

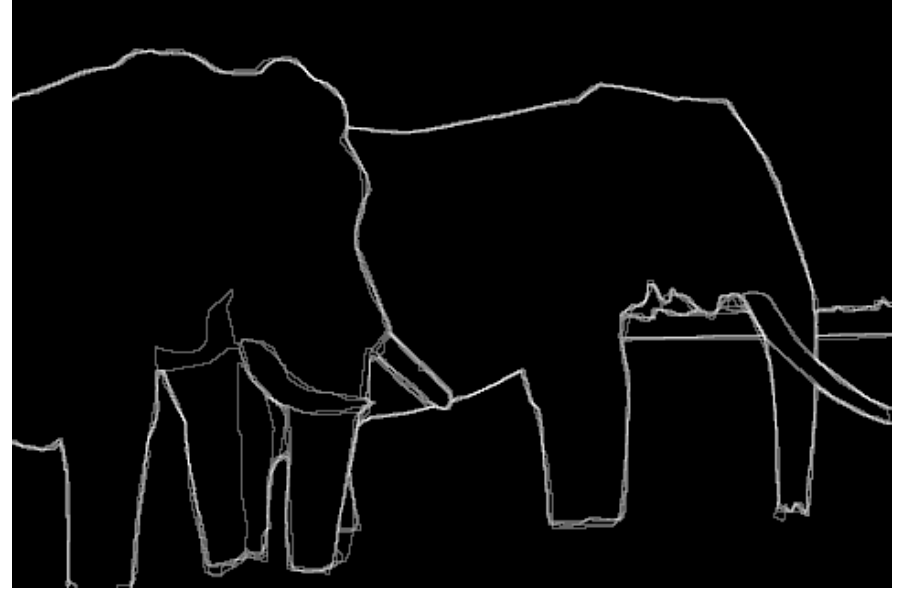
Hough Transformation for Fitting

Jianping Fan
Dept of Computer Science
UNC-Charlotte

Course Website:

<http://webpages.uncc.edu/jfan/itcs5152.html>

Boundaries of Objects



Marked by many users

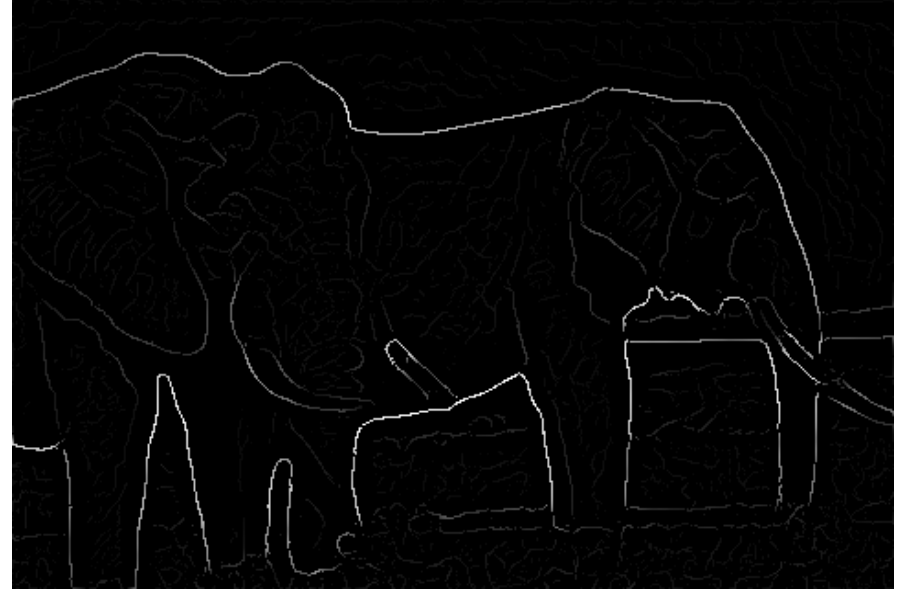
Boundaries of Objects from Edges



Brightness Gradient (Edge detection)

- **Missing edge continuity, many spurious edges**

Boundaries of Objects from Edges



Multi-scale Brightness Gradient

- But, low strength edges may be very important

Missing point recovery by edge tracking

Boundaries of Objects from Edges



Image



Machine Edge Detection



Human Boundary Marking

Boundaries in Medical Imaging

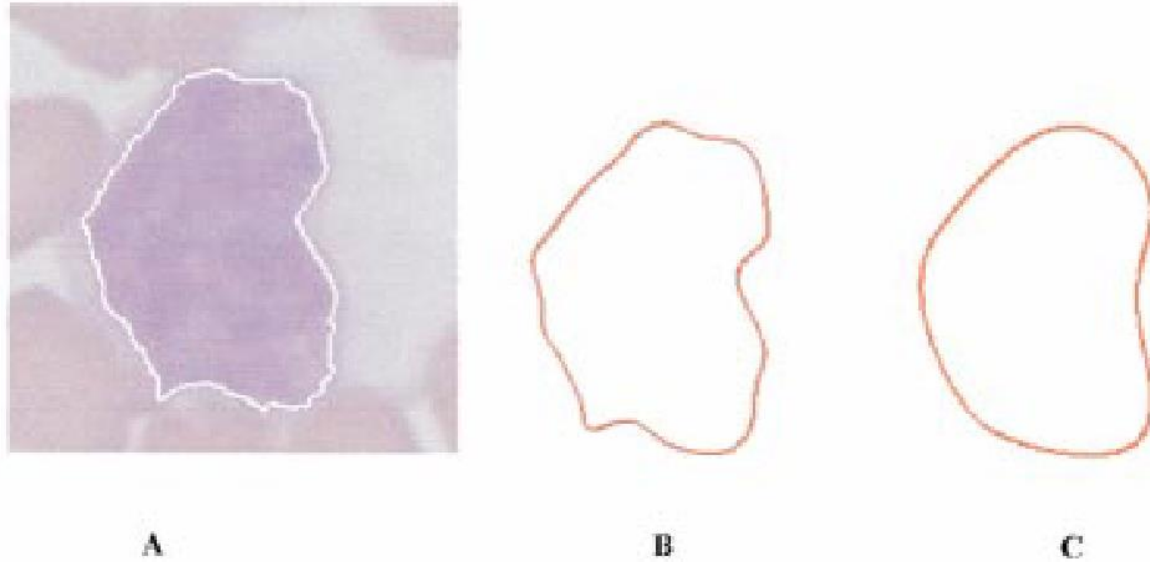
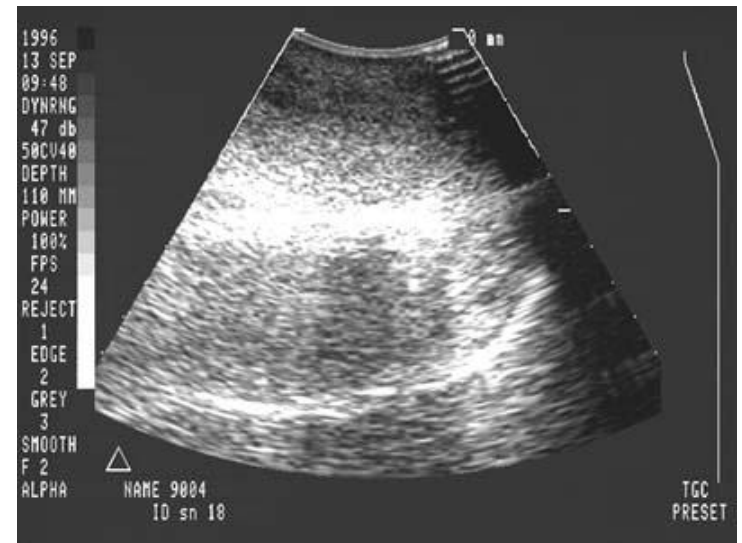


Fig. 2. Representation of a closed contour by elliptic Fourier descriptors. (a) Input. (b) Series truncated at 16 harmonics. (c) Series truncated to four harmonics.

Detection of cancerous regions.

Boundaries in Ultrasound Images



Hard to detect in the presence of large amount of speckle noise

Boundaries of Objects

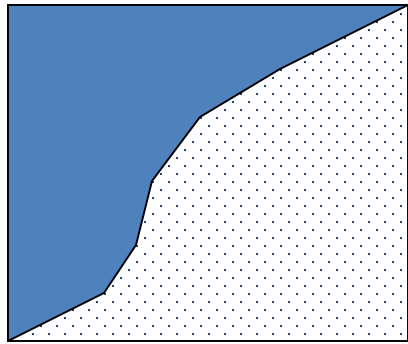


Sometimes hard even for humans!

Topics

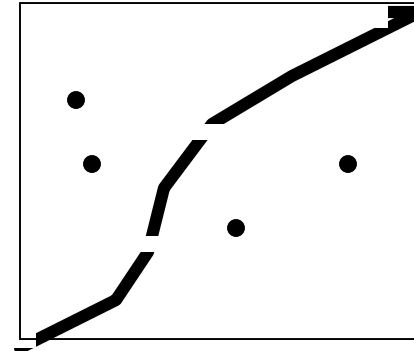
- **Preprocessing Edge Images**
- **Edge Tracking Methods**
- **Fitting Lines and Curves to Edges**
- **The Hough Transform**

Preprocessing Edge Images



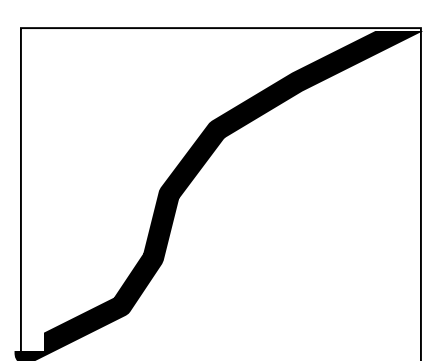
Image

Edge detection
and Thresholding

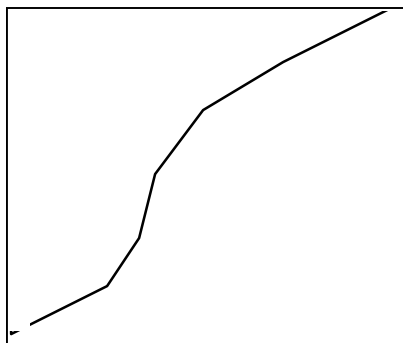
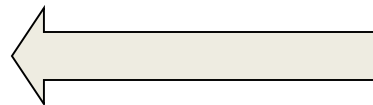


Noisy edge image
Incomplete boundaries

Shrink and Expand



Thinning



Edge Tracking Methods

Adjusting a priori Boundaries:

Given: Approximate Location of Boundary

Task: Find Accurate Location of Boundary

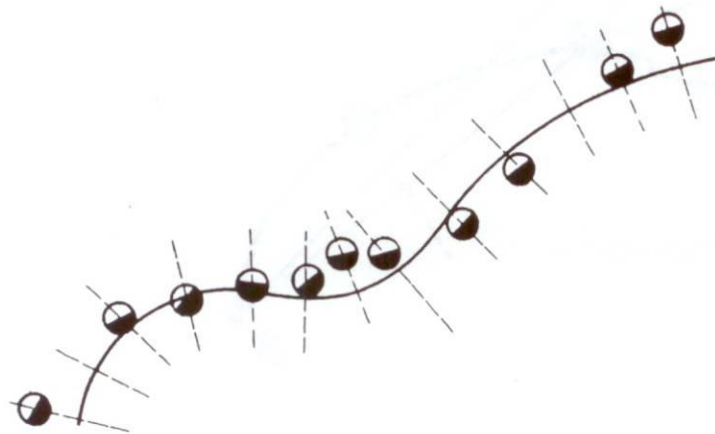


Fig. 4.2 Search orientations from an approximate boundary location.

- Search for **STRONG EDGES** along normals to approximate boundary.
- Fit curve (eg., polynomials) to strong edges.

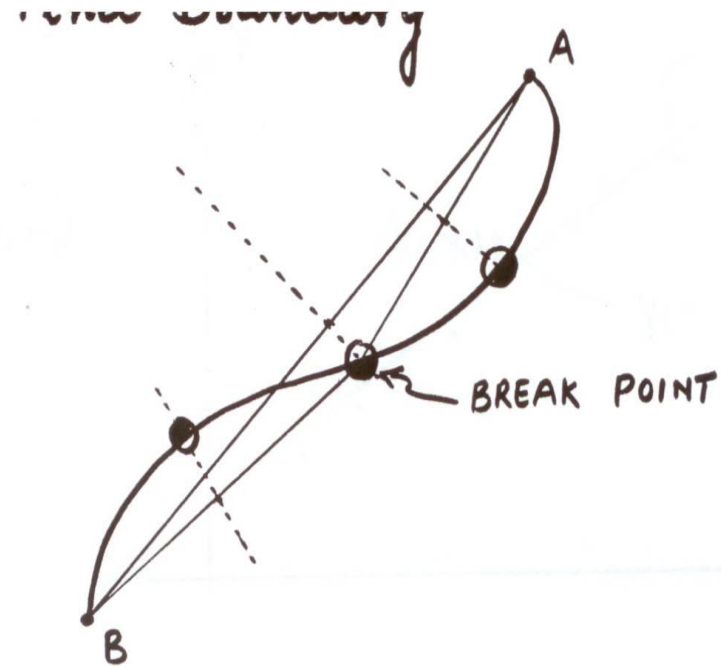
Edge Tracking Methods

Divide and Conquer:

Given: Boundary lies between points A and B

Task: Find Boundary

- Connect A and B with Line
- Find strongest edge along line bisector
- Use edge point as break point
- Repeat



Line Fitting

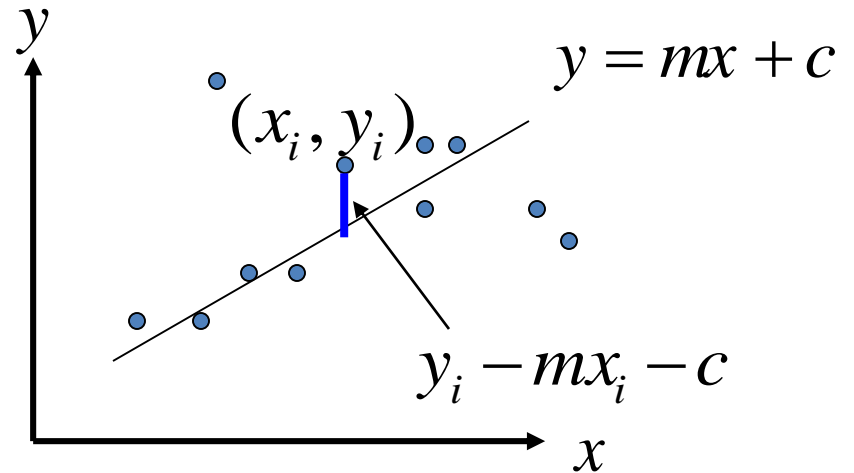
Fitting Lines to Edges (Least Squares)

Given: Many (x_i, y_i) pairs

Find: Parameters (m, c)

Minimize: Average square distance:

$$E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$$



Using:

$$\frac{\partial E}{\partial m} = 0 \quad \& \quad \frac{\partial E}{\partial c} = 0$$

Note:

$$\bar{y} = \frac{\sum_i y_i}{N} \quad \bar{x} = \frac{\sum_i x_i}{N}$$

$$c = \bar{y} - m \bar{x}$$
$$m = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

Voting schemes

- Let each feature vote for all the models that are compatible with it
- Hopefully the noise features will not vote consistently for any single model
- Missing data doesn't matter as long as there are enough features remaining to agree on a good model

Hough transform

- An early type of voting scheme
- General outline:
 - **Discretize parameter space into bins**
 - For each feature point in the image, put a vote in every bin in the parameter space that could have generated this point
 - **Find parameter bins that have the most votes**

