

# Alignment and Object Instance Recognition

Computer Vision  
CS 543 / ECE 549  
University of Illinois

Derek Hoiem

# Today's class

- Alignment (continued)
- Object instance recognition
- Example of alignment-based category recognition

# Line Fitting Demo

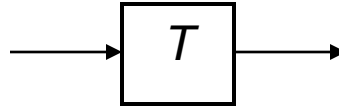
# Alignment

- Alignment: find parameters of model that maps one set of points to another
- Typically want to solve for a global transformation that accounts for \*most\* true correspondences
- Difficulties
  - Noise (typically 1-3 pixels)
  - Outliers (often 50%)
  - Many-to-one matches or multiple objects

# Parametric (global) warping



$$\mathbf{p} = (x, y)$$



$$\mathbf{p}' = (x', y')$$

Transformation  $T$  is a coordinate-changing machine:

$$\mathbf{p}' = T(\mathbf{p})$$

What does it mean that  $T$  is global?

- Is the same for any point  $\mathbf{p}$
- can be described by just a few numbers (parameters)

For linear transformations, we can represent  $T$  as a matrix

$$\mathbf{p}' = \mathbf{T}\mathbf{p}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

# Common transformations



original

## Transformed



translation



rotation



aspect



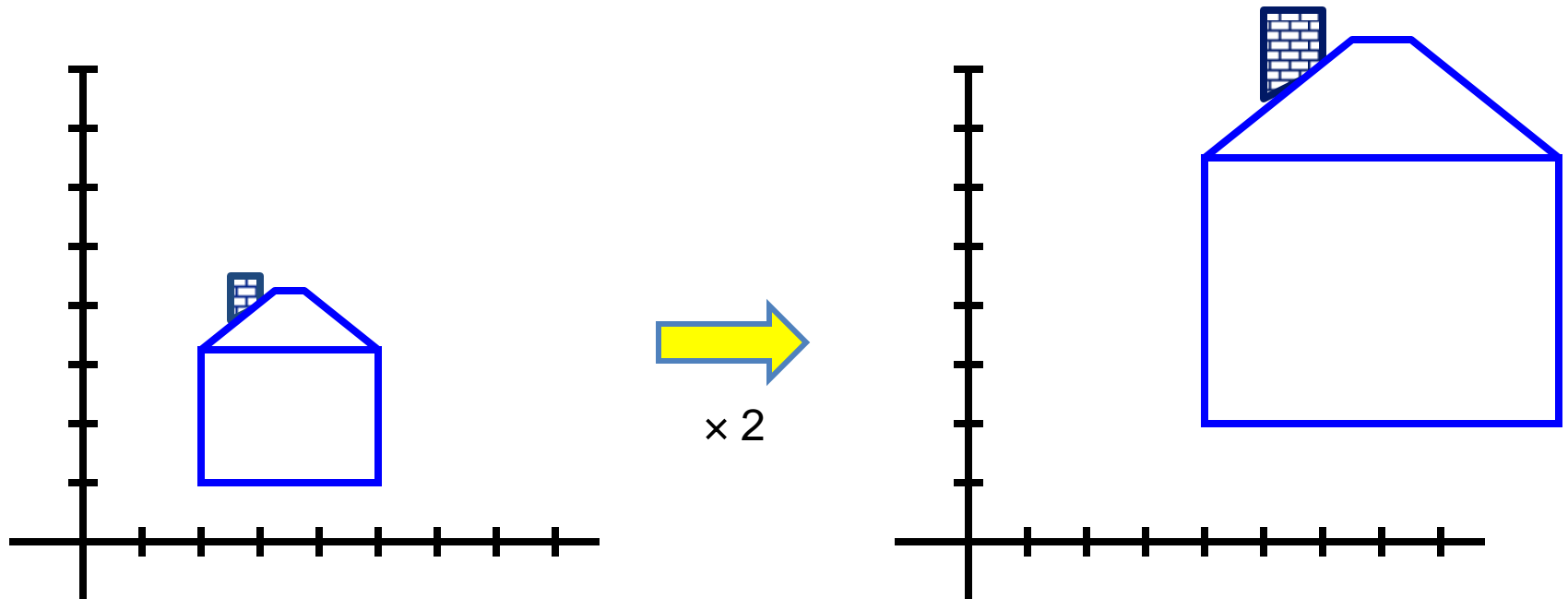
affine



perspective

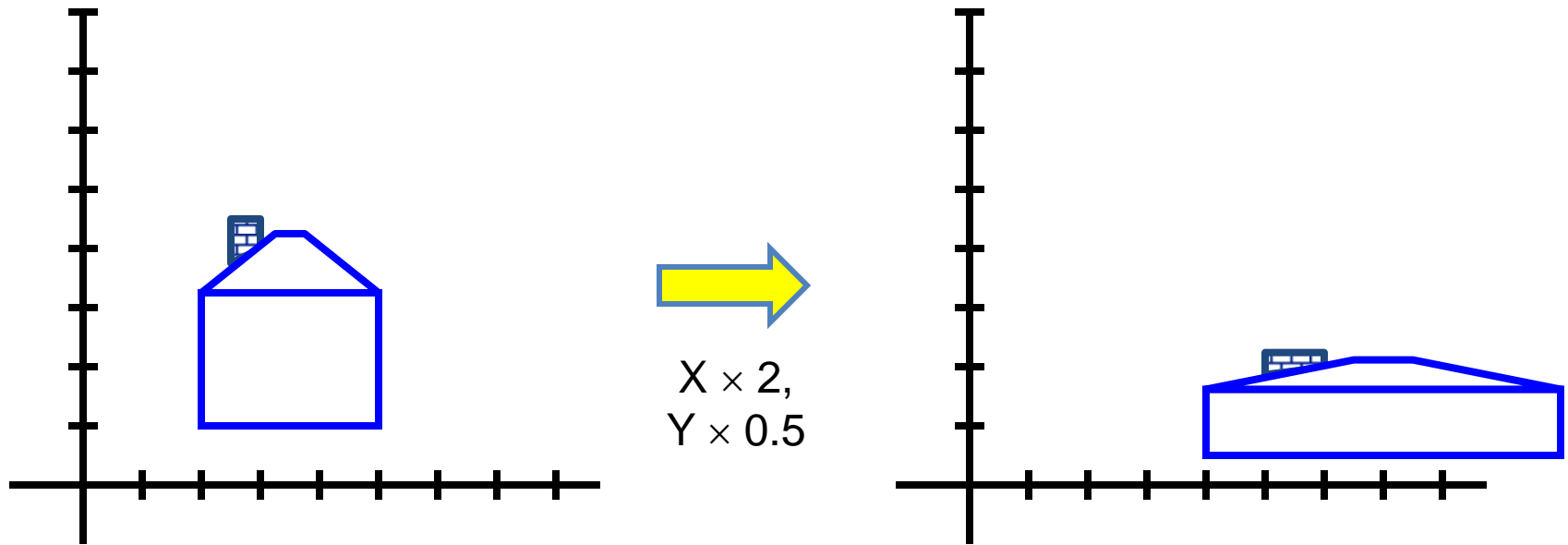
# Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:



# Scaling

- *Non-uniform scaling*: different scalars per component:

