Alignment and Object Instance Recognition

Dr. Sander Ali Khowaja,
Assistant Professor, Department of Telecommunication Engineering
Faculty of Engineering and Technology, University of Sindh, Pakistan
Senior Member, IEEE – Member, ACM
https://sander-ali.github.io

Computer Vision & Image Processing

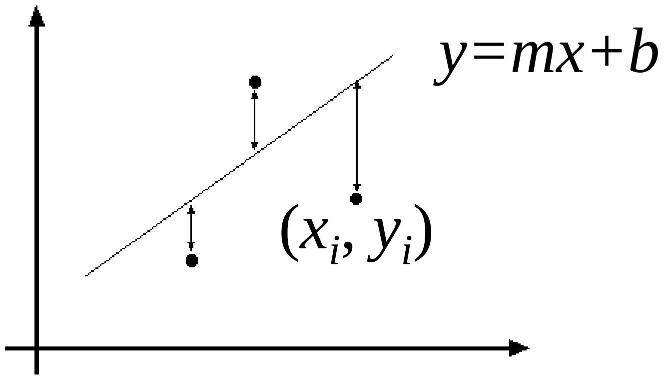
Previous Class

- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

Least Square Line Fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m\,x_i + b$
- Find (m, b) to minimize

$$\boxed{E = \sum_{i=1}^{n} (y_i - mx_i - b)^2}$$



$$E = \sum_{i=1}^{n} \left(\begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$

$$= \mathbf{y}^T\mathbf{y} - 2(\mathbf{Ap})^T\mathbf{y} + (\mathbf{Ap})^T(\mathbf{Ap})$$

$$\frac{dE}{dp} = 2\mathbf{A}^T\mathbf{Ap} - 2\mathbf{A}^T\mathbf{y} = 0$$

$$\boxed{\text{Matlab: } \texttt{p = A \textbackslash\ y;}}$$

$$\mathbf{A}^T\mathbf{Ap} = \mathbf{A}^T\mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{y}$$

Least Squares Line Fitting

```
function [m, b] = lsqfit(x, y)
% y = mx + b
% find line that best predicts y given x
% minimize sum_i (m*x_i + b - y_i).^2
A = [x(:) ones(numel(x), 1)];
b = y(:);
p = A\b;
m = p(1);
b = p(2);
```

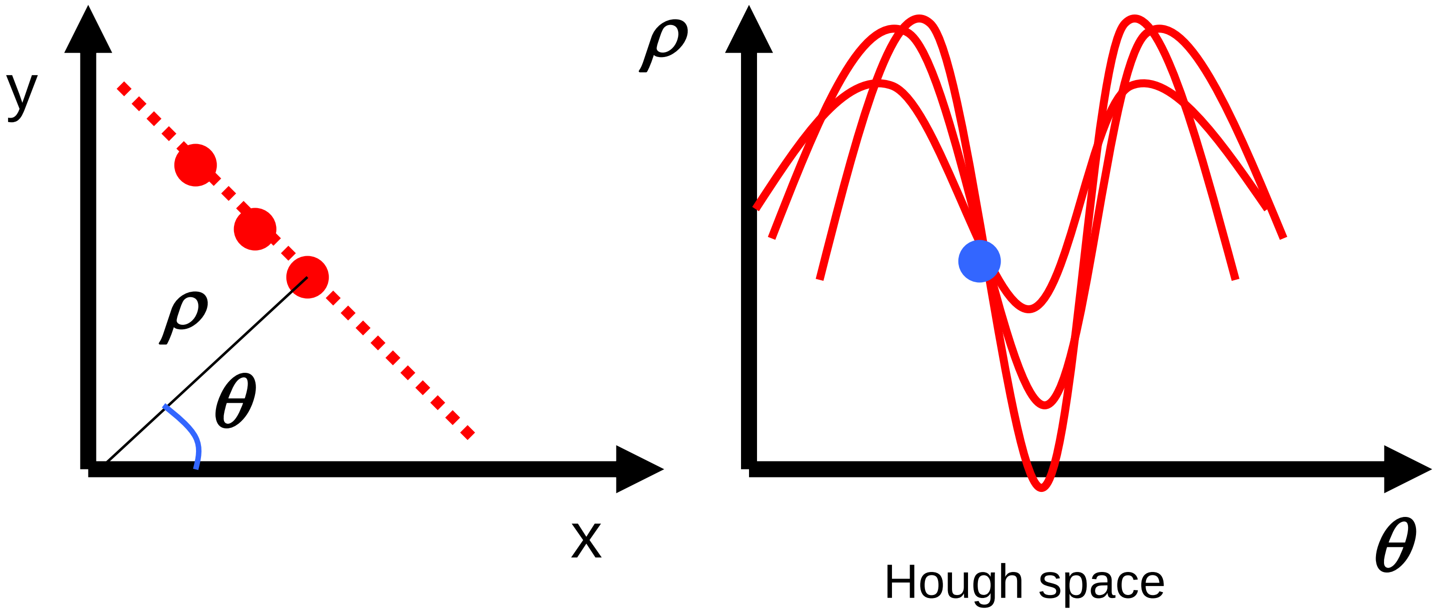
$$\left\| \underbrace{\begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}}_{\mathrm{A}} \underbrace{\begin{bmatrix} m \\ b \end{bmatrix}}_{\mathrm{p}} - \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}}_{\mathrm{y}} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$

Hough Transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Use a polar representation for the parameter space

$$x\cos\theta + y\sin\theta = \rho$$

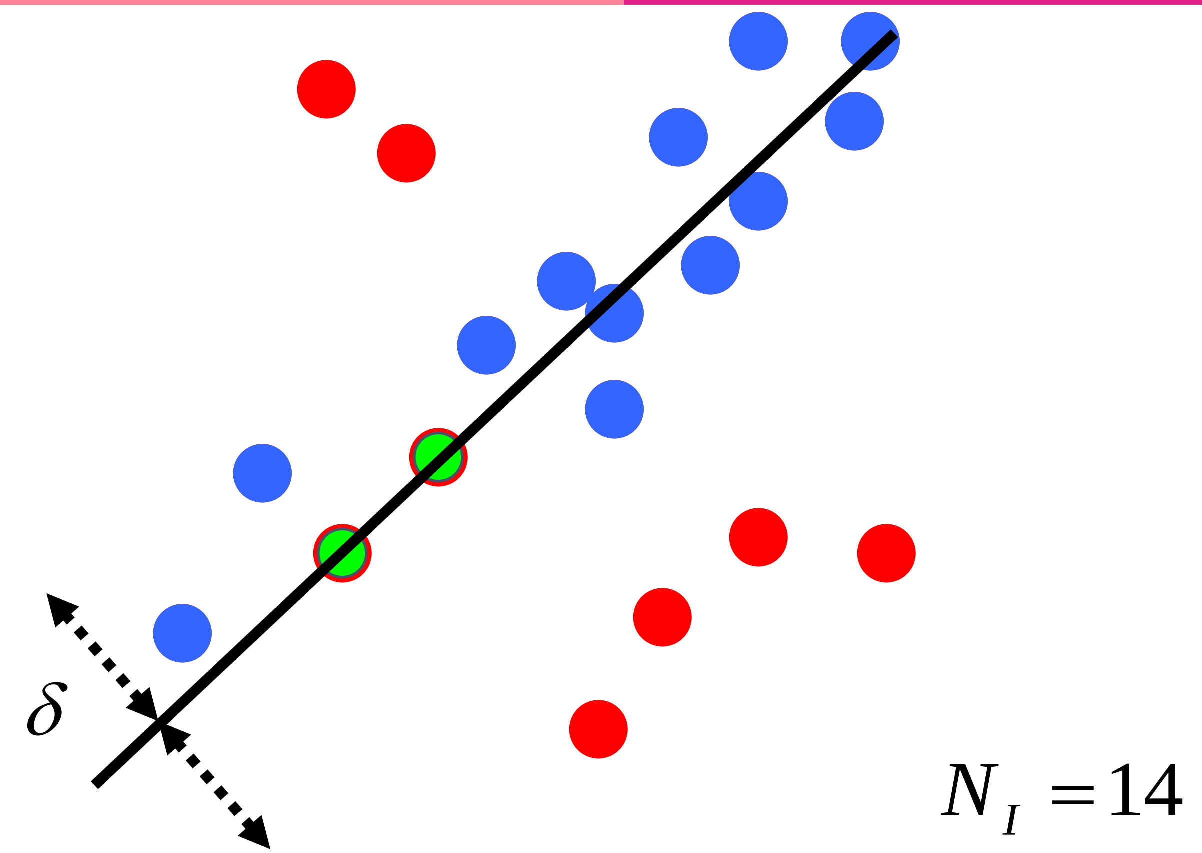


Slide from S. Savarese

Hough Transform

```
function [m, b] = houghfit(x, y)
% y = mx + b
% x*cos(theta) + y*sin(theta) = r
% find line that best predicts y given x
% minimize sum_i (m*x_i + b - y_i).^2
thetas = (-pi+pi/50):(pi/100):pi;
costhetas = cos(thetas);
sinthetas = sin(thetas);
minr = 0; stepr = 0.005; maxr = 1;

% count hough votes
counts = zeros(numel(thetas),(maxr-minr)/stepr+1);
for k = 1:numel(x)
    r = x(k)*costhetas + y(k)*sinthetas;
  % only count parameters within the range of r
  inrange = find(r >= minr & r <= maxr);
  rnum = round((r(inrange)-minr)/stepr)+1;
  ind = sub2ind(size(counts), inrange, rnum);
  counts(ind) = counts(ind) + 1;
end
```

```
% smooth the bin counts
counts = imfilter(counts,
fspecial('gaussian', 5, 0.75));

% get best theta, rho and show counts
[maxval, maxind] = max(counts(:));
[thetaind, rind] = ind2sub(size(counts),
maxind);
theta = thetas(thetaind);
r = minr + stepr*(rind-1);

% convert to slope-intercept
b = r/sin(theta);
m = -cos(theta)/sin(theta);
```



