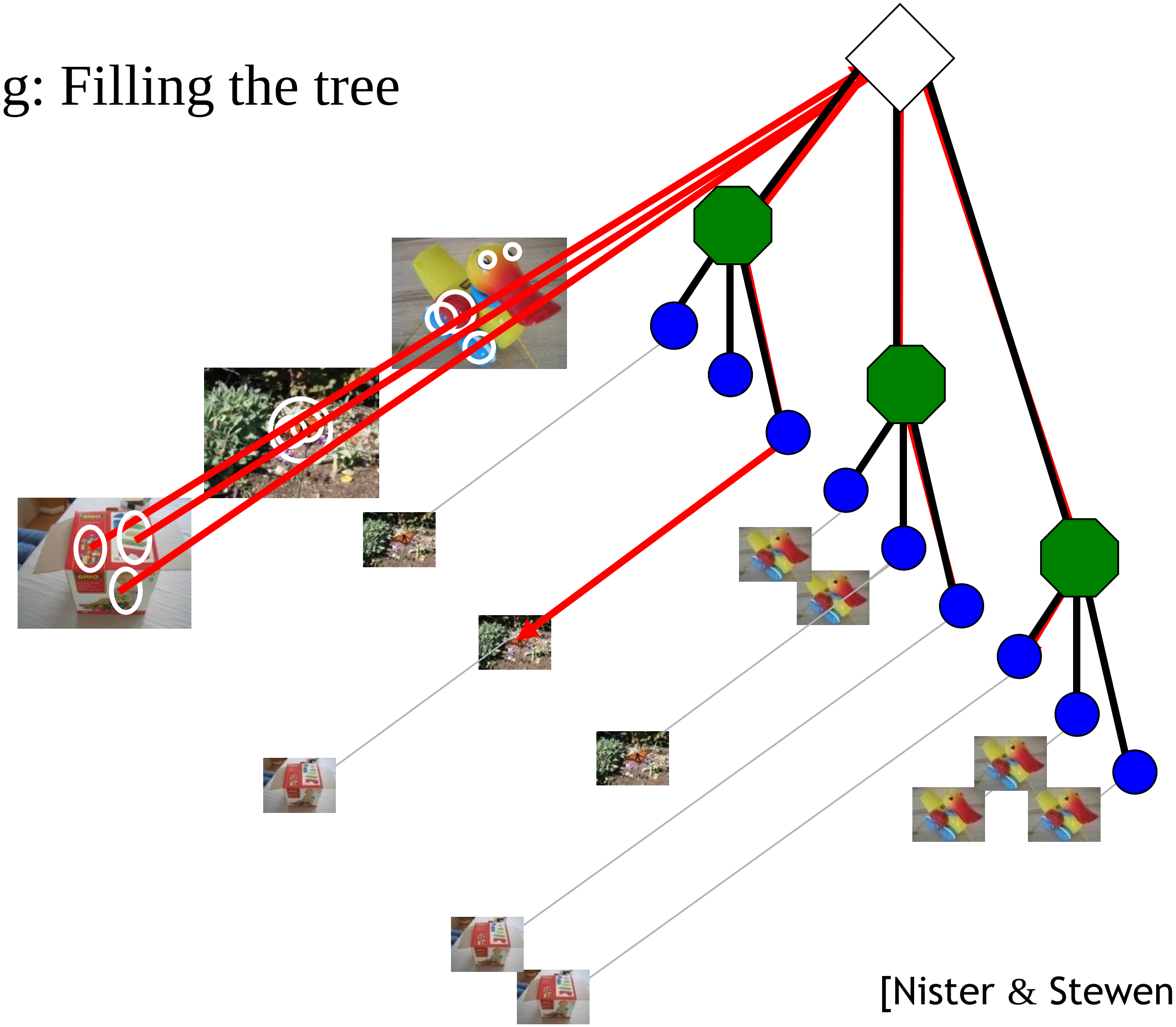


[Nister & Stewenius, CVPR'06]

