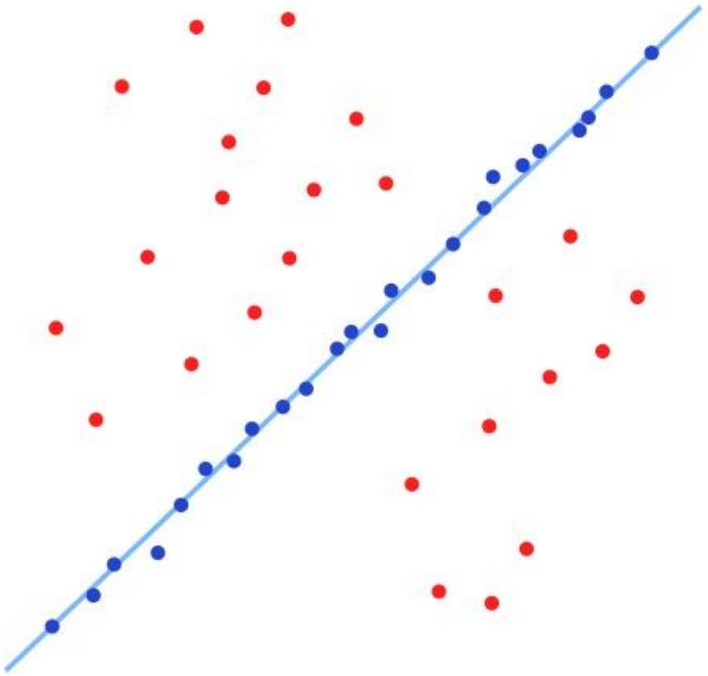
Fitting and Alignment



Computer Vision

Jia-Bin Huang, Virginia Tech

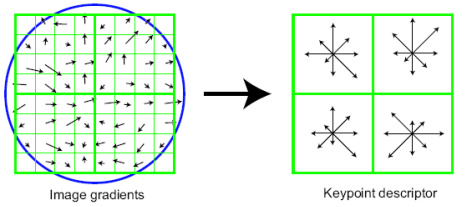
Many slides from S. Lazebnik and D. Hoiem

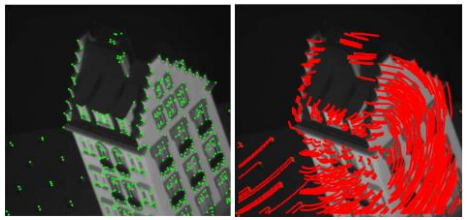
Administrative Stuffs

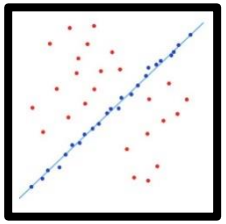
- Homework grading policy
  - Graduate students: graded out of 600 points
  - Undergrad students: graded out of 525 points
  - For example, grad students need to complete on average 25 points extra credits.
- HW 1
  - Extra credits due 11:59 PM Friday 9/23
  - Competition: Edge Detection
    - Submission link
    - Leaderboard
- Anonymous feedback
  - Lectures are too fast, too many slides

Where are we?

- 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 (this class)
  - find the transformation parameters that best align matched points
- Object instance recognition (next class)
  - Keypoint-based object instance recognition and search



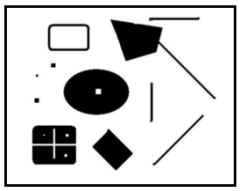




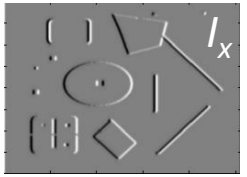
Review: Harris Corner Detector

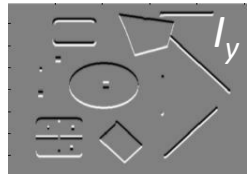
- Second moment matrix

$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

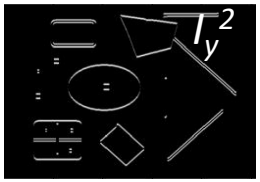


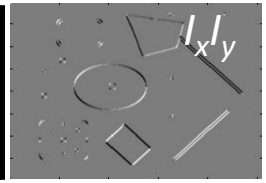
1. Image derivatives (optionally, blur first)



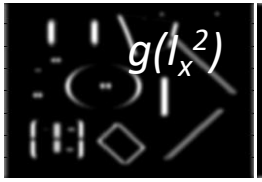


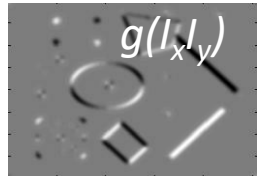
2. Square of derivatives





3. Gaussian filter $g(\sigma_I)$

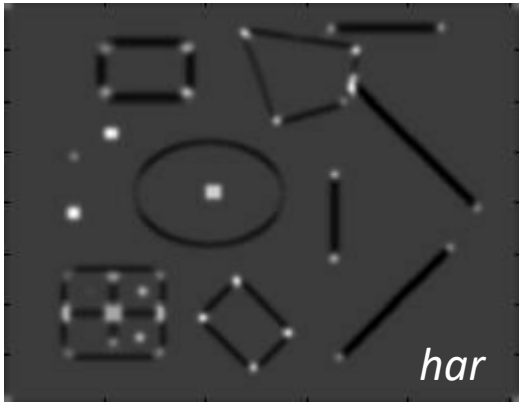




$$\det M = \lambda_1 \lambda_2$$
$$\text{trace}\, M = \lambda_1 + \lambda_2$$

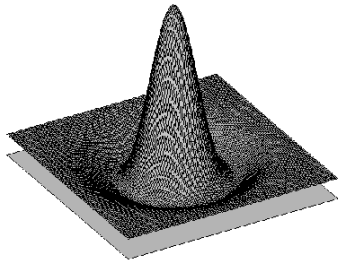
4. Cornerness function – both eigenvalues are strong

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I, \sigma_D))^2] =$$
$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$



5. Non-maxima suppression

Review: Find local maxima in position-scale space of Difference-of-Gaussian



$$L_{xx}(\sigma) + L_{yy}(\sigma)$$

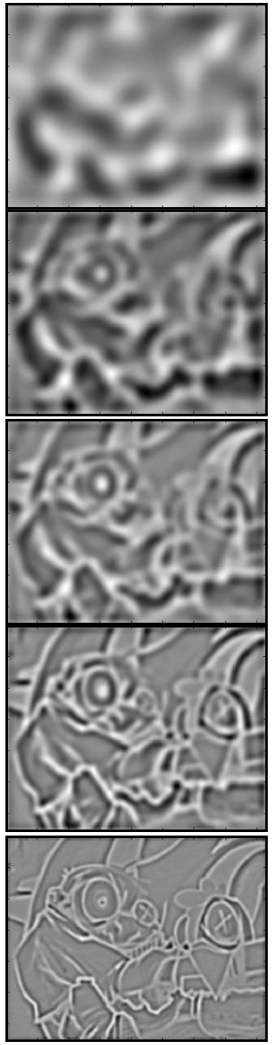
σ^5

σ^4

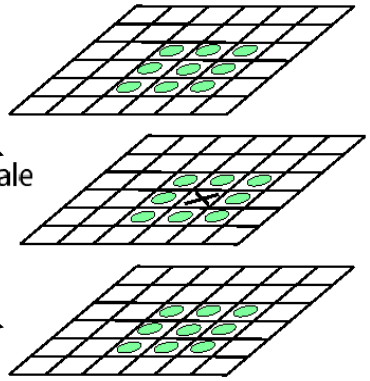
σ^3

σ^2

σ

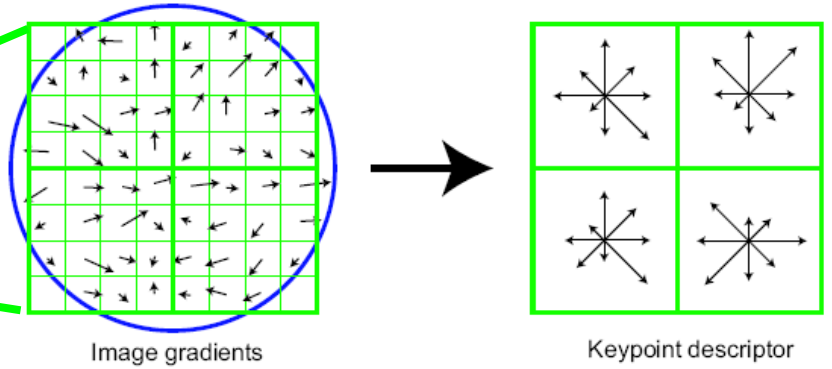


Scale



⇒ List of
(x, y, s)

Review: SIFT Descriptor

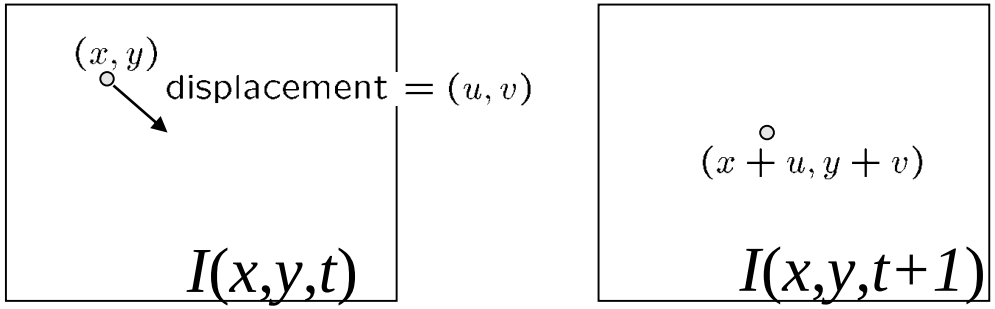


Histogram of oriented gradients

- **Captures important texture information**
- **Robust to small translations / affine deformations**

[Lowe, ICCV 1999]

Review: Lucas-Kanade Tracker



$$I(x,y,t) = I(x+u, y+v, t+1)$$

Brightness consistency

$$I(x+u, y+v, t+1) \approx I(x,y,t) + I_x \cdot u + I_y \cdot v + I_t$$

Small motion

$$I_x \cdot u + I_y \cdot v + I_t \approx 0$$

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

Spatial coherence

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $\qquad$ $A^T b$

Dealing with larger movements: Iterative refinement

Original (x,y) position

1. Initialize (x',y') = (x,y)
2. Compute (u,v) by

$I_t = I(x', y', t+1) - I(x, y, t)$

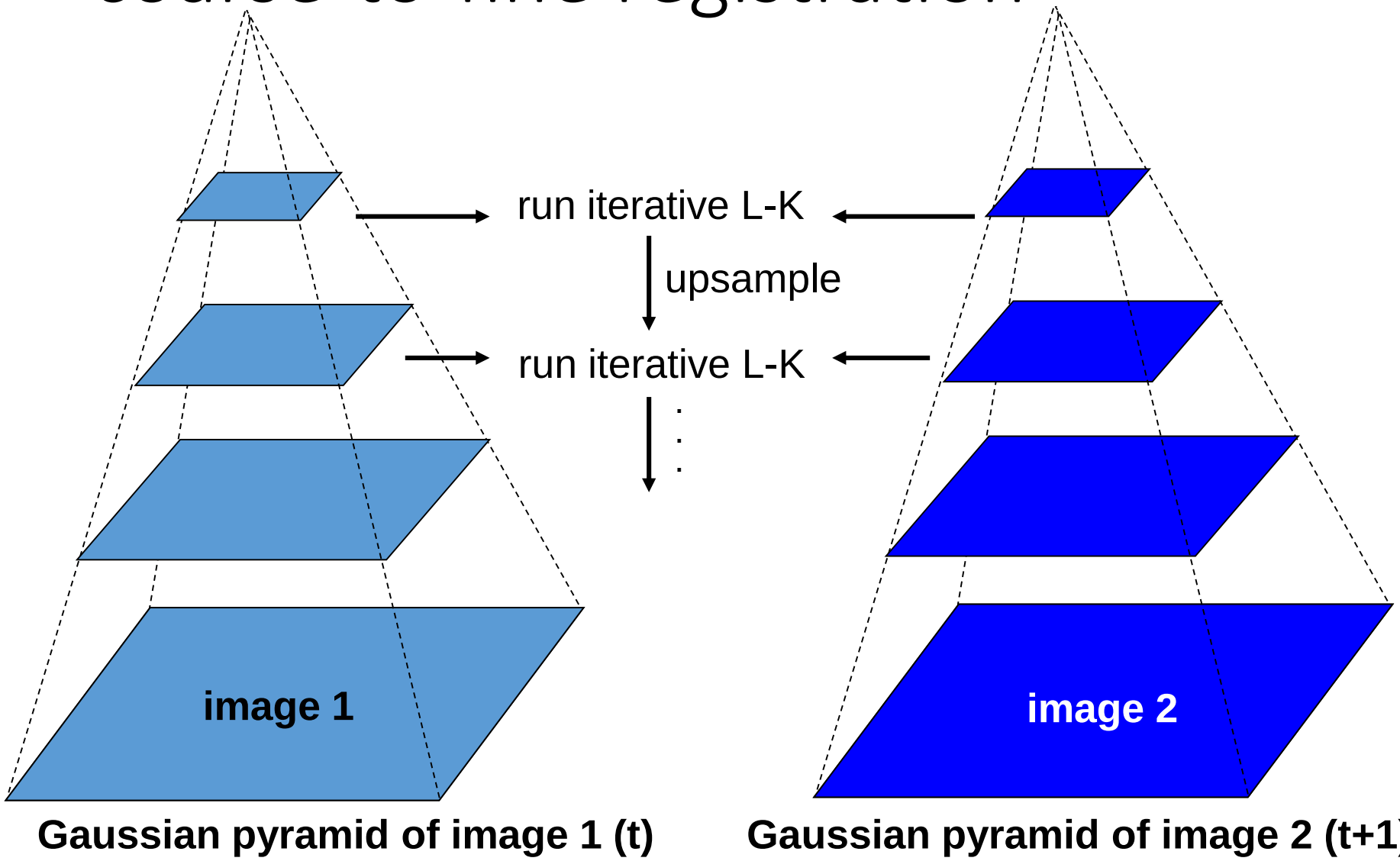
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

2nd moment matrix for feature patch in first image

displacement

3. Shift window by (u, v): `x' =x' +u;  y' =y' +v;`
4. Recalculate I_t
5. Repeat steps 2-4 until small change
   - Use interpolation for subpixel values

Dealing with larger movements: coarse-to-fine registration



Fitting [ˈfidiNG]:

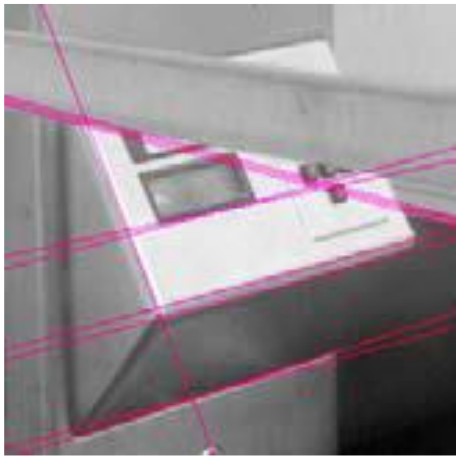
find the parameters of a model that best fit the data