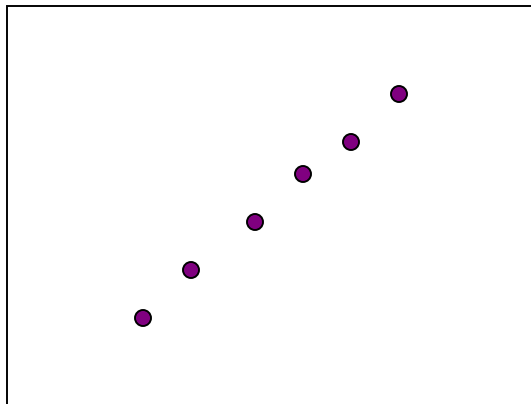
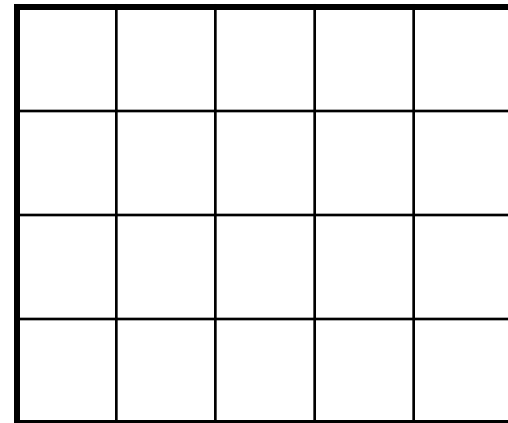
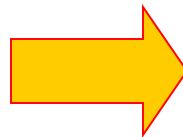


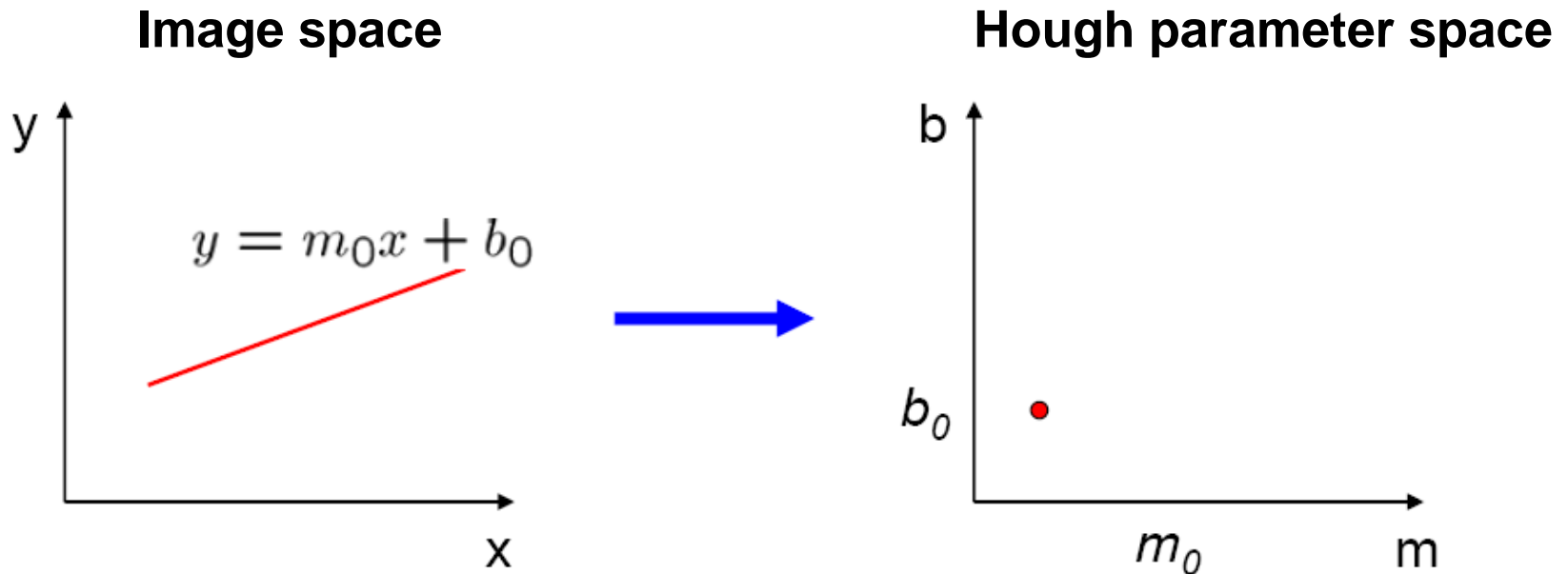
Image space



Hough parameter space

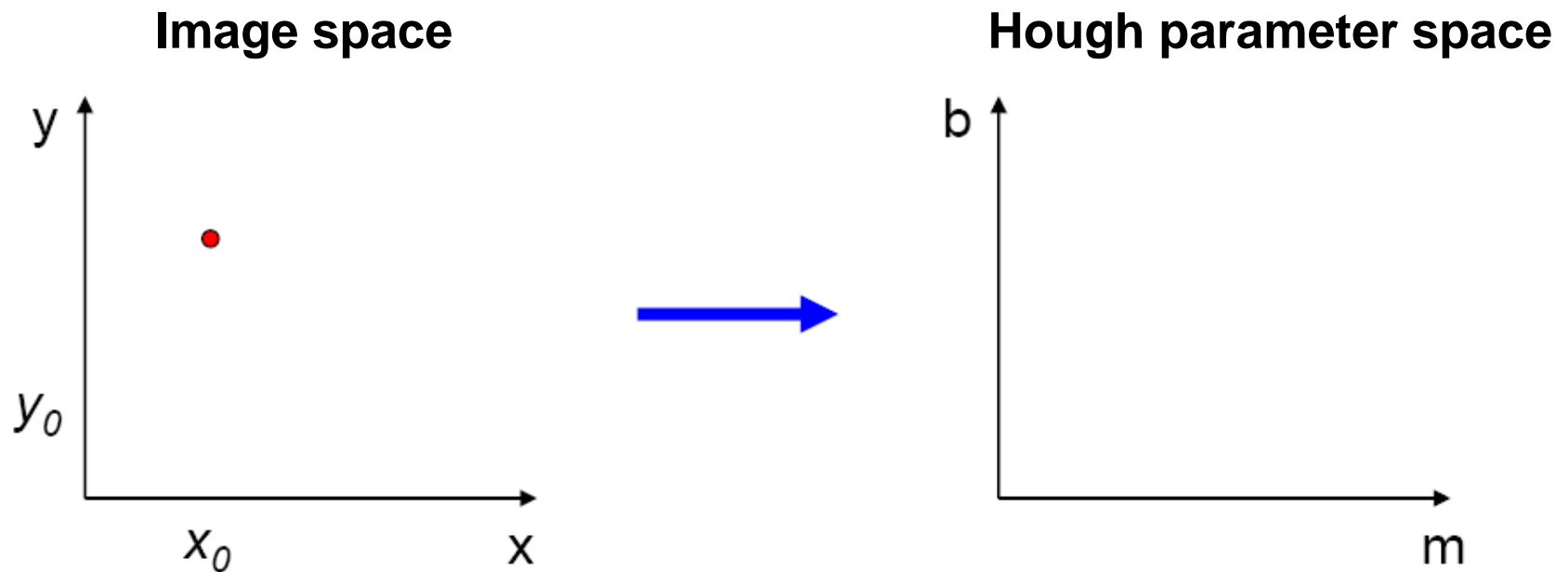
Parameter space representation

- A line in the image corresponds to a point in Hough space



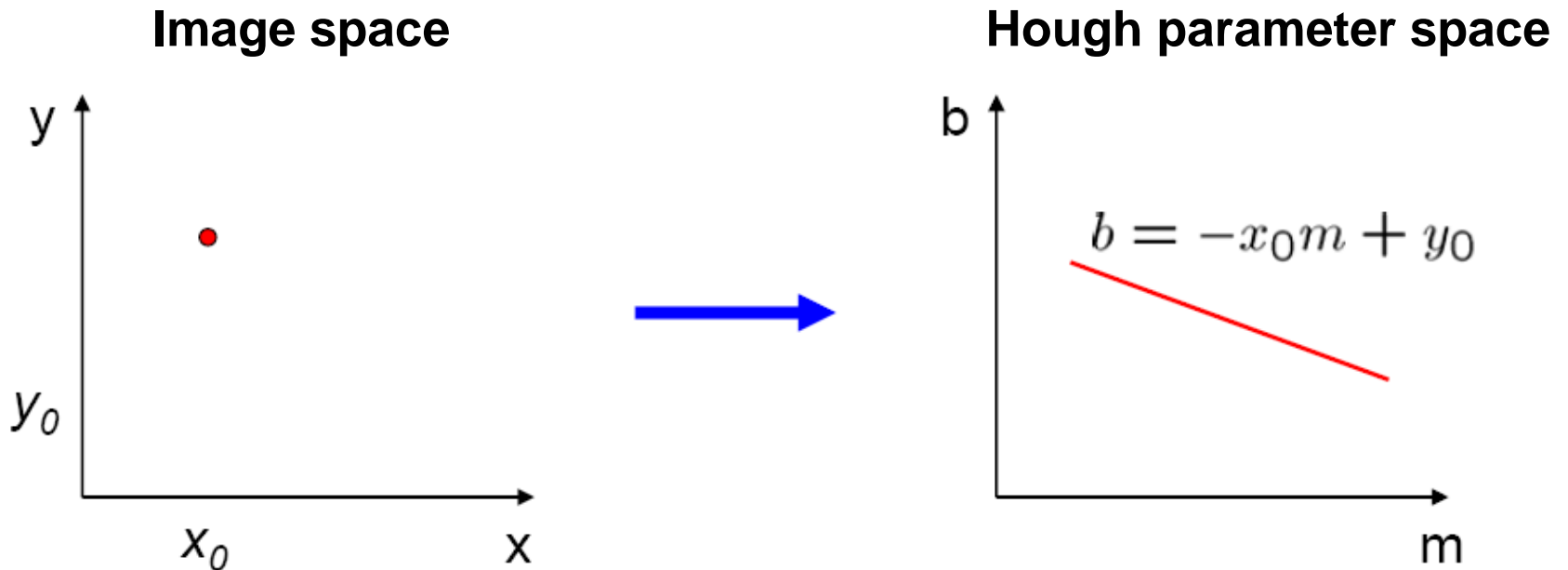
Parameter space representation

- What does a point (x_0, y_0) in the image space map to in the Hough space?



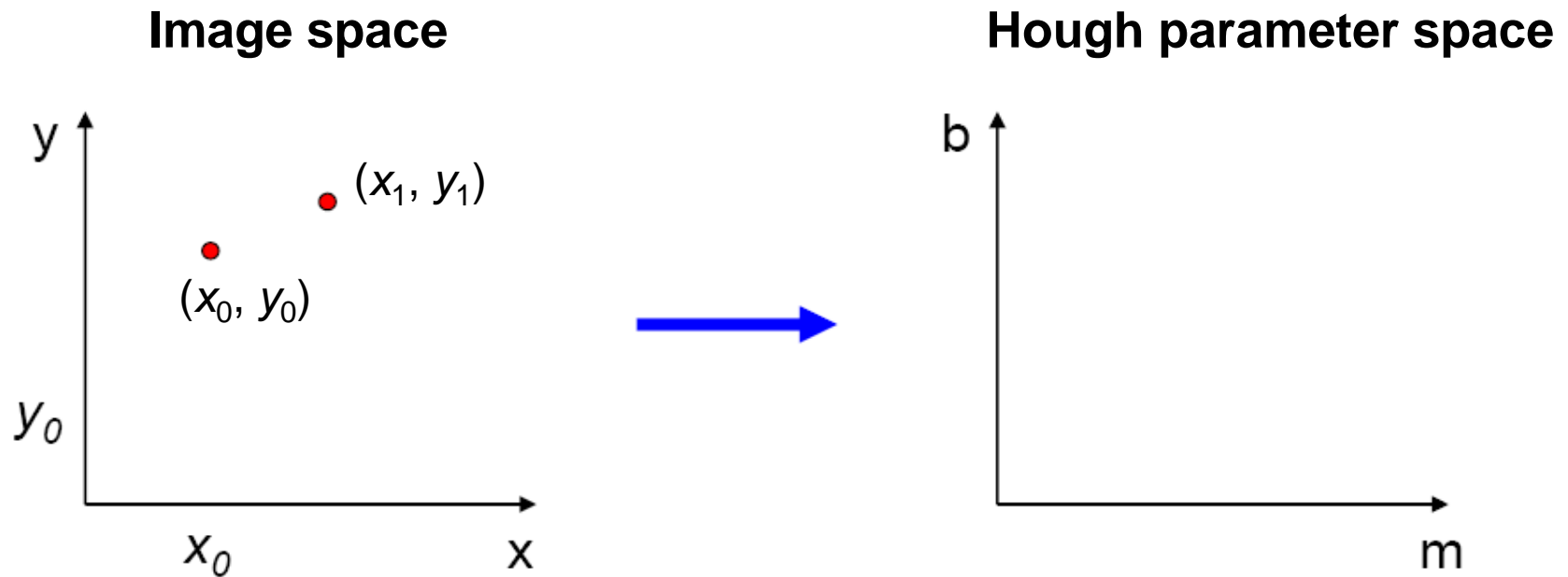
Parameter space representation

- What does a point (x_0, y_0) in the image space map to in the Hough space?
 - Answer: the solutions of $b = -x_0m + y_0$
 - This is a line in Hough space



Parameter space representation

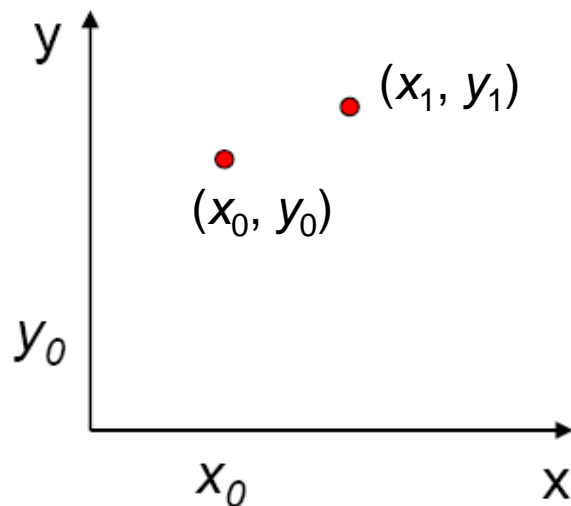
- Where is the line that contains both (x_0, y_0) and (x_1, y_1) ?