Slide credit: David Nister

Vocabulary Tree

Training: Filling the tree

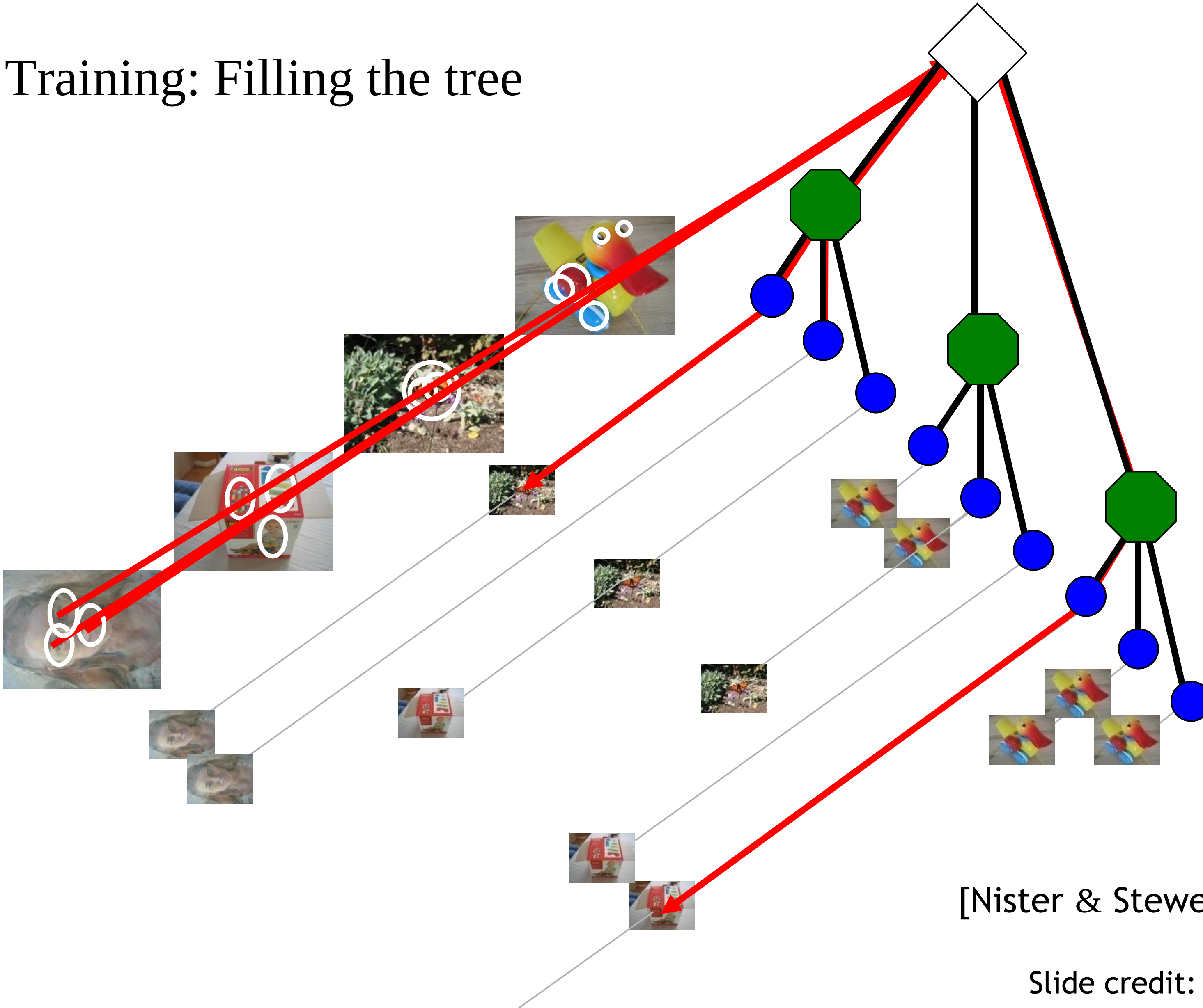


[Nister & Stewenius, CVPR'06]