Alignment [əˈlīnmənt]:

find the parameters of the transformation that best align matched points

Fitting and alignment

- Choose a *parametric model* to represent a set of features



simple model: lines

simple model: circles

complicated model: face shape

complicated model: car

Source: K. Grauman

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)

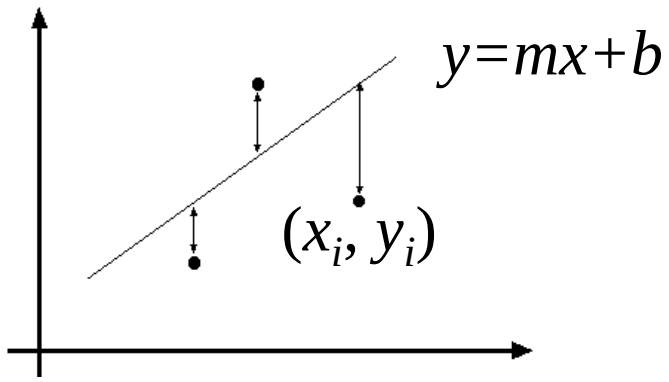
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

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

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



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

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

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

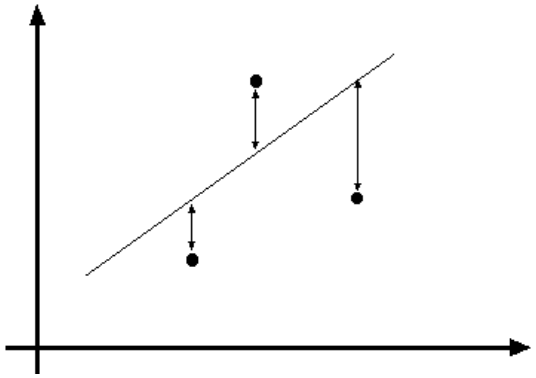
Matlab: `p = A \ y;`

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

Modified from S. Lazebnik

Problem with “vertical” least squares

- Not rotation-invariant
- Fails completely for vertical lines

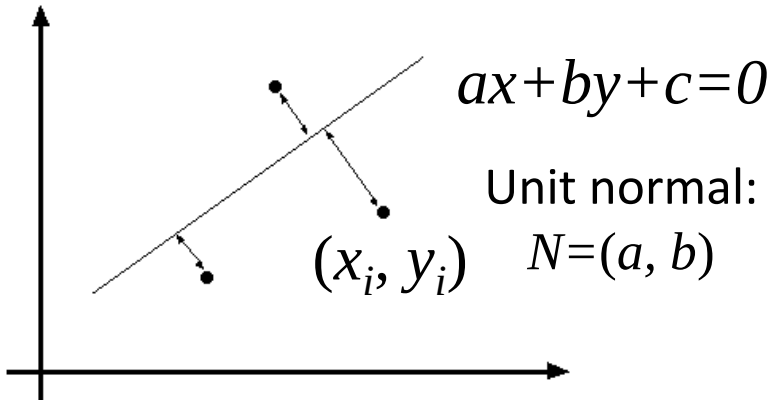


Slide from S. Lazebnik

Total least squares

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

Distance between point (x_i, y_i) and line $ax+by+c=0$ is $|ax_i + by_i + c|$

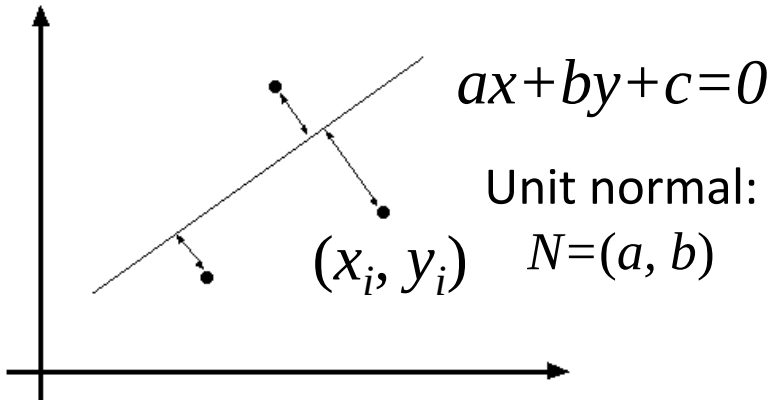


proof:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

Total least squares

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

Distance between point (x_i, y_i) and line $ax+by+c=0$ is $|ax_i + by_i + c|$



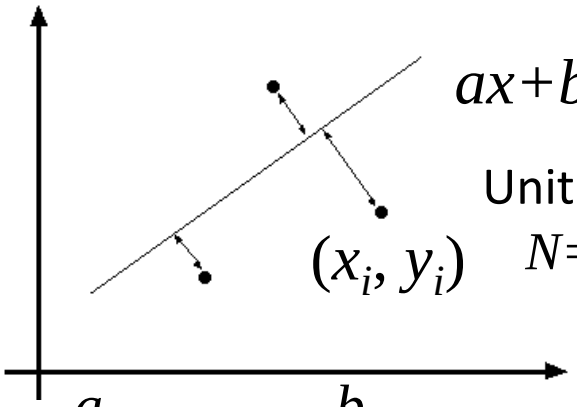
Find (a, b, c) to minimize the sum of squared perpendicular distances

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

Total least squares

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

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



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

$$c = -\frac{a}{n}\sum_{i=1}^{n} x_i - \frac{b}{n}\sum_{i=1}^{n} y_i = -a\bar{x} - b\bar{y}$$

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

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

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
$\text{minimize} \quad \|\mathbf{Ax}-\mathbf{b}\|^2$	$\mathbf{x}=\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$
$\text{least squares solution to } \mathbf{Ax}=\mathbf{b}$	$\mathbf{x}=\mathbf{A}\setminus\mathbf{b}$ (matlab)

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

Least squares (global) optimization

Good

- Clearly specified objective
- Optimization is easy

Bad

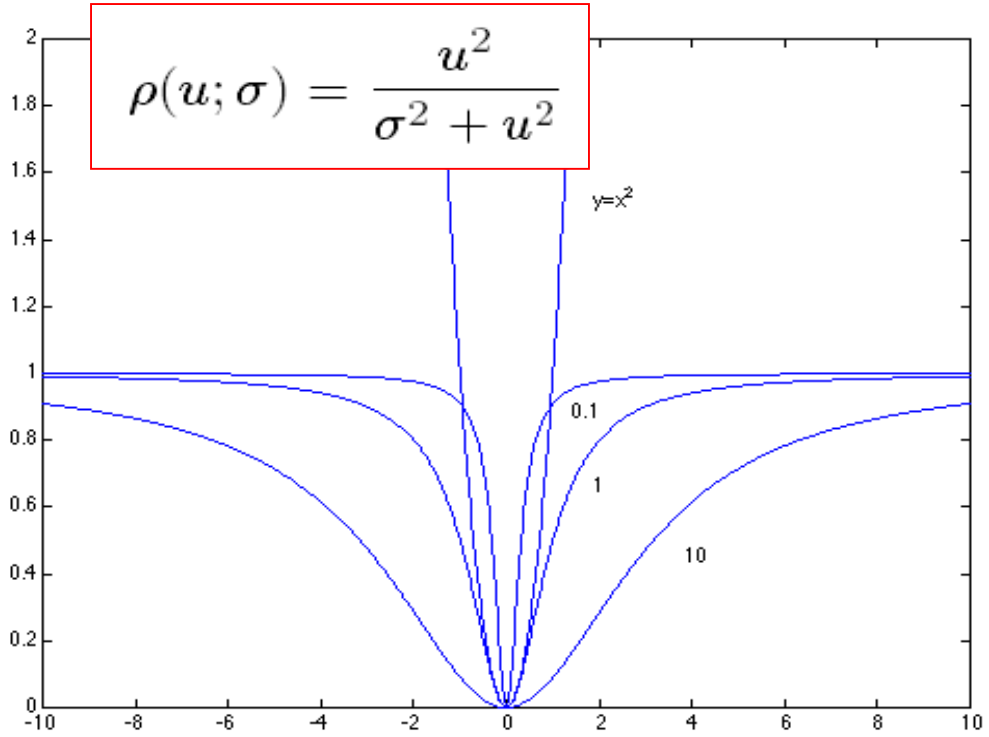
- May not be what you want to optimize
- Sensitive to outliers
  - Bad matches, extra points
- Doesn’t allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.

Robust least squares (to deal with outliers)

General approach:

minimize $\sum_{\mathrm{i}} \rho\left(\mathrm{u}_{\mathrm{i}}\left(\mathrm{x}_{\mathrm{i}}, \boldsymbol{\theta}\right); \boldsymbol{\sigma}\right)$ $\quad u^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2$

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



The robust function ρ

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

Slide from S. Savarese

Robust Estimator

$$\sigma = 1.5 \cdot \text{median}(error)$$

1. Initialize: e.g., choose θ by least squares fit and

$$\sum_i \frac{error(\theta, data_i)^2}{\sigma^2 + error(\theta, data_i)^2}$$

2. Choose params to minimize:
   - E.g., numerical optimization

$$\sigma = 1.5 \cdot \text{median}(error)$$

3. Compute new

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

Other ways to search for parameters (for when no closed form solution exists)

- Line search
  1. For each parameter, step through values and choose value that gives best fit
  2. Repeat (1) until no parameter changes

- Grid search
  1. Propose several sets of parameters, evenly sampled in the joint set
  2. Choose best (or top few) and sample joint parameters around the current best; repeat

- Gradient descent
  1. Provide initial position (e.g., random)
  2. Locally search for better parameters by following gradient

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: Outline

1. Create a grid of parameter values

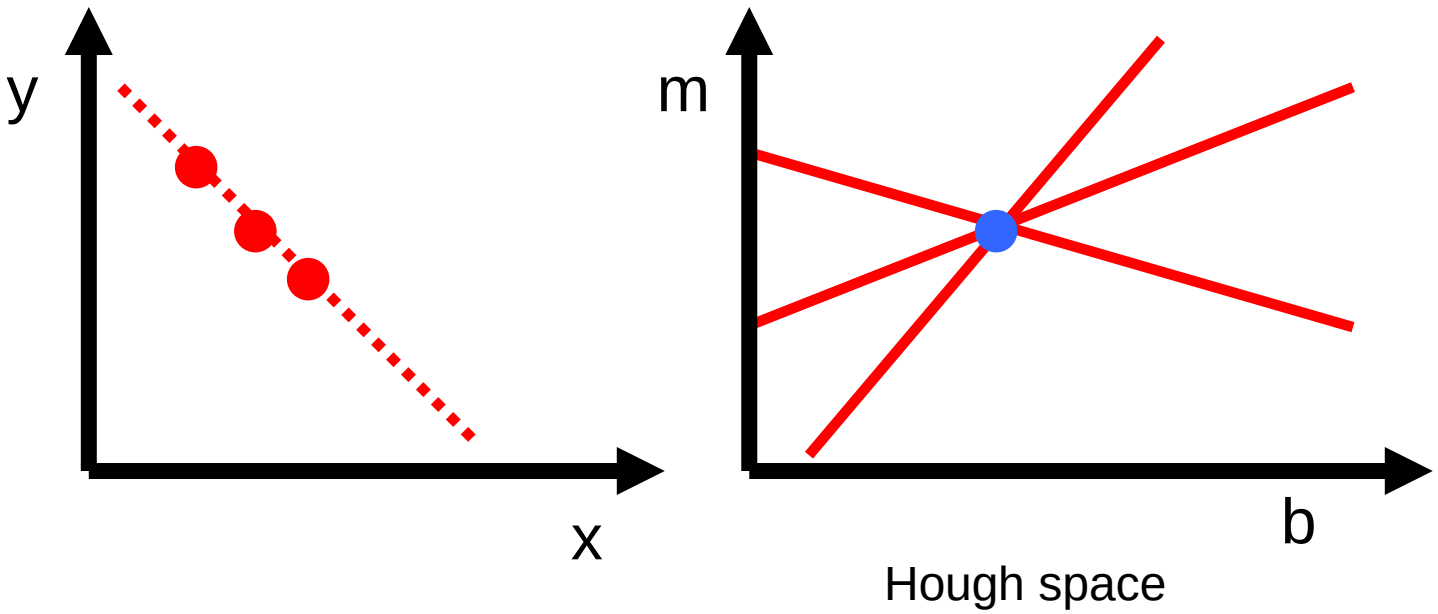
2. Each point votes for a set of parameters, incrementing those values in grid

3. Find maximum or local maxima in grid

Hough transform

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

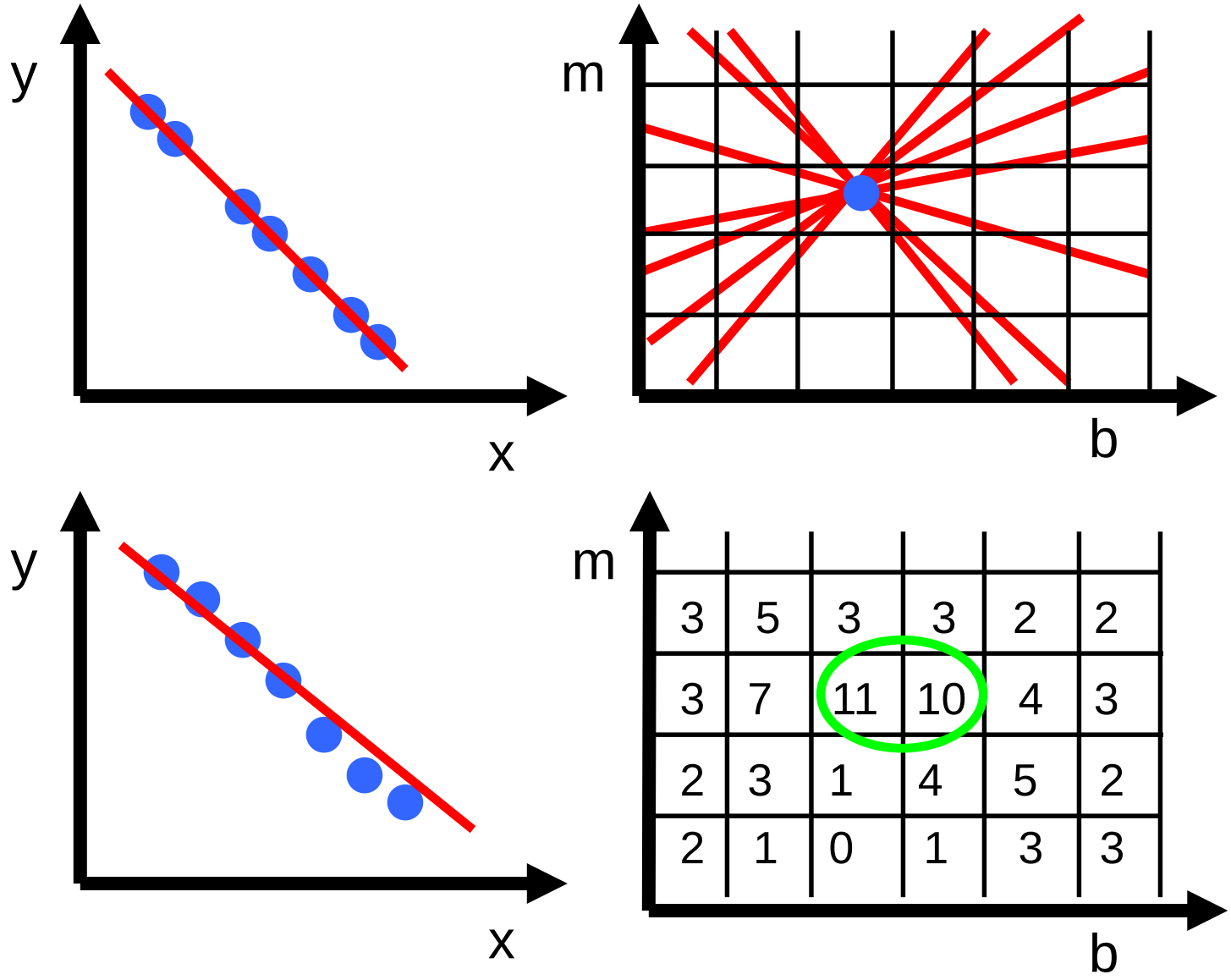
Given a set of points, find the curve or line that explains the data points best



