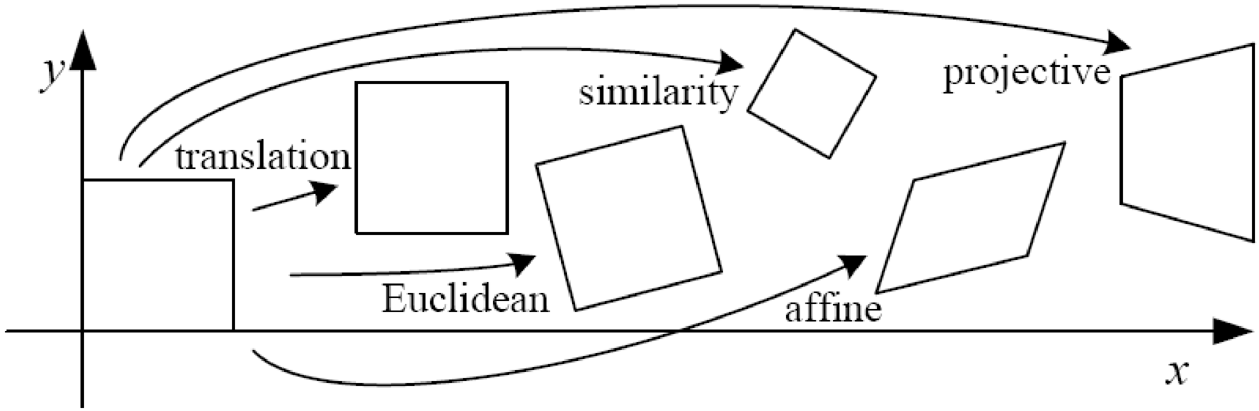
Object Instance Recognition

Slides by Derek Hoiem

Today’s class

- Object instance recognition
- Example of alignment-based category recognition

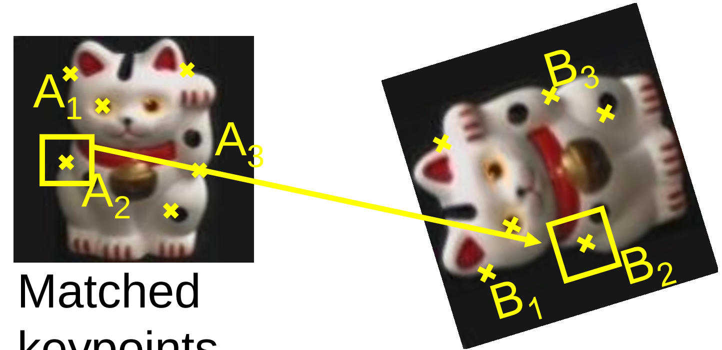
Recall: 2D image transformations



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	8	straight lines	

Object Instance Recognition

1. Match keypoints to object model
2. Solve for affine transformation parameters
3. Score by inliers and choose solutions with score above threshold



