Fitting and Registration

Computer Vision

CS 543 / ECE 549

University of Illinois

Derek Hoiem

Announcements

- HW 1 due today

- HW 2 out on Thursday
  - Compute edges and find circles in image using Hough transform
  - Create dictionary of texture responses and use it to match texture images
  - Derive the EM algorithm for a mixture of multinomials
  - Estimate foreground and background color distributions using EM and segment the object using graph cuts

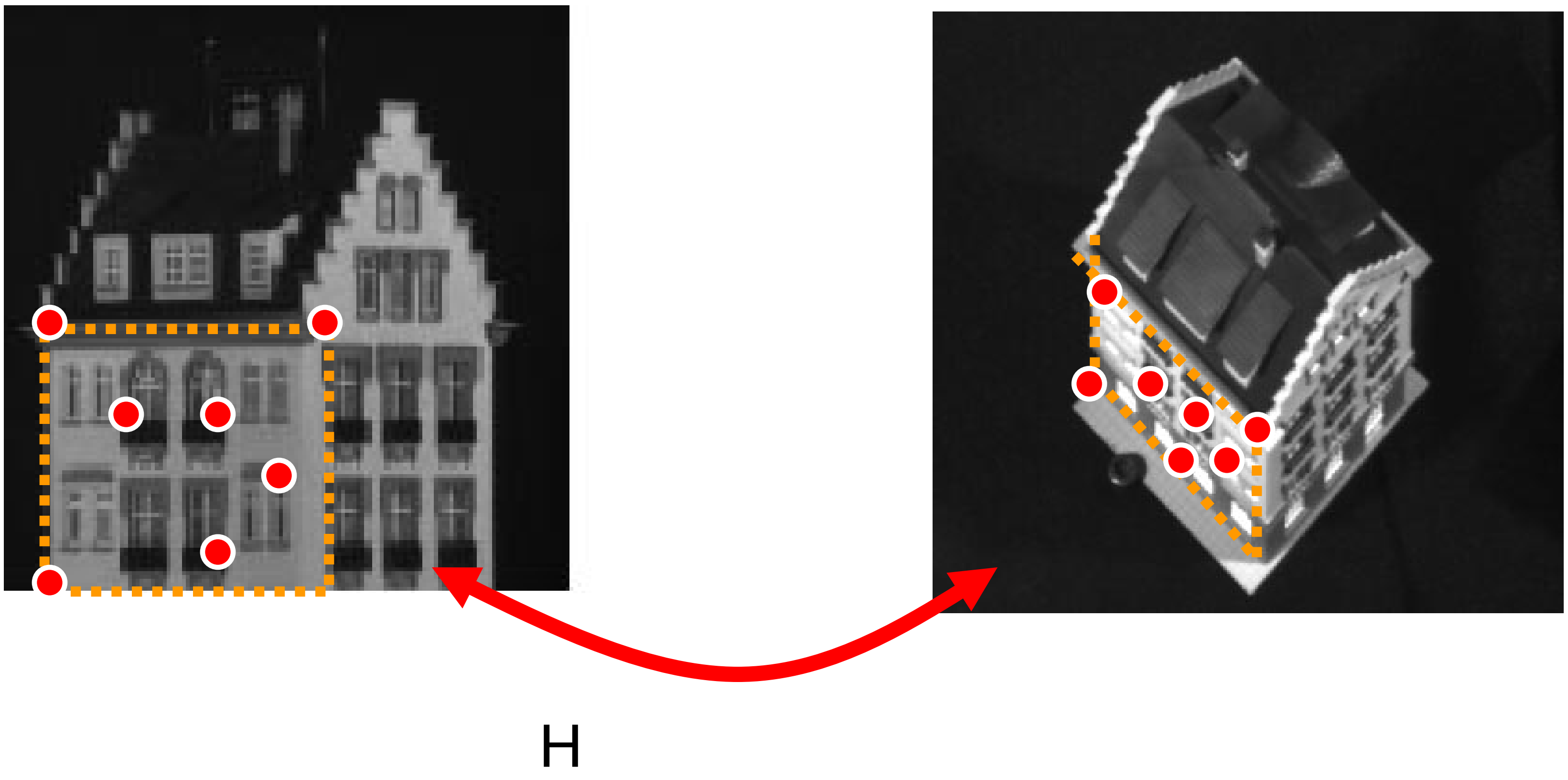
Fitting: find the parameters of a model that best fit the data

Alignment: find the parameters of the transformation that best align matched points

