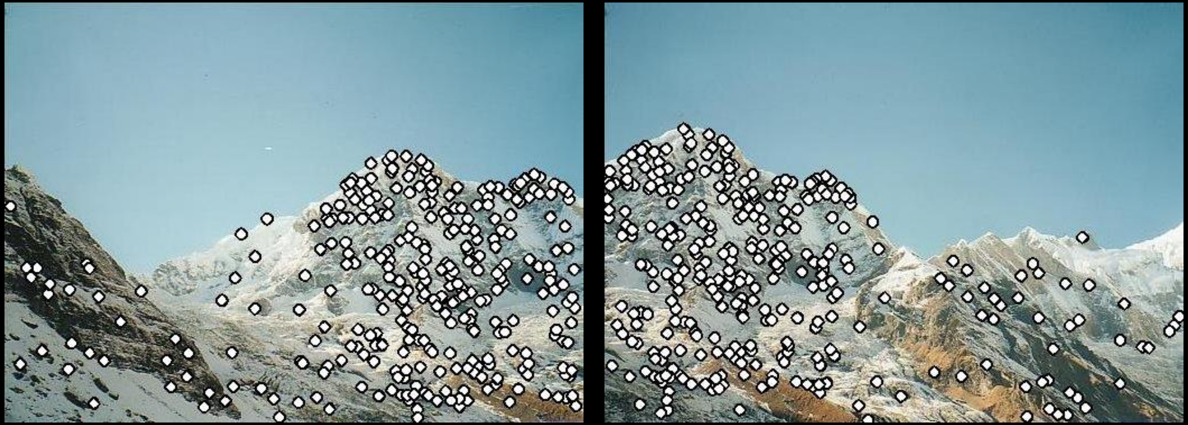
CS4495/6495
Introduction to Computer Vision

