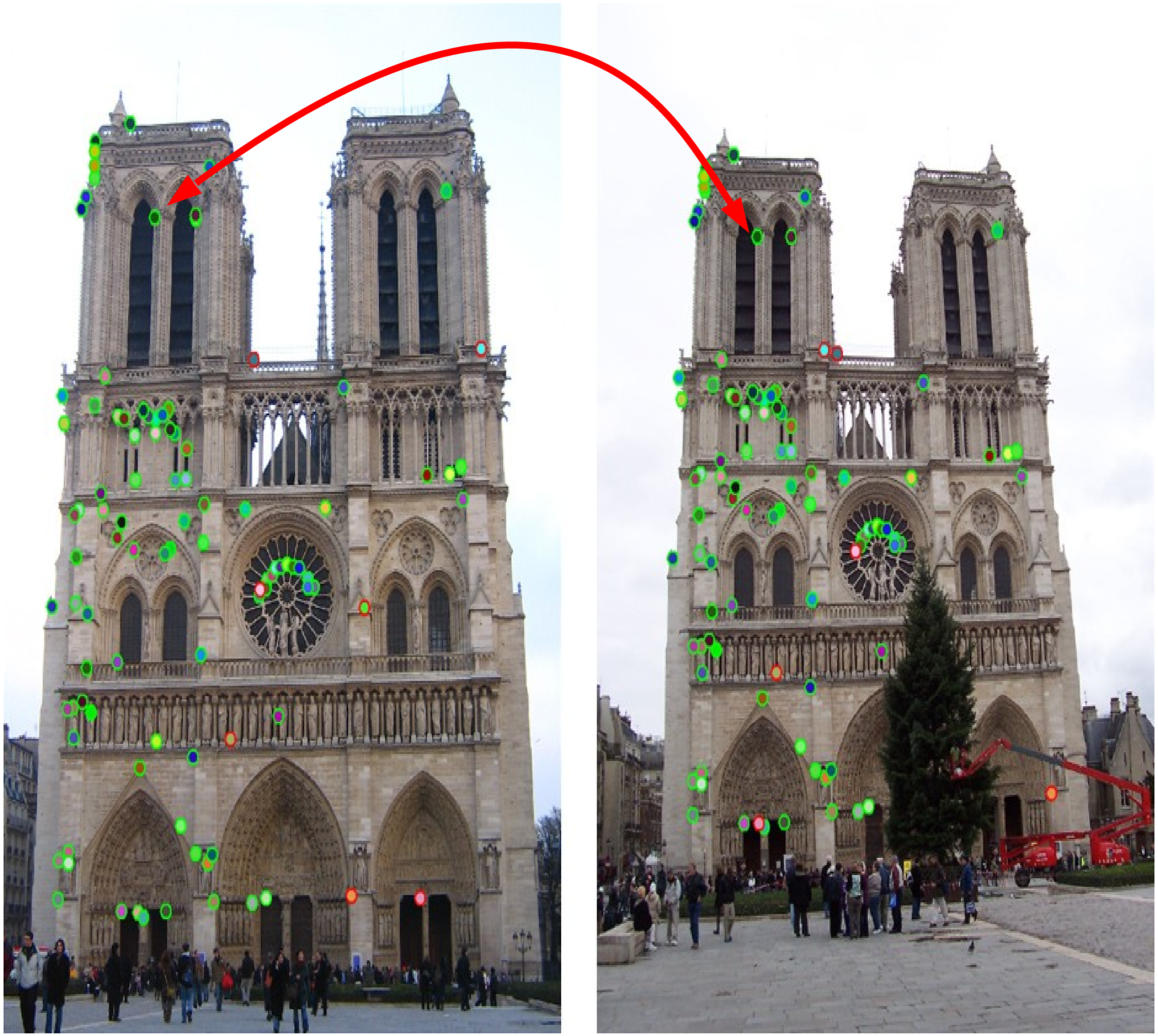
Lecture 5
Model Fitting and Optimization: Least Squares, Hough Transforms, RANSAC

COS 429: Computer Vision

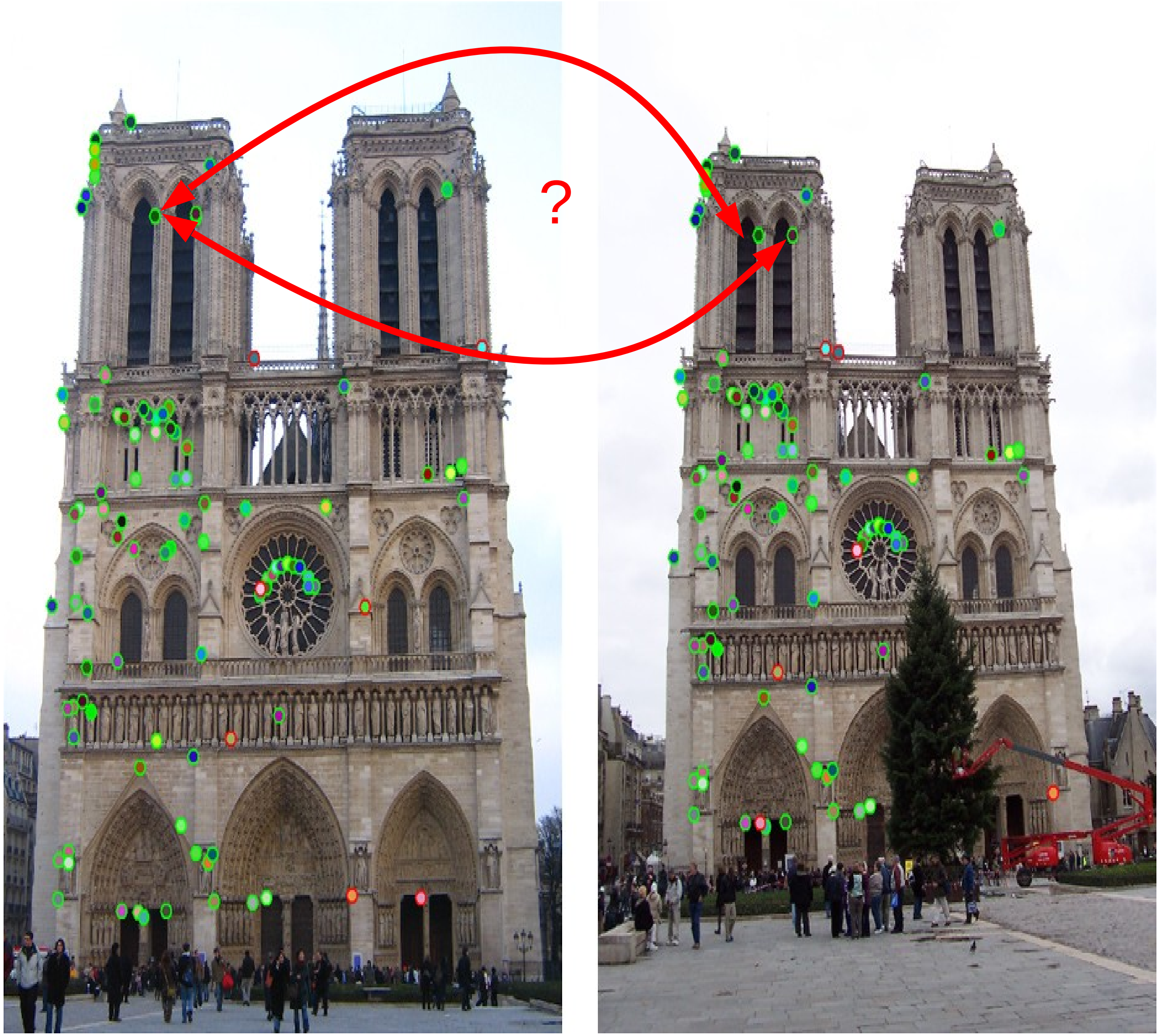
PRINCETON UNIVERSITY

Review: Feature Matching



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

Matching ambiguity



Locally feature matches are ambiguous

=> need to fit a **model** to find globally consistent matches

Model Fitting & Optimization

- Design challenges
  - Design an appropriate **model** [next time]
    - Enough degrees of freedom (DOFs) to allow good mapping
    - As few DOFs as possible to enable good fitting
  - Design a suitable **goodness of fit** measure between data and model
    - Similarity should reflect application goals
    - Encode robustness to outliers and noise
  - Design an **optimization** method to find parameters of model
    - Avoid local optima
    - Find best parameters quickly

Goodness of Fit: Loss Function

Model:

Input data

Model Parameters

Model estimate

$$\hat{z} = f(X; \Theta)$$

Loss Function:

$$L(Z; \hat{Z})$$

Output data

Estimated Output

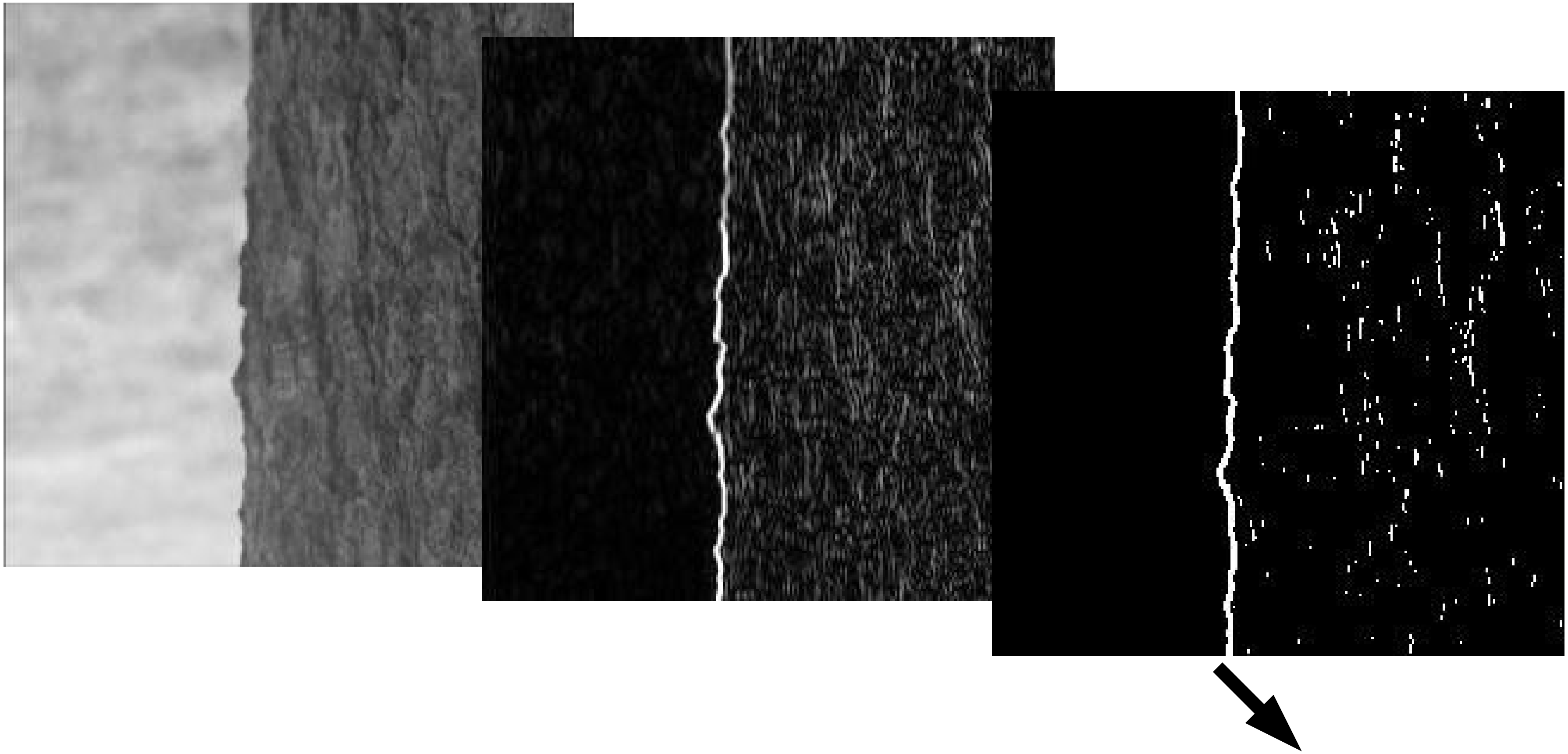
Goal of Optimization:

$$\underset{\Theta}{argmin} \sum_i L(Z_i; f(X; \Theta))$$

Also know as: cost, objective function
Negative of loss: reward, profit, utility, fitness function

1D Starter Example: Find the Vertical Edge

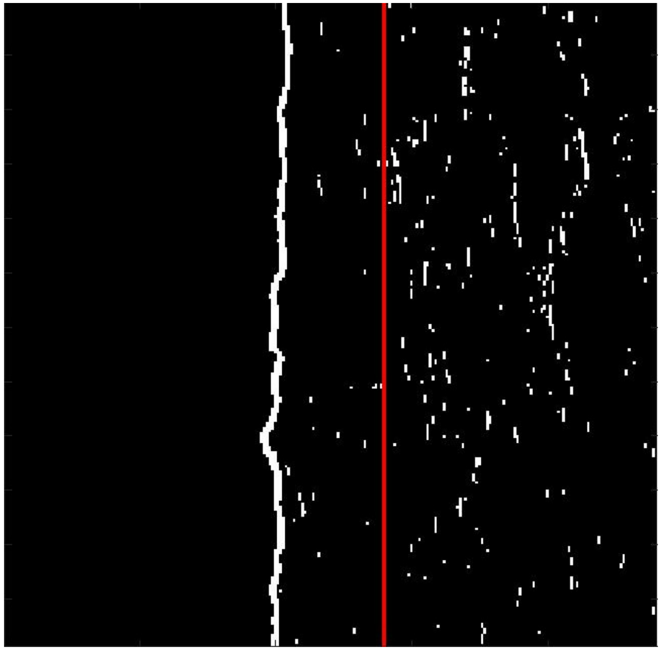
1D Starter Example: Find the Vertical Edge



[List of <x,y> coordinates]

```
dx = abs(conv2(im, [1 2 1]'*[-1 0 1]/4, 'valid')); %dx filter
dxt = dx>=33; %threshold at edge energy=33
[y,x]=find(dxt); % find x,y coordinate of thresholded points
```

Squared Loss (L2)



Looking for a vertical line, so model is

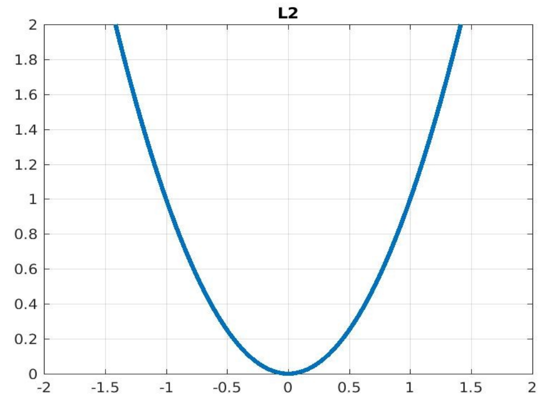
$$\hat{z} = f(\Theta) = \Theta = x\hat{p}os$$

Let's start with L2 (Squared, regression) loss:

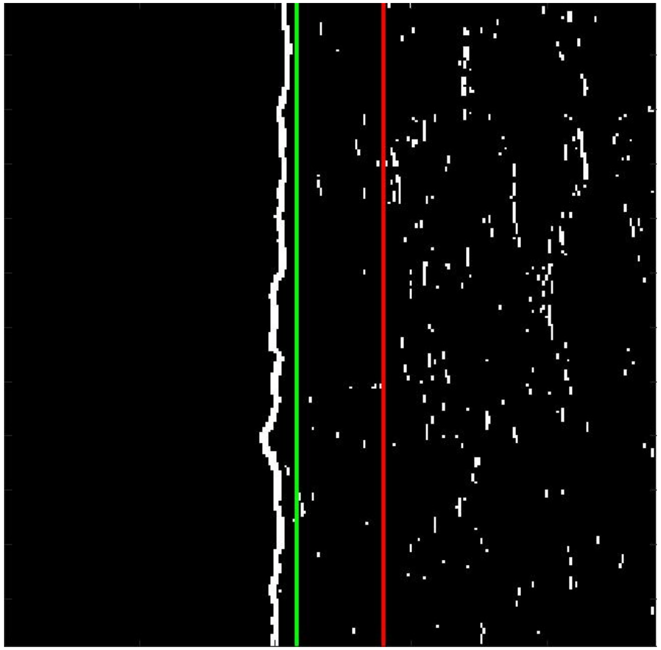
$$L(z_i; \hat{z}) = (z_i - \hat{z})^2$$

And find Θ such that sum(L_i) is minimum.

This is the mean!



Absolute Value Loss (L1)



Looking for a vertical line, so model

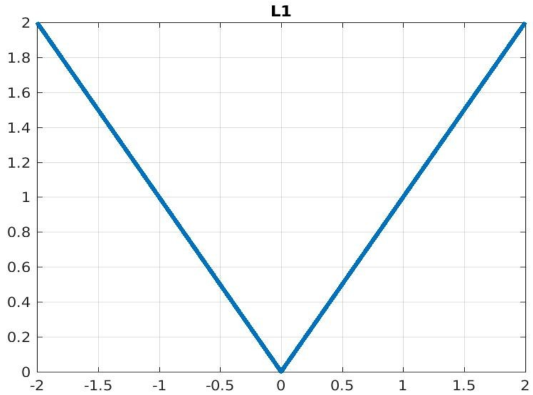
$$\hat{z} = f(\Theta) = \Theta = \hat{xpos}$$

Now try L1 (Absolute Value) loss:

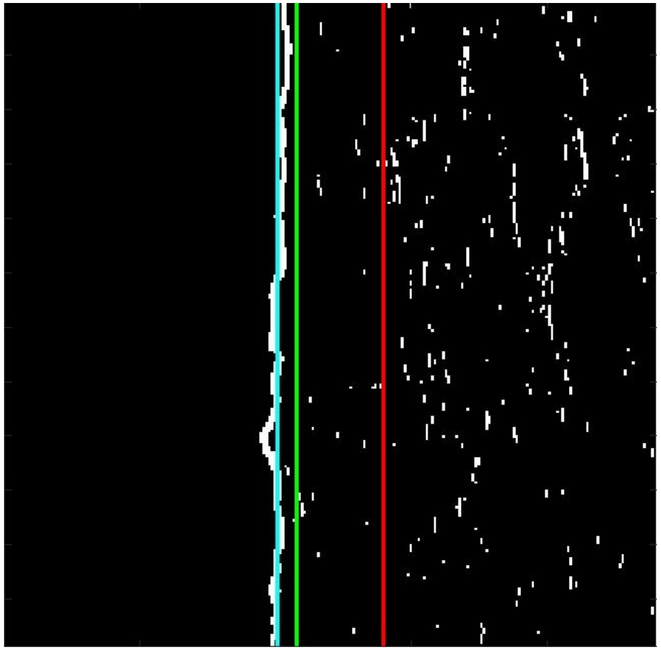
$$L(z_i; \hat{z}) = |z_i - \hat{z}|$$

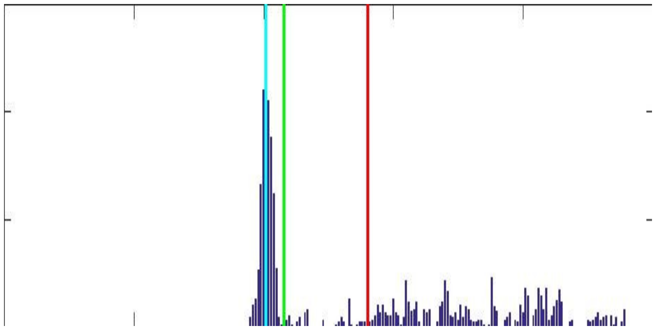
And find Θ such that sum(L_i) is minimum.

This is the median!