Parameter space representation

- Where is the line that contains both (x_0, y_0) and (x_1, y_1) ?
 - It is the intersection of the lines $b = -x_0m + y_0$ and $b = -x_1m + y_1$

Image space



Hough parameter space

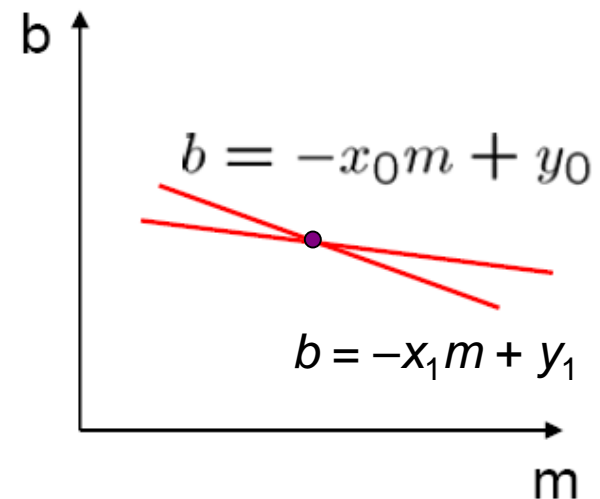


Image and Parameter Spaces

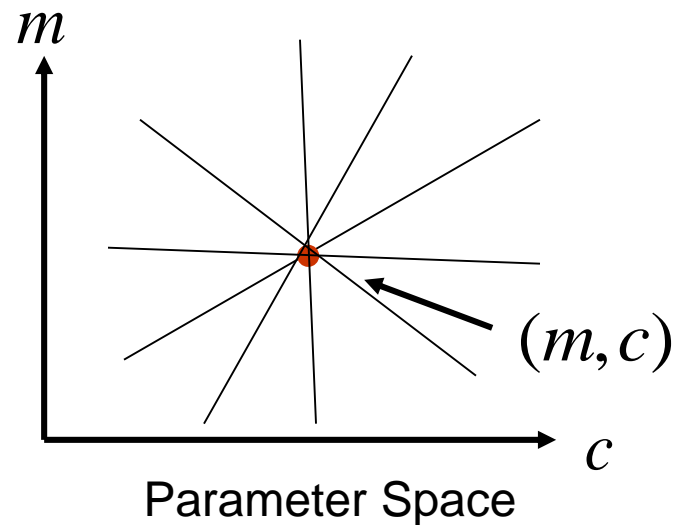
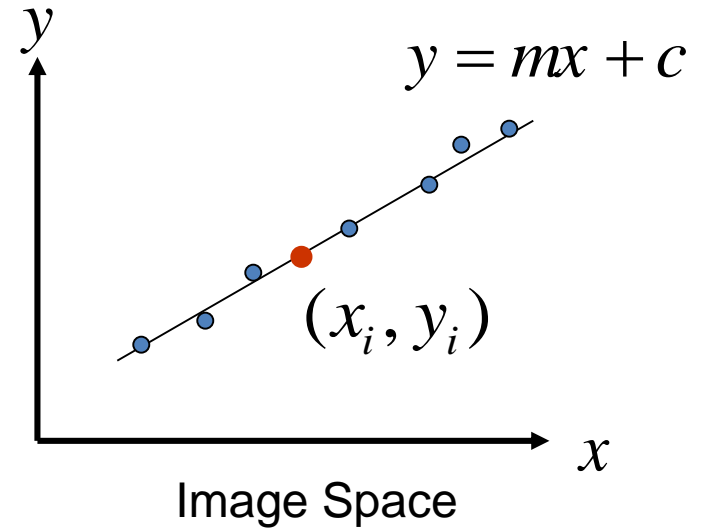
Equation of Line: $y = mx + c$

Find: (m, c)

Consider point: (x_i, y_i)

$$y_i = mx_i + c \quad \text{or} \quad c = -x_i m + y_i$$

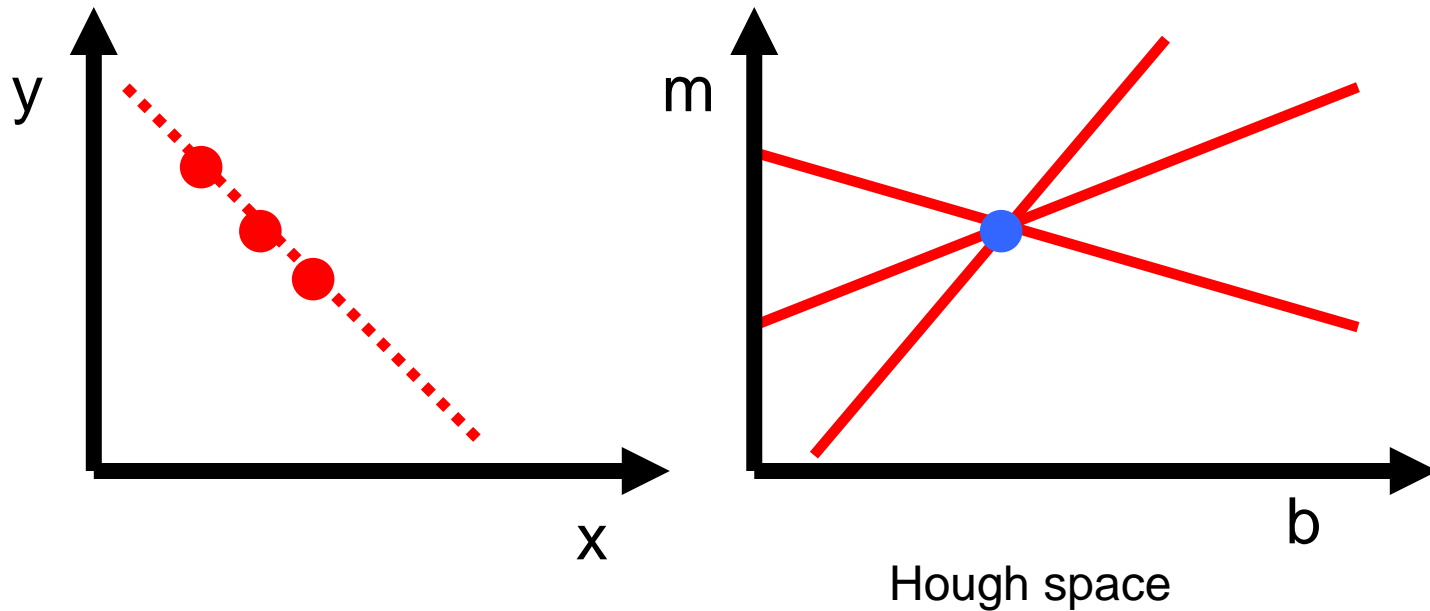
Parameter space also called Hough Space



Hough transform

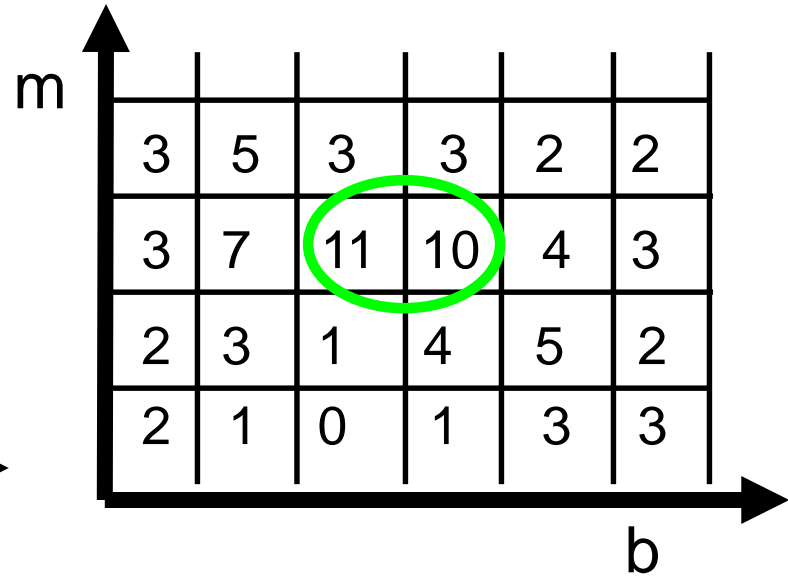
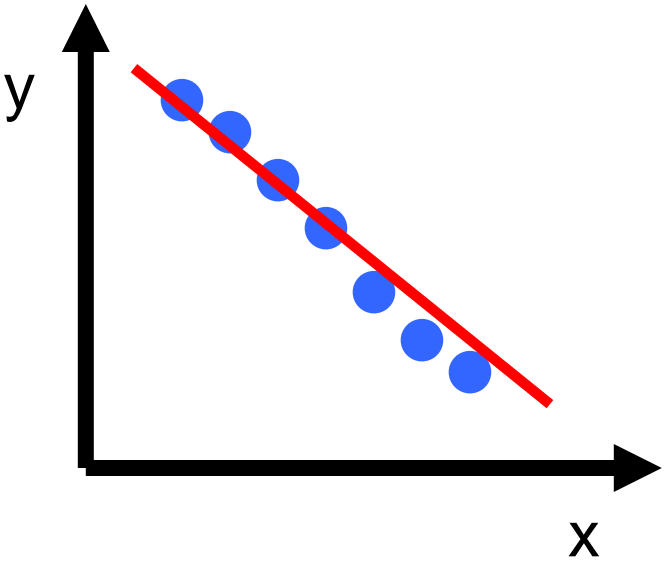
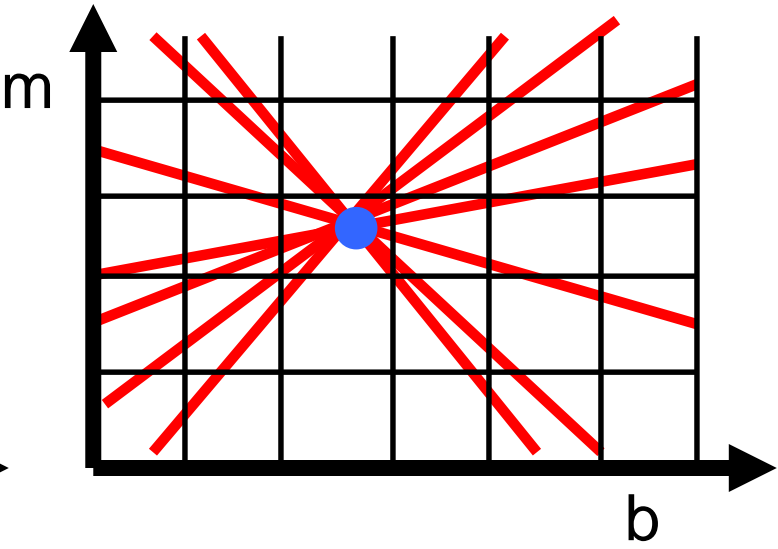
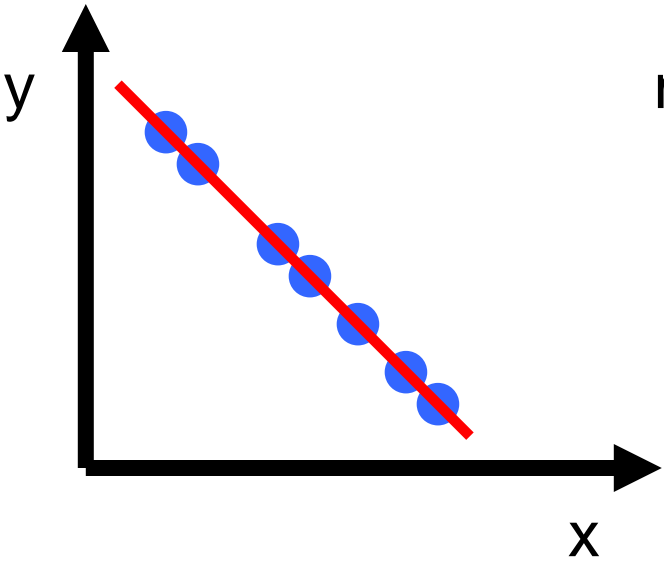
P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures*, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

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



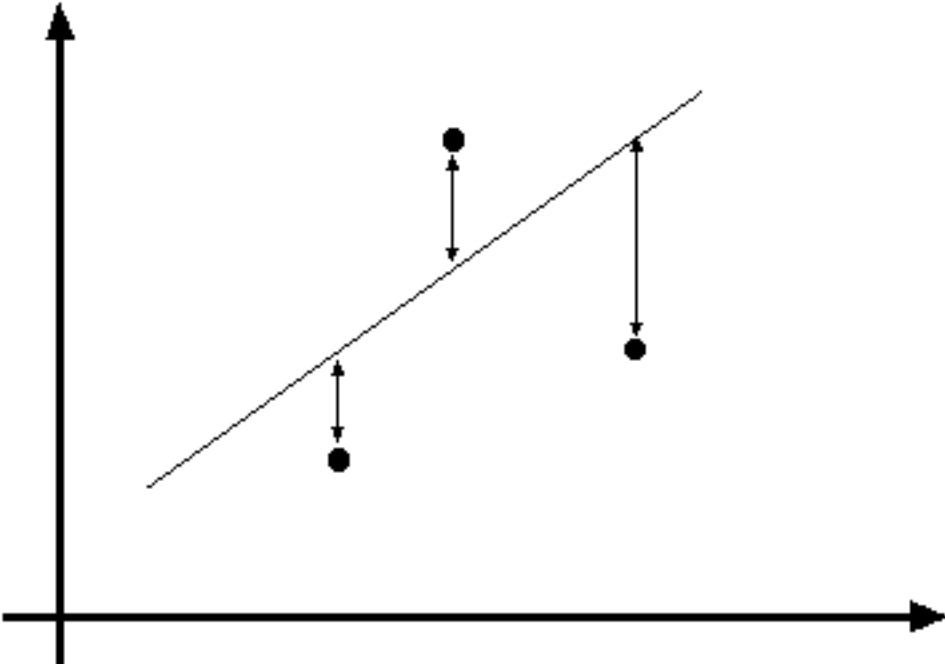
$$y = m x + b$$

Hough transform

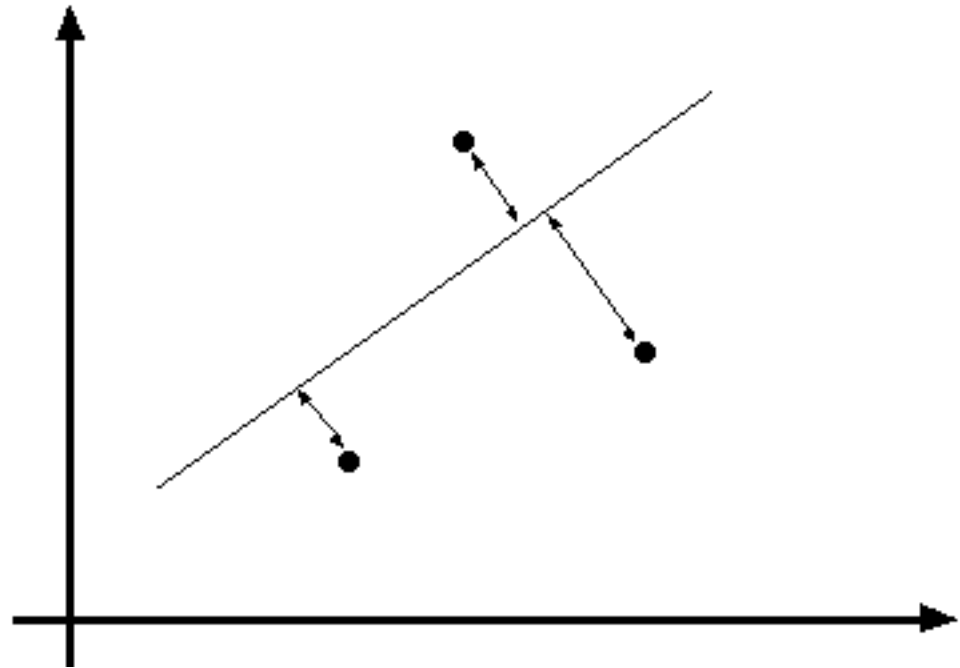


Parameter space representation

- Problems with the (m, b) space:
 - Unbounded parameter domain
 - Vertical lines require infinite m



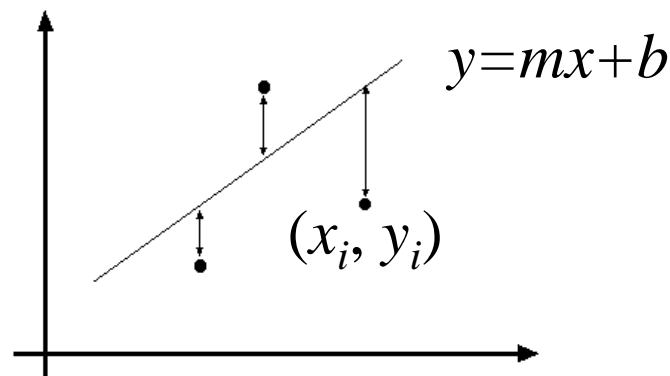
Line fitting can be max. likelihood - but choice of model is important