4B-L2 *Matching feature points (a little)*

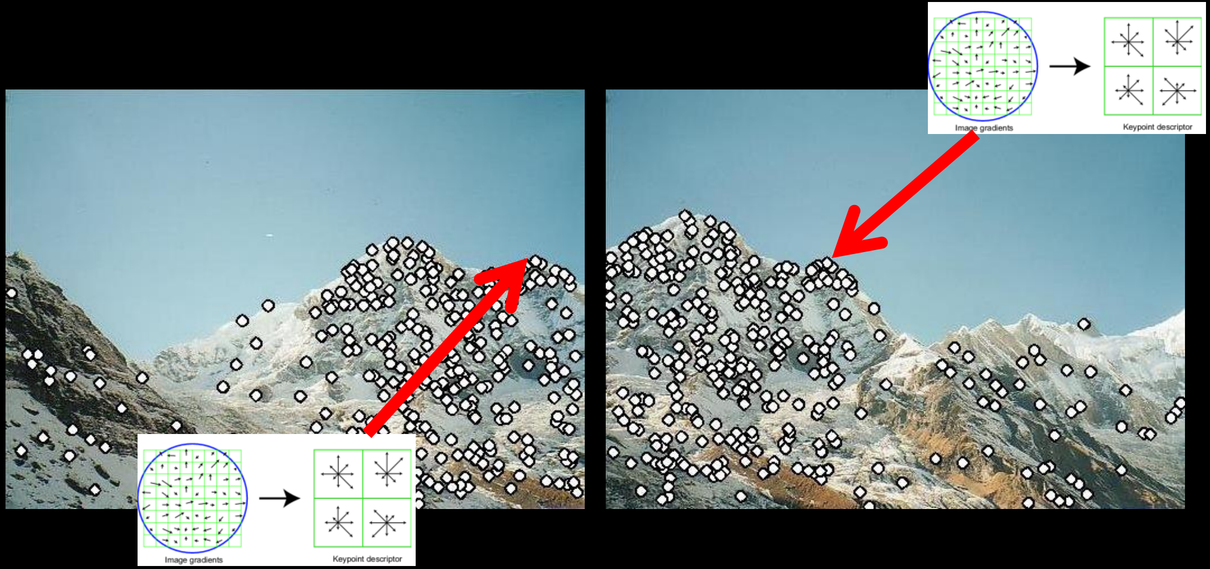
Feature Points

- We know how to detect points



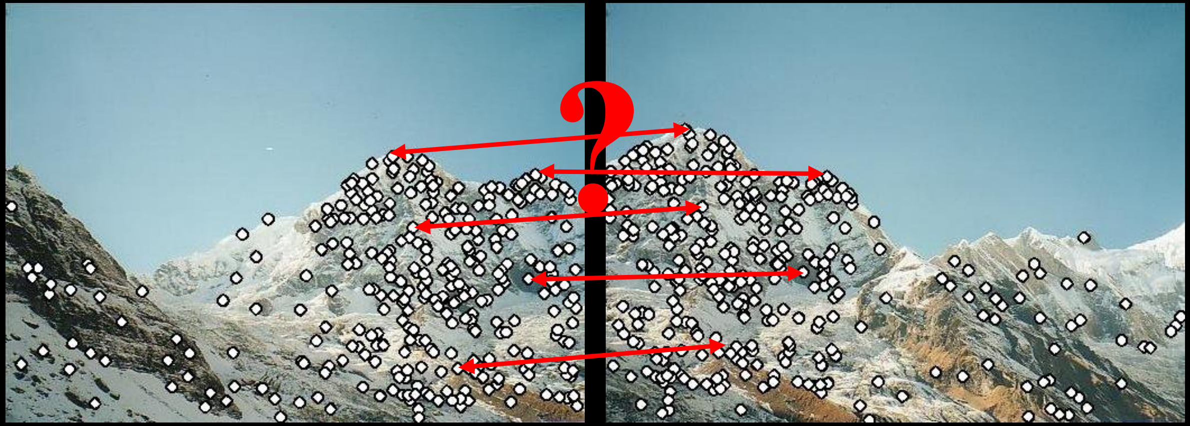
Feature Points

- We know how to describe them



Feature Points

- Next question: How to match them?



How to match feature points?

- Could just do nearest-neighbor search
  - [OMS students: You will!]

- But that’s really expensive…SIFT tests have 10,000’s of points!

Nearest-neighbor matching to feature database

- Better: Hypotheses are generated by *approximate nearest neighbor* matching of each feature to vectors in the database
  - SIFT uses best-bin-first (Beis & Lowe, 97) modification to k-d tree algorithm
  - Use heap data structure to identify bins in order by their distance from query point

Nearest-neighbor matching to feature database

- Result: Can give speedup by factor of *100-1000* while finding nearest neighbor (of interest) 95% of the time

Nearest neighbor techniques

- *k*-D tree and
- Best Bin First (BBF)