Histogram





Looking for a vertical line, so model

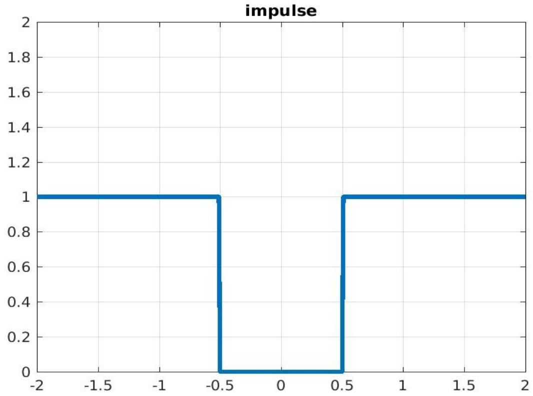
$$\hat{z} = f(\Theta) = \Theta = \hat{xpos}$$

How about just histogram and find maximum most popular bin. You can think of this as an impulse Loss:

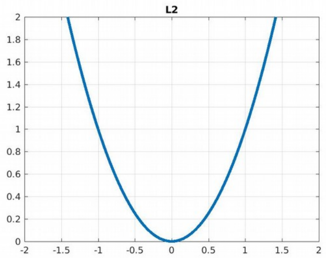
$$L(x;\hat{x}) = (|x_i - \hat{x}| > \gamma)$$

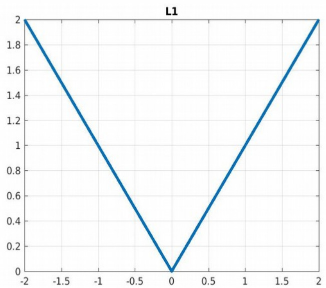
This is the mode!

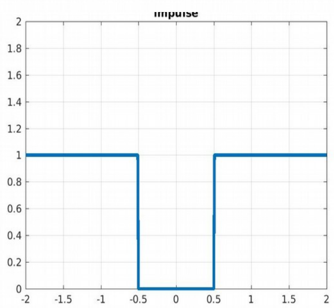
Questions: what happens as you change the bin size? What if you blur the bins of the Histogram?

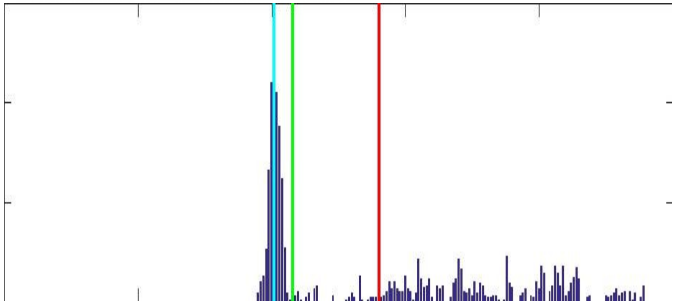


More Robust Distance Loss Functions



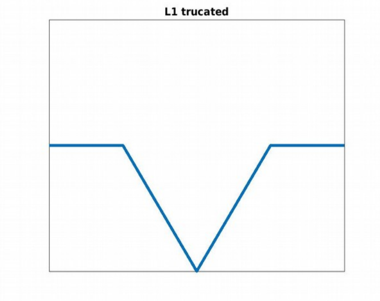






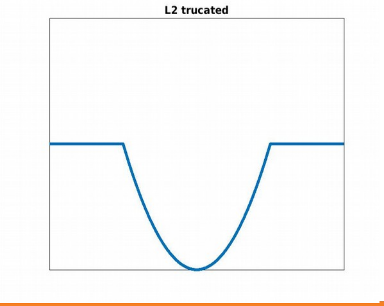
Truncated L1:

$$L(z;\hat{z};\gamma)=(\min(|z-\hat{z}|,\gamma))$$

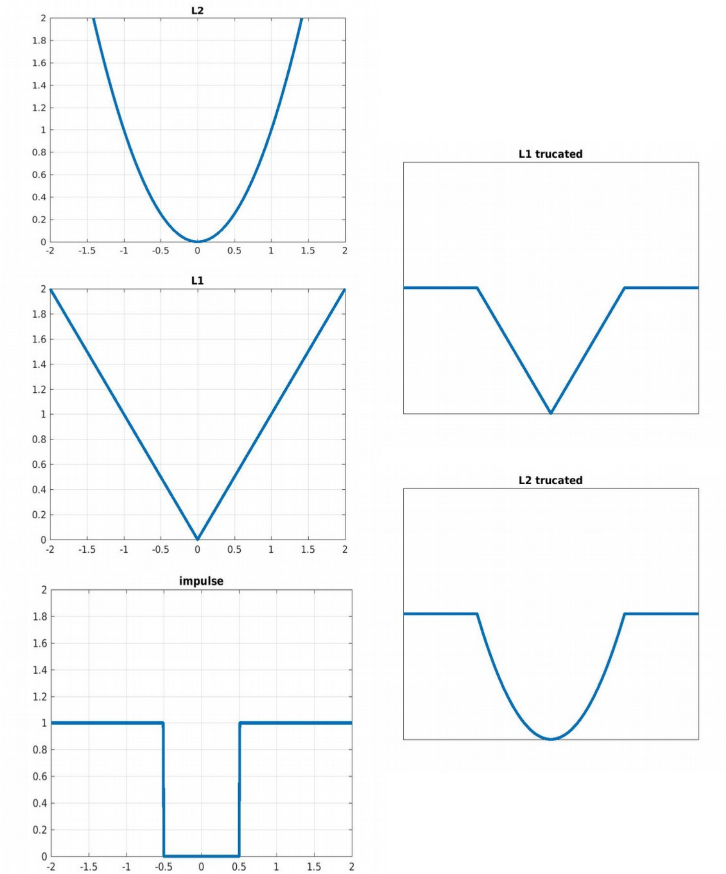


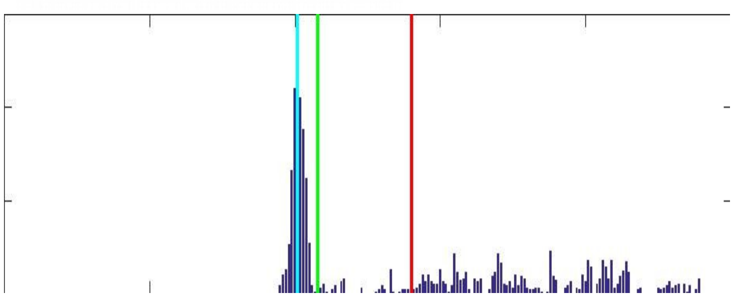
Truncated L2:

$$L(z;\hat{z};\gamma)=(\min((z-\hat{z})^2,\gamma^2))$$



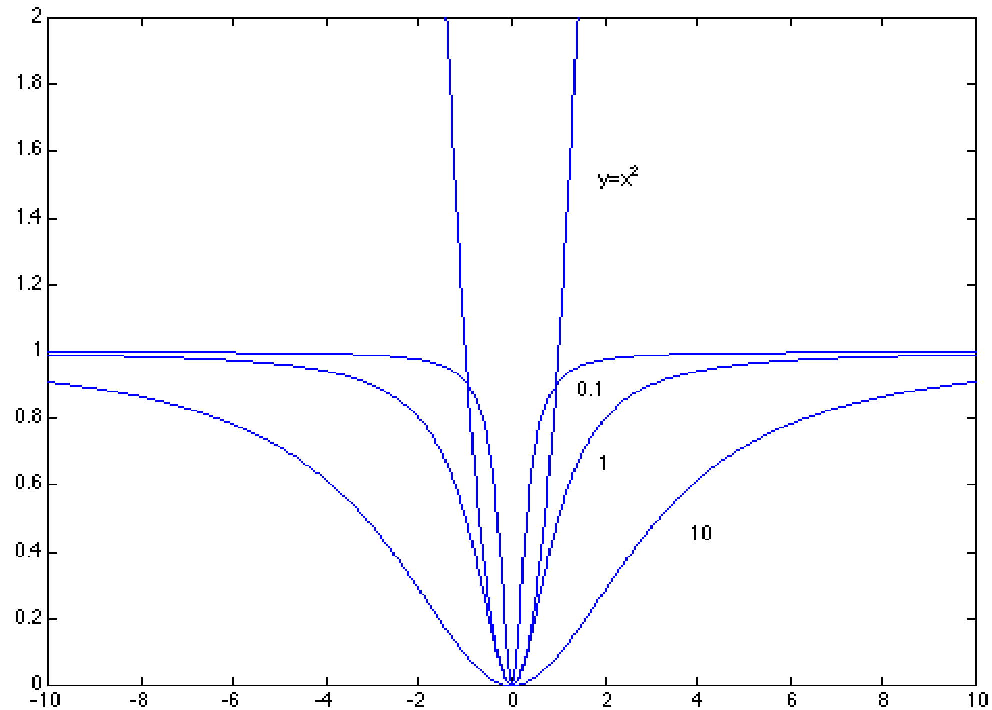
More Robust Distance Loss Functions





Cauchy:

$$L(z;\hat{z};\sigma)=\frac{(z_i-\hat{z})^2}{\sigma^2+(z_i-\hat{z})^2}$$



Terminology

M-estimators - minimize some function other than L2

Search/Optimization Algorithms

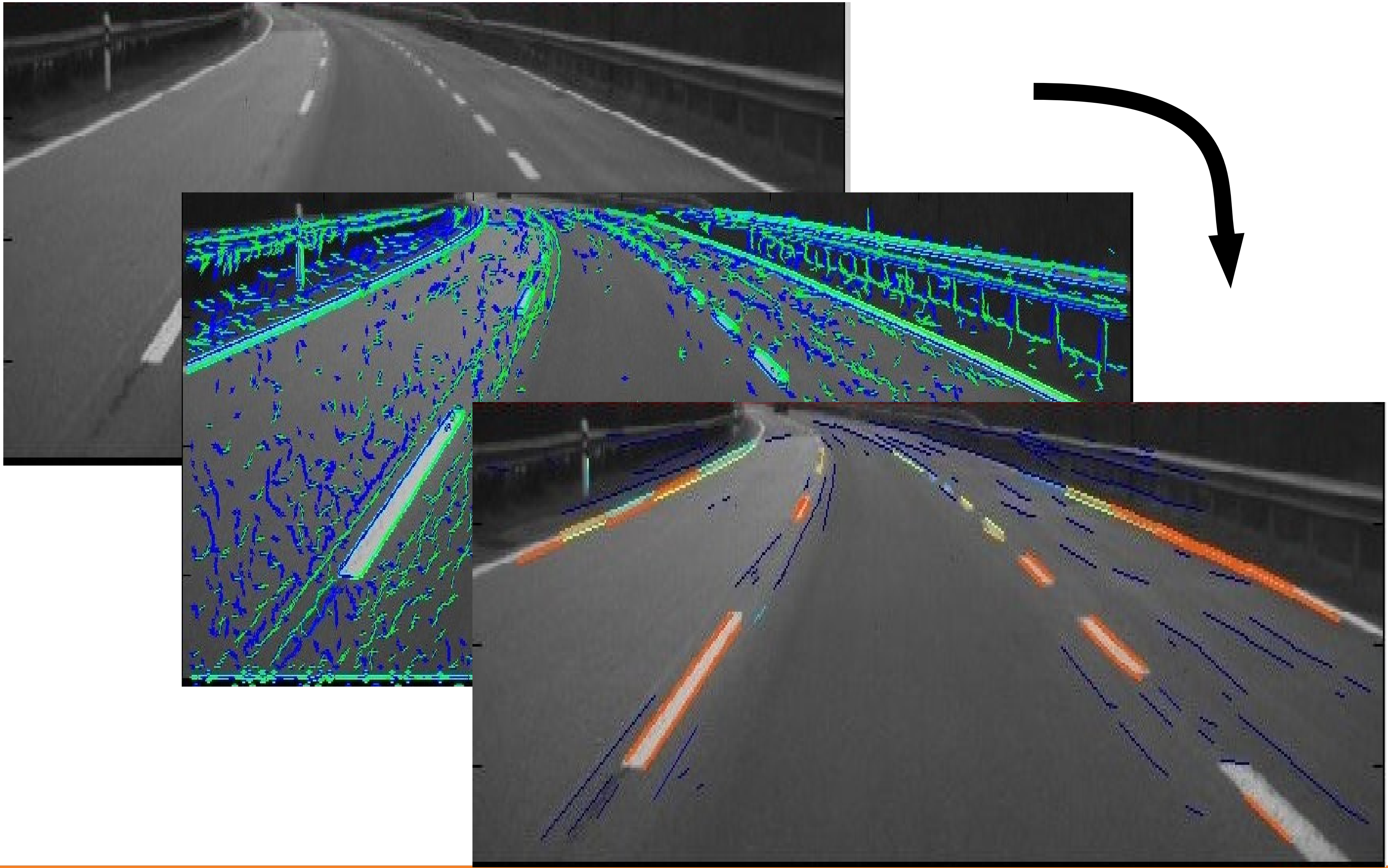
Given a loss function, how do you find a minimum?

- Direct Methods:
  - Least Squares fit (only for L2 norm)

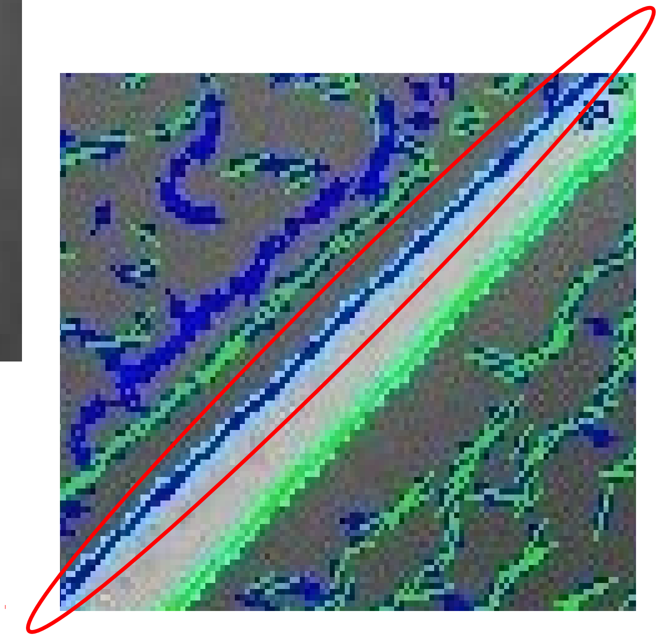
- Iterative Methods:
  - Start at some (random) initial location, and then
  - Gradient descent (1st order) (stochastic variants) [lot more on this later]
  - (Pseudo-) Newton methods (2nd order)
  - Iterated Re-weighted Least Squares
  - Iterated closest point (ICP)

- Search (Hypothesize and test)
  - Dense sampling (histogram, Hough transform)
  - RANSAC
  - Many other problem specific algorithms: Multi-grid, branch&bound, etc.

2D Example: Finding Straight Lines



2D Example: Finding Straight Lines

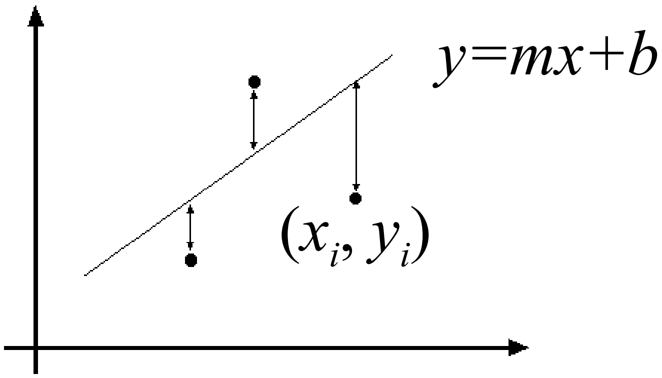


Assume you know these points belong to a line.
How do your fit a line to it?

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

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



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

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

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

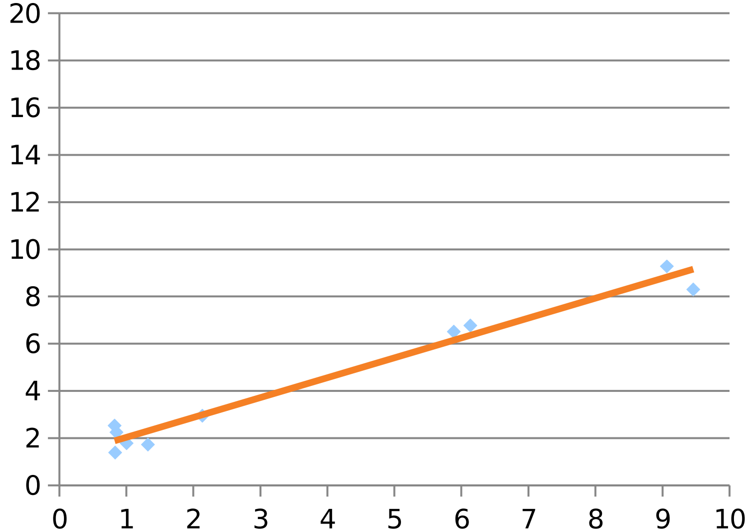
Matlab: `p = A \ y;`

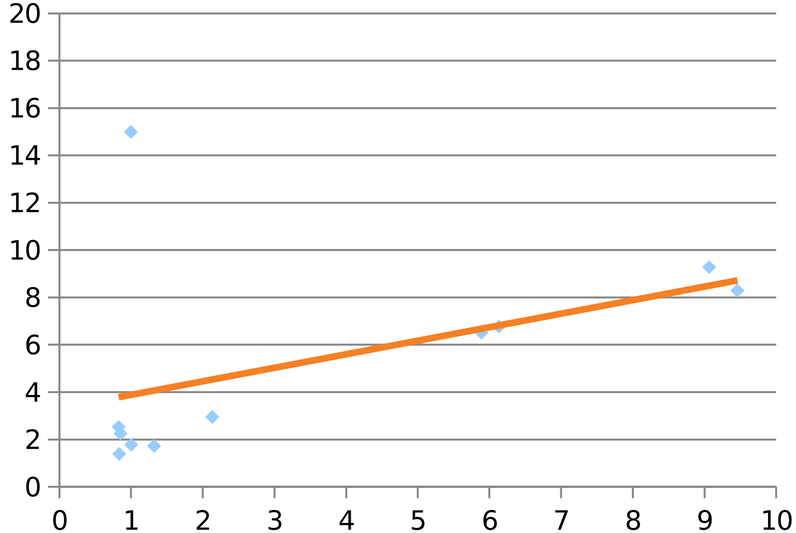
$$\mathbf{A}^T \mathbf{Ap} = \mathbf{A}^T \mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{y}$$

Modified from S. Lazebnik

Outliers

- Least squares assumes Gaussian errors
- Outliers: points with extremely low probability of occurrence (according to Gaussian statistics)
- Have strong influence on least squares





Reminder: Robust Estimation

- Goal: develop parameter estimation methods insensitive to *small* numbers of *large* errors
- General approach: try to give large deviations less weight
- M-estimators: minimize some loss function other than L2 [square of $y - f(x, a, b, \ldots)$]

Iteratively Reweighted Least Squares

- We can iteratively approximate L1
- Approximation: convert to weighted least squares

$$\sum_i \left| y_i - f(x_i, a, b, \ldots) \right|$$
$$= \sum_i \frac{1}{\left| y_i - f(x_i, a, b, \ldots) \right|} \left(y_i - f(x_i, a, b, \ldots) \right)^2$$
$$= \sum_i w_i \left(y_i - f(x_i, a, b, \ldots) \right)^2$$

with w_i based on previous iteration

Approximating other Losses

- Different options for weights to approximate other Loss functions
  - Avoid problems with infinities
  - Give even less weight to outliers

$$w_i = \frac{1}{\left|y_i - f(x_i, a, b, \ldots)\right|}$$

L_1

$$w_i = \frac{1}{\varepsilon + \left|y_i - f(x_i, a, b, \ldots)\right|}$$

"Fair"

$$w_i = \frac{1}{\varepsilon + \left(y_i - f(x_i, a, b, \ldots)\right)^2}$$

Cauchy / Lorentzian

$$w_i = e^{-k\left(y_i - f(x_i, a, b, \ldots)\right)^2}$$

Welsch

Summary: Least Squares

Ordinary

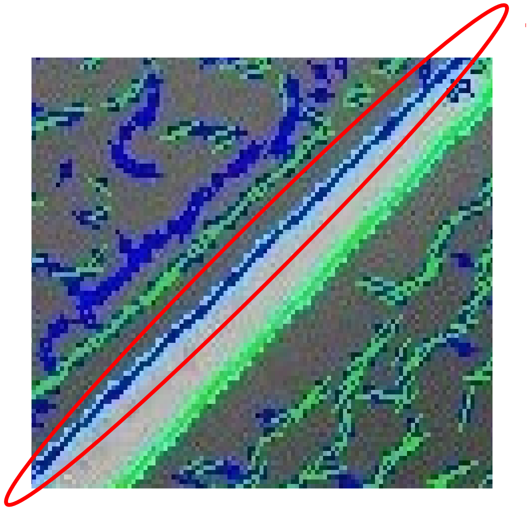
+ Clearly specified objective
+ Optimization is easy
+ Finds global optimum
- Loss is L2: may not be what you want to optimize
- Sensitive to outliers
- Hard to detect multiple objects, lines, etc.

