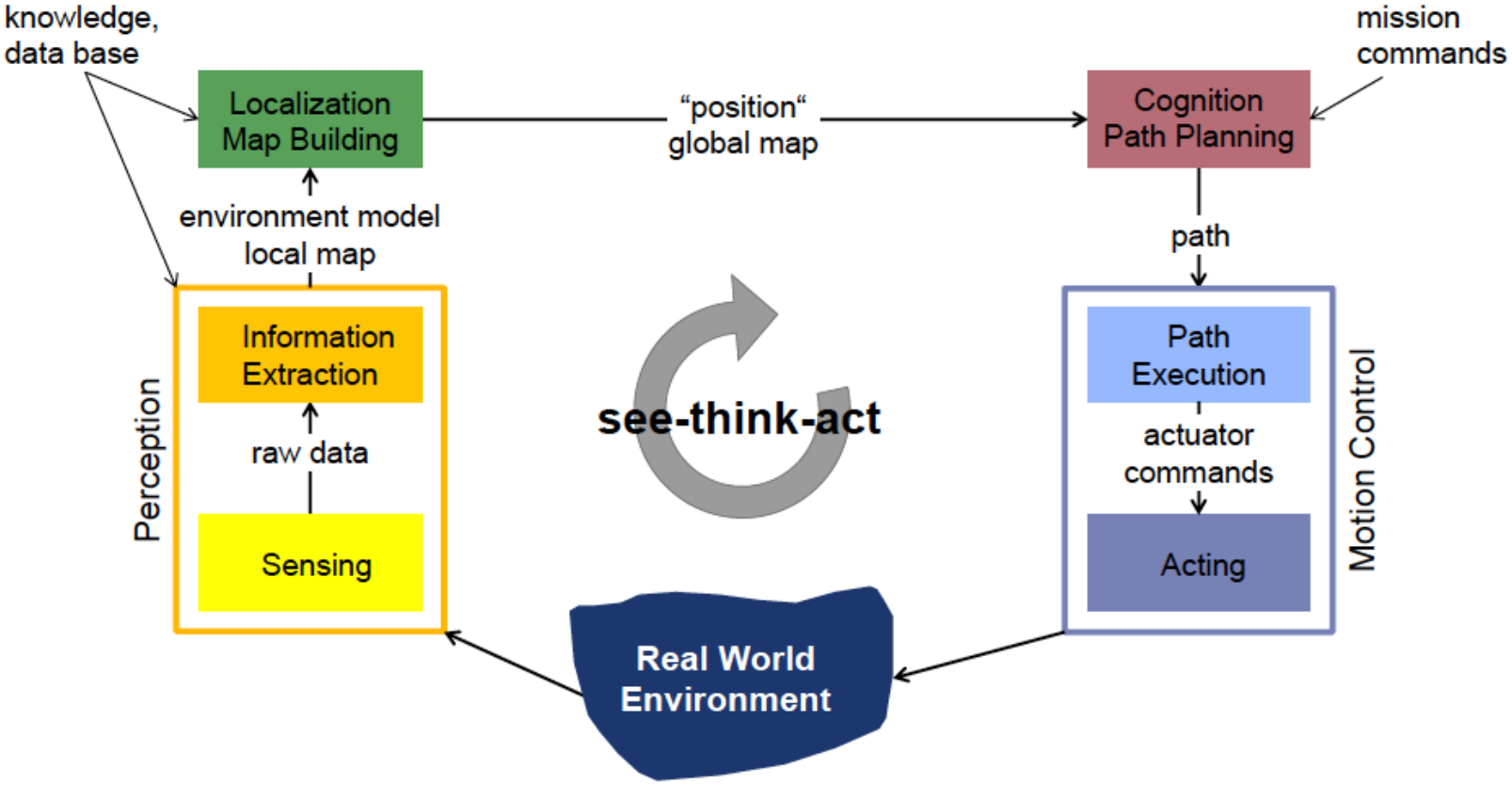
CMPE 185 Autonomous Mobile Robots

Perception: Feature Extraction Based on Range Data

Dr. Wencen Wu
Computer Engineering Department
San Jose State University

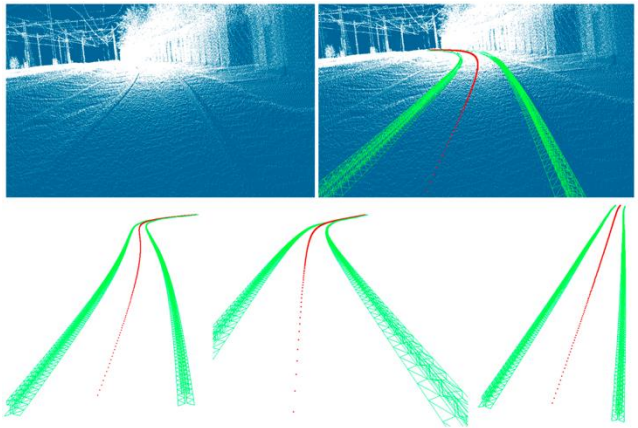
Recall: Perception

- **Mobile robot perception** refers to the process by which a mobile robot acquires, interprets, and understands sensory information from its environment. This perception is crucial for the robot to perform tasks like navigation, object detection, obstacle avoidance, and interaction with its surroundings.

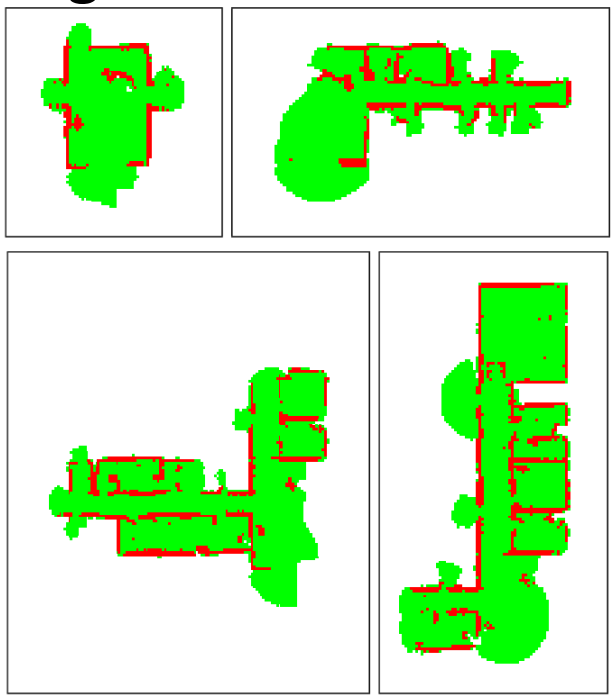


Line Extraction Problem

- Given **range data**, how do we **extract line segments** (or planes)?
- These features (line segments) can be used in many cases such as building maps or extracting road centerlines