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

Slide from S. Savarese

Hough transform



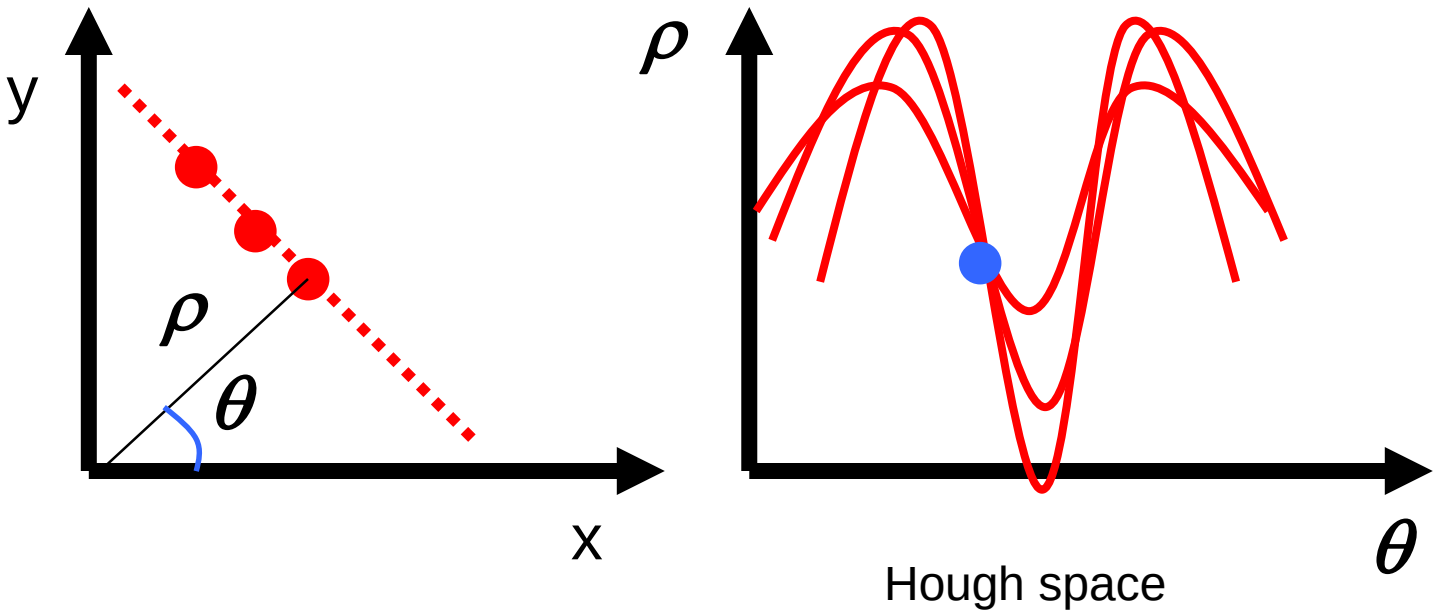
Slide from S. Savarese

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…

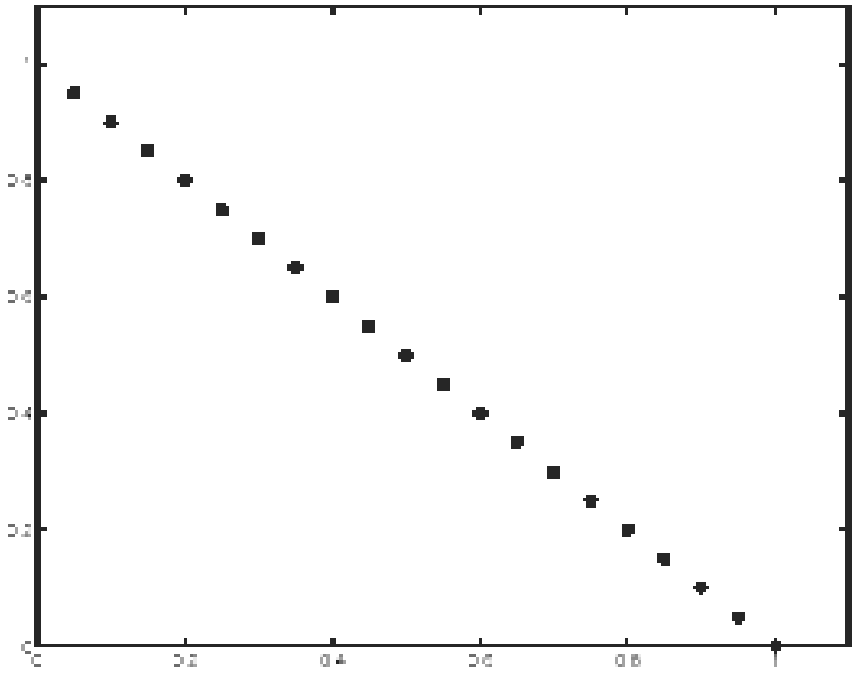
Use a polar representation for the parameter space