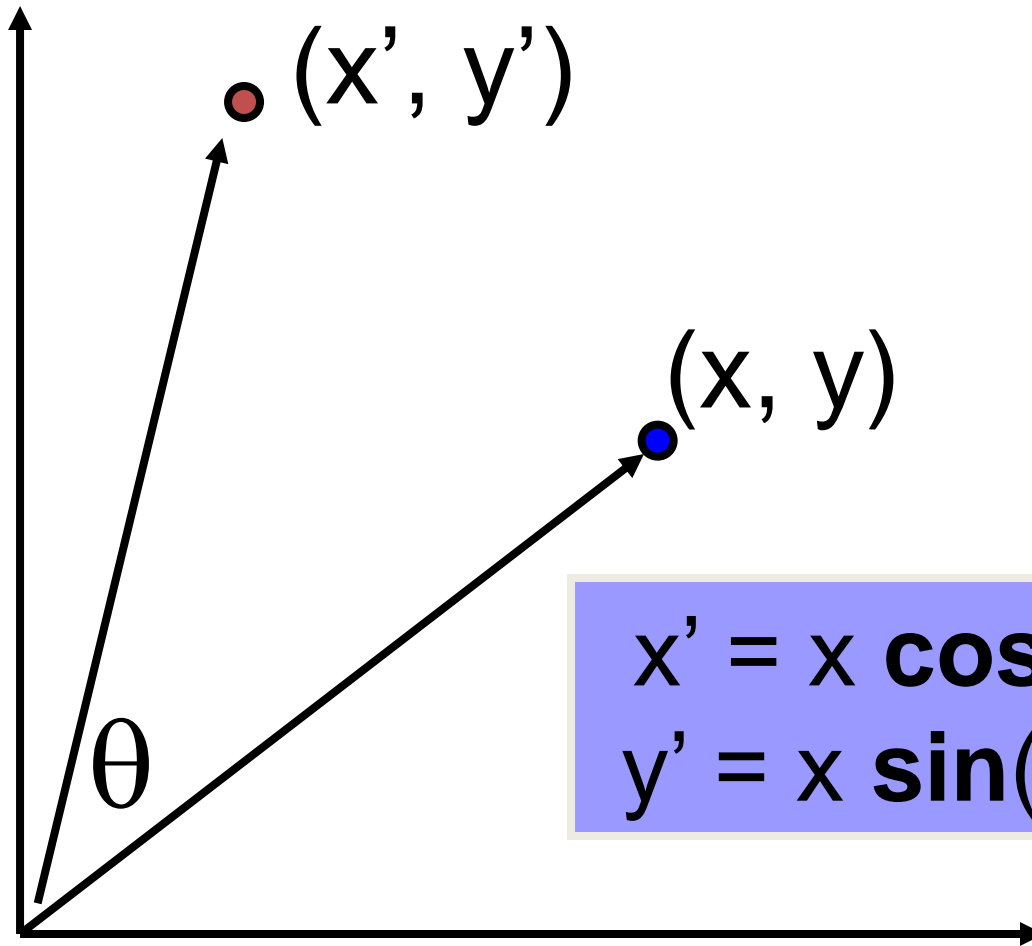




# Scaling

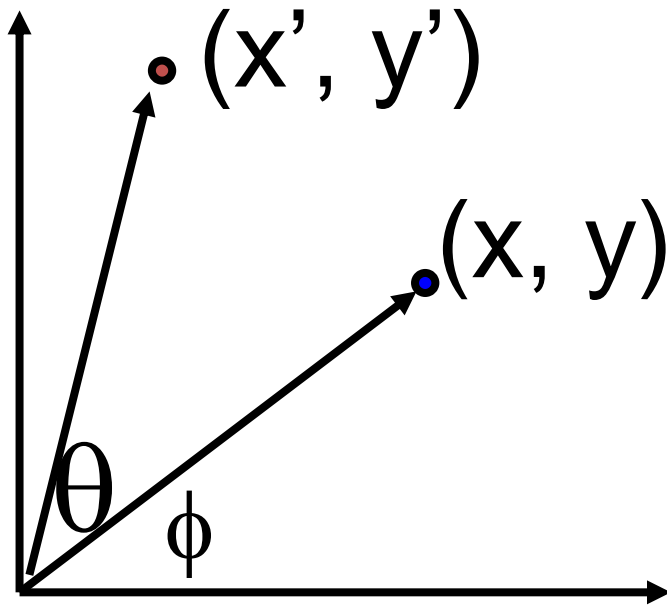
- Scaling operation:  $x' = ax$   
 $y' = by$
- Or, in matrix form: 
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

# 2-D Rotation



$$\begin{aligned}x' &= x \cos(\theta) - y \sin(\theta) \\y' &= x \sin(\theta) + y \cos(\theta)\end{aligned}$$

# 2-D Rotation



Polar coordinates...

$$x = r \cos(\phi)$$

$$y = r \sin(\phi)$$

$$x' = r \cos(\phi + \theta)$$

$$y' = r \sin(\phi + \theta)$$

Trig Identity...

$$x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$$

$$y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)$$

Substitute...

$$x' = x \cos(\theta) - y \sin(\theta)$$

$$y' = x \sin(\theta) + y \cos(\theta)$$

# 2-D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Even though  $\sin(\theta)$  and  $\cos(\theta)$  are nonlinear functions of  $\theta$ ,

- *$x'$  is a linear combination of  $x$  and  $y$*
- *$y'$  is a linear combination of  $x$  and  $y$*

What is the inverse transformation?

- Rotation by  $-\theta$
- For rotation matrices  $\mathbf{R}^{-1} = \mathbf{R}^T$

# Basic 2D transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta \\ \sin \Theta & \cos \Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

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

Affine

Affine is any combination of translation, scale, rotation, shear

# Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

Properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

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

or

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Projective Transformations

Projective transformations are combos of

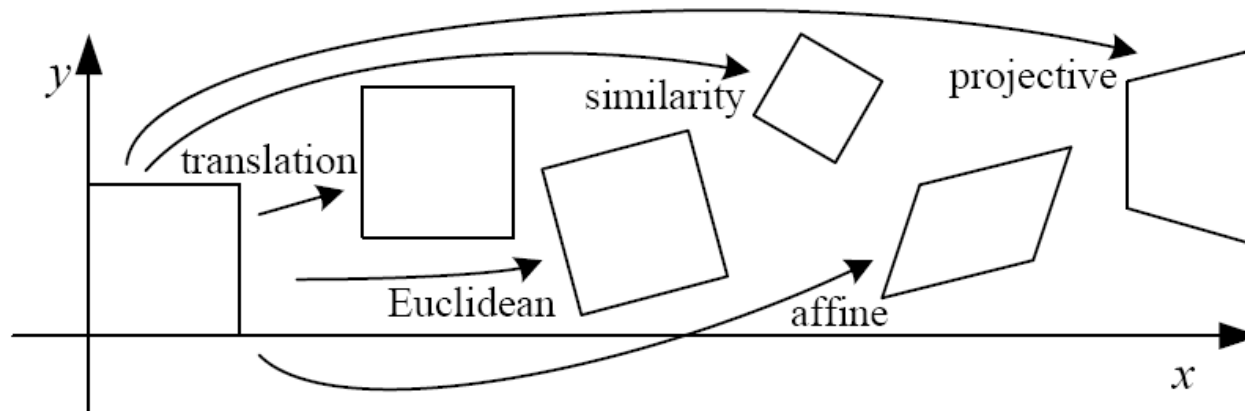
- Affine transformations, and
- Projective warps






$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis
- Projective matrix is defined up to a scale (8 DOF)

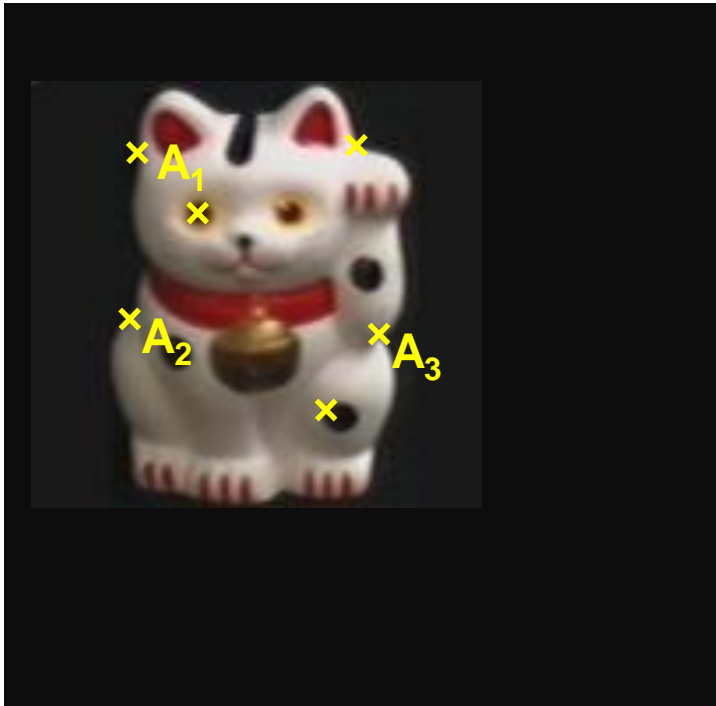
# 2D image transformations (reference table)



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I & t \end{bmatrix}_{2 \times 3}$	2	orientation + ...	
rigid (Euclidean)	$\begin{bmatrix} R & t \end{bmatrix}_{2 \times 3}$	3	lengths + ...	
similarity	$\begin{bmatrix} sR & t \end{bmatrix}_{2 \times 3}$	4	angles + ...	
affine	$\begin{bmatrix} A \end{bmatrix}_{2 \times 3}$	6	parallelism + ...	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3 \times 3}$	8	straight lines	