Matched keypoints

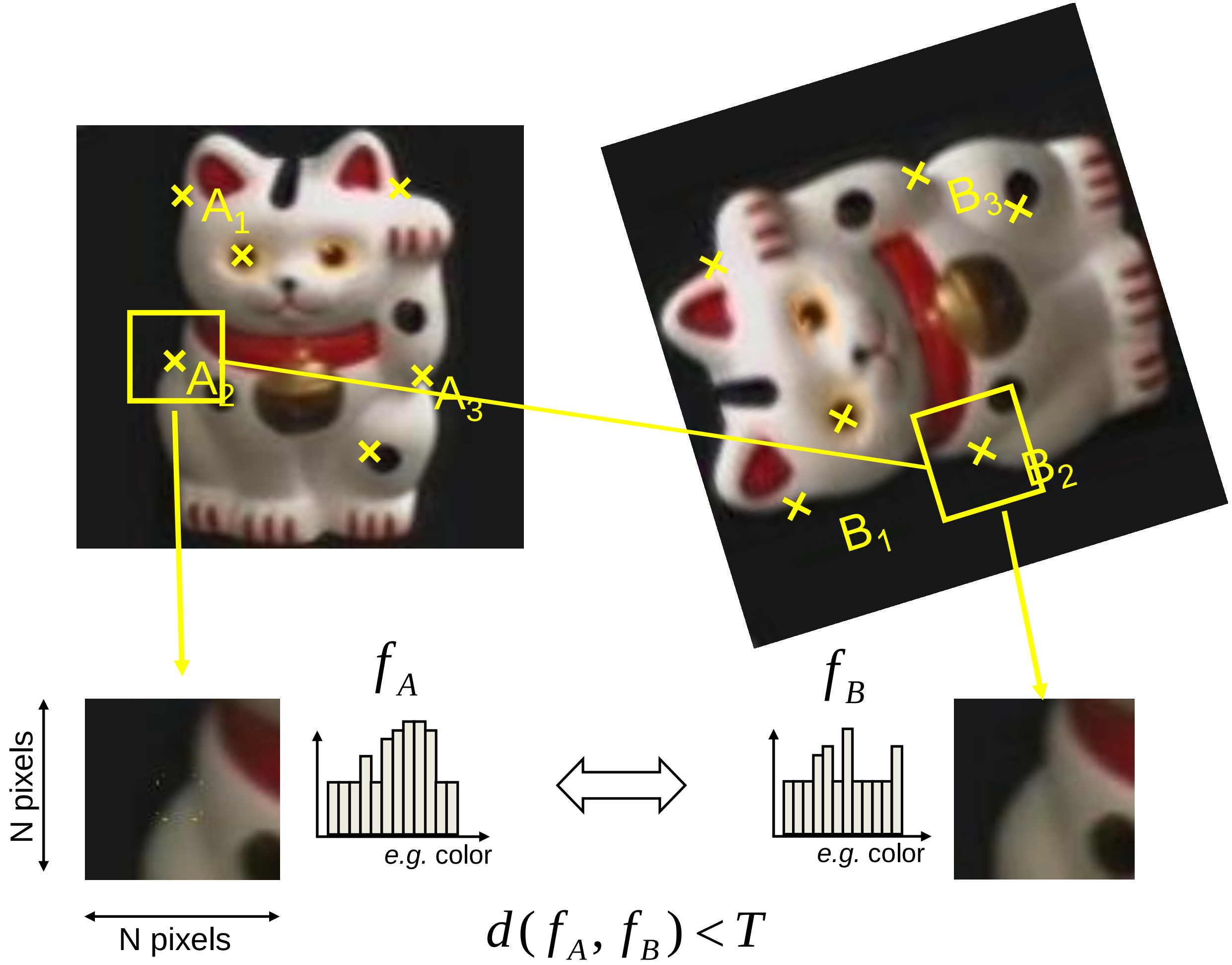
Affine Parameters

Inliers

Choose hypothesis with max score above threshold

This Class

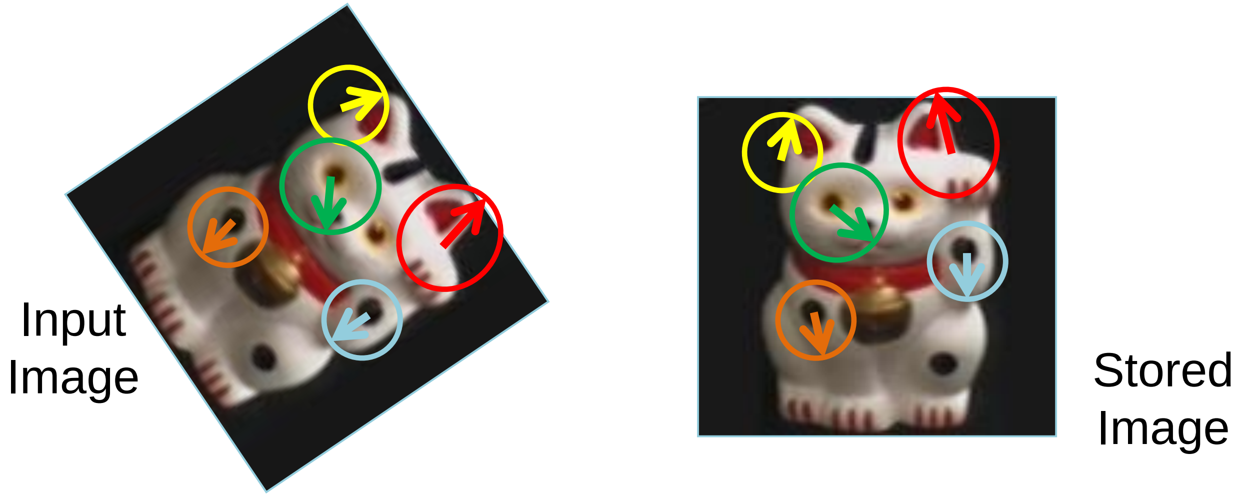
Overview of Keypoint Matching



1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**
4. **Compute a local descriptor from the normalized region**
5. **Match local descriptors**

K. Grauman, B. Leibe

Finding the objects (overview)



1. Match interest points from input image to database image
2. Matched points vote for rough position/orientation/scale of object
3. Find position/orientation/scales that have at least three votes
4. Compute affine registration and matches using iterative least squares with outlier check
5. Report object if there are at least T matched points