Iterated Reweighted

+ Allows more robust objectives
+ Better sensitive to limited number of outliers
- Iterative, more costly to optimize
- Dependent on starting point: can fall into local minima
- Hard to detect multiple objects, lines, etc.

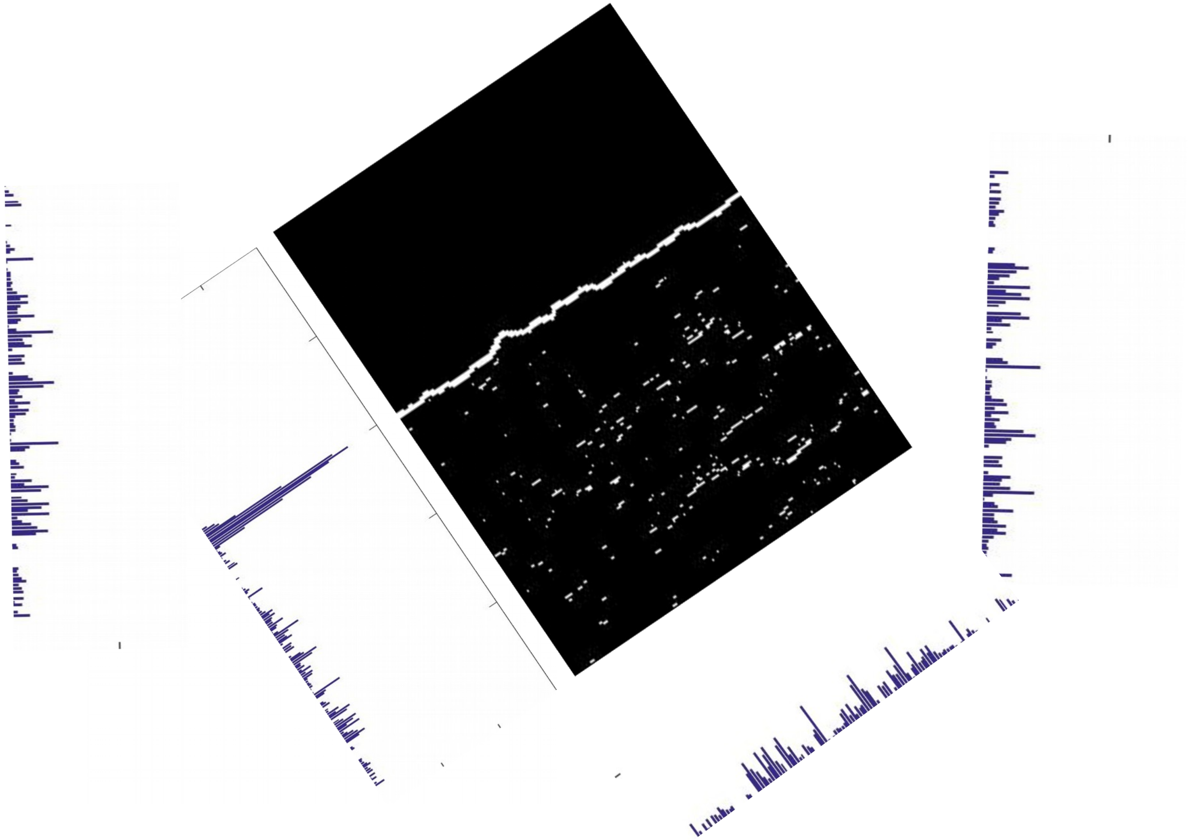
Neither still solves this problem:

how do we figure out that these points belong together?



2D Histogram

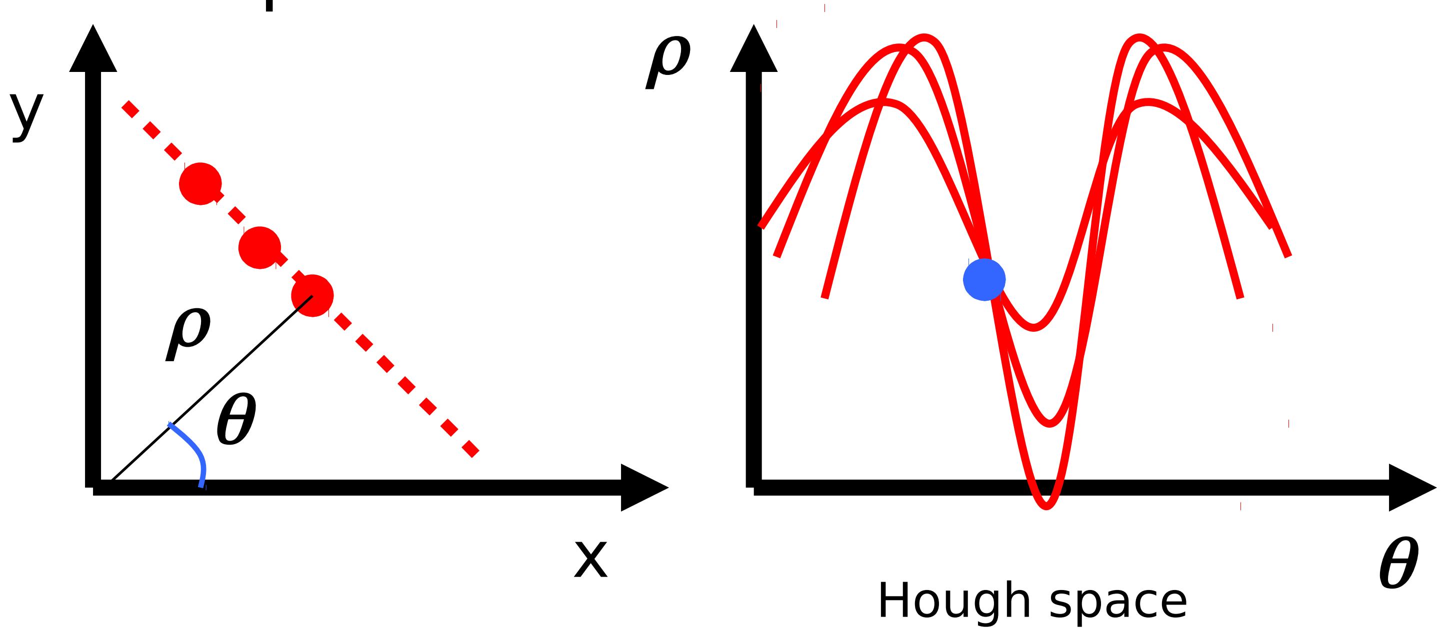
Recall what worked best in 1-D: Histogram!
Can we do the same thing, now for 2 DOFs? Rotate it...
Note: each edge point votes in each 1D histogram



Hough transform

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

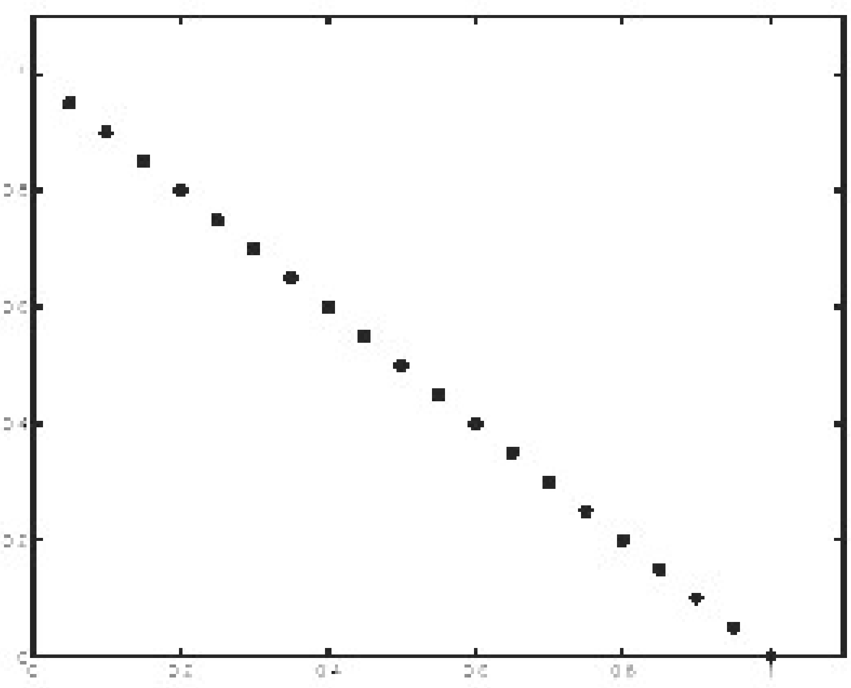
Assemble the 1D histograms into a 2D histogram: use a polar representation for the parameter space