Example: Computing vanishing points

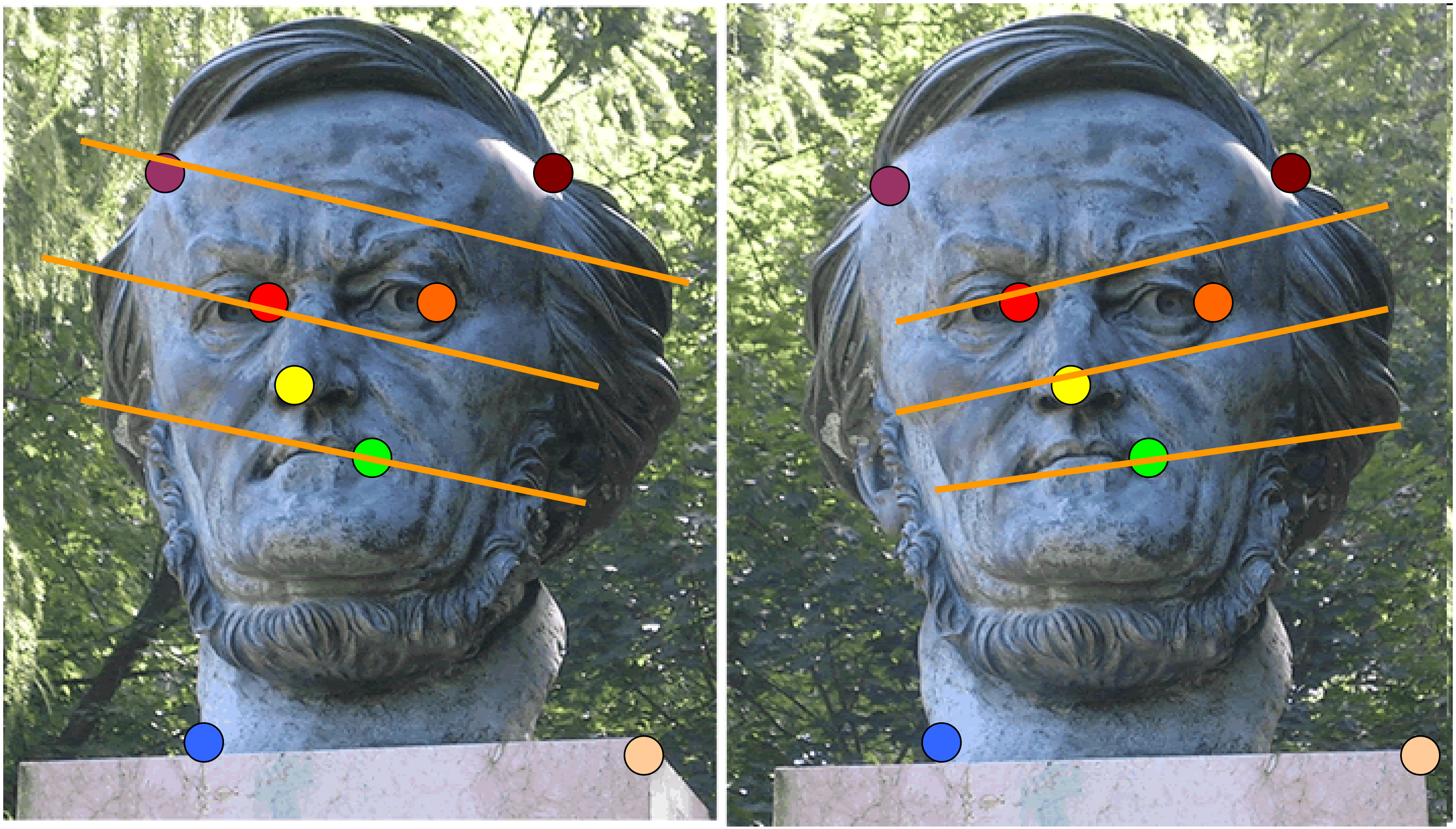
Slide from Silvio Savarese

Example: Estimating an homographic transformation



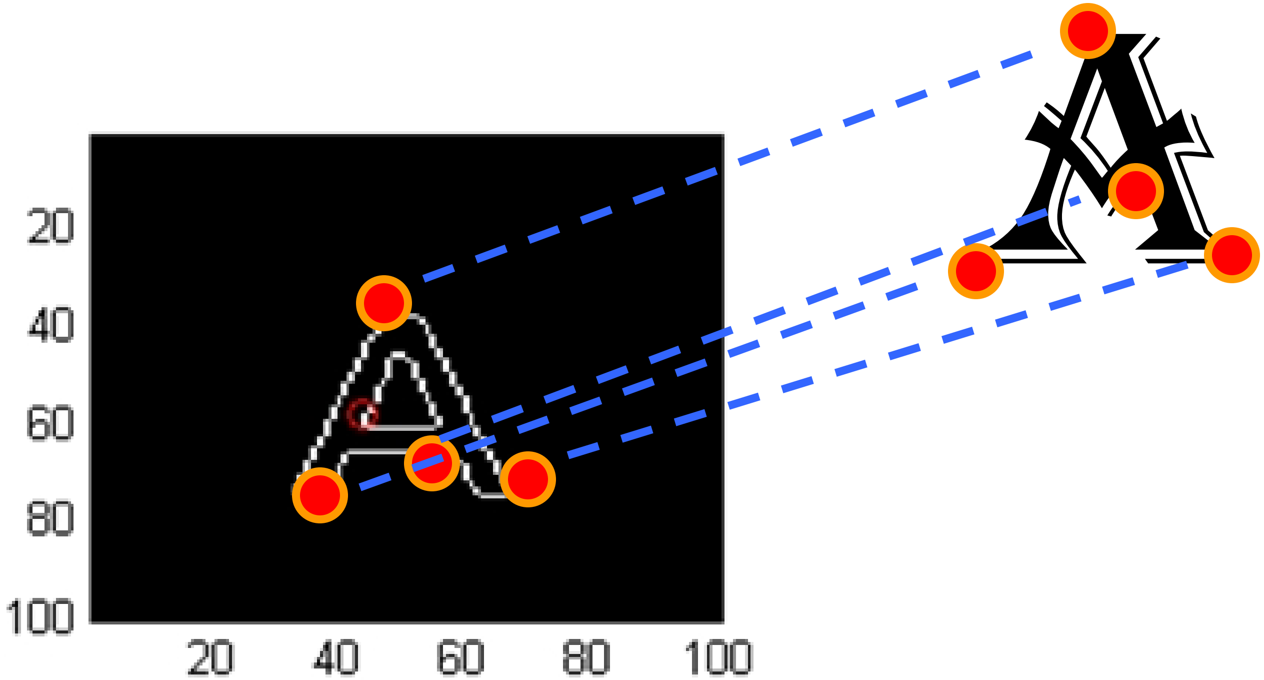
Slide from Silvio Savarese

Example: Estimating “fundamental matrix” that corresponds two views



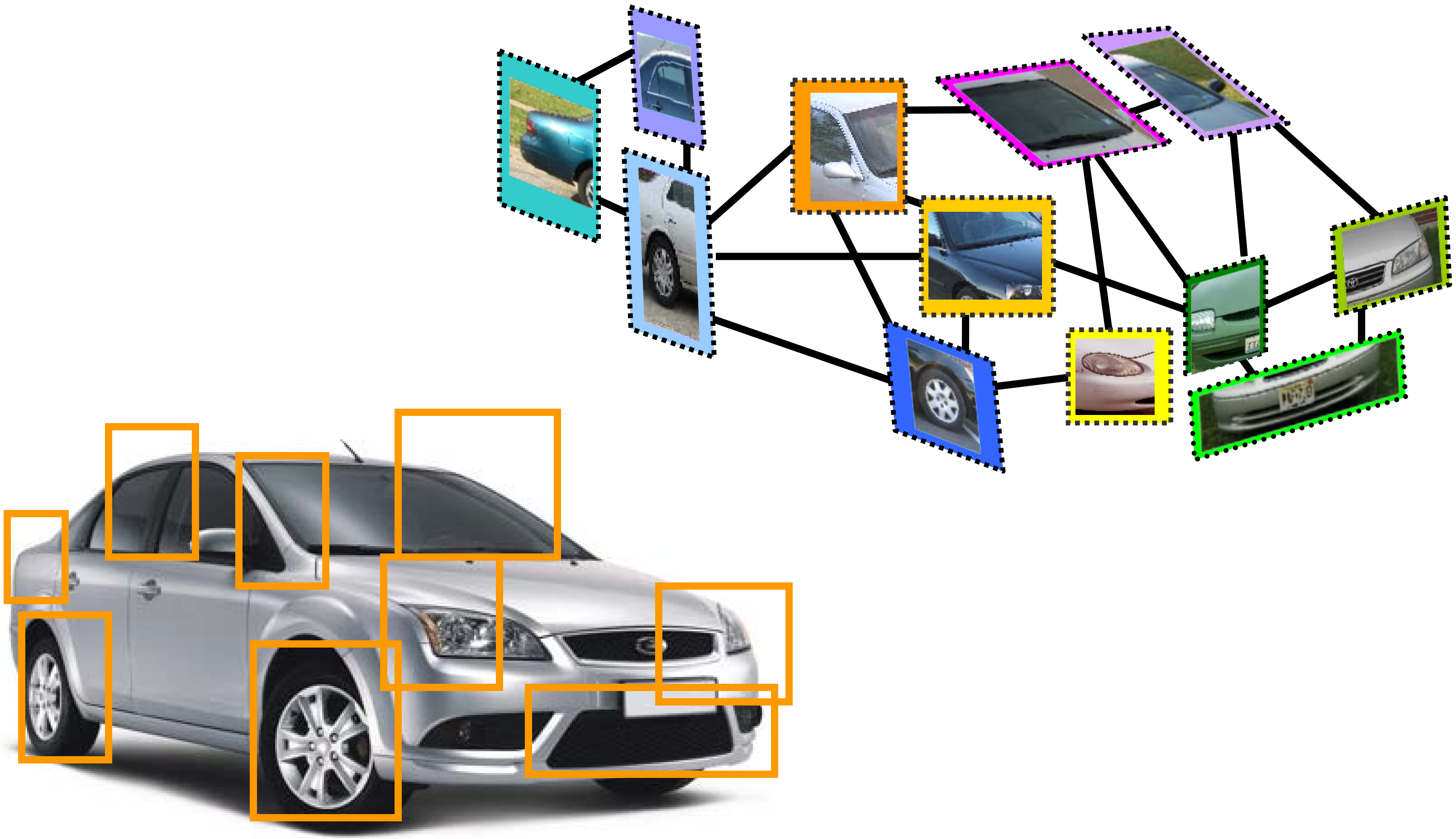
Slide from Silvio Savarese

Example: fitting an 2D shape template



Slide from Silvio Savarese

Example: fitting a 3D object model



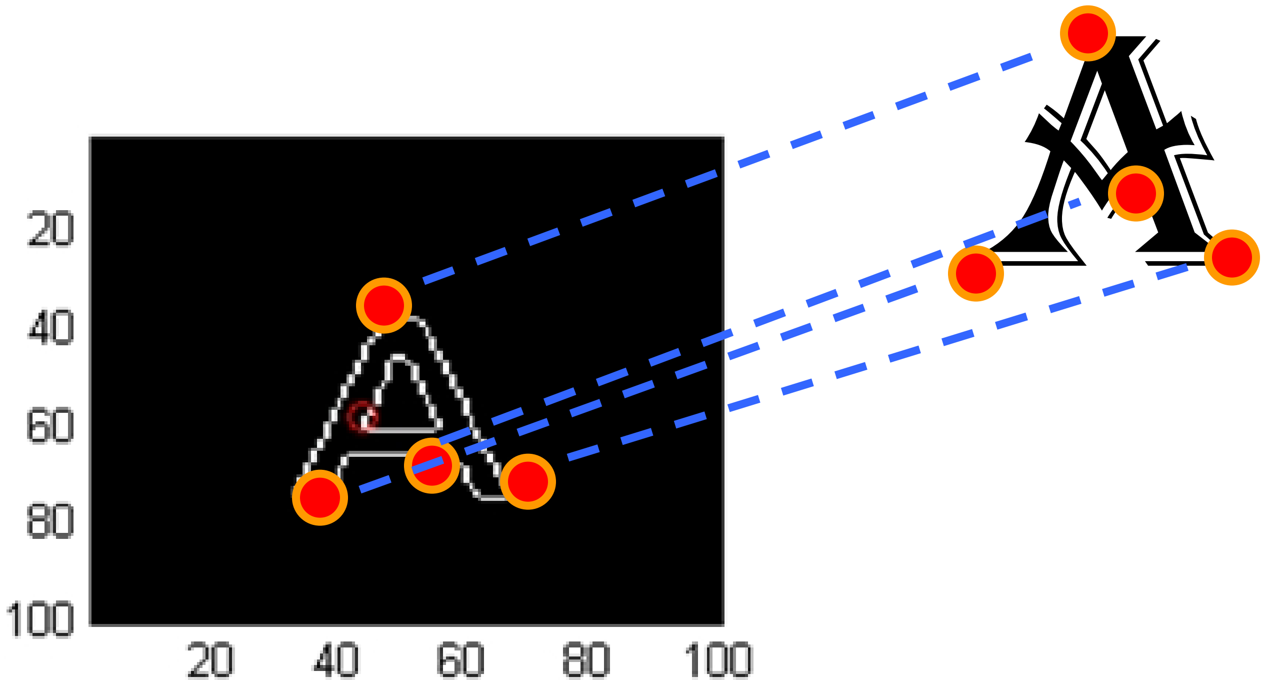
Slide from Silvio Savarese

Critical issues: noisy data

Slide from Silvio Savarese

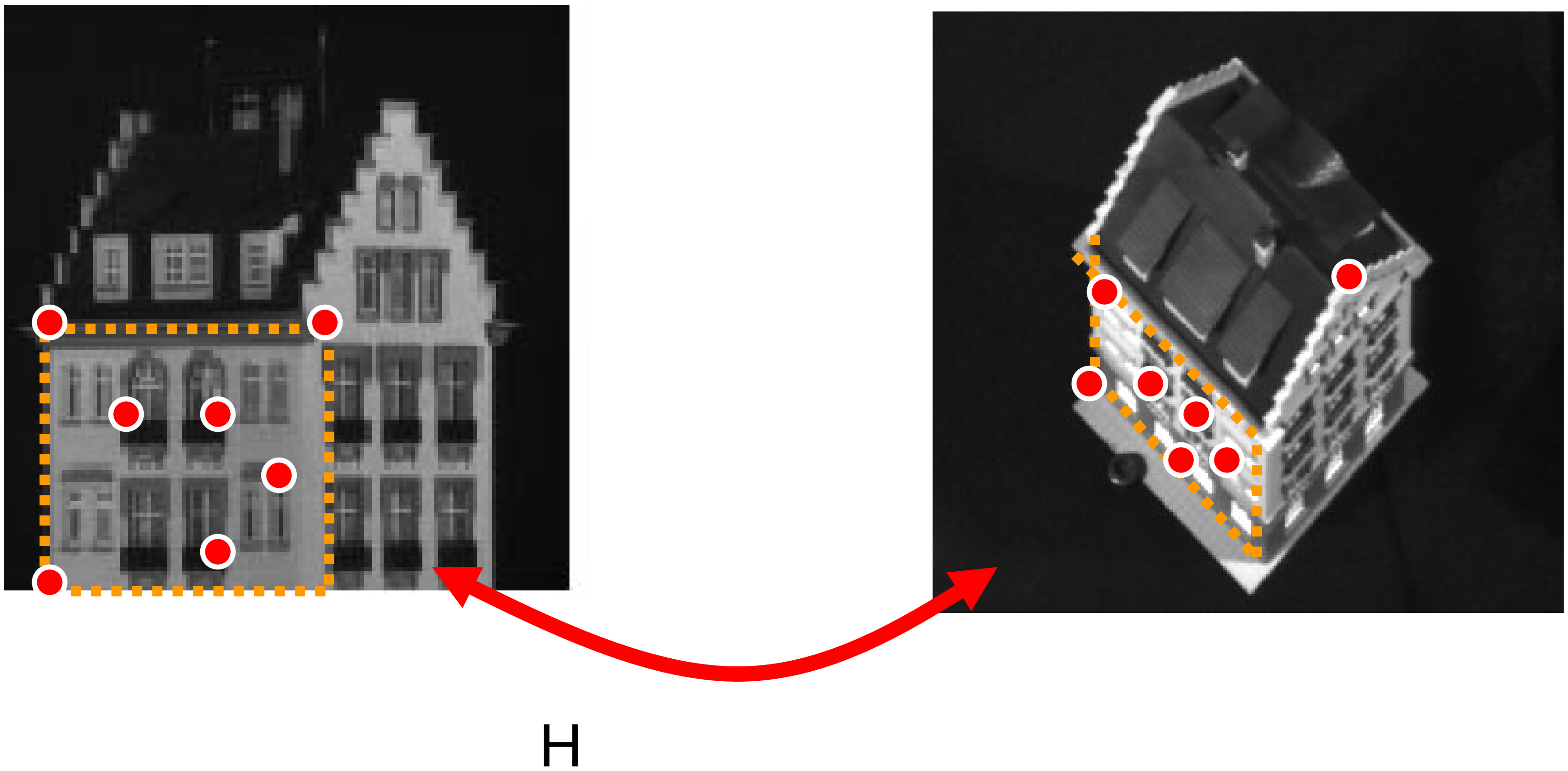
Critical issues: intra-class variability

