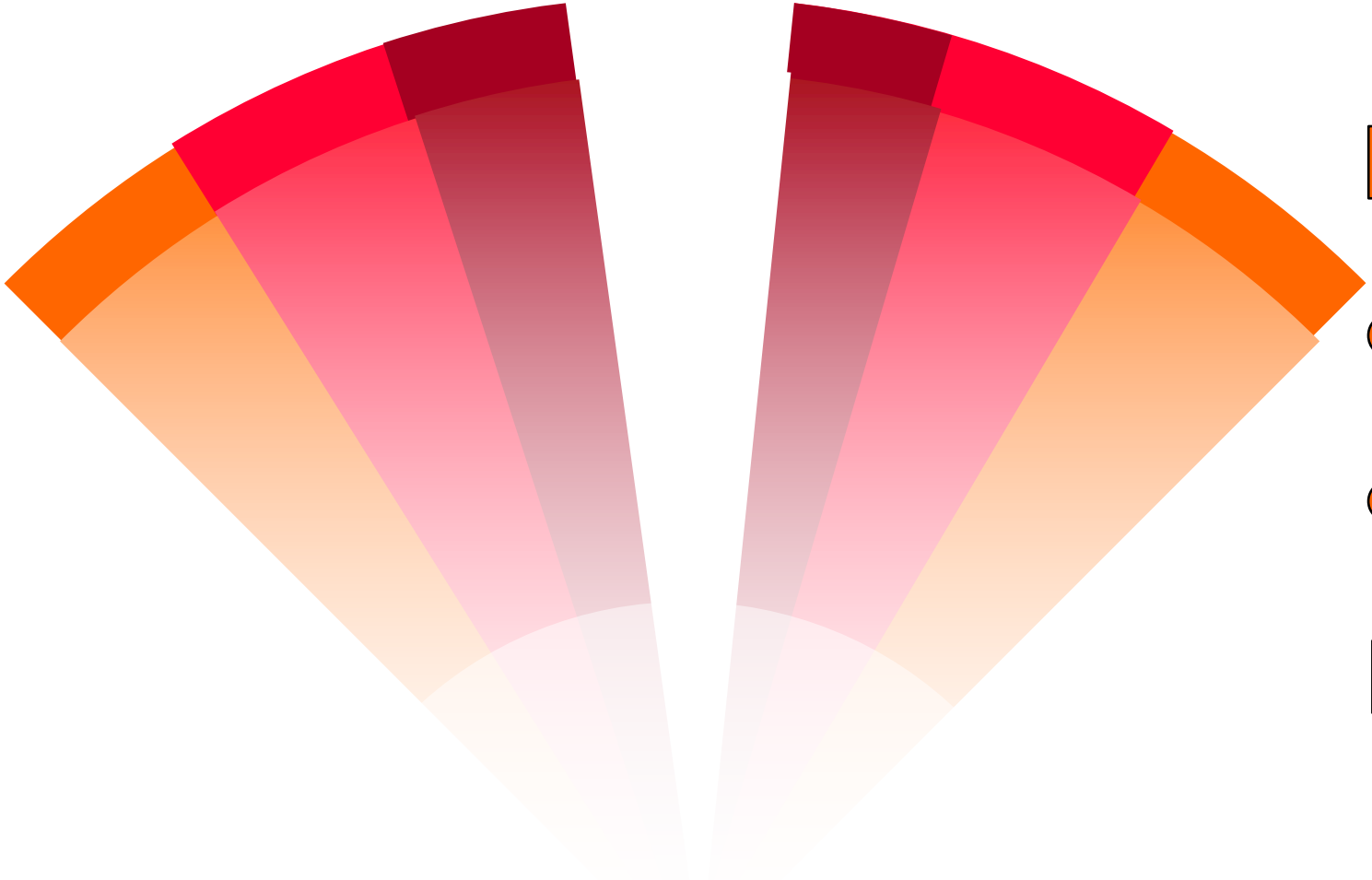
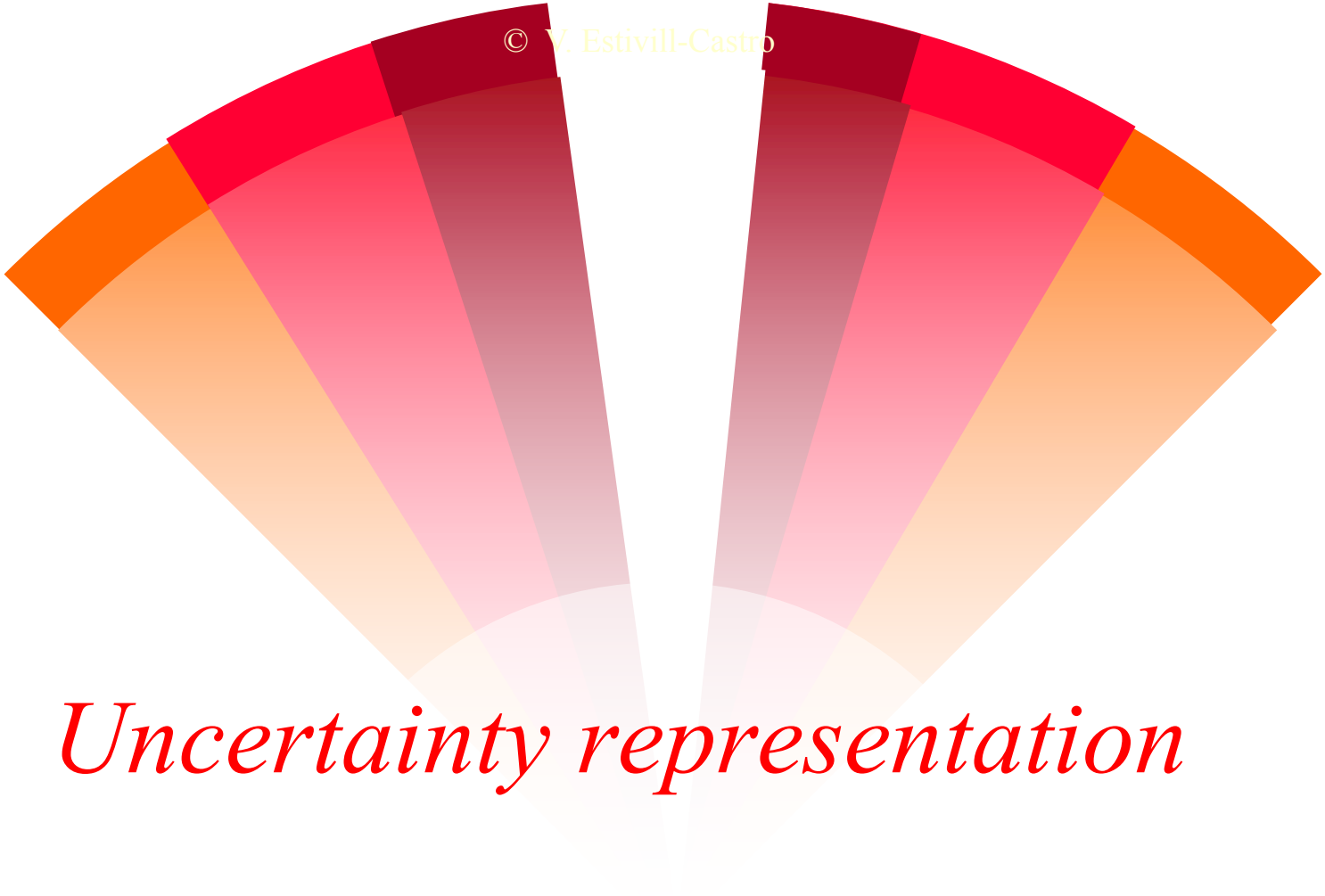