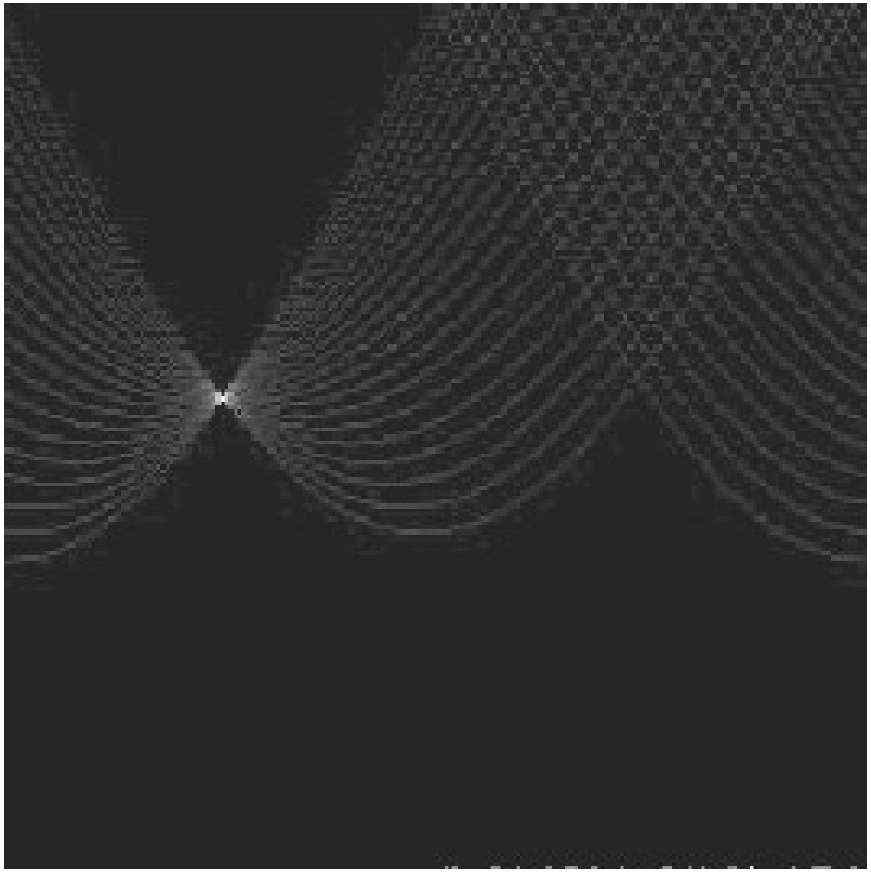
$$x \cos\theta + y \sin\theta = \rho$$

Slide from S. Savarese

Hough transform



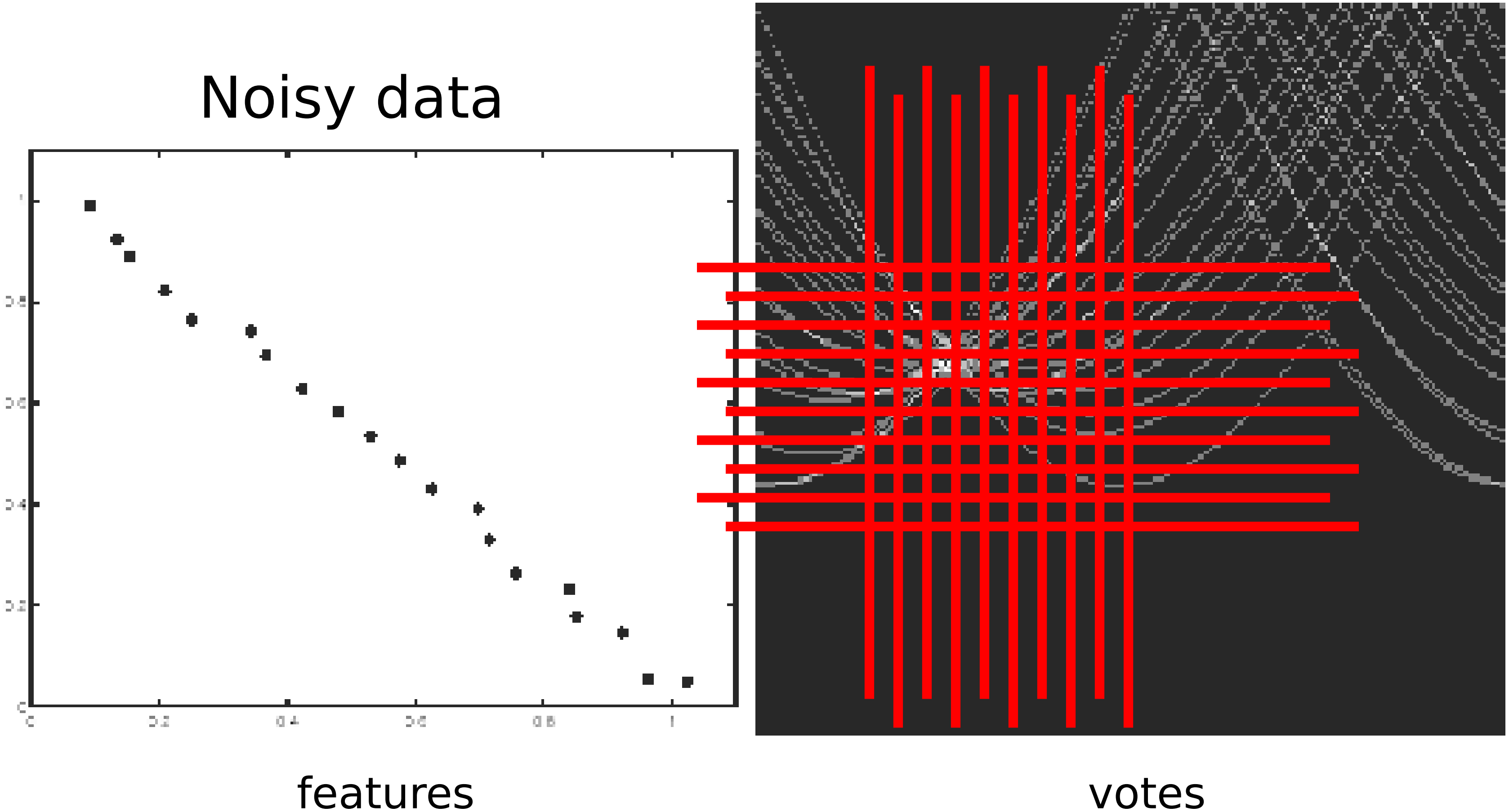
features



votes

Slide from S. Savarese

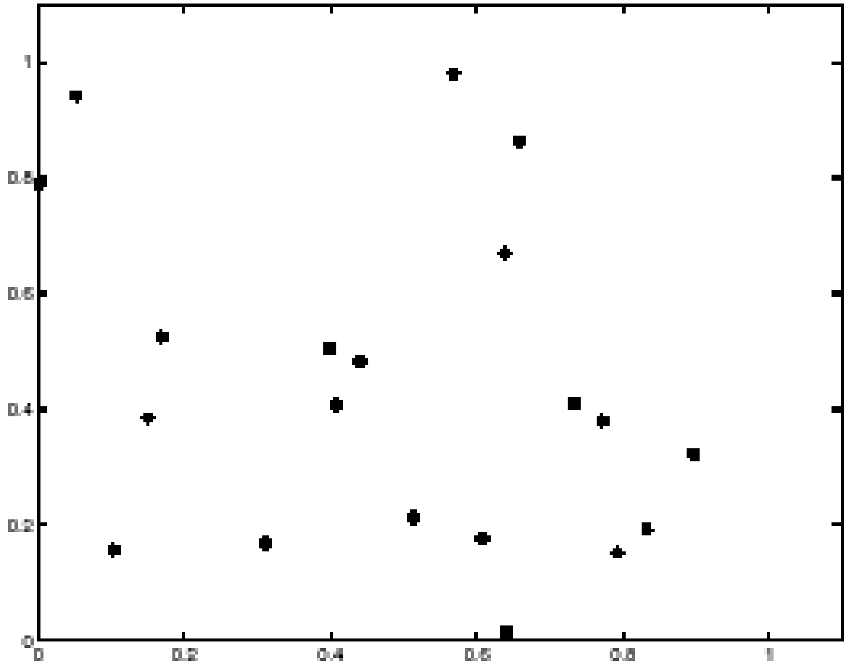
Hough transform - experiments



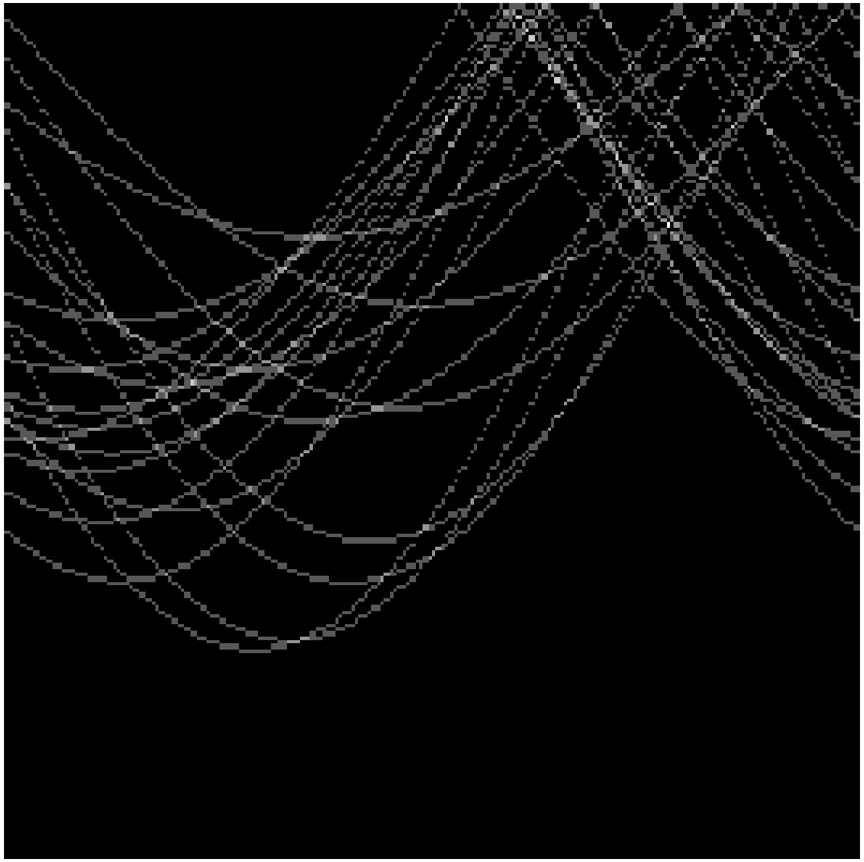
Need to adjust grid size or smooth

Slide from S. Savarese

Hough transform - experiments



features



votes

Issue: spurious peaks due to uniform noise

Slide from S. Savarese

1. Image -> Canny

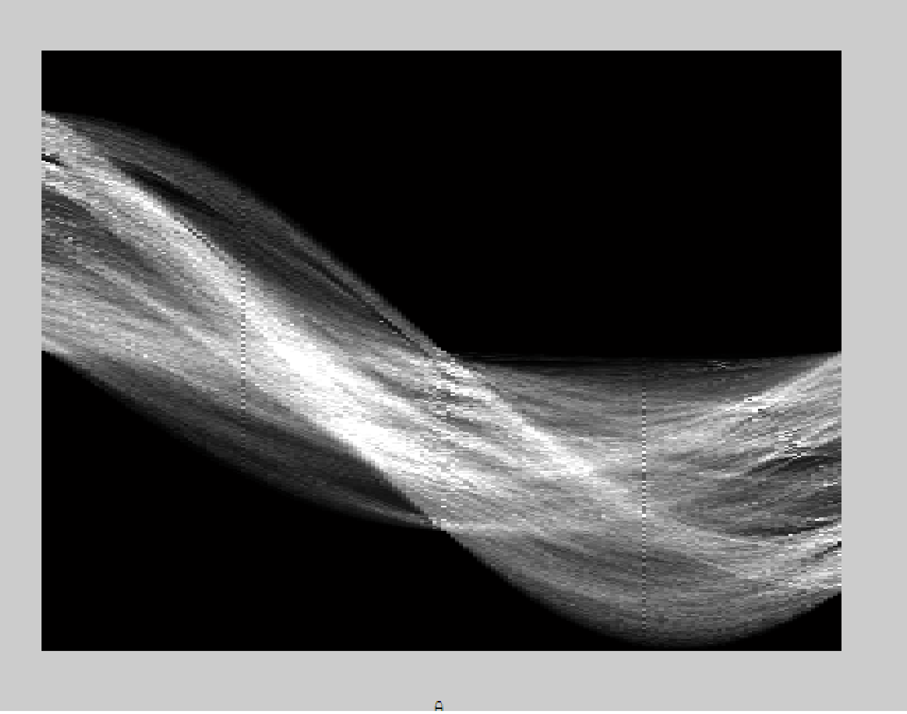
Slide from S. Savarese

2. Canny -> Hough votes

Slide from S. Savarese

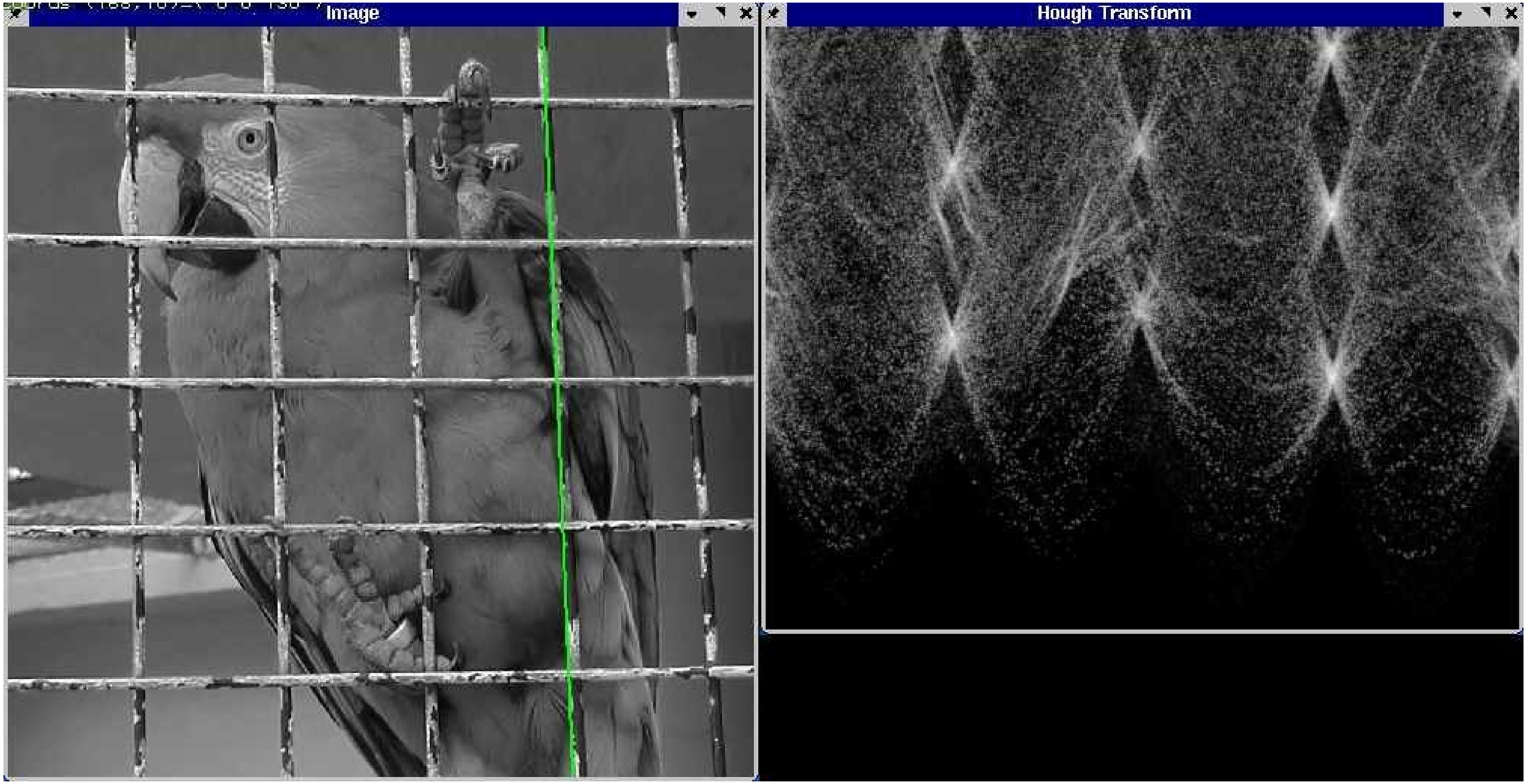