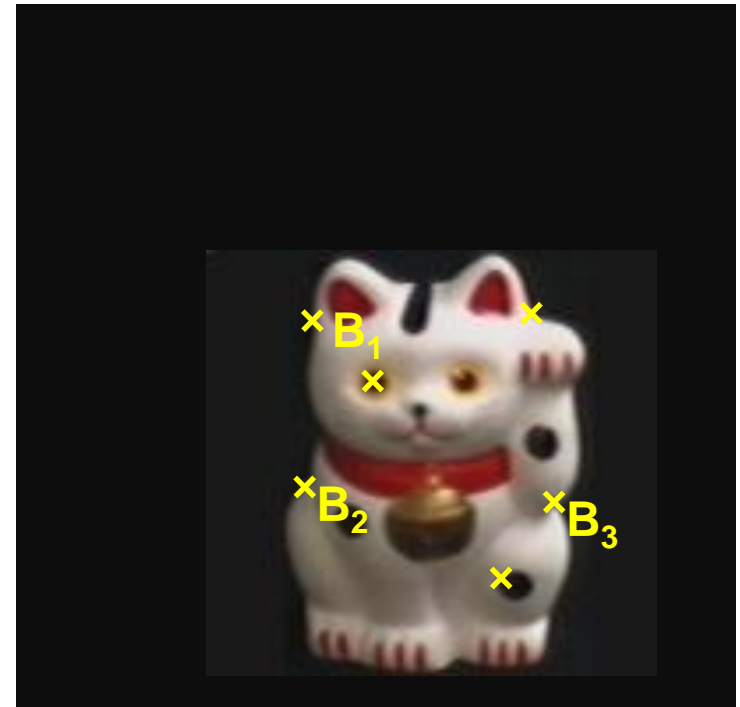
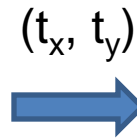
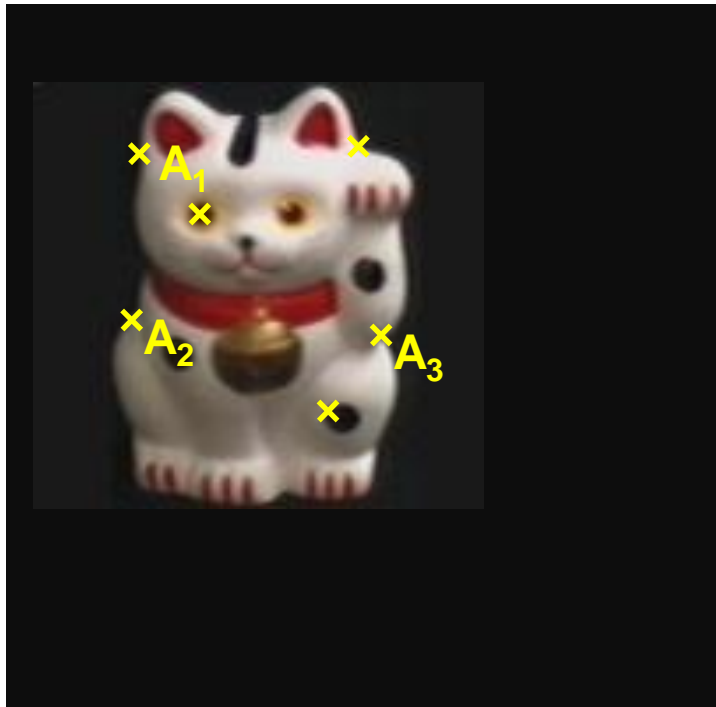
# Example: solving for translation



Given matched points in  $\{A\}$  and  $\{B\}$ , estimate the translation of the object

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

# Example: solving for translation



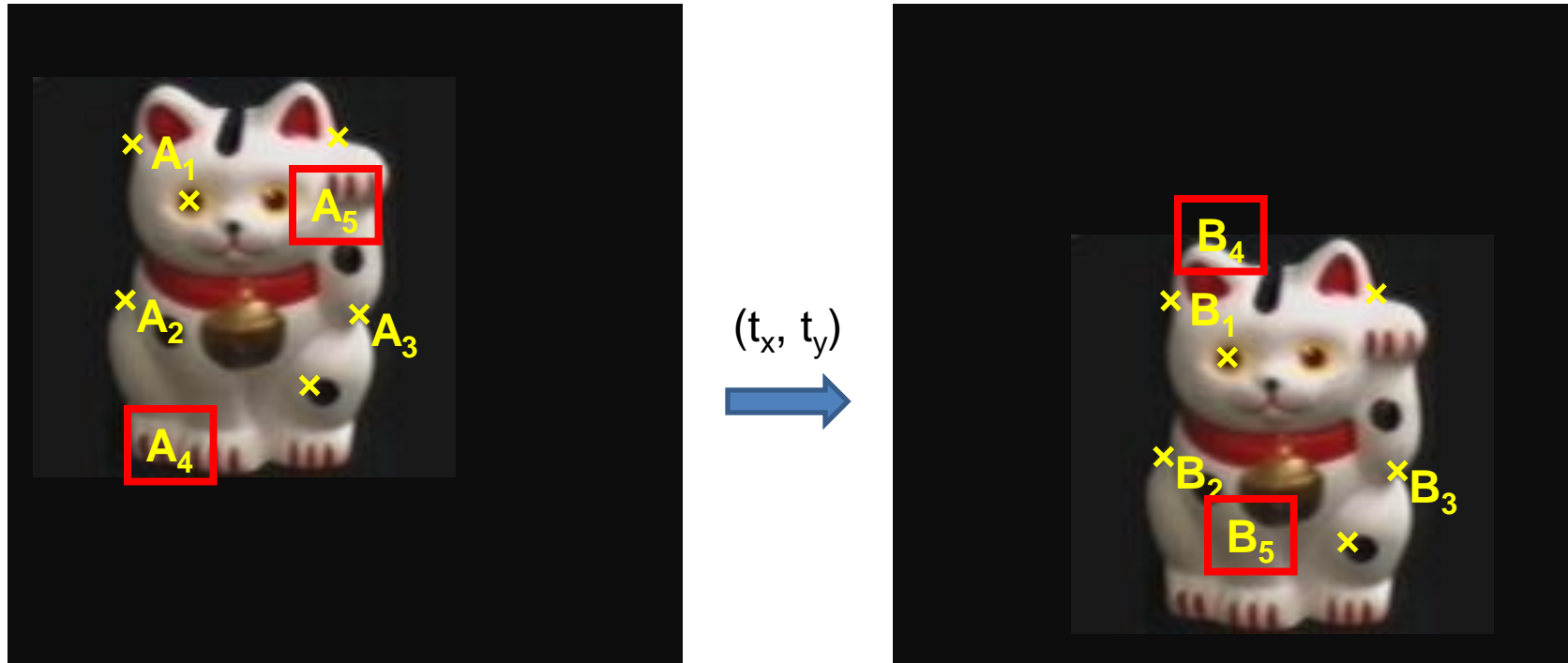
## Least squares solution

1. Write down objective function
2. Derived solution
  - a) Compute derivative
  - b) Compute solution
3. Computational solution
  - a) Write in form  $Ax=b$
  - b) Solve using pseudo-inverse or eigenvalue decomposition

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$

# Example: solving for translation



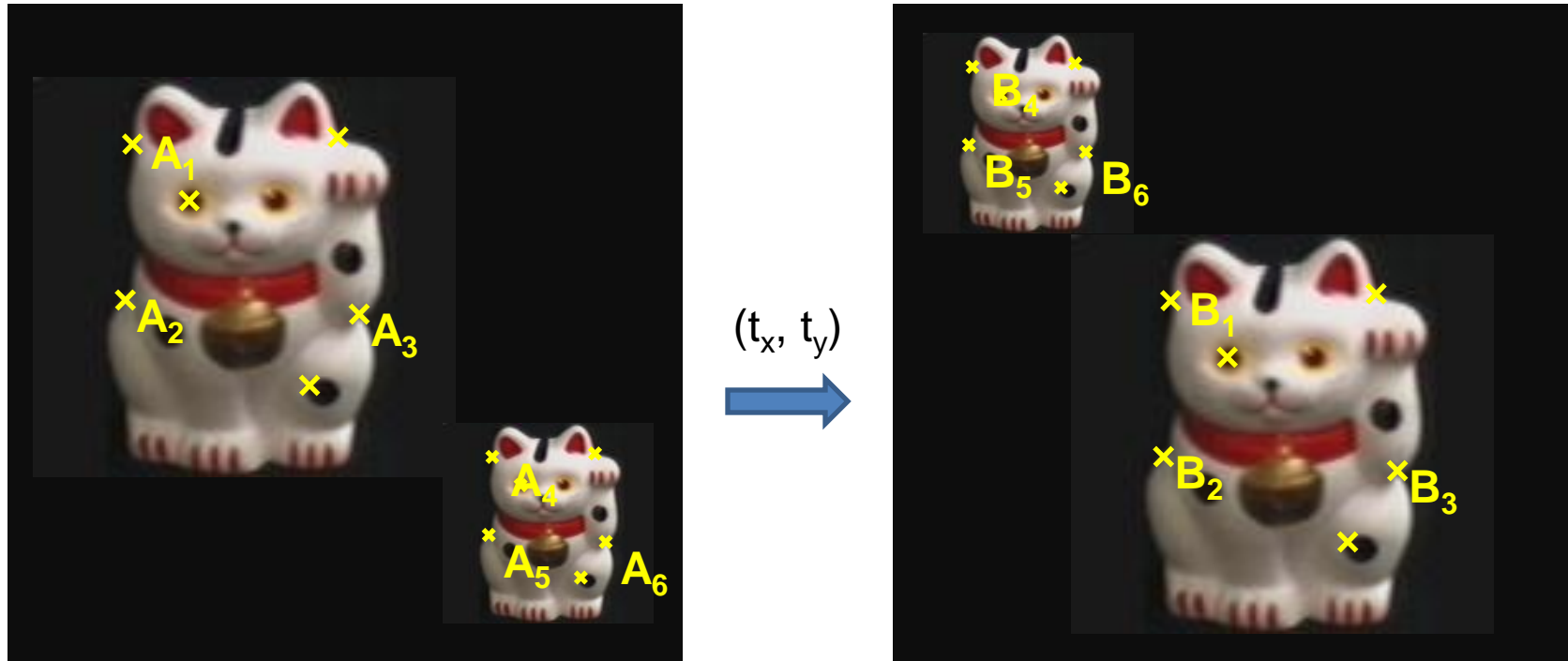
**Problem: outliers**

## RANSAC solution

1. Sample a set of matching points (1 pair)
2. Solve for transformation parameters
3. Score parameters with number of inliers
4. Repeat steps 1-3 N times