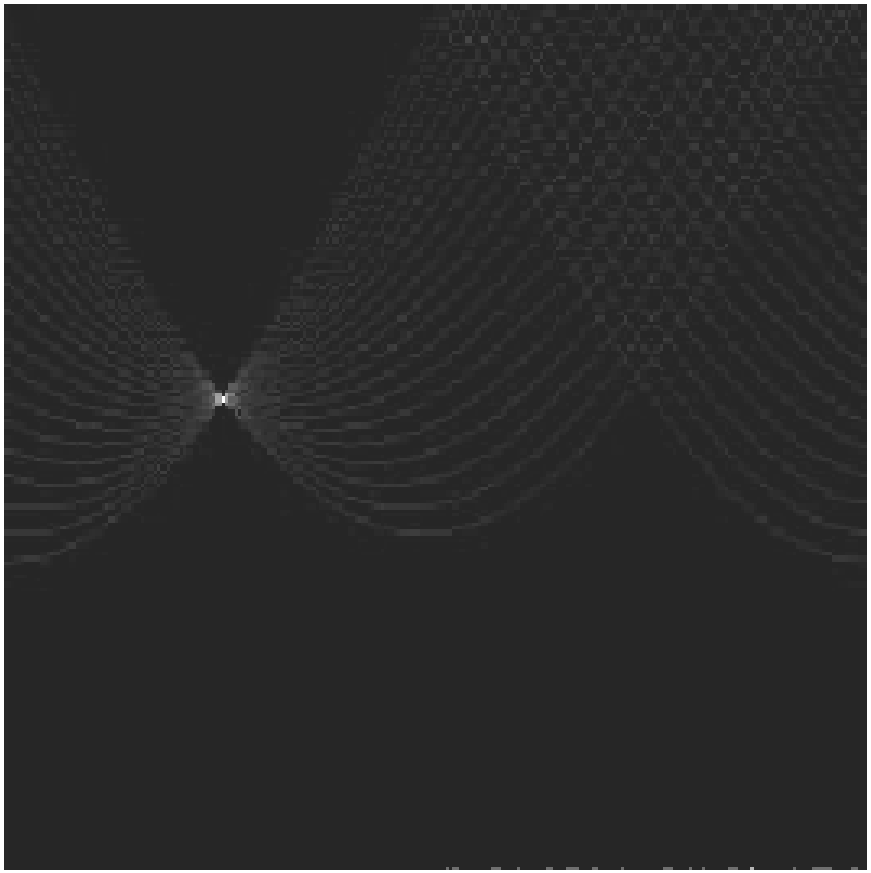
$$x\cos\theta + y\sin\theta = \rho$$

Slide from S. Savarese

Hough transform - experiments



features

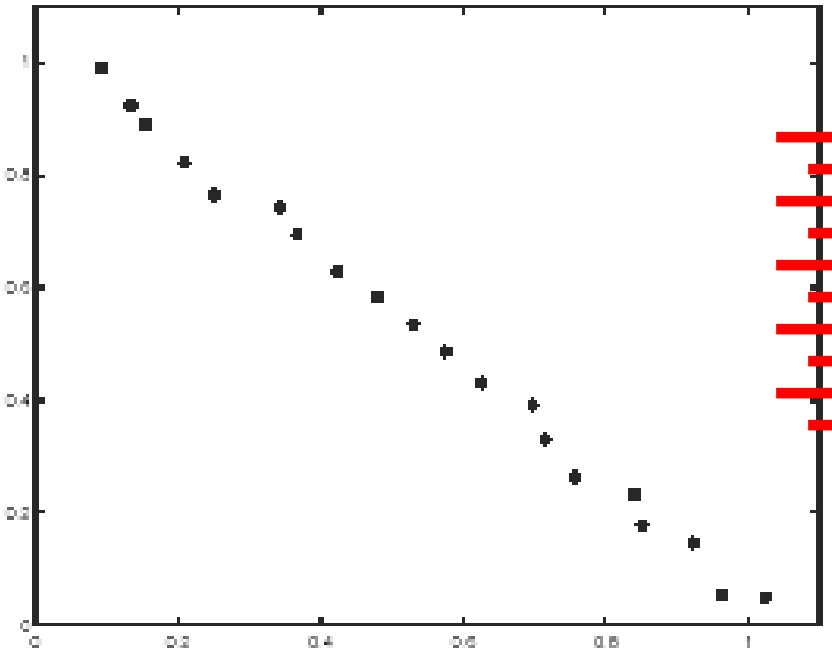


votes

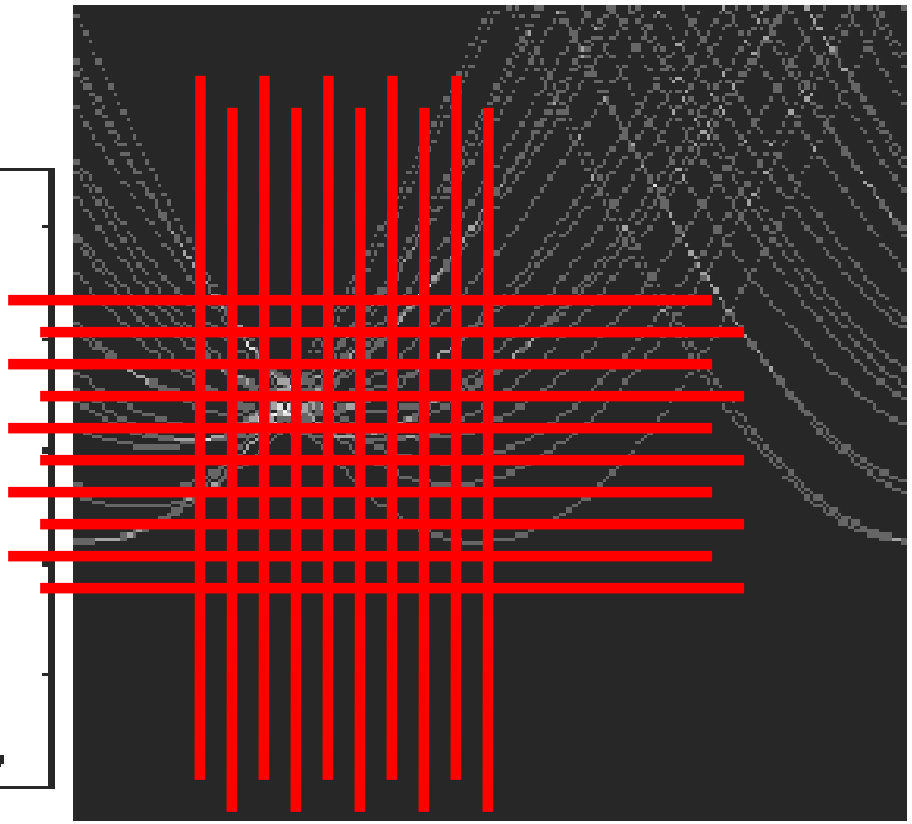
Slide from S. Savarese

Hough transform - experiments

Noisy data



features

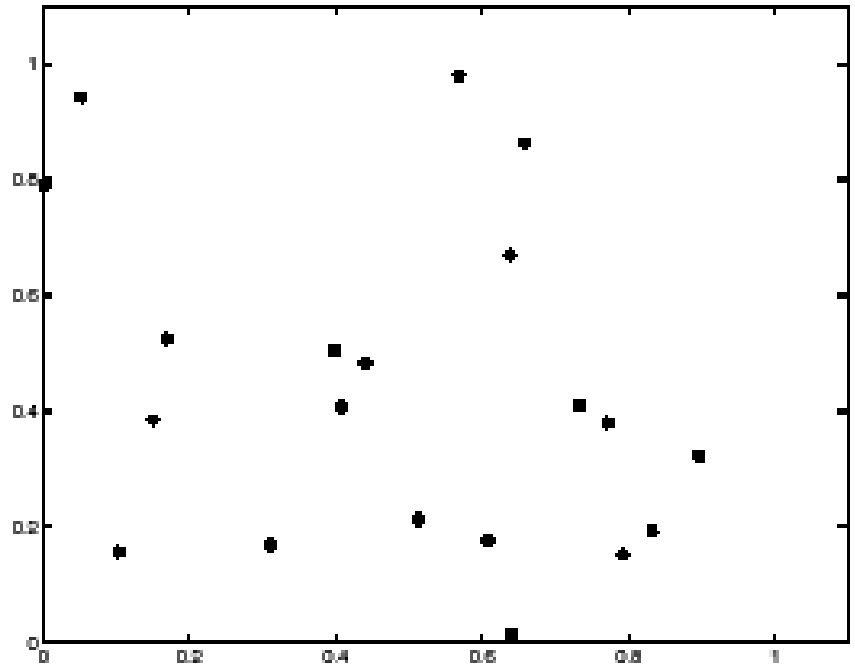


votes

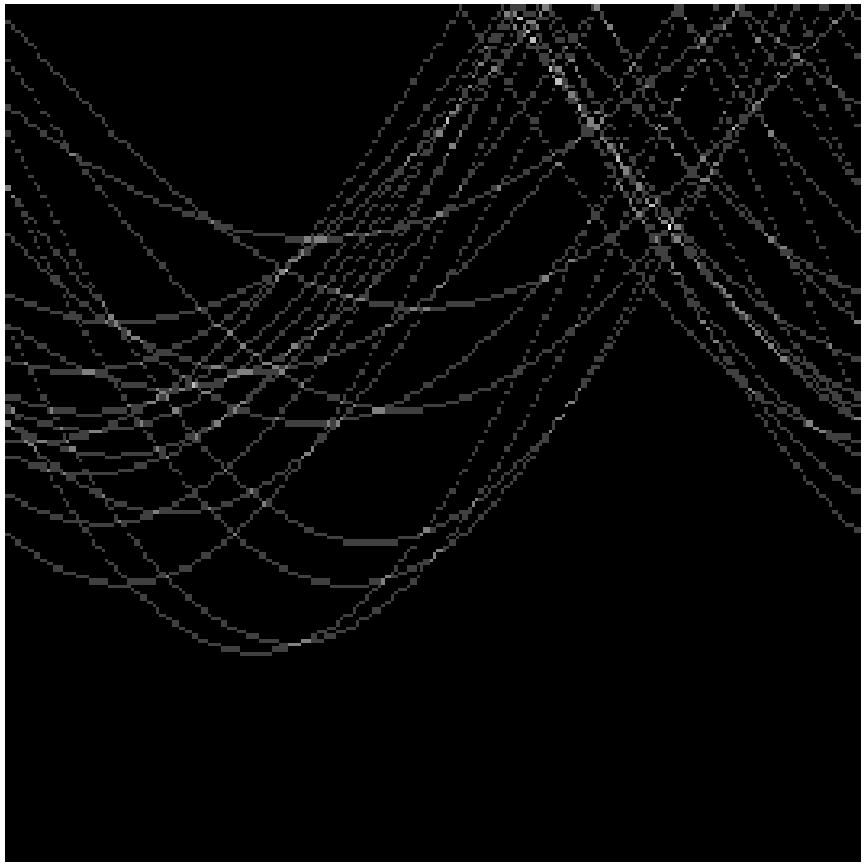
Need to adjust grid size or smooth

Slide from S. Savarese

Hough transform - experiments



features



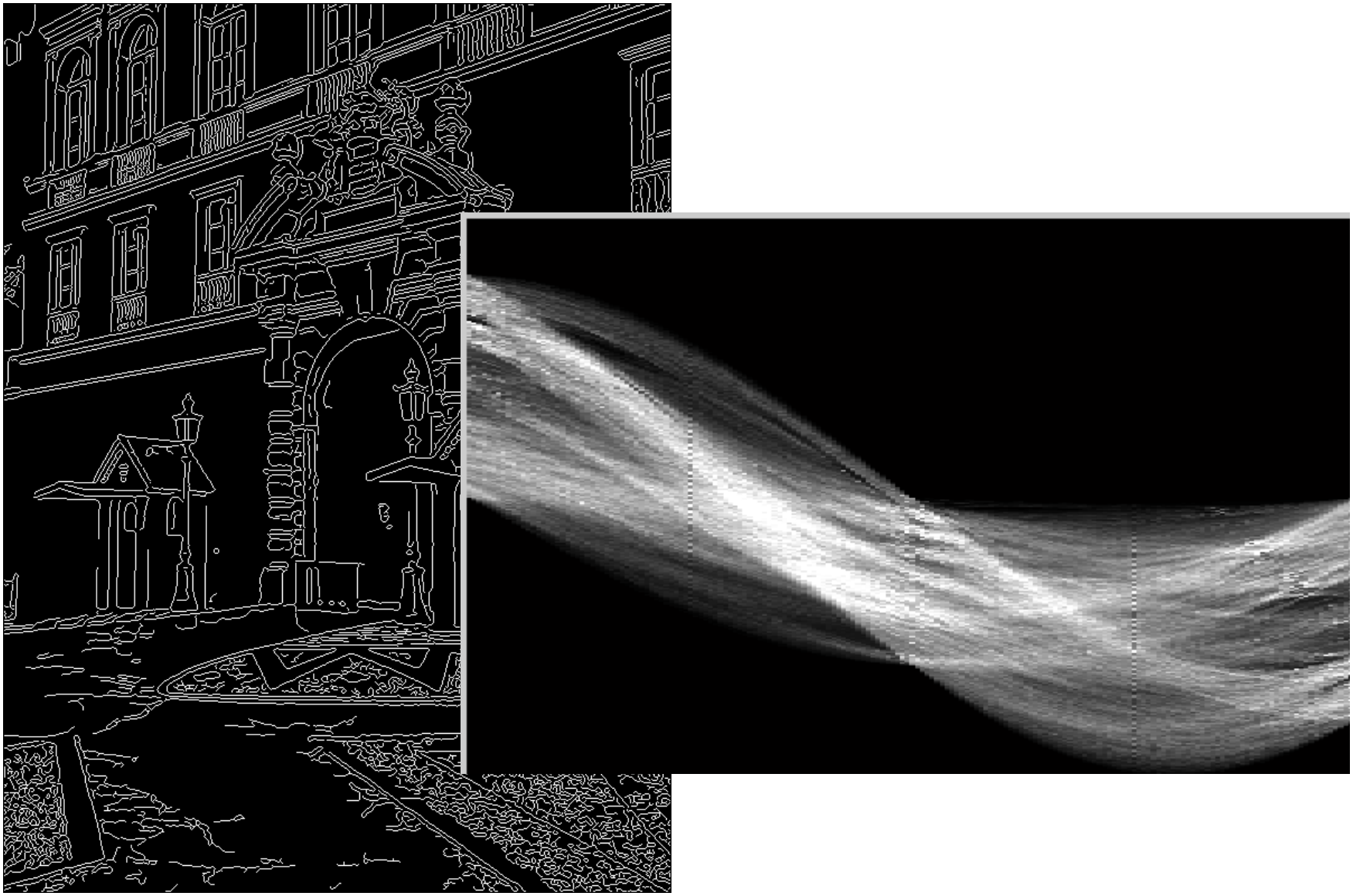
votes

Issue: spurious peaks due to uniform noise

Slide from S. Savarese

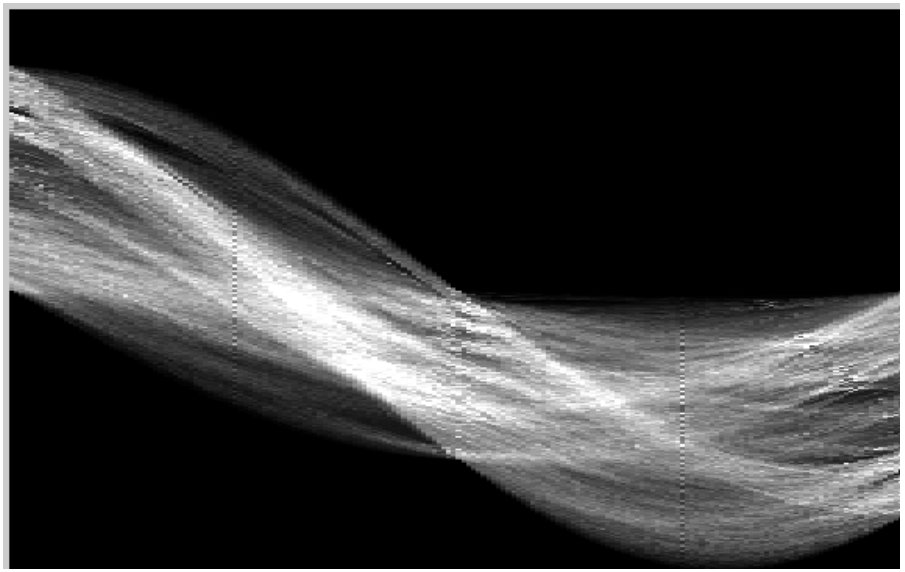
1. Image → Canny

2. Canny → Hough votes

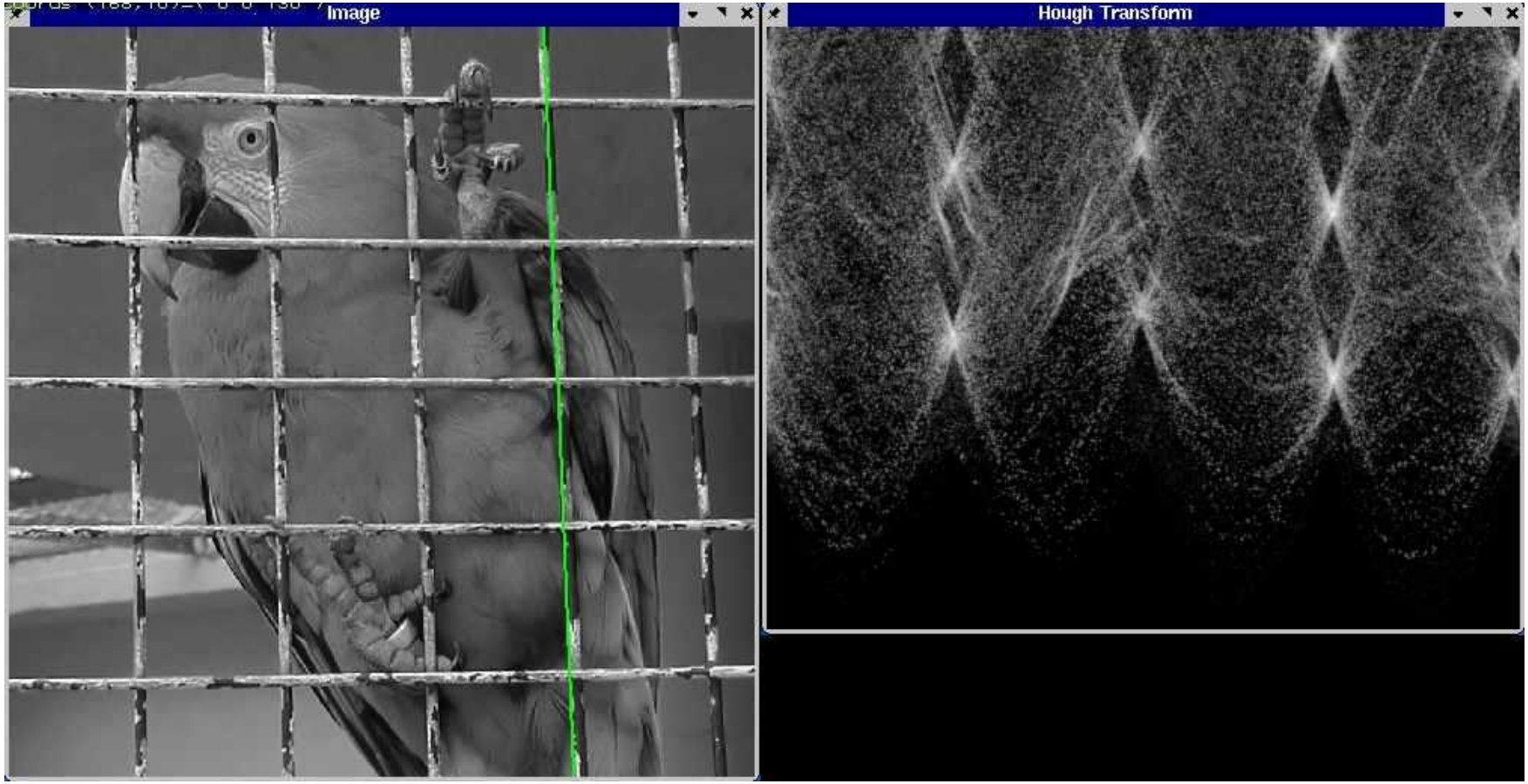


3. Hough votes → Edges

Find peaks and post-process



Hough transform example



http://ostatic.com/files/images/ss_hough.jpg

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Equation of circle?

- For a fixed radius r

Equation of set of circles that all pass through a point?

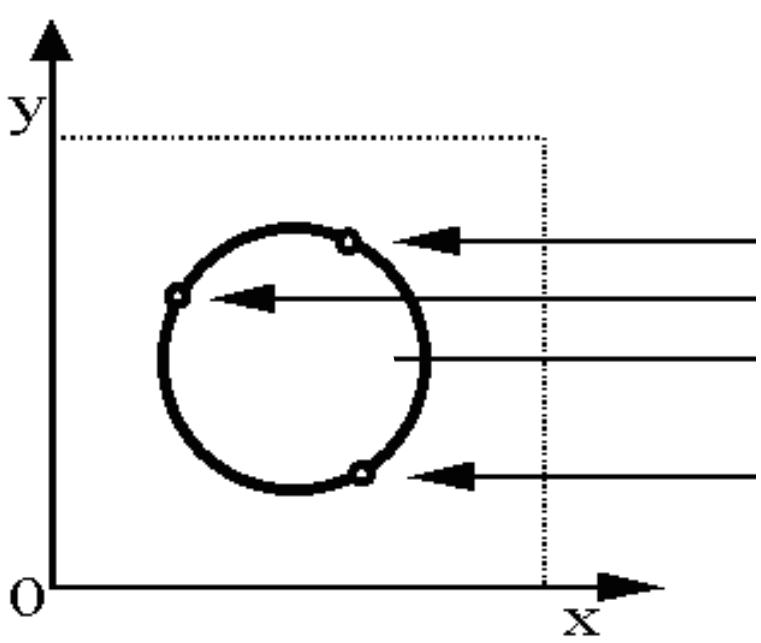
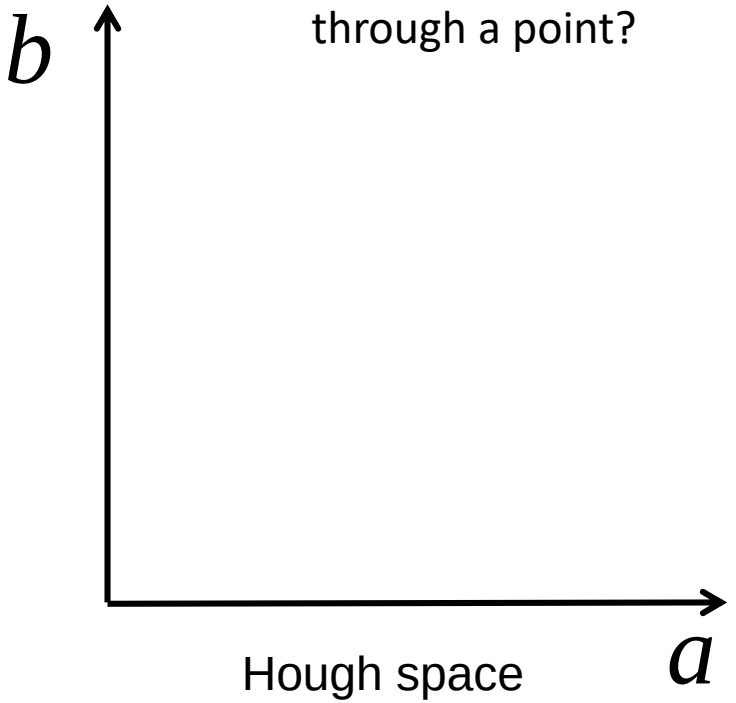