Least squares line fitting

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

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



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

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

Matlab: $\mathbf{p} = \mathbf{A} \setminus \mathbf{y};$

$$\frac{dE}{d\mathbf{p}} = 2\mathbf{A}^T \mathbf{A}\mathbf{p} - 2\mathbf{A}^T \mathbf{y} = 0$$

Python: $\mathbf{p} =$
`numpy.linalg.lstsq(A, y)`

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

Least squares (global) optimization

Good

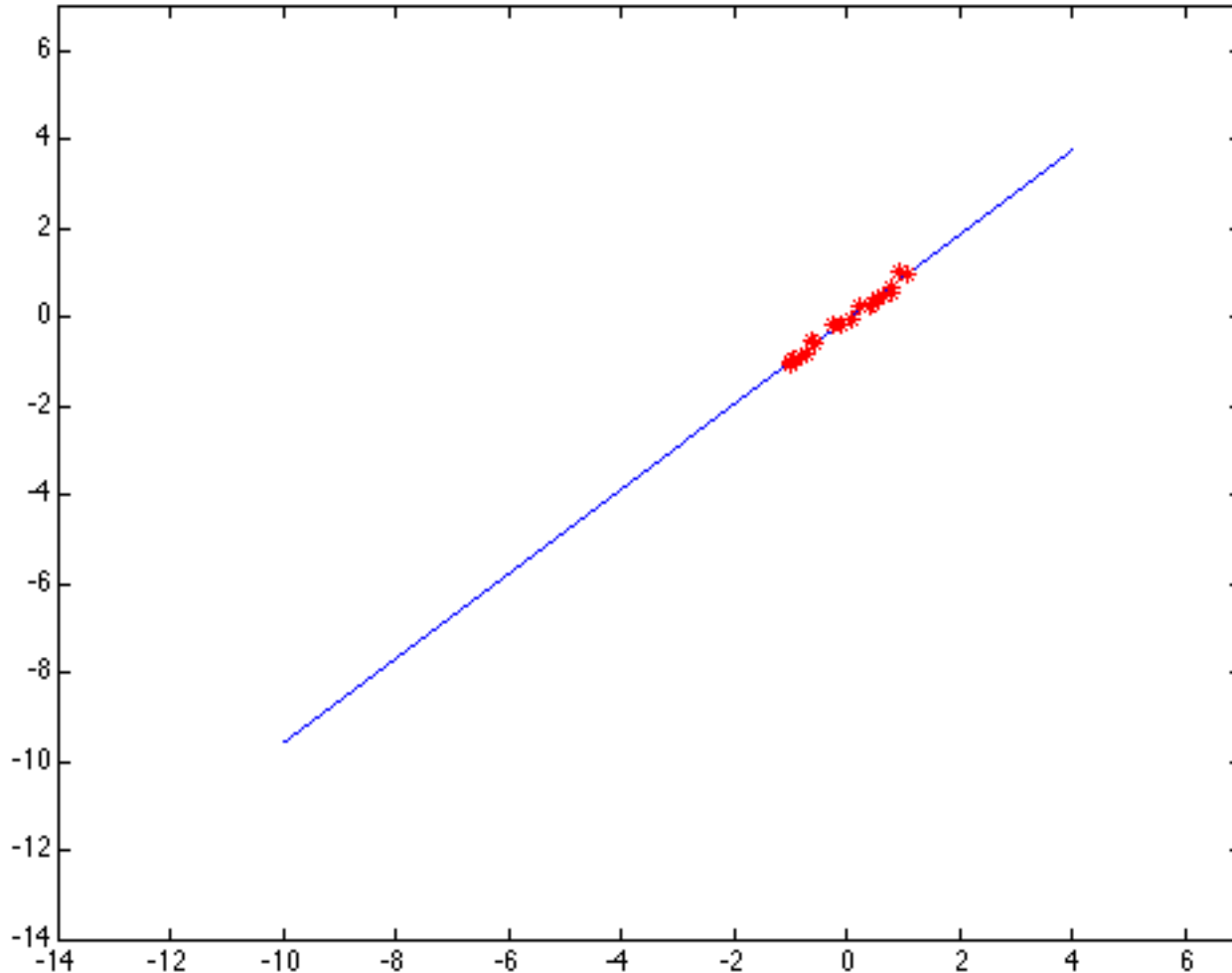
- Clearly specified objective
- Optimization is easy

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.

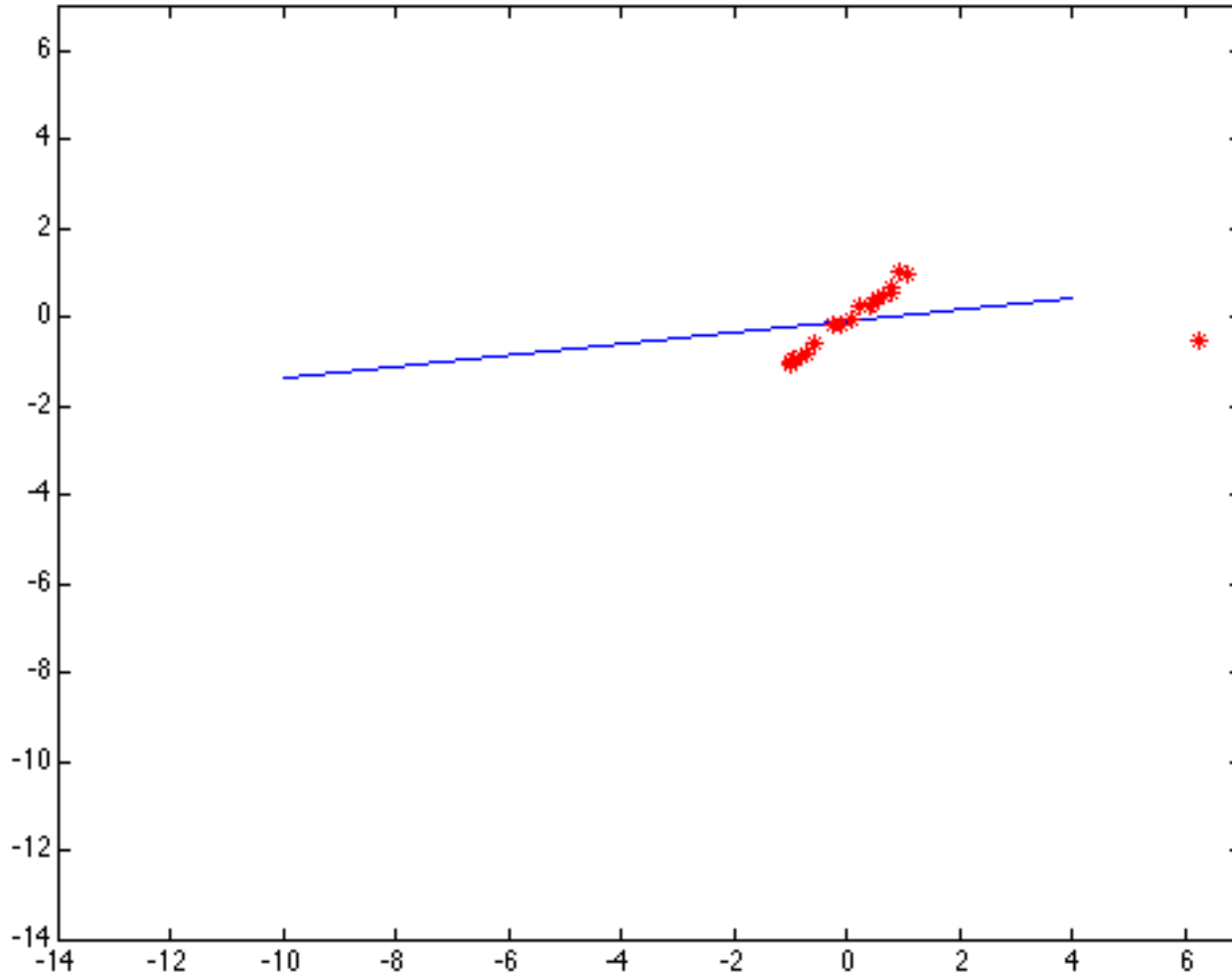
Least squares: Robustness to noise

- Least squares fit to the red points:



Least squares: Robustness to noise

- Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

Robust least squares (to deal with outliers)

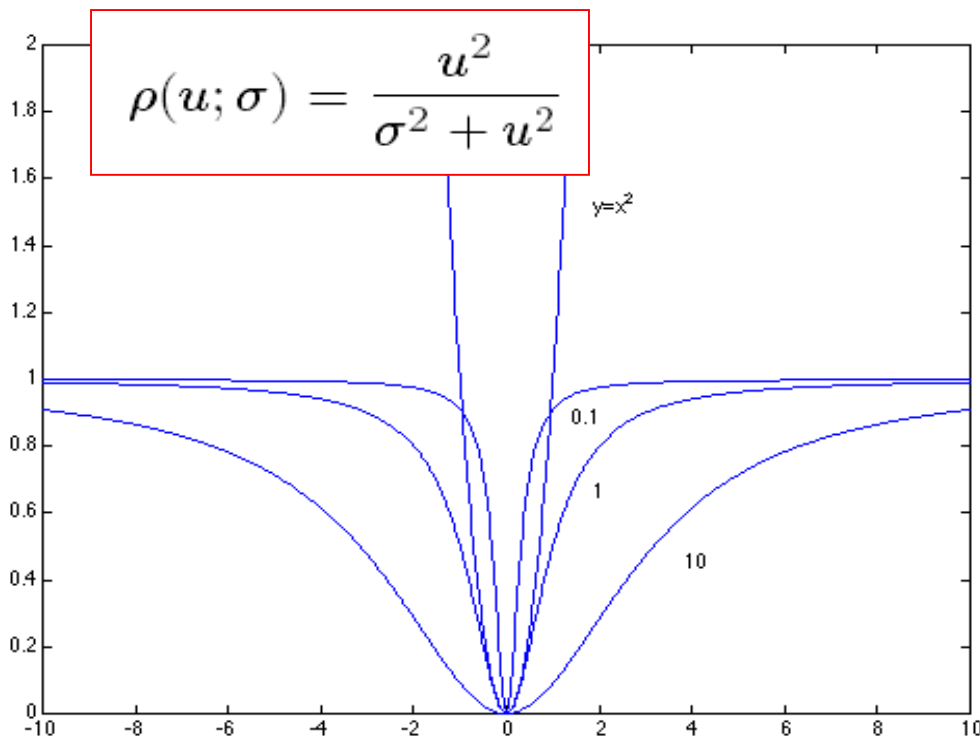
General approach:

minimize

$$\sum_i \rho(u_i(x_i, \theta); \sigma) \quad u^2 = \sum_{i=1}^n (y_i - mx_i - b)^2$$

$u_i(x_i, \theta)$ – residual of i^{th} point w.r.t. model parameters ϑ

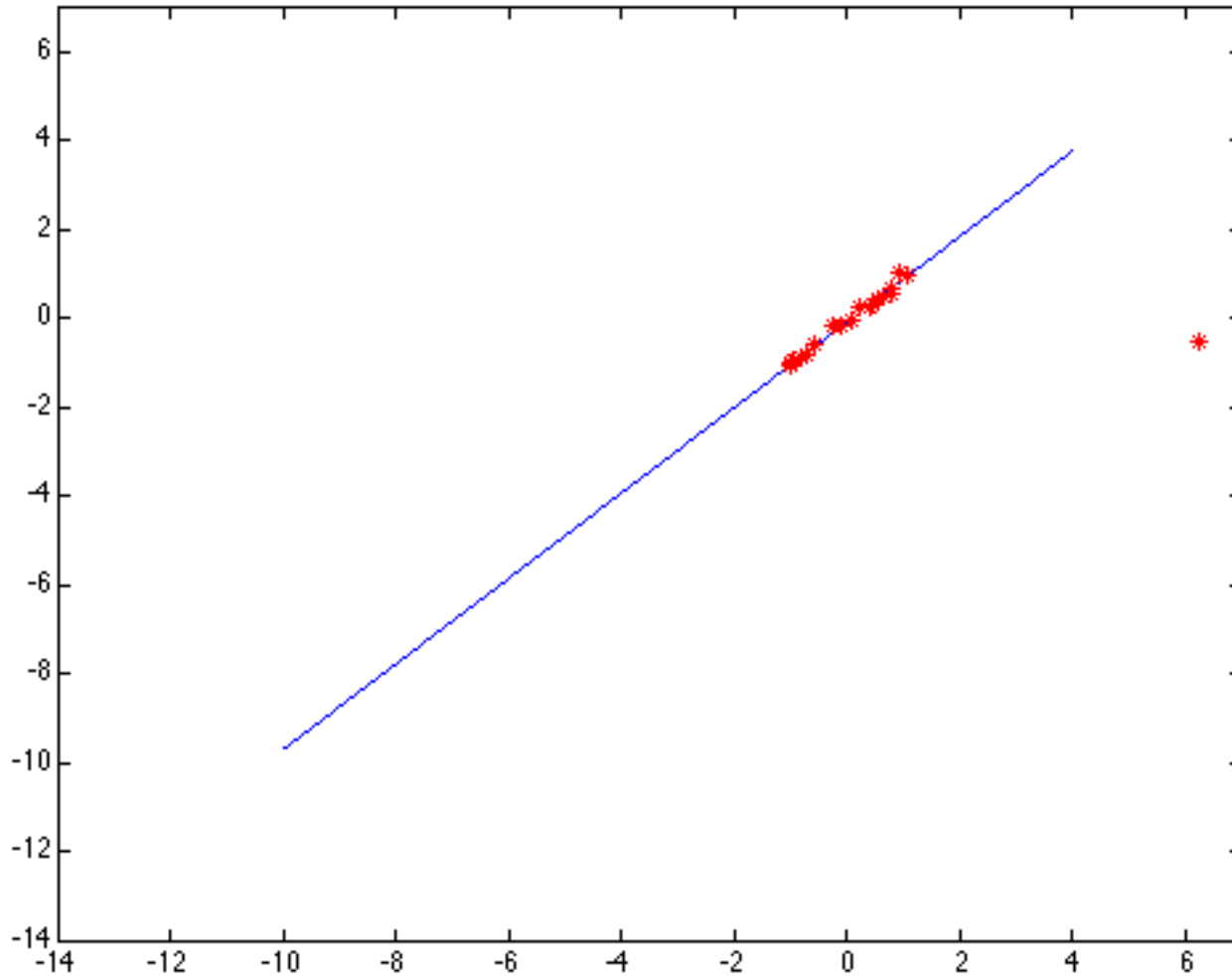
ρ – robust function with scale parameter σ