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

# Example: solving for translation



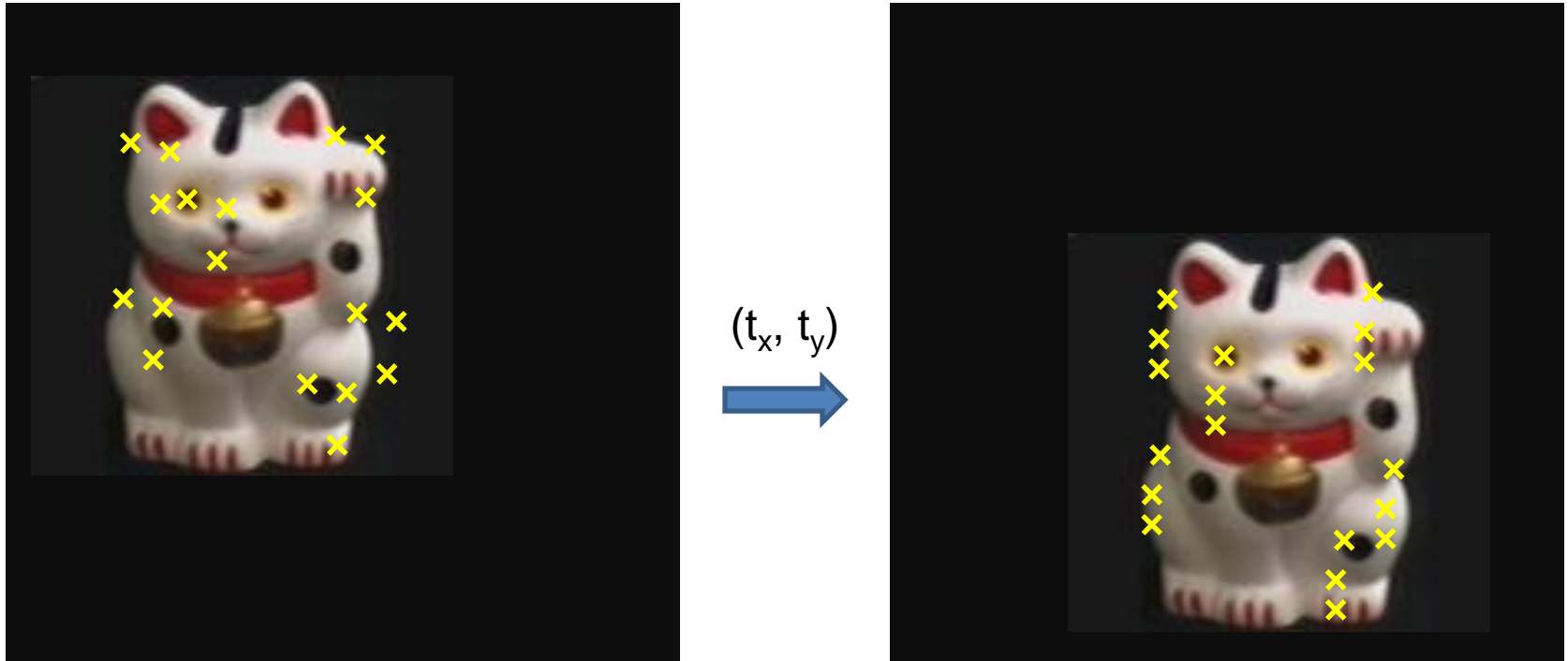
**Problem: outliers, multiple objects, and/or many-to-one matches**

## Hough transform solution

1. Initialize a grid of parameter values
2. Each matched pair casts a vote for consistent values
3. Find the parameters with the most votes
4. Solve using least squares with inliers

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

# Example: solving for translation

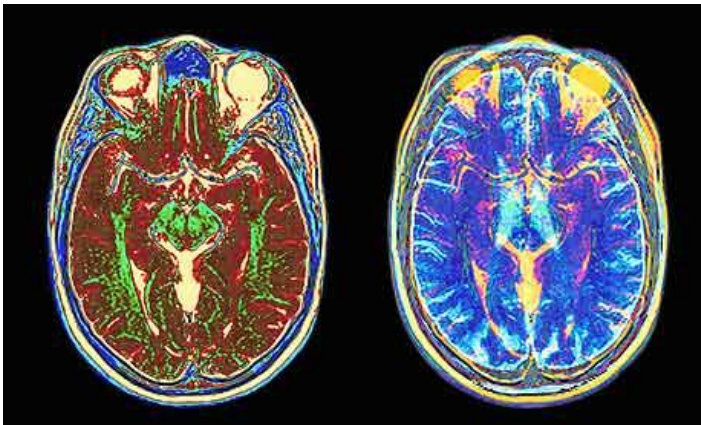


**Problem: no initial guesses for correspondence**

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

# What if you want to align but have no prior matched pairs?

- Hough transform and RANSAC not applicable
- Important applications



Medical imaging: match brain scans or contours



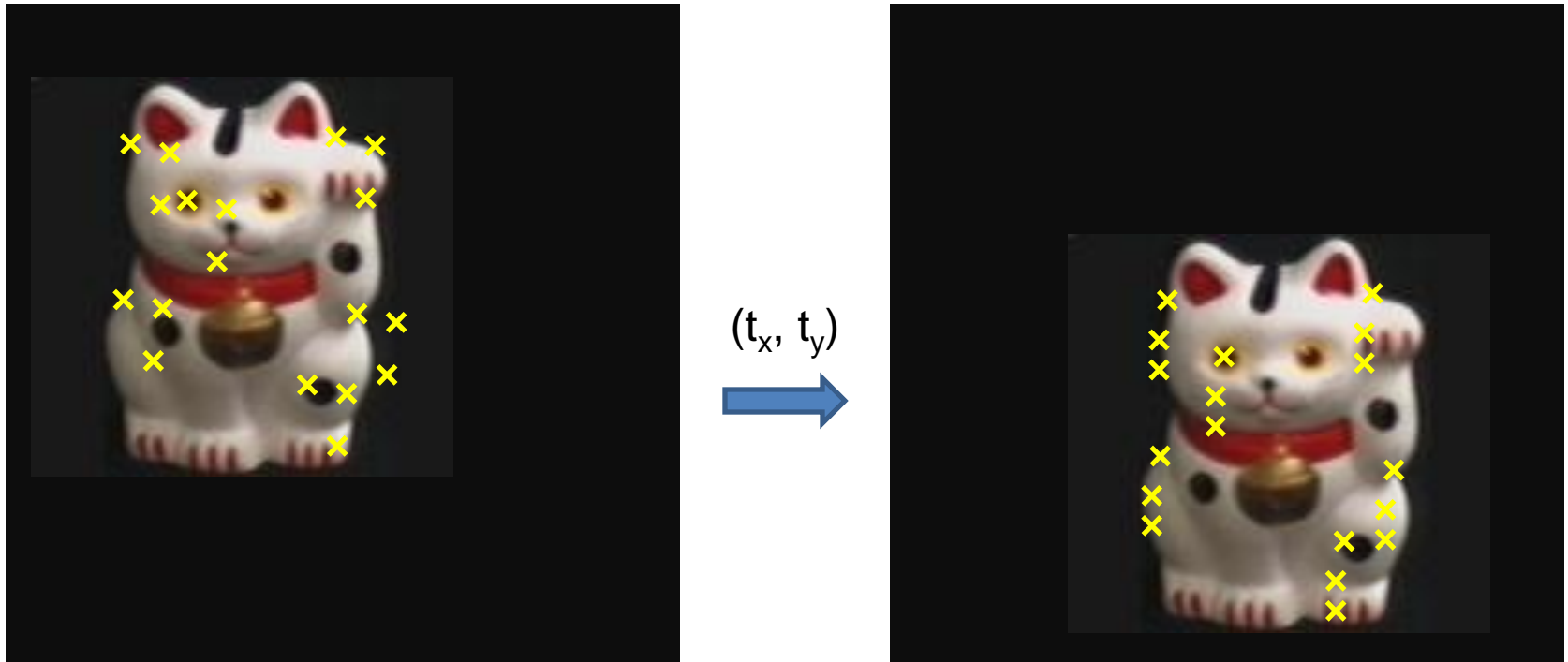
Robotics: match point clouds

# Iterative Closest Points (ICP) Algorithm

Goal: estimate transform between two dense sets of points

1. **Initialize** transformation (e.g., compute difference in means and scale)
2. **Assign** each point in {Set 1} to its nearest neighbor in {Set 2}
3. **Estimate** transformation parameters
  - e.g., least squares or robust least squares
4. **Transform** the points in {Set 1} using estimated parameters
5. **Repeat** steps 2-4 until change is very small

