

3D Vision 1

Week 7

Introduction to 3D Vision
Model Fitting

Announcements

- Where student lab attendance dropped below a threshold for multiple weeks, the School has taken action to merge labs

ORIGINAL				NEW				
Day	Start Time	End Time	Location	Day	Start Time	End Time	Location	
Wed	13:00	15:00	Rm N113_CSIT Bldg 108	→	Wed	13:00	15:00	Rm N109_CSIT Bldg 108
Wed	14:00	16:00	Rm N112_CSIT Bldg 108	→	Wed	14:00	16:00	Comp Lab 1.23_Hanna Neumann Bldg 145
Wed	15:00	17:00	Rm N109_CSIT Bldg 108	→	Wed	14:00	16:00	Comp Lab 1.23_Hanna Neumann Bldg 145
Fri	11:00	13:00	Rm N113_CSIT Bldg 108	→	Fri	11:00	13:00	Rm N114_CSIT Bldg 108
Fri	13:00	15:00	Comp Lab 1.23_Hanna Neumann Bldg 145	→	Fri	13:00	15:00	Rm N114_CSIT Bldg 108
Fri	13:00	15:00	Rm N113_CSIT Bldg 108	→	Fri	13:00	15:00	Rm N114_CSIT Bldg 108

Announcements

- **Reminder:** 30-minute tutor drop-in session
 - E.g., for a lab scheduled from 2:00–4:00PM, we have:

Time	Activity
2:00PM–2:30PM	Tutor Drop-in Session
2:30PM–4:00PM	Lab

Announcements

- Tutorial 5, on backpropagation, is available on Wattle and will be covered during the labs this week
- Assignment 2 due 11:59pm Friday 26 April (end of Week 8)
 - This includes a one week extension that has already been applied
 - **Zero marks** if either report or code submitted late (unless extension)
 - Submit early; you can always resubmit an updated version later
 - Depending on your internet connection and load on the TurnItIn servers, uploading can sometimes be slow, so please factor this into your submission schedule
 - Submit your report (PDF) and code (ZIP file) **separately under the correct tab** in the submission box
 - Follow the instructions under Submission Requirements

Weekly Study Plan: Overview

Wk	Starting	Lecture	Lab	Assessment
1	19 Feb	Introduction	X	
2	26 Feb	Low-level Vision 1	1	
3	4 Mar	Low-level Vision 2	1	
		Mid-level Vision 1		
4	11 Mar	Mid-level Vision 2	1	CLab1 report due Friday
		High-level Vision 1		
5	18 Mar	High-level Vision 2	2	
6	25 Mar	High-level Vision 3 ¹	2	
	1 Apr	Teaching break	X	
	8 Apr	Teaching break	X	
7	15 Apr	3D Vision 1	2	CLab2 report due Friday
8	22 Apr	3D Vision 2	3	
9	29 Apr	3D Vision 3	3	
10	6 May	3D Vision 4	3	
		Mid-level Vision 3		
11	13 May	High-level Vision 4	X	CLab3 report due Friday
12	20 May	Course Review	X	

Weekly Study Plan: Part B

Wk	Starting	Lecture	By
7	15 Apr	3D vision: introduction, camera model, single-view geometry	Dylan
8	22 Apr	3D vision: camera calibration, two-view geometry (homography)	Dylan
9	29 Apr	3D vision: two-view geometry (epipolar geometry, triangulation, stereo)	Dylan
10	6 May	3D vision: multiple-view geometry	Weijian
		Mid-level vision: optical flow, shape-from-X	Dylan
11	13 May	High-level vision: self-supervised learning, detection, segmentation	Dylan
12	20 May	Course review	Dylan

Outline

1. Introduction to 3D vision
2. Model fitting (line fitting)
 1. Least squares
 2. M-estimation
 3. RANSAC
 4. Hough transform
3. Image formation (review): pinhole camera model
4. Camera projection matrix and single view geometry
5. Camera calibration
6. Resectioning and camera pose

Introduction to 3D Vision

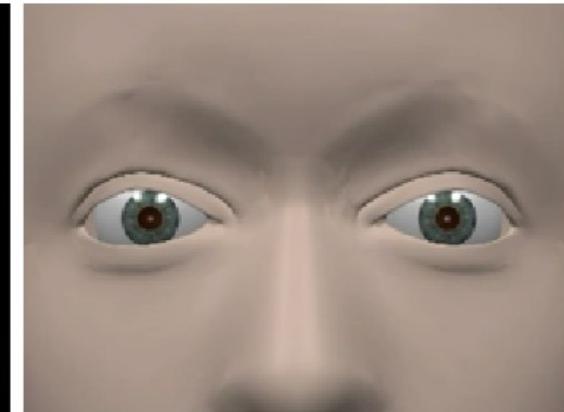
3D Vision: Eight Human Tricks



binocular stereo



accommodation



vergence



motion perspective



occlusion



apparent height



texture density



aerial perspective

Image Credit: Peter Corke

3D Vision: Eight Nine Human Tricks



Perspective Distortion

Image Credit: Pikist

3D Vision: Eight Human Tricks

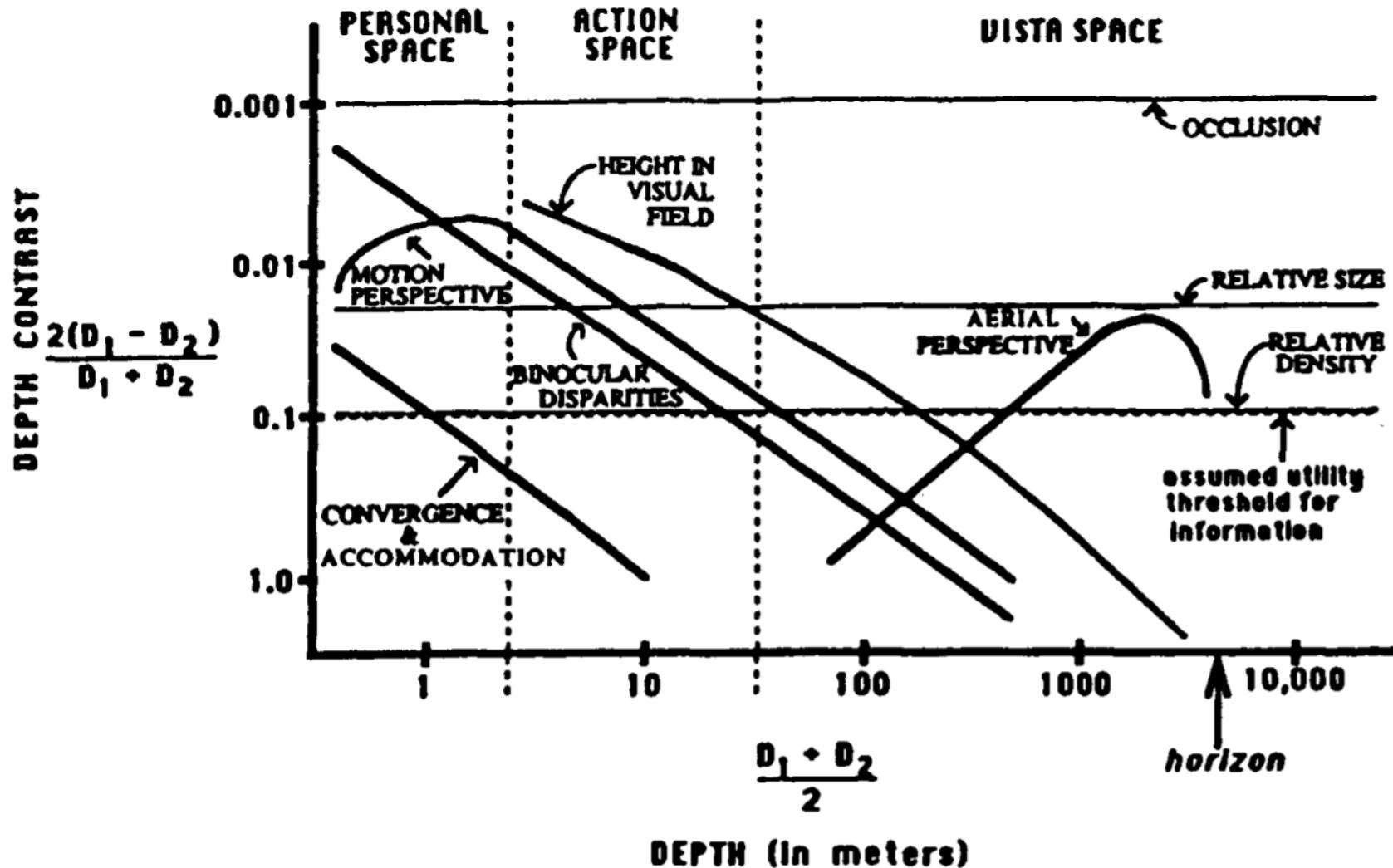
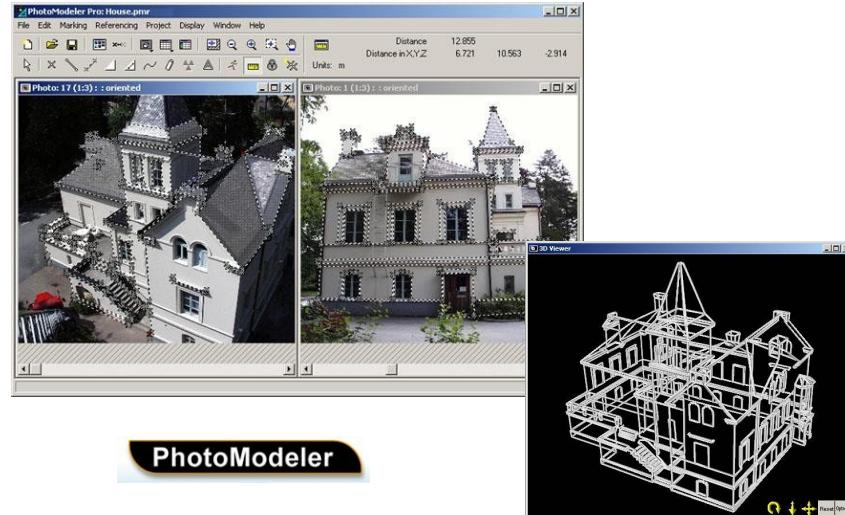


Image Credit: J. Cutting, "How the eye measures reality and virtual reality", 1997

Why 3D Vision?

Structure-from-Motion (SfM):

- Input: set or sequence of images
- Output: camera poses, 3D model
- Applications:
 - Architecture (survey, stability analysis, renovation planning)
 - City modelling
 - Archaeology
 - Inspection (from drones)
 - Surgery



Building Rome in a Day



Credit: S. Agarwal et al., "Building Rome in a day", 2009

NeRF



Credit: J. Barron et al., "Zip-NeRF", ICCV 2023

DUST3R

- <https://dust3r.europe.naverlabs.com/>

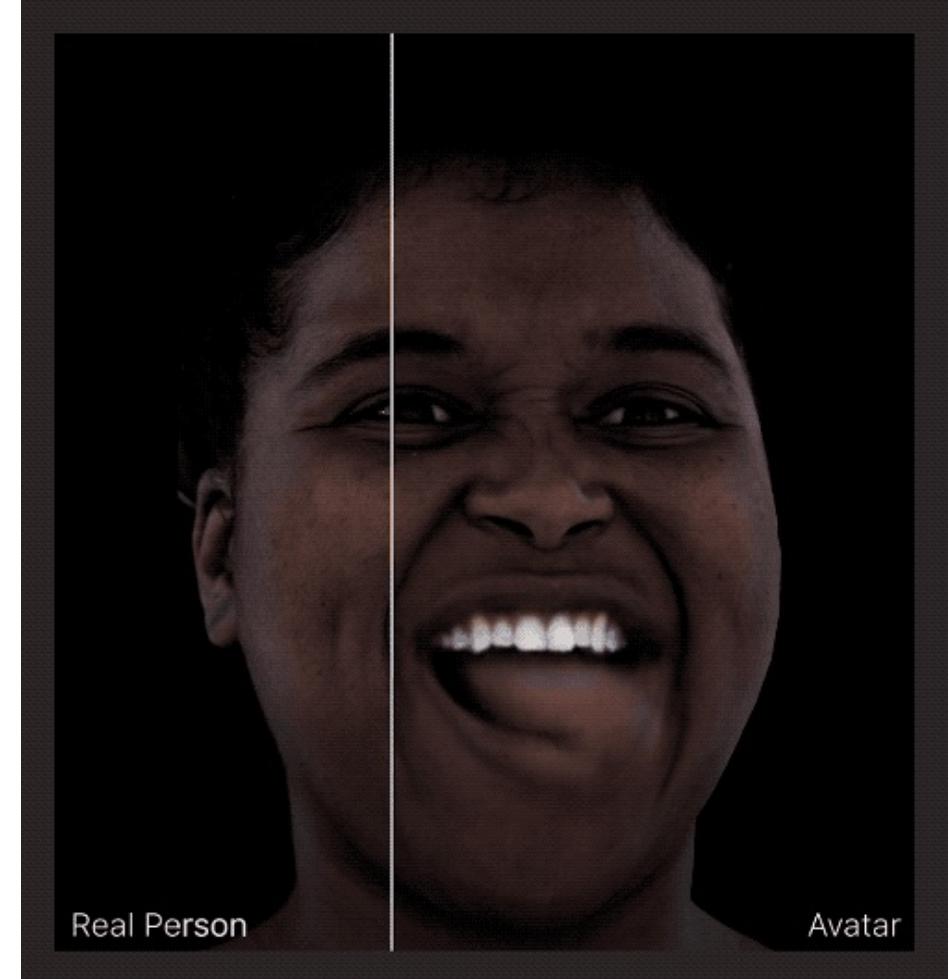
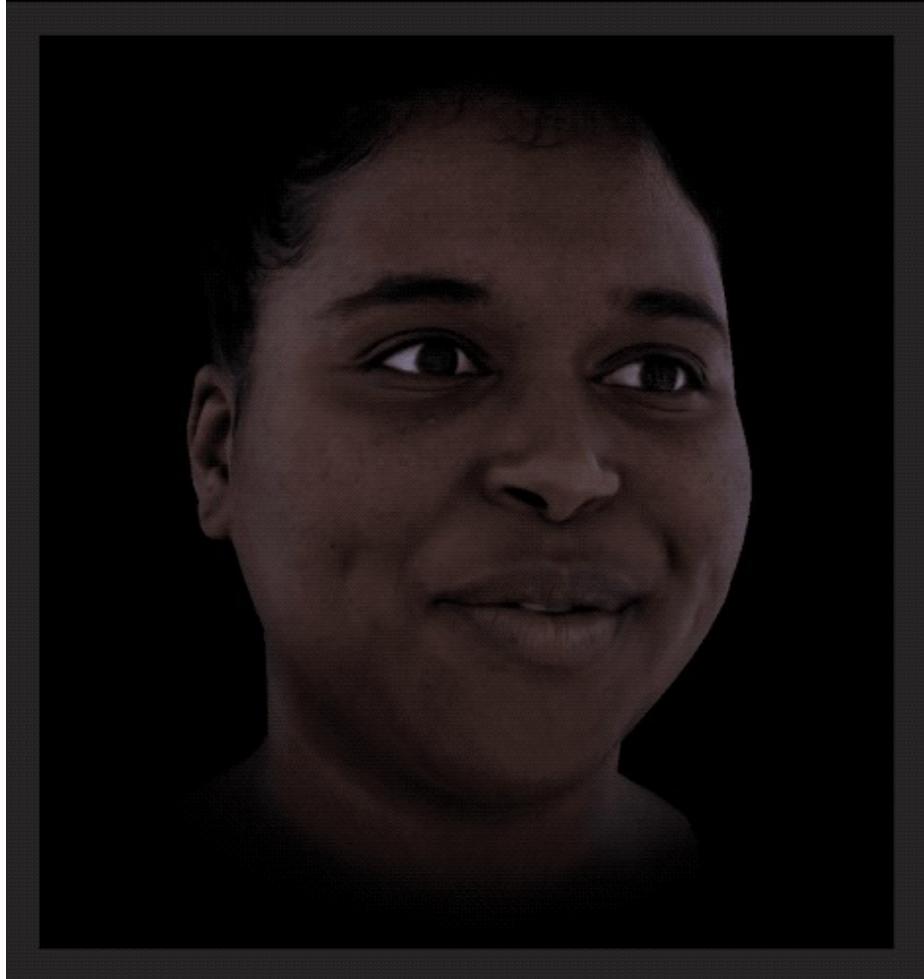
Telepresence