"All models are wrong, but some are useful." Box and Draper 1979



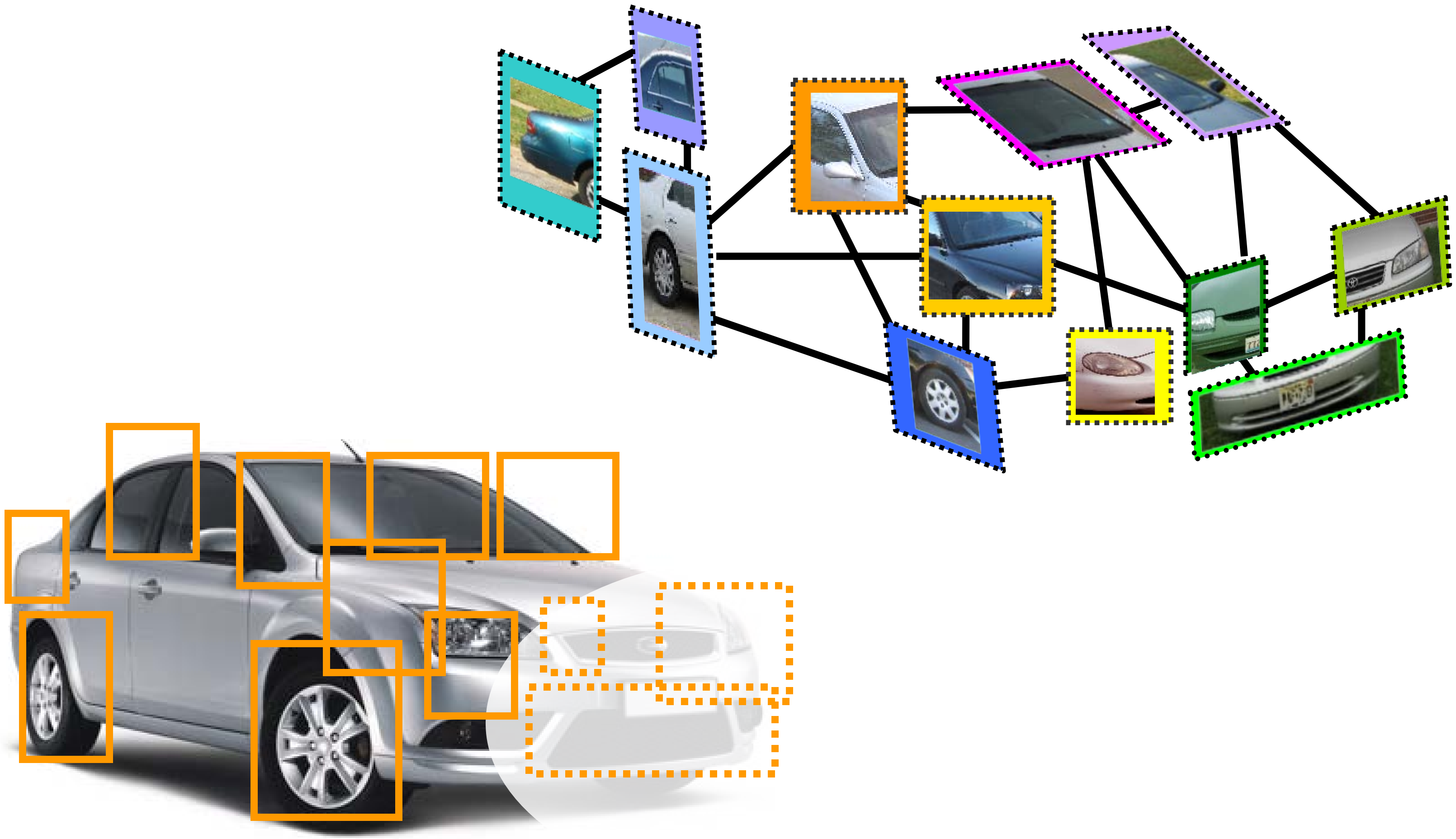
Slide from Silvio Savarese

Critical issues: outliers



Slide from Silvio Savarese

Critical issues: missing data (occlusions)



Slide from Silvio Savarese

Fitting and Alignment

- Design challenges
  - Design a suitable **goodness of fit** measure
    - Similarity should reflect application goals
    - Encode robustness to outliers and noise
  - Design an **optimization** method
    - Avoid local optima
    - Find best parameters quickly

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Iterative closest point (ICP)

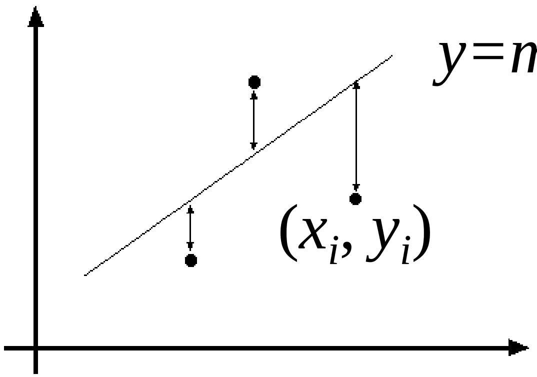
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

Simple example: Fitting a line

Least squares 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

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



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

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

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

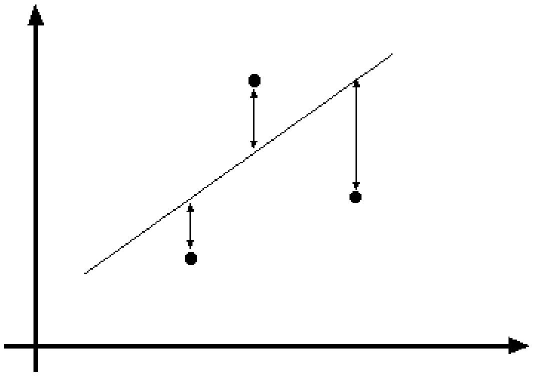
$$\mathbf{A}^T \mathbf{Ap} = \mathbf{A}^T \mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{y}$$

Matlab: `p = A \ y;`

Modified from S. Lazebnik

Problem with “vertical” least squares

- Not rotation-invariant
- Fails completely for vertical lines



Slide from S. Lazebnik

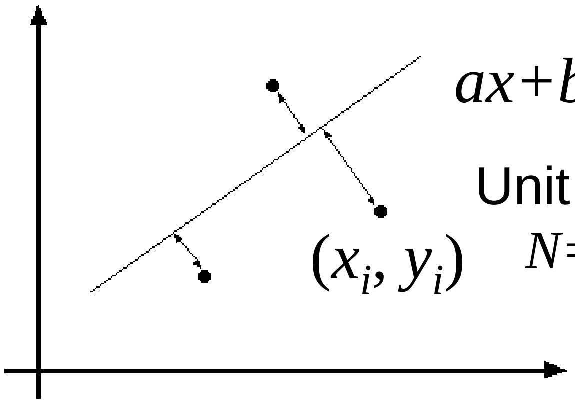
Total least squares

If ($a^2+b^2=1$) then

Distance between point (x_i, y_i) is

$$|ax_i + by_i + c|$$

proof:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

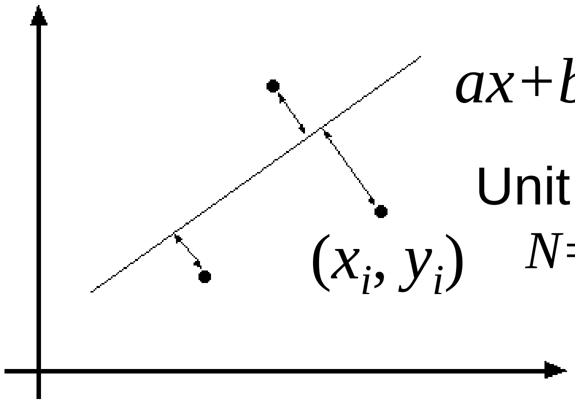


Total least squares

If ($a^2+b^2=1$) then

Distance between point (x_i, y_i) is

$$|ax_i + by_i + c|$$

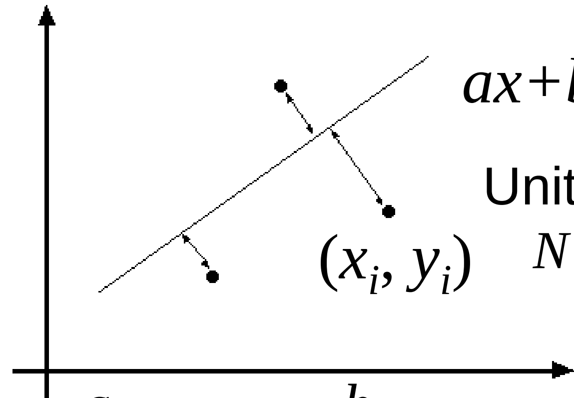


Find (a, b, c) to minimize the sum of squared perpendicular distances

$$\boxed{E = \sum_{i=1}^{n}(ax_i + by_i + c)^2}$$

Total least squares

Find (a, b, c) to minimize the sum of squared perpendicular distances



$$\boxed{E = \sum_{i=1}^{n}(a x_i + b y_i + c)^2}$$

$$\frac{\partial E}{\partial c} = \sum_{i=1}^{n} -2(a x_i + b y_i + c) = 0 \qquad c = -\frac{a}{n}\sum_{i=1}^{n} x_i - \frac{b}{n}\sum_{i=1}^{n} x_i = -a\bar{x} - b\bar{y}$$

$$E = \sum_{i=1}^{n}(a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}$$

$$\text{minimize}\, \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p} \quad \text{s.t.}\ \mathbf{p}^T \mathbf{p} = 1 \quad \Rightarrow \quad \text{minimize} \frac{\mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}}{\mathbf{p}^T \mathbf{p}}$$

Solution is eigenvector corresponding to smallest eigenvalue of A^TA

See details on Raleigh Quotient: http://en.wikipedia.org/wiki/Rayleigh_quotient

Recap: Two Common Optimization Problems

Problem statement	Solution
minimize $\left\Vert \mathbf{A}\mathbf{x} - \mathbf{b} \right\Vert^2$ least squares solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$	$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$ $\mathbf{x} = \mathbf{A} \setminus \mathbf{b}$ (matlab)

Problem statement	Solution
minimize $\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}$ s.t. $\mathbf{x}^T\mathbf{x} = 1$ minimize $\dfrac{\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$ non - trivial lsq solution to $\mathbf{M}\mathbf{x} = 0$	$[\mathbf{v}, \lambda] = \mathrm{eig}(\mathbf{A}^T\mathbf{A})$ $\lambda_1 < \lambda_{2..n} : \mathbf{x} = \mathbf{v}_1$

Search / Least squares conclusions

Good

- Clearly specified objective
- Optimization is easy (for least squares)

Bad

- Not appropriate for non-convex objectives
  - May get stuck in local minima
- Sensitive to outliers
  - Bad matches, extra points
- Doesn’t allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.