# Example: solving for translation



**Problem: no initial guesses for correspondence**

## ICP solution

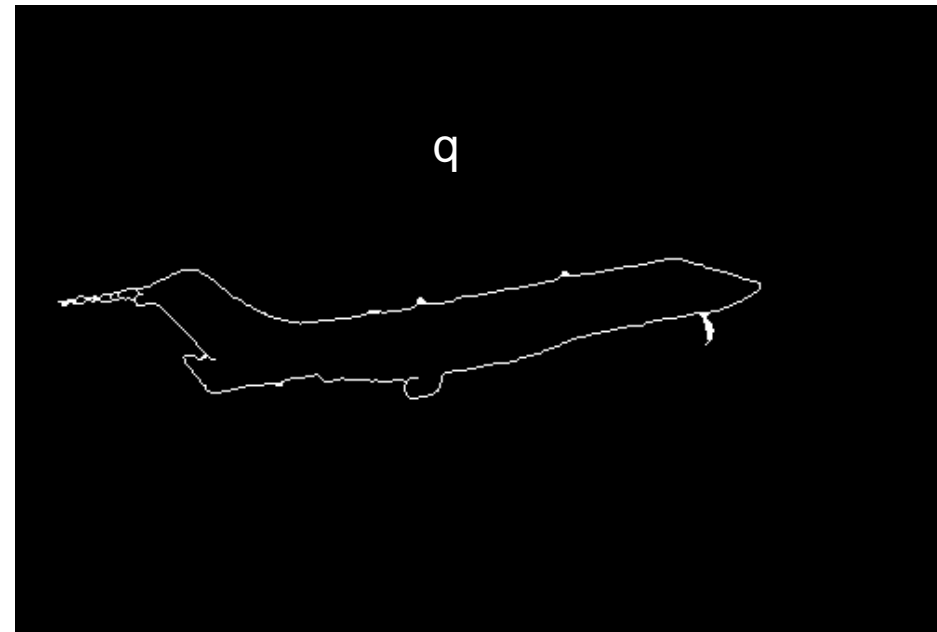
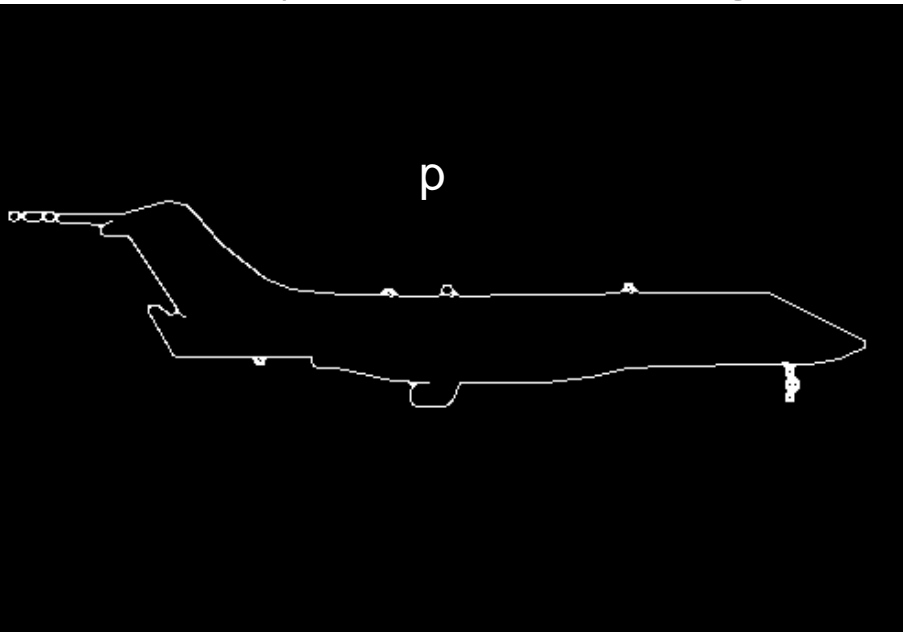
1. Find nearest neighbors for each point
2. Compute transform using matches
3. Move points using transform
4. Repeat steps 1-3 until convergence

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



# Example: aligning boundaries

1. Extract edge pixels  $p_1 \dots p_n$  and  $q_1 \dots q_m$
2. Compute initial transformation (e.g., compute translation and scaling by center of mass, variance within each image)
3. Get nearest neighbors: for each point  $p_i$  find corresponding  $\text{match}(i) = \underset{j}{\operatorname{argmin}} \operatorname{dist}(p_i, q_j)$
4. Compute transformation  $\mathbf{T}$  based on matches
5. Warp points  $\mathbf{p}$  according to  $\mathbf{T}$
6. Repeat 3-5 until convergence

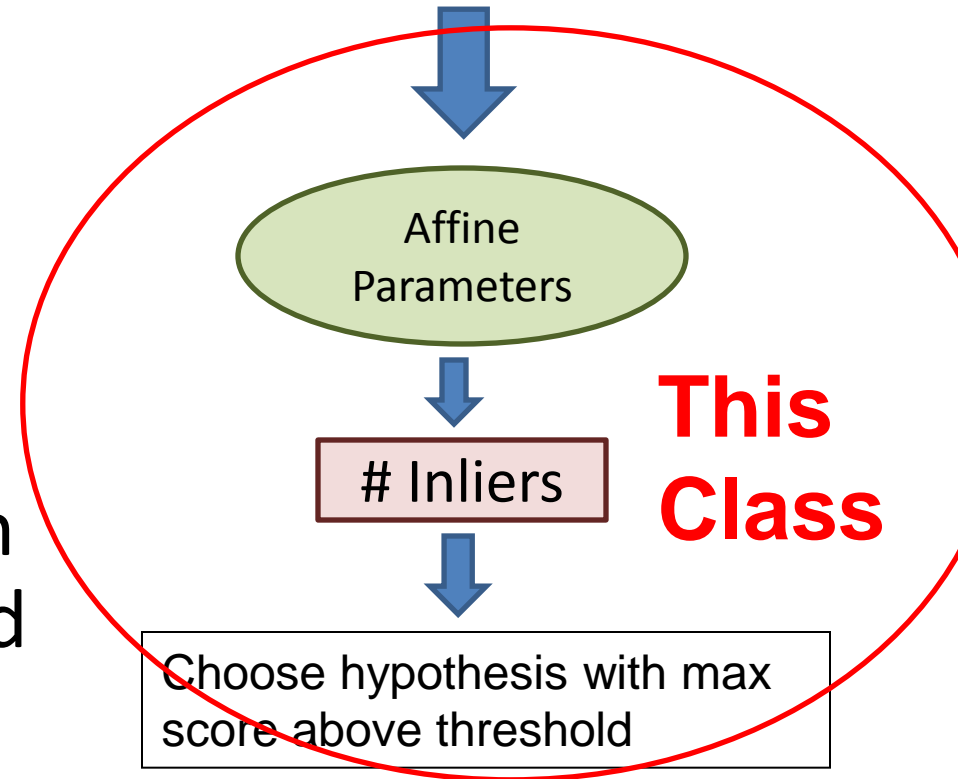
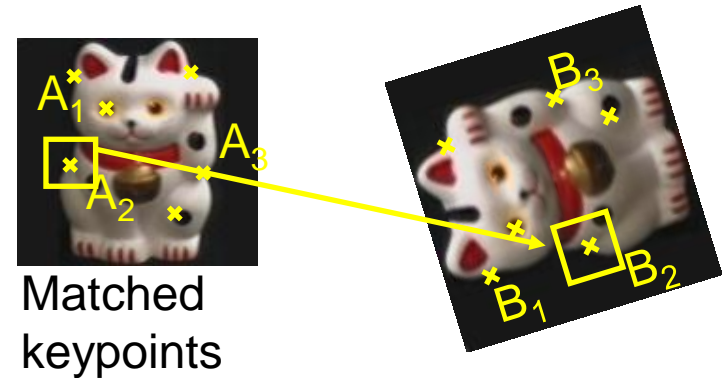