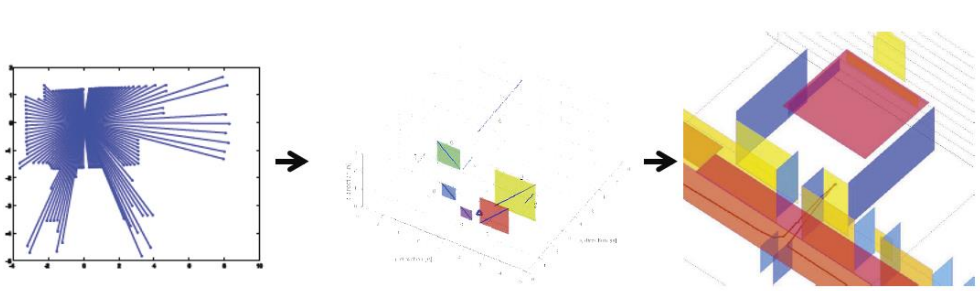
Extraction of road centerlines

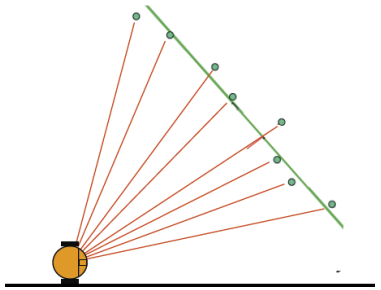


Merging maps from multiple robots

Line Extraction Problem

- Three main problems in line extraction in unknown environments
  - How many lines are there?
  - **Segmentation**: Which points belong to which line?
  - **Line Fitting/Extraction**: Given points that belong to a line, how to estimate the line parameters?





Line Extraction: Representing a line

- A line can be represented by

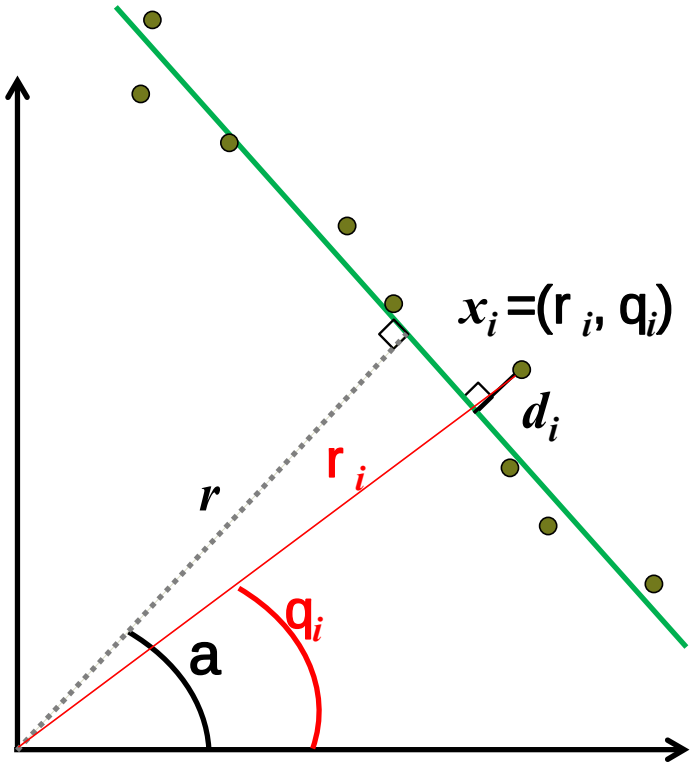
$$y = ax + b$$

- Q: how to represent a vertical line?

- Polar coordinate $\boldsymbol{x} = (r, \alpha)$
  - r: distance from the origin to the closest the line
  - α: angle of the line
  - $r = x * cos\alpha + y * sin\alpha$

- The line can represented by

$$y = \frac{-cos\alpha}{sin\alpha} * x + \frac{r}{sin\alpha}$$

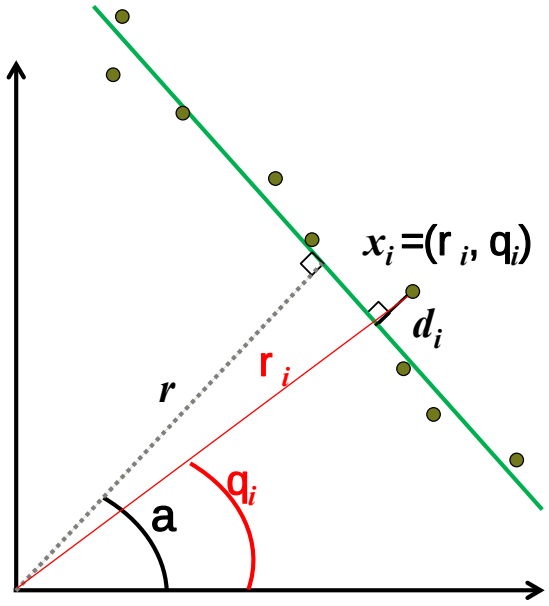


for α ∈ $[0,180]$ and $r \in \mathbf{R}$ (can be positive or negative)
or α ∈ $[0,360]$ and $r \geq 0$ (only positive)

Line Extraction Problem

- Given a measurement vector of *N* range and bearing measurements $\boldsymbol{x_i} = (\rho_i, \theta_i)$, what are the parameters (r, α) that define a line feature for these measurements.

pointing angle of sensor q_i [deg]	range r_i [m]
0	0.5197
5	0.4404
10	0.4850
15	0.4222
20	0.4132
25	0.4371
30	0.3912
35	0.3949
40	0.3919
45	0.4276
50	0.4075
55	0.3956
60	0.4053
65	0.4752
70	0.5032
75	0.5273
80	0.4879