Robust least squares (to deal with outliers)

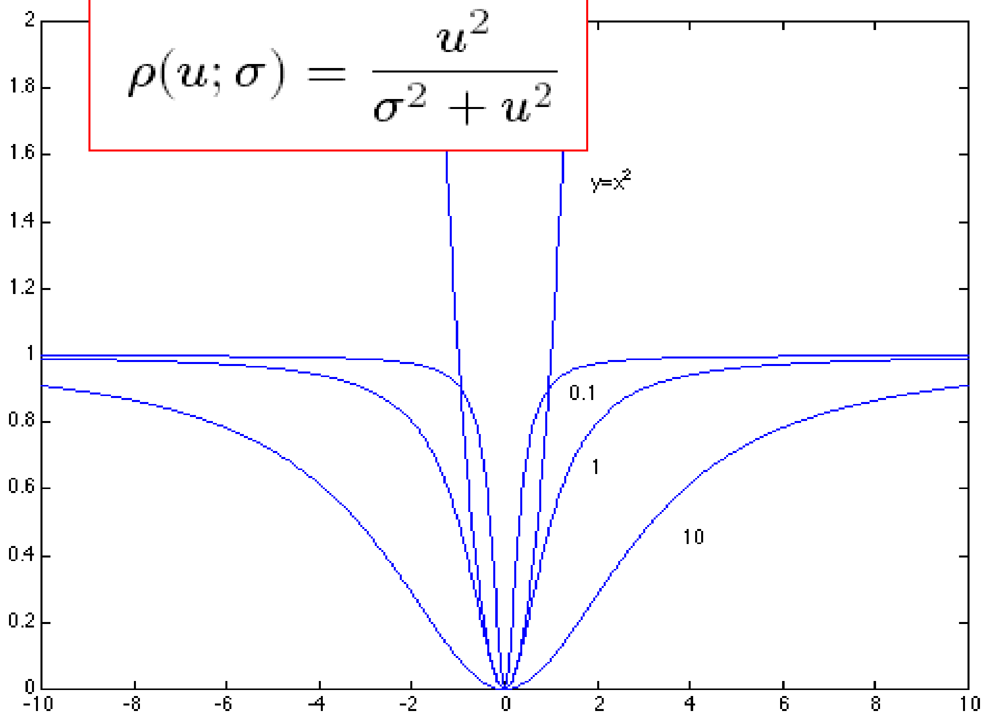
General approach:

minimize

$$\sum_{i} \rho\left(u_i\left(x_i, \theta\right); \sigma\right) \qquad u = \sum_{i=1}^{n} (y_i - m x_i - b)^2$$

$u_i\,(x_i,\, \theta)$ – residual of i^th point w.r.t. model parameters ϑ
ρ – robust function with scale parameter σ

$$\rho(u; \sigma) = \frac{u^2}{\sigma^2 + u^2}$$



The robust function ρ

- Favors a configuration with small residuals
- Constant penalty for large residuals

Slide from S. Savarese

Robust Estimator (M-estimator)

1. Initialize σ=0

2. Choose params to minimize: $$\sum_i \frac{error(\theta, data_i)^2}{\sigma^2 + error(\theta, data_i)^2}$$
   – E.g., numerical optimization

3. Compute new σ: $\sigma = 1.5 \cdot \mathrm{median}(error)$

4. Repeat (2) and (3) until convergence

Demo – part 1

Hypothesize and test

1. Propose parameters
   - Try all possible
   - Each point votes for all consistent parameters
   - Repeatedly sample enough points to solve for parameters

2. Score the given parameters
   - Number of consistent points, possibly weighted by distance

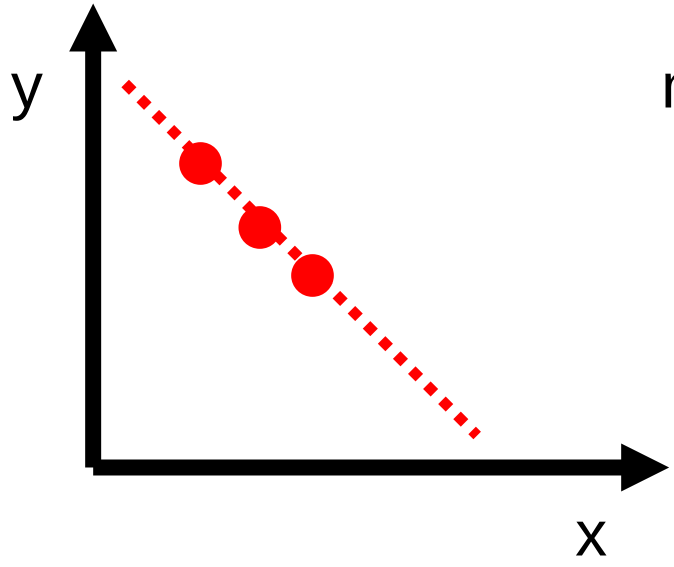
3. Choose from among the set of parameters
   - Global or local maximum of scores

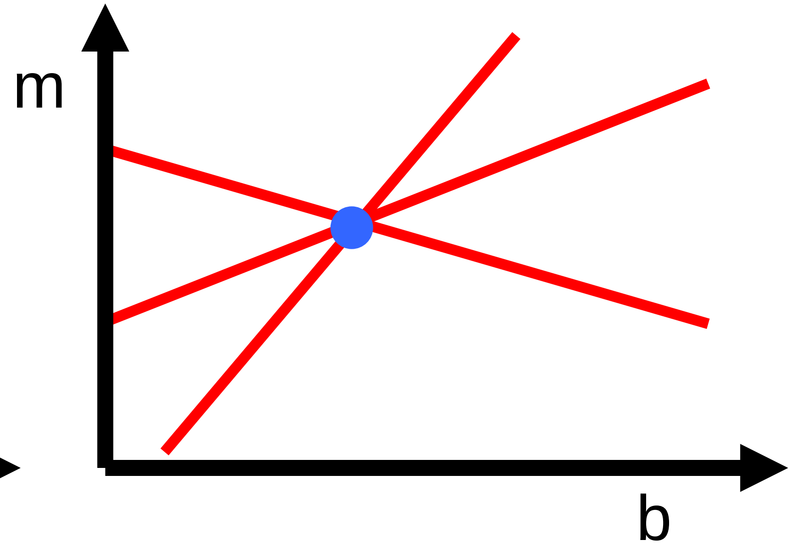
4. Possibly refine parameters using inliers