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

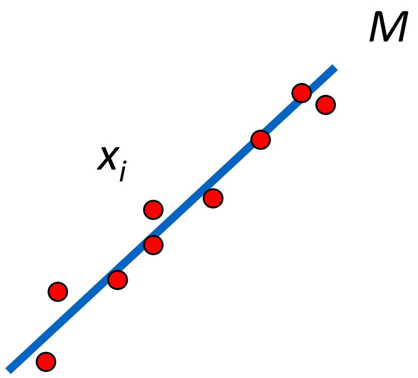
```
function [m, b] = ransacfit(x, y)
% y = mx + b
N = 200;
thresh = 0.03;
bestcount = 0;

for k = 1:N
    rp = randperm(numel(x));
    tx = x(rp(1:2));
    ty = y(rp(1:2));
    m = (ty(2)-ty(1)) ./ (tx(2)-tx(1));
    b = ty(2)-m*tx(2);

    nin = sum(abs(y-m*x-b)<thresh);
    if nin > bestcount
        bestcount = nin;
        inliers = (abs(y - m*x - b) < thresh);
    end
end
% total least square fitting on inliers
[m, b] = total_lsqfit(x(inliers), y(inliers));
```

Which algorithm should I use?

- ✓ If we know which points belong to the line, how do we find the "optimal" line parameters?
  - ✓Least squares
- ✓ What if there are outliers?
  - ✓Robust fitting, RANSAC
- What if there are many lines?
  - Voting methods: RANSAC, Hough transform

Alignment vs Fitting

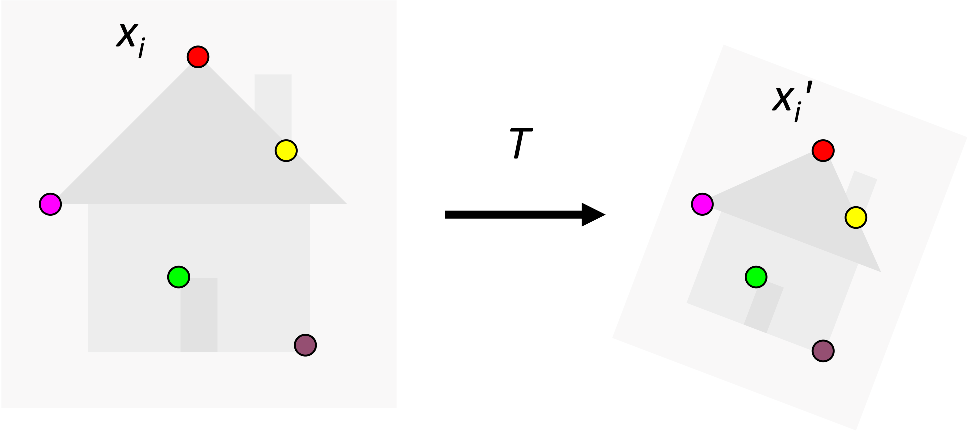
- Previous lectures: fitting a model to features in one image



Find model M that minimizes

$$\sum_i \text{residual}(x_i, M)$$

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



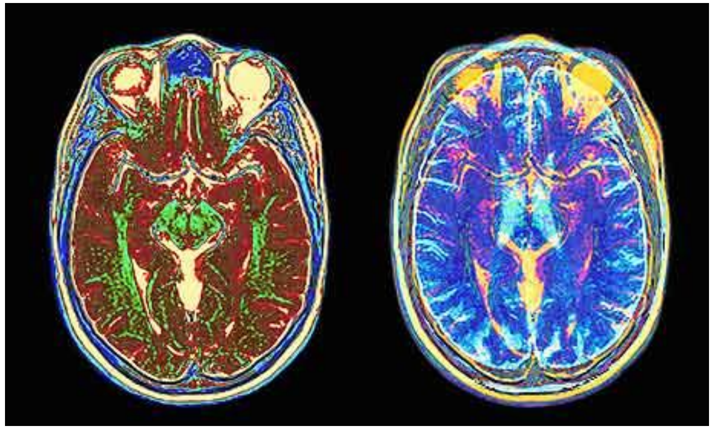
Find transformation T that minimizes

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

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

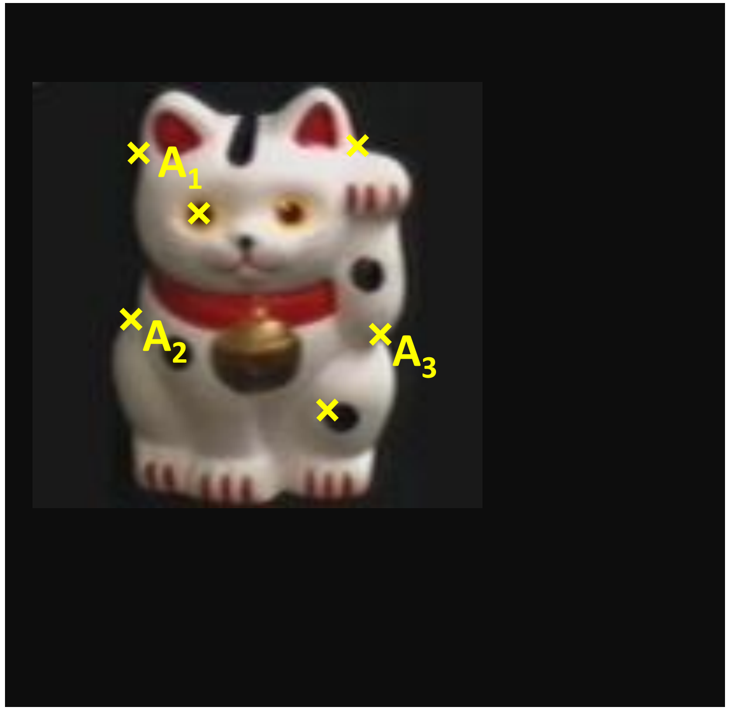
Robotics: match point clouds

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

Solving for translation: Example



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

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

Solving for Translation: Example

(t_x, t_y)