Robots and People

Vlad Estivill-Castro (2016)

Robots for People

--- A project for intelligent integrated systems



Uncertainty representation

Vision

Chapter 4 (textbook)

Sections 4.1.3

Robots and People

What is the course about?

- textbook
- **Introduction to Autonomous Mobile Robots**
- second edition

  Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza

Uncertainty representation

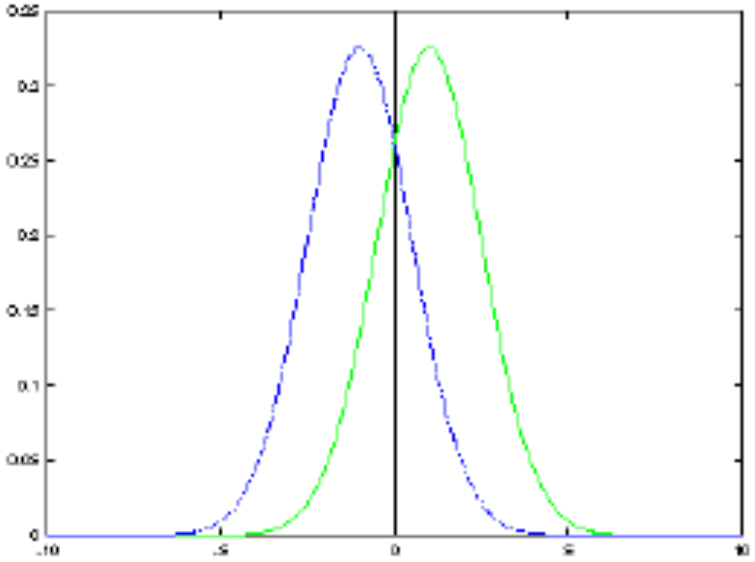
- Sensing in the real world is **always uncertain**
- How can uncertainty be **represented** or quantified?
- How does uncertainty **propagate**?
  - fusing uncertain inputs into a system, what is the resulting uncertainty?
- What is the merit of all this for mobile robotics?

Uncertainty representation

- Use a Probability Density Function (PDF) to characterize the statistical properties of a variable *X*:
  - area adds to 1=integral$_{-\infty}^{\infty}$ $f(x)dx$
- Expected value of variable *X*:
  - $\mu = E[X] =$ integral$_{-\infty}^{\infty}$ $x\, f(x)dx$
- Variance of variable *X*
  - $\sigma^2 =$integral$_{-\infty}^{\infty}$ $(x-\mu)^2 f(x)dx$

Gaussian distribution

- Most common PDF for characterizing uncertainties
  - $f(x) = \exp((x-\mu)^2 / 2\sigma^2) / [\ \sigma\sqrt{2\pi}\ ]$
  - called normal if $\mu=0$ and $\sigma=1$