Hough transform

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

Given a set of points, find the curve or line that explains the data points best



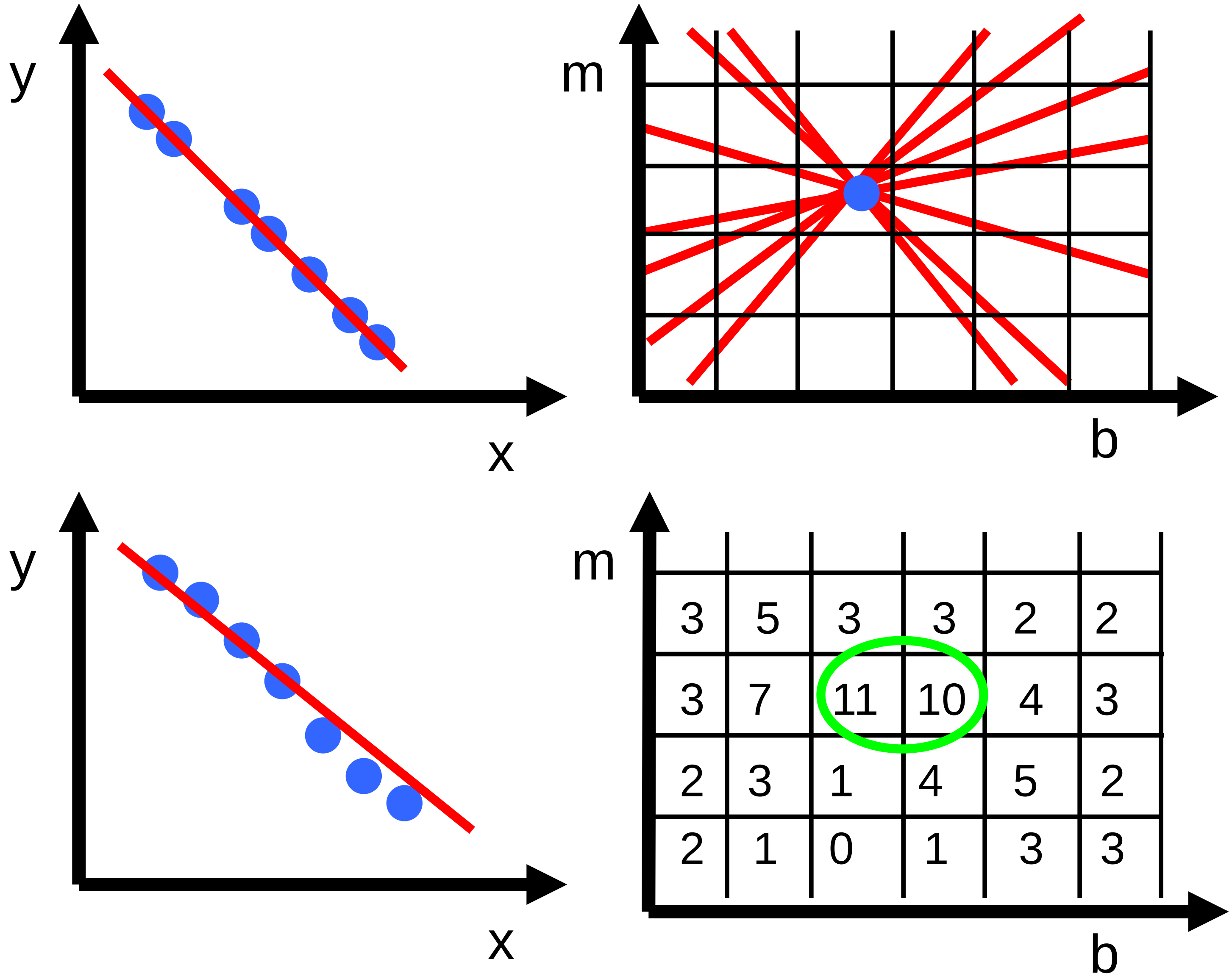


Hough space

$$y = m\,x + b$$

Slide from S. Savarese

Hough transform



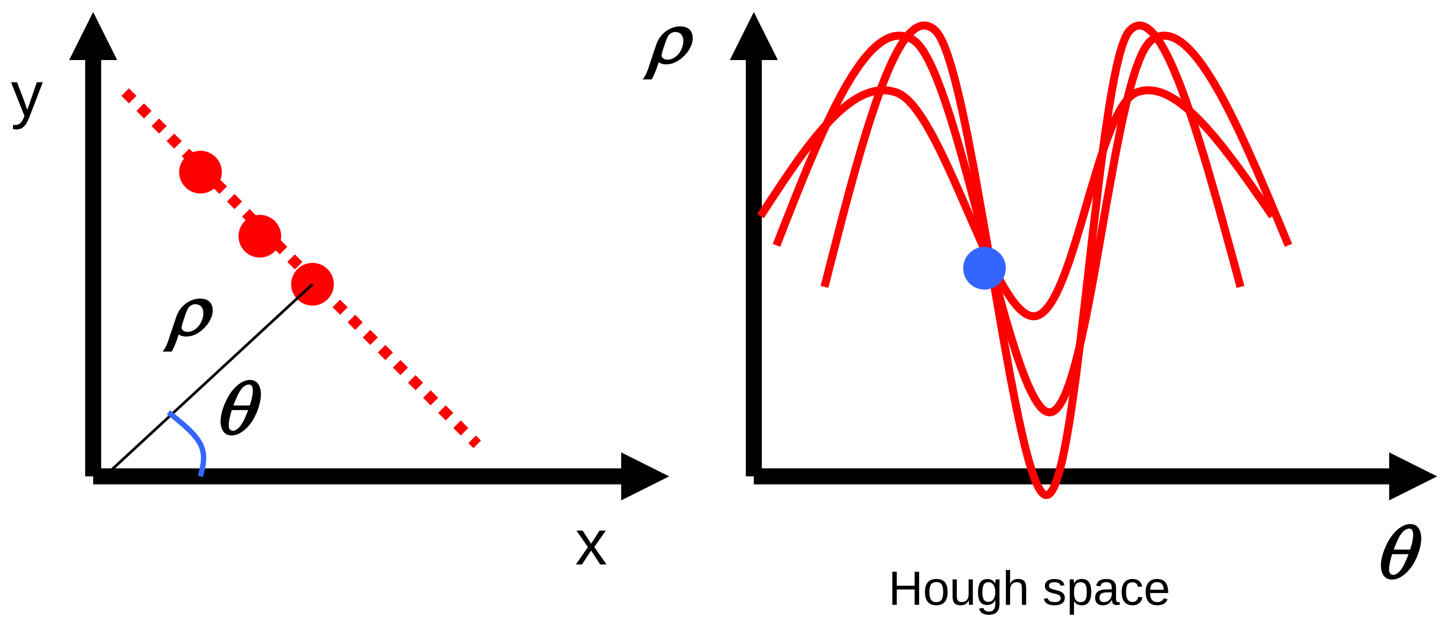
Slide from S. Savarese

Hough transform

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

Issue : parameter space [m,b] is unbounded…

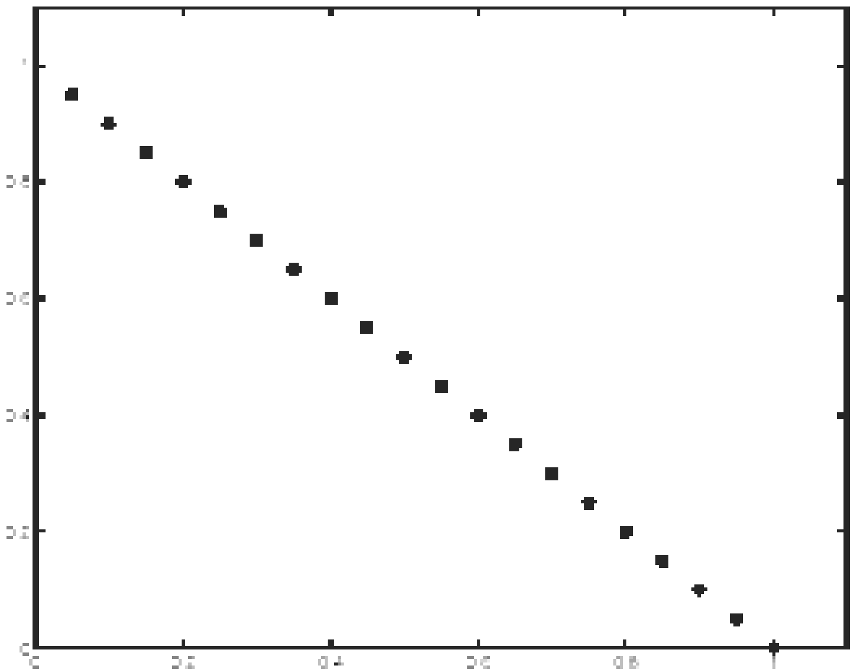
Use a polar representation for the parameter space