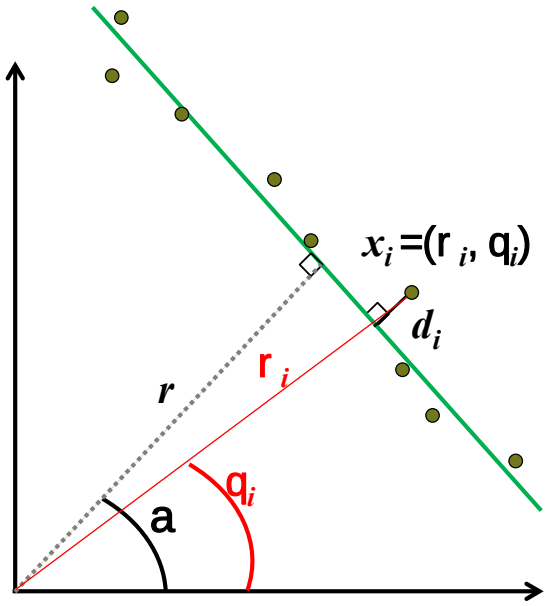
Line Extraction

- Polar measurement $\boldsymbol{x_i} = (\rho_i, \theta_i)$
- The corresponding Euclidean coordinate:
  - $x_i = \rho_i cos\theta_i$
  - $y_i = \rho_i sin\theta_i$
- Point-Line distance

$$\rho_i \cos(\theta_i - \alpha) - r = d_i$$

- If each measurement is equally uncertain, the sum of squared errors:

$$S = \sum_i d_i^2 = \sum_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$



Line Extraction

- Each sensor measurement may have its own, unique uncertainty σ_i^2

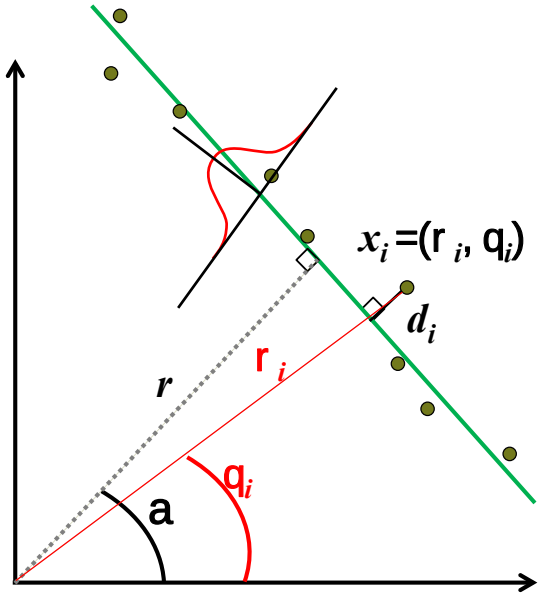
$$S = \sum w_i d_i^2 = \sum w_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

$$w_i = \frac{1}{\sigma_i^2}$$

- Goal: minimize S when selecting (r, α)

$$\frac{\partial S}{\partial \alpha} = 0 \qquad \frac{\partial S}{\partial r} = 0$$

- "Unweighted Least Squares"
- "Weighted Least Squares"



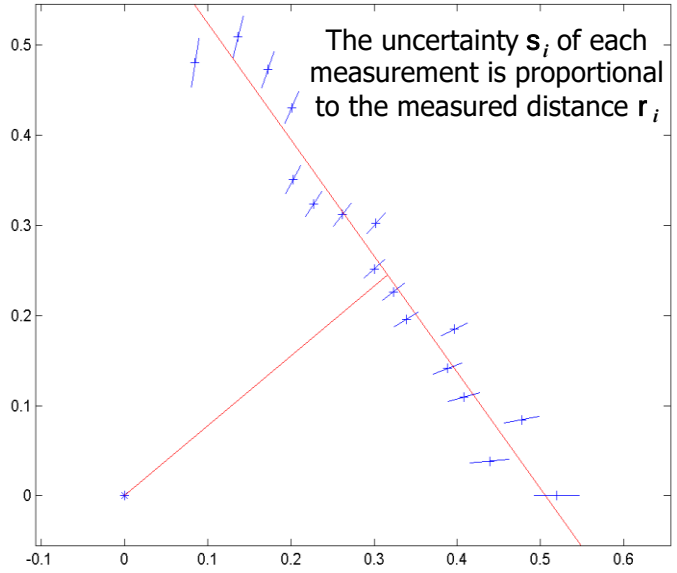
Line Extraction

- Weighted least squares and solving the equations

$$\frac{\partial S}{\partial \alpha} = 0 \qquad \frac{\partial S}{\partial r} = 0$$

- The line parameters are

$$\alpha = \frac{1}{2}\text{atan}\left[\frac{\sum w_i \rho_i^2 sin2\theta_i - \frac{2}{\sum w_i}\sum\sum w_i w_j \rho_i \rho_j cos\theta_i sin\theta_j}{\sum w_i \rho_i^2 cos2\theta_i - \frac{1}{\sum w_i}\sum\sum w_i w_j \rho_i \rho_j cos(\theta_i + \theta_j)}\right]$$

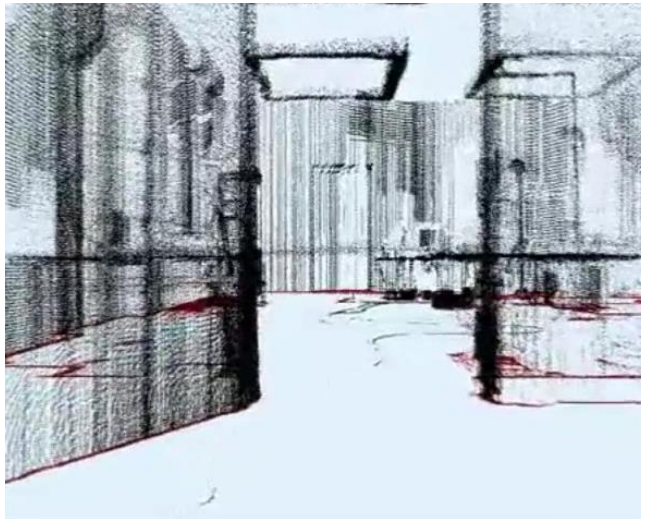


$$r = \frac{\sum w_i \rho_i \cos(\theta_i - \alpha)}{\sum w_i}$$

Line Extraction from a Point Cloud

- Extract lines from a point cloud (e.g. range scan)
- Three main problems:
  - How many lines are there?
  - **Segmentation**: Which points belong to which line?
  - **Line Fitting/Extraction**: Given points that belong to a line, how to estimate the line parameters?

- Algorithms we will see:
  - Split-and-merge
  - Linear regression
  - RANSAC
  - Hough-Transform



Line Extraction – Split and Merge (standard)

- Popular algorithm, originates from Computer Vision.
- A recursive procedure of fitting and splitting.