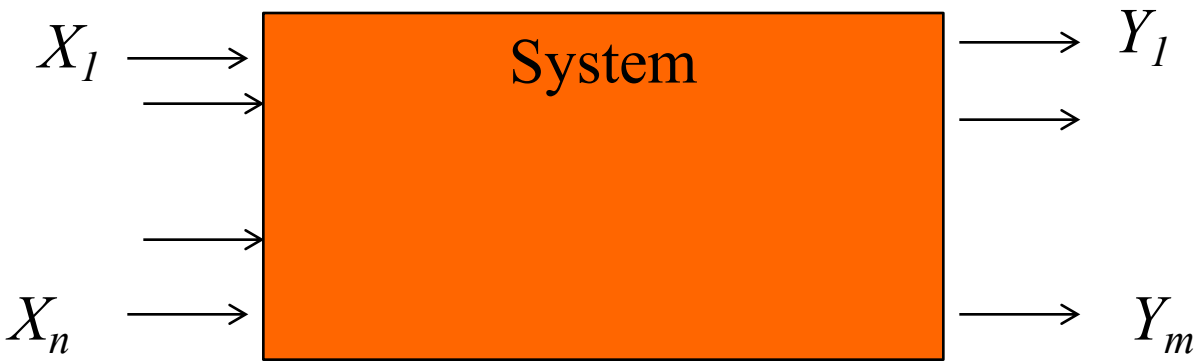


Properties of Gaussian

- 68.26% within +/- σ
- 95.44% within +/- 2σ
- 99.72% within +/-3σ

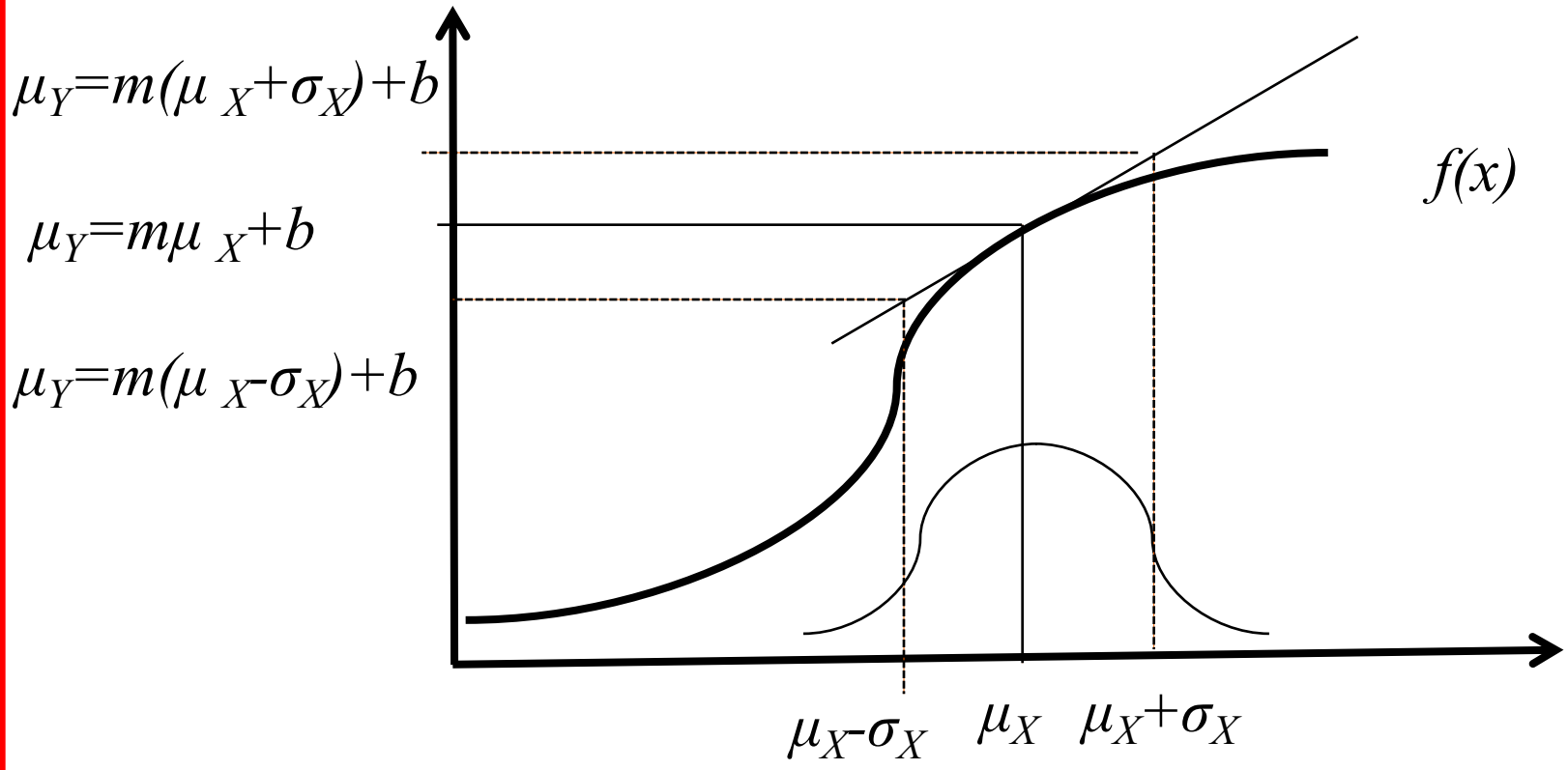
The error propagation law

- Error propagation in a multiple-input multi-output system with *n* inputs and *m* outputs
  - $Y_j = f_j(X_1, X_2, \ldots, X_n)$



Examine in 1D (one dimension)

- It is a nonlinear propagation problem
  - approximate linearly
    - $\mu_Y=m\mu_X+b$ $\sigma_Y=m\sigma_X$ Therefore $\sigma_Y^2=m^2\sigma_X^2$



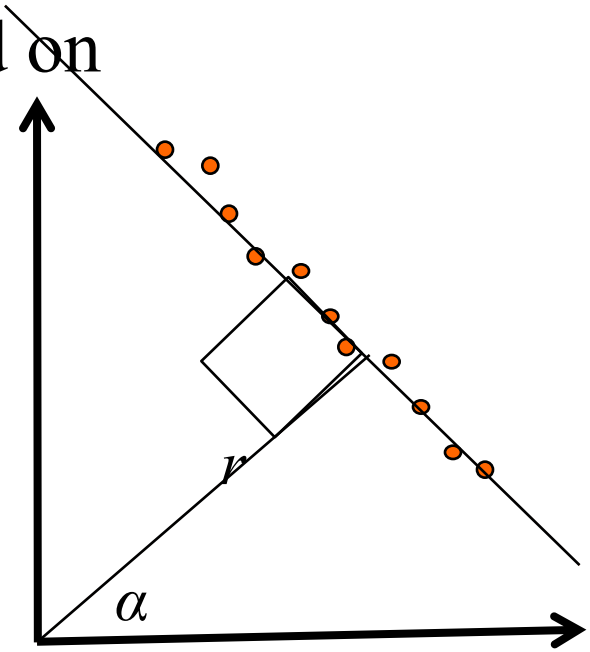
In higher dimensions

- It can be shown that the output covariance matrix C_Y is given by the error propagation law:
  - $C_Y = F_X C_X F_X^T$
  - where
    - C_X:covariance matrix representing the input uncertainties
    - C_Y:covariance matrix representing the propagated output uncertainties
    - is the Jacobian matrix (the transpose off the gradient of $f(X)$ defined as