Slide credit: David Nister

Vocabulary Tree

Training: Filling the tree



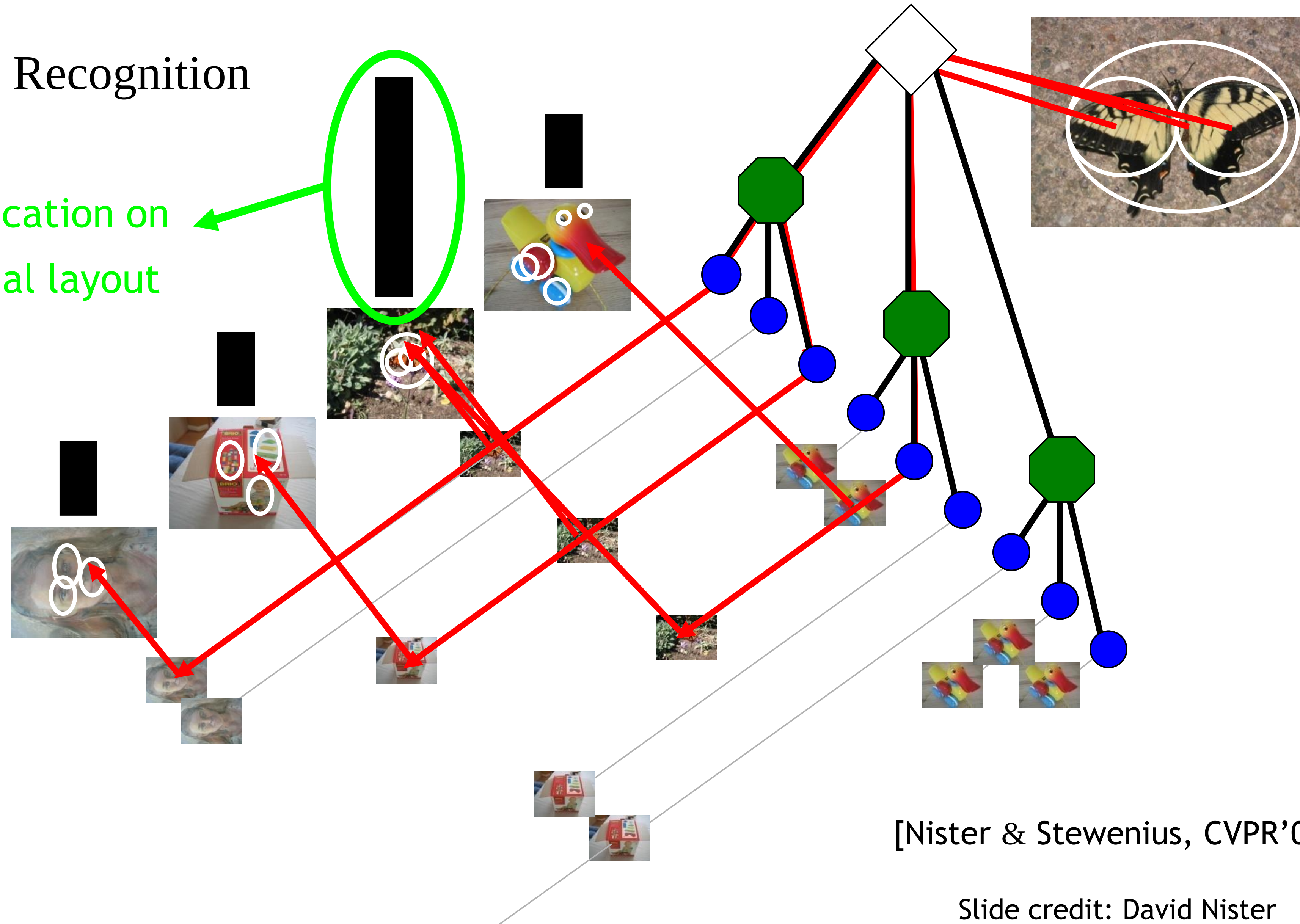
[Nister & Stewenius, CVPR'06]