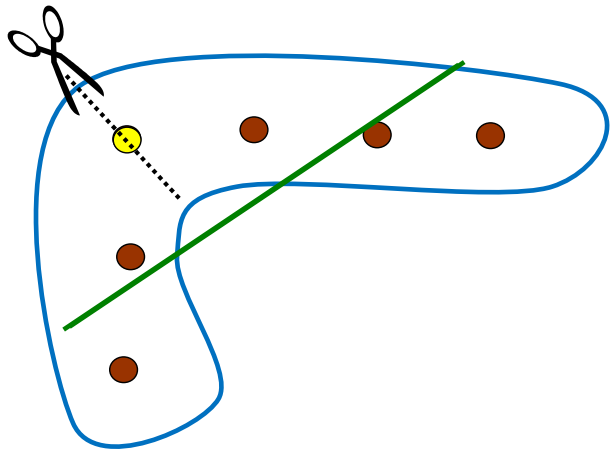
Let **S** be the set of all data points

Split

- Fit a line to points in current set **S**
- Find the most distant point to the line
- If distance > threshold ⇒ split set & repeat with left and right point sets

Merge

- If two consecutive segments are collinear enough, obtain the common line and find the most distant point
- If distance <= threshold, merge both segments



Line Extraction – Split and Merge (standard)

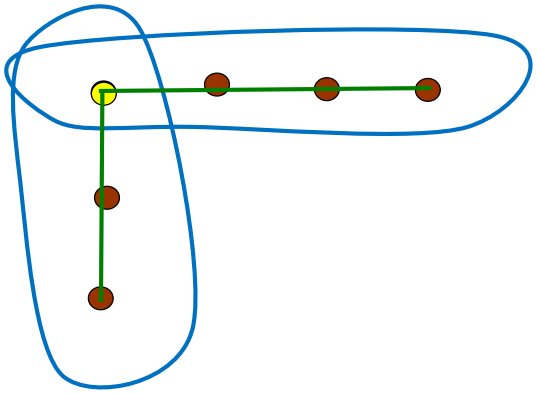
Let **S** be the set of all data points

Split

- Fit a line to points in current set **S**
- Find the most distant point to the line
- If distance > threshold ⓐ split set & repeat with left and right point sets

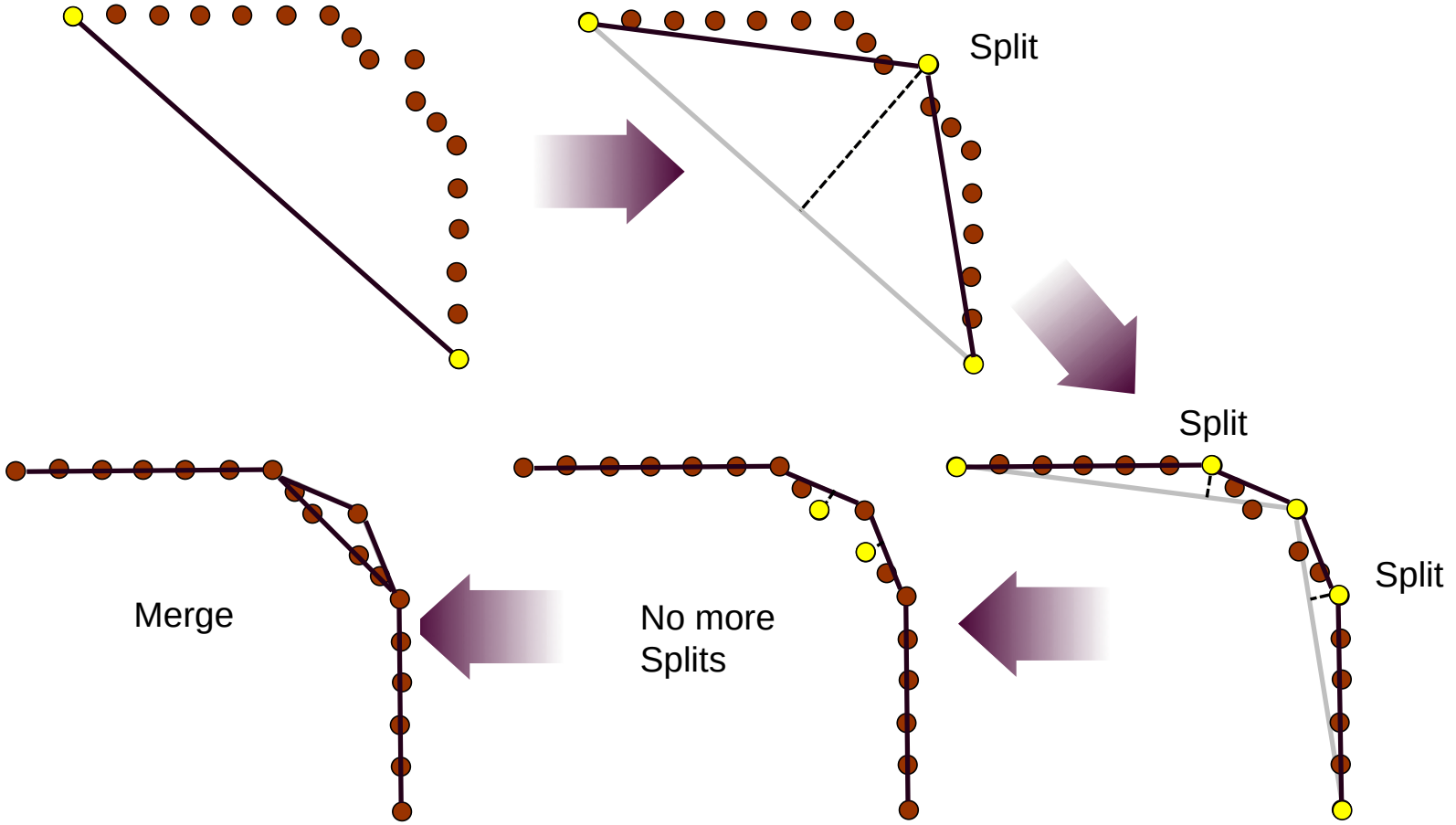
Merge

- If two consecutive segments are collinear enough, obtain the common line and find the most distant point
- If distance <= threshold, merge both segments



Line Extraction – Split and Merge (iterative end-point-fit)