Credit: Microsoft HoloLens

16

Telepresence

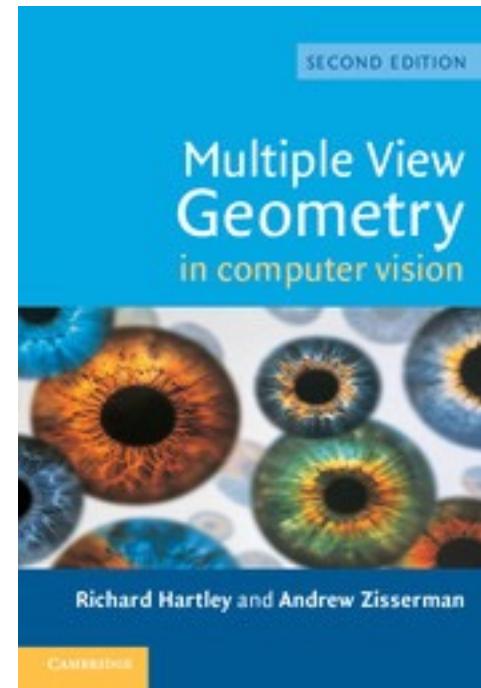
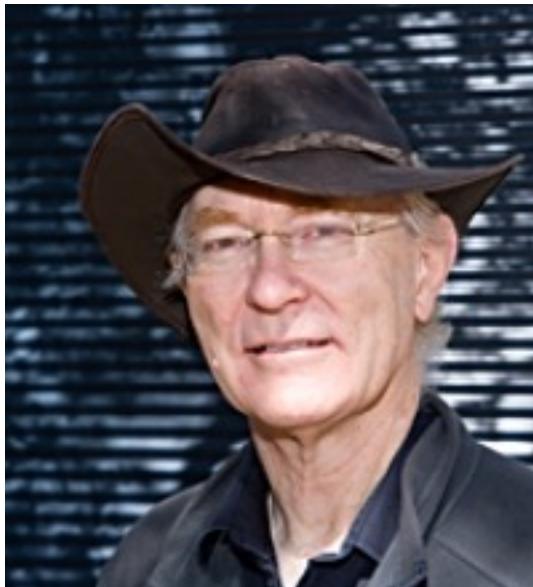


Credit: Facebook (Meta)

17

Reference Textbook for 3D Vision

- Multiple View Geometry in Computer Vision



Richard Hartley (ANU) | Andrew Zisserman (Oxford)

Recommended Reading

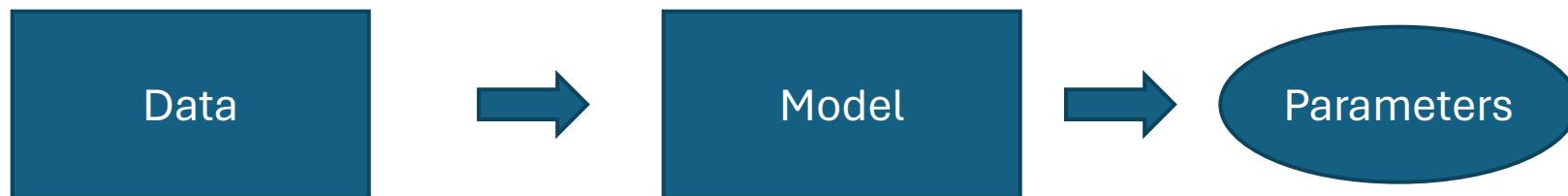
- Ch 6: Camera models
 - Ch 7: Computation of camera matrix
 - Ch 4: DLT algorithm
-
- More in-depth than the lectures
 - Goes beyond what is required for this course

Model Fitting

Line Fitting

Model Fitting in Computer Vision

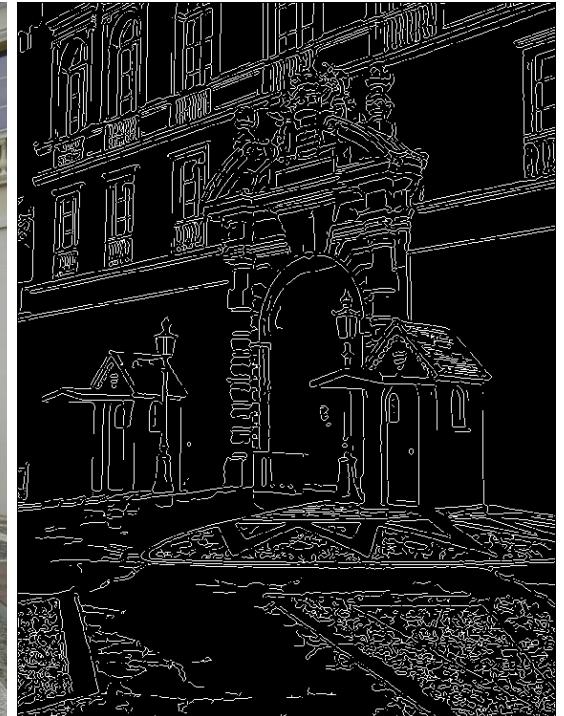
- Model fitting is a fundamental approach in computer vision



- Find **model parameters** that **best fit the data**
 - → **Optimisation** problem
- Model can be as simple as 2D lines, or more complex such as camera transform or 3D articulated object (e.g., skeleton model)
- We will introduce using 2D lines now, cameras later

A Binary Edge Map is Often Insufficient

- Fragmented edges
- Many broken lines
- Noise in edge orientations
- Membership not clear: which points on which line?



Any Feature Here?



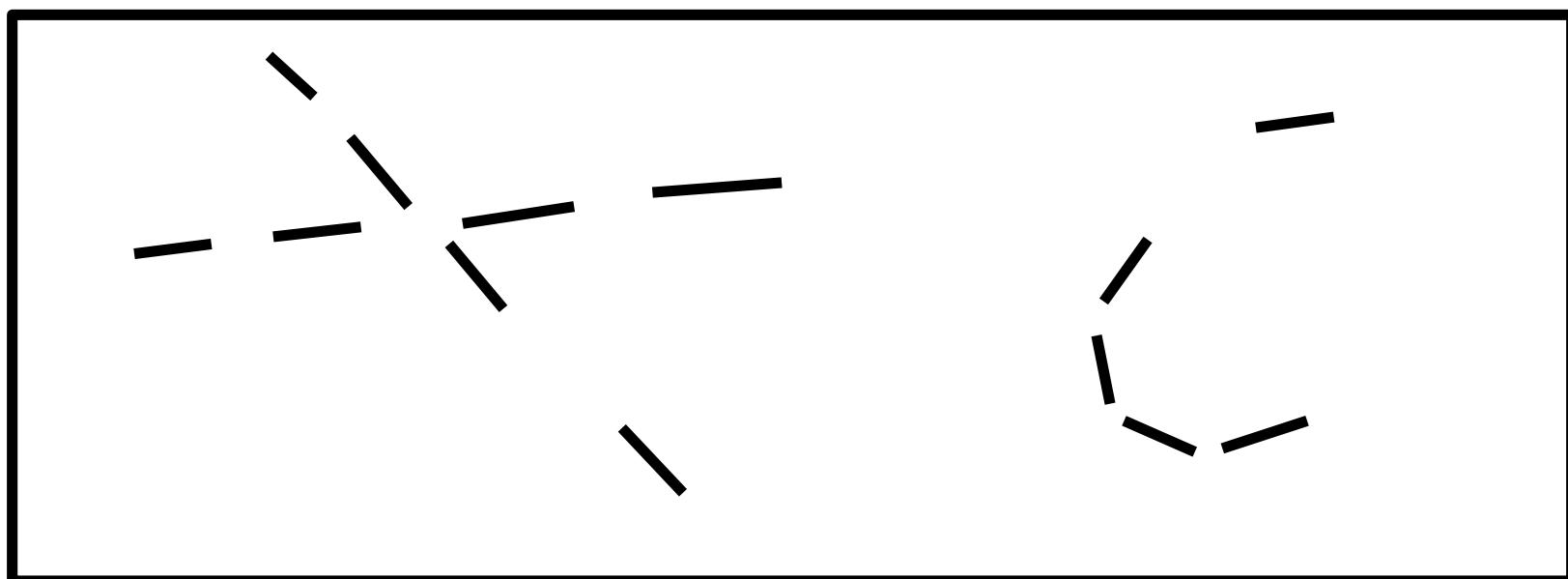
Local versus global

Vanishing Points and Lines



Goal of Line Fitting

- From fragmented edges to straight lines or curves



Goal of Line Fitting

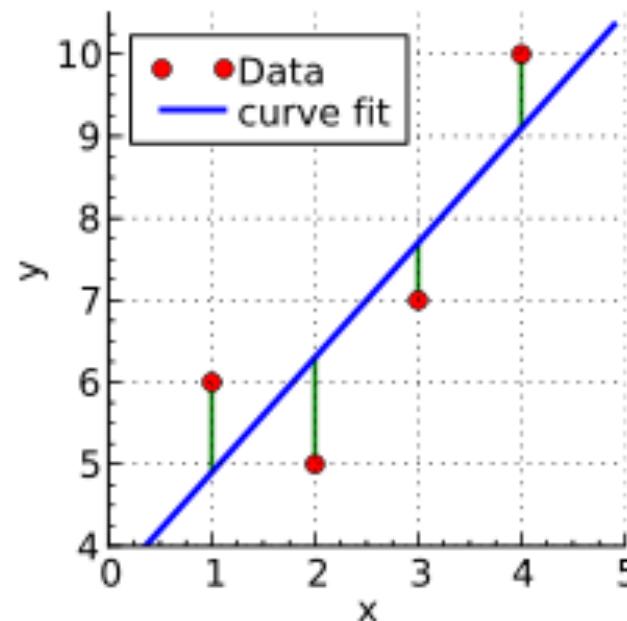


“Fitting” in general context

- Choose a parametric model to represent a set of features
- Membership determination
 - Decide which feature belongs to which model
 - Not a local decision; must look at the full image globally
- Key questions:
 - What is the best parametric model?
 - How many models (e.g., lines) are there?
 - Which point belongs to which line?
 - Remove noise points, wrong edge points

Basic Line Fitting: Least Squares

- Given a set of points that belong to a straight line, find the equation of the line
- Find the **line parameters** that yield **minimal error** in a **least-square sense**



Basic Line Fitting: Least Squares

- Given data: $(x_1, y_1), \dots, (x_n, y_n)$
- Line model: $y_i = mx_i + b$
- Find $\mathbf{p} = (m, b)$ to minimise

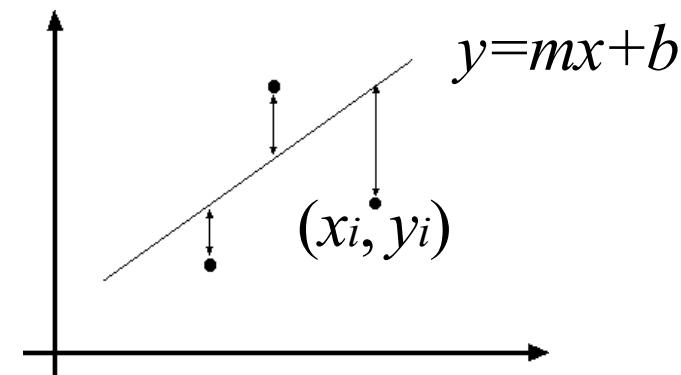
$$e = \sum_{i=1}^n (y_i - mx_i - b)^2 \quad \text{SSE loss}$$

$$e = \sum_{i=1}^n \left(y_i - [x_i \ 1] \begin{bmatrix} m \\ b \end{bmatrix} \right)^2$$

$$e = \left\| \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \right\|^2$$

$$e = \|\mathbf{y} - A\mathbf{p}\|^2$$

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



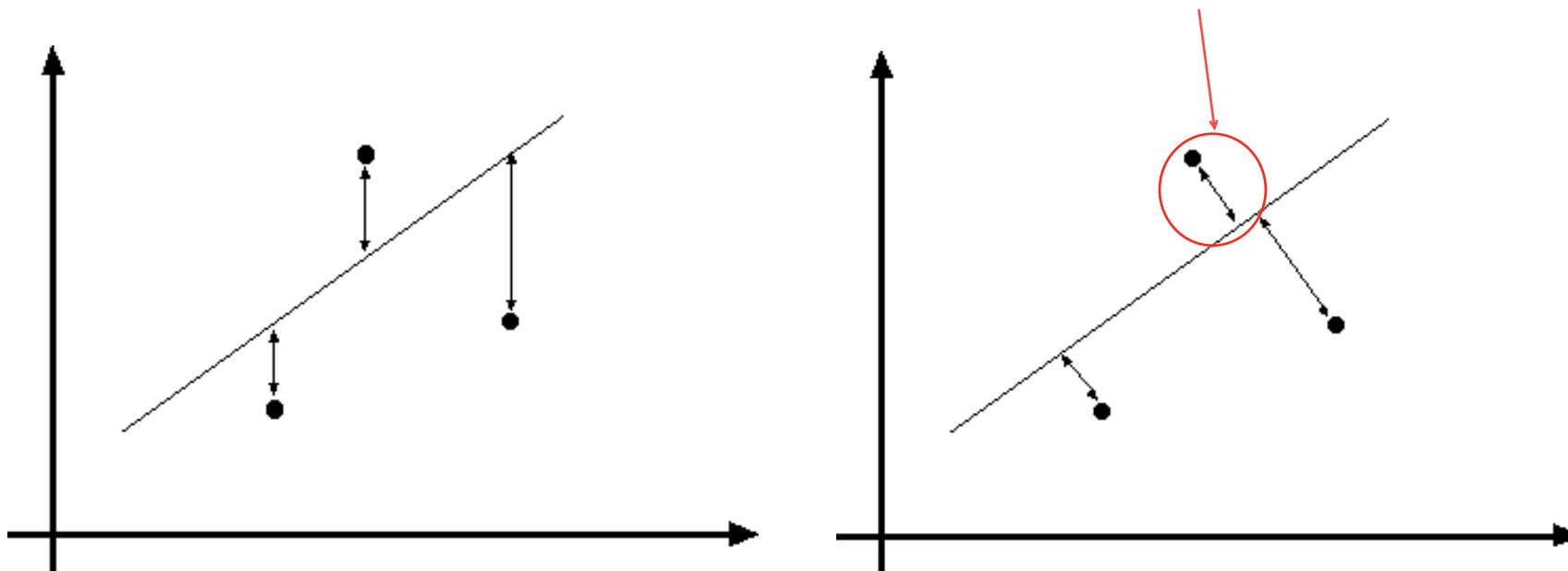
$$\frac{de}{d\mathbf{p}} = 2\mathbf{A}^T \mathbf{A} \mathbf{p} - 2\mathbf{A}^T \mathbf{y} = 0$$