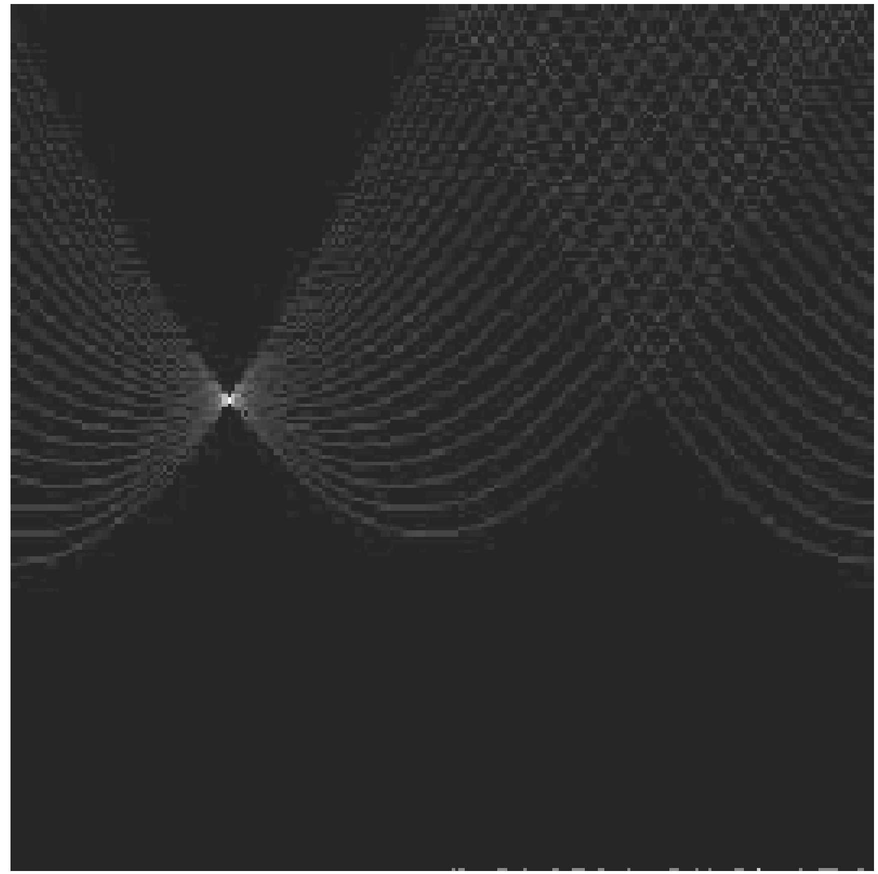
$$x\cos\theta + y\sin\theta = \rho$$

Slide from S. Savarese

Hough transform - experiments



features

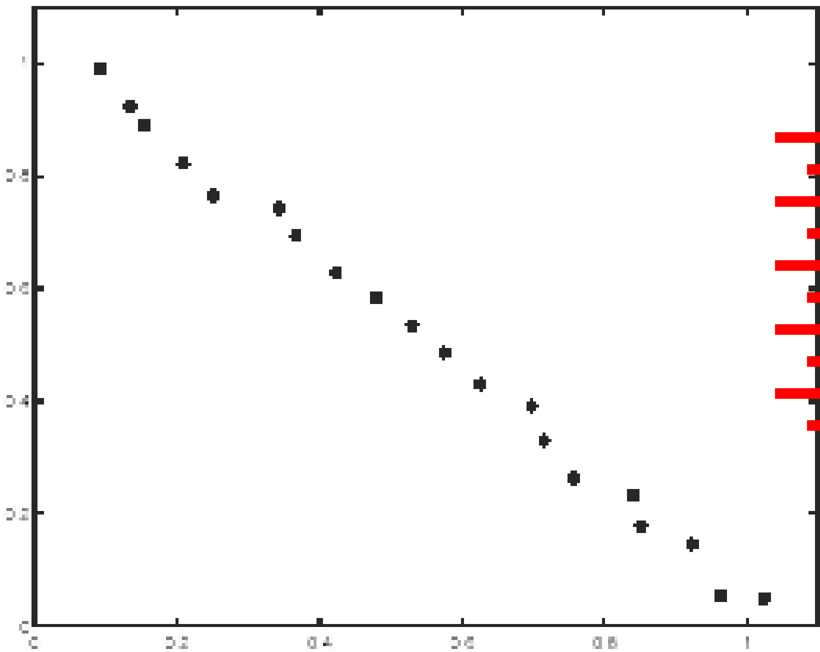


votes

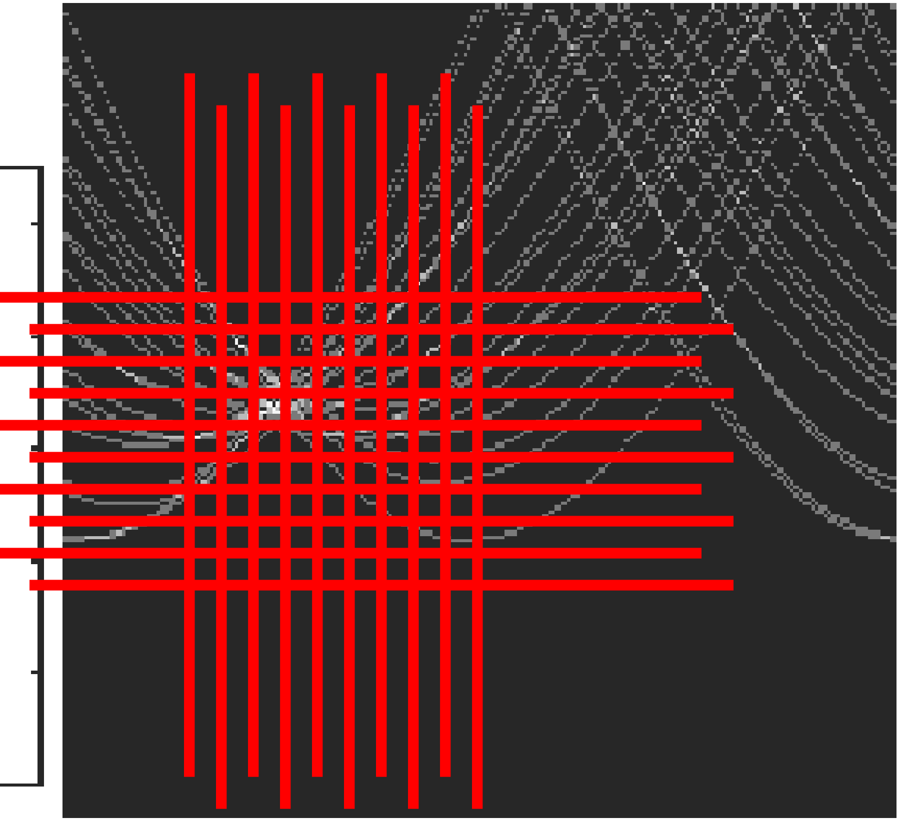
Slide from S. Savarese

Hough transform - experiments

Noisy data



features

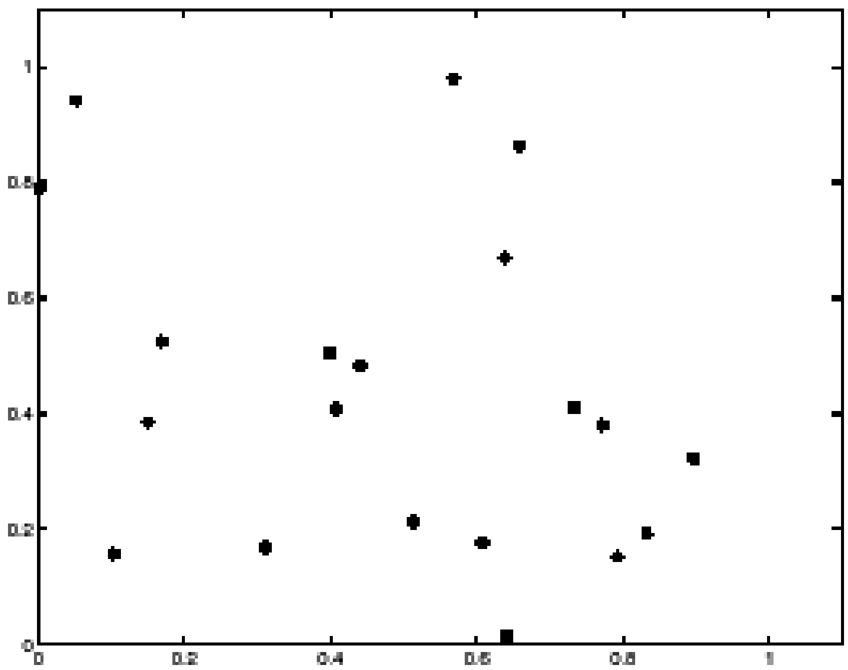


votes

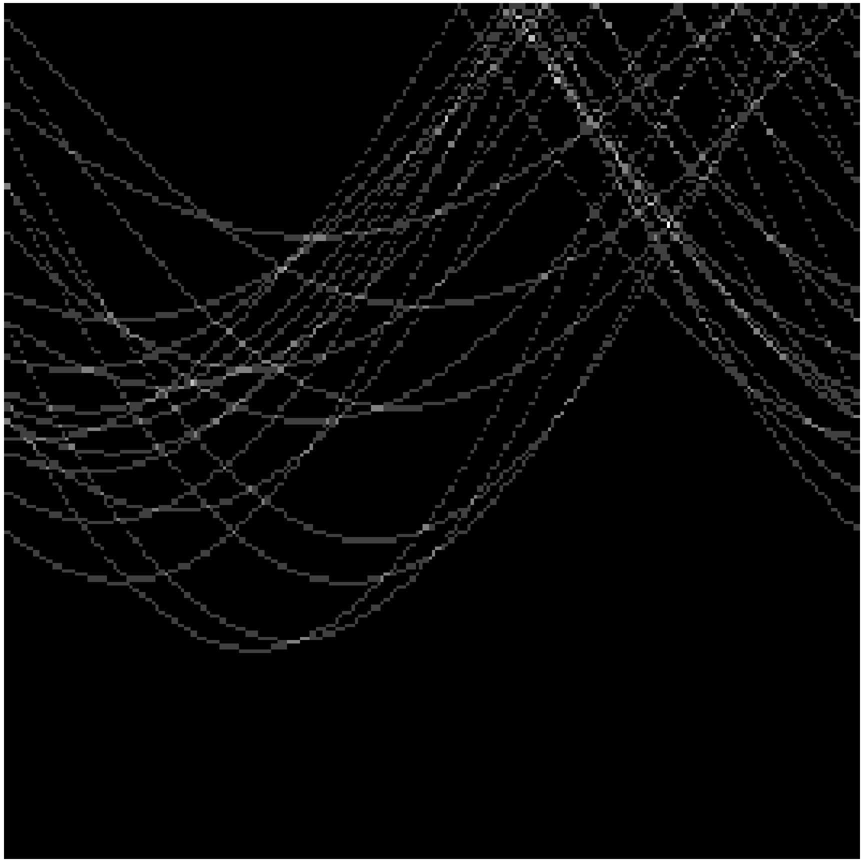
Issue: Grid size needs to be adjusted…

Slide from S. Savarese

Hough transform - experiments



features



votes

Issue: spurious peaks due to uniform noise

Slide from S. Savarese

Hough transform

- Fitting a circle (x, y, r)

Hough transform conclusions

Good

- Robust to outliers: each point votes separately
- Fairly efficient (much faster than trying all sets of parameters)
- Provides multiple good fits

Bad

- Some sensitivity to noise
- Bin size trades off between noise tolerance, precision, and speed/memory
  - Can be hard to find sweet spot
- Not suitable for more than a few parameters
  - grid size grows exponentially

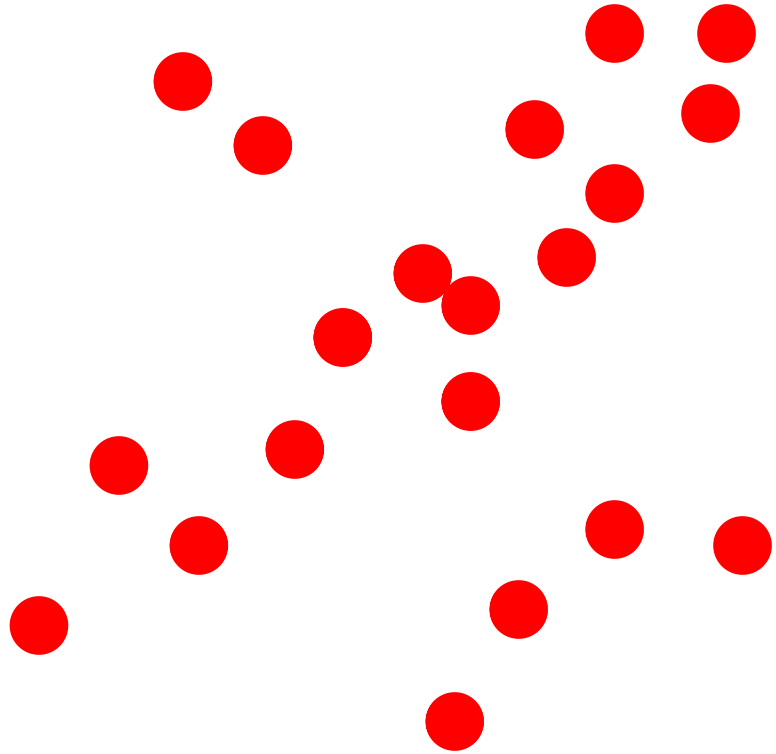
Common applications

- Line fitting (also circles, ellipses, etc.)
- Object instance recognition (parameters are affine transform)
- Object category recognition (parameters are position/scale)

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



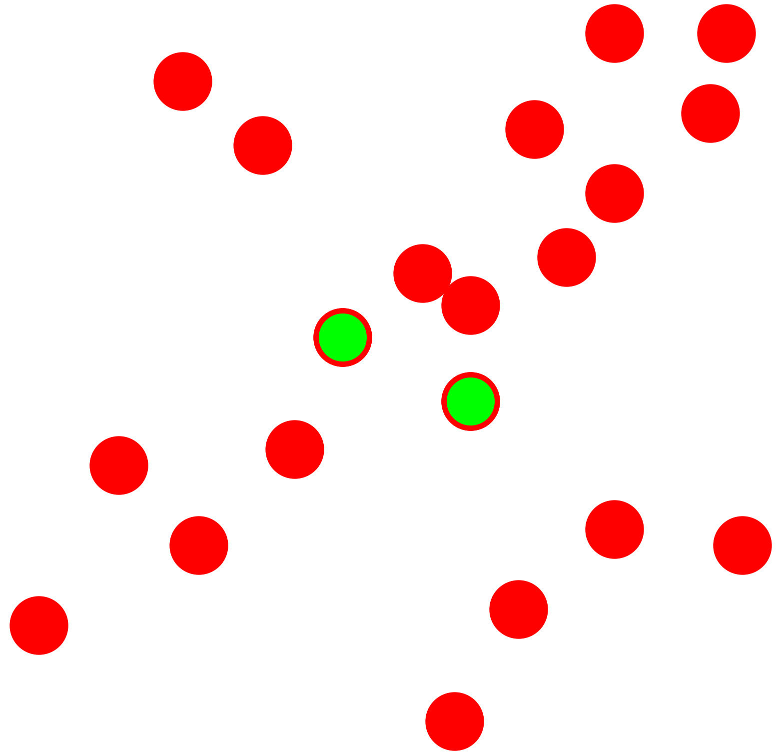
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
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