Matching Keypoints

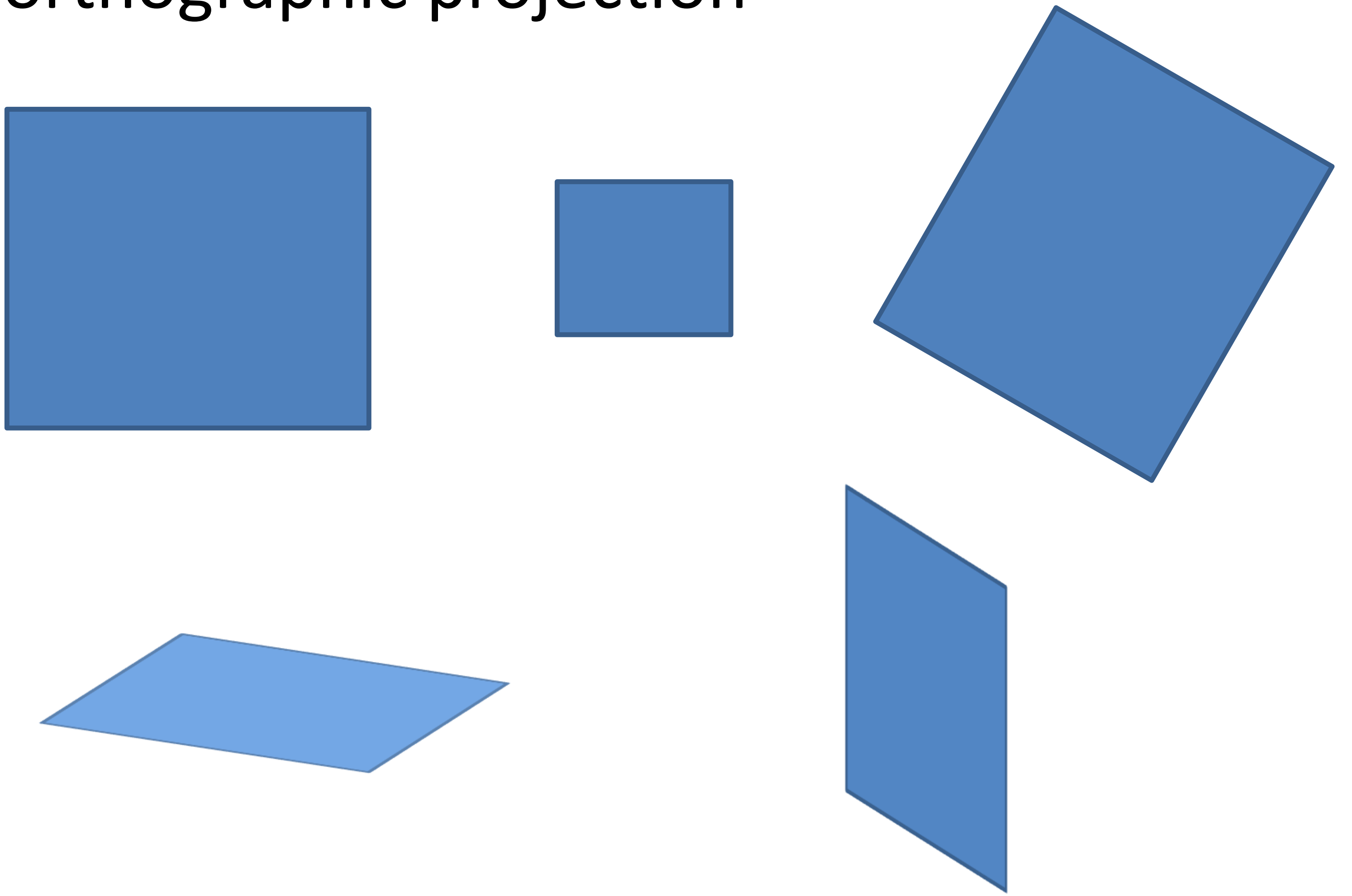
- Want to match keypoints between:
    1. Query image
    2. Stored image containing the object

- Given descriptor x_0, find two nearest neighbors x_1, x_2 with distances d_1, d_2

- x_1 matches x_0 if $d_1/d_2 < 0.8$
    - This gets rid of 90% false matches, 5% of true matches in Lowe's study

Affine Object Model

- Accounts for 3D rotation of a surface under orthographic projection



Affine Object Model

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ & & \vdots & & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ \vdots \end{bmatrix}$$

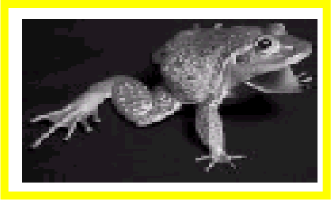
$$\mathbf{x} = [\mathbf{A}^{\mathrm{T}}\mathbf{A}]^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