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

closed-form solution

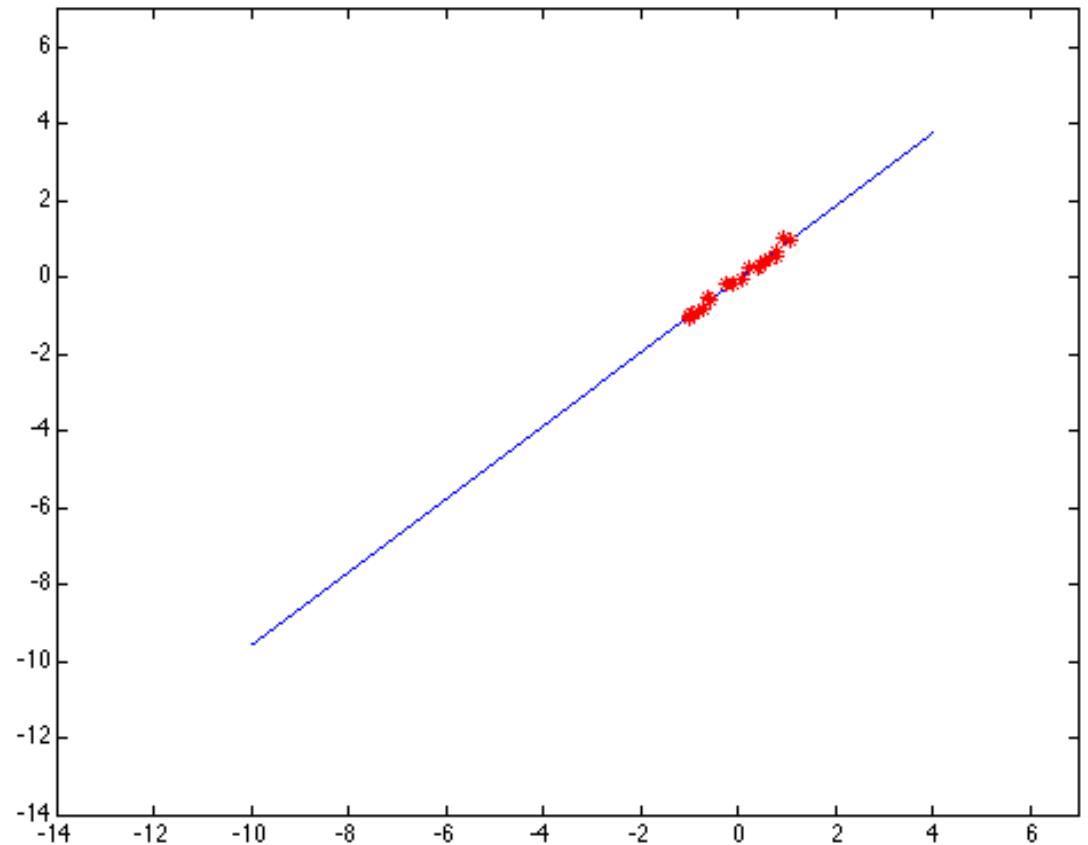
Basic Line Fitting: Issues

- Fitting performance is affected by the slope
- Fails completely for vertical lines when labels are totally independent from features
- Modified least squares using a perpendicular distance



Least Squares: Robustness to Noise

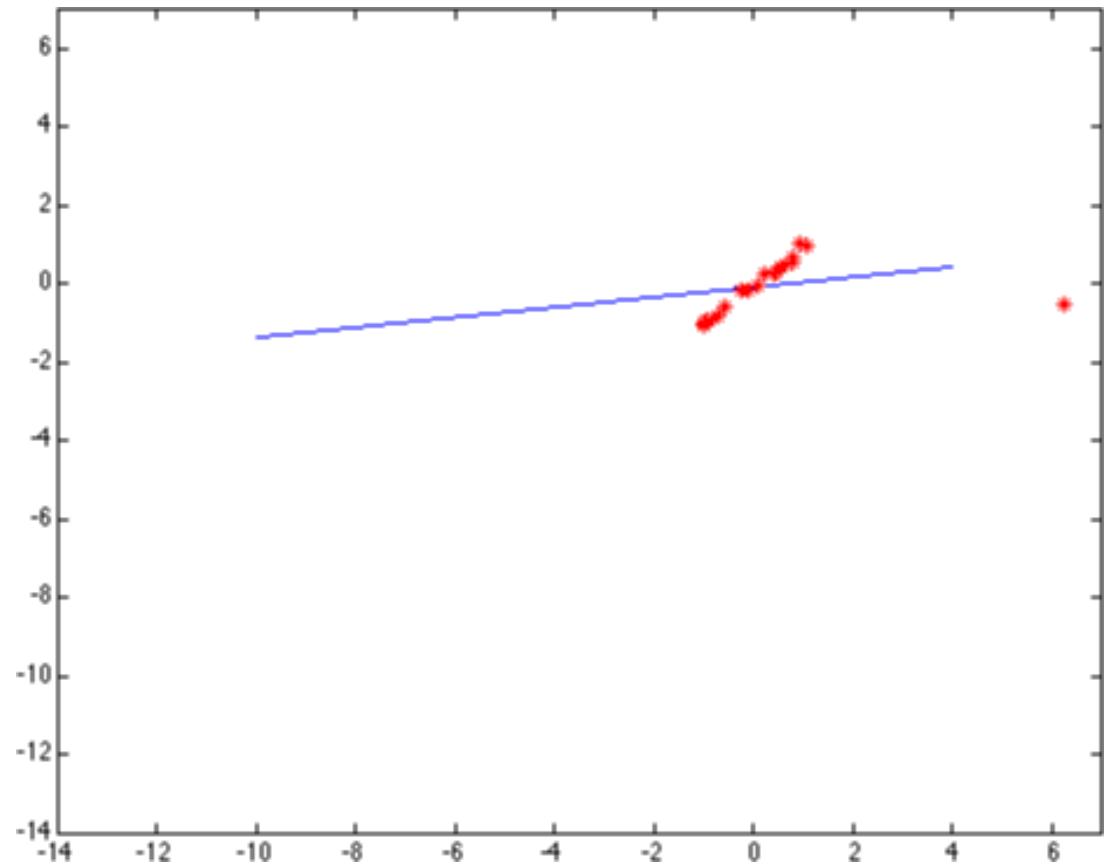
- Least squares fit to the red points



Least Squares: Robustness to Noise

- Least squares fit with an outlier
- Problem: squared error heavily penalises outliers

outliers contribute more in the total loss,
params more fitting with outliers



Robust M-estimators

- General approach: find model parameters θ that minimise

$$\sum_i \rho(r_i(x_i, \theta); \sigma)$$

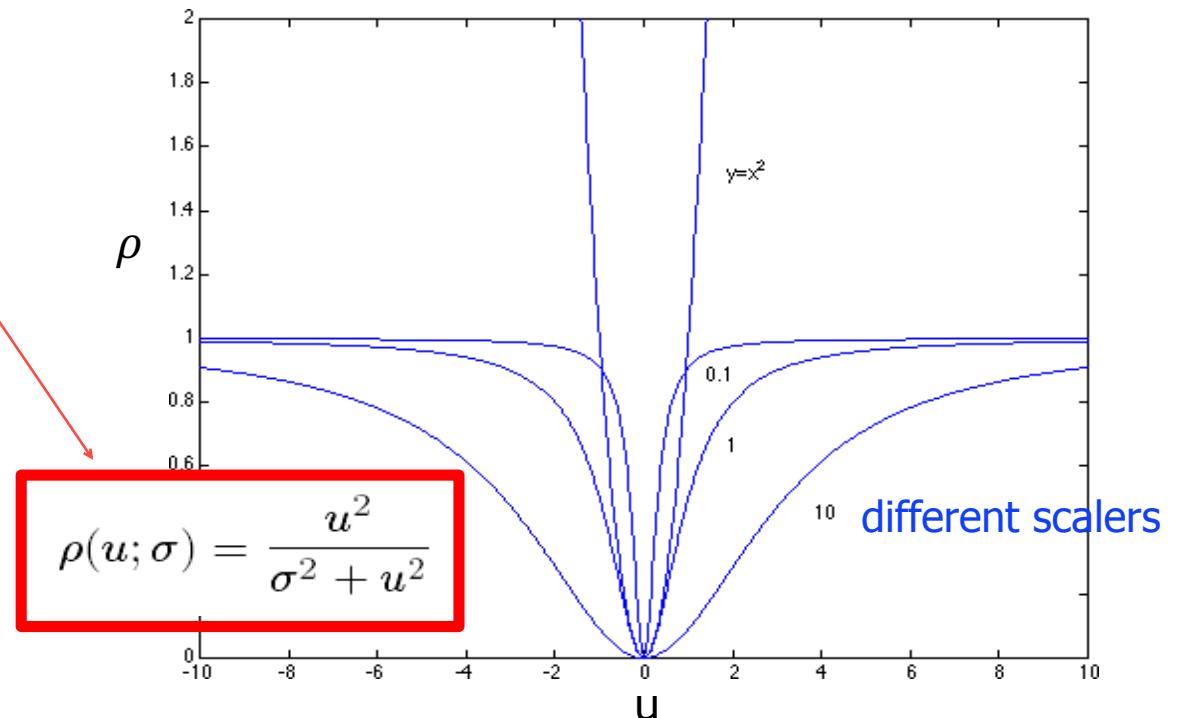
residual; scalar

where:

- r_i is the residual/loss of the i th point w.r.t. model parameters θ
- ρ is a robust function with scale parameter σ

outliers contribute large residuals, where robust m-estimators can moderate its influence

- Robust function behaves like squared distance for small values of the residual but saturates for larger values



Robust M-estimators

- General approach: find model parameters θ that minimise

$$\sum_i \rho(r_i(x_i, \theta); \sigma)$$

where:

- r_i is the residual/loss of the i th point w.r.t. model parameters θ
- ρ is a robust function with scale parameter σ

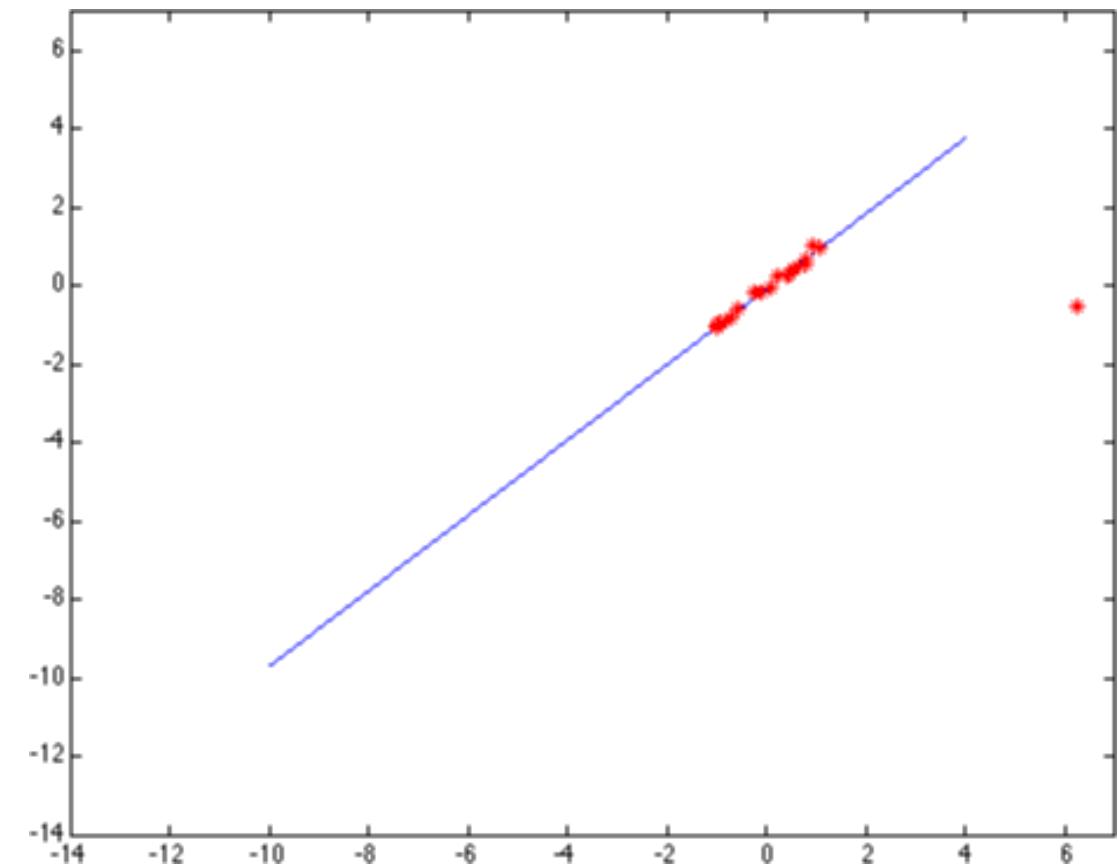
- Robust fitting is a **nonlinear optimisation problem** that must be solved iteratively
- Least squares solution** can be used for **initialisation**
- Scale** of the robust function should be **chosen carefully**
 - You need to know something about the **noise!**