The robust function ρ

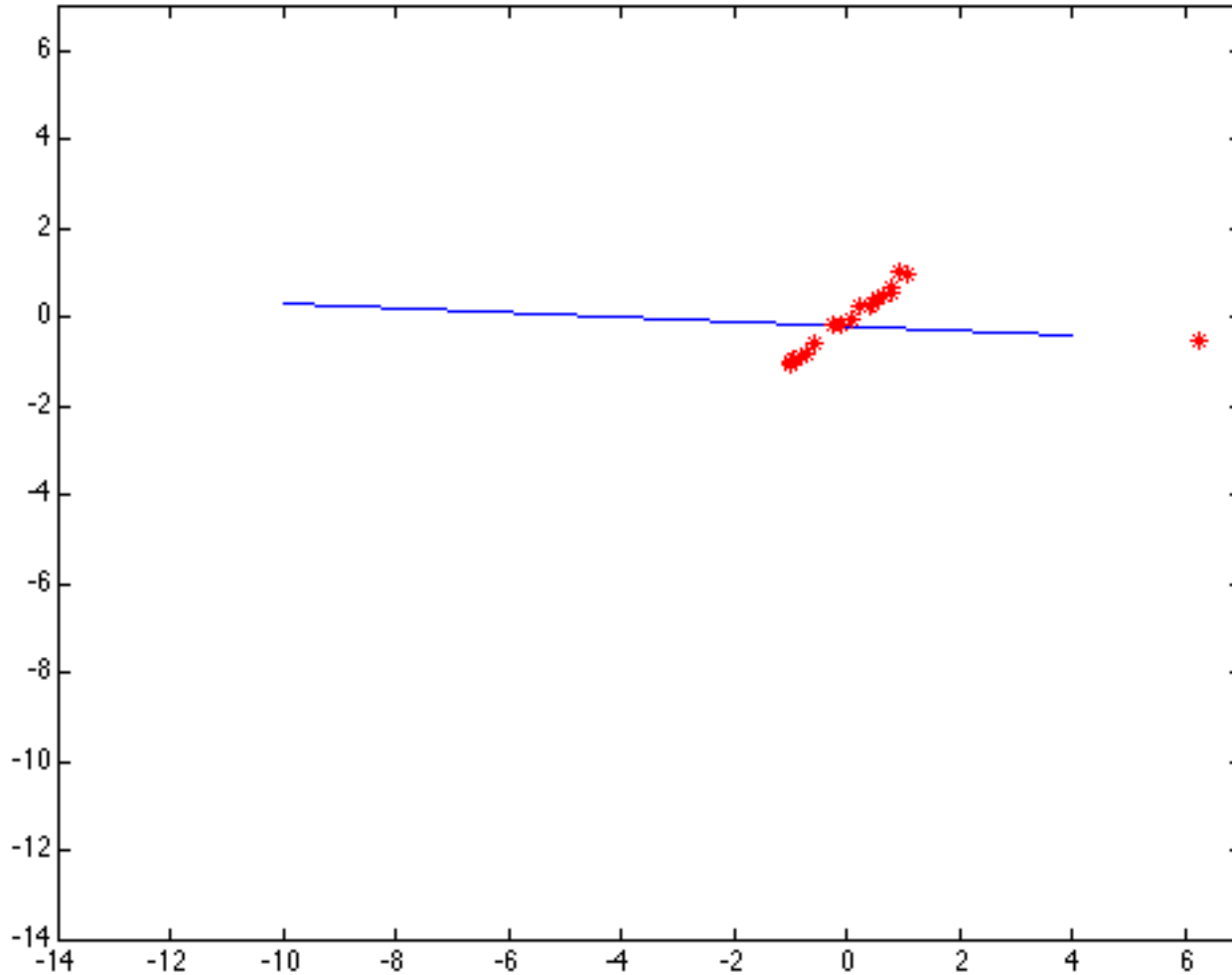
- Favors a configuration with small residuals
- Constant penalty for large residuals

Choosing the scale: Just right



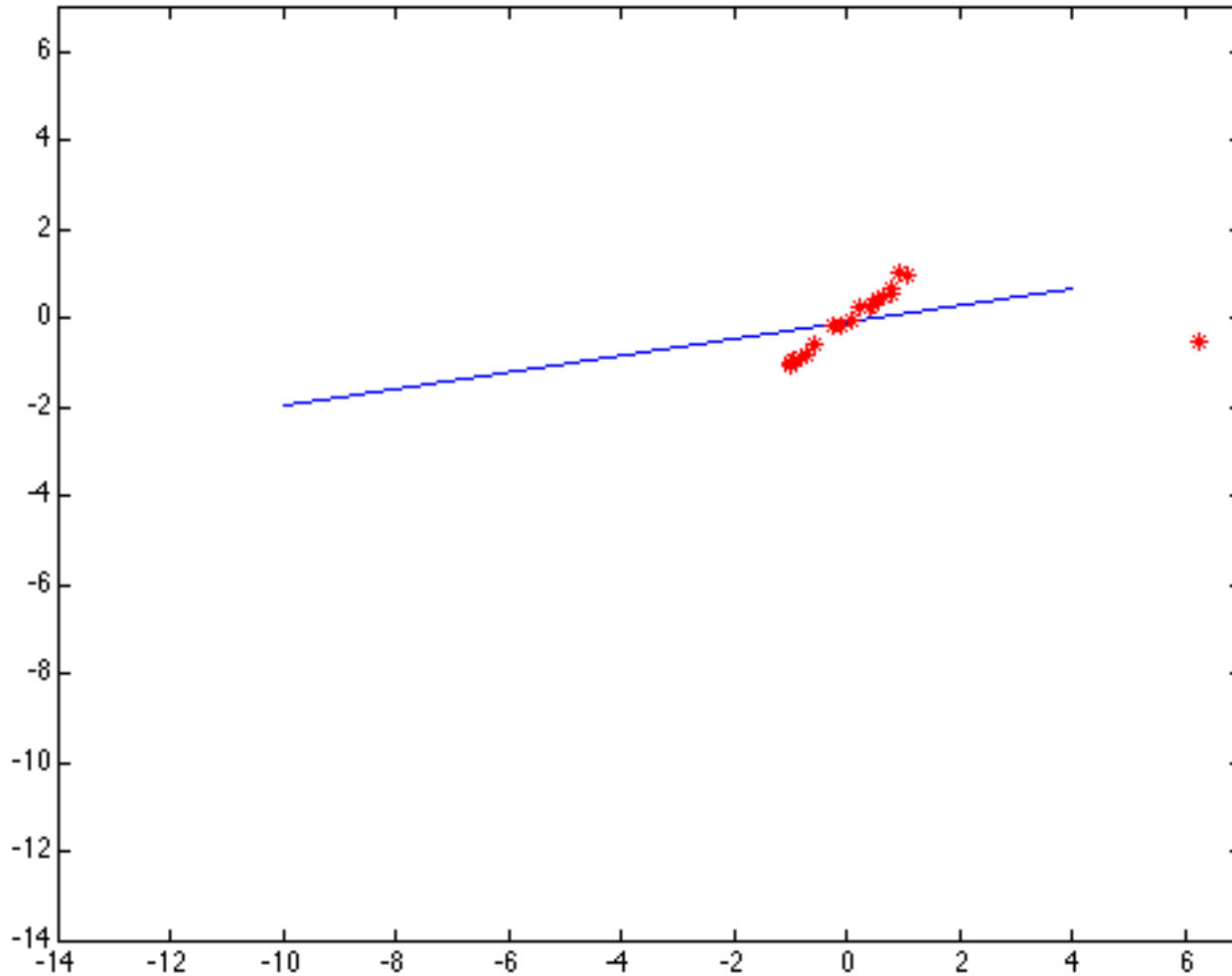
The effect of the outlier is minimized

Choosing the scale: Too small



The error value is almost the same for every point and the fit is very poor

Choosing the scale: Too large



Behaves much the same as least squares

Robust estimation: Details

- Robust fitting is a nonlinear optimization problem that must be solved **iteratively**
- Least squares solution can be used for initialization
- Scale of robust function should be chosen adaptively based on median residual

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

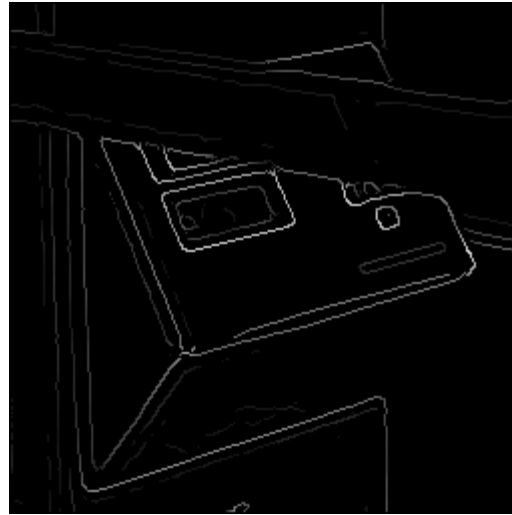
Finding lines using Hough transform

- Using m, b parameterization
- Using r, θ parameterization
 - Using oriented gradients
- Practical considerations
 - Bin size
 - Smoothing
 - Finding multiple lines
 - Finding line segments

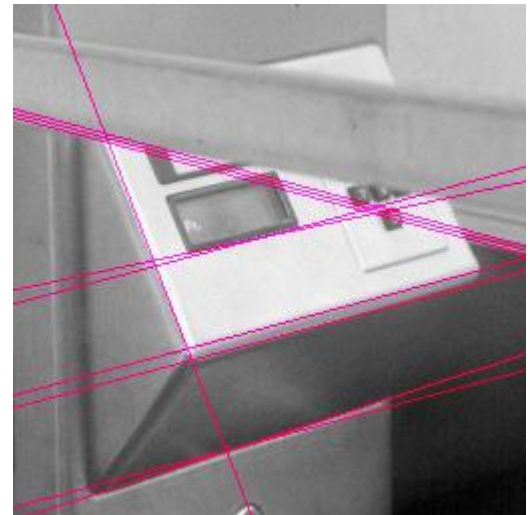
Real World Example



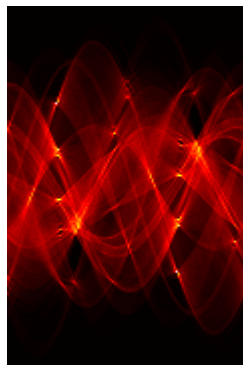
Original



Edge
Detection



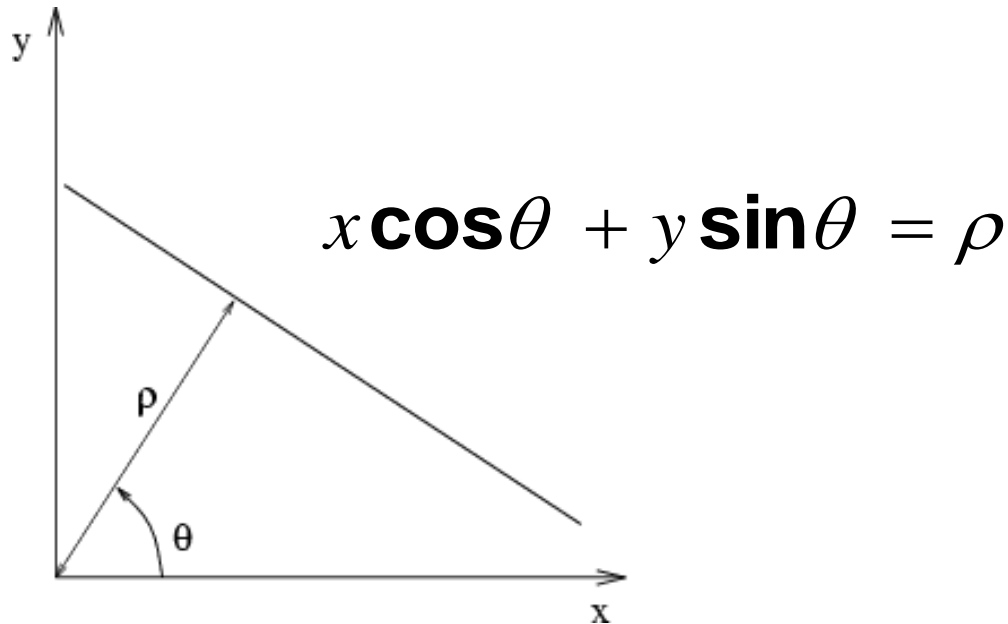
Found Lines



Parameter Space

Parameter space representation

- Problems with the (m,b) space:
 - Unbounded parameter domain
 - Vertical lines require infinite m
- **Alternative: polar representation**

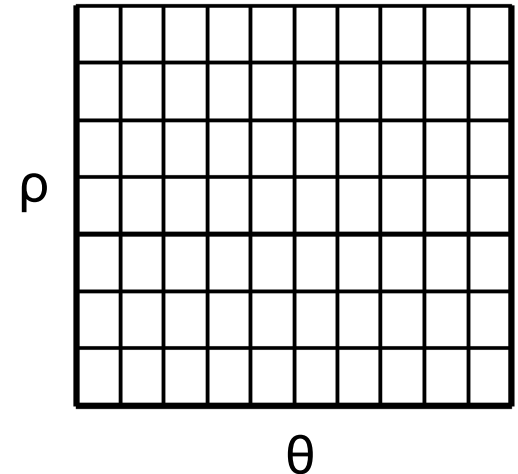


Each point will add a sinusoid in the (θ, ρ) parameter space

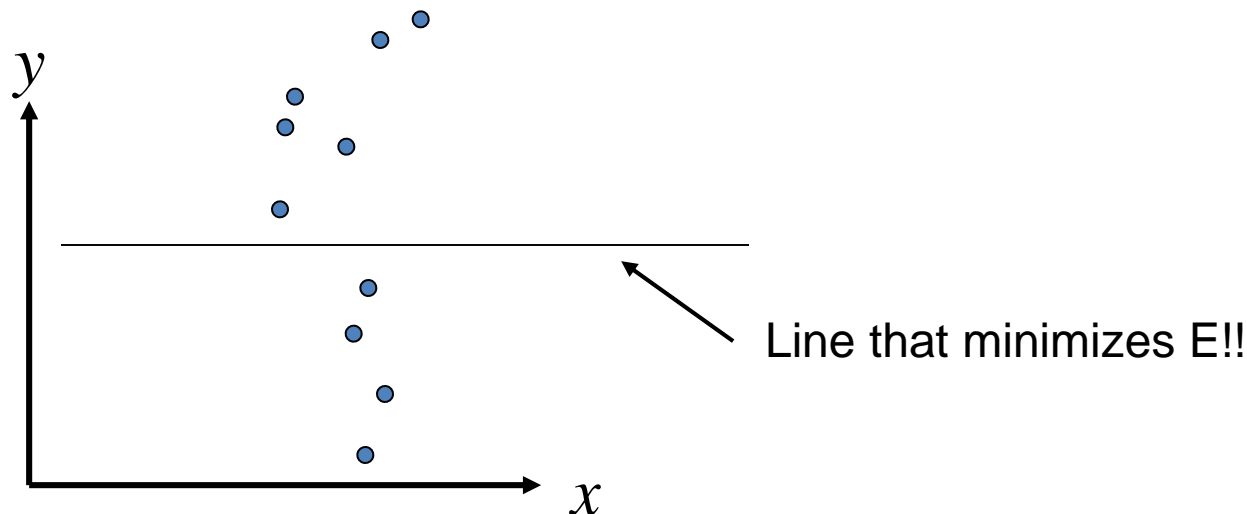
Algorithm outline

- Initialize accumulator H to all zeros
- For each edge point (x,y) in the image
 - For $\theta = 0$ to 180
 - $\rho = x \cos \theta + y \sin \theta$
 - $H(\theta, \rho) = H(\theta, \rho) + 1$
 - end
- end
- Find the value(s) of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
 - The detected line in the image is given by $\rho = x \cos \theta + y \sin \theta$

H: accumulator array (votes)



Problem with Parameterization



Solution: Use a different parameterization

(same as the one we used in computing Minimum Moment of Inertia)

$$E = \frac{1}{N} \sum_i (\rho - x_i \cos \theta + y_i \sin \theta)^2$$

Note: Error E must be formulated carefully!