Slide credit: David Nister

Vocabulary Tree

Recognition

Verification on spatial layout



[Nister & Stewenius, CVPR’06]

Slide credit: David Nister

Vocabulary Tree: Performance

Evaluated on large databases

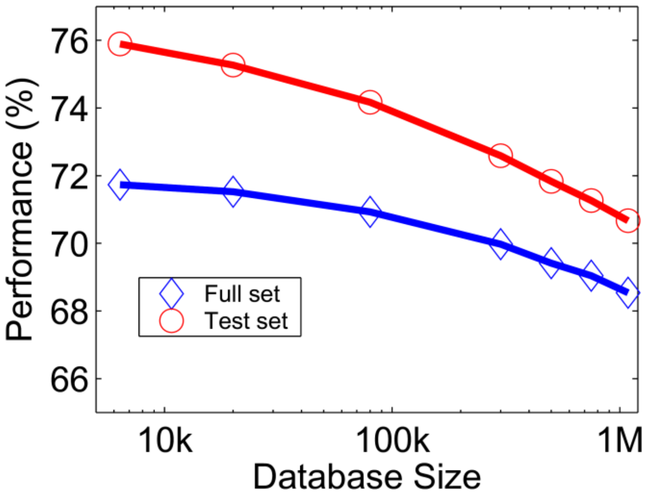
- Indexing with up to 1M images

Online recognition for database of 50,000 CD covers

- Retrieval in ~1s

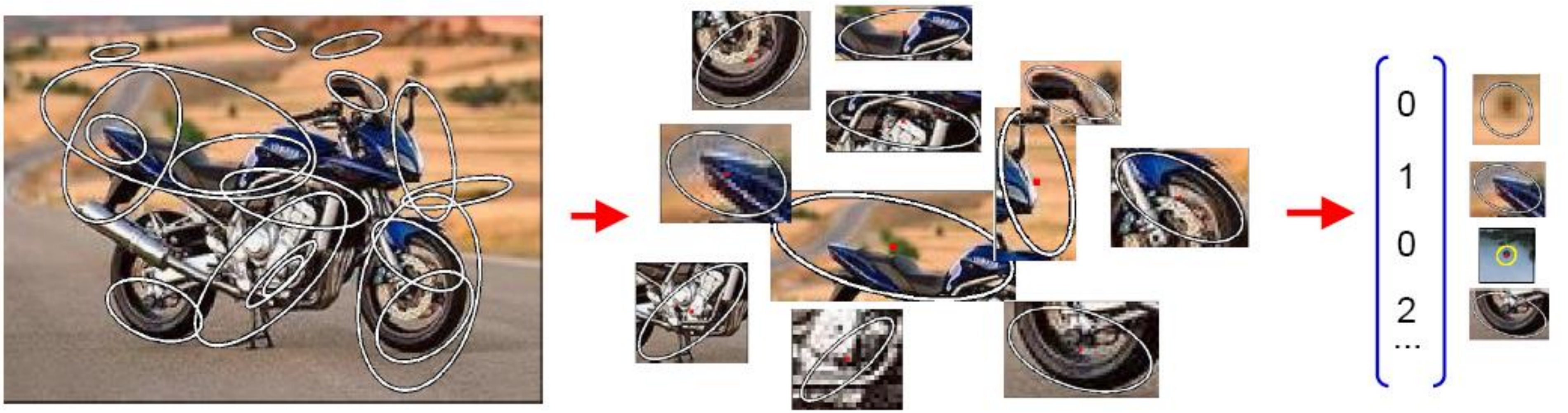
Find experimentally that large vocabularies can be beneficial for recognition

[Nister & Stewenius, CVPR'06]



Slide credit: J. Sivic

“Bag of visual words”



Slide credit: J. Sivic