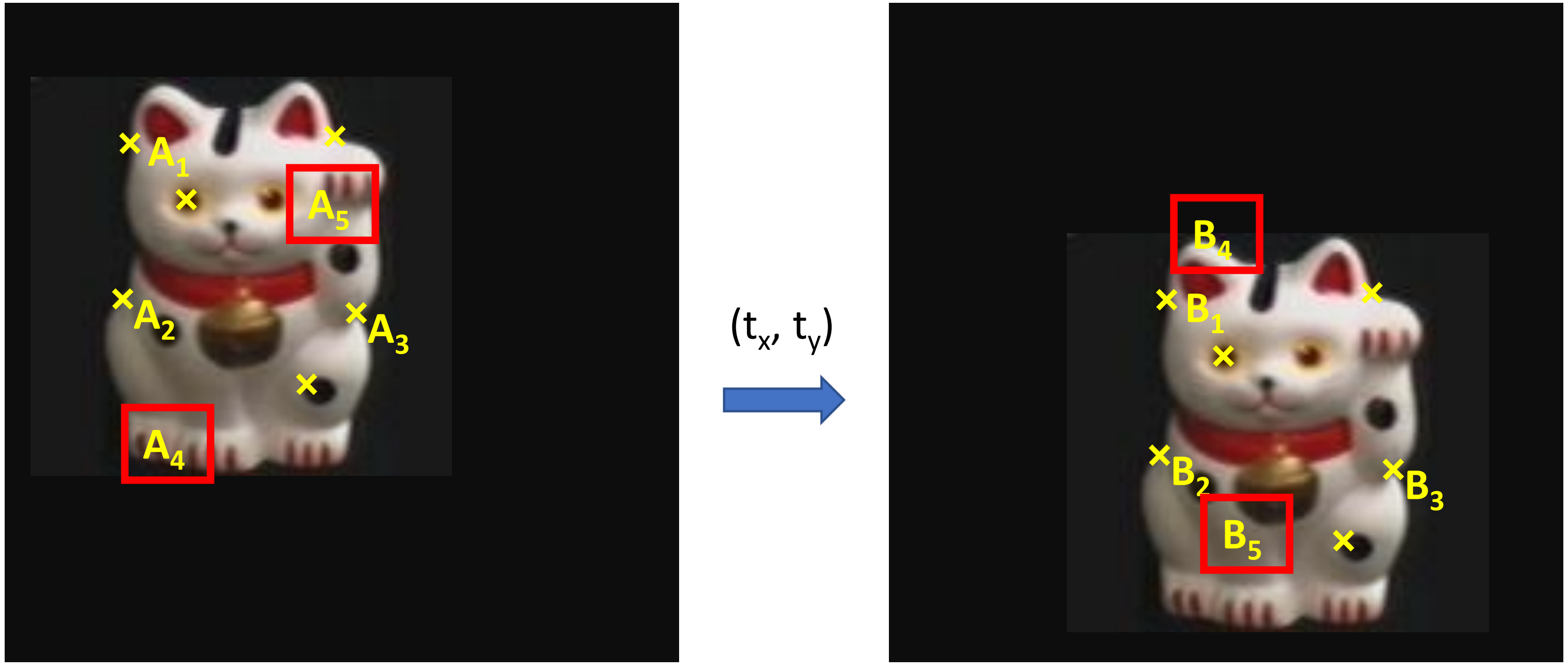
Least squares solution

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

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

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

Solving for Translation: Example



Problem: outliers

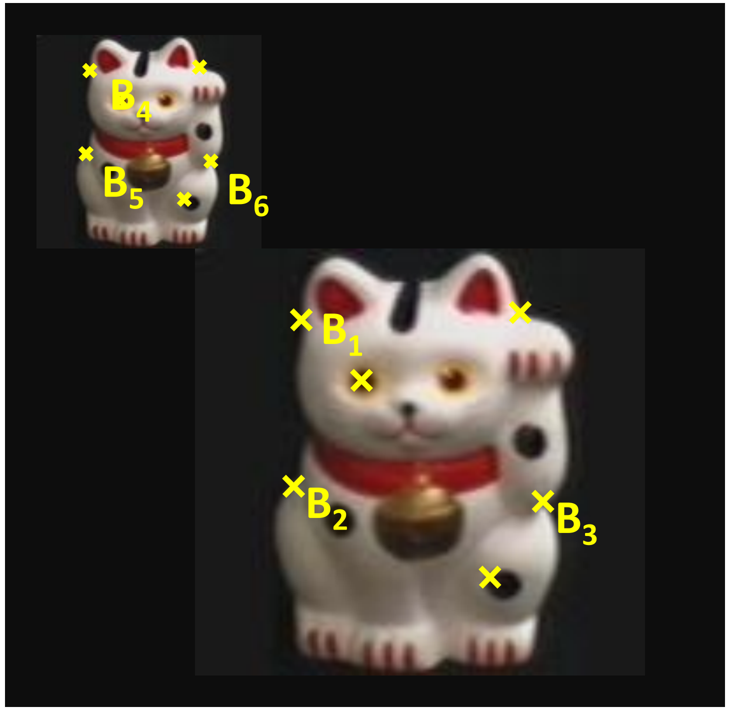
RANSAC solution

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

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

Solving for Translation: Example

(t_x, t_y)



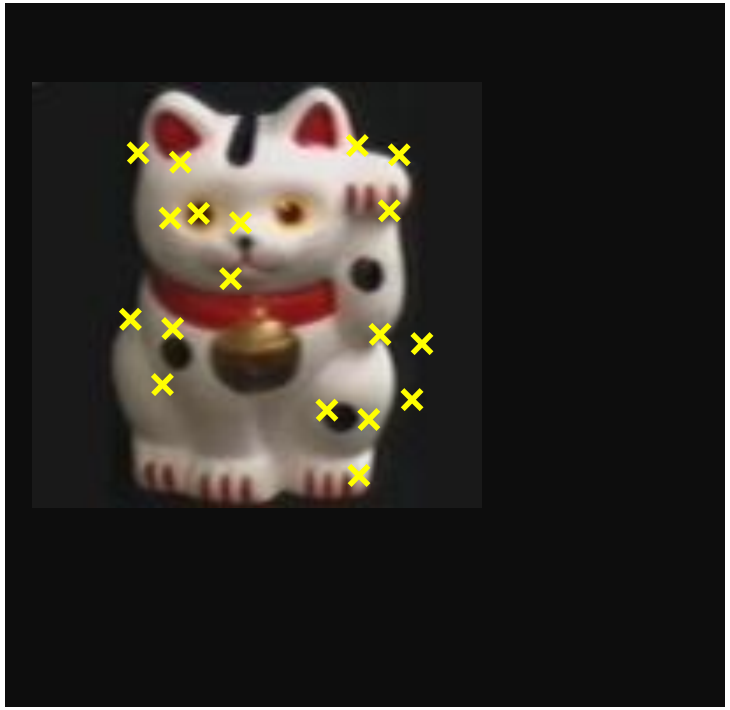
Problem: outliers, multiple objects, and/or many-to-one matches

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

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

Solving for Translation: Example



(t_x, t_y)

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

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

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

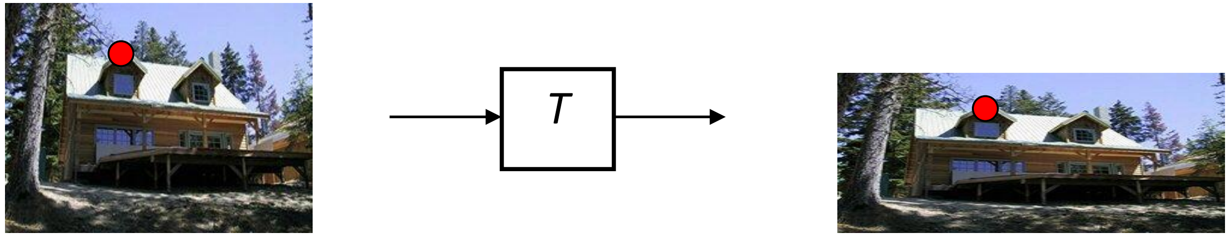
Alignment

- 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 30-50%)
  - Many-to-one matches or multiple objects

Parametric Global Warping



p = (x,y) **p'** = (x',y')

Transformation T is a coordinate-changing machine:

$$p' = T(p)$$

What does it mean that *T* is global?

- Is the same for any point p
- can be described by just a few numbers (parameters)