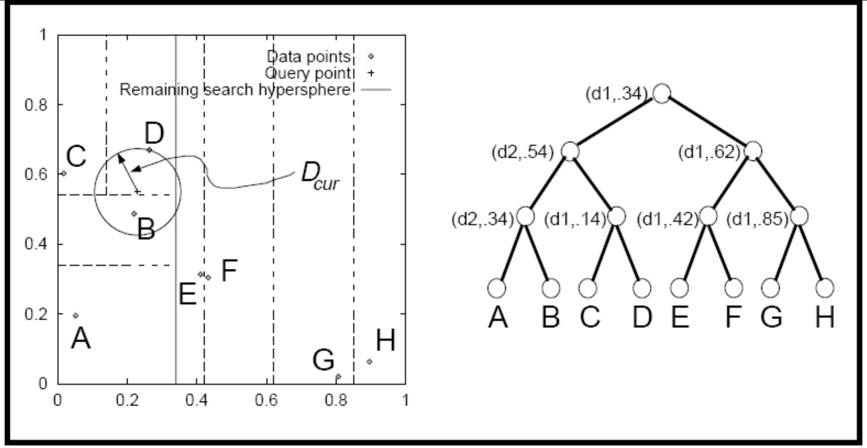
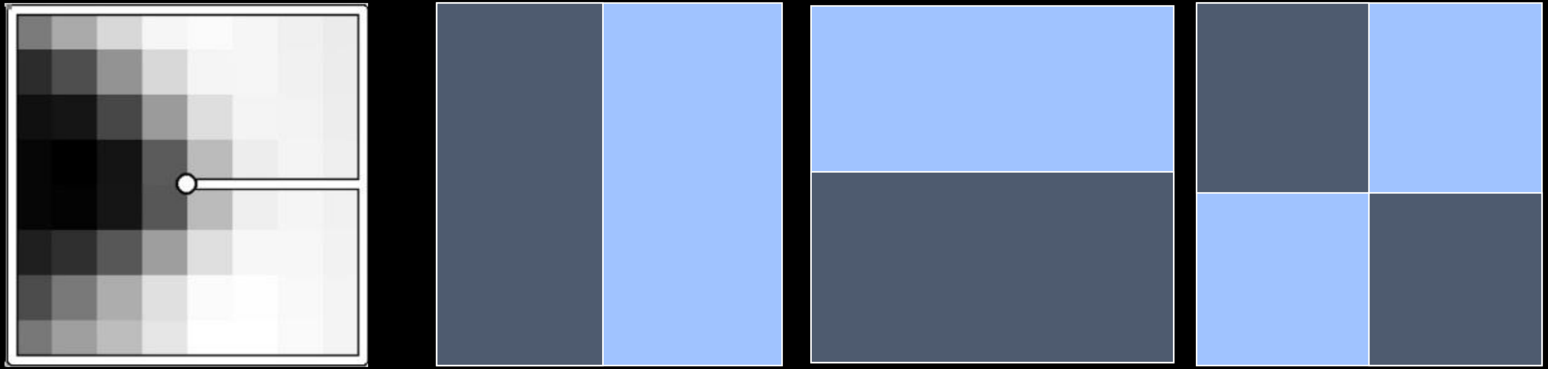


Figure 6: kd-tree with 8 data points labelled A-H, dimension of space k=2. On the right is the full tree, the leaf nodes containing the data points. Internal node information consists of the dimension of the cut plane and the value of the cut in that dimension. On the left is the 2D feature space carved into various sizes and shapes of bin, according to the distribution of the data points. The two representations are isomorphic. The situation shown on the left is after initial tree traversal to locate the bin for query point "+" (contains point D). In standard search, the closest nodes in the tree are examined first (starting at C). In BBF search, the closest bins to query point q are examined first (starting at B). The latter is more likely to maximize the overlap of (i) the hypersphere centered on q with radius D_{cur}, and (ii) the hyperrectangle of the bin to be searched. In this case, BBF search reduces the number of leaves to examine, since once point B is discovered, all other branches can be pruned.