the larger the scalar, the stronger its moderation effect on outliers

Choosing the Scale: Just Right

only limit the influence of outliers

- The effect of the outlier is minimised

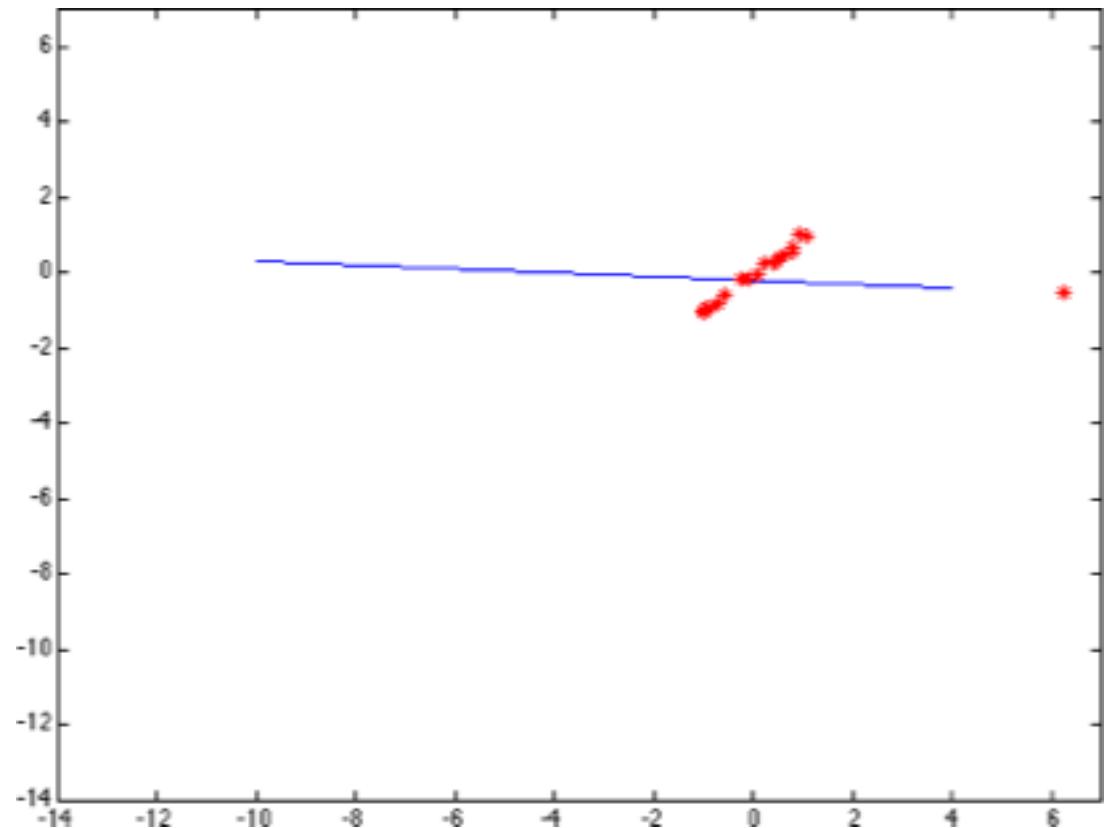
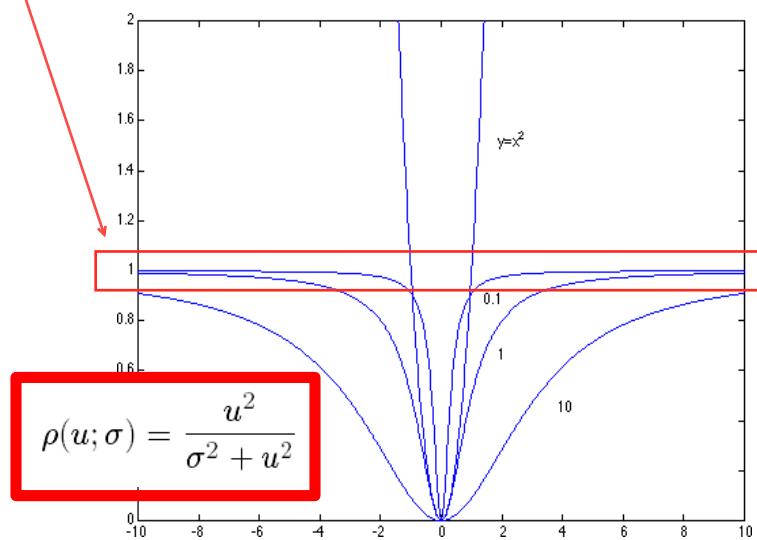


Choosing the Scale: Too Small

every data point contributes the same to the loss function

close to 1

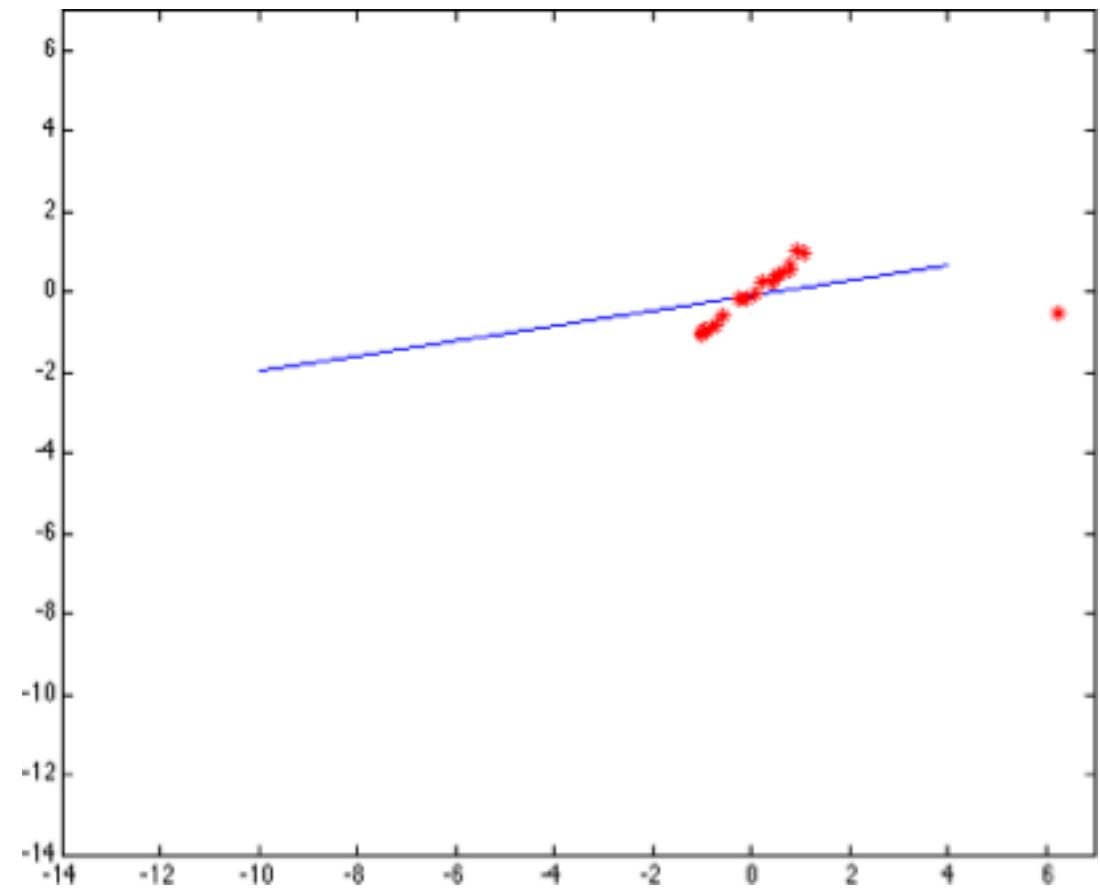
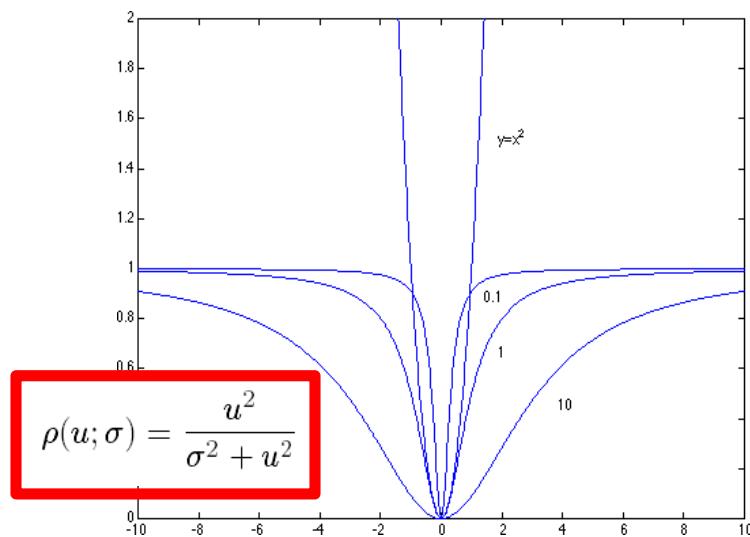
- The error value is almost the same for every point and the fit is very poor



Choosing the Scale: Too Large

outliers contribute more loss

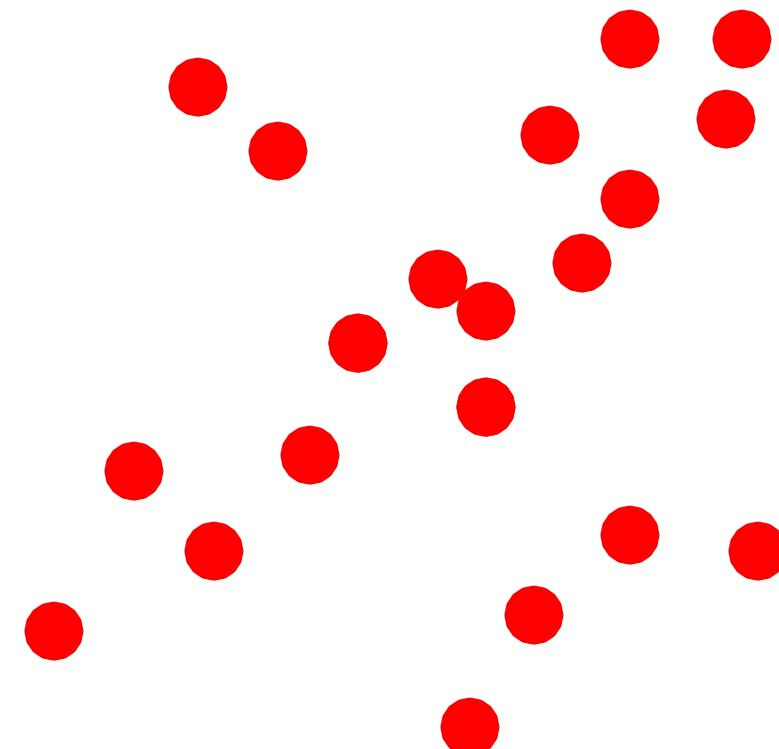
- Behaves much the same as least squares



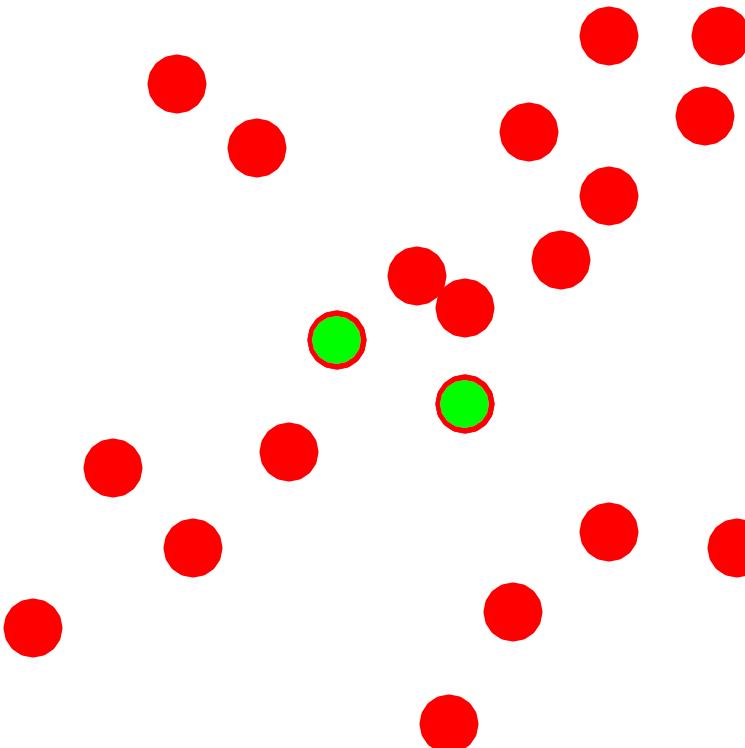
RANSAC

RANSAC: RANdom SAmple Consensus

- Fischler & Bolles 1981
- Basic idea (for line fitting):
 1. Randomly select two samples (minimal number for a line)
 2. Check how many other samples fall close to the line (within a distance threshold)
 3. Choose the line which has the largest number (“consensus”)



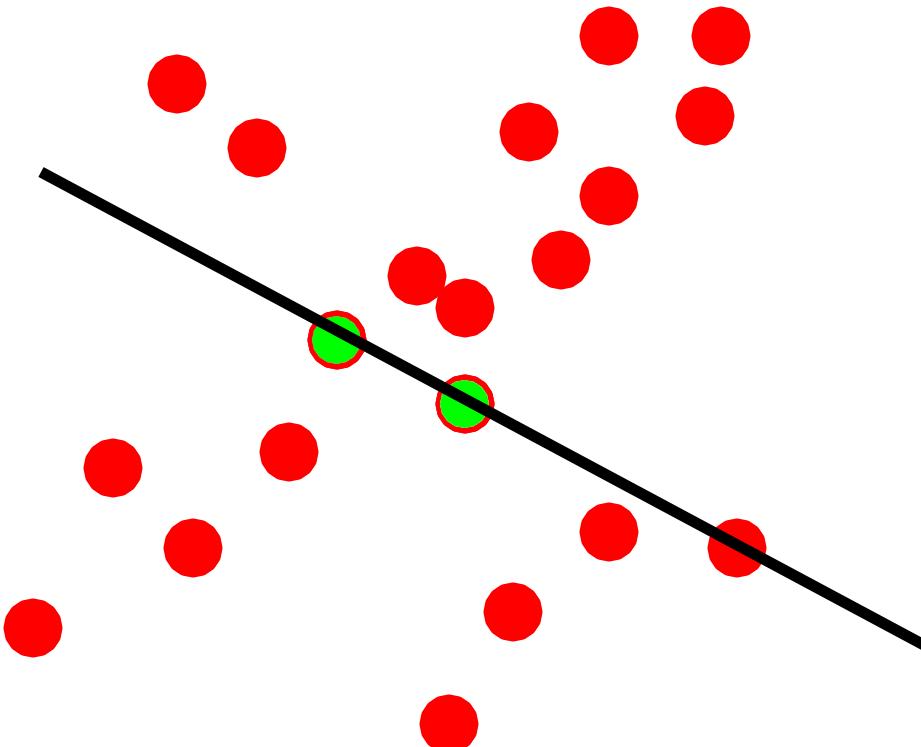
RANSAC



Algorithm:

1. **Sample** (randomly) the minimum number of points required to fit the model ($n=2$)
 2. **Solve** for model parameters
 3. **Score** by the fraction of inliers within a preset threshold
- Repeat** 1–3 until the best model is found

RANSAC



Algorithm:

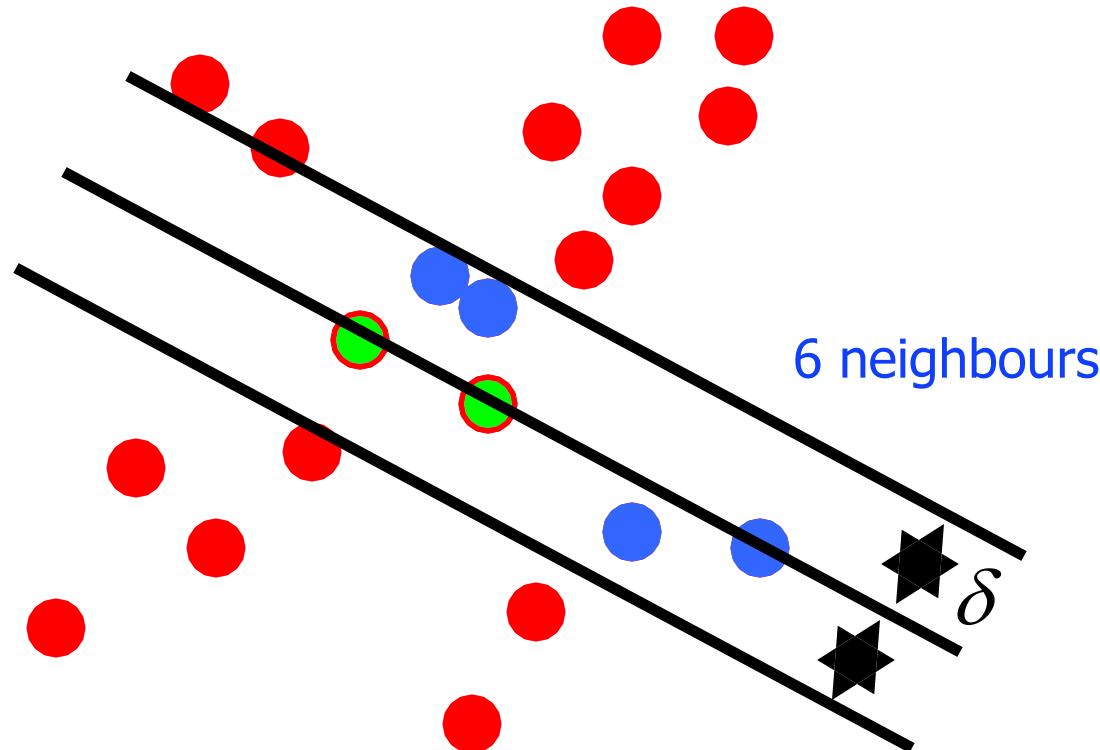
1. **Sample** (randomly) the minimum number of points required to fit the model ($n=2$)
 2. **Solve** for model parameters
 3. **Score** by the fraction of inliers within a preset threshold
- Repeat** 1–3 until the best model is found

RANSAC

$$N_I = 6$$

Algorithm:

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



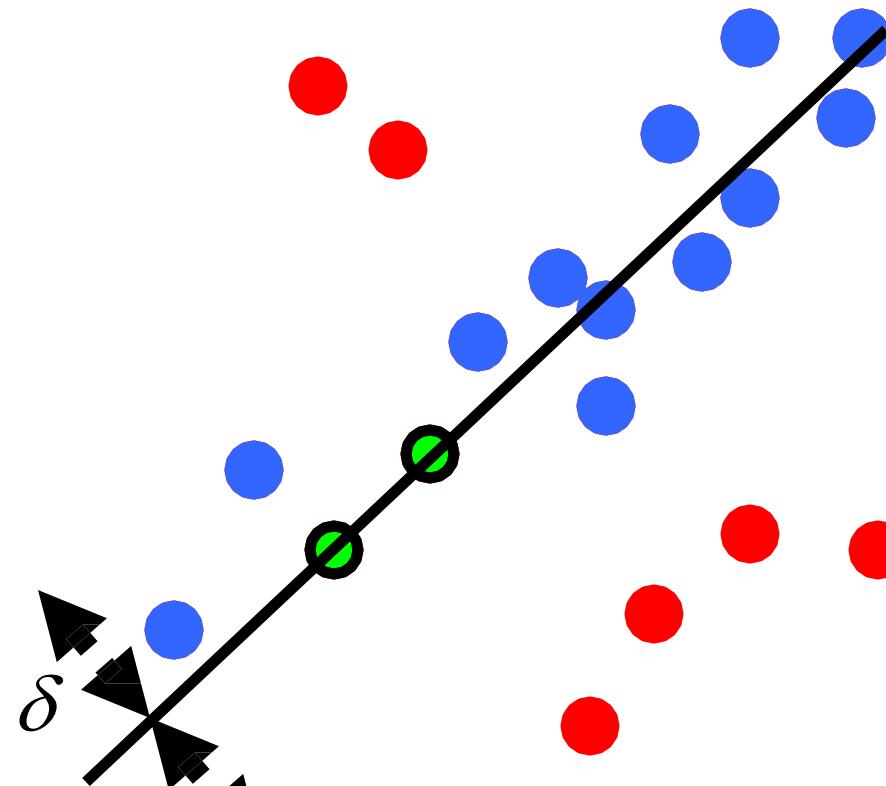
Repeat 1–3 until the best model is found

RANSAC

$$N_I = 14$$

Algorithm:

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



Repeat 1–3 until the best model is found

RANSAC: How Many Samples to Choose?

- e : probability that a point is an outlier
 s : number of points in a sample
 N : number of samples (we want to compute this)
 p : desired probability that we get a good sample

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

- Probability of choosing s inliers (sample only contains inliers)
all the neighbours are not outliers

RANSAC: How Many Samples to Choose?

- e : probability that a point is an outlier ($1 - e$): prob of a non-outlier
 s : number of points in a sample num of neighbours
 N : number of samples (we want to compute this)
 p : desired probability that we get a good sample

$$1 - (1 - (1 - e)^s)^N = p$$

- Probability that one or more points in the sample were outliers (sample is contaminated)

RANSAC: How Many Samples to Choose?

- e : probability that a point is an outlier
 s : number of points in a sample
 N : number of samples (we want to compute this)
 p : desired probability that we get a good sample

$$1 - \boxed{(1 - (1 - e)^s)^N} = p$$

- Probability that all the N samples were contaminated as at least one outliers

RANSAC: How Many Samples to Choose?

- e : probability that a point is an outlier
 s : number of points in a sample
 N : number of samples (we want to compute this)
 p : desired probability that we get a good sample

$$1 - (1 - (1 - e)^s)^N = p$$

- Probability that at least one sample is good
doesn't contain any outlier

RANSAC: How Many Samples to Choose?

- e : probability that a point is an outlier $e = 0.5$
- s : number of points in a sample $s = 2$
- N : number of samples (we want to compute this) $N = 17$
- p : desired probability that we get a good sample $p = 0.99$

$$1 - (1 - (1 - e)^s)^N = p$$

$$\therefore (1 - (1 - e)^s)^N = 1 - p$$

$$\therefore N \log(1 - (1 - e)^s) = \log(1 - p)$$

$$\therefore N = \frac{\log(1 - p)}{\log(1 - (1 - e)^s)}$$

0.99
0.5

$$= 16.008 \rightarrow 17$$

- For a 99% probability of obtaining a good model, with a 50% outlier probability and s=2 samples per model

Sample size	Proportion of outliers						
	5%	10%	20%	25%	30%	40%	50%
N							e
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

N

Figure Credit: Hartley & Zisserman

RANSAC: Pros and Cons

- Pros:
 - Robust to outliers
 - Applicable for large number of parameters
- Cons:
 - Computational time grows quickly with the fraction of outliers and the number of parameters each sample needs to solve its params
 - Not good for getting multiple line fits
 - Can rapidly find a good model, not necessarily a great model

the larger the fraction of outliers,
the more times is required to get a good sample
- Common applications:
 - Computing a rotation and translation between two images (will be used later for fundamental matrix in 2- view geometry)

Hough Transform

Fitting Multiple Lines Using a Hough Transform

- Given a **binary edge image**, find the **lines (or curves)** that **explain the data points** best in the **parameter space**
- This parameter space is called a **Hough space**

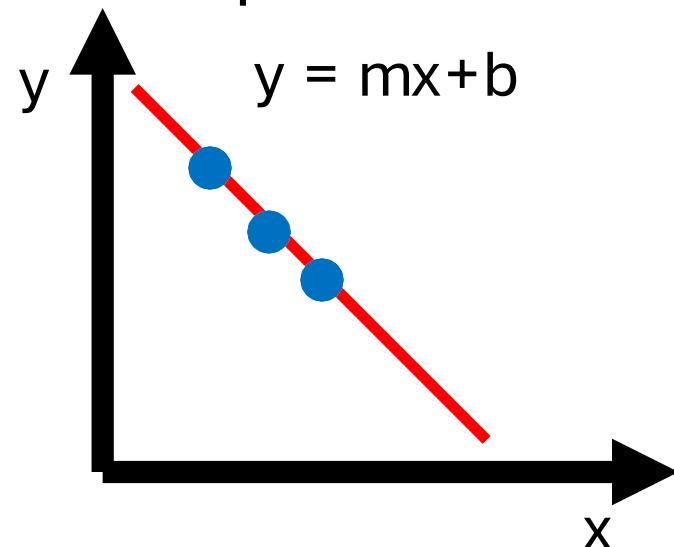
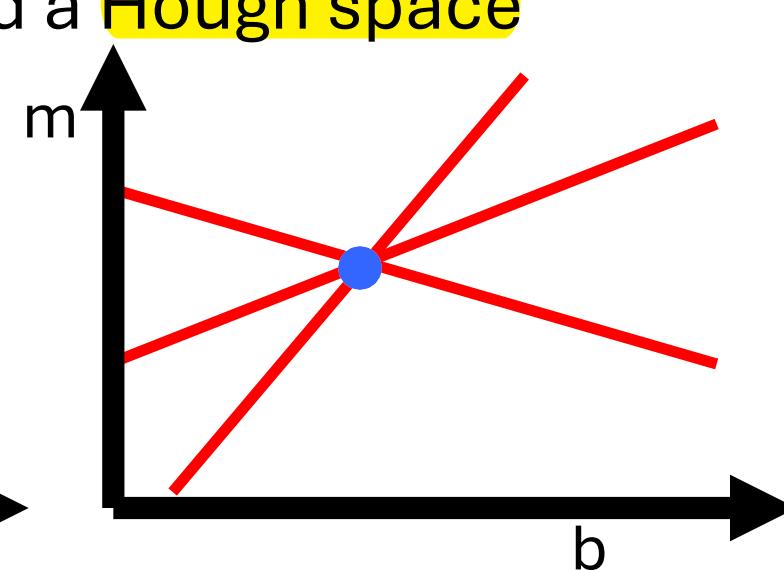


Image space

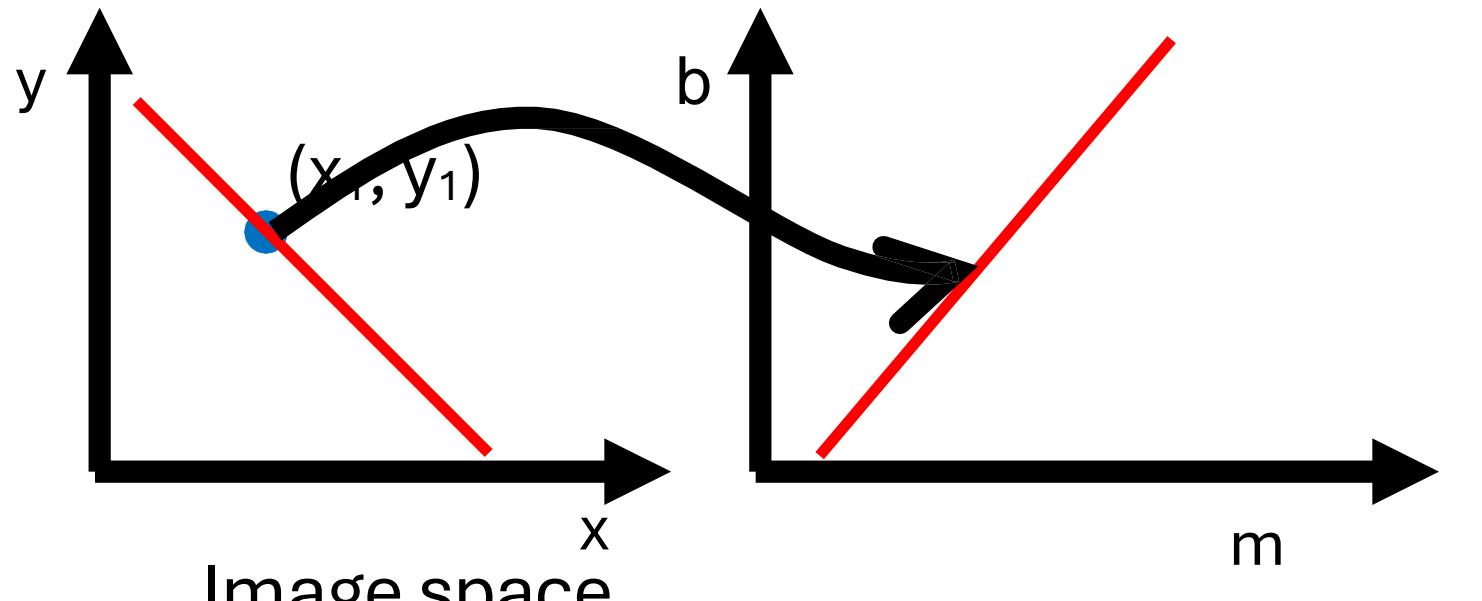


Parameter (Hough) space

A point (x_1, y_1) is mapped to a line in Hough space

$$y_1 = mx_1 + b$$

$b = -x_1 m + y_1$ (rewrite the equation in terms of m and b)



Parameter (Hough) space

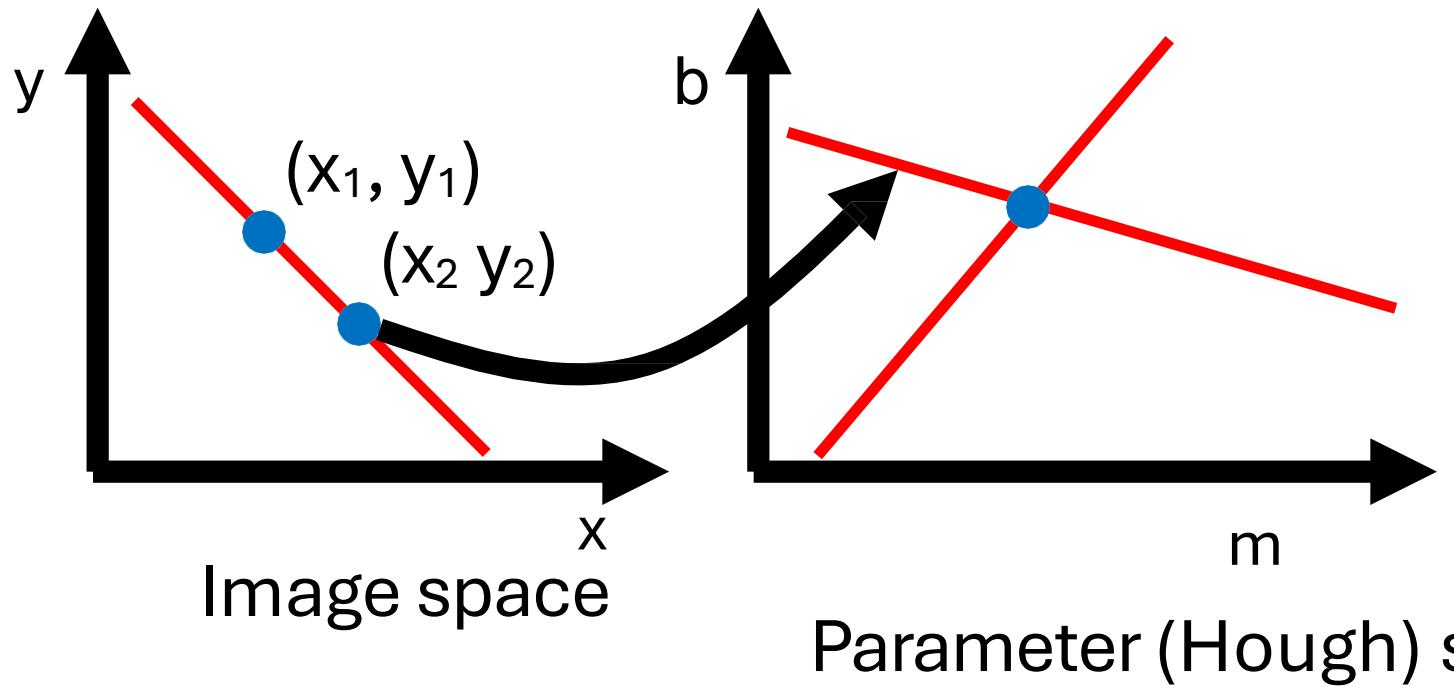
A point (x_2, y_2) is mapped to a line in Hough space