3. Hough votes -> Edges

Find peaks and post-process



 Slide from S. Savarese

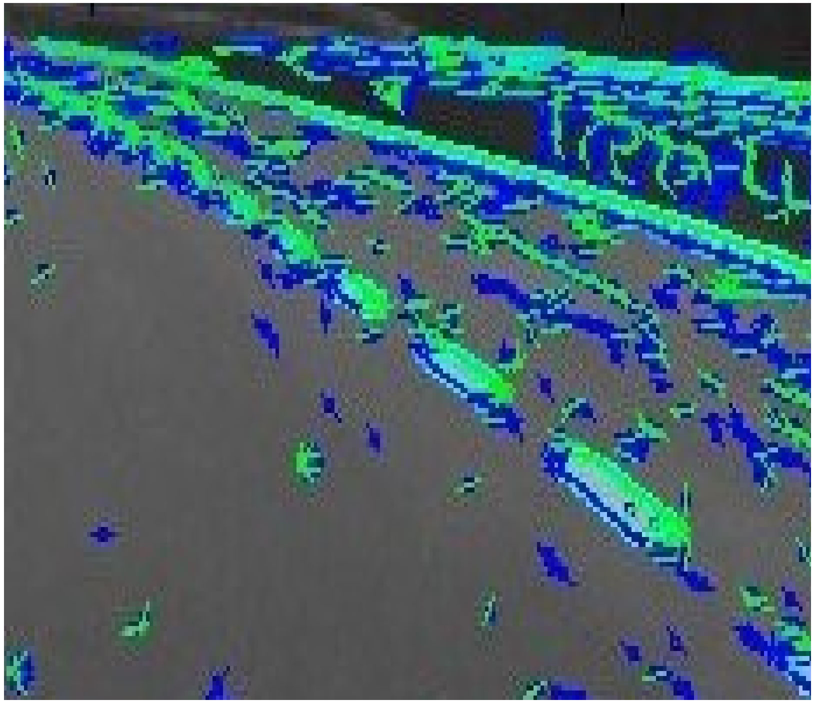
Another Hough transform example



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

Analyzing the Hough transform

- Think about parameterization & bin size
  - Is the bin size uniform everywhere?
- Make it more robust: smoothing?
- Cost: O(mn + mp)
  - Reduce using edge orientation?
- How to find multiple lines? Segments?
- Can you find a circle with Hough?
- What is the effect as #Dims increases?

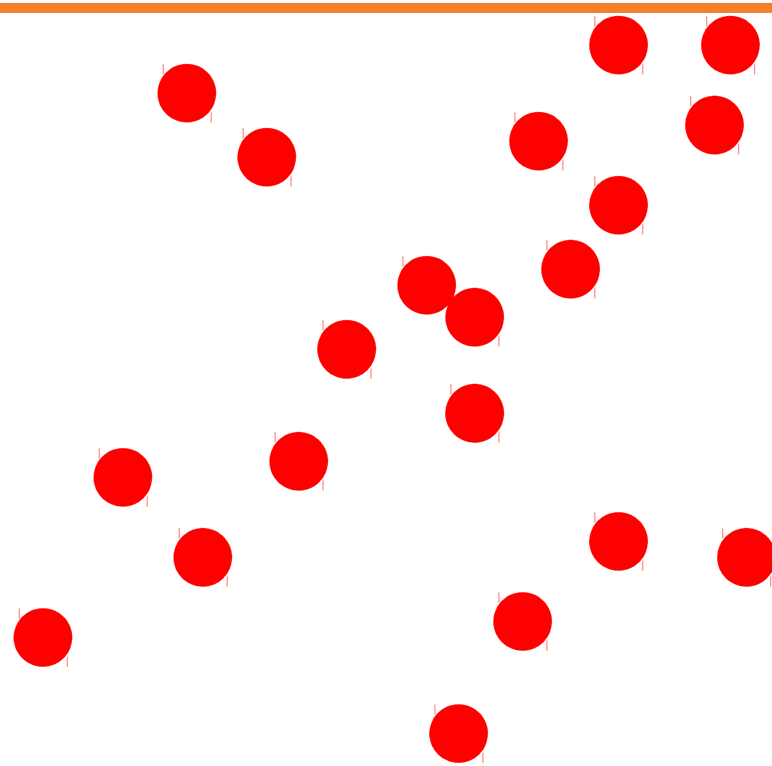


- Hypothesize and test: Do we need to sample in such a dense grid, or is there a more efficient strategy?