Image space



Hough space

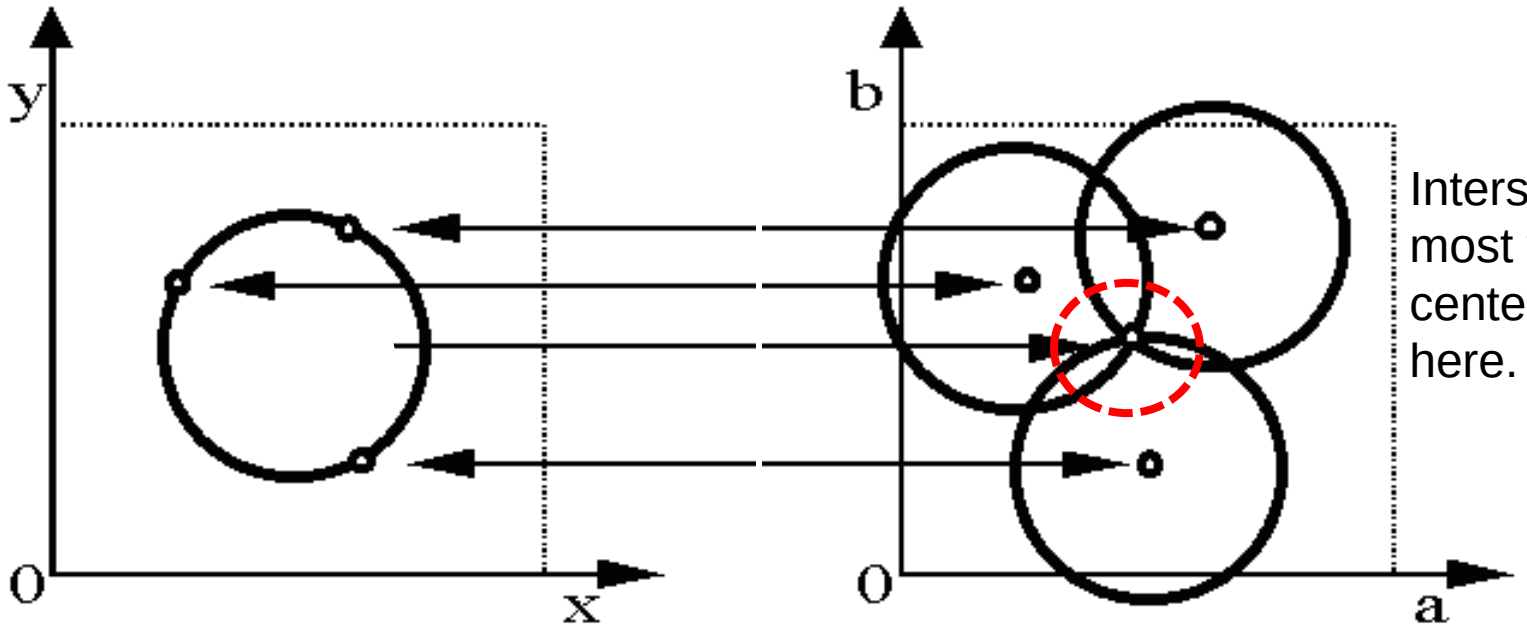
Adapted by Devi Parikh from: Kristen Grauman

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For a fixed radius r

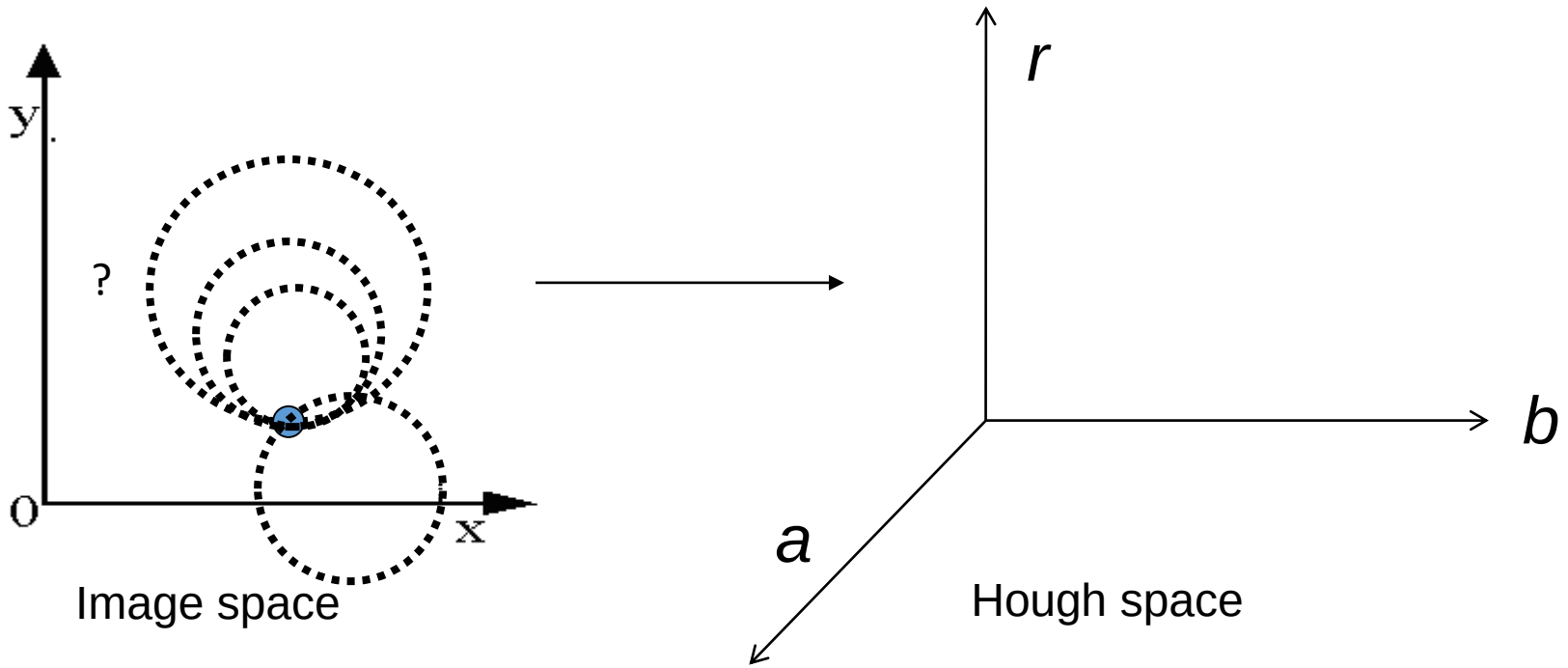


Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r



Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r

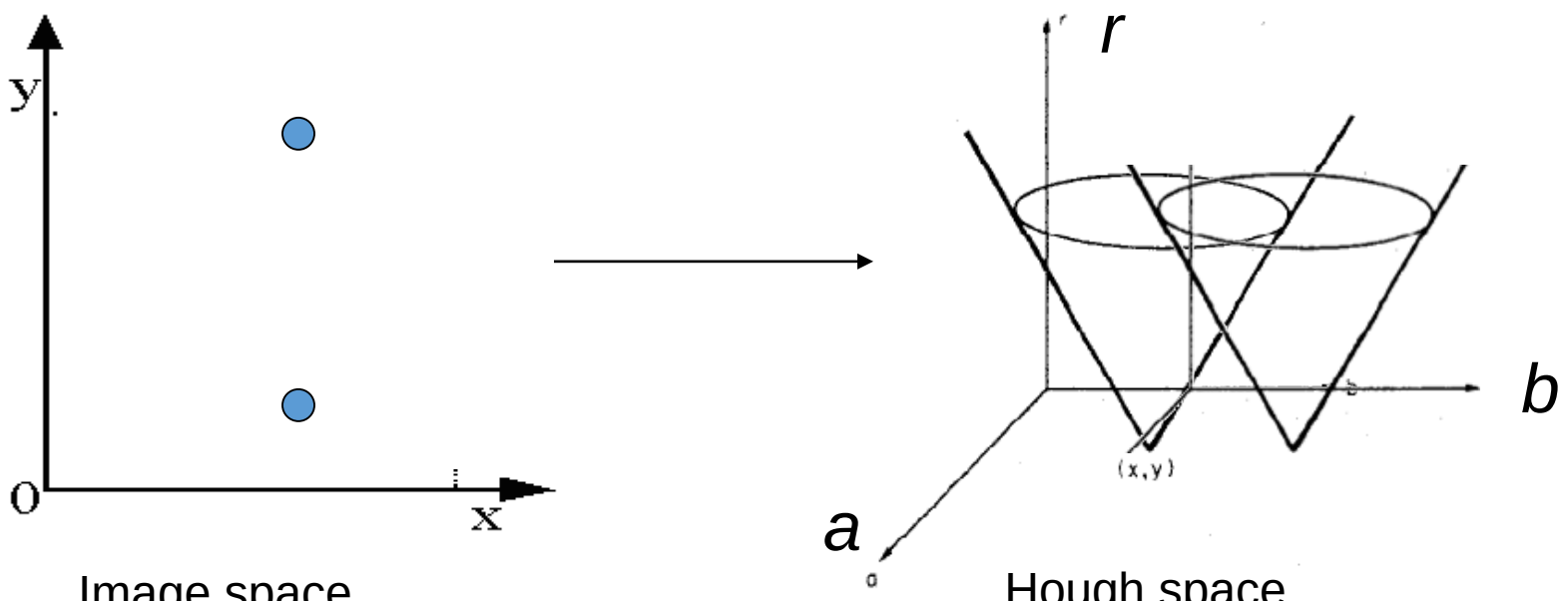


Image space

Hough space

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r, **known** gradient direction

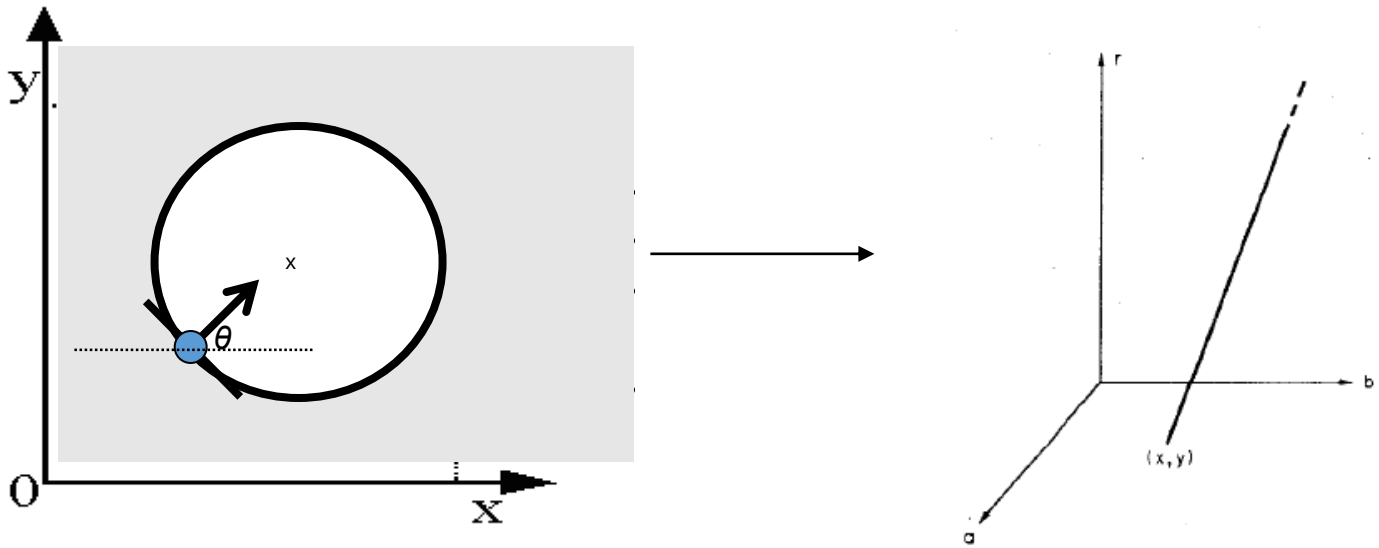
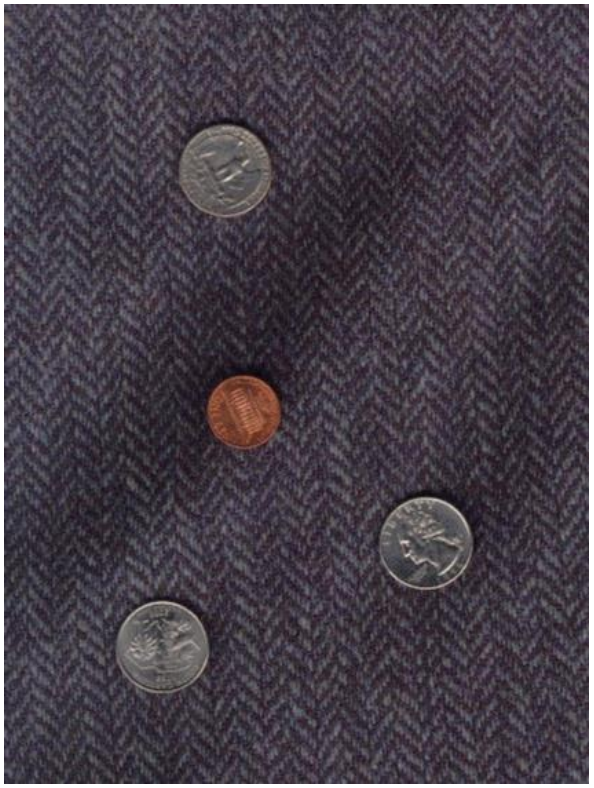