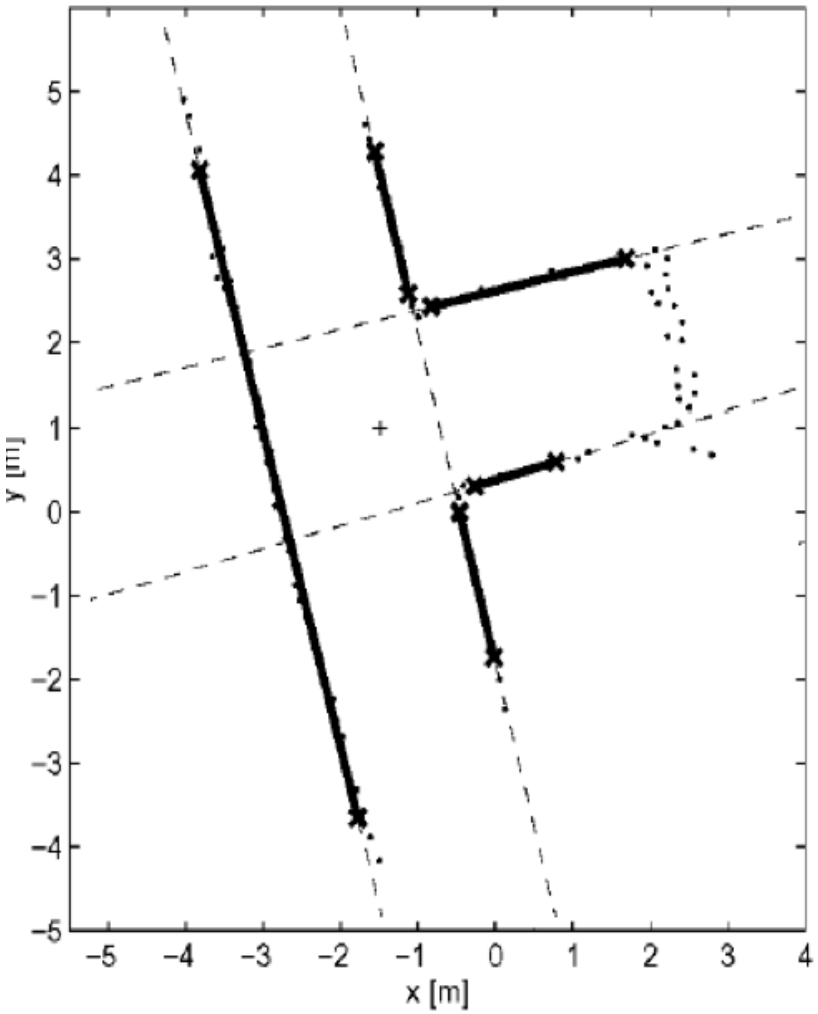
$$\begin{bmatrix} \delta f_1/\delta X_1 & \cdots & \delta f_1/\delta X_n \\ \vdots & & \\ \delta f_m/\delta X_1 & & \delta f_m/\delta X_n \end{bmatrix}$$

Where do we need this?

- Imagine extracting a line based on point measurements with uncertainties.
- Model parameters in polar coordinates [(r,α) uniquely identifies a line]
- The question:
  - What is the uncertainty of the extracted line knowing the uncertainties of the measurement points that contribute to it ?

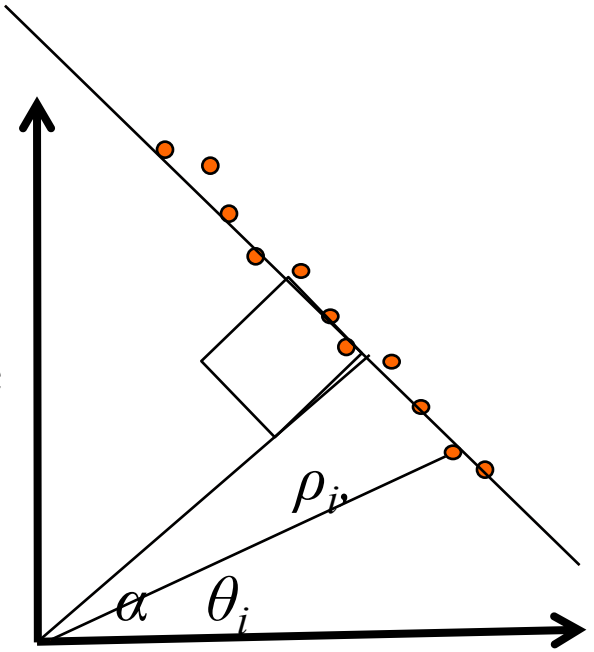


Line extraction from a laser scan



Line extraction (1)

- Each observation (in polar coordinates)
  - $X_i=(\rho_i,\theta_i)$
- Error as point-line distance
  - $d_i = \rho_i \cos(\theta_i - \alpha) - r$
- If each measurement is equally uncertain and independent, the sum of squared errors:
  - $S= \Sigma d_i^2=\Sigma[\ \rho_i \cos(\theta_i - \alpha) - r]^2$



minimize S
when selection (r,α):
$\delta S/d\alpha=0$ $\delta S/dr=0$

Unweigthed Least Squares

Line extraction

- Each sensor measurement may have its own unique uncertainty
  - $\omega_i = 1/\sigma_i^2$
  - $S== \Sigma\omega_i\, d_i^2=\Sigma\omega_i\, [\, \rho_i\, cos(\theta_i\, -\alpha)-r]^2$
  - **Weighted Least Squares**
    - $\delta S/d\alpha=0\ \delta S/dr=0$
  - The solution is given by

$$\alpha = \frac{1}{2}\text{atan}\left(\frac{\sum w_i\rho_i^2\sin 2\theta_i - \frac{2}{\Sigma_{w_i}}\sum\sum w_i w_j \rho_i\rho_j \cos\theta_i \sin\theta_j}{\sum w_i\rho_i^2\cos 2\theta_i - \frac{1}{\Sigma_{w_i}}\sum\sum w_i w_j \rho_i\rho_j \cos(\theta_i+\theta_j)}\right)$$

$$r = \frac{\sum w_i\rho_i\cos(\theta_i-\alpha)}{\sum w_i}$$

What is the uncertainty?