Indexing Without Invariants in 3D Object Recognition, Beis and Lowe, PAMI'99

Wavelet-based hashing

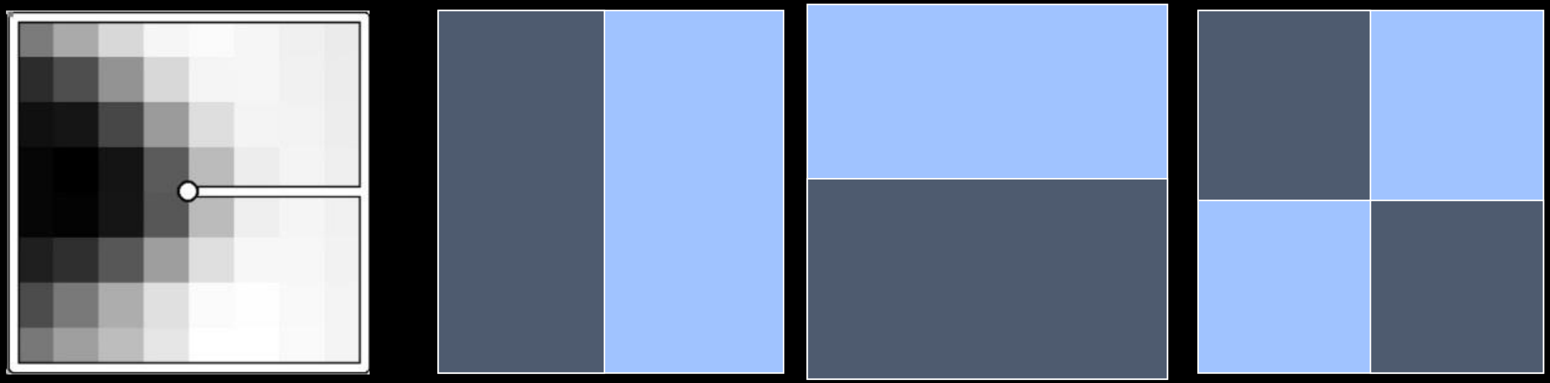
Compute a short (3-vector) descriptor from the neighborhood using a Haar "wavelet"



[Brown, Szeliski, Winder, CVPR'2005]

Wavelet-based hashing

Quantize each value into 10 (overlapping) bins (10^3 total entries)

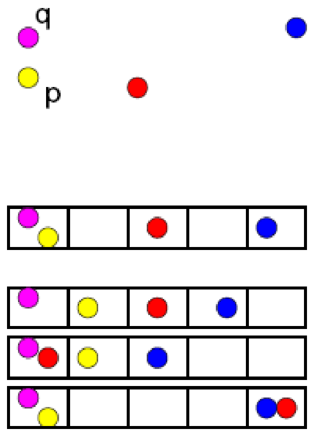


[Brown, Szeliski, Winder, CVPR’2005]

Locality sensitive hashing

[Indyk-Motwani'98]

- Idea: construct hash functions $g: R^d \rightarrow U$ such that for any points p,q:
  - If $D(p,q) \leq r$, then $Pr[g(p)=g(q)]$ is ~~“high”~~ “not-so-small”
  - If $D(p,q) > cr$, then $Pr[g(p)=g(q)]$ is “small”
- Then we can solve the problem by hashing



Kulis & Grauman, “Kernelized Locality-Sensitive Hashing for Scalable Image Search” *ICCV*, 2009.

3D Object Recognition

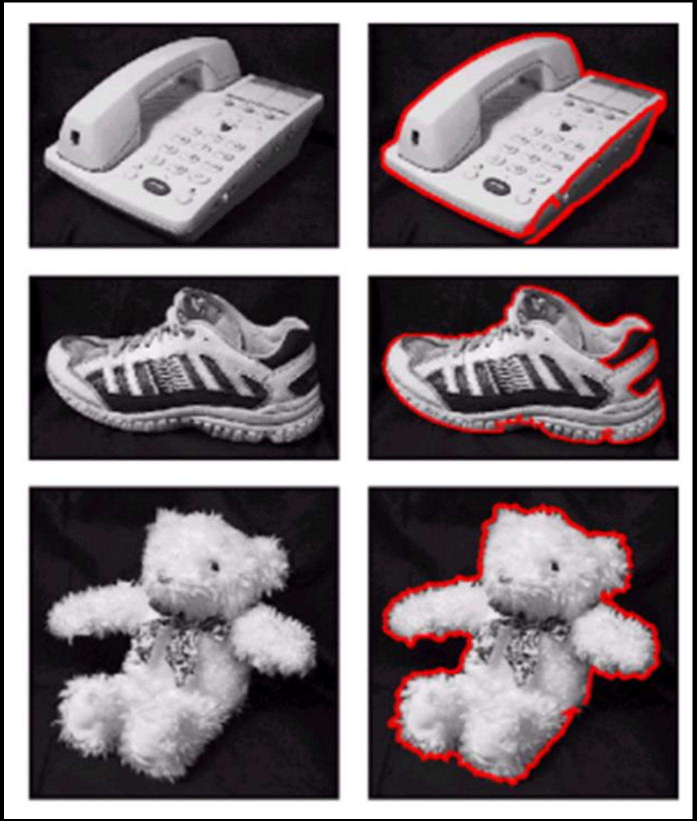
Train:

1. Extract outlines with background subtraction
2. Compute "keypoints" – interest points and descriptors.

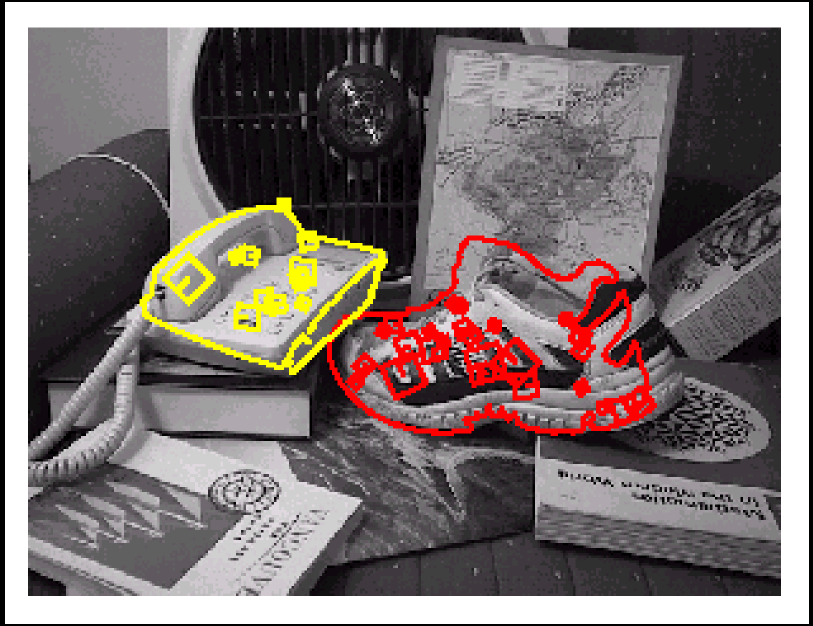
3D Object Recognition

Test:

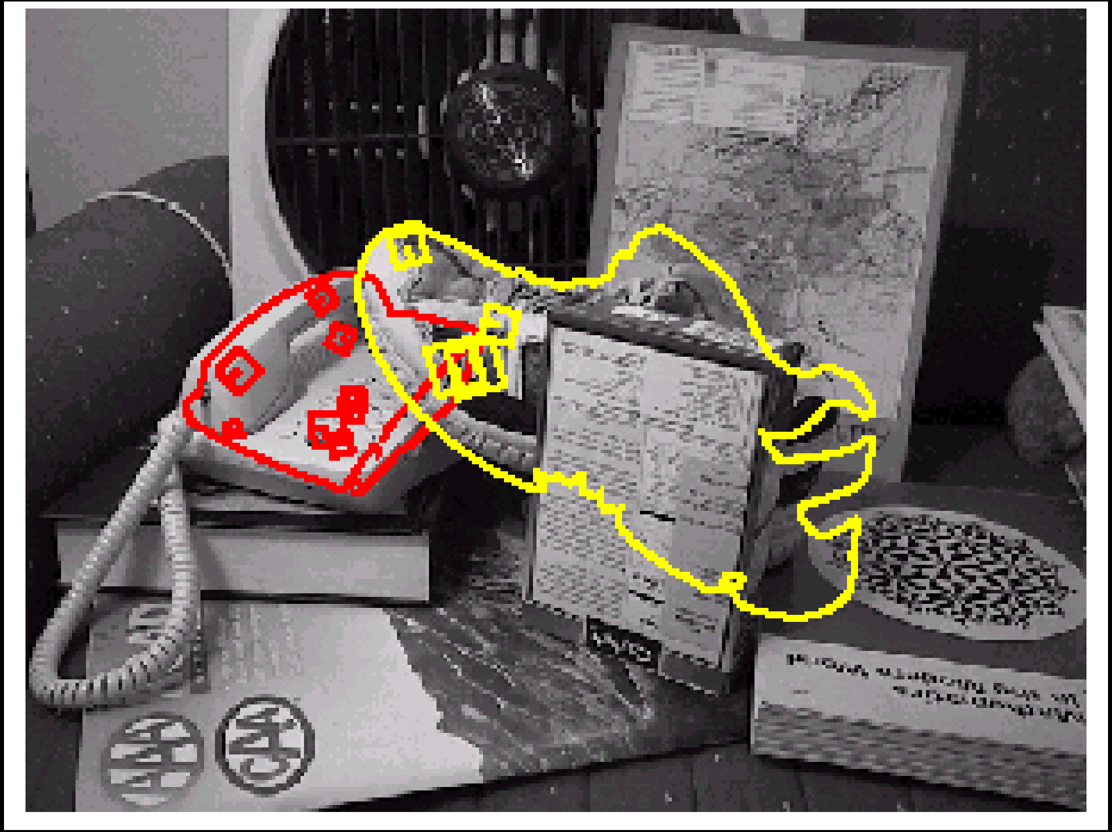
1. Find possible matches.
2. Search for consistent solution – such as *affine*. *(How many points?!?!?)*