Image space

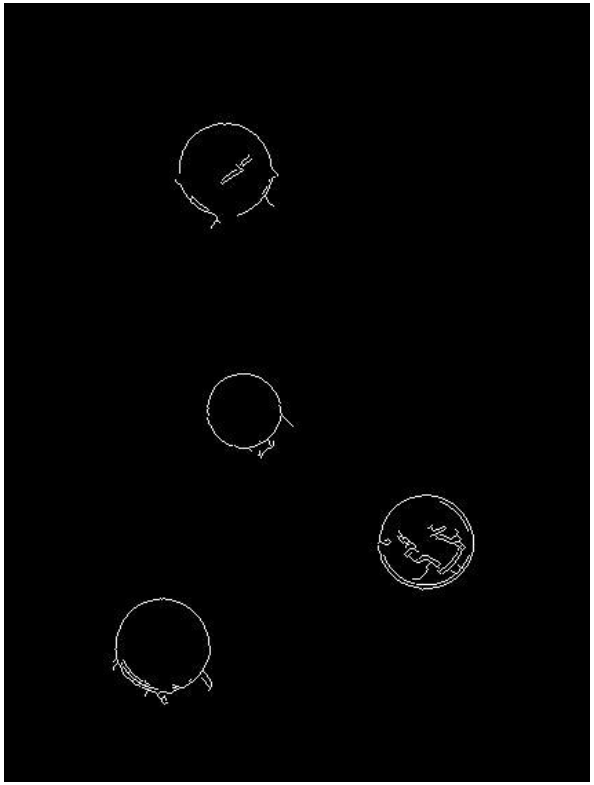
Hough space

Example: detecting circles with Hough

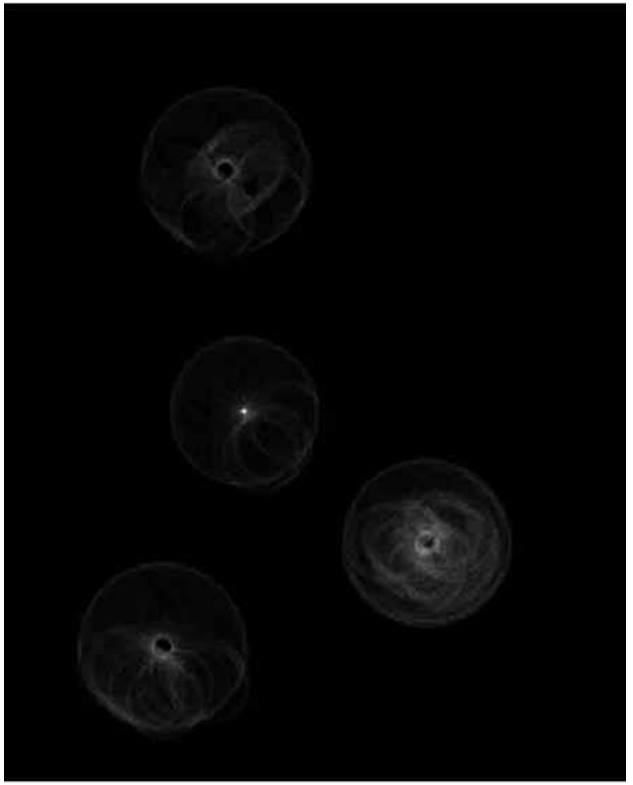
Original



Edges



Votes: Penny

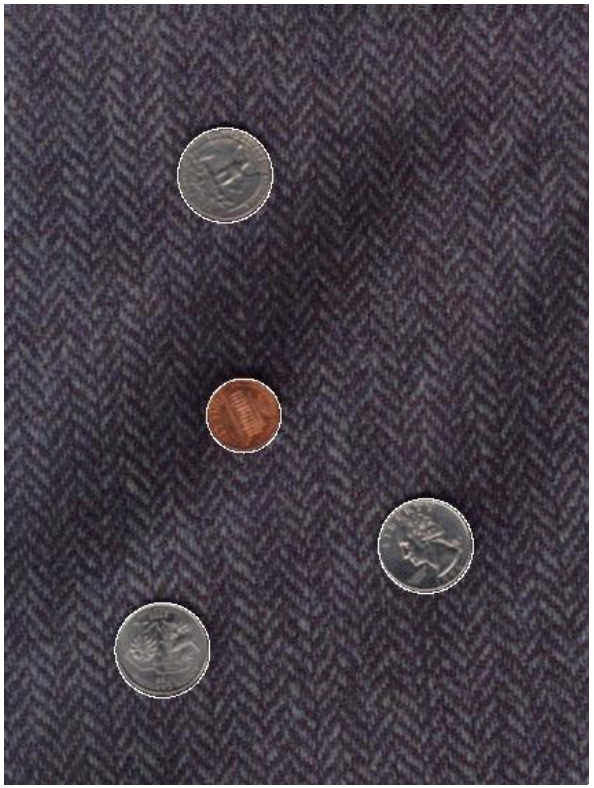


Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

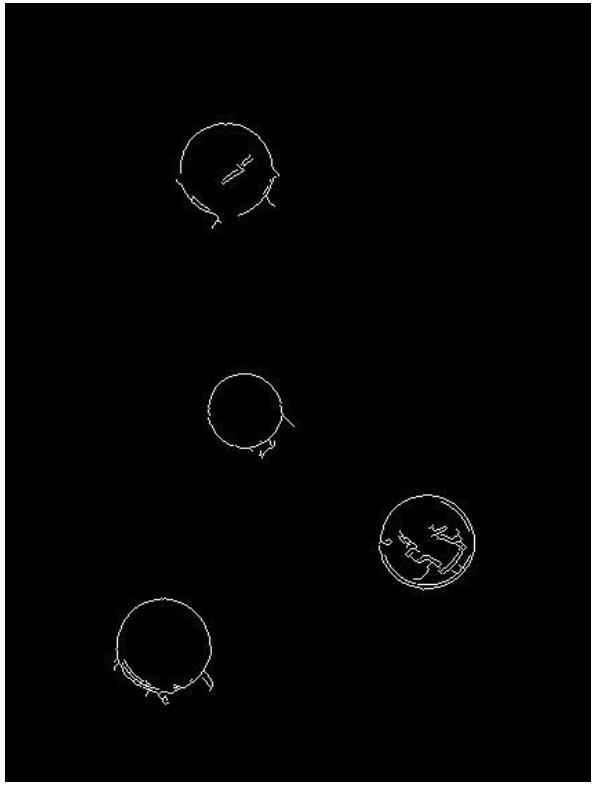
Slide credit: Kristen Grauman

Example: detecting circles with Hough

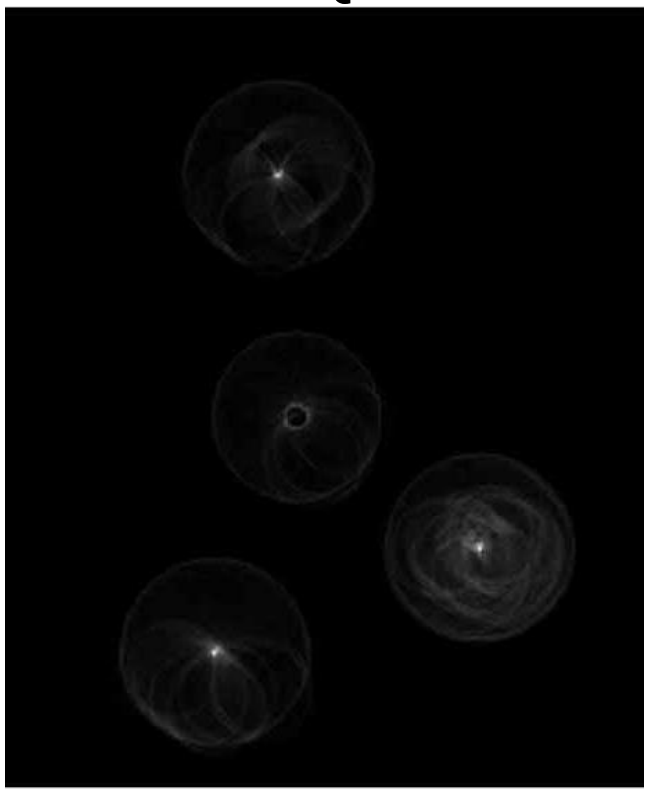
Original



Edges



Votes: Quarter



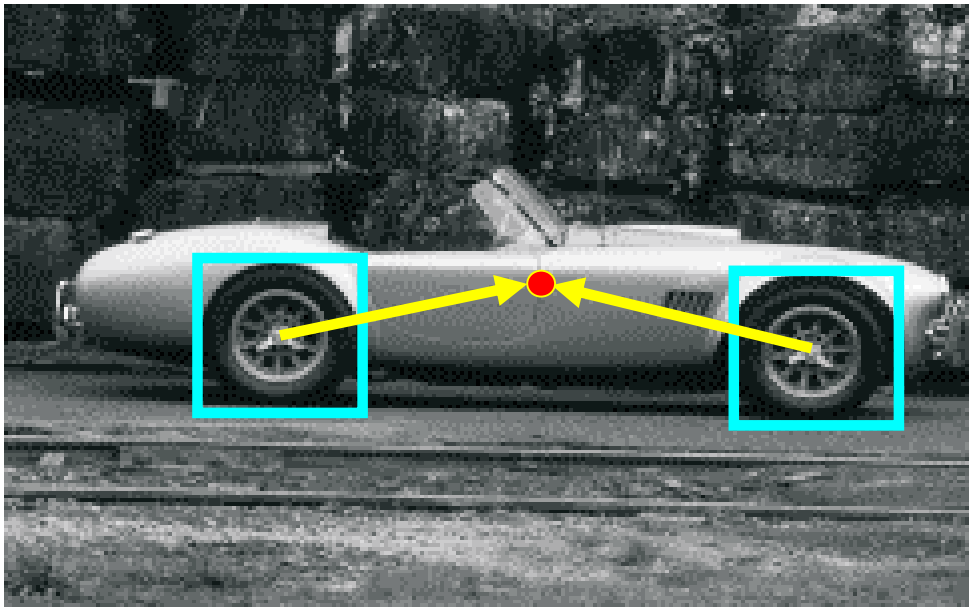
Combined detections

Slide credit: Kristen Grauman

Coin finding sample images from: Vivek Kwatra

Generalized Hough for object detection

- Instead of indexing displacements by gradient orientation, index by matched local patterns.



training image

“visual codeword” with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Source: L. Lazebnik

Generalized Hough for object detection

- Instead of indexing displacements by gradient orientation, index by “visual codeword”

test image

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Source: L. Lazebnik

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

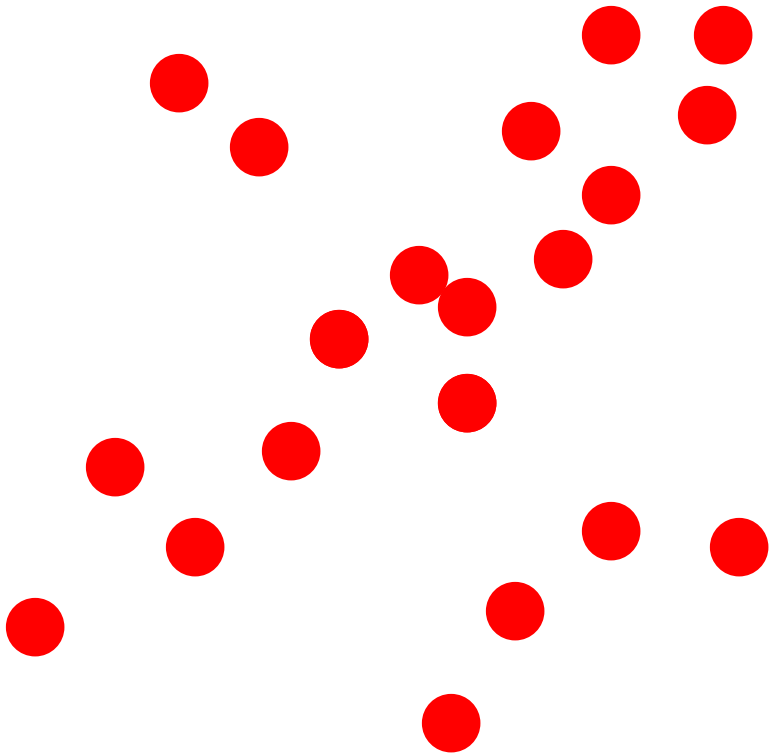
Common applications

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

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



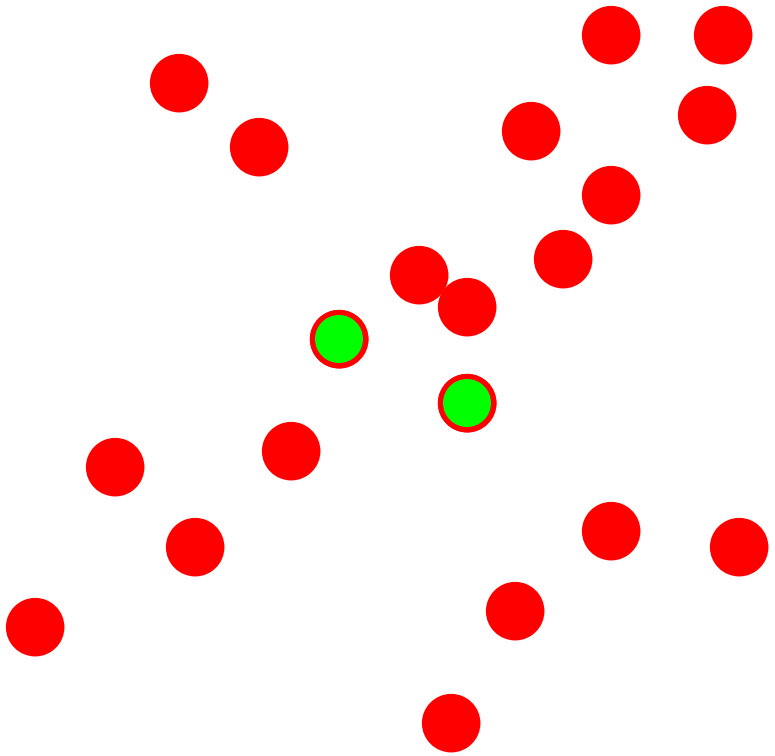
Algorithm:

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:

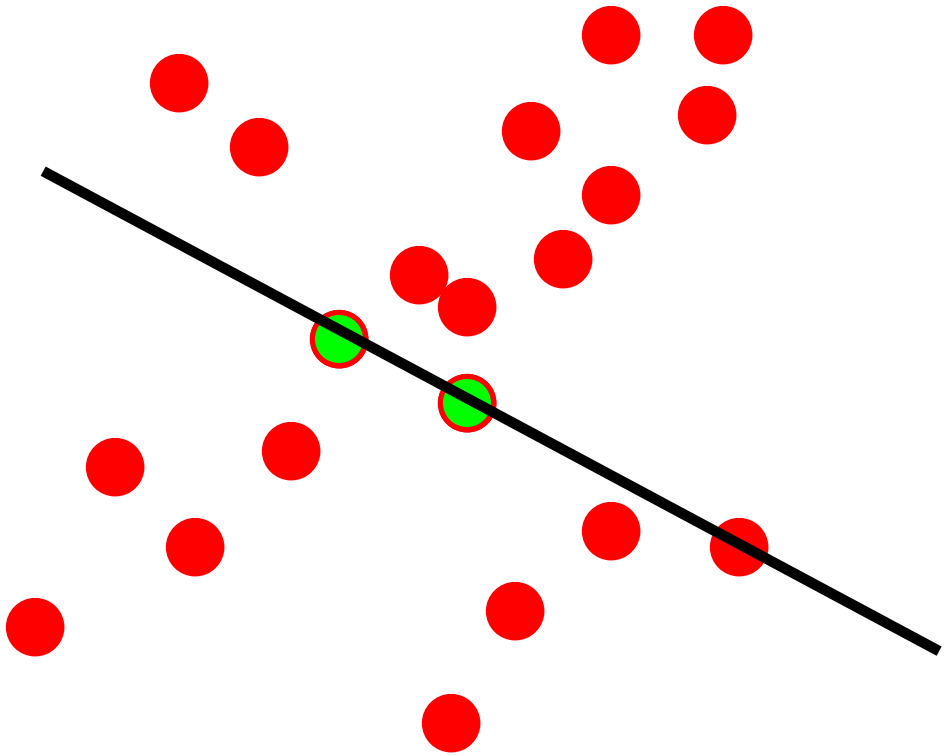
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