- What error in (r,α)?
- If we assume normal error
  - $\rho_i \approx N(\boldsymbol{\rho_i}, \sigma_{\rho i}^2)$
  - $\theta_i \approx N(\boldsymbol{\theta_i}, \sigma_{\theta i}^2)$
- And that they are independent, we can give a covariance matrix for each measurement $C_{Xi} = \begin{bmatrix} \sigma_{\rho i}^2 & 0 \\ 0 & \sigma_{\theta i}^2 \end{bmatrix}$
- We will apply the *Error Uncertainty Law*

Using the error uncertainty law

- Define $C_X = \begin{bmatrix} \text{diag}(\sigma_\rho^2) & 0 \\ 0 & \text{diag}(\sigma_\theta^2) \end{bmatrix}$
  - size *2nx2n*
- *And a very large Jacobian*
  - $F_{\rho\theta} = \begin{bmatrix} \delta\alpha/\delta\rho_i & \delta\alpha/\delta\rho_{i+1} \ldots \delta\alpha/\delta\theta_i & \delta\alpha/\delta\theta_{i+1} .. \\ \ldots\delta r/\delta\rho_i & \delta r/\delta\rho_{i+1} \ldots \delta r/\delta\theta_i & \delta r/\delta\theta_{i+1} .. \end{bmatrix}$
- *We get* $C_{\alpha r} = F_{\rho\theta}\, C_X\, F_{\rho\theta}^T$

Robotsoapple

This is the technique used to extract features from range data

- How lines are reported in laser scanners
- How planes are reported in 3D scanners

There are also other techniques for line extraction

- They all try to extract lines from a cloud of points.
  - Issues:
    - Enumeration: How many lines are there?
    - Segmentation: Which points belong to a line?
    - Line Fitting/Extraction: Estimate line parameters (give vectorial form)
  - Algorithms
    - Split and merge
    - Linear regression
    - RANSAC
    - Hough-Transform

Split-and-Merge

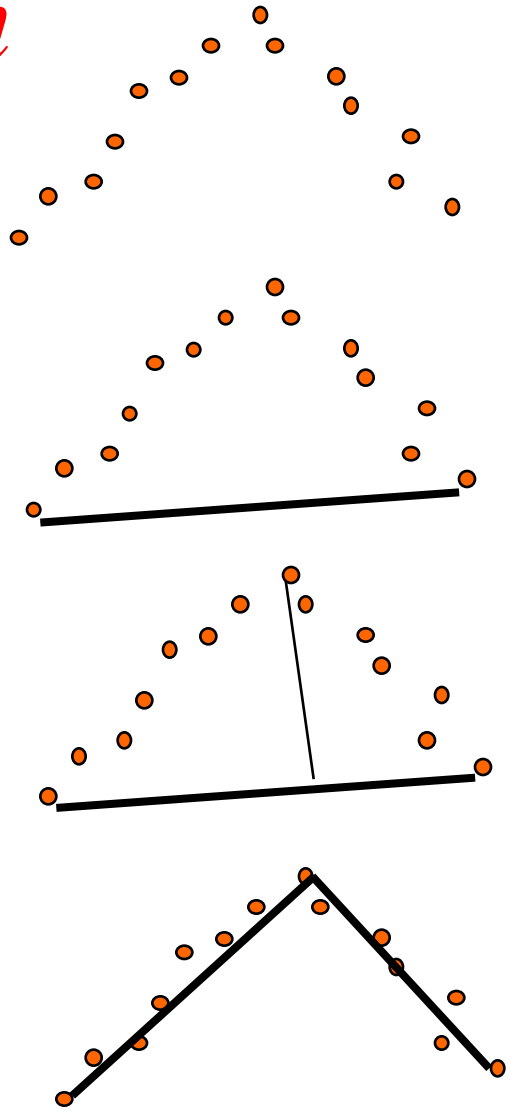
- Claimed to be the most-popular algorithm from computer vision.
  - but rediscovered in the GIS community
    - Ramer–Douglas–Peucker algorithm line simplification
- A recursive procedure of fitting and splitting
  - there are many variations

The pseudo-code (split)

- Given a sequence $S=<p_1,..p_n>$ of points.
- Draw a line between the extremes (p_1,p_n)
- If all are close enough to this line
  - terminate
- Otherwise find the point p_f most at fault
- Apply recursively to
  - $S_l=<p_1,\ldots,p_f>$
  - $S_r=<p_f,\ldots,p_n>$

Robots and people

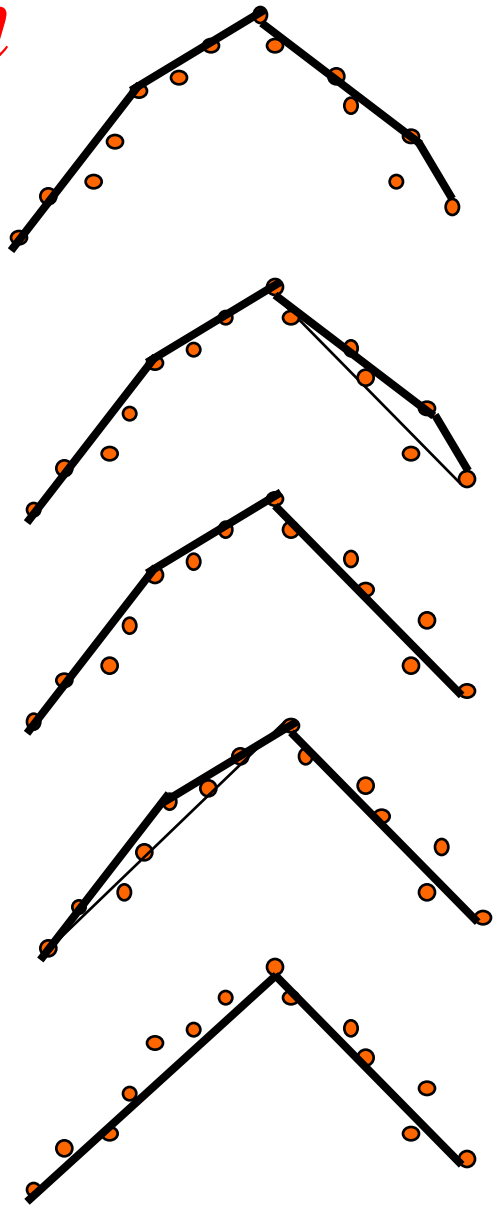
Illustration



The pseudo-code (merge)

- Given a sequence $S=<p_1,..p_n>$ of points and a sequence of lines $L=<l_1,..l_m>$
- Repeat until no loner possible
  - Check if the line $<l_i,l_{i+1}>$ can be replaced by a single line.