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



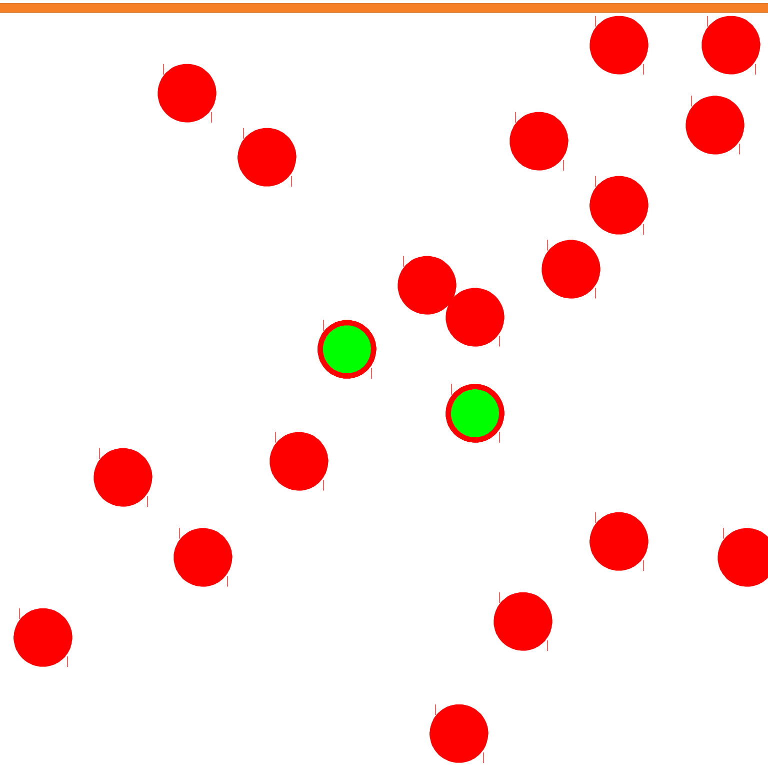
Algorithm:

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:

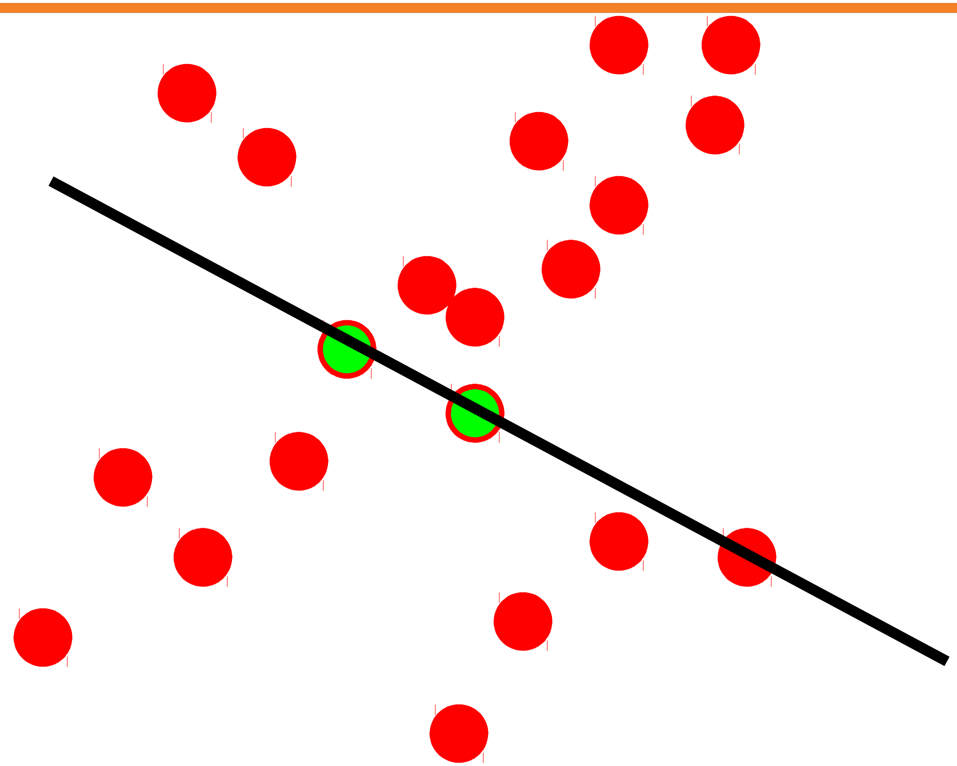
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

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

Illustration by Savarese

RANSAC

Line fitting example



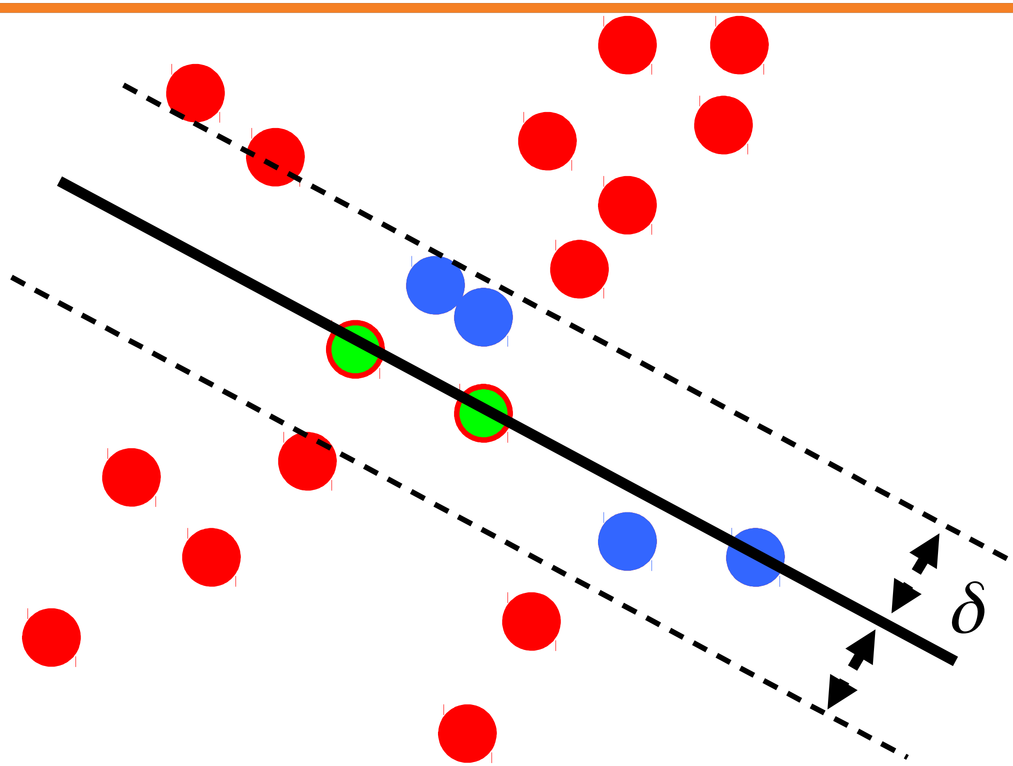
Algorithm:

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



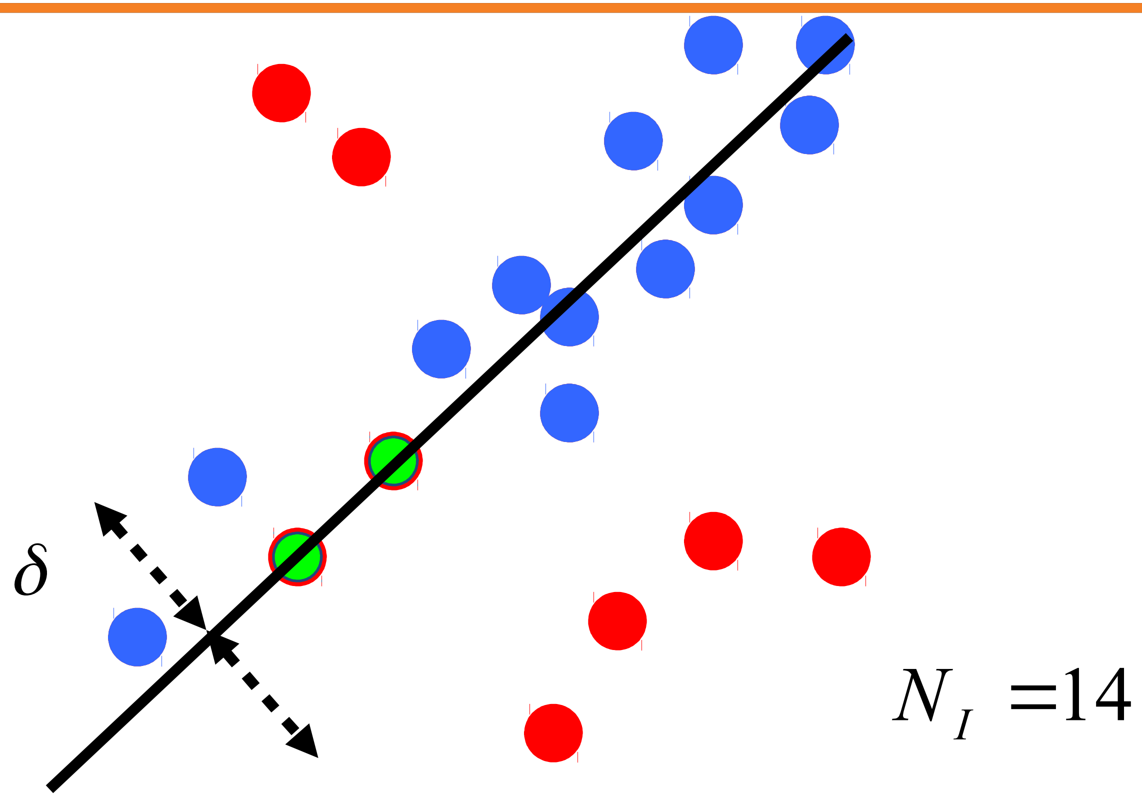
$$N_I = 6$$

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 more degrees of freedom (DOFs: 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 great multiple fits

Common applications [next time]

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