- Iterative end-point-fit: simply connects the end points for line fitting



Line Extraction – Split and Merge

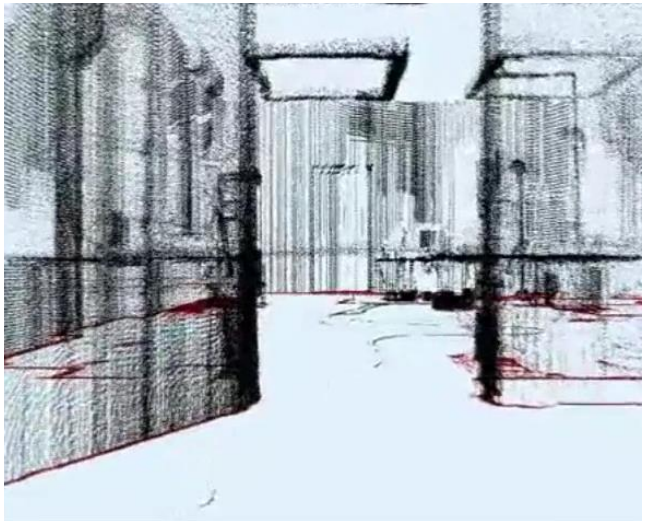
Algorithm 1: Split-and-Merge

Data: Set S consisting of all N points, a distance threshold $d > 0$
Result: L, a list of sets of points each resembling a line
$L \leftarrow (S), i \leftarrow 1;$
while $i \leq len(L)$ **do**
  fit a line (r, α) to the set L_i;
  detect the point $P \in L_i$ with the maximum distance D to the line (r, α);
  if $D < d$ **then**
    $i \leftarrow i + 1$
  else
    split L_i at P into S_1 and S_2;
    $L_i \leftarrow S_1; L_{i+1} \leftarrow S_2;$
  end
end
Merge collinear sets in L;

Line Extraction from a Point Cloud

- Extract lines from a point cloud (e.g. range scan)
- Three main problems:
  - How many lines are there?
  - **Segmentation**: Which points belong to which line?
  - **Line Fitting/Extraction**: Given points that belong to a line, how to estimate the line parameters?

- Algorithms we will see:
  - Split-and-merge
  - Linear regression
  - RANSAC
  - Hough-Transform

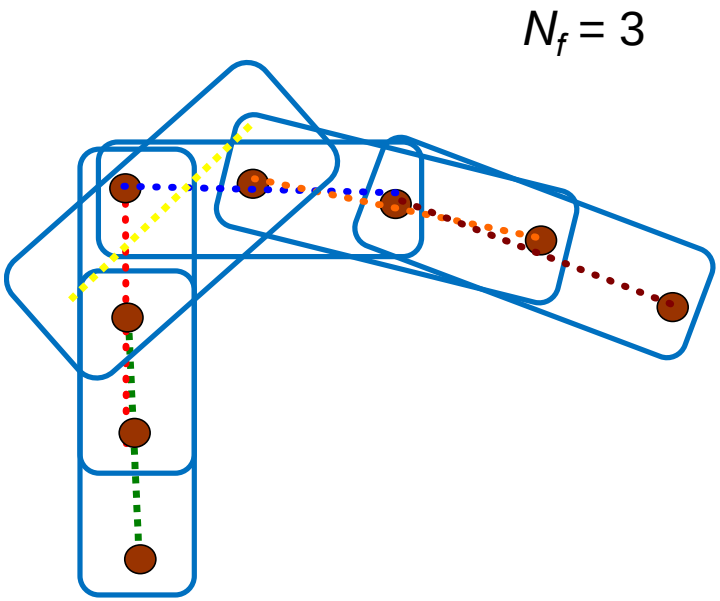


Line Extraction – Line Regression

- "Sliding window" of size N_f points
- Fit line-segment to all points in each window

Line-Regression

- Initialize sliding window size N_f
- Fit a line to every N_f consecutive points (i.e. in each window)
- Merge overlapping line segments + re-compute line parameters for each segment

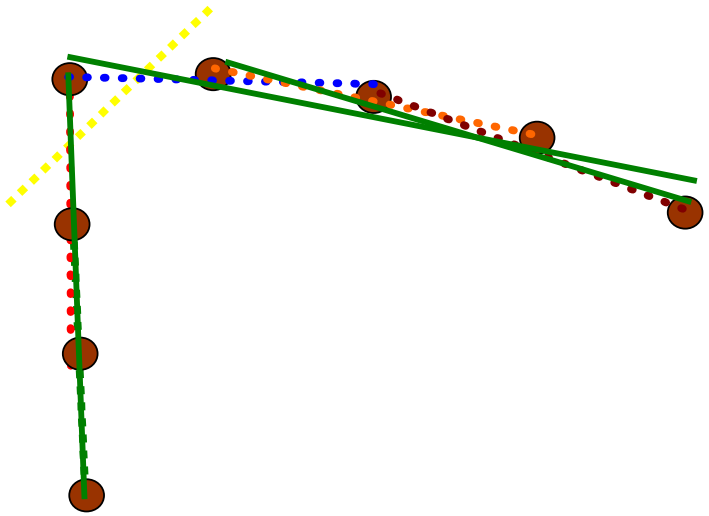


Line Extraction – Line Regression

Line-Regression

- Initialize sliding window size N_f
- Fit a line to every N_f consecutive points (i.e. in each window)
- Merge overlapping line segments + re-compute line parameters for each segment