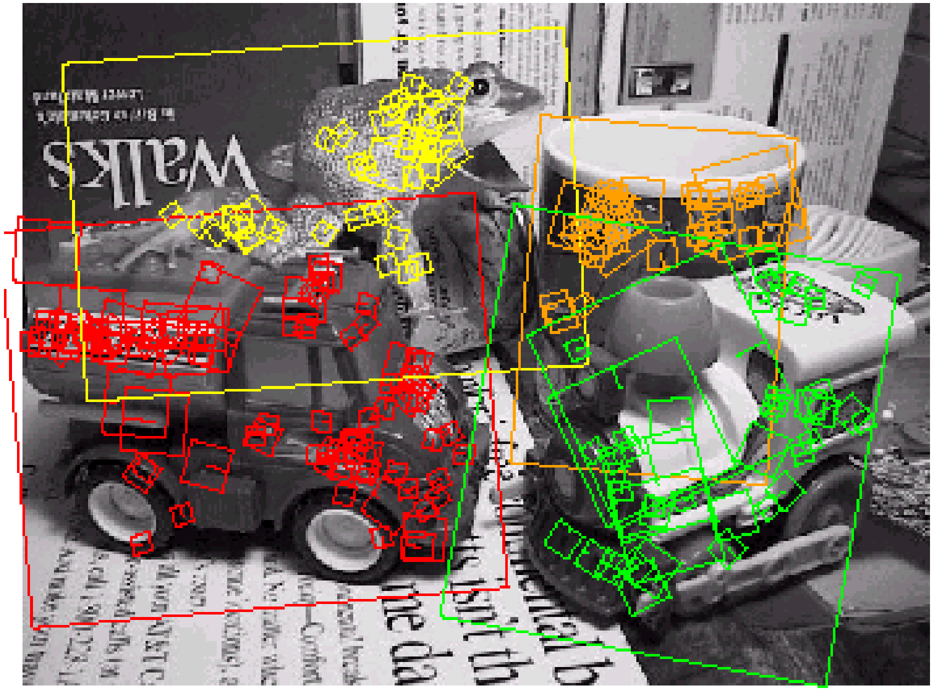
What is the minimum number of matched points that we need?

Finding the objects (in detail)

1. Match interest points from input image to database image
2. Get location/scale/orientation using Hough voting
   - In training, each point has known position/scale/orientation wrt whole object
   - Matched points vote for the position, scale, and orientation of the entire object
   - Bins for x, y, scale, orientation
     - Wide bins (0.25 object length in position, 2x scale, 30 degrees orientation)
     - Vote for two closest bin centers in each direction (16 votes total)
3. Geometric verification
   - For each bin with at least 3 keypoints
   - Iterate between least squares fit and checking for inliers and outliers
4. Report object if > T inliers (T is typically 3, can be computed to match some probabilistic threshold)

Examples of recognized objects



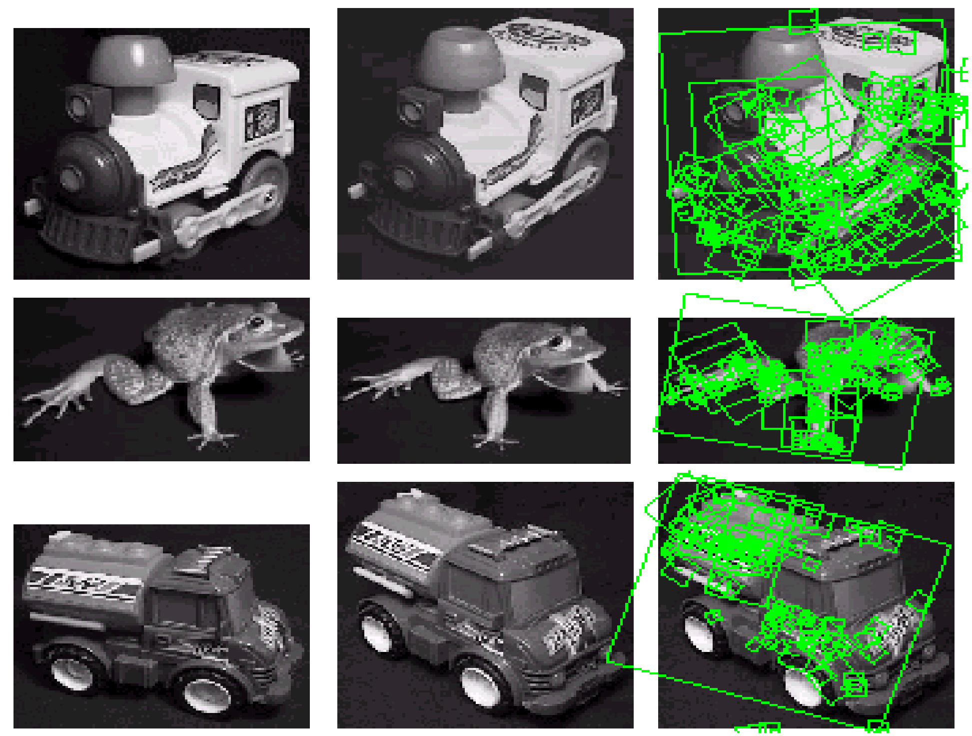


View interpolation

- Training
  - Given images of different viewpoints
  - Cluster similar viewpoints using feature matches
  - Link features in adjacent views