For linear transformations, we can represent T as a matrix

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

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

Common Transformations

original

Transformed

translation

rotation

aspect

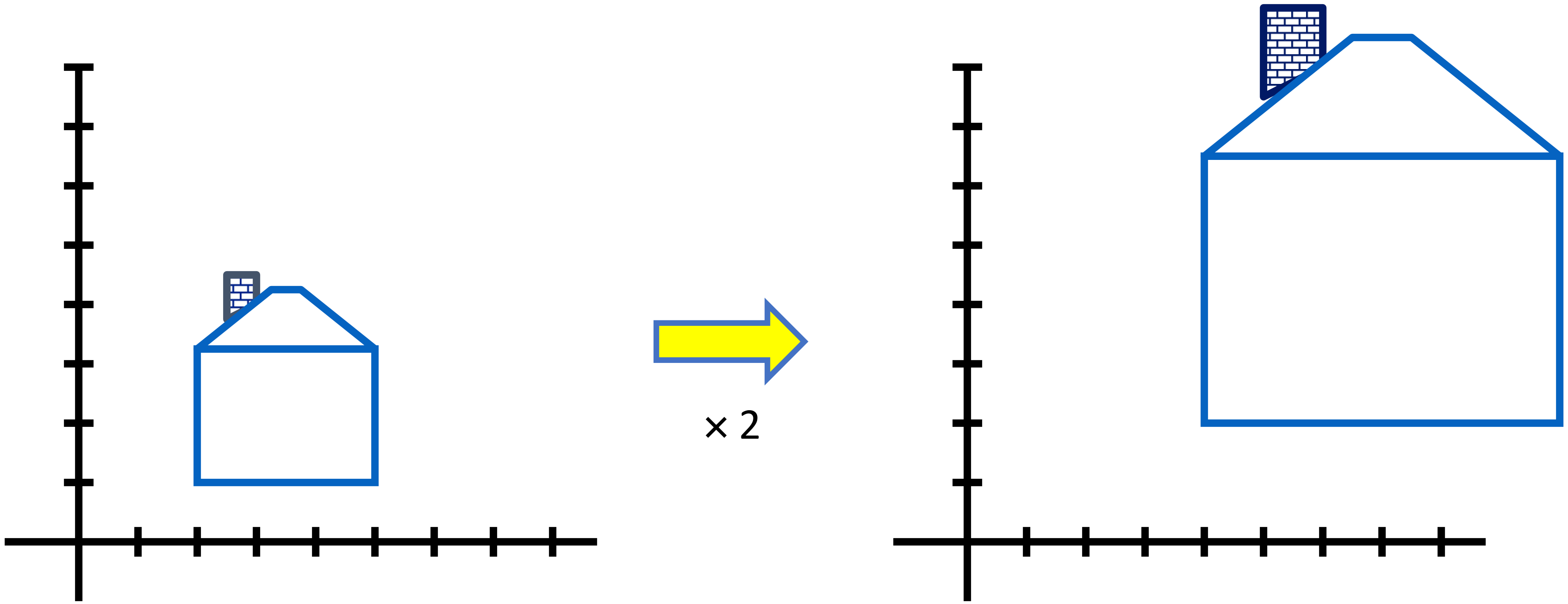
affine

perspective

Slide credit (next few slides):
A. Efros and/or S. Seitz

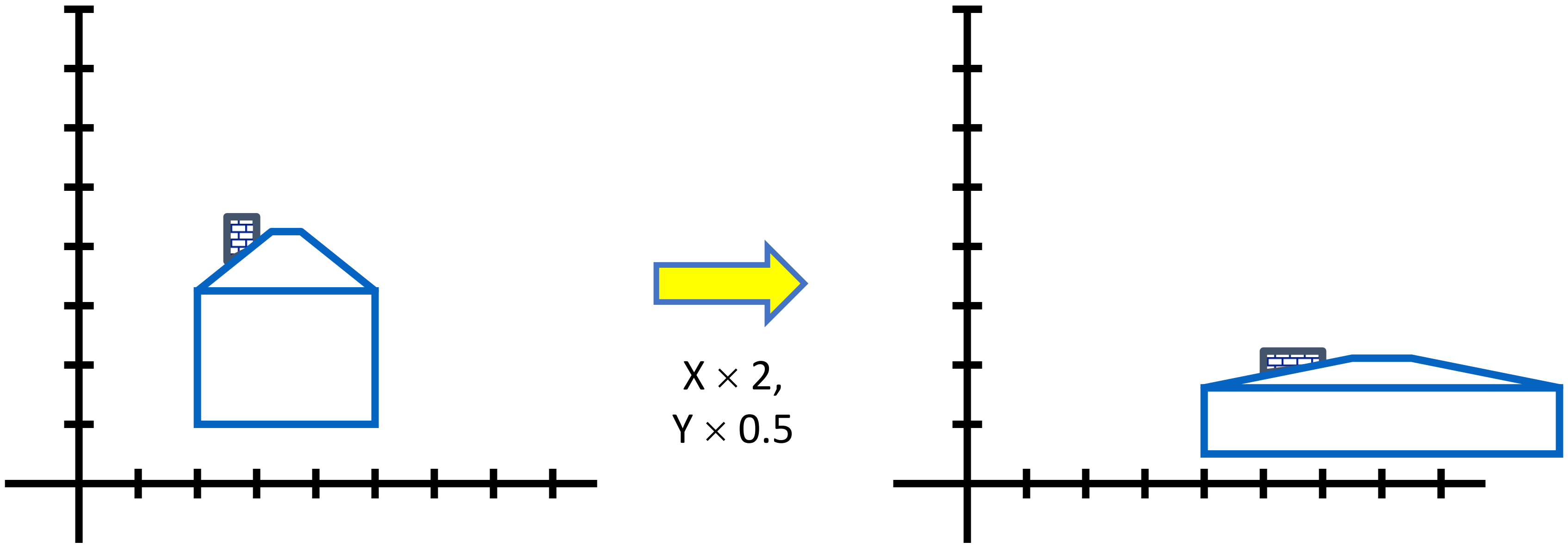
Scaling

- *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:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textit{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

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

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

or

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

Properties of affine transformations:

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

Projective Transformations

Projective transformations are combos of

- Affine transformations, and
- Projective warps

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

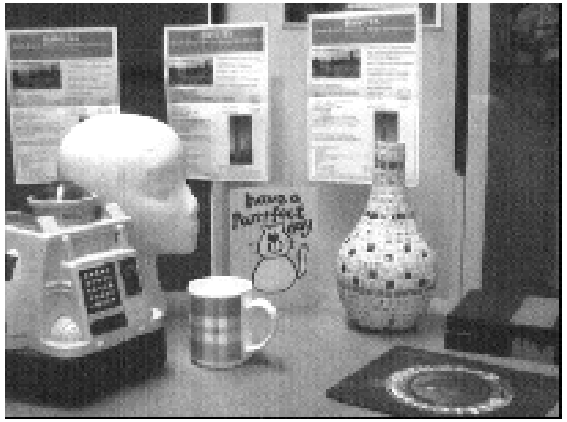
Properties of projective transformations:

- 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)

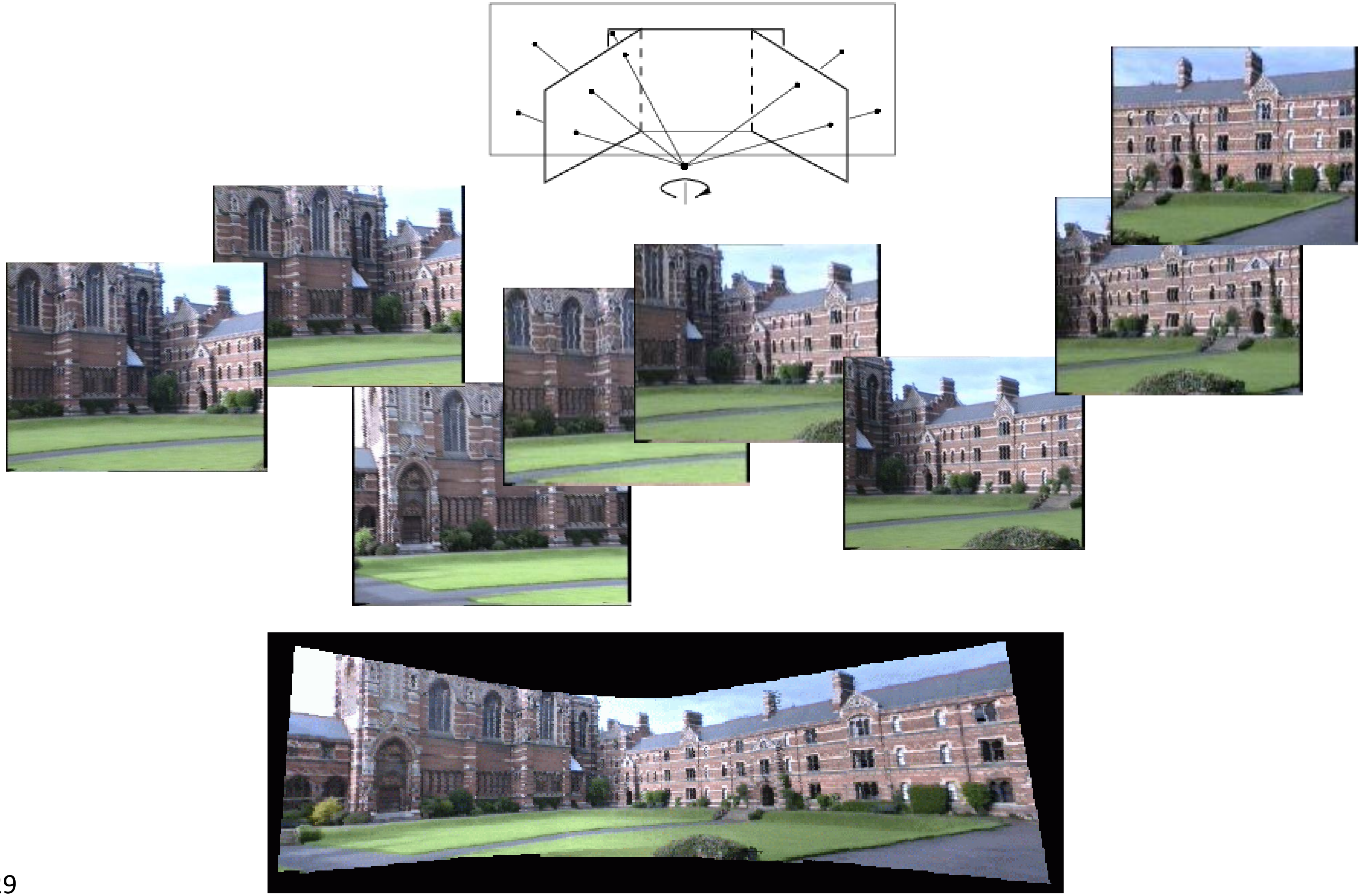
Projective Transformations (homography)