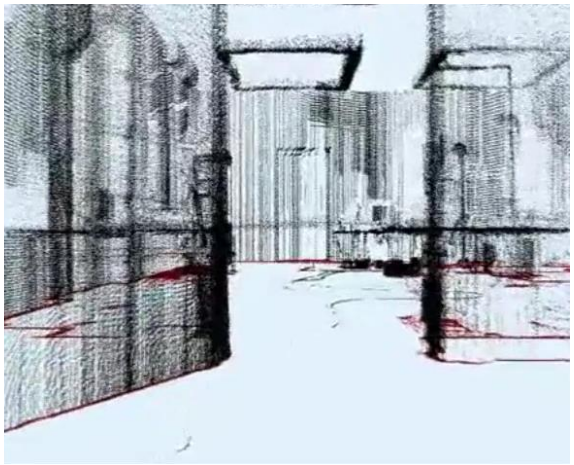
$N_f = 3$



Line Extraction from a Point Cloud

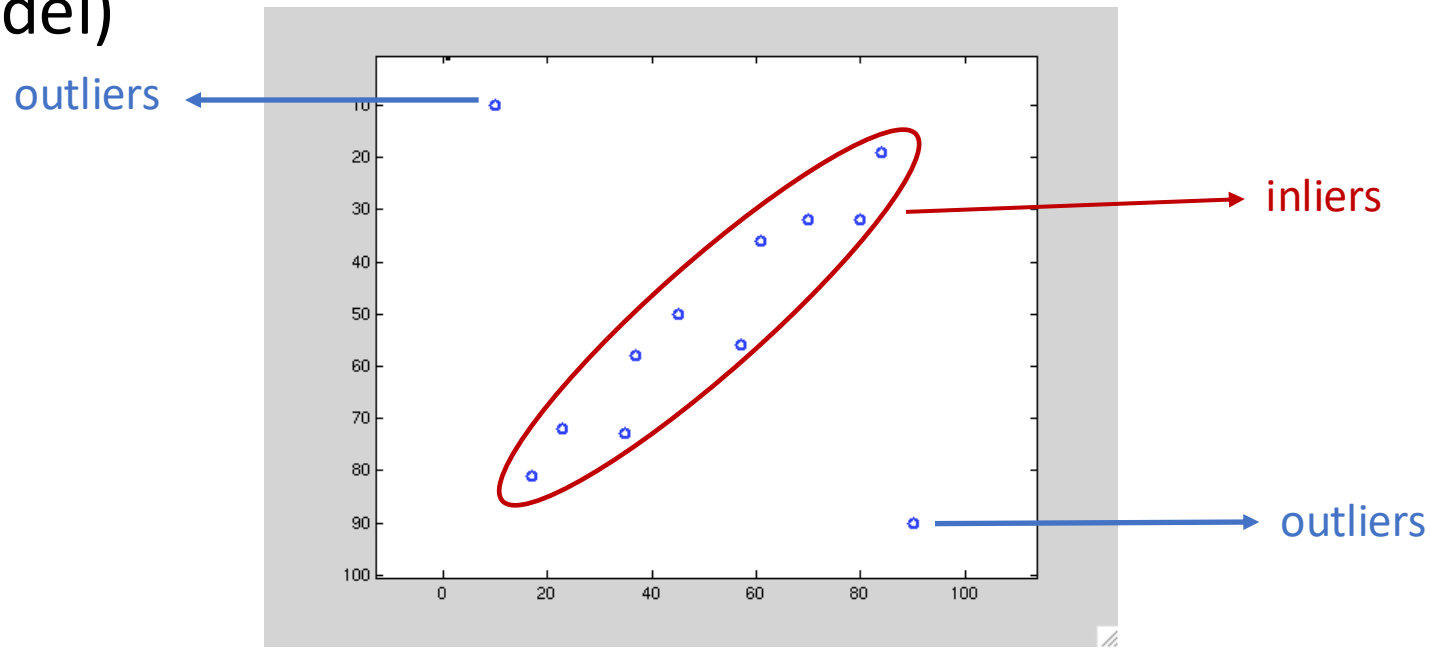
- Extract lines from a point cloud (e.g. range scan)
- Three main problems:
  - How many lines are there?
  - **Segmentation**: Which points belong to which line?
  - **Line Fitting/Extraction**: Given points that belong to a line, how to estimate the line parameters?

- Algorithms we will see:
  - Split-and-merge
  - Linear regression
  - RANSAC
  - Hough-Transform



Line Extraction – RANSAC

- RANSAC = RANdom Sample Consensus
- A generic & robust fitting algorithm of models **in the presence of outliers** (i.e. points which do not satisfy a model)



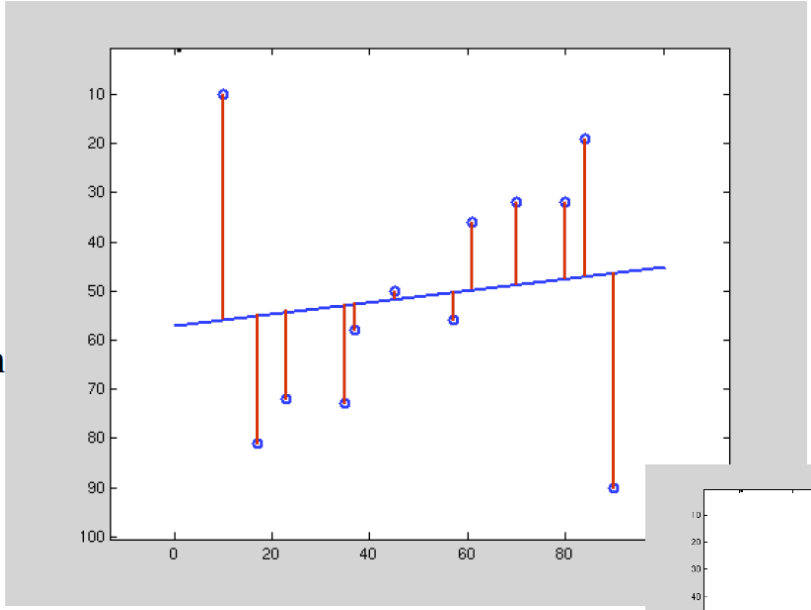
M. Fischler & R. C.Bolles. RANndom SAmple Consensus:
A paradigm for model fitting with applicatIons to image analysis and automated cartography. Graphics and Image Processing, **1981**.

Line Extraction – RANSAC

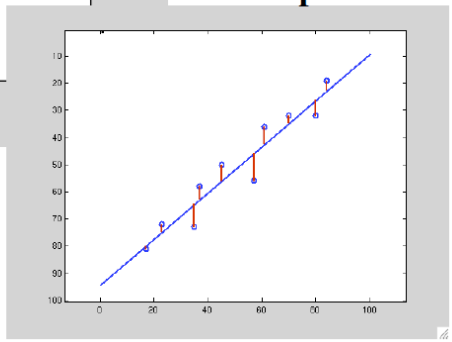
- RANSAC = RANdom Sample Consensus

Least squares estimation is sensitive to outliers, so that a few outliers can greatly skew the result.