$$y_2 = mx_2 + b$$

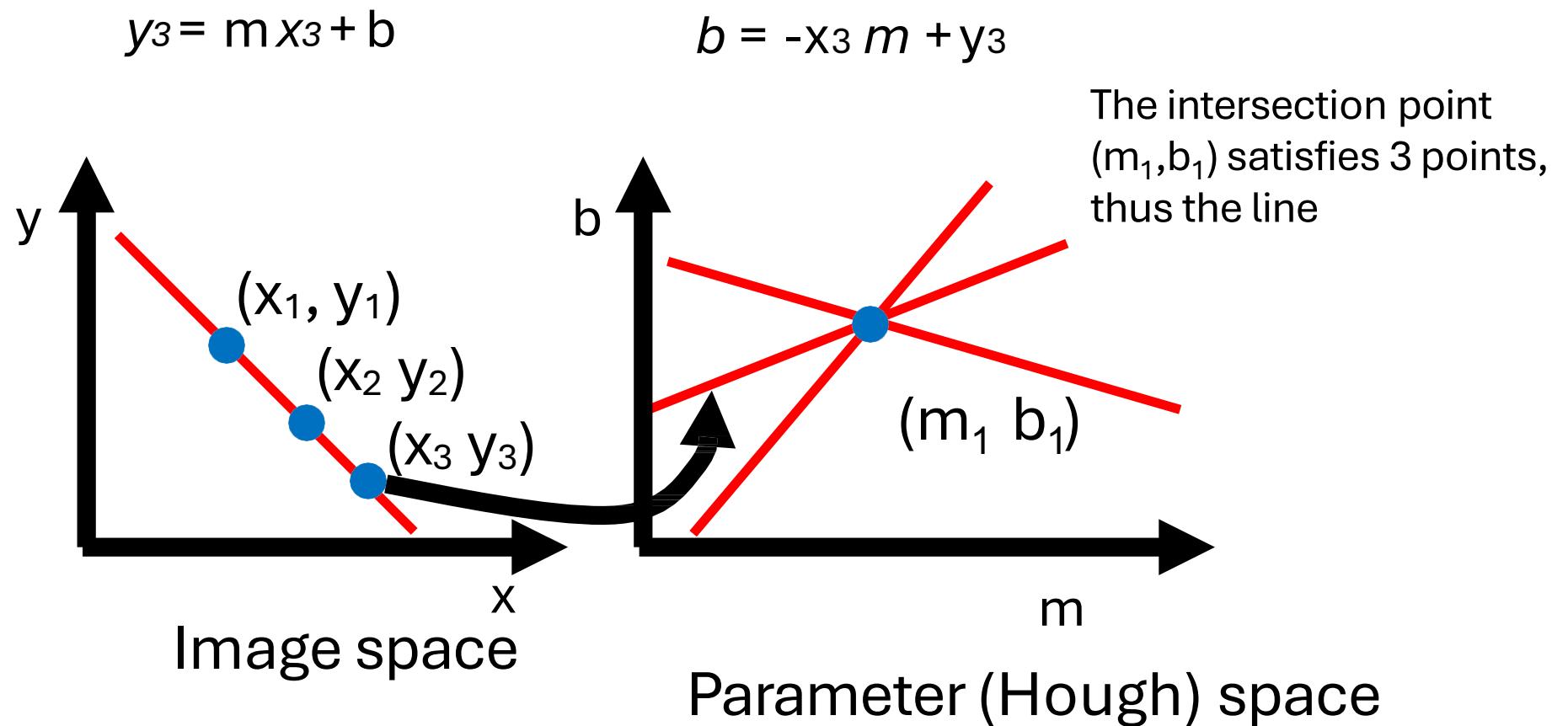
slope constant
 $b = -x_2 m + y_2$

The intersection point satisfies both (x_1, y_1) and (x_2, y_2) , thus the line

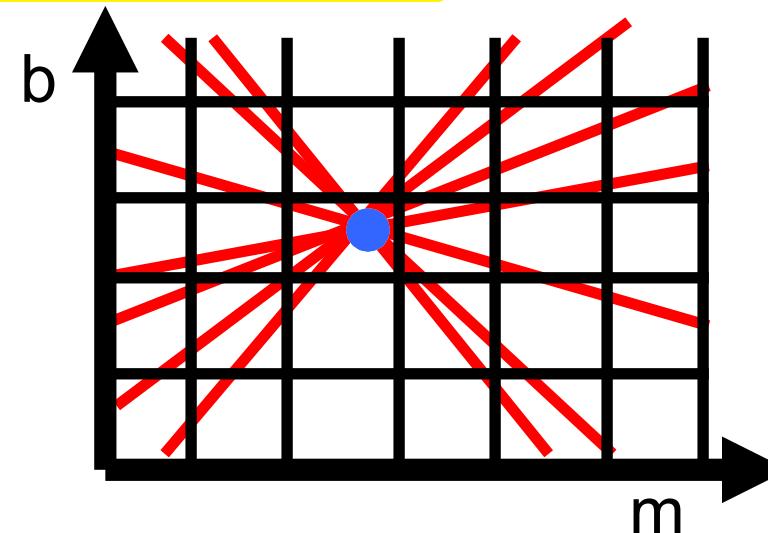
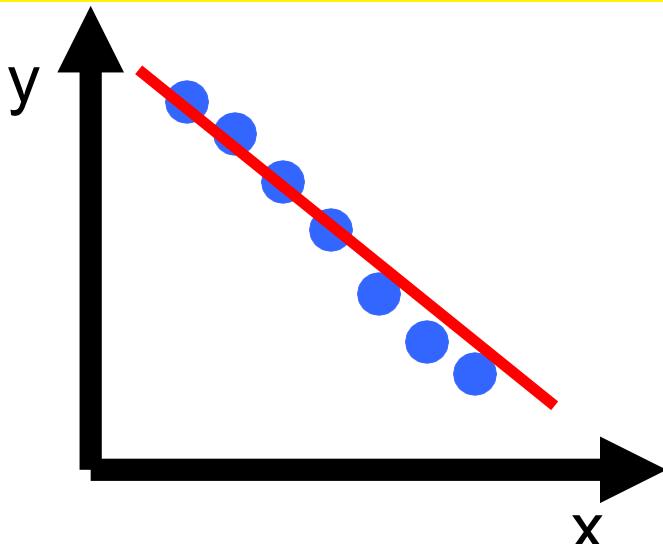
share the same params



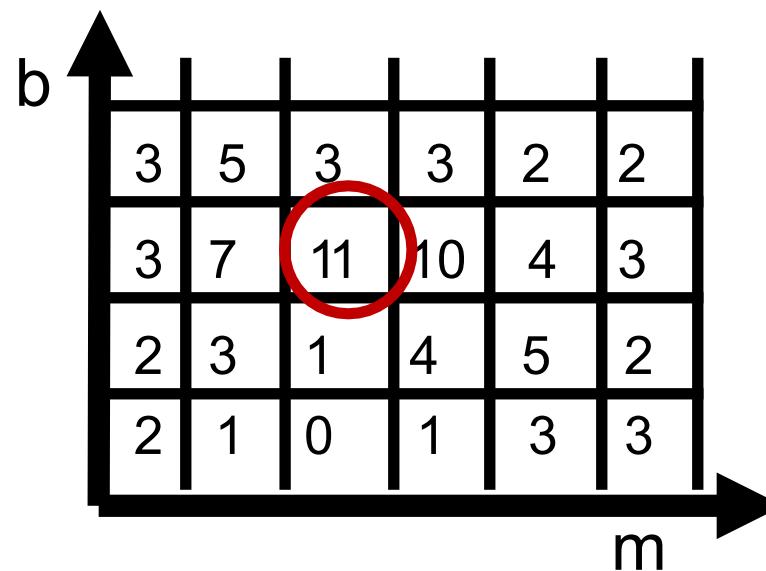
A point (x_3, y_3) is mapped to a line in Hough space



Hough Transform: Accumulator



- For each pixel, draw a line in the discretised Hough space, assigning a value of one to all the discretised positions it passes through
- Process all image pixels



Hough Transform

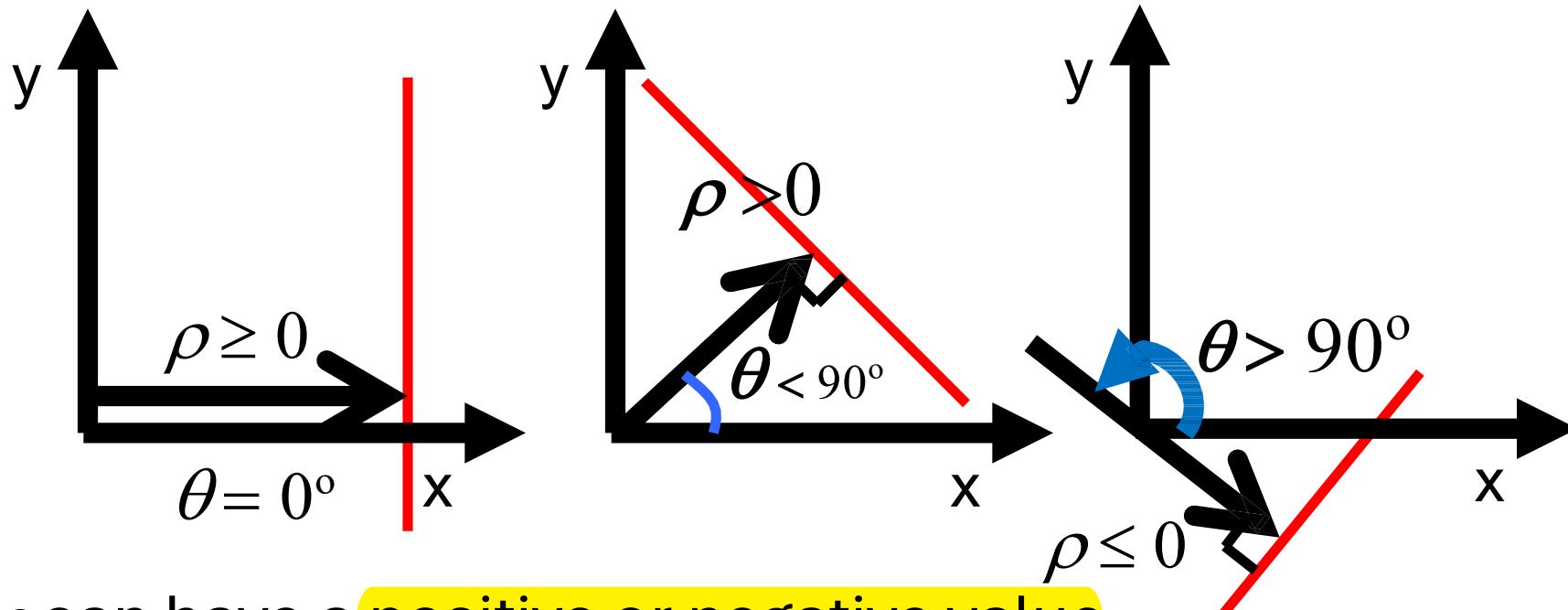
- Can detect multiple lines in an image
 - from multiple local maxima
- Can easily be extended for circles and ellipses
- Computationally efficient
- Problem: (m, b) are unbounded
 - E.g., the slope parameter m can have an infinite value
 x -axis

Hough Transform: Polar Form

the radius and polar angle are limited

- Use a **polar representation** for the **parameter space**

$$\bullet x \cos \theta + y \sin \theta = \rho \rightarrow y = -\frac{\cos \theta}{\sin \theta} x + \frac{\rho}{\sin \theta} \quad (0 \leq \theta \leq 180^\circ)$$



- Note: ρ can have a **positive or negative value**

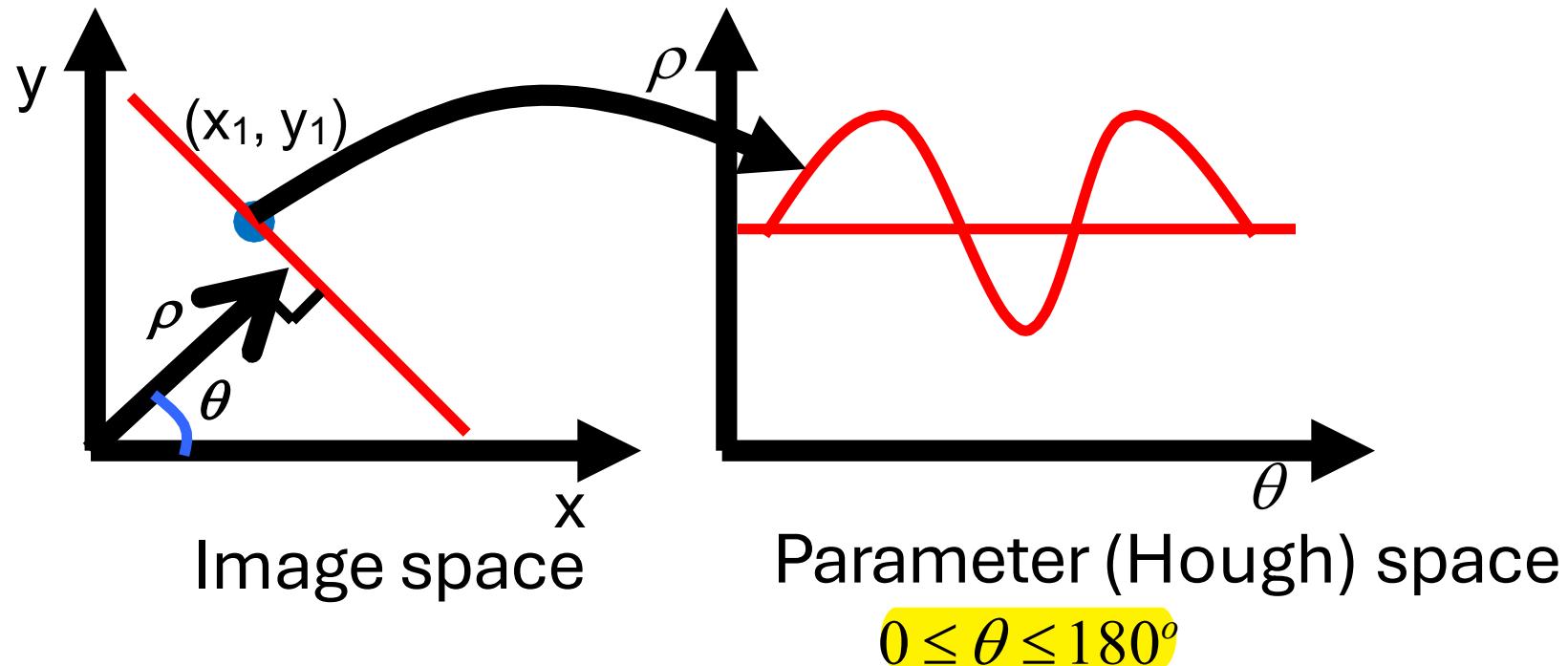
- Given a point (x, y) in Cartesian coordinates:
 - **Radius (r):** Calculate the distance from the origin using the Pythagorean theorem:
$$r = \sqrt{x^2 + y^2}$$
 - **Polar Angle (θ):** Determine the angle the point makes with the positive x-axis using trigonometric functions:
$$\theta = \arctan\left(\frac{y}{x}\right)$$
- Note: The arctan function returns values in the range $(-\pi, \pi)$. Adjustments may be necessary to ensure the angle is in the desired range.