# Algorithm Summary

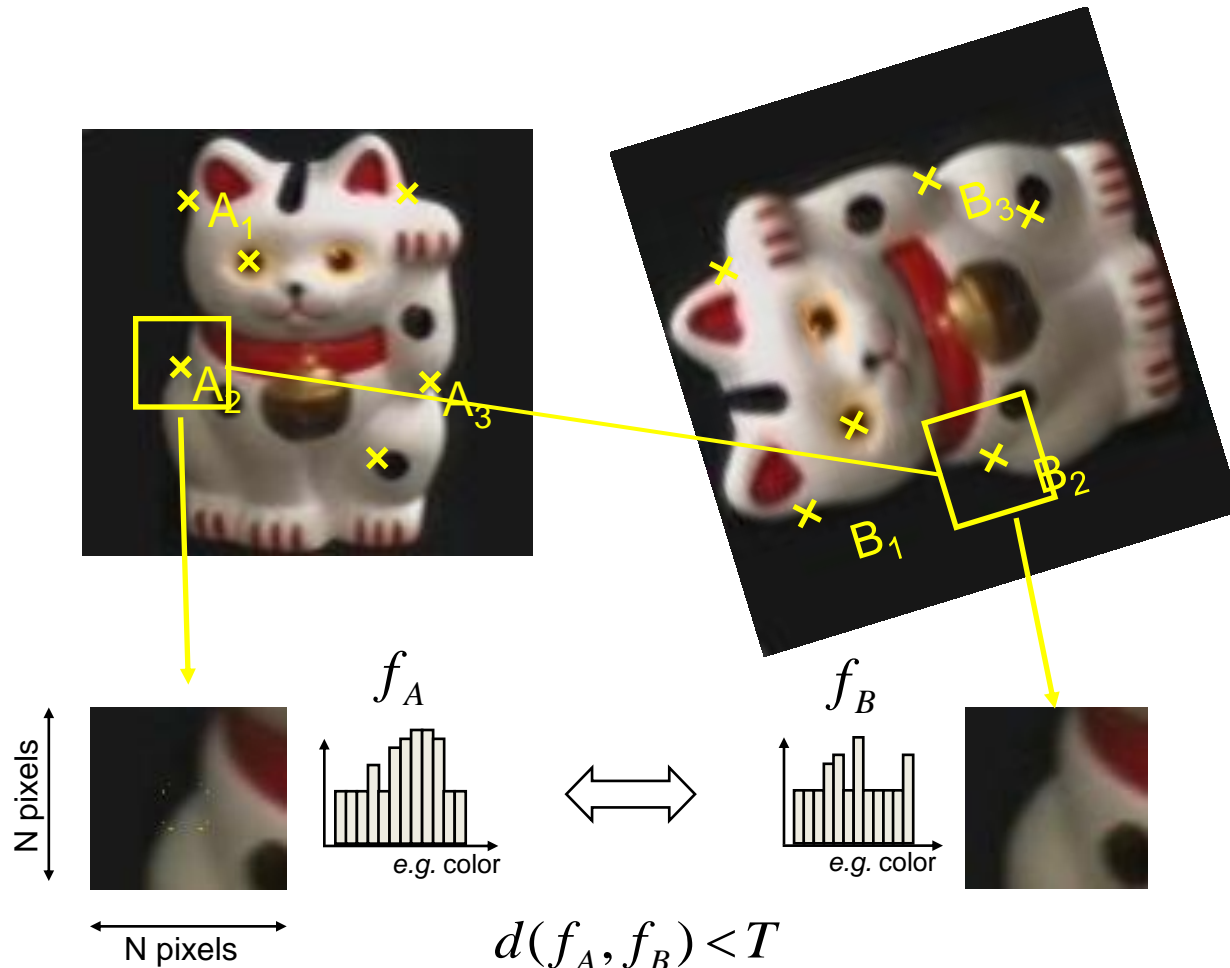
- Least Squares Fit
  - closed form solution
  - robust to noise
  - not robust to outliers
- Robust Least Squares
  - improves robustness to noise
  - requires iterative optimization
- Hough transform
  - robust to noise and outliers
  - can fit multiple models
  - only works for a few parameters (1-4 typically)
- RANSAC
  - robust to noise and outliers
  - works with a moderate number of parameters (e.g, 1-8)
- Iterative Closest Point (ICP)
  - For local alignment only: does not require initial correspondences

# 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

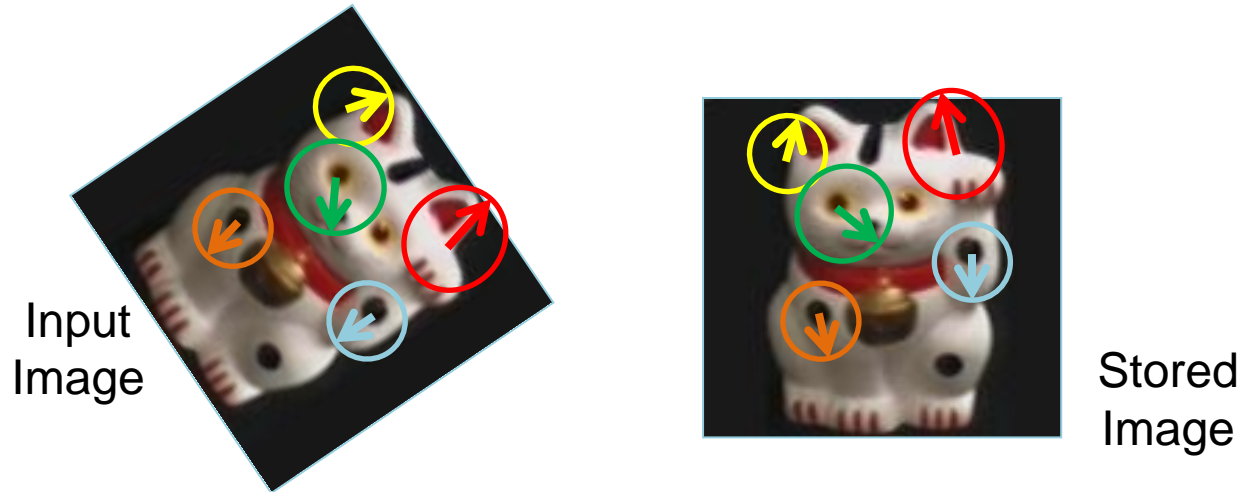


# Overview of Keypoint Matching



1. Find a set of distinctive keypoints
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

# Finding the objects (overview)



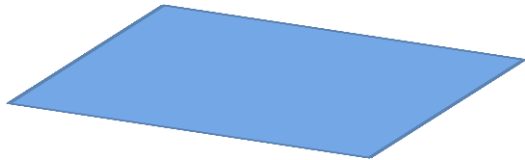
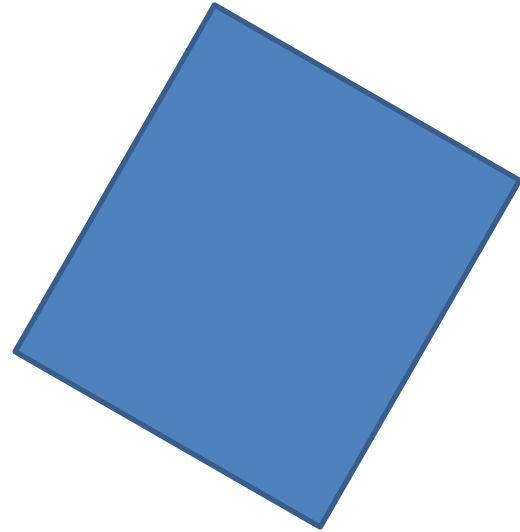
1. Match interest points from input image to database image
2. Matched points vote for rough position/orientation/scale of object
3. Find position/orientation/scales that have at least three votes
4. Compute affine registration and matches using iterative least squares with outlier check
5. Report object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

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

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

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ & & \vdots & & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ \vdots \end{bmatrix}$$

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

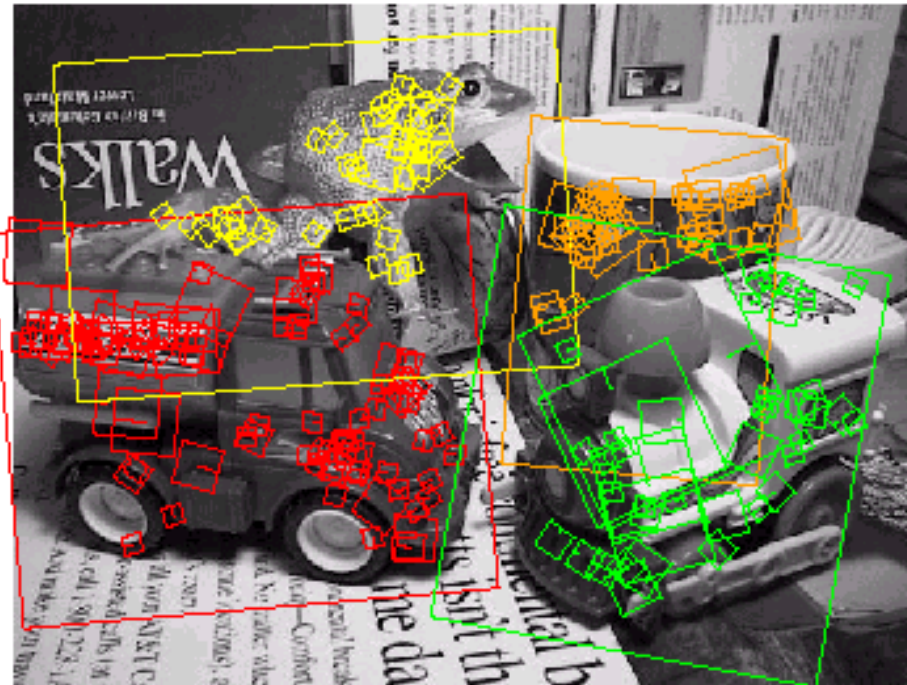
What is the minimum number of matched points that we need?