Least squares regression with outliers



compare



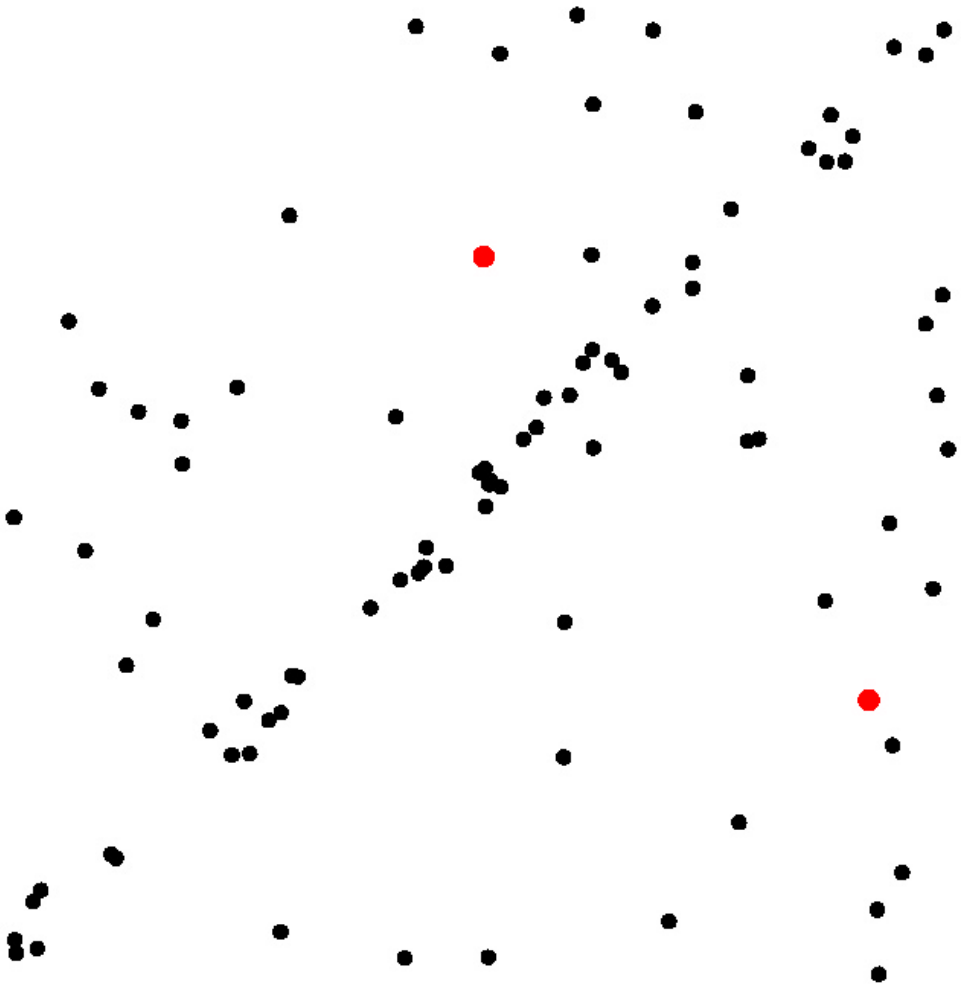
Solution: Estimation methods that are robust to outliers.

Line Extraction – RANSAC

- RANSAC = RANdom Sample Consensus
- Can be applied in general to any problem, where the goal is to **identify the inliers which satisfy a predefined model**
- Typical applications in robotics are:
  line extraction from 2D range data, plane extraction from 3D data, feature matching, structure from motion, camera calibration, homography estimation, etc.
- RANSAC is **iterative** and **non-deterministic:** the probability to find a set free of outliers increases as more iterations are used
- Drawback: a non-deterministic method, results are different between runs

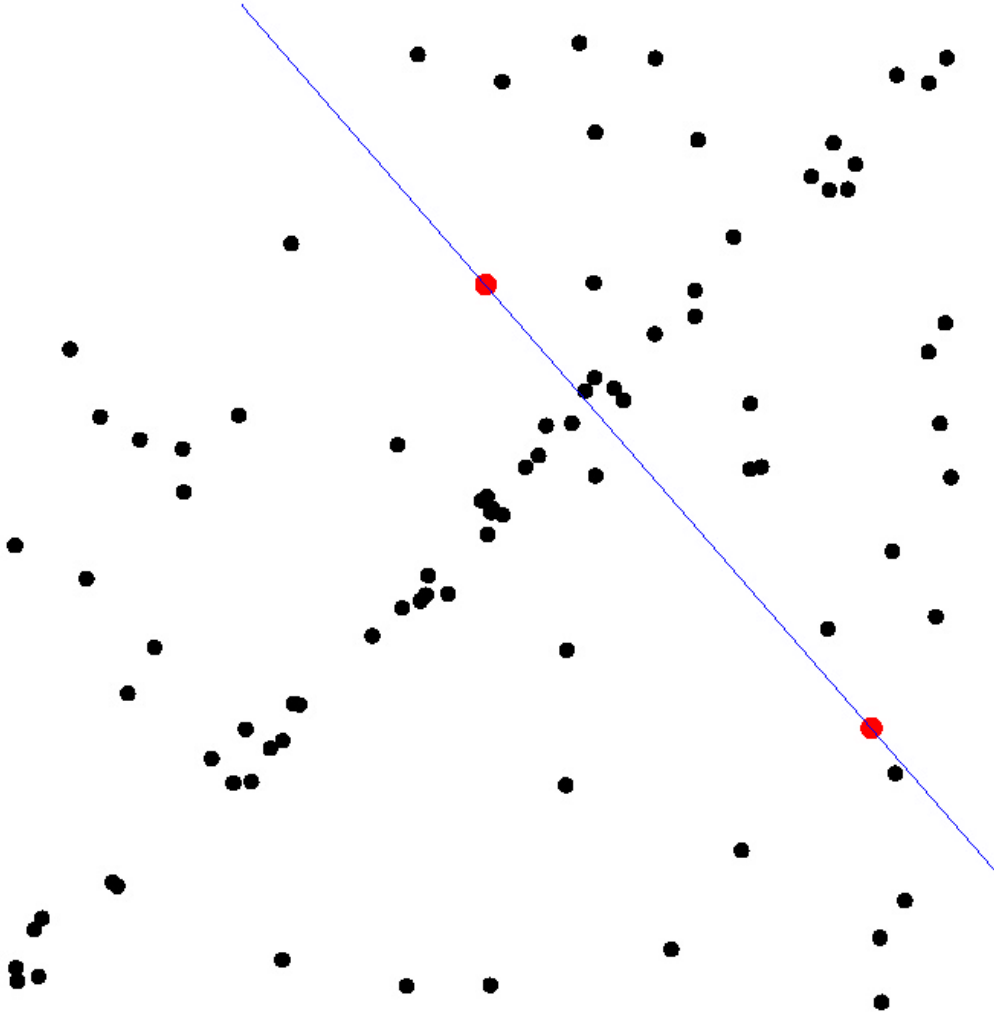
M. Fischler & R. C.Bolles. RANndom SAmple Consensus:
A paradigm for model fitting with applicatlons to image analysis and automated cartography. Graphics and Image Processing, **1981**.