A point (x_1, y_1) is mapped to a sinusoid in the polar parameter space

$$x_1 \cos \theta + y_1 \sin \theta = \rho$$

theta and ro are params for the linear model

$$\begin{aligned}\rho &= x_1 \cos \theta + y_1 \sin \theta \\ &= \alpha_1 \sin(\theta + \beta_1)\end{aligned}$$

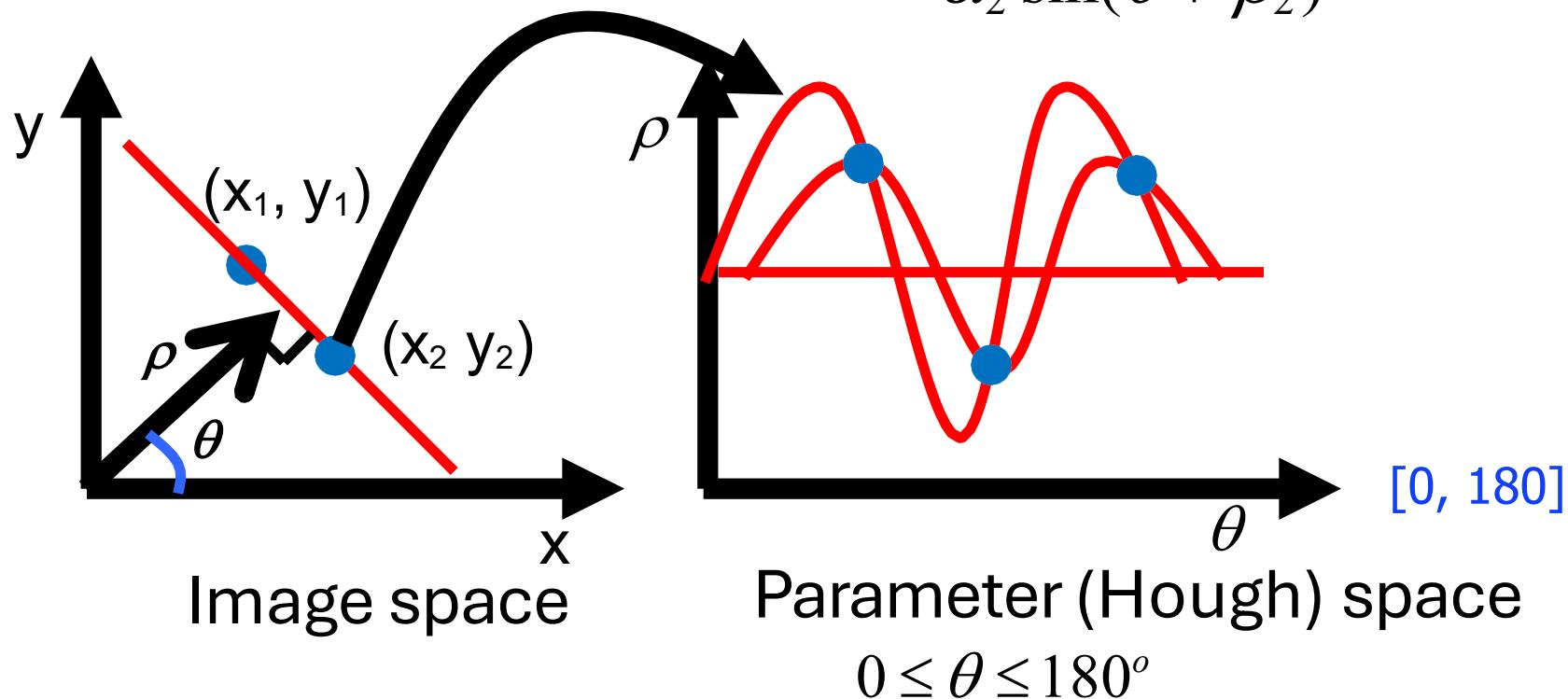


How to map: just divide the right-hand side by $\sqrt{x_1^2 + y_1^2}$ and multiply by the same

A point (x_2, y_2) is mapped to a sinusoid in the polar parameter space

$$x_2 \cos \theta + y_2 \sin \theta = \rho$$

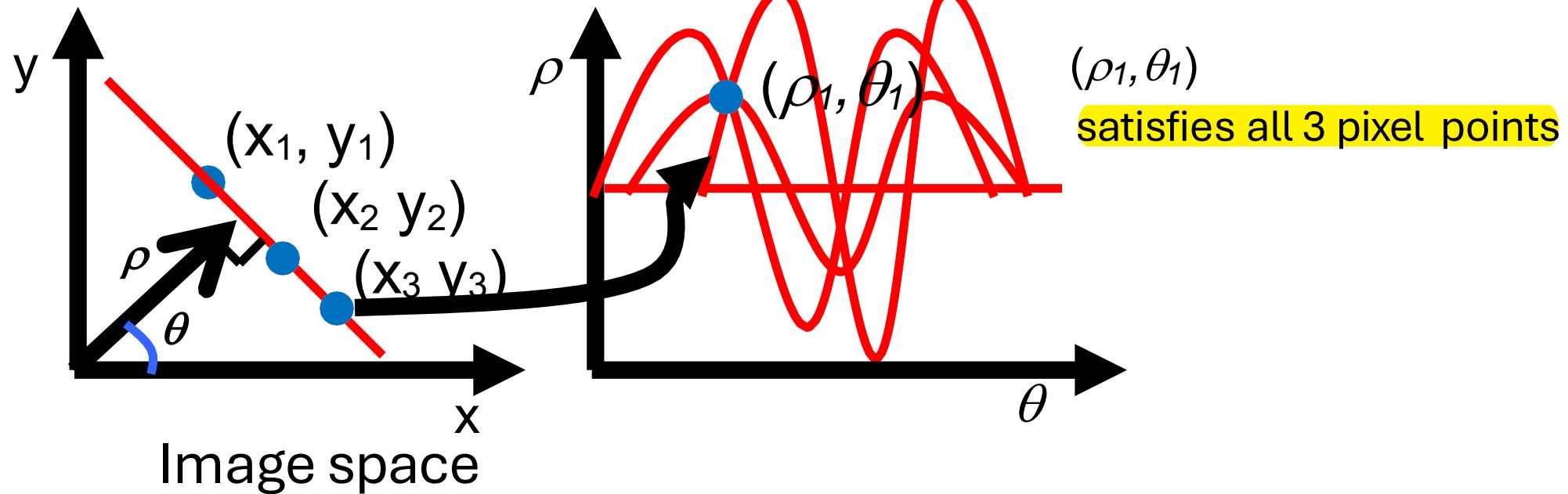
$$\begin{aligned}\rho &= x_2 \cos \theta + y_2 \sin \theta \\ &= \alpha_2 \sin(\theta + \beta_2)\end{aligned}$$



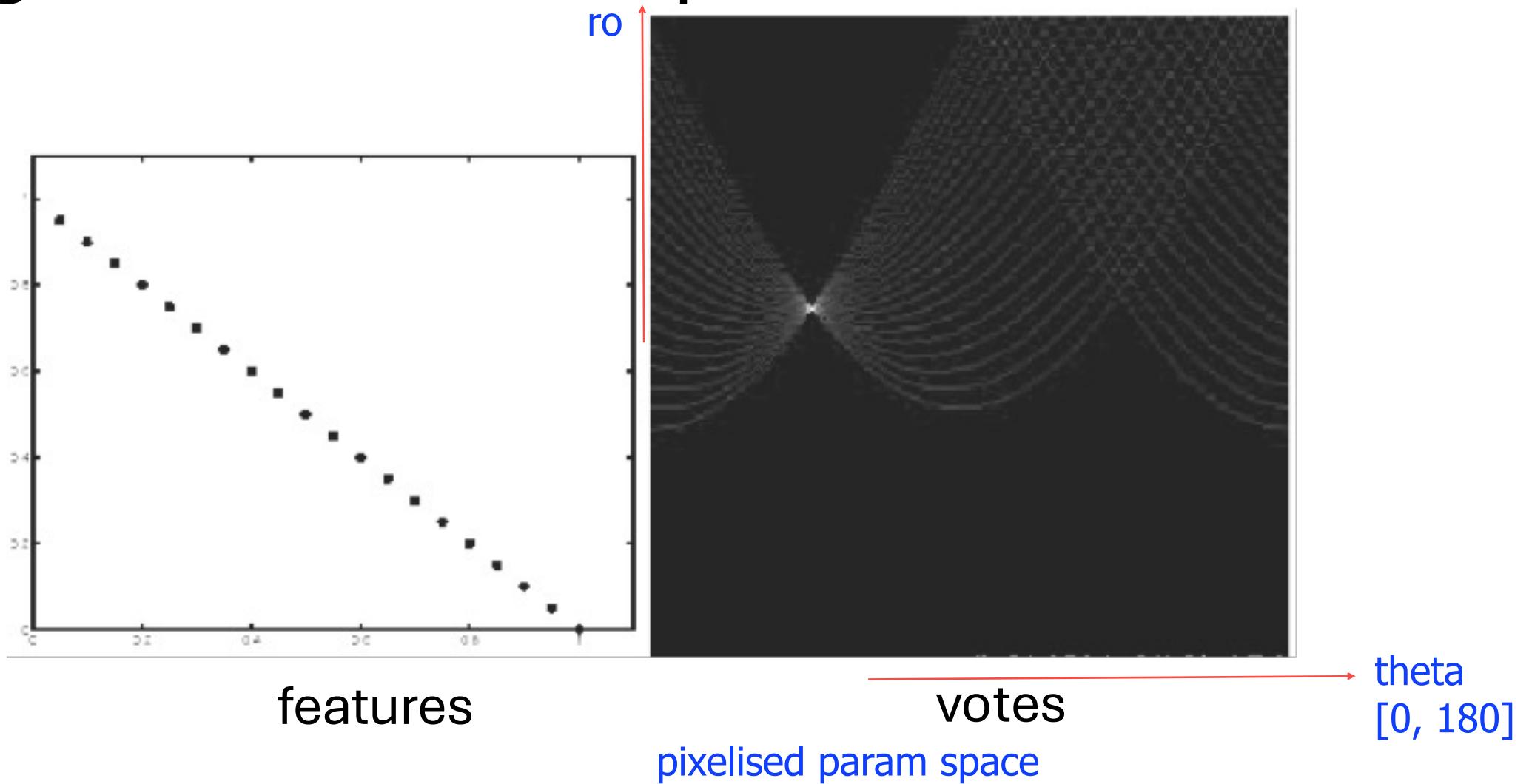
A point (x_3, y_3) is mapped to a sinusoid in the polar parameter space

$$x_3 \cos \theta + y_3 \sin \theta = \rho$$

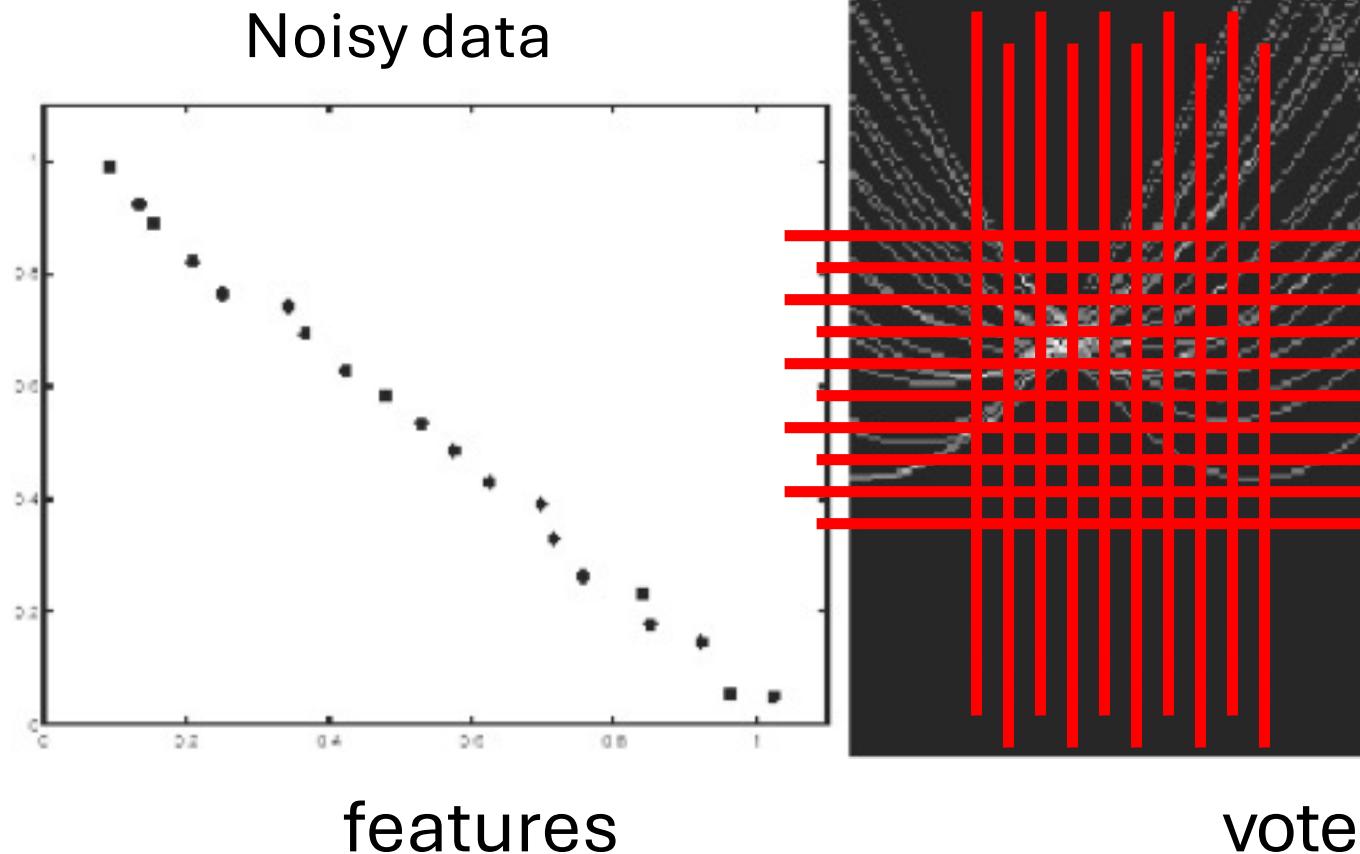
$$\begin{aligned}\rho &= x_3 \cos \theta + y_3 \sin \theta \\ &= \alpha_3 \sin(\theta + \beta_3)\end{aligned}$$



Hough Transform: Example

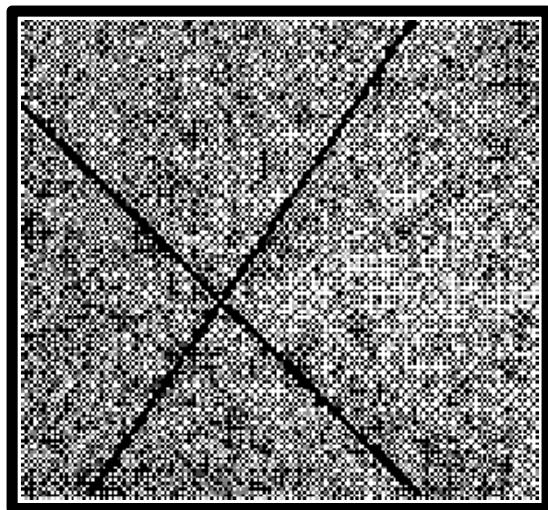


Hough Transform: Example



Issue: the **grid size** needs to be adjusted...