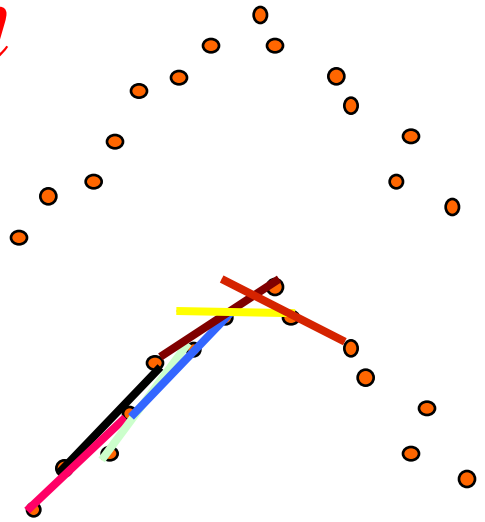
Illustration



Line-regression

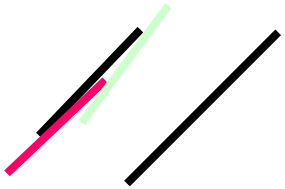
- Segment finding phase
  - use a window of size N_f and compute the line (by line regression) though each consecutive subsequence of N_f point form the input sequence
- Smoothing phase
  - check each consecutive set of 3 windows
  - If the Mahalonobis distance is small enough merge the segments into a single line
  - Add also on both ends if the line parameters are close enough (using clustering)

Illustration

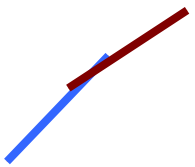


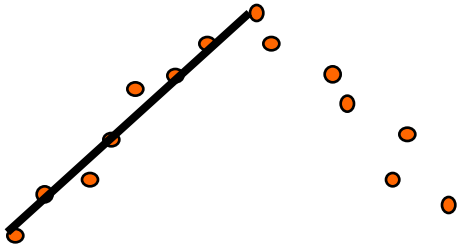
N_f=4

parameters
are close



so are 2
more lines





RANSAC

RANSAC = RANdom SAmple Consensus

- Generic and robust fitting algorithm in the presence of outliers (ie, points which do not satisfy the model)
- Generally applicable algorithm to any problem where the goal is to **identify the inliers which satisfy a predefined model**
  - find a circle
- Typical application in robotics
  - line extraction in 2D data, plane extraction from range 3D data, feature matching, structure from motion
- iterative method and is non-deterministic and the probability to find set free of outliers increases with more interations

Non-deterministic algorithms

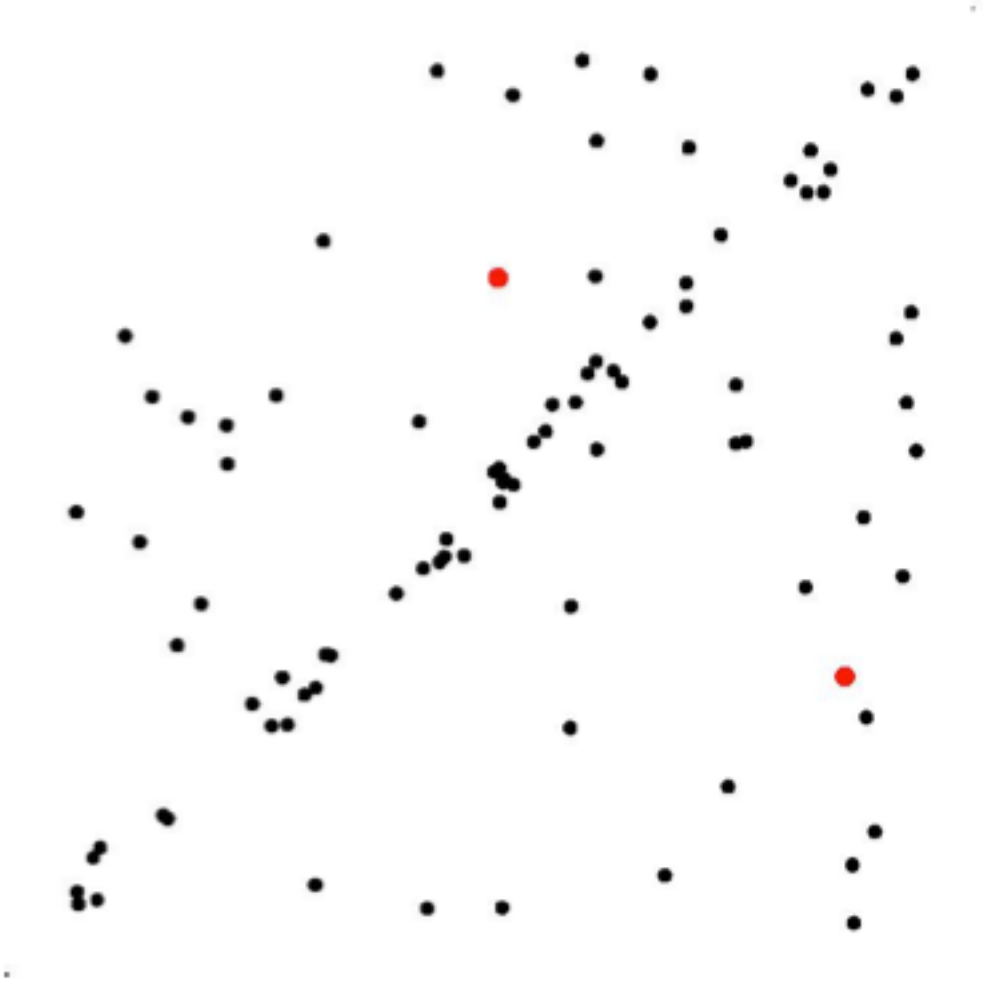
- Use random numbers as guesses
- Provide different results in each execution
  - (if programmed well).