- The transformation between two views of a planar surface


- The transformation between images from two cameras that share the same center




Application: Panorama Stitching



Source: Hartley & Zisserman

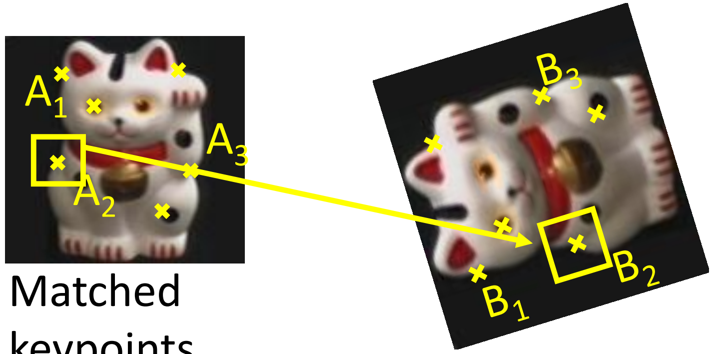
Application: Document Scanning

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

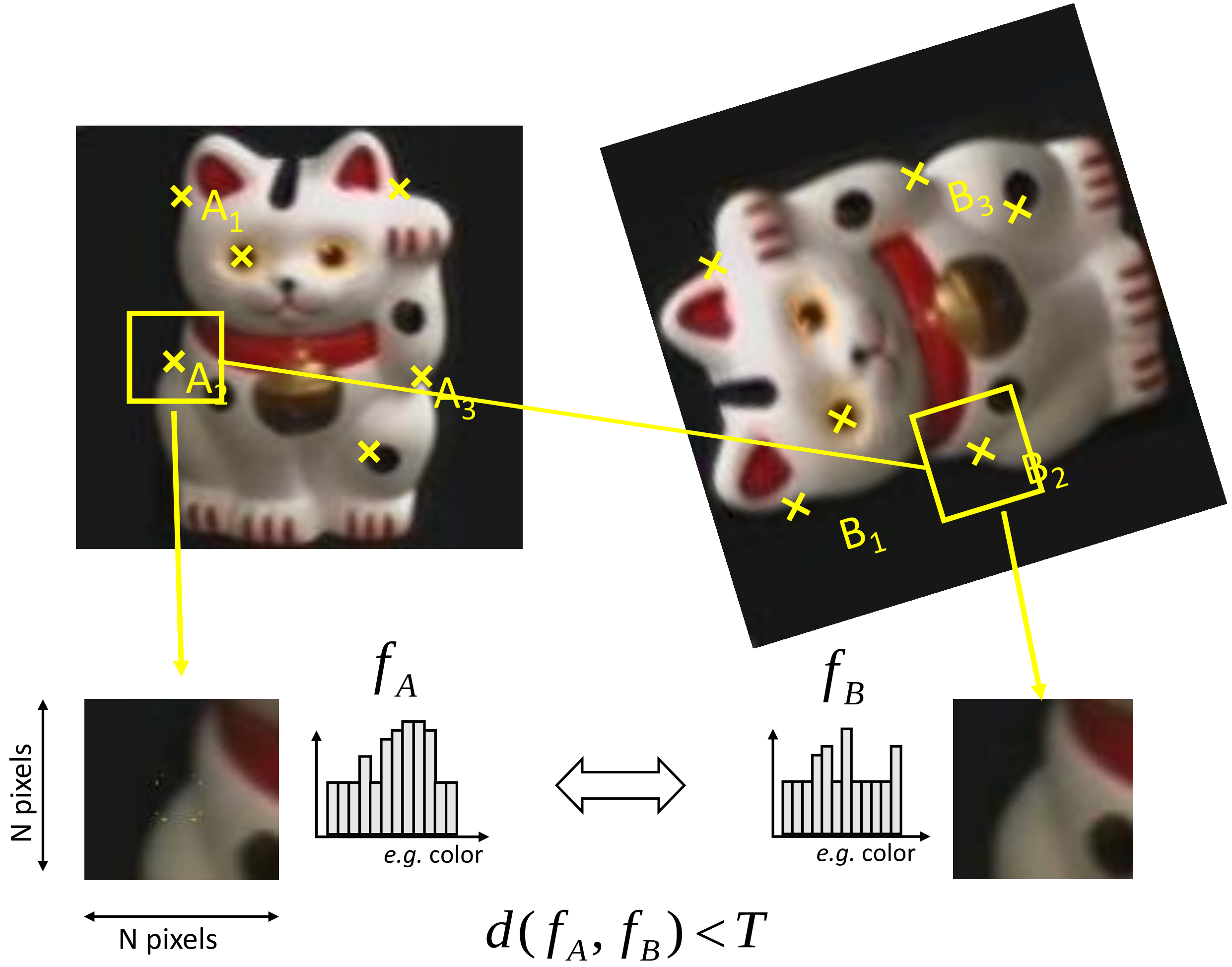
Affine Parameters

This Class

Inliers

Choose hypothesis with max score above threshold

Overview of Keypoint Matching

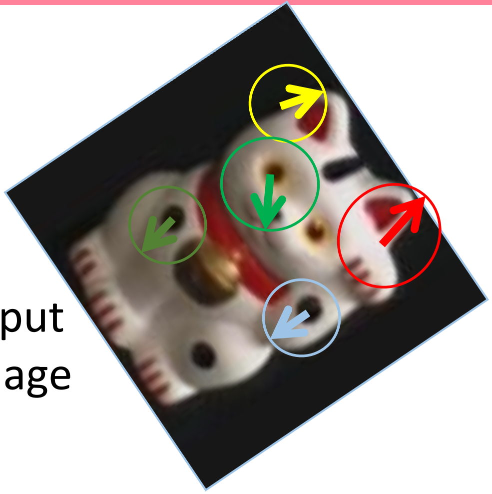


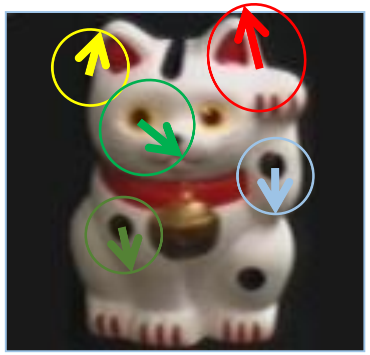
K. Grauman, B. Leibe

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**

Finding the objects (overview)

Input Image





Stored Image

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

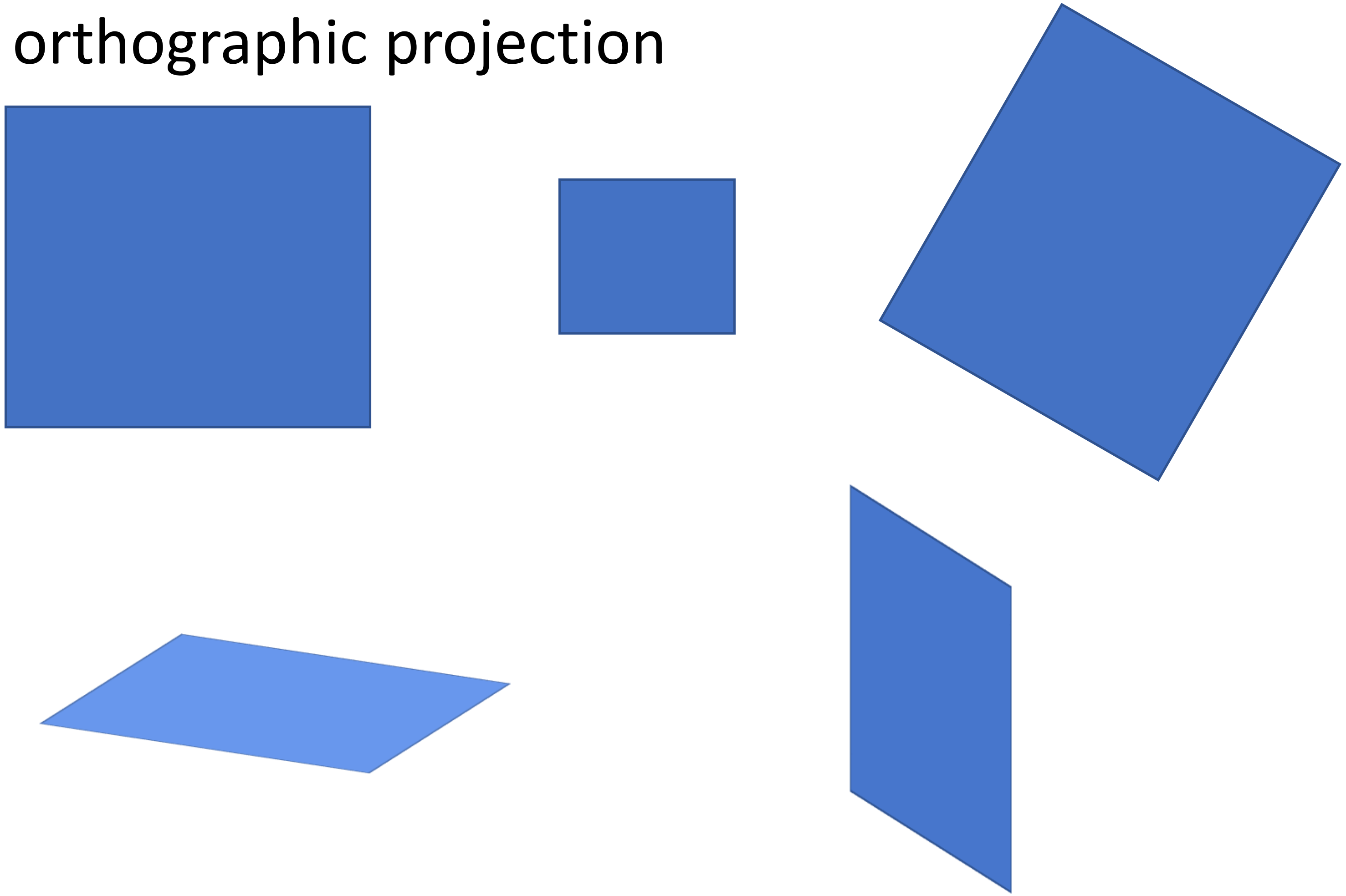
- Want to match keypoints between:
  1. Query image
  2. Stored image containing the object