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

Illustration by Savarese

RANSAC

Line fitting example



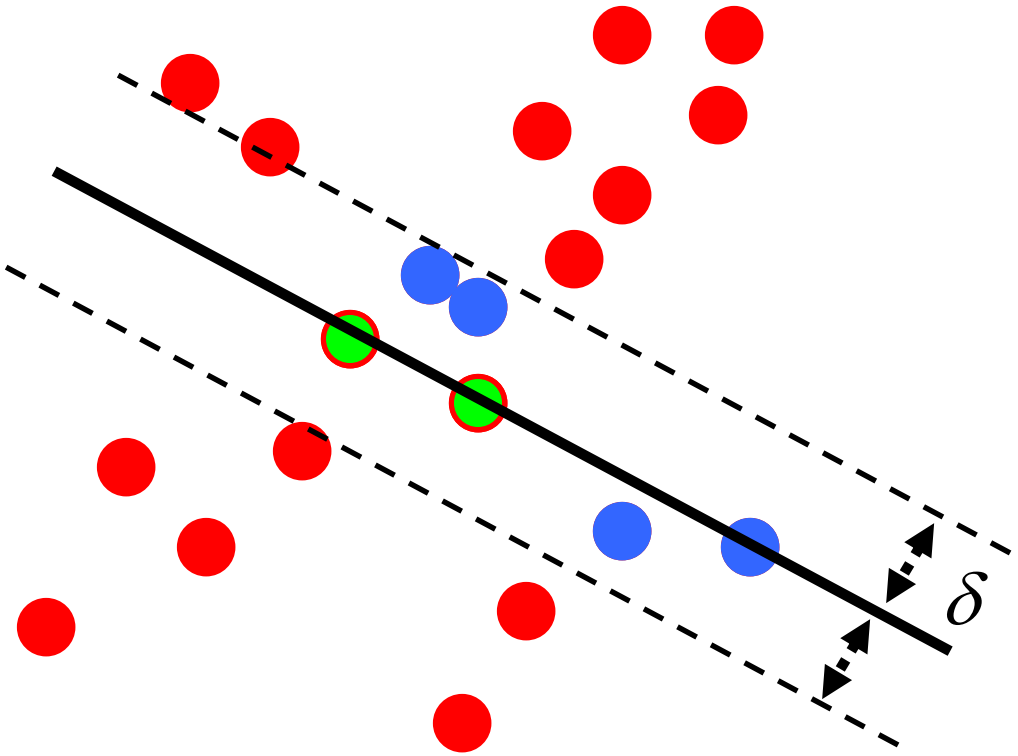
Algorithm:

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



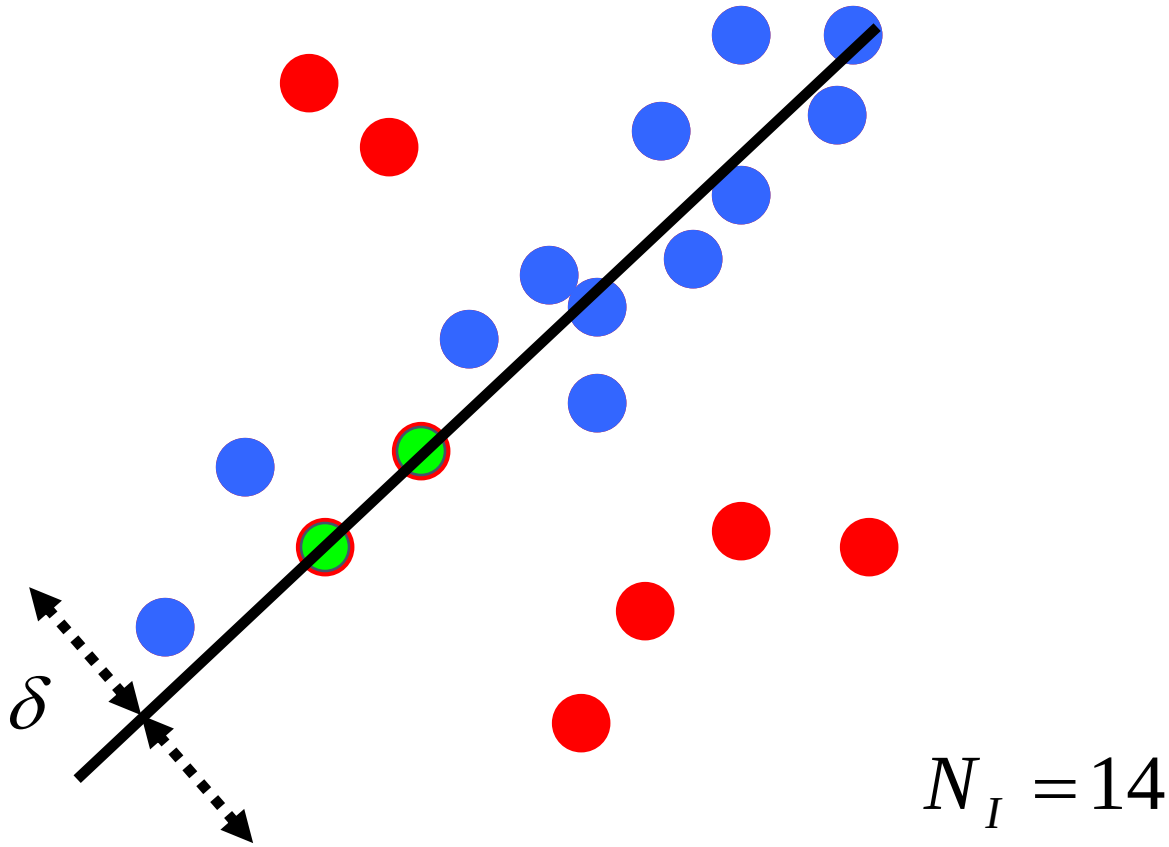
$$N_I = 6$$

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

RANSAC conclusions

Good

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

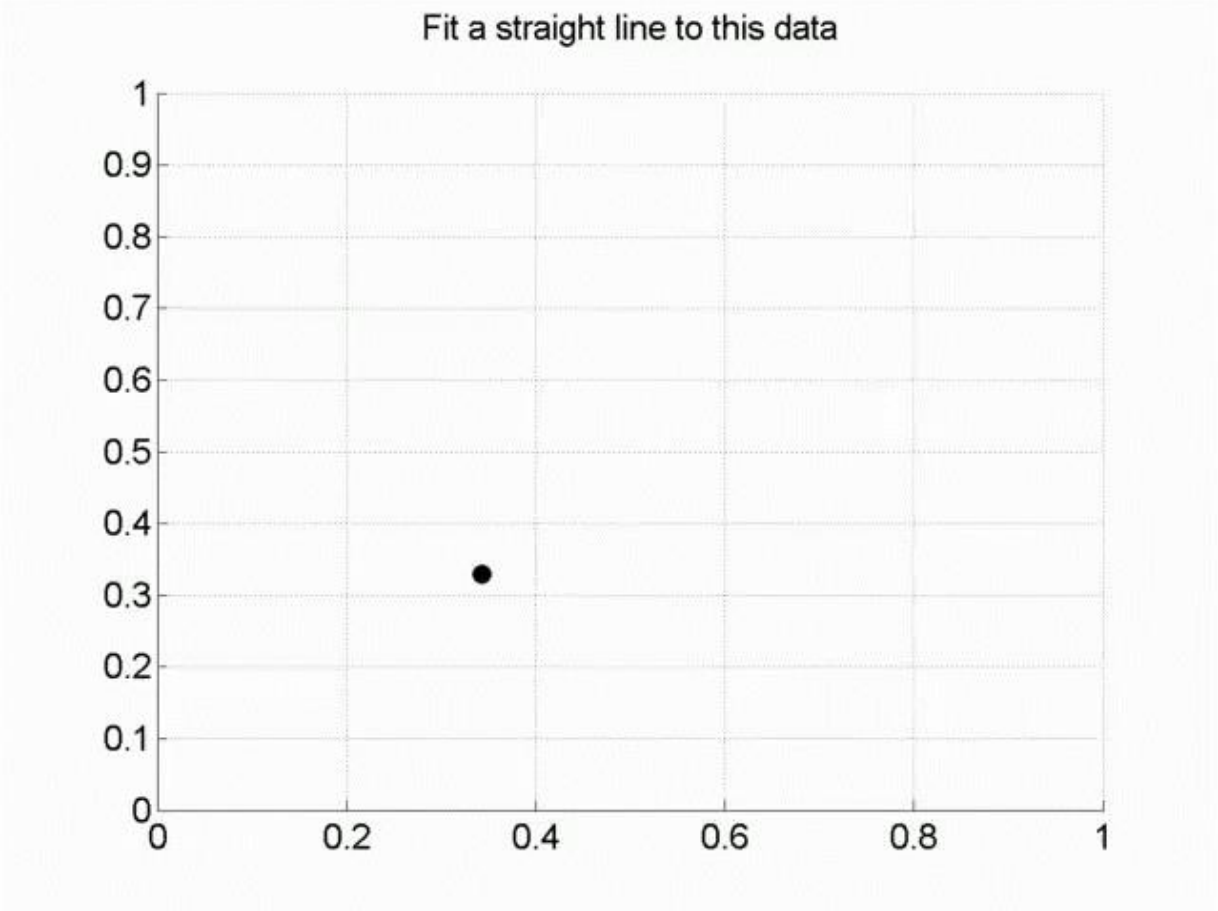
Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not as good for getting multiple fits (though one solution is to remove inliers after each fit and repeat)

Common applications

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

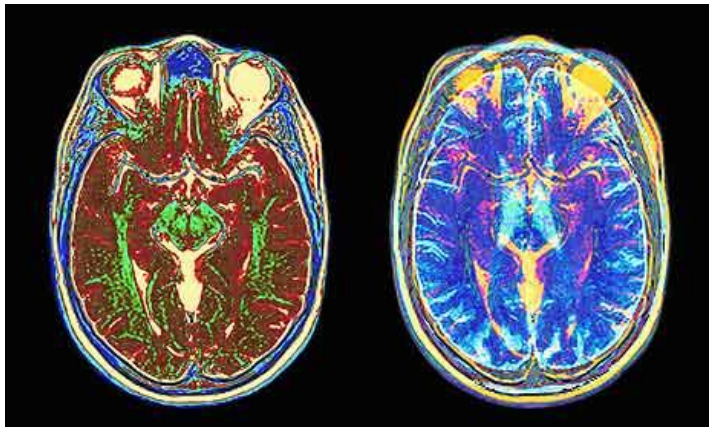
RANSAC Song


Fit a straight line to this data
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1

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

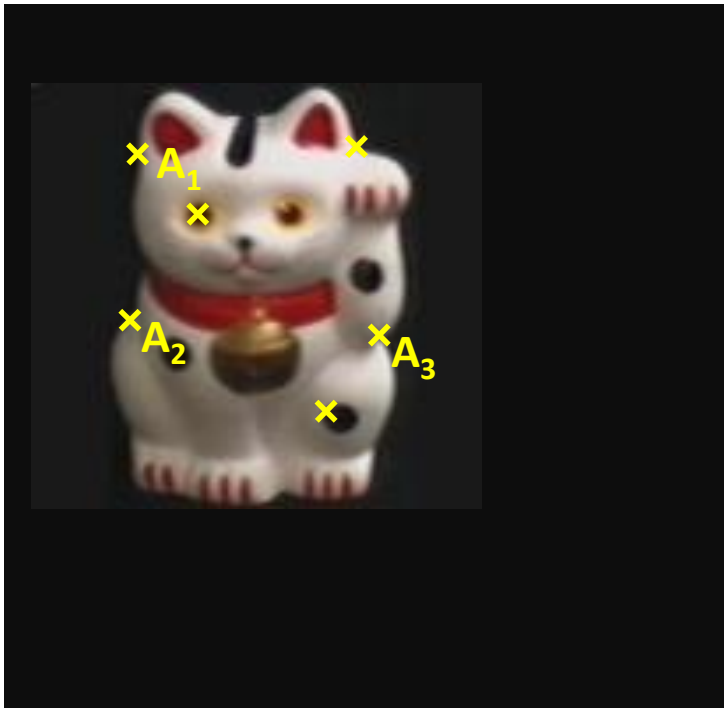
Robotics: match point clouds

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

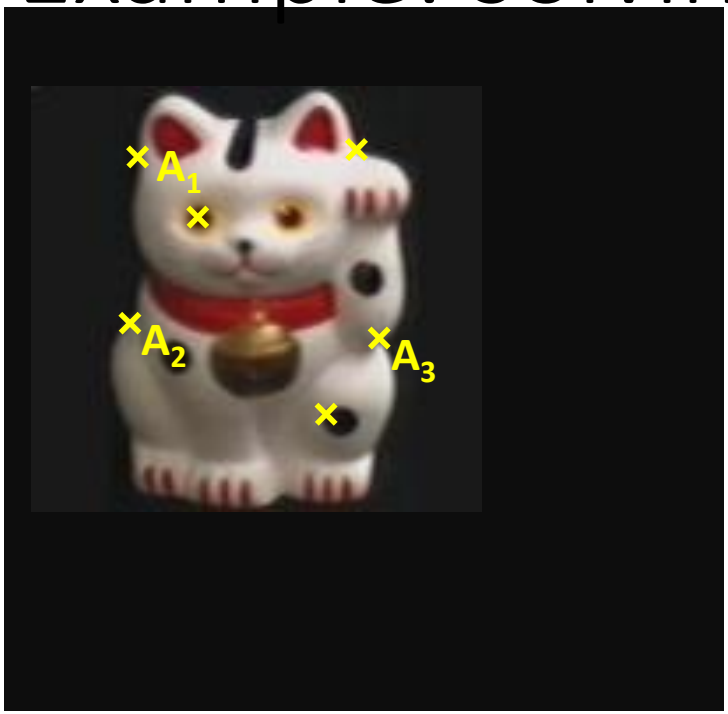
Example: solving for translation



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)