- Recognition
  - Feature matches may be spread over several training viewpoints

  ⇒ Use the known links to "transfer votes" to other viewpoints



[Lowe01]

Slide credit: David Lowe

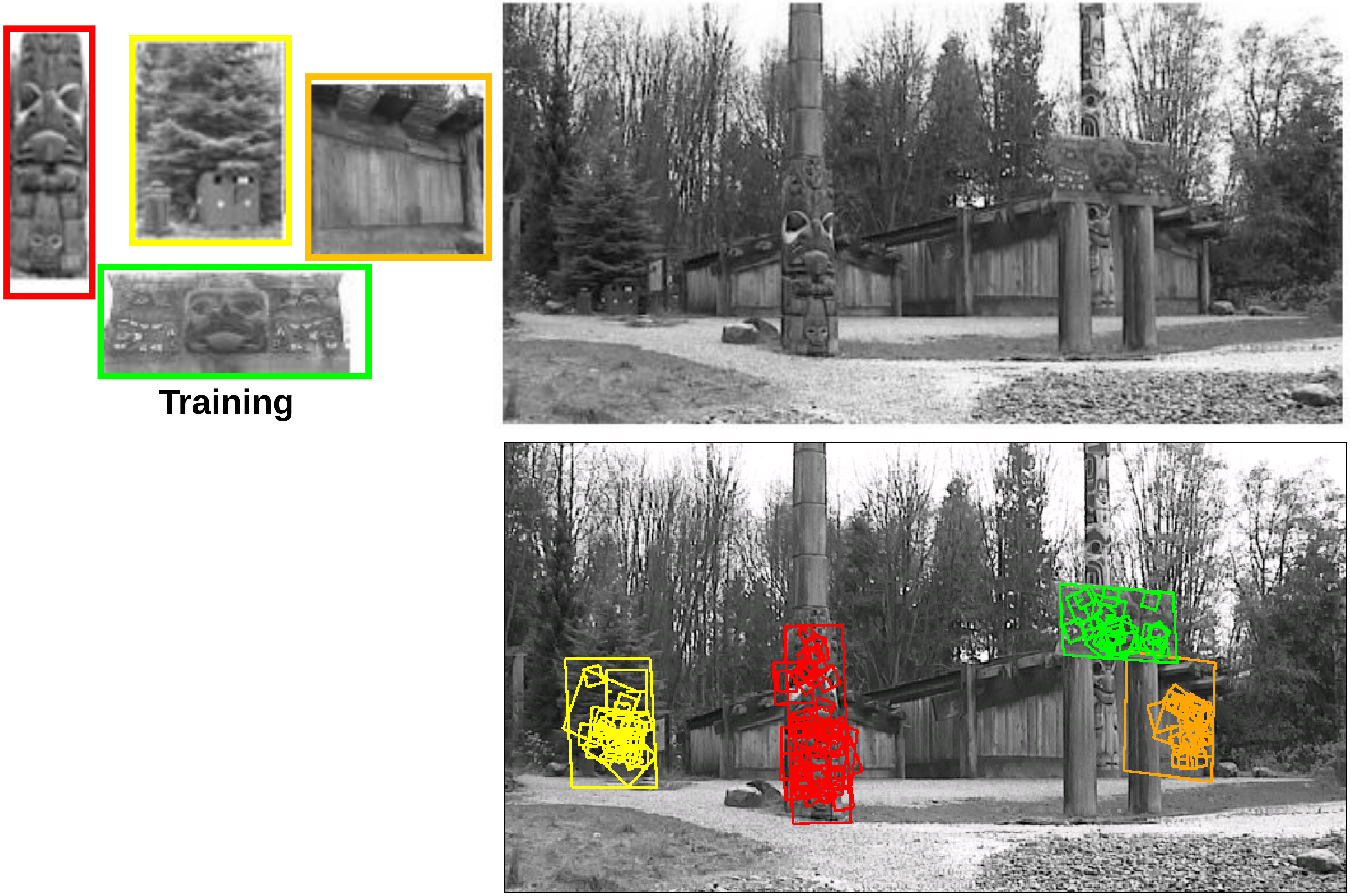
Applications

- Sony Aibo
  (Evolution Robotics)