- 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

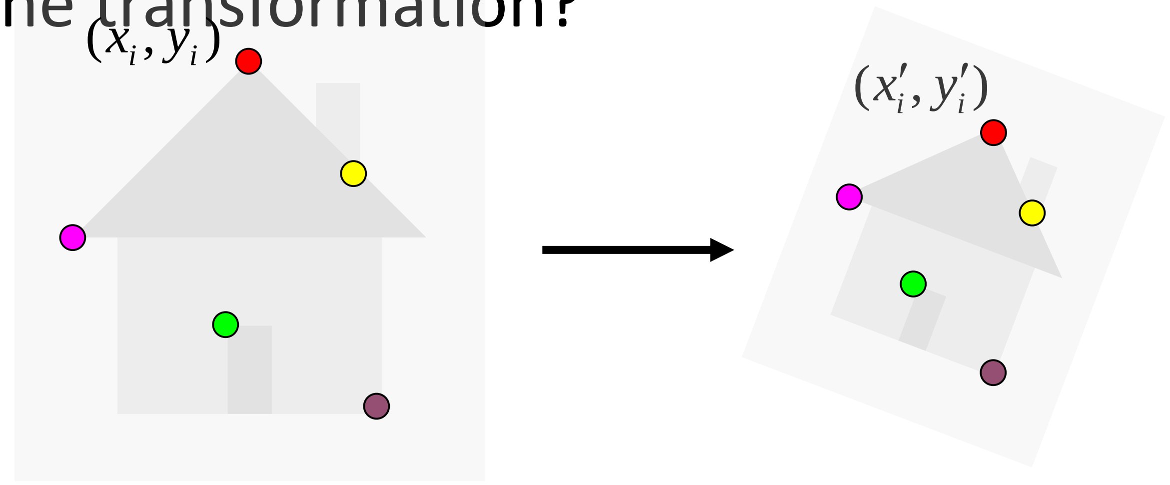
Affine Object Model

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



Fitting an Affine Transformation

- Assume we know the correspondences, how do we get the transformation?



$$\mathbf{x}_i' = \mathbf{M}\mathbf{x}_i + \mathbf{t}$$

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Want to find M, t to minimize

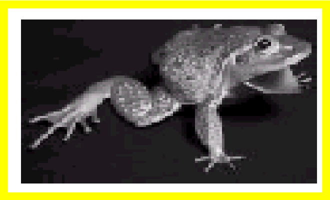
$$\sum_{i=1}^{n} \| \mathbf{x}_i' - \mathbf{M}\mathbf{x}_i - \mathbf{t} \|^2$$

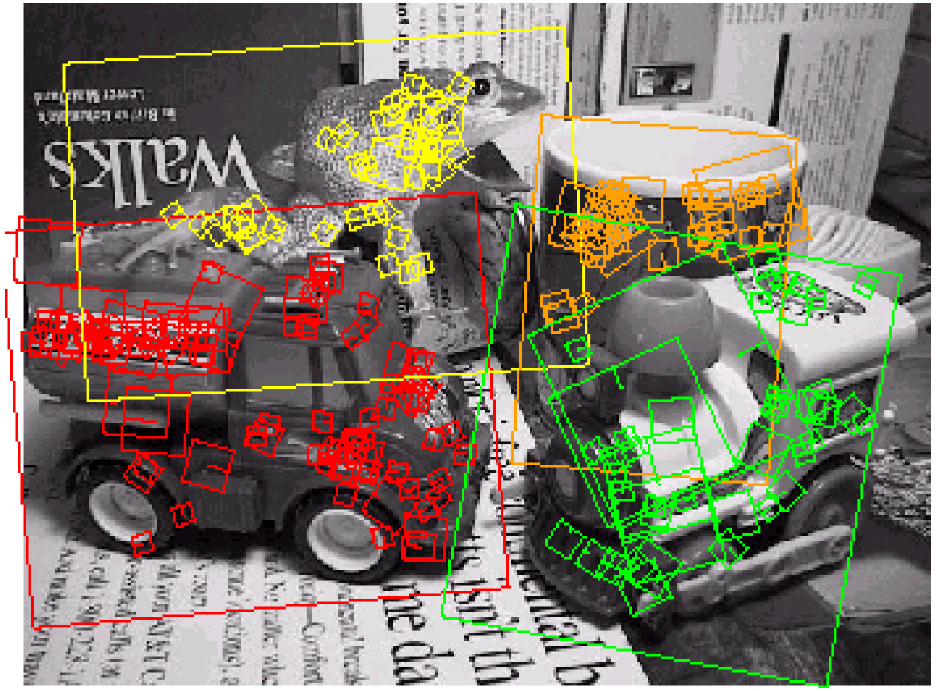
Finding the objects (in detail)

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)

Dr. Sander Ali Khowaja

Examples of Recognized Objects



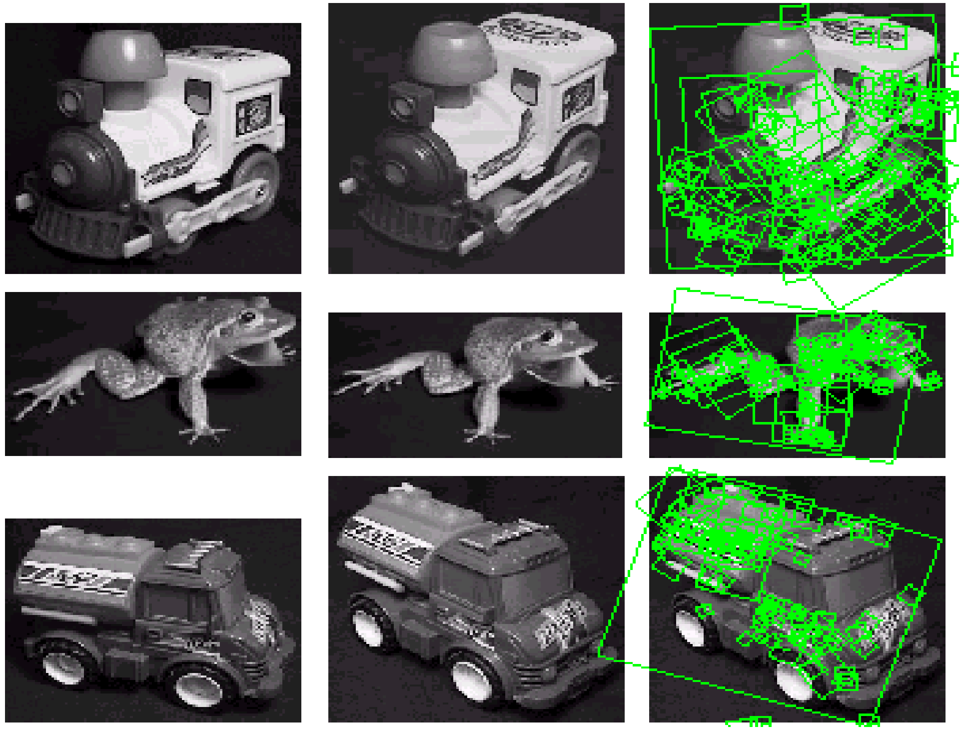


View Interpolation

- 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

  $\Rightarrow$ Use the known links to "transfer votes" to other viewpoints



[Lowe01]