# Finding the objects (in detail)

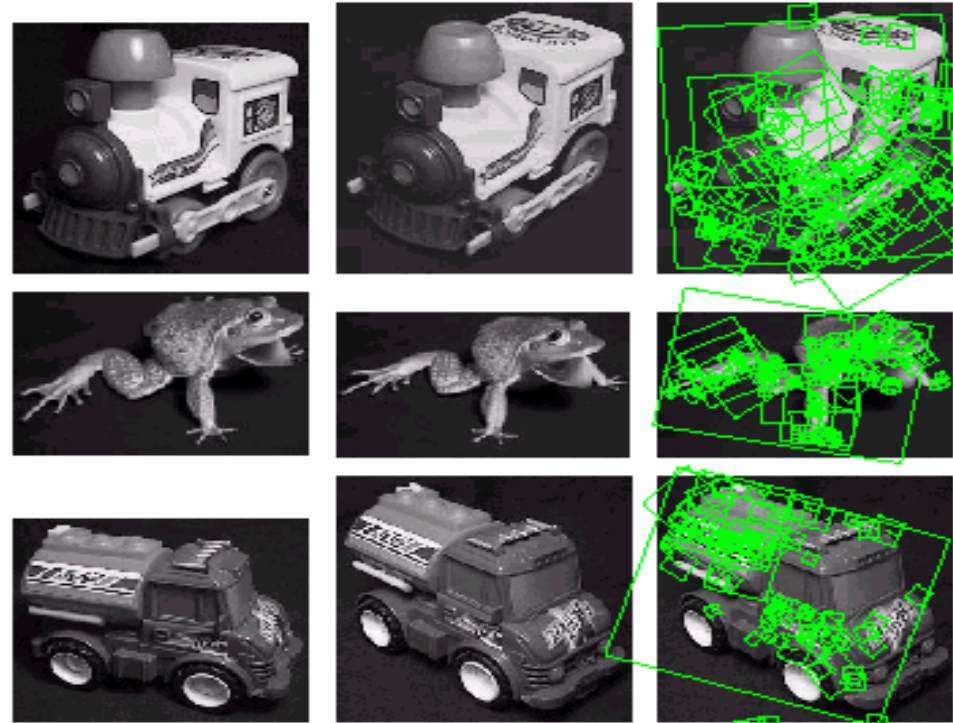
1. Match interest points from input image to database image
2. Get location/scale/orientation using Hough voting
  - In training, each point has known position/scale/orientation wrt whole object
  - Matched points vote for the position, scale, and orientation of the entire object
  - Bins for x, y, scale, 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

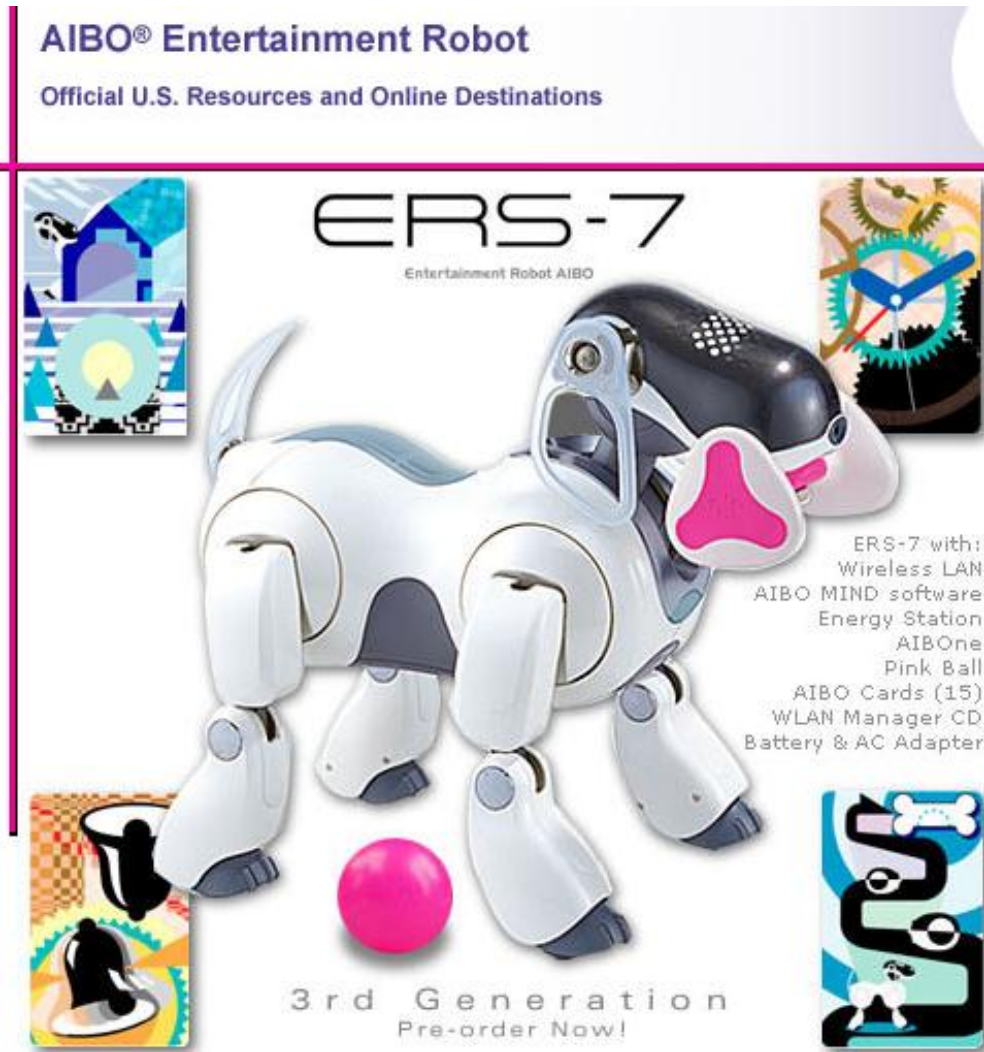
- Training
  - Given images of different viewpoints
  - Cluster similar viewpoints using feature matches
  - Link features in adjacent views
- Recognition
  - Feature matches may be spread over several training viewpoints
  - ⇒ Use the known links to “transfer votes” to other viewpoints



[Lowe01]

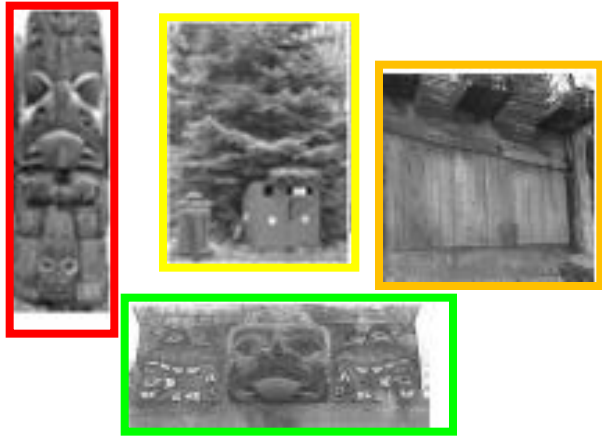
# Applications

- Sony Aibo (Evolution Robotics)
- SIFT usage
  - Recognize docking station
  - Communicate with visual cards
- Other uses
  - Place recognition
  - Loop closure in SLAM

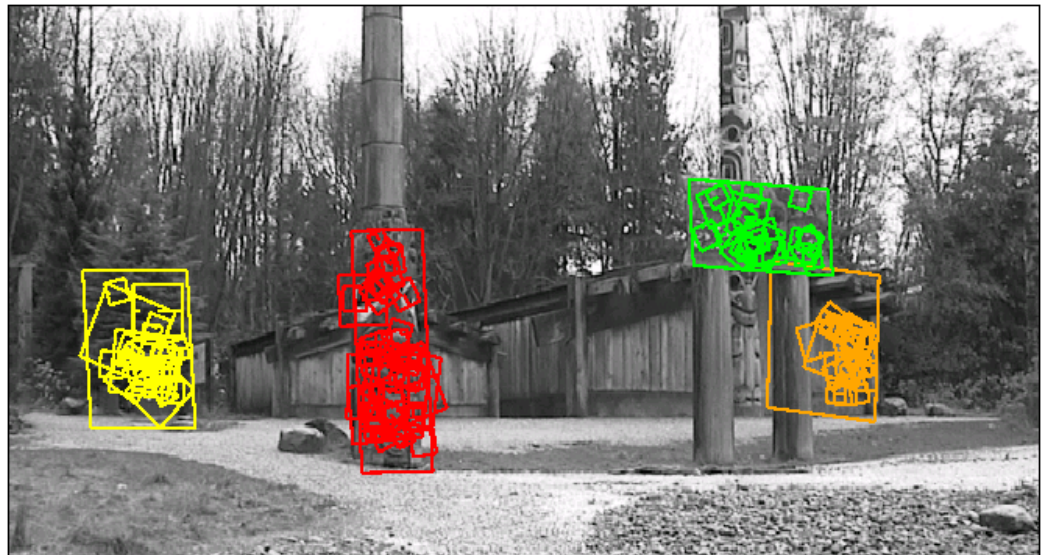




# Location Recognition



Training



[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



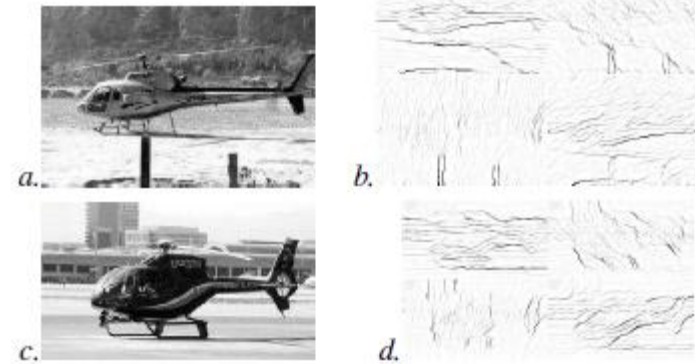
?