- SIFT usage
  – Recognize docking station
  – Communicate with visual cards

- Other uses
  – Place recognition
  – Loop closure in SLAM

Location Recognition

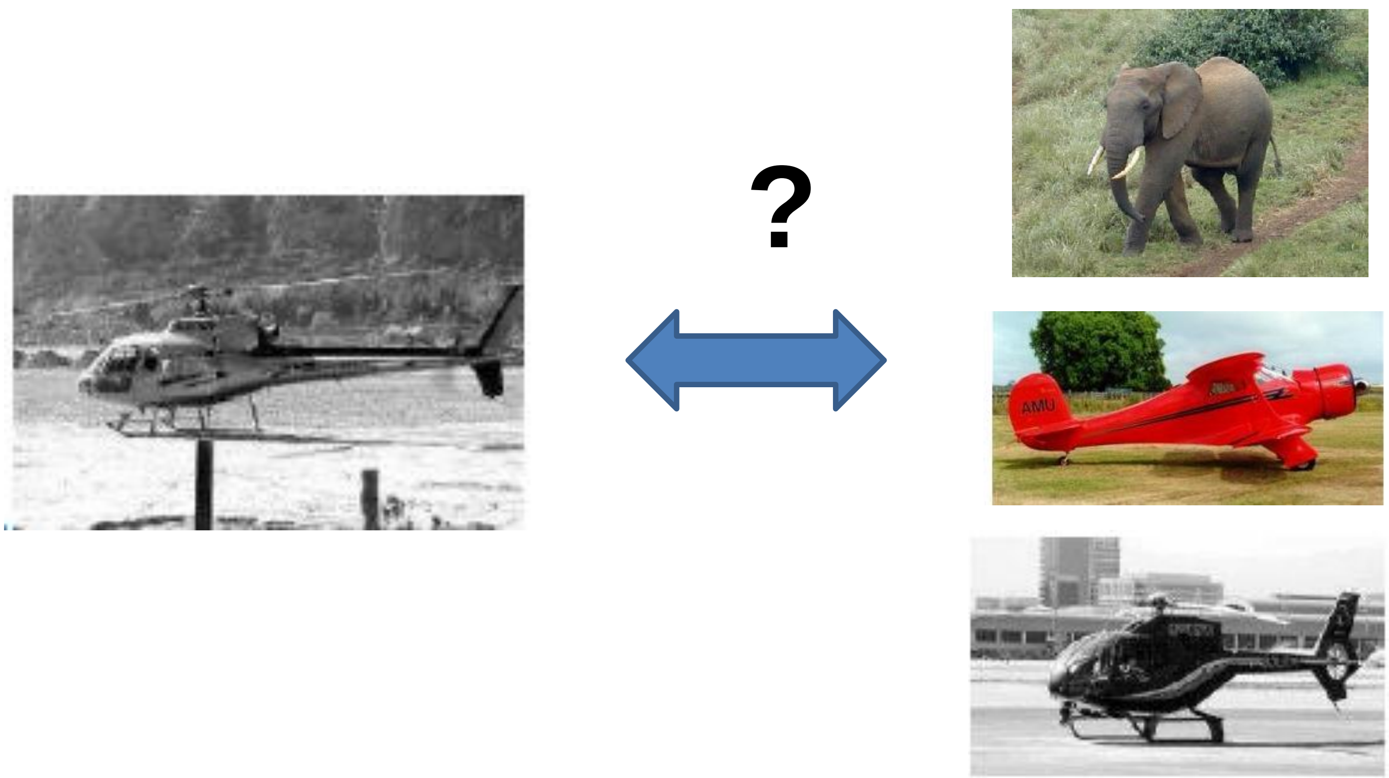


[Lowe04]

Slide credit: David Lowe

Another application: category recognition

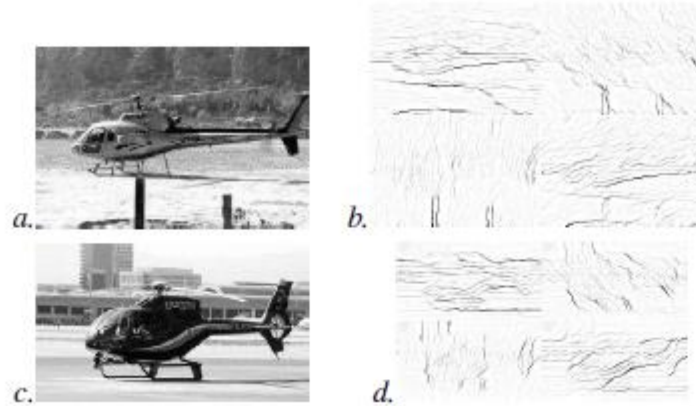
- Goal: identify what type of object is in the image
- Approach: align to known objects and choose category with best match



“Shape matching and object recognition using low distortion correspondence”, Berg et al., CVPR 2005: http://www.cnbc.cmu.edu/cns/papers/berg-cvpr05.pdf