Curve Fitting

Curve Fitting

Find Polynomial:

$$y = f(x) = ax^3 + bx^2 + cx + d$$

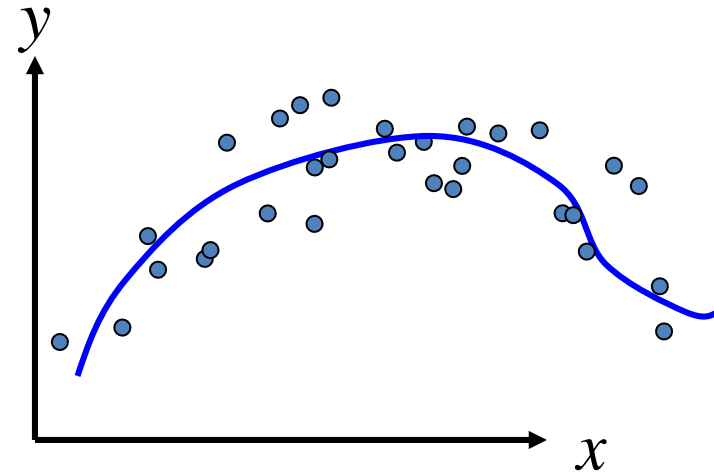
that best fits the given points (x_i, y_i)

Minimize:

$$\frac{1}{N} \sum_i [y_i - (ax_i^3 + bx_i^2 + cx_i + d)]^2$$

Using: $\frac{\partial E}{\partial a} = 0$, $\frac{\partial E}{\partial b} = 0$, $\frac{\partial E}{\partial c} = 0$, $\frac{\partial E}{\partial d} = 0$

Note: $f(x)$ is LINEAR in the parameters (a, b, c, d)



Fitting

Circles

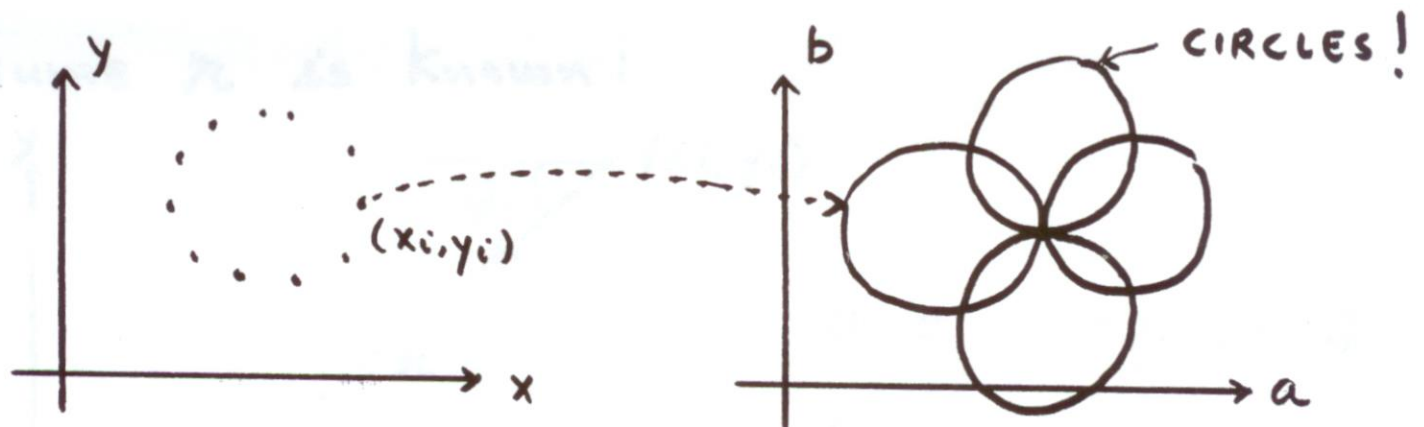
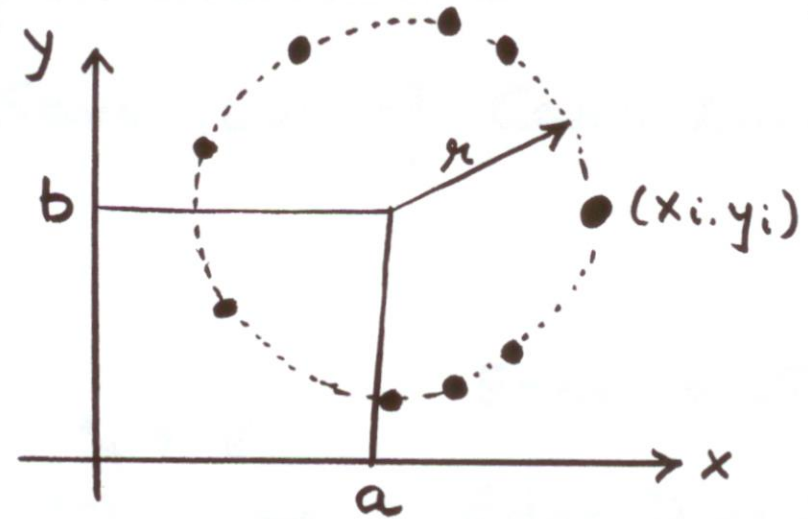
Finding Circles by Hough Transform

Equation of Circle:

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

If radius is known: (2D Hough Space)

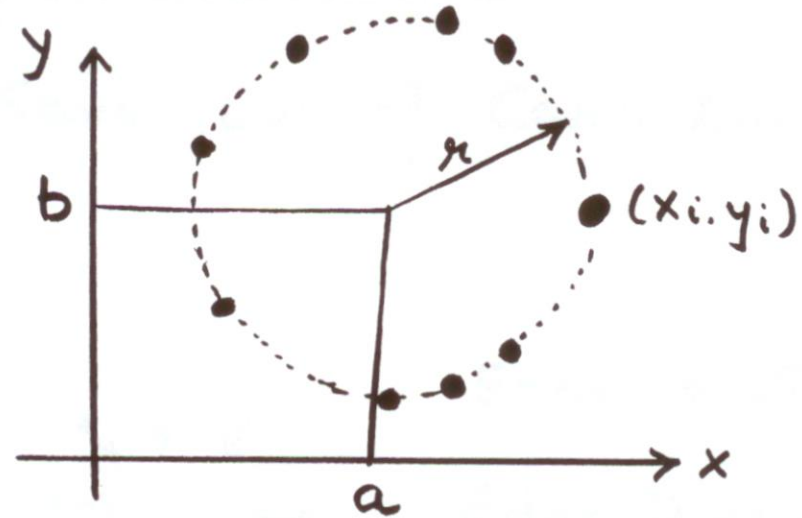
Accumulator Array $A(a, b)$



Finding Circles by Hough Transform

Equation of Circle:

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



If radius is not known: **3D Hough Space!**

Use Accumulator array $A(a, b, r)$

What is the surface in the hough space?

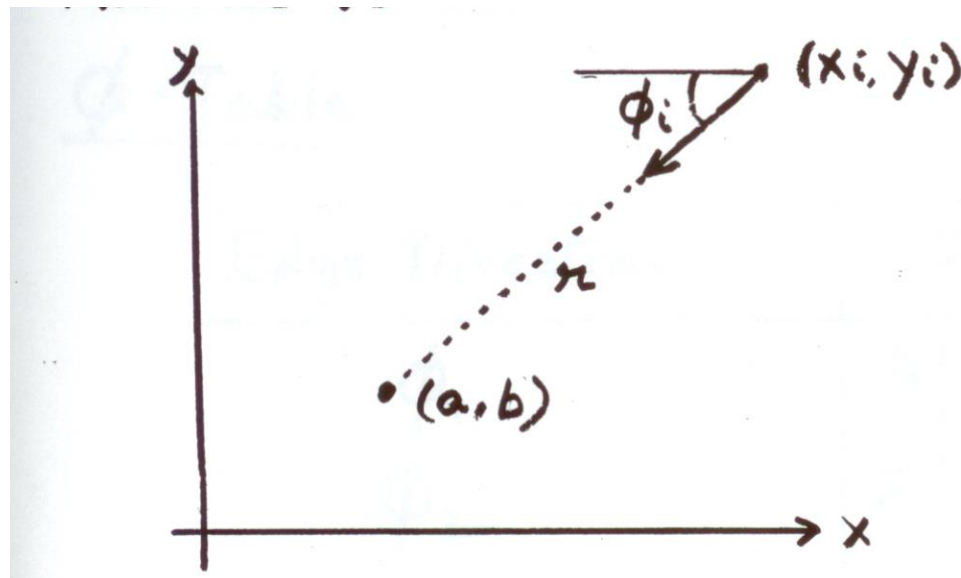
Using Gradient Information

- Gradient information can save lot of computation:

Edge Location (x_i, y_i)

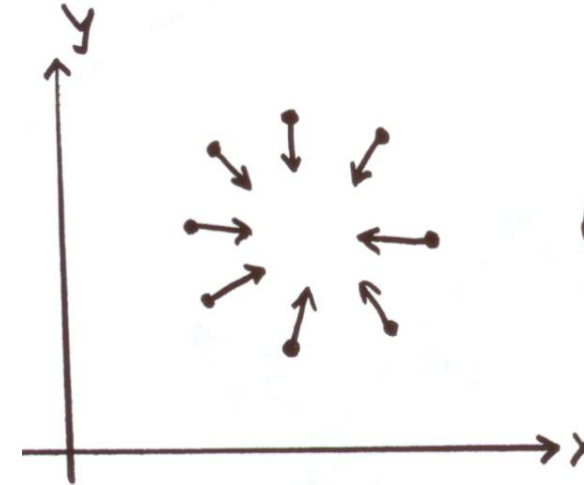
Edge Direction ϕ_i

Assume radius is known:



$$a = x - r \cos \phi$$

$$b = y - r \sin \phi$$



Need to increment only one point in Accumulator!!

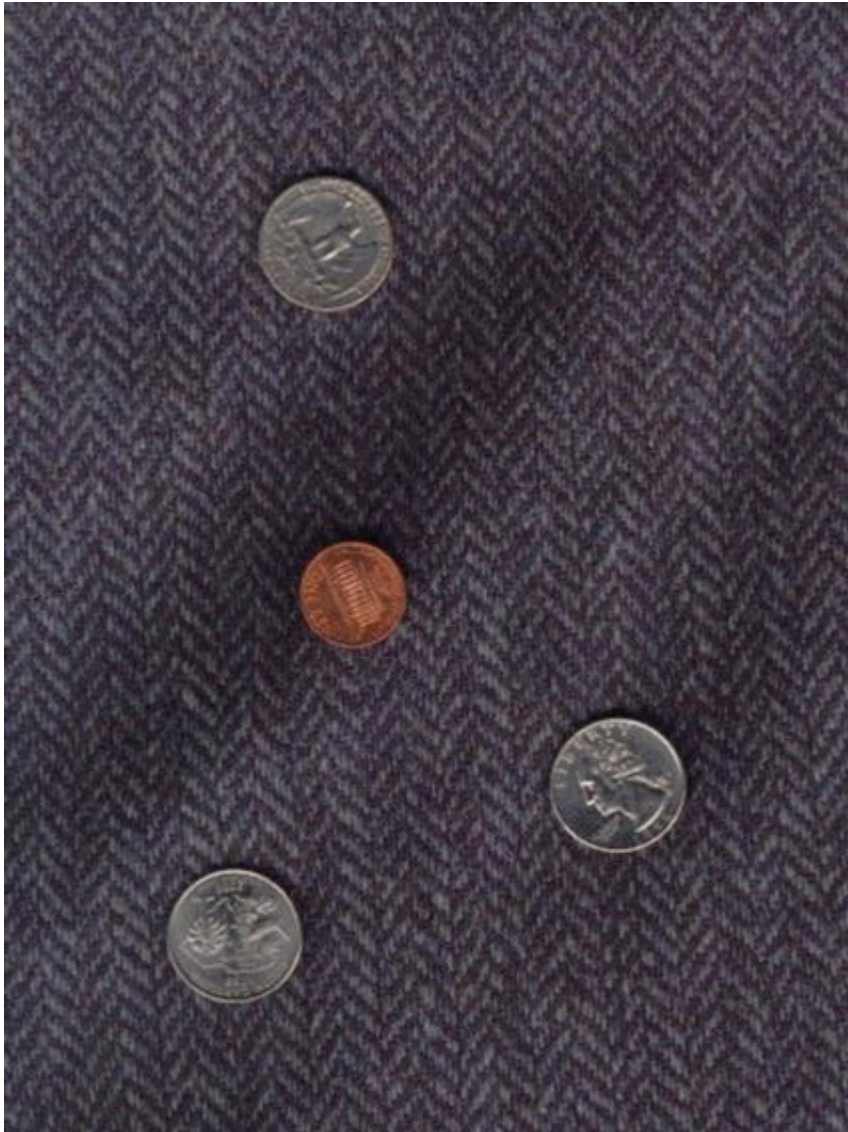
Real World Circle Examples



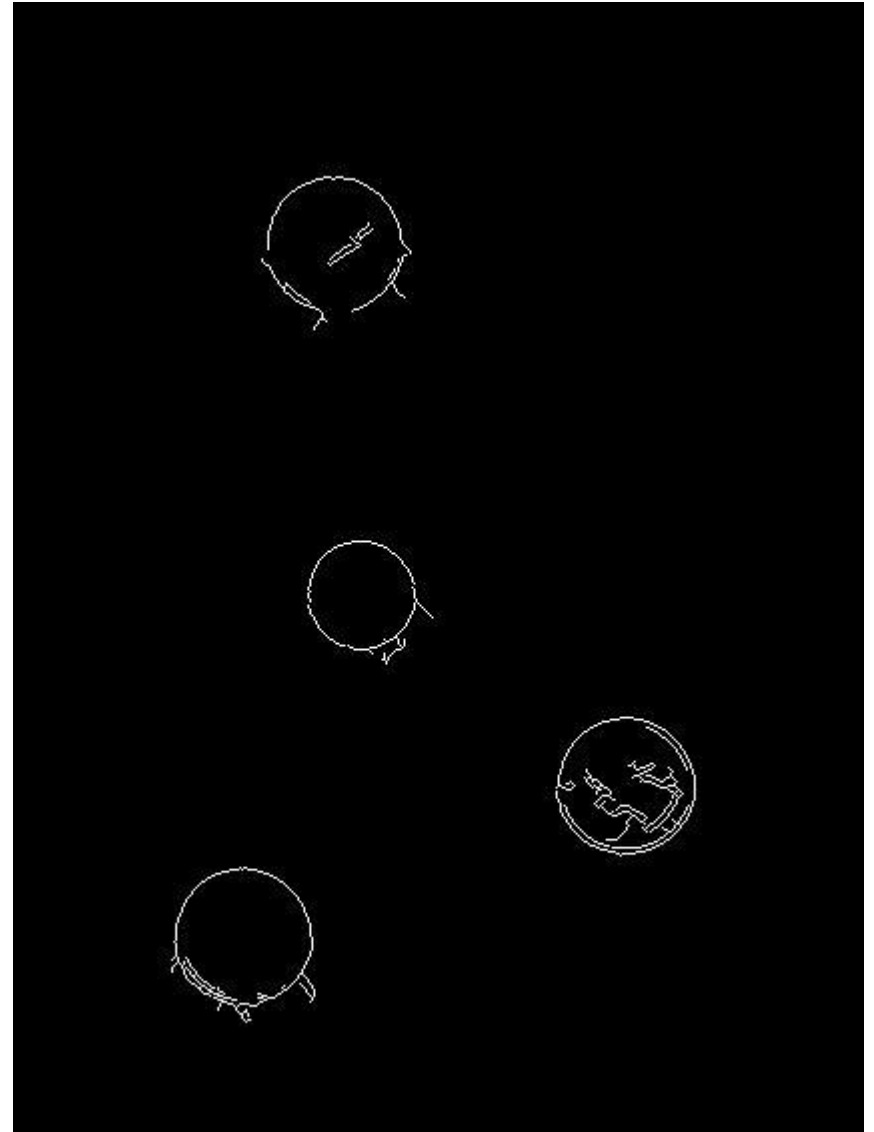
Crosshair indicates results of Hough transform, bounding box found via motion differencing.