Robots and people

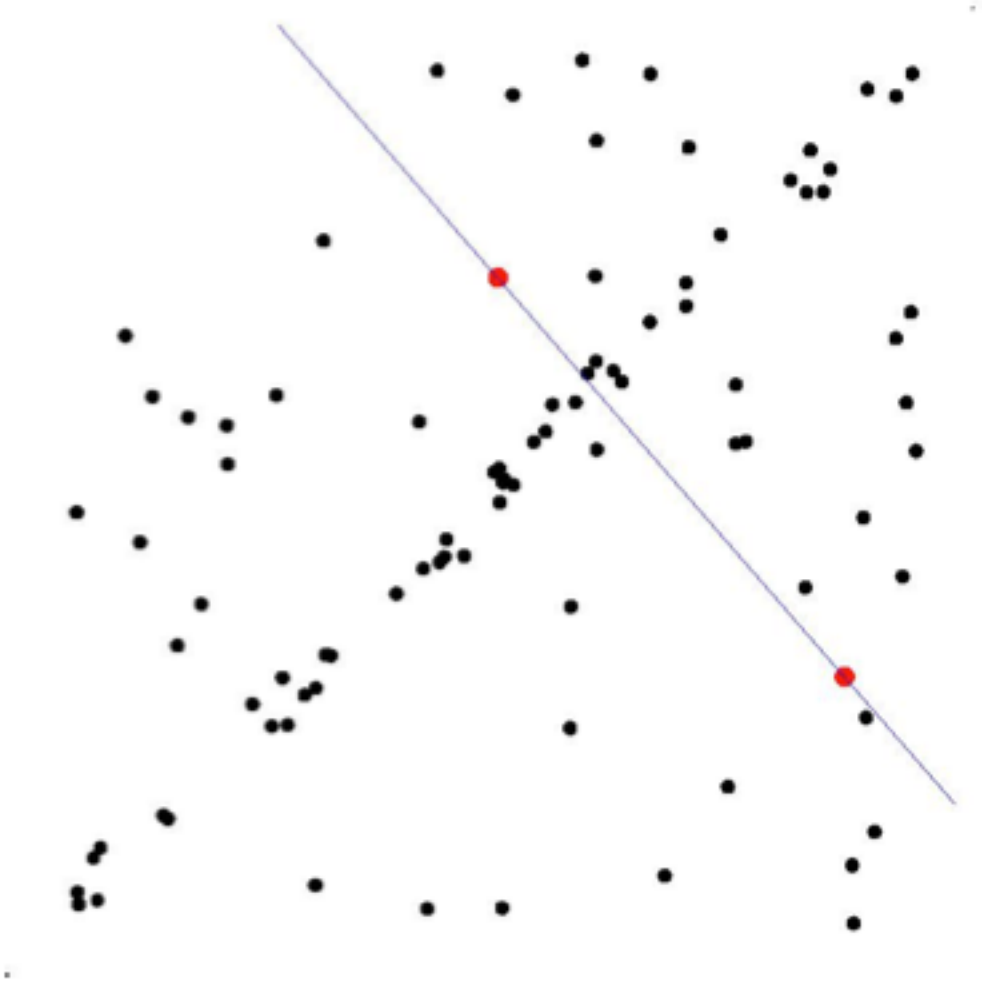
Pseudo code

1. Select a sample of points at random that define the model
    1. for example 2 for a line
2. Calculate the parameters of the model that fits those points.
3. Calculate the error other points make with respect to models
4. Select data that do support the model
5. Repeat until maximum number of iterations is reached
    1. or large proportion of data supports the model
6. Report the model that was supported by the largest data