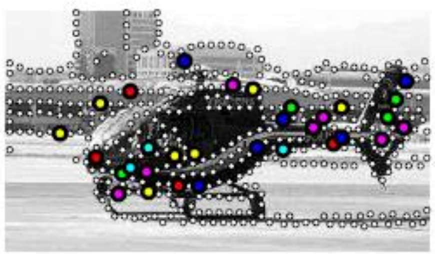
Summary of algorithm

- Input: query *q* and exemplar *e*
- For each: sample edge points and create “geometric blur” descriptor
- Compute match cost **c** to match points in *q* to each point in *e*
- Compute deformation cost **H** that penalizes change in orientation and scale for pairs of matched points
- Solve a binary quadratic program to get correspondence that minimizes **c** and **H**, using thin-plate spline deformation
- Record total cost for *e*, repeat for all exemplars, choose exemplar with minimum cost

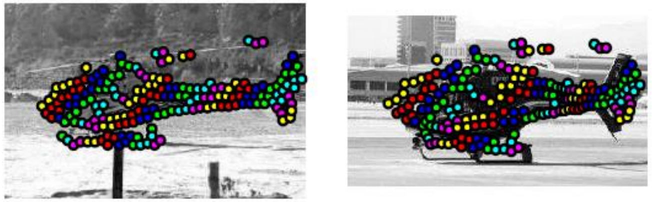


Input, Edge Maps

Geometric Blur

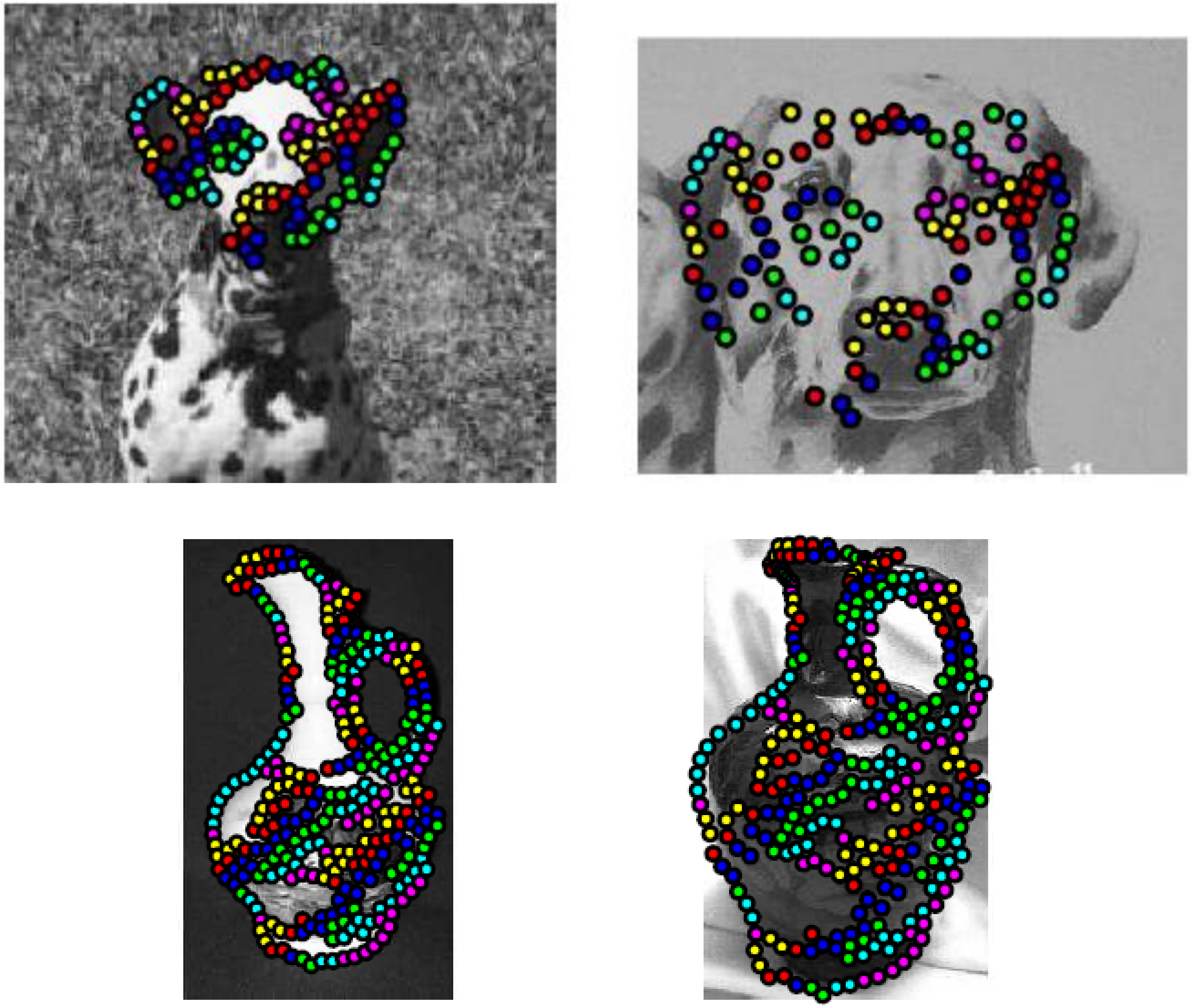


Feature Points