Line fitting example



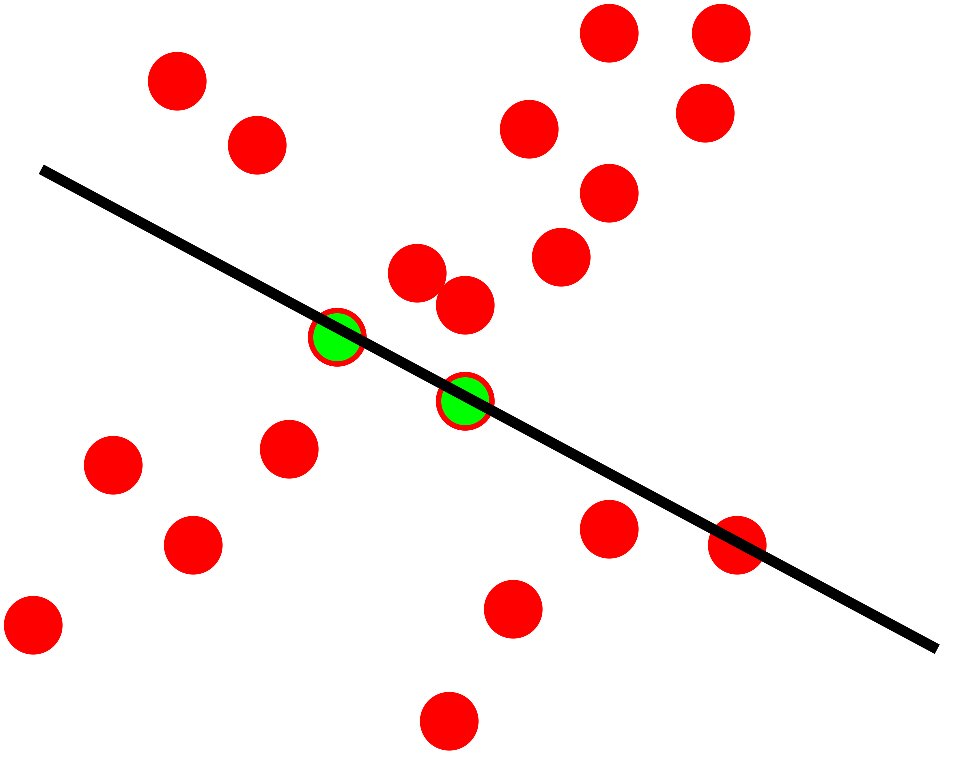
Algorithm:

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

Illustration by Savarese

RANSAC



Line fitting example

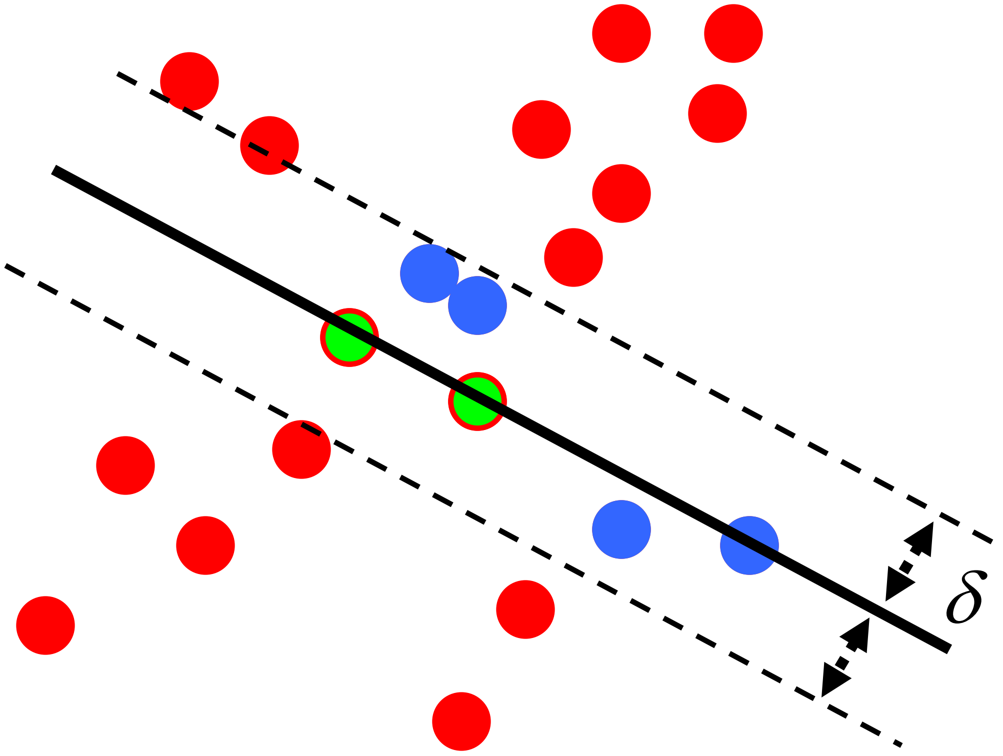
Algorithm:

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

Line fitting example



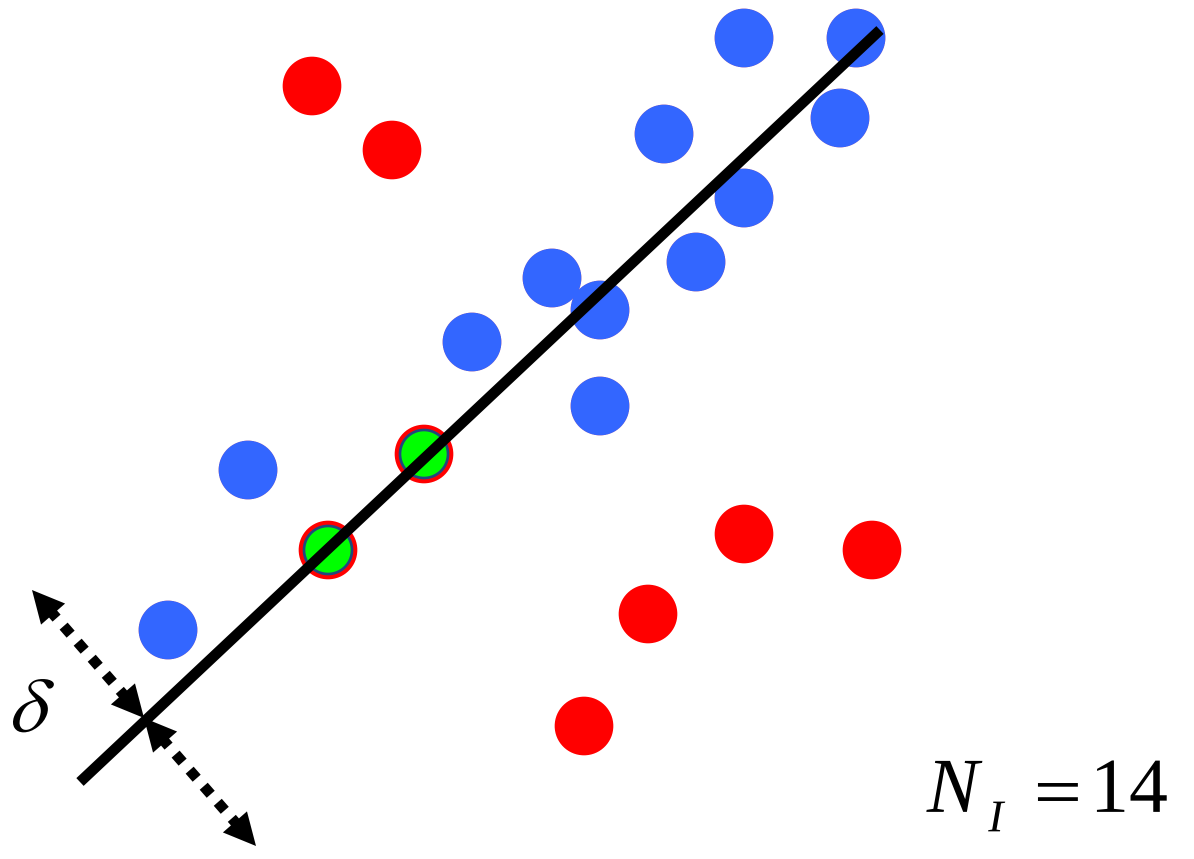
$$N_I = 6$$

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



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

How to choose parameters?

- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)
- Number of sampled points *s*
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

	proportion of outliers *e*						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

modified from M. Pollefeys

RANSAC conclusions

Good

- Robust to outliers
- Applicable for larger number of parameters than Hough transform
- Parameters are easier to choose than Hough transform

Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not good for getting multiple fits

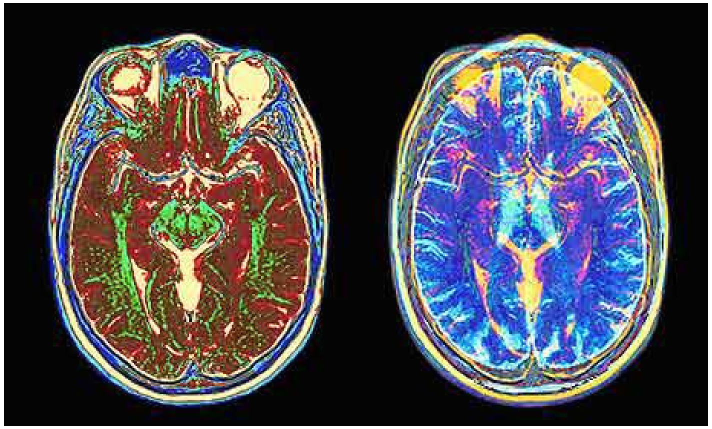
Common applications

- Computing a homography (e.g., image stitching)
- Estimating fundamental matrix (relating two views)