“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

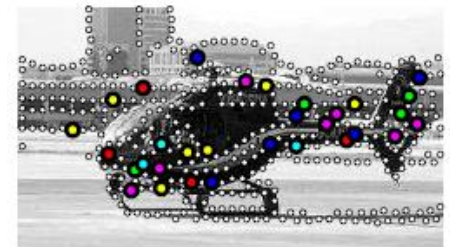
- Input: query  $q$  and exemplar  $e$
- For each: sample edge points and create “geometric blur” descriptor
- Compute match cost  $\mathbf{c}$  to match points in  $q$  to each point in  $e$
- Compute deformation cost  $\mathbf{H}$  that penalizes change in orientation and scale for pairs of matched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



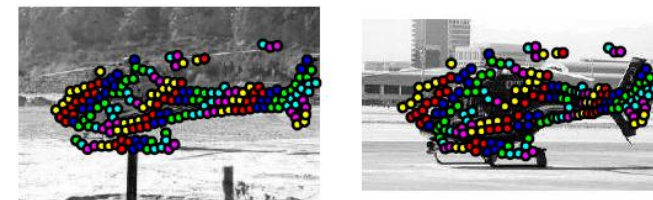
Input, Edge Maps



Geometric Blur



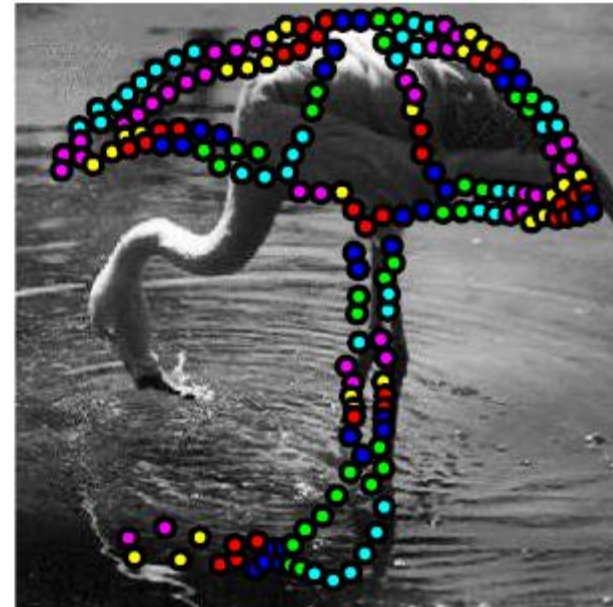
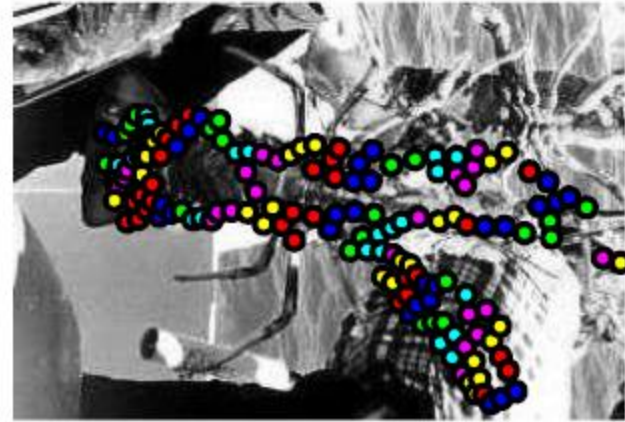
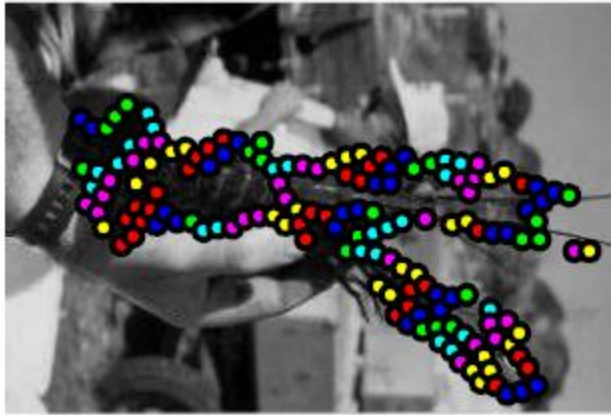
Feature Points



Correspondences

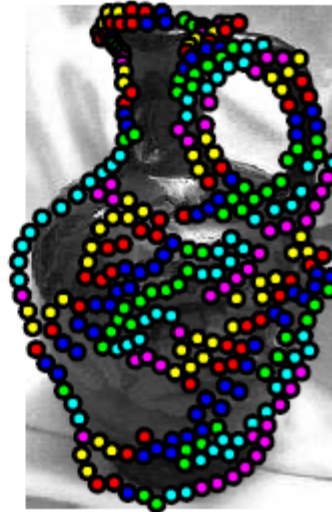
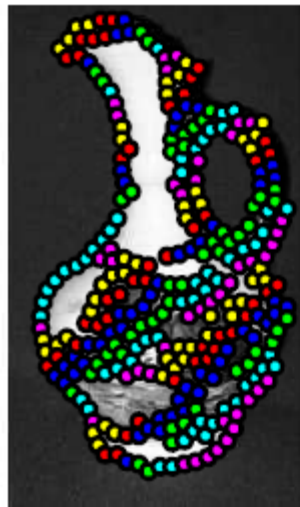
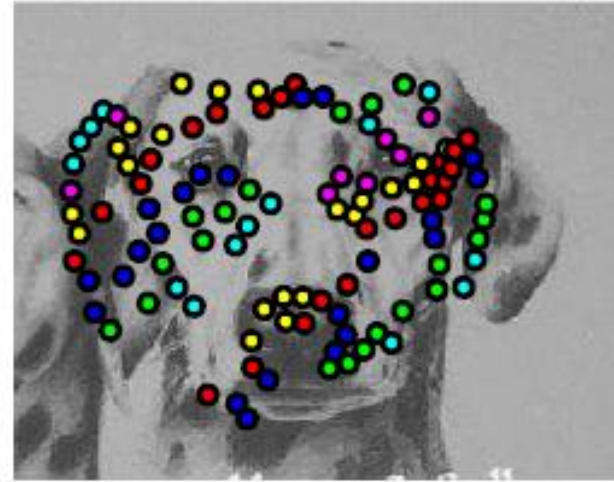


# Examples of Matches





# Examples of Matches



# Other ideas worth being aware of

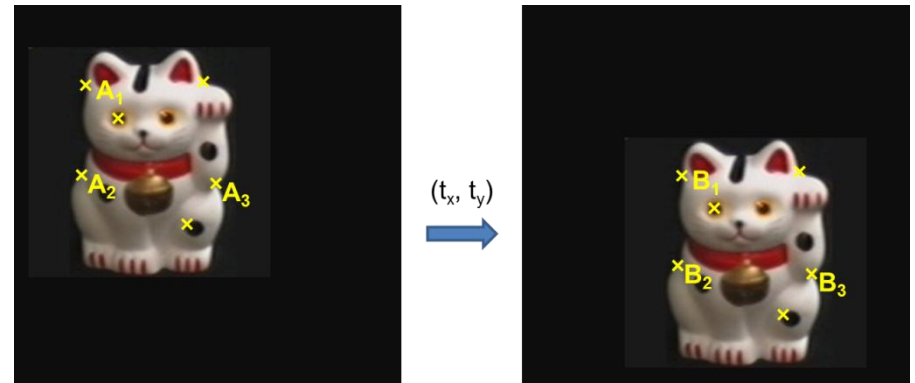
- Thin-plate splines: combines global affine warp with smooth local deformation
- Robust non-rigid point matching:  
<http://noodle.med.yale.edu/~chui/tps-rpm.html>  
(includes code, demo, paper)

# Think about your final projects

- Strongly encouraged to work in groups of 2-4 (but if you have a good reason to work by self, could be ok)
- Projects don't need to be of publishable originality but should evince independent effort to learn about a new topic, try something new, or apply to an application of interest
- Project ideas from Cinda Hereen
  1. **Classroom attendance:** I teach in Siebel 1404, and I'd like to be able to track attendance by taking photos of the room (students who opt out can just put their heads down). Extra facts in the problem: a) I have photo rosters; b) i'm willing to use a tagging system wherein students can verify our assertions; c) the results don't have to be exact.
  2. **Medication tracking:** I would like for a person to be able to chart his/her medicine consumption by taking a photo of the med bottles. Extra facts: a. it has to work on a mobile device (limited computation), b. it could be a matching problem--could photograph known meds and use the photos as the labels.
  3. **Thermometer reading:** In a related problem, i'd like to be able to take a picture of a thermometer (digital or analogue) and record it's reading.

# Key concepts

- Alignment
  - Hough transform
  - RANSAC
  - ICP



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