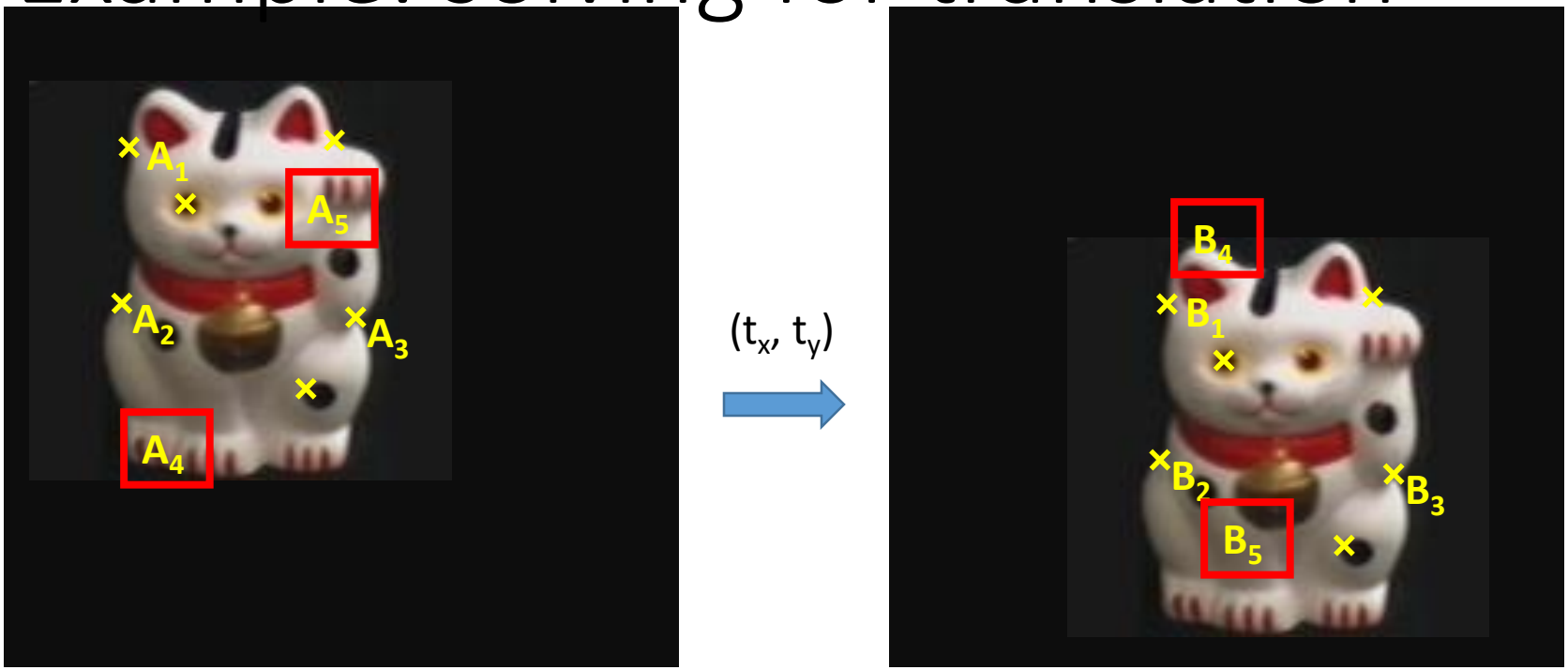
Least squares solution

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



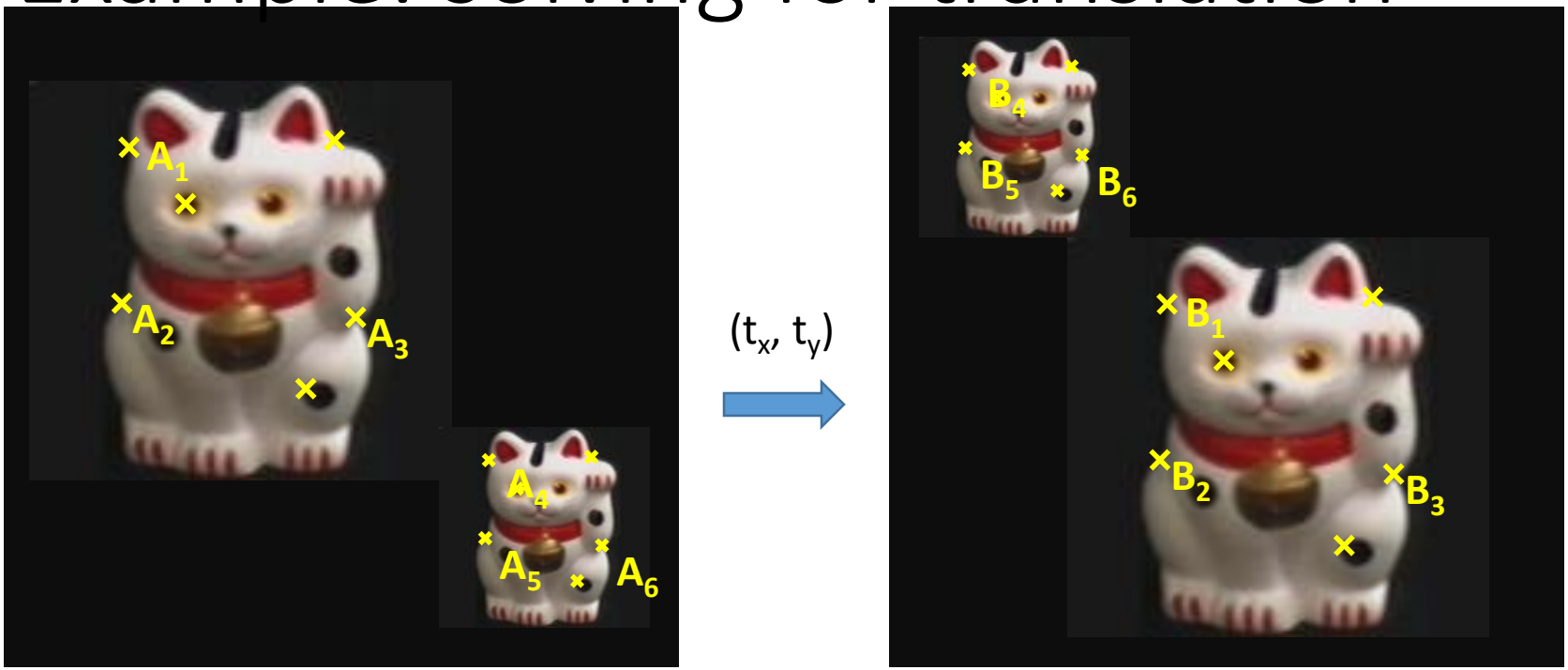
Problem: outliers

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



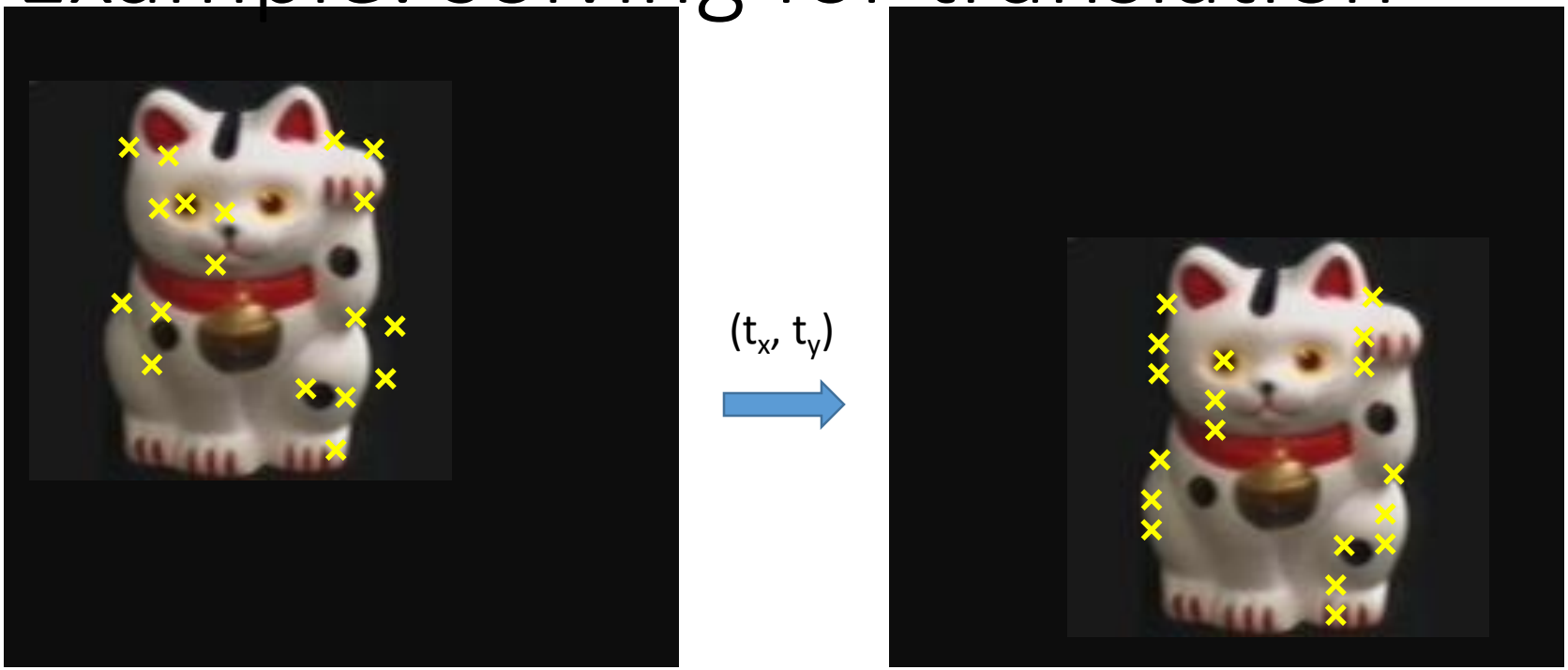
Problem: outliers, multiple objects, and/or many-to-one matches

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

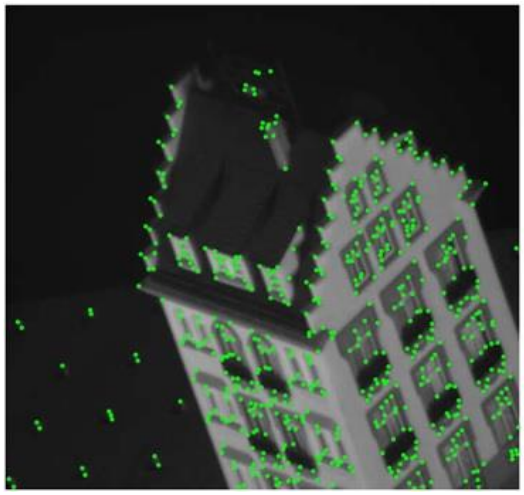
ICP solution

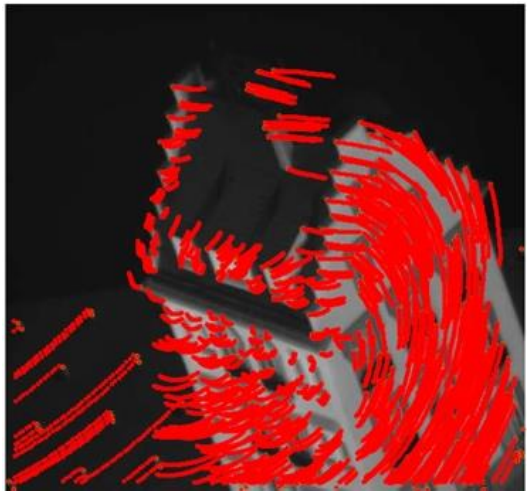
1. Find nearest neighbors for each point
2. Compute transform using matches
3. Move points using transform
4. Repeat steps 1-3 until convergence

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

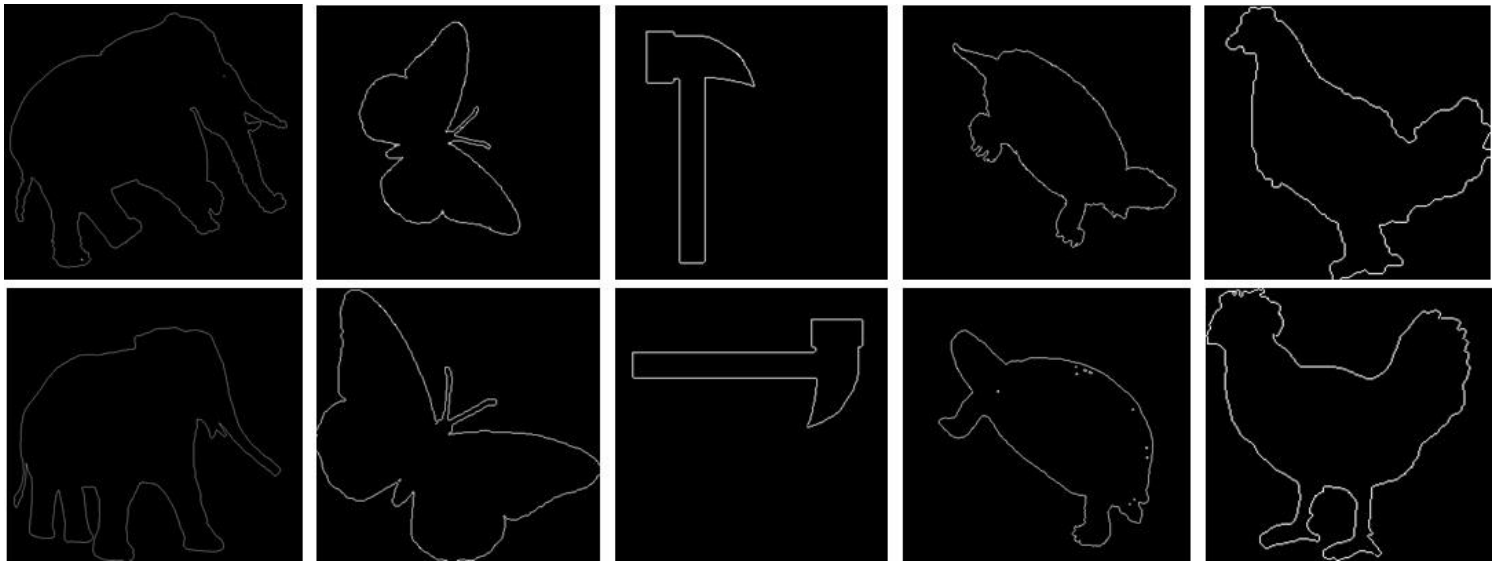
HW 2 – Feature tracker

- Keypoint detection
  - Compute second moment matrix
  - Harris corner criterion
  - Threshold
  - Non-maximum suppression
- Tracking
  - Kanade-Lucas-Tomasi tracking
  - Show keypoint trajectories
  - Show points have moved out of frames





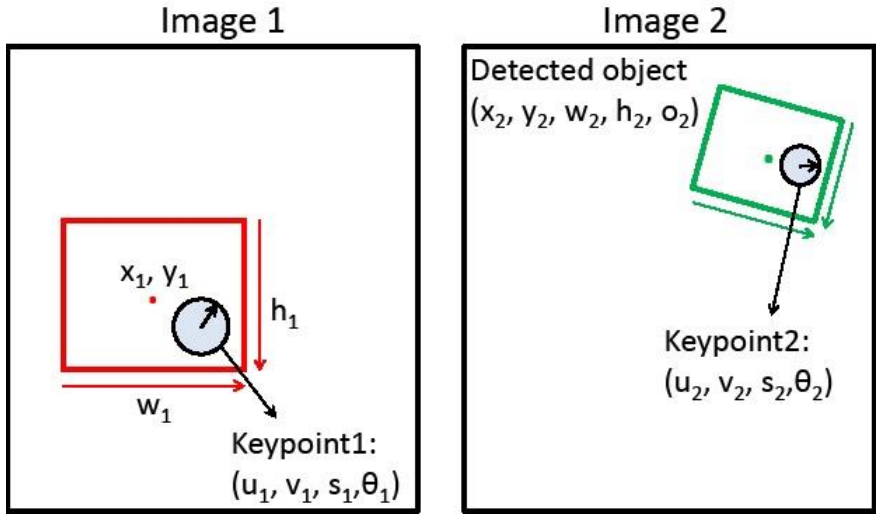
HW 2 – Shape Alignment



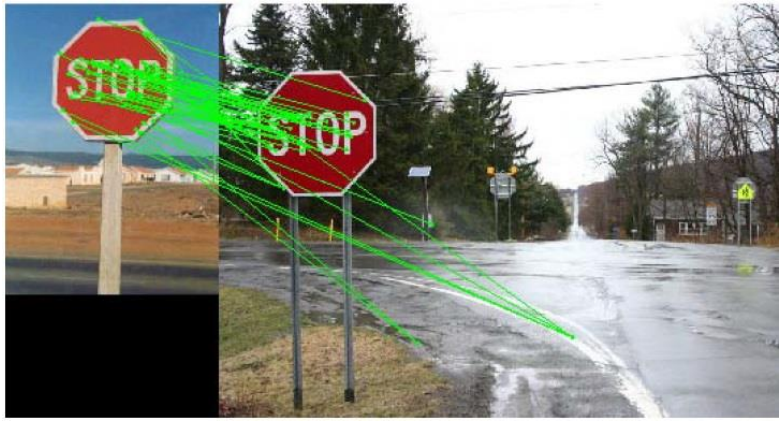
- Global transformation (similarity, affine, perspective)
- Iterative closest point algorithm

HW 2 – Local Feature Matching

- Express location of the detected object



- Implement ratio test feature matching algorithm



Things to remember

- 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

Next week

- Object instance recognition