Demo – part 2

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

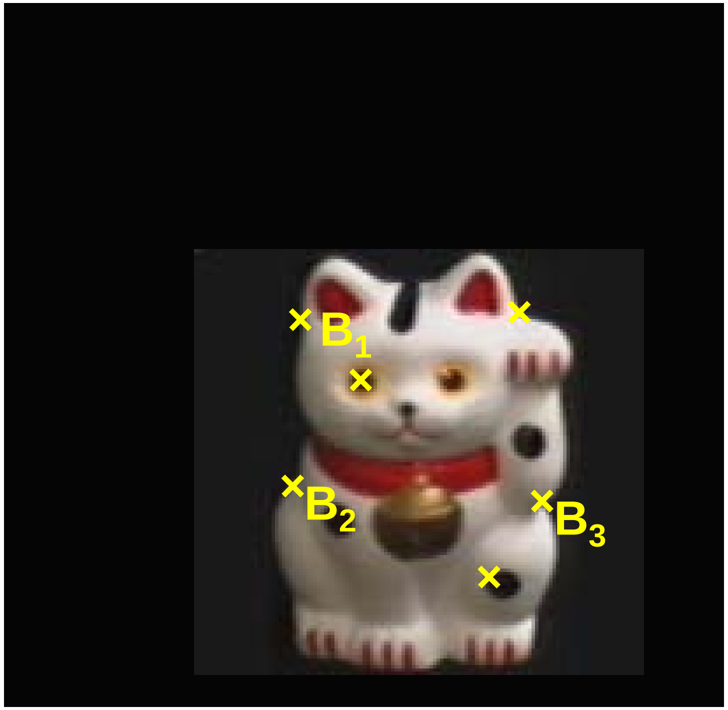
Robotics: match point clouds

Iterative Closest Points (ICP) Algorithm

Goal: estimate transform between two dense sets of points

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

Example: solving for translation

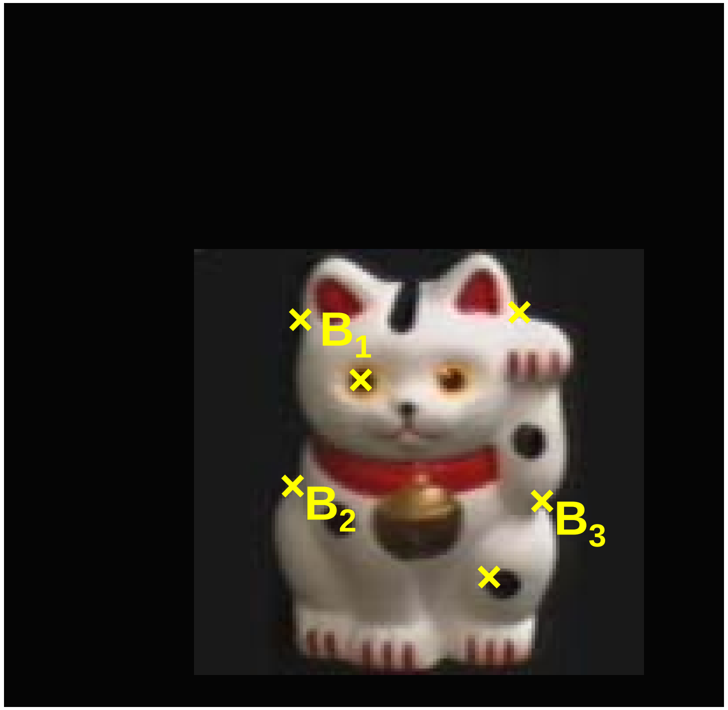


Given matched points in {A} and {B}, estimate the translation of the object

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

Example: solving for translation

(t_x, t_y)



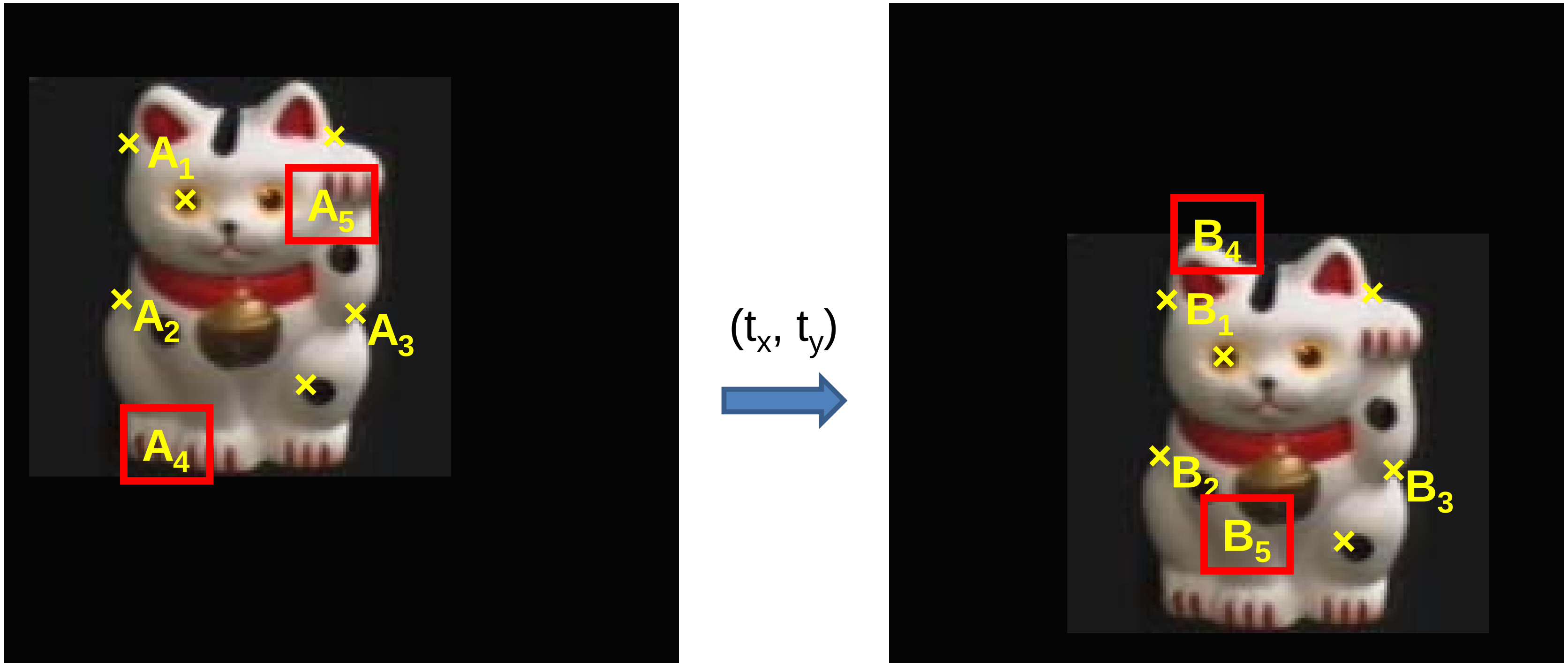
Least squares solution

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

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

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

Example: solving for translation



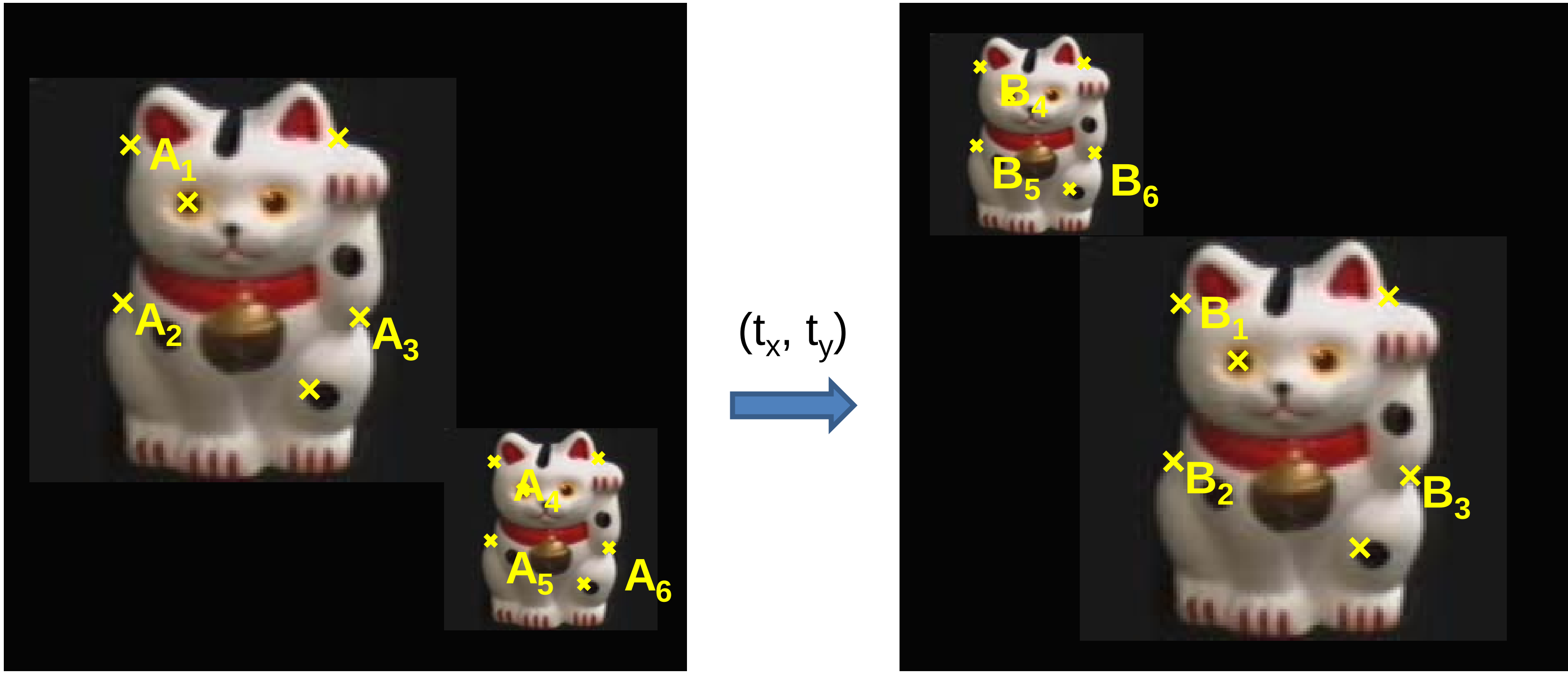
Problem: outliers

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



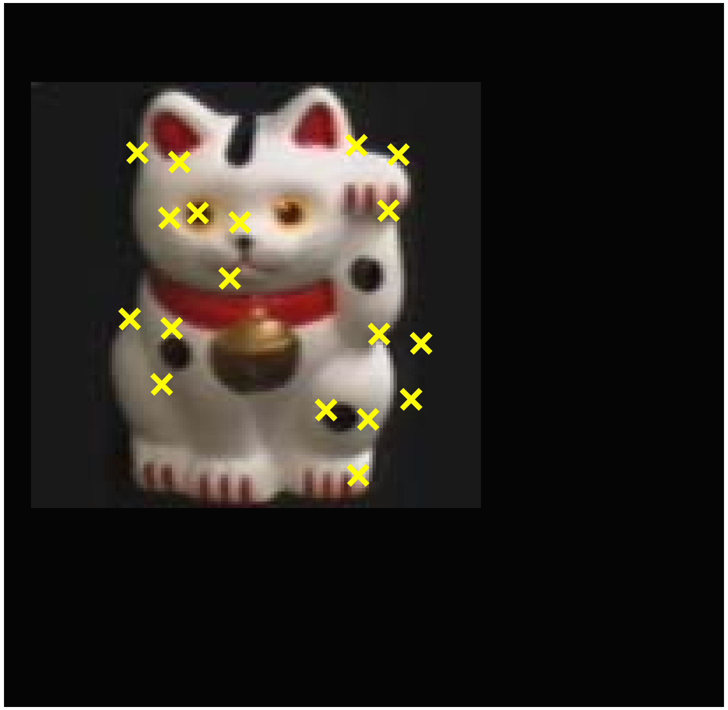
Problem: outliers, multiple objects, and/or many-to-one matches

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

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

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

Next class: Clustering

- Clustering algorithms
  - K-means
  - K-medoids
  - Hierarchical clustering
- Model selection