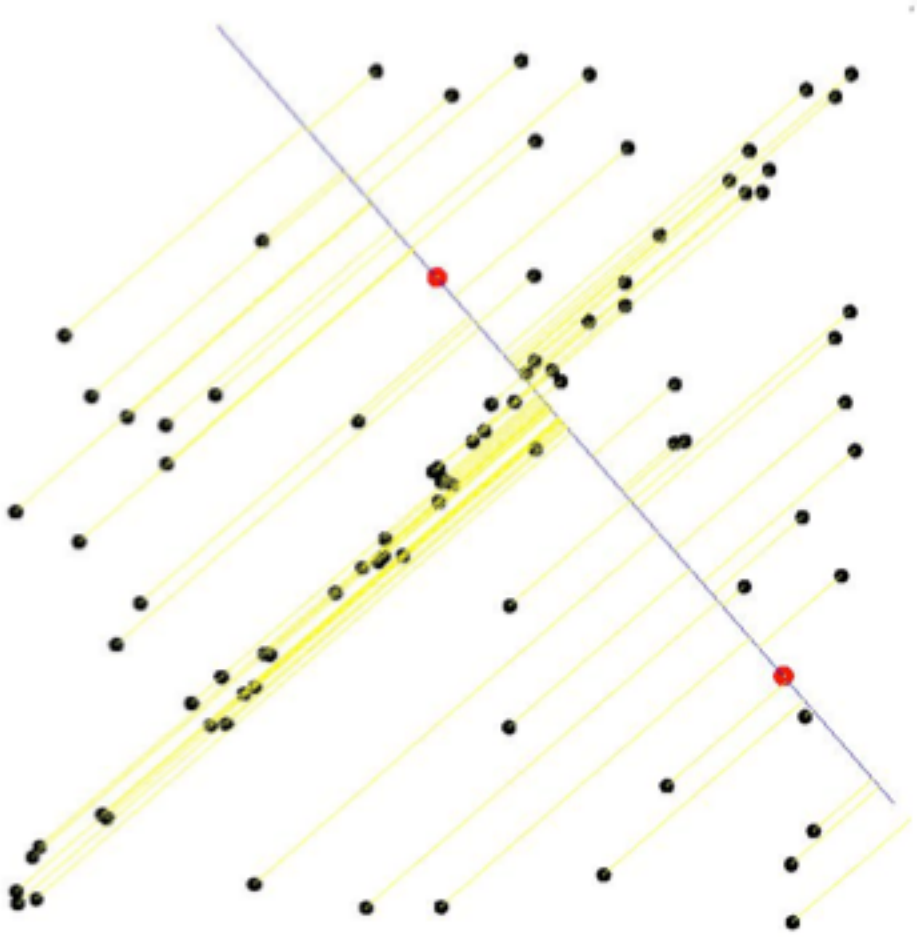
Illustration



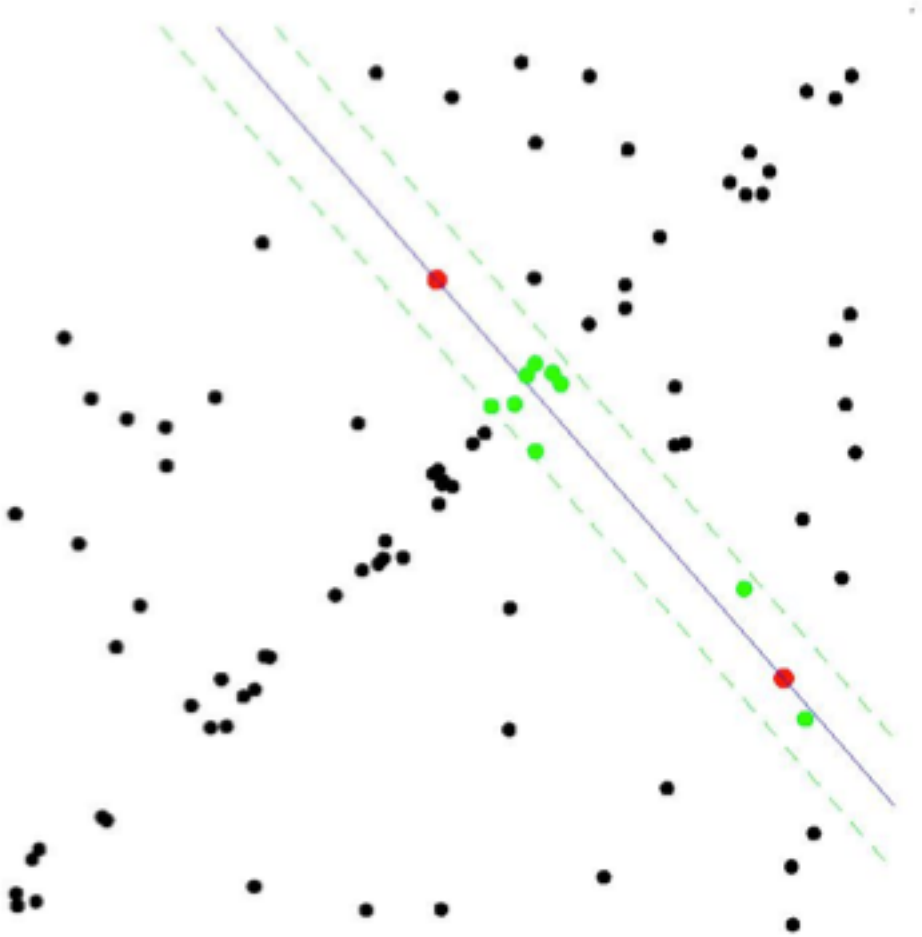
Illustration



Illustration



Illustration



How many iterations does it need?

- Depends on how representative is the model
- Exhaustive search is not feasible
  - For lines determined by 2 points O(n^2)
    - 360 points implies 64,000 iterations
- We need an estimate of the percentage of points that satisfy the model (in-liers) to estimate the iterations

The estimation

- W=fraction of inliers = #inliers/N
  - N is the total number of points and W an estimate of the probability of selecting an in-lier
- If we are selecting independently
  - W^2 = probability both are inlier
  - 1-W^2 = probability at least one is an out-lier
- Since k independent repetitions
  - $(1-W^2)^k$ = prob RANSAC never choses two points that are inliers
  - let $1-p=(1-W^2)^k$ where p is prob of finding set free of outliers

The value of k (the iterations)

- $k = \log(1-p)/\log(1-W^2)$

- With k=16, we have 99% probability of finding a line if 50% of the points are outliers
  - How much does an iteration cost
    - O(N)

Robots and people

Another poplar algorithm to fit shapes

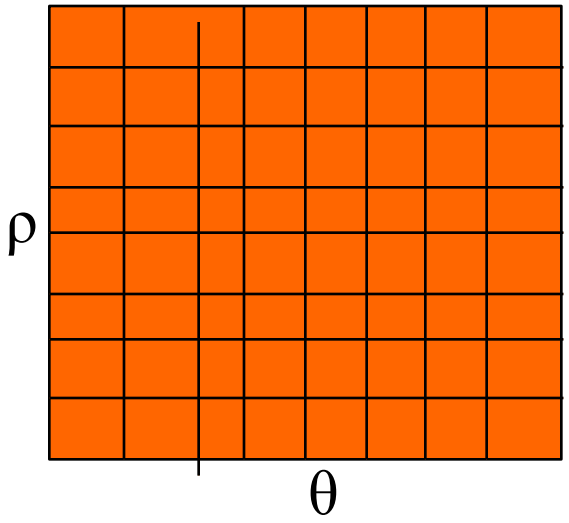
- The Hough-Transform
  - a voting scheme.
  - Same pattern as RANSAC
  - Each point in feature space votes for a model that fits it
  - The model with the most votes is the one fitted.

How does a point vote?

- In image space we have the point
  - $p=(x.y)$
- votes for all the lines that pass trough it
  - the lines have parameters (m,b)
  - for $y=m\,x + b$
- If we have two points
  - $p_1=(x_1,y_1)\ p_2=(x_2,y_2)$
- there is only one pair of parameters (m_0,b_0) that gets 2 votes
  - all other get 1 or 0 votes

Precision, perturbations in the data

- Not all points lie perfectly in the desired line (model)
- The (*m*,*b*) space is unbounded
  - so use polar representation and a two dimensional array
  - line given by
  - $x \cos\theta + y \sin = \rho$



H: accumulator array (votes)

Robots and people

Pseudo code

1. Initialize accumulator H to all zeros
2. for each point (x,y) in the image
   A. for all θ in [0,180)
      I. compute $\rho = x\cos\theta + y\sin\theta$
      II. $H(\theta,\rho)$++
3. Find the values of (θ,ρ) where H is a local maximum
4. Report the line detected as the line with equation $\rho = x\cos\theta + y\sin\theta$

Comparison reported in slides for the course

- Split-and-merge, incremental and line-regression
  - fastest
    - deterministic & make use of the sequential ordering
- If points are captured randomly only line-regression, RANSAC and Hough-transform
- RANSAC and HT produce greater precision in presence of outliers
  - other models besides lines

Comparison s a table

	Complexity	Spped(Hz)	False Positives	Precision
Split-and-Merge	$N \log N$	1500	10%	+++
Incremental	$S\,N$	600	6%	+++
Line-Regression	$N\,N_f$	400	10%	+++
RANSAC	$S\,N\,k$	30	30%	++++
Hogh-Transform	$S\,N\,N_c$ $S\,N_r\,N_c$	10	30%	++++
Expectation Maximization	$SN_1\,N_2\,N$	1	50%	++++

Summary

- There are many algorithms for obtaining more sophisticated features
  - and recognize shapes or objects
  - speed remains a trade-off with precision
- Probability distributions are the standard approach to model uncertainty in the information built as the result of noisy inputs.

THANK YOU