Finding Coins

Original



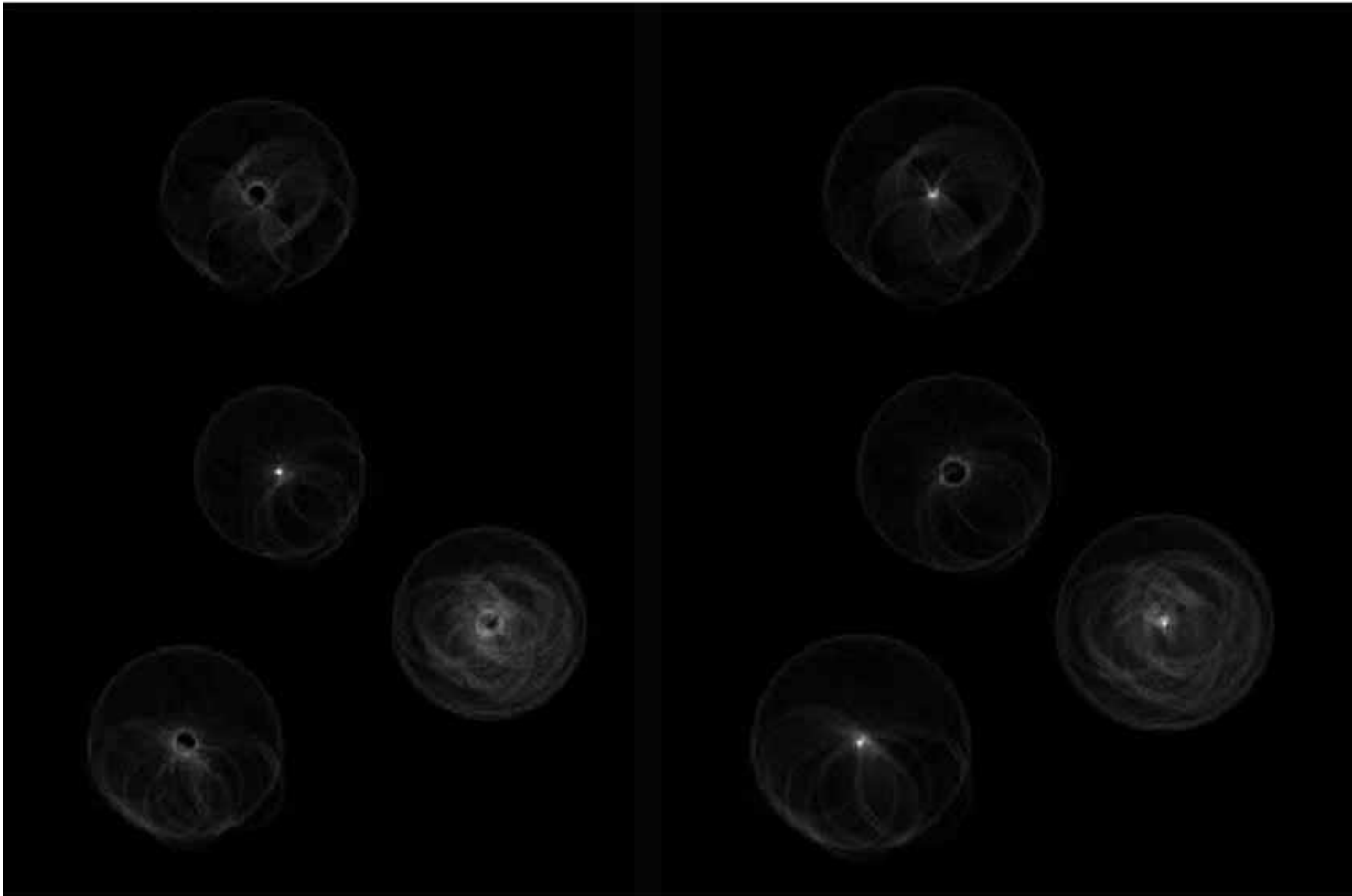
Edges (note noise)



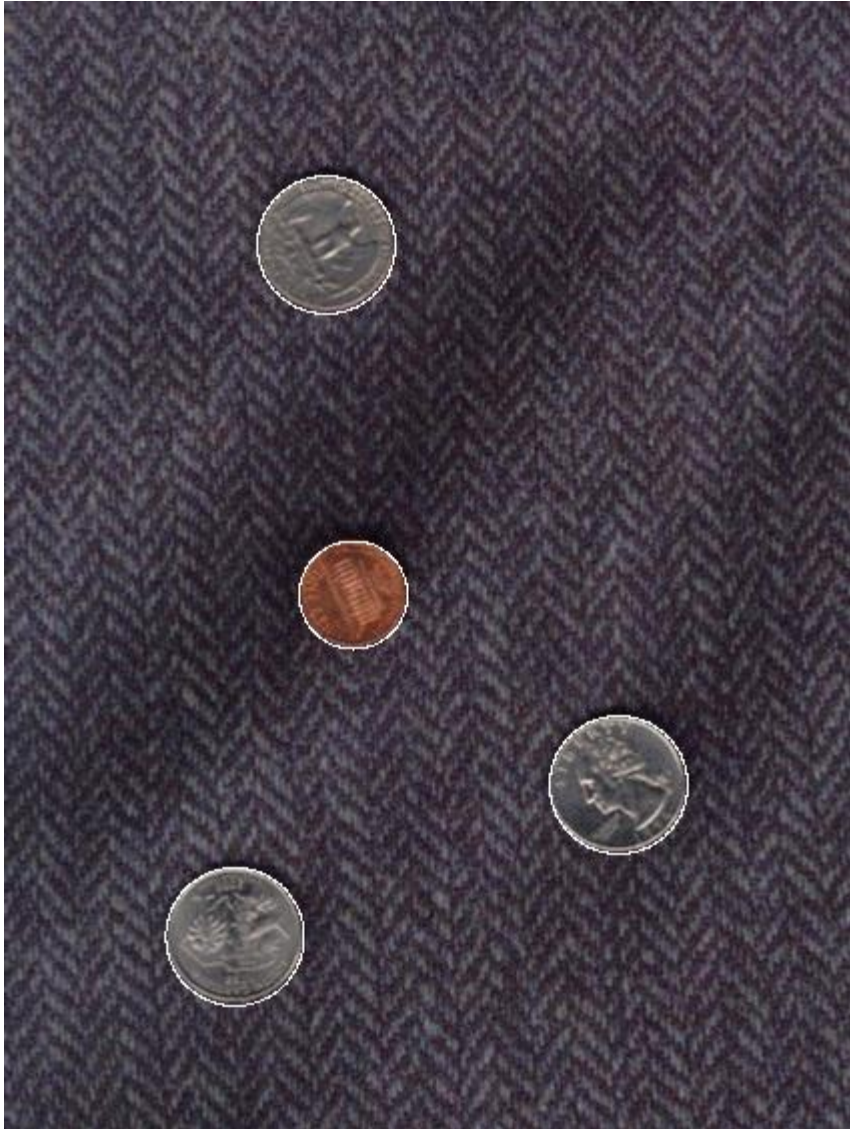
Finding Coins (Continued)

Penn

Quarters



Finding Coins (Continued)



Note that because the quarters and penny are different sizes, a different Hough transform (with separate accumulators) was used for each circle size.

Coin finding sample images from: Vivek Kwatra

Mechanics of the Hough transform

- **Difficulties**
 - how big should the cells be? (too big, and we merge quite different lines; too small, and noise causes lines to be missed)
- **How many lines?**
 - Count the peaks in the Hough array
 - Treat adjacent peaks as a single peak
- **Which points belong to each line?**
 - Search for points close to the line
 - Solve again for line and iterate

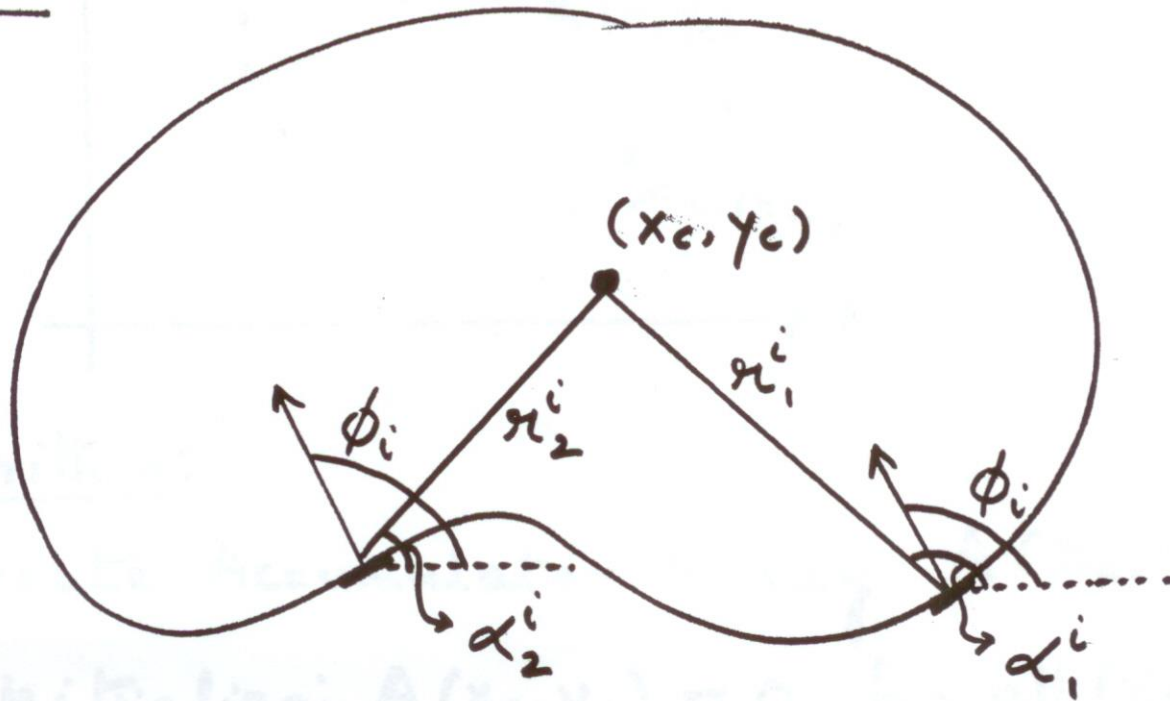
This is difficult because of:

- **Extraneous data:** clutter or multiple models
 - We do not know what is part of the model?
 - Can we pull out models with a few parts from much larger amounts of background clutter?
- **Missing data:** only some parts of model are present
- **Noise**
- **Cost:**
 - It is not feasible to check all combinations of features by fitting a model to each possible subset

Generalized Hough Transform

- **Model Shape NOT described by equation**

Model :



Generalized Hough Transform

- Model Shape NOT described by equation

ϕ -Table

Edge Direction	$\bar{\pi} = (\pi, \alpha)$
ϕ_1	$\bar{\pi}_1^1, \bar{\pi}_2^1, \bar{\pi}_3^1$
ϕ_2	$\bar{\pi}_1^2, \bar{\pi}_2^2$
\vdots	\vdots
ϕ_i	$\bar{\pi}_1^i, \bar{\pi}_2^i$
\vdots	\vdots
ϕ_n	$\bar{\pi}_1^n, \bar{\pi}_2^n$

Generalized Hough Transform

Find Object Center (x_c, y_c) given edges (x_i, y_i, ϕ_i)

Create Accumulator Array $A(x_c, y_c)$

Initialize: $A(x_c, y_c) = 0 \quad \forall (x_c, y_c)$

For each edge point (x_i, y_i, ϕ_i)

For each entry \overline{r}_k^i in table, compute:

$$x_c = x_i + r_k^i \cos \alpha_k^i$$

$$y_c = y_i + r_k^i \sin \alpha_k^i$$

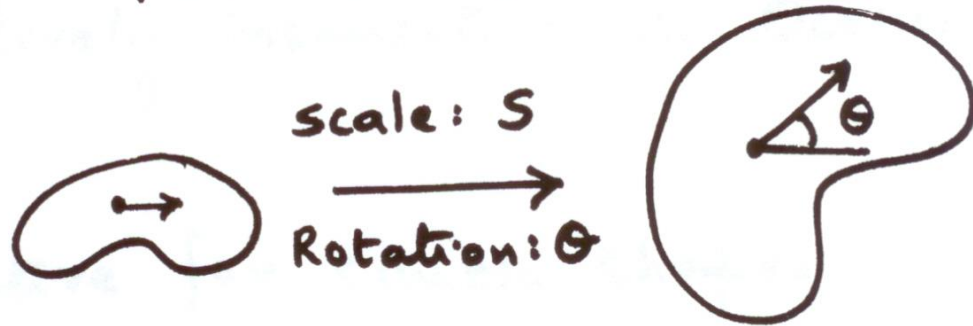
Increment Accumulator: $A(x_c, y_c) = A(x_c, y_c) + 1$

Find Local Maxima in $A(x_c, y_c)$

Scale & Rotation:

Use Accumulator Array:

$$A[x_c, y_c, S, \theta]$$



Use:

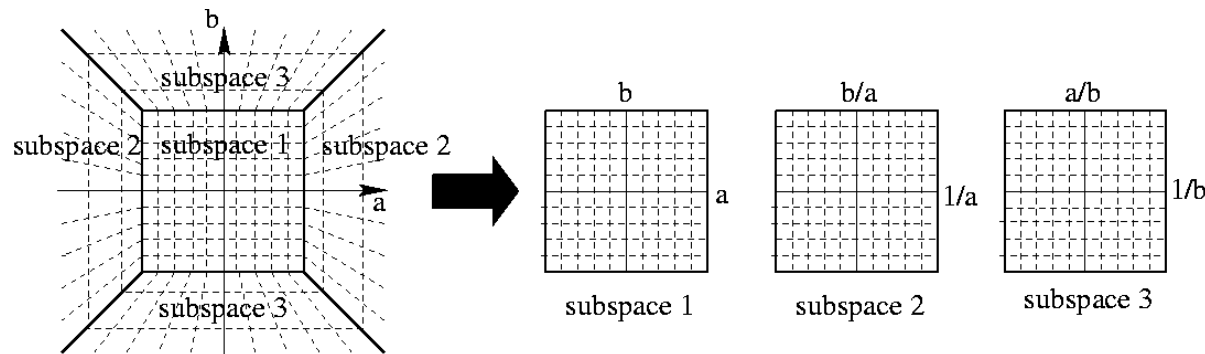
$$x_c = x_i + r_k^i S \cos(\alpha_k^i + \theta)$$

$$y_c = y_i + r_k^i S \sin(\alpha_k^i + \theta)$$

$$A(x_c, y_c, S, \theta) = A(x_i, y_i, S, \theta) + 1.$$

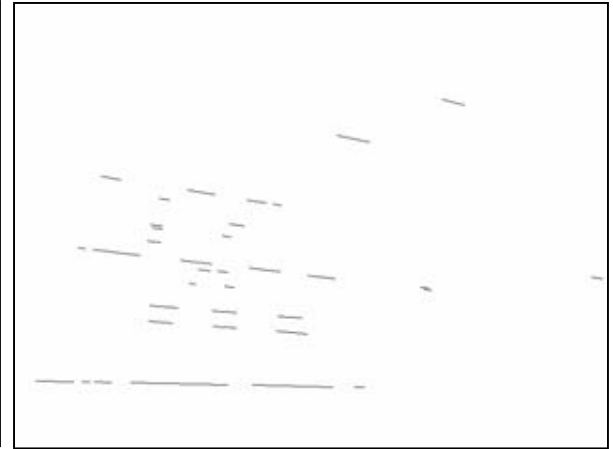
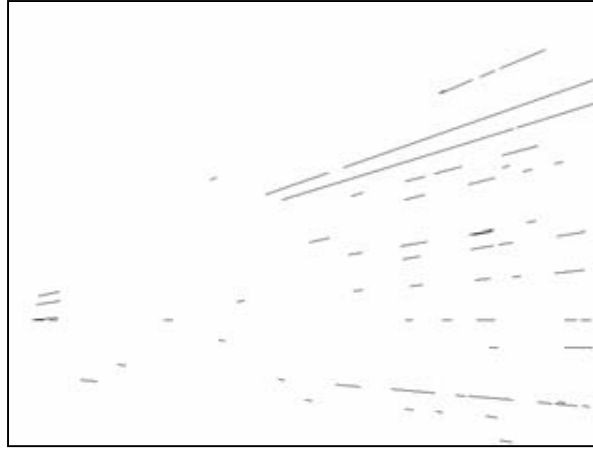
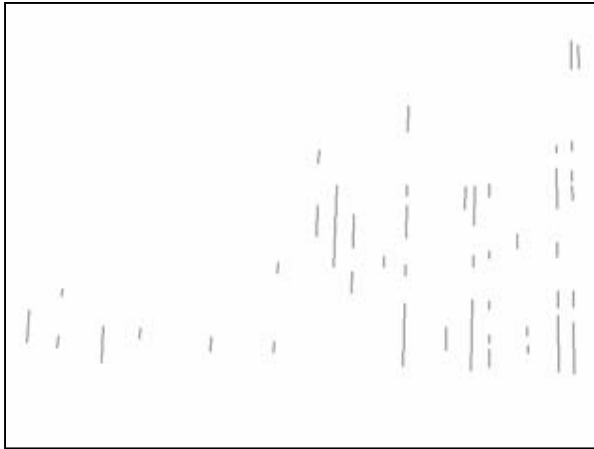
Extension: Cascaded Hough transform