Line Extraction – RANSAC



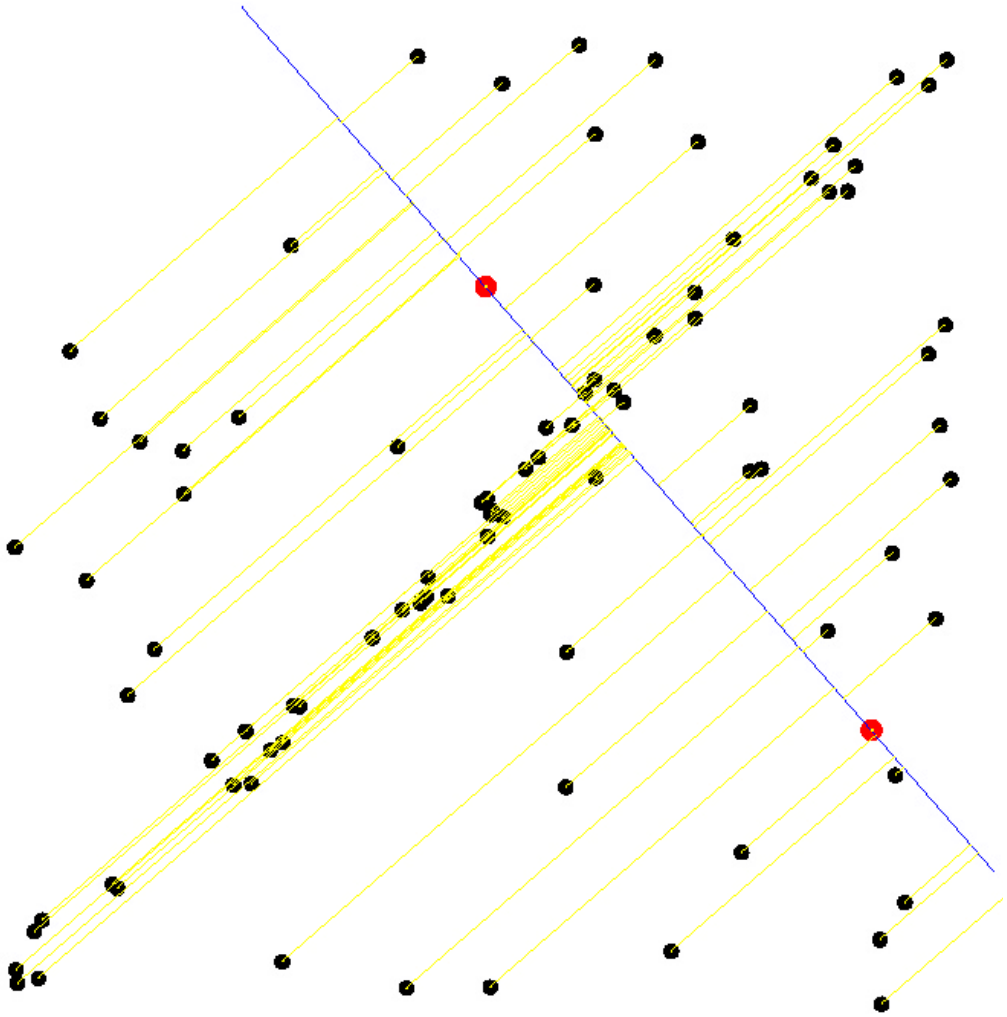
- Select a sample of two points at random

Line Extraction – RANSAC



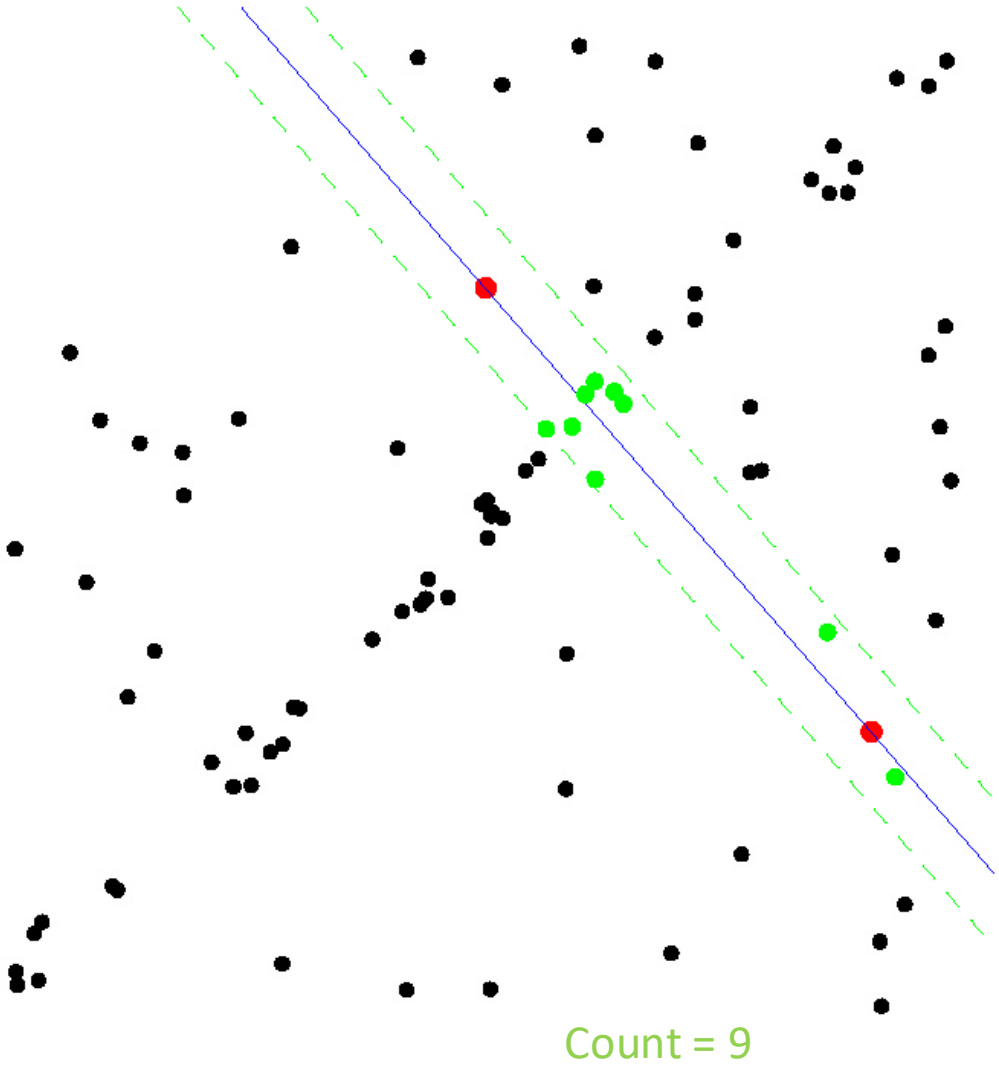
- Select a sample of two points at random
- Calculate model parameters that fit the data in the sample

Line Extraction – RANSAC