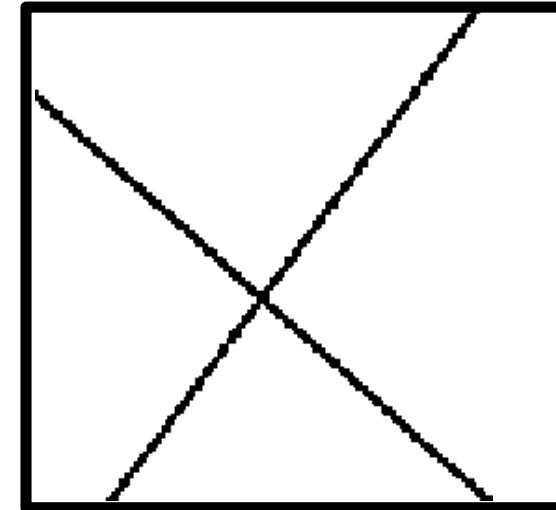
Hough Transform: Example



Image



Edge Detection

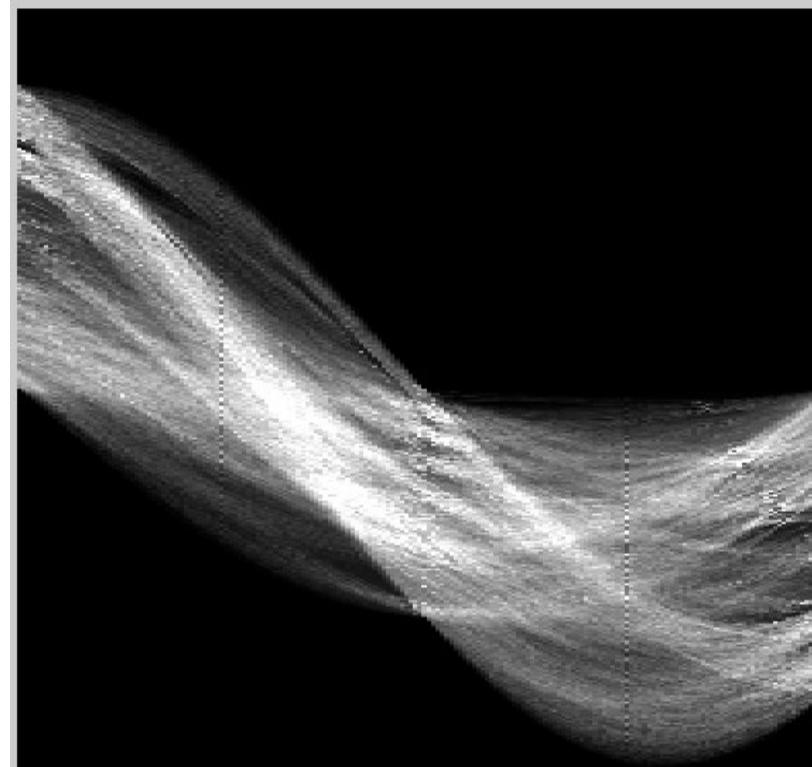


Hough Transform

Hough Transform: 1) Image → Canny Edges

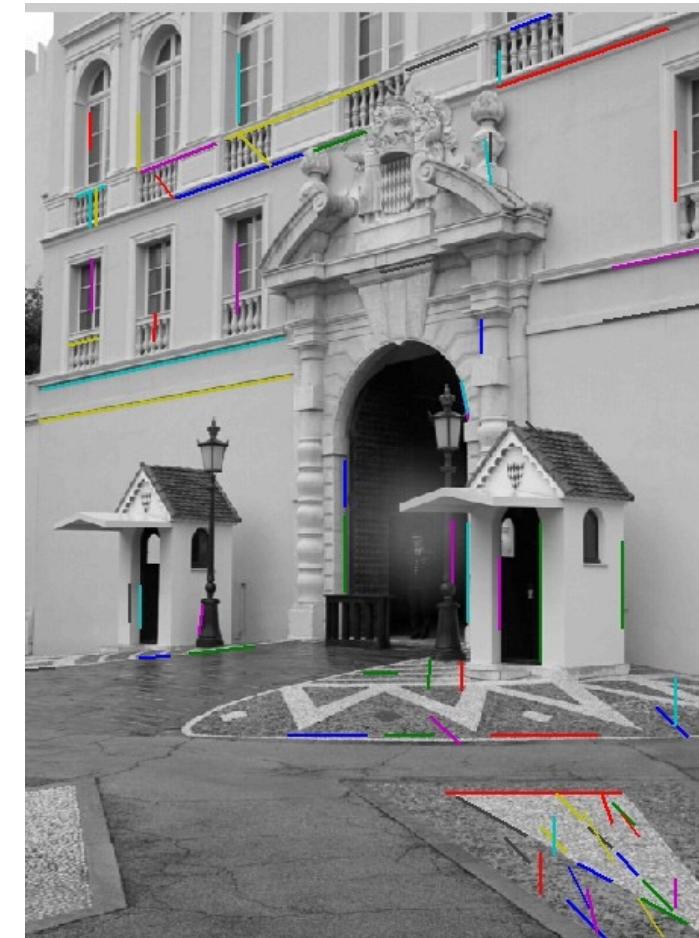
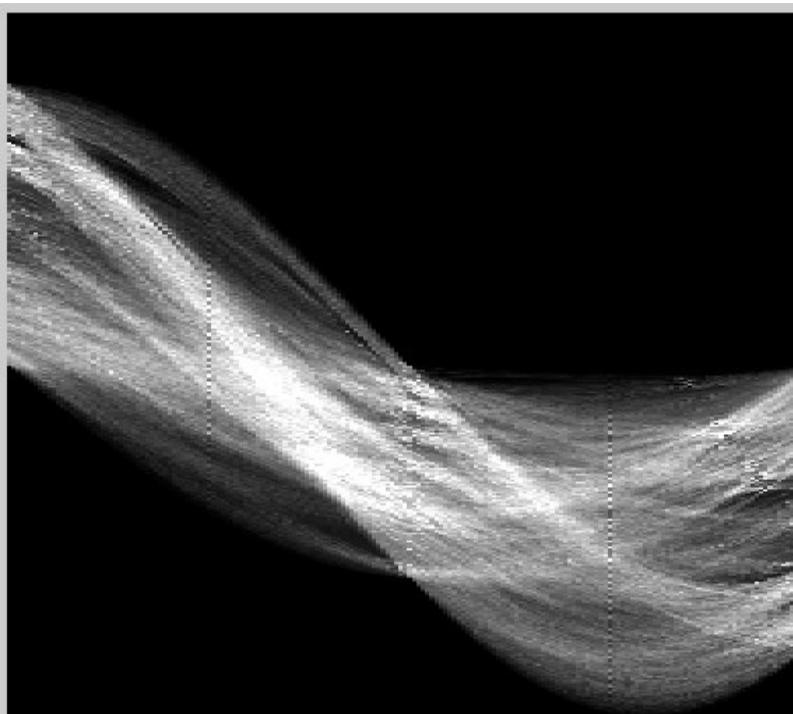


Hough Transform: 2) Canny Edges → Votes

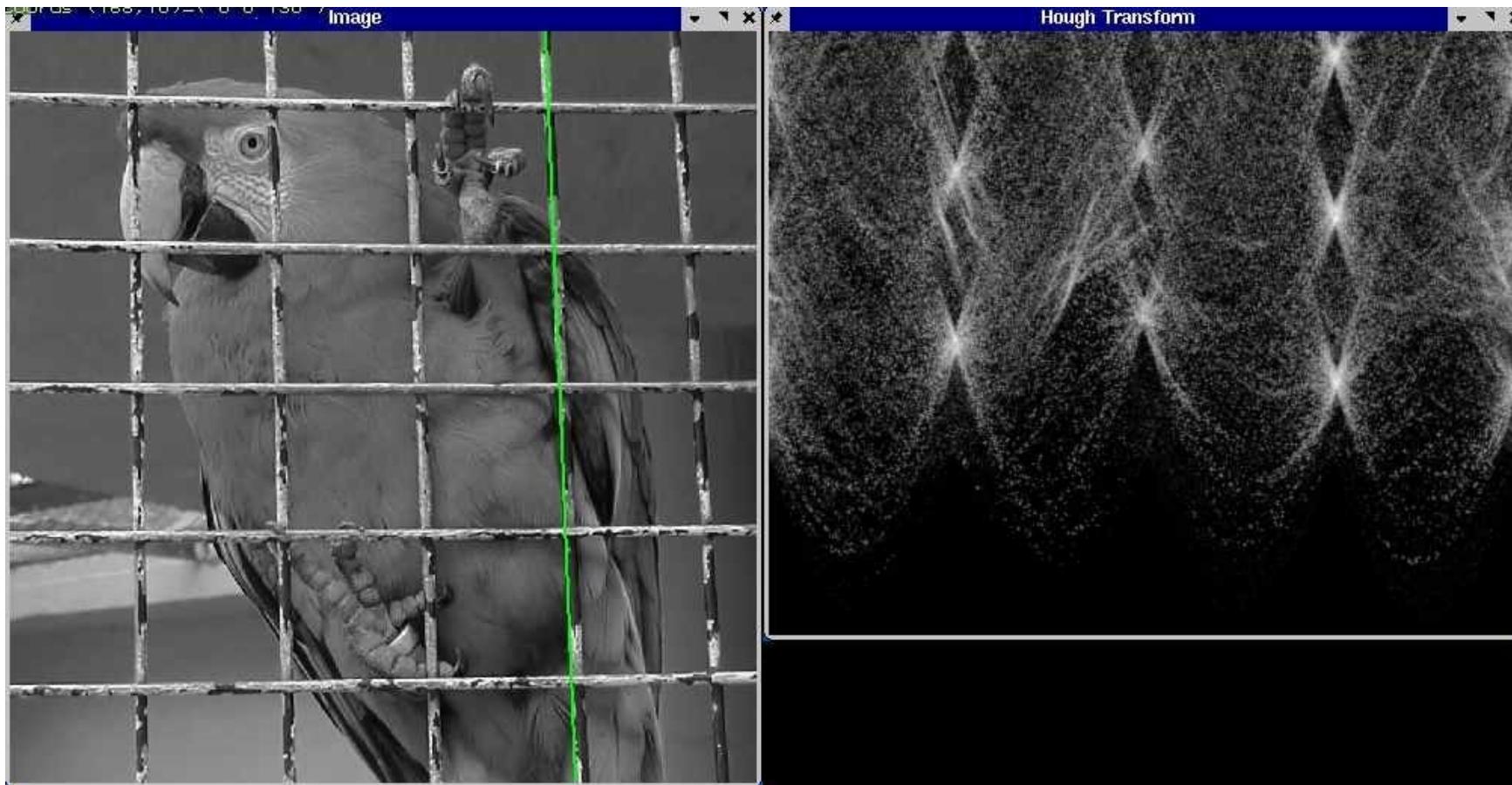


Hough Transform: 3) Votes → Lines

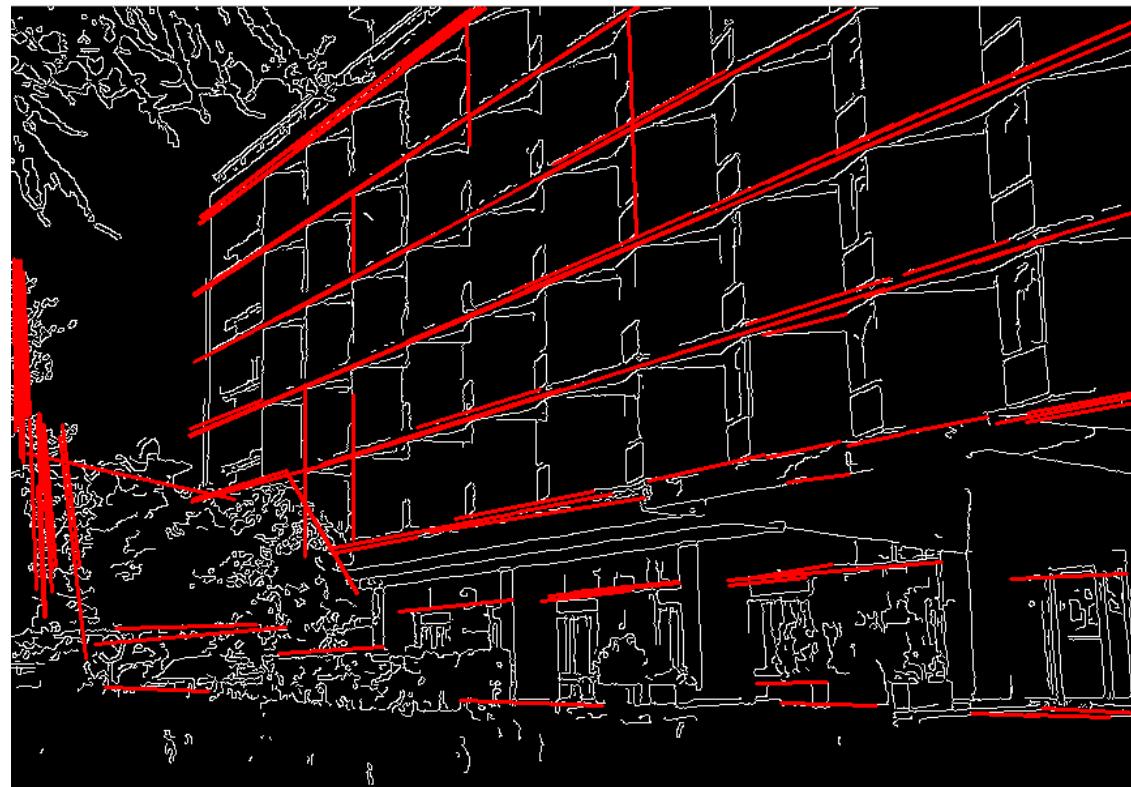
- Find peaks and post-process



Hough Transform: Example



Hough Transform: Example



Hough Transform: Pros and Cons

Pros:

- All points processed independently, so can cope with occlusion
- Some robustness to noise: noisy points are unlikely to contribute consistently to any single bin
- Can detect multiple instances of line/circle/ellipse in a single pass

Cons:

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

Summary

- Line features can be extracted using least squares or RANSAC fitting methods
- Both have difficulties in fitting multiple lines
- The Hough transform can handle multiple lines and the polar form is efficient due to the bounded search region
- Further reading: Chapter 4.3 in the book
- We will apply RANSAC to 3D Vision in future lectures