- Let's go back to the original (m, b) parametrization
- A line in the image maps to a pencil of lines in the Hough space
- What do we get with parallel lines or a pencil of lines?
 - Collinear peaks in the Hough space!
- **So we can apply a Hough transform to the output of the first Hough transform to find vanishing points**
- Issue: dealing with unbounded parameter space



T. Tuytelaars, M. Proesmans, L. Van Gool ["The cascaded Hough transform,"](#)
ICIP, vol. II, pp. 736-739, 1997.

Cascaded Hough transform



T. Tuytelaars, M. Proesmans, L. Van Gool ["The cascaded Hough transform,"](#)
ICIP, vol. II, pp. 736-739, 1997.

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 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

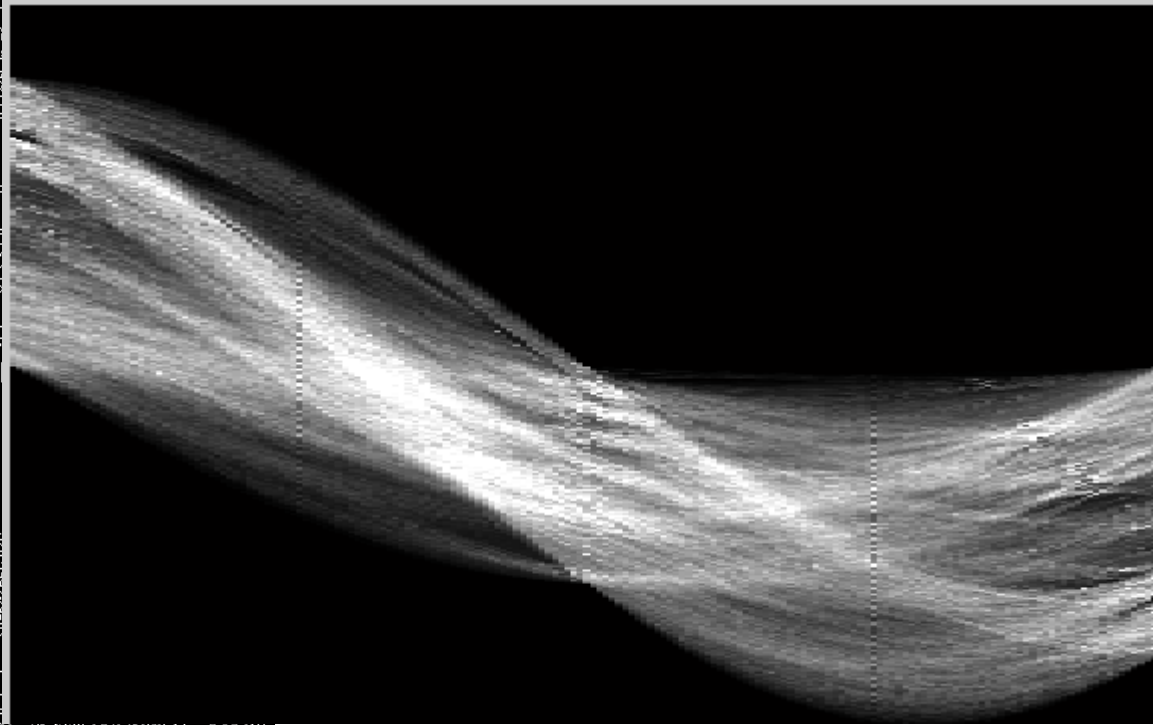
Common applications

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

1. Image \rightarrow Canny



2. Canny \rightarrow Hough votes

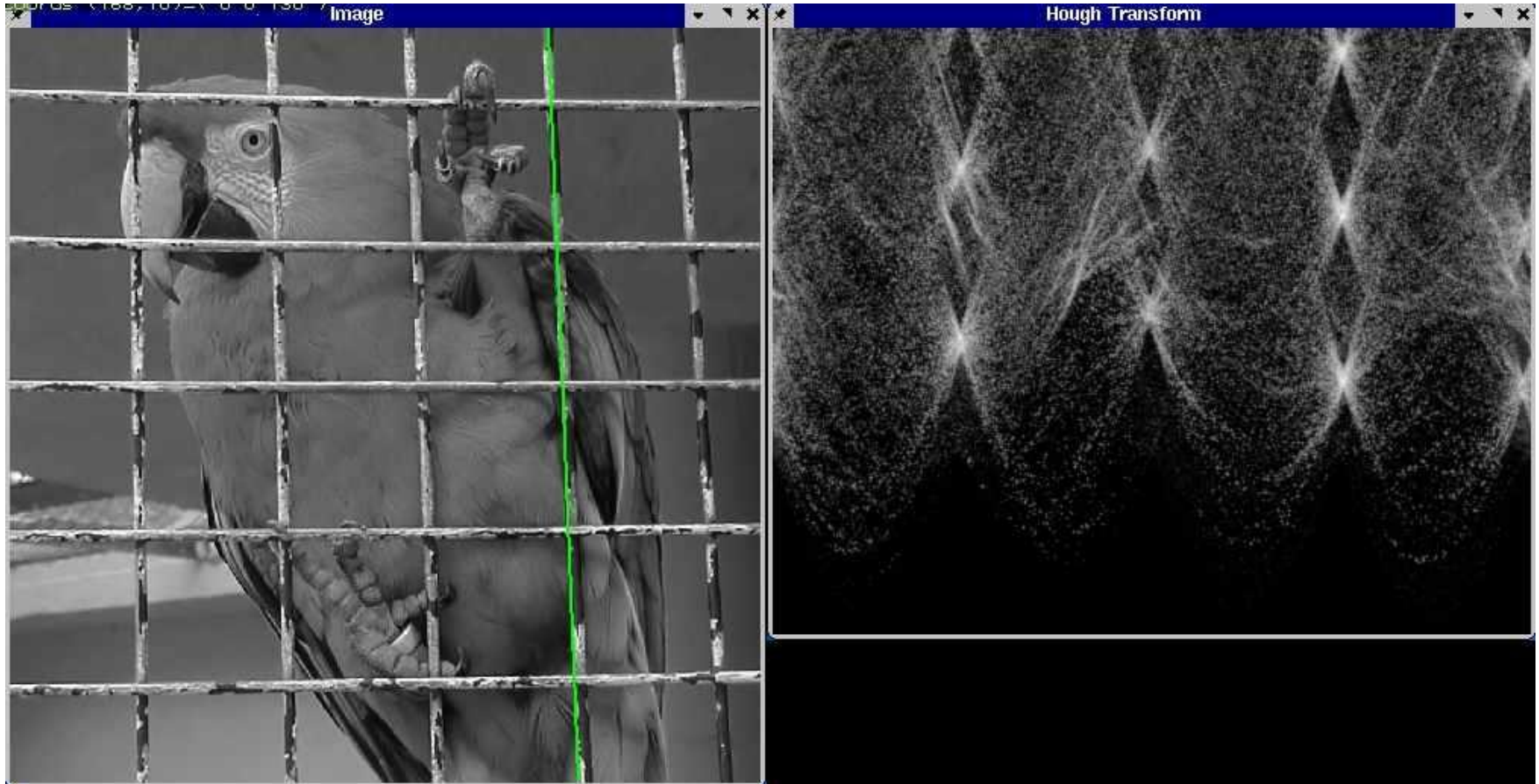


3. Hough votes \rightarrow Edges

Find peaks and post-process



Hough transform example



Additional Contents

Better Parameterization

NOTE: $-\infty \leq m \leq \infty$

Large Accumulator

More memory and computations

Improvement: (Finite Accumulator Array Size)

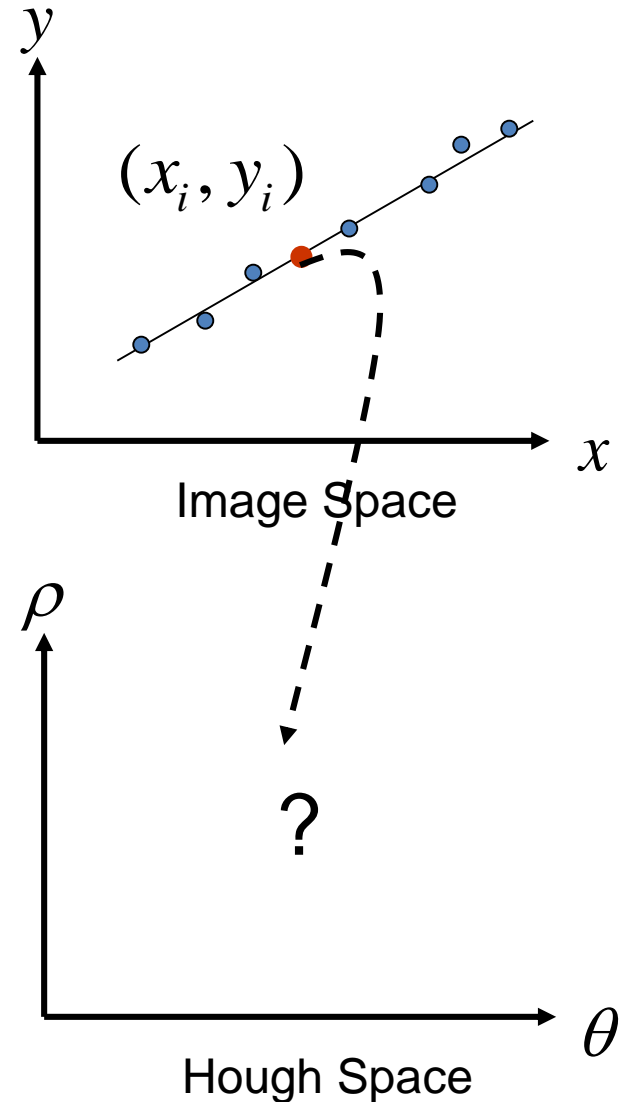
Line equation: $\rho = -x \cos \theta + y \sin \theta$

Here $0 \leq \theta \leq 2\pi$

$$0 \leq \rho \leq \rho_{\max}$$

Given points (x_i, y_i) find (ρ, θ)

Hough Space Sinusoid



Hough Transform: Comments

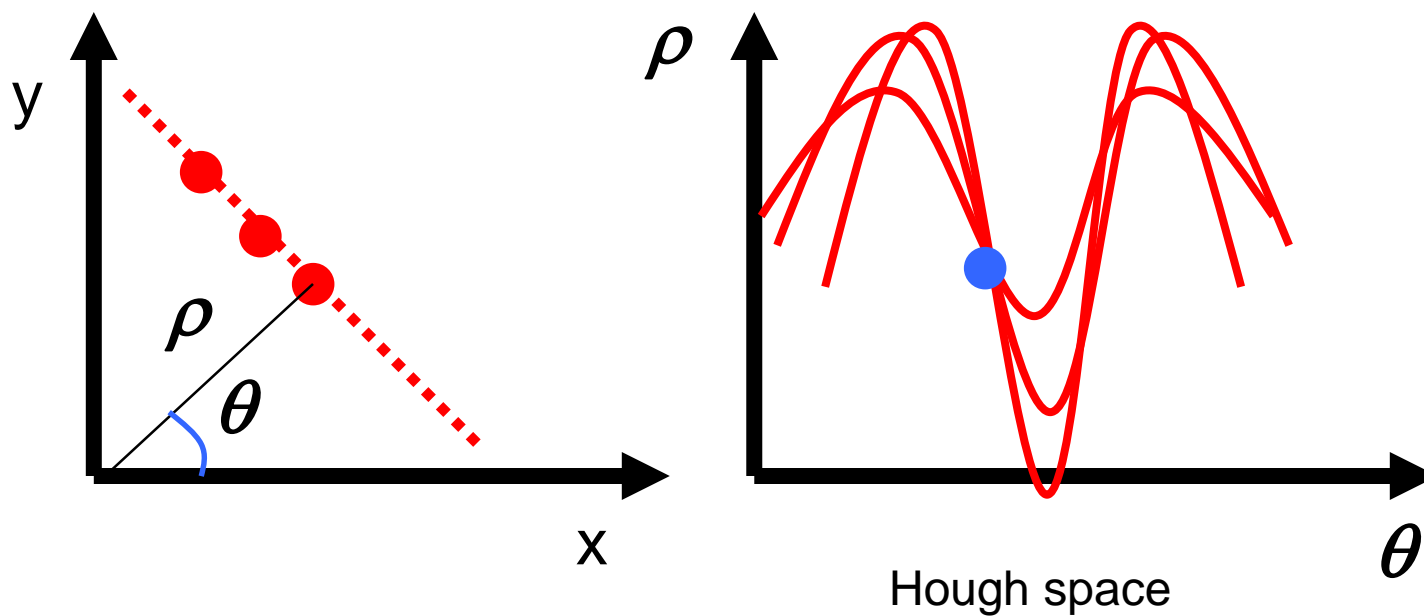
- Works on Disconnected Edges
- Relatively insensitive to occlusion
- Effective for simple shapes (lines, circles, etc)
- Trade-off between work in Image Space and Parameter Space
- Handling inaccurate edge locations:
 - Increment Patch in Accumulator rather than a single point

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



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

Practical details

- Try to get rid of irrelevant features
 - Take only edge points with significant gradient magnitude
- Choose a good grid / discretization
 - Too coarse: large votes obtained when too many different lines correspond to a single bucket
 - Too fine: miss lines because some points that are not exactly collinear cast votes for different buckets
- Increment neighboring bins (smoothing in accumulator array)
- Who belongs to which line?
 - Tag the votes

Hough transform: Pros

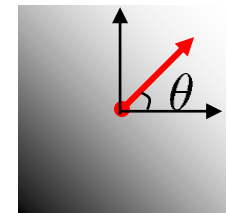
- Can deal with non-locality and occlusion
- Can detect multiple instances of a model in a single pass
- Some robustness to noise: noise points unlikely to contribute consistently to any single bin

Hough transform: Cons

- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- It's hard to pick a good grid size

Extension: Incorporating image gradients

- Recall: when we detect an edge point, we also know its gradient direction
 - But this means that the line is uniquely determined!
 - Modified Hough transform:
 - For each edge point (x,y)
 - $\theta =$ gradient orientation at (x,y)
 - $\rho = x \cos \theta + y \sin \theta$
 - $H(\theta, \rho) = H(\theta, \rho) + 1$
- end


$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$
$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$