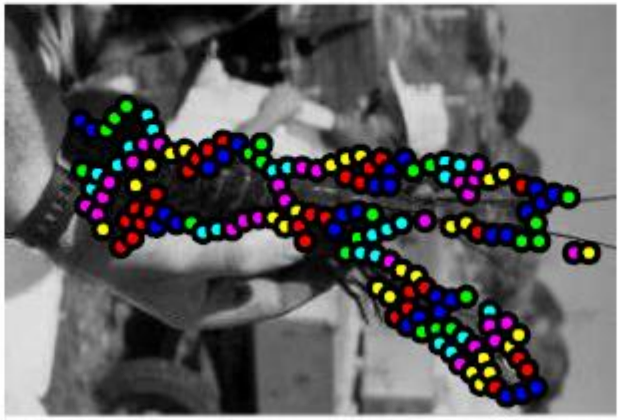


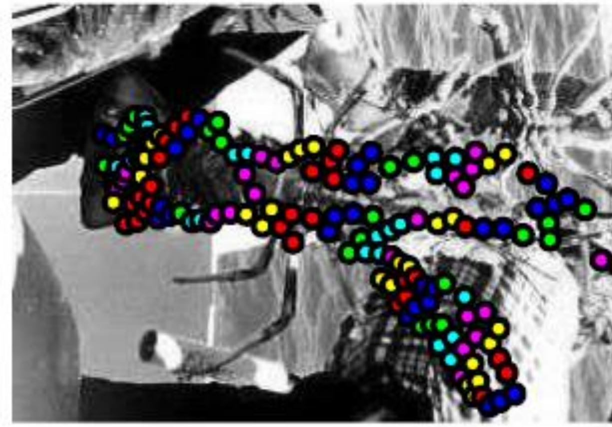
Correspondences

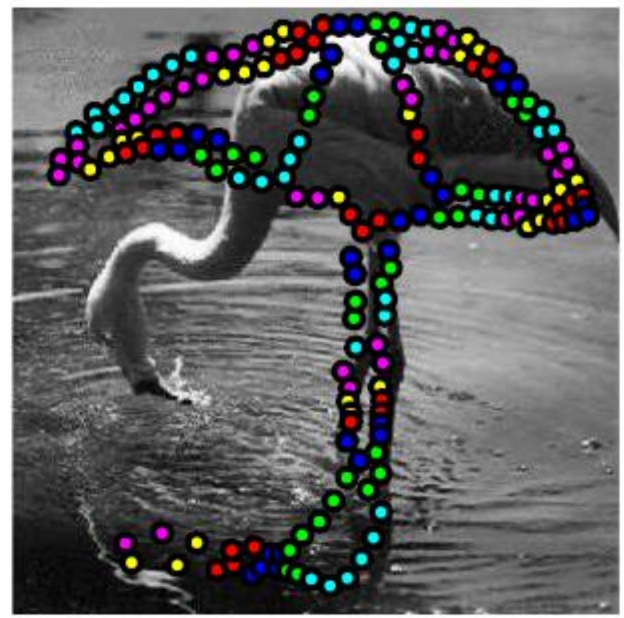
Examples of Matches



Examples of Matches







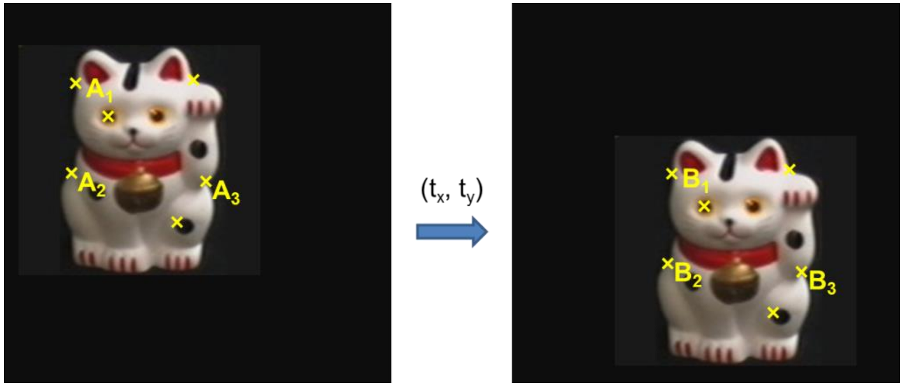
Other ideas worth being aware of

- Thin-plate splines: combines global affine warp with smooth local deformation

- Robust non-rigid point matching: http://noodle.med.yale.edu/~chui/tps-rpm.html (includes code, demo, paper)

Key concepts

- Alignment
  - Hough transform
  - RANSAC
  - ICP



- Object instance recognition
  - Find keypoints, compute descriptors
  - Match descriptors
  - Vote for / fit affine parameters
  - Return object if # inliers > T