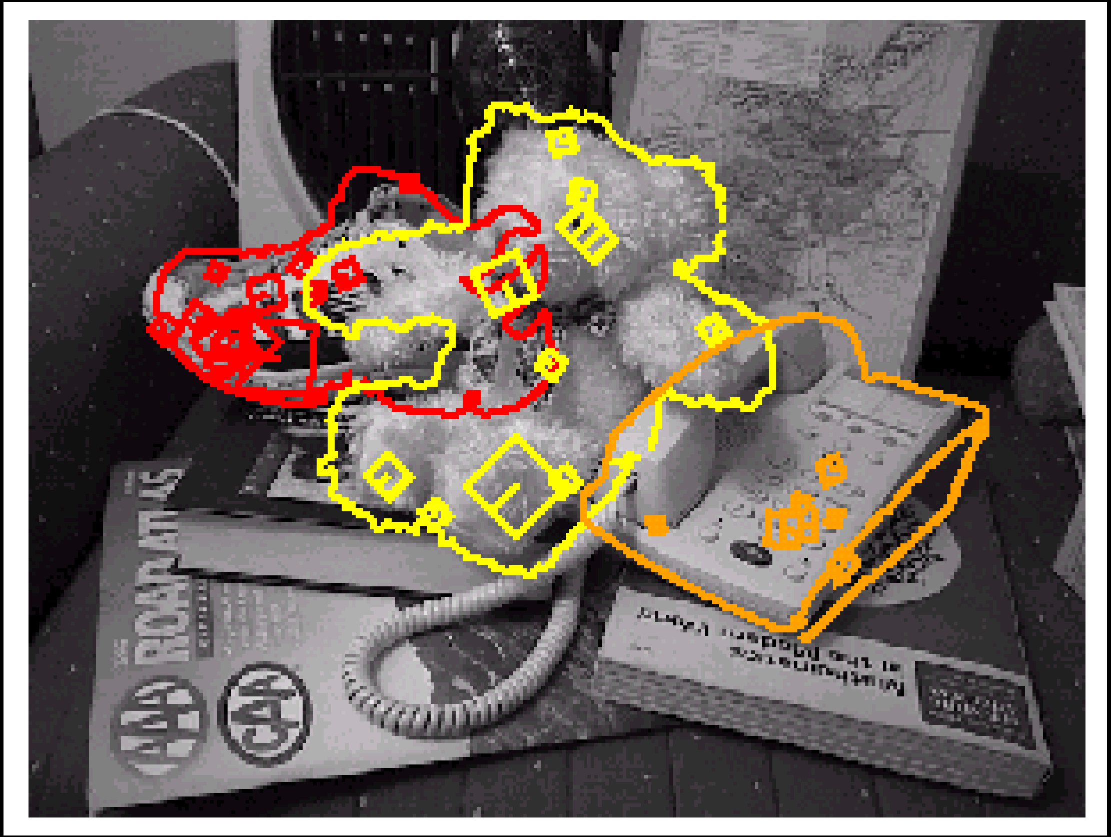
Results



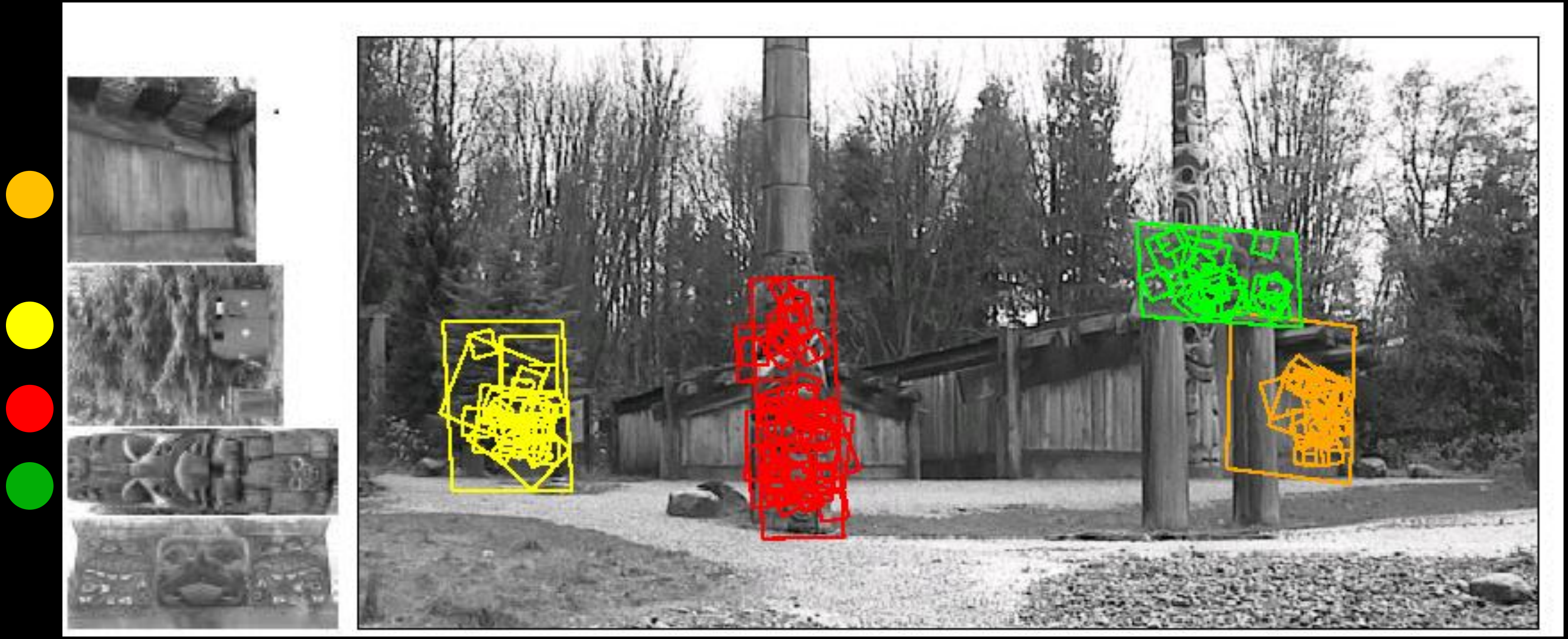
Recognition under occlusion



Recognition under occlusion



Locating object pieces



(From last lesson)

SIFT in Sony Aibo (Evolution Robotics)

SIFT usage:

- Recognize charging station
- Communicate with visual cards