Slide credit: David Lowe

Applications

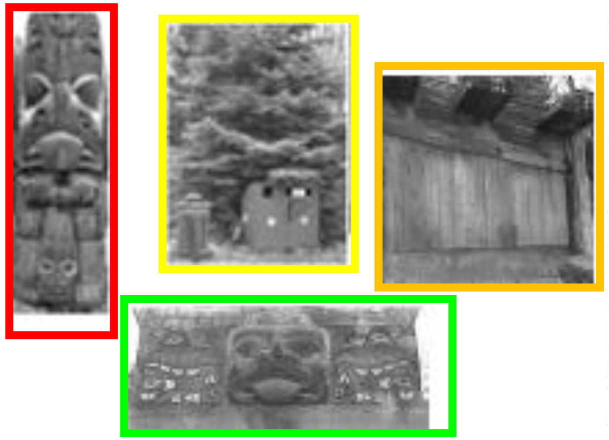
- Sony Aibo
  (Evolution Robotics)

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

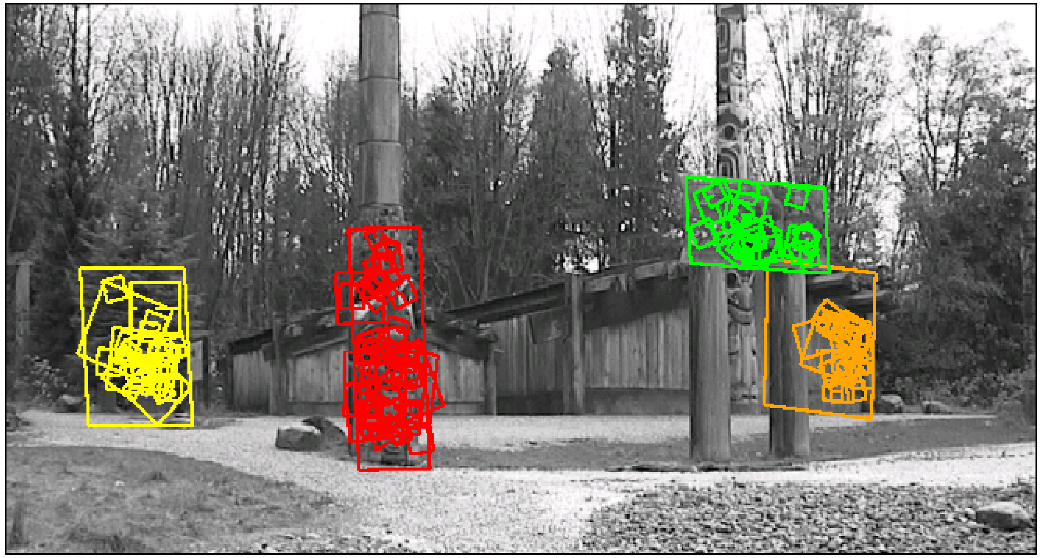
- Other uses
  - Place recognition
  - Loop closure in SLAM

K. Grauman, B. Leibe

Location Recognition



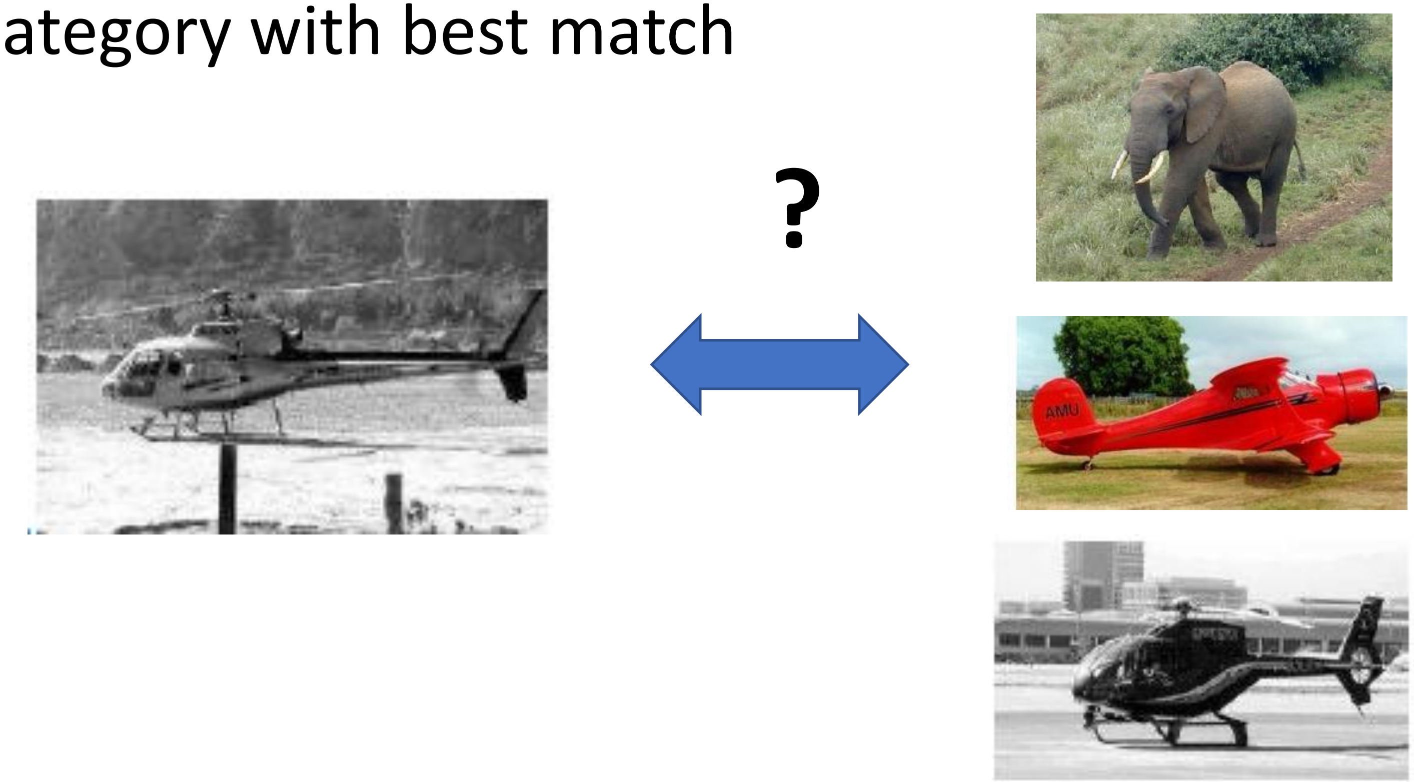
Training



[Lowe04]

Slide credit: David Lowe

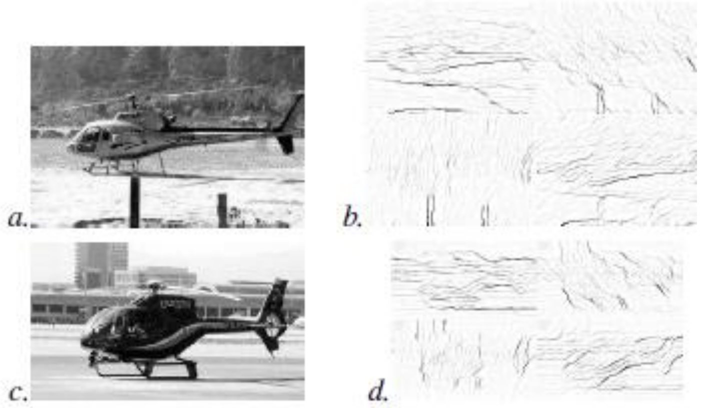
- Goal: identify what type of object is in the image
- Approach: align to known objects and choose category with best match



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

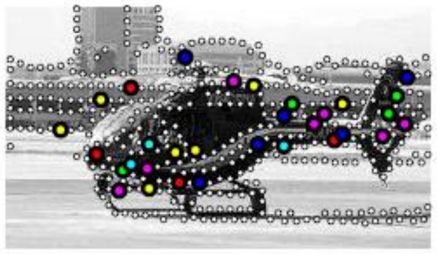
Summary of Algorithm

- 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

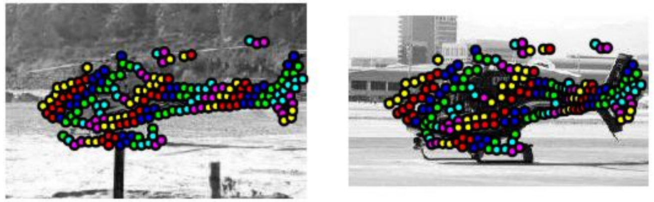


Input, Edge Maps

Geometric Blur

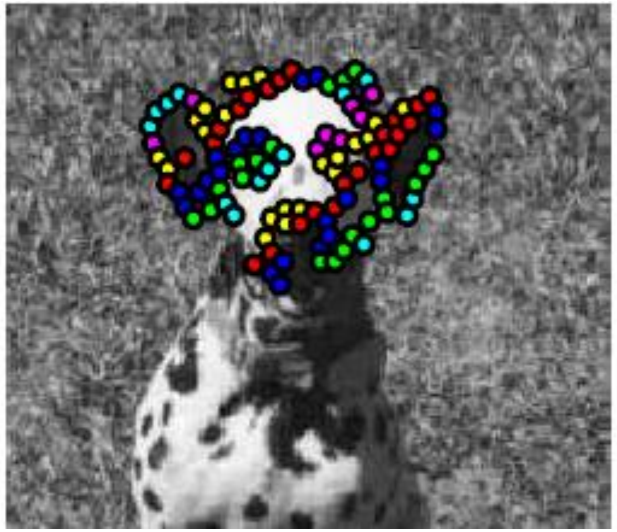
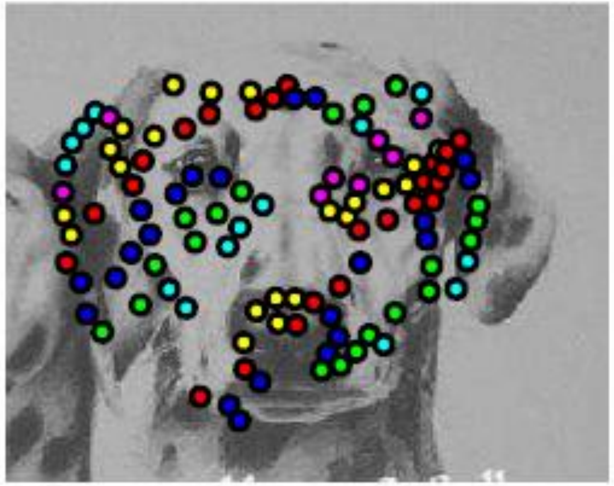
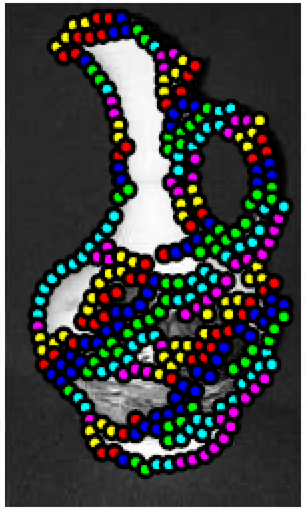
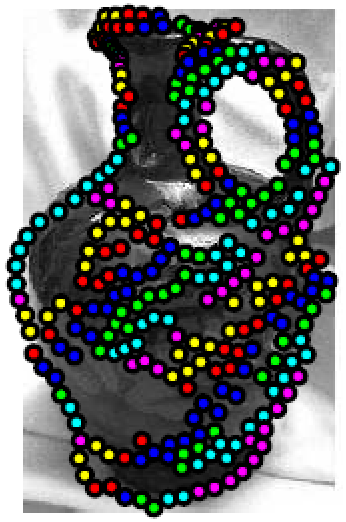


Feature Points

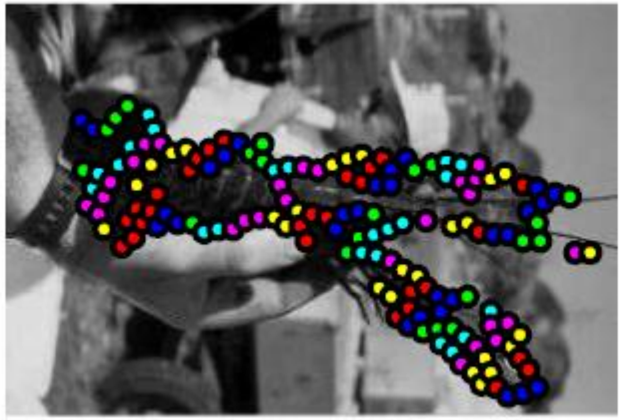


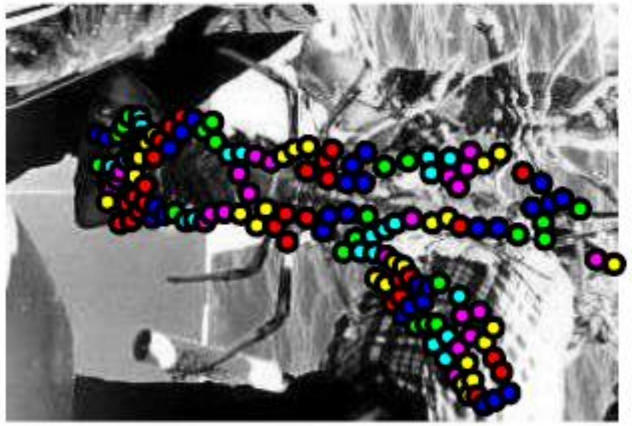
Correspondences

Example of Matches

Examples of Matches





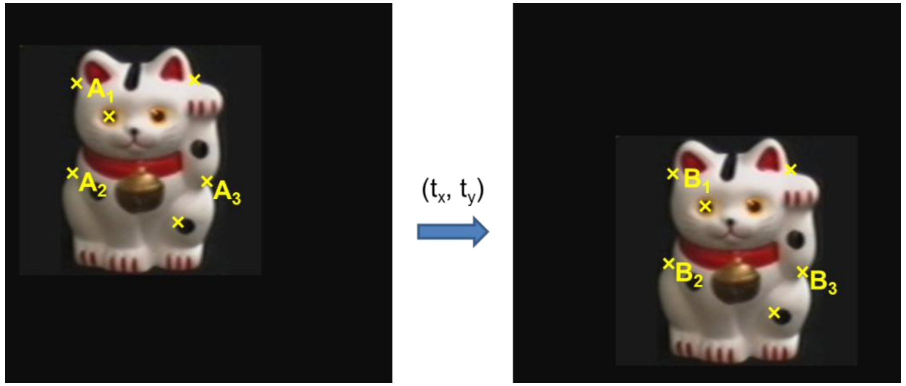
Other ideas worth being aware of

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

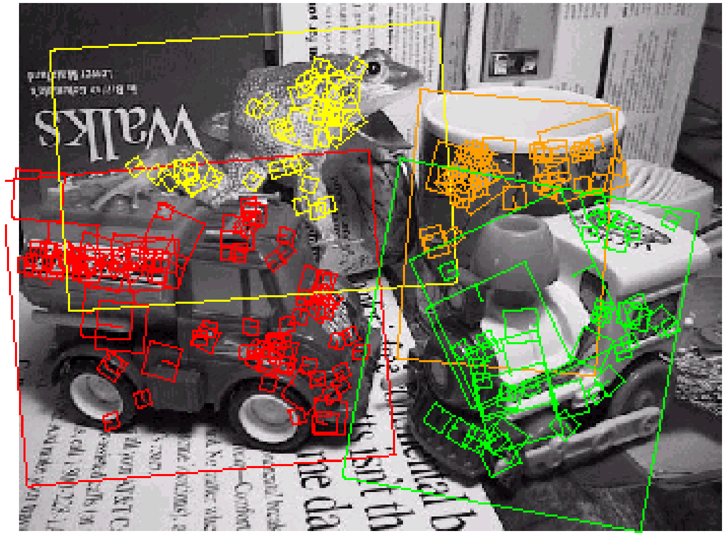
- Robust non-rigid point matching: A new point matching algorithm for non-rigid registration, CVIU 2003 (includes code, demo, paper)

Things to Remember

- Alignment
  - Hough transform
  - RANSAC
  - ICP

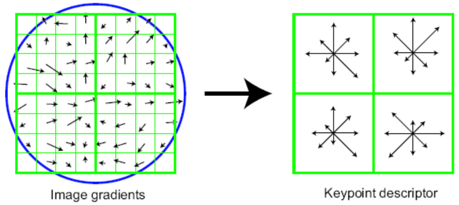


- Object instance recognition
  - Find keypoints, compute descriptors
  - Match descriptors
  - Vote for / fit affine parameters
  - Return object if # inliers > T

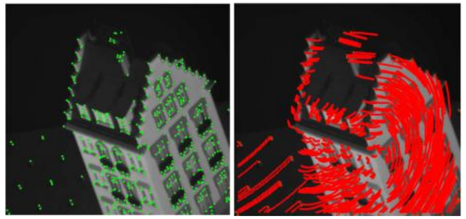


What have we learned

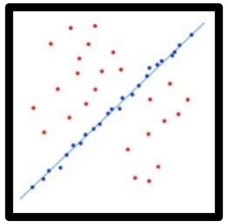
- **Interest points**
  - Find *distinct* and *repeatable* points in images
  - Harris-> corners, DoG -> blobs
  - SIFT -> feature descriptor



- **Feature tracking and optical flow**
  - Find motion of a keypoint/pixel over time
  - Lucas-Kanade:
    - brightness consistency, small motion, spatial coherence
  - Handle large motion:
    - iterative update + pyramid search



- **Fitting and alignment**
  - find the transformation parameters that best align matched points



- **Object instance recognition**
  - Keypoint-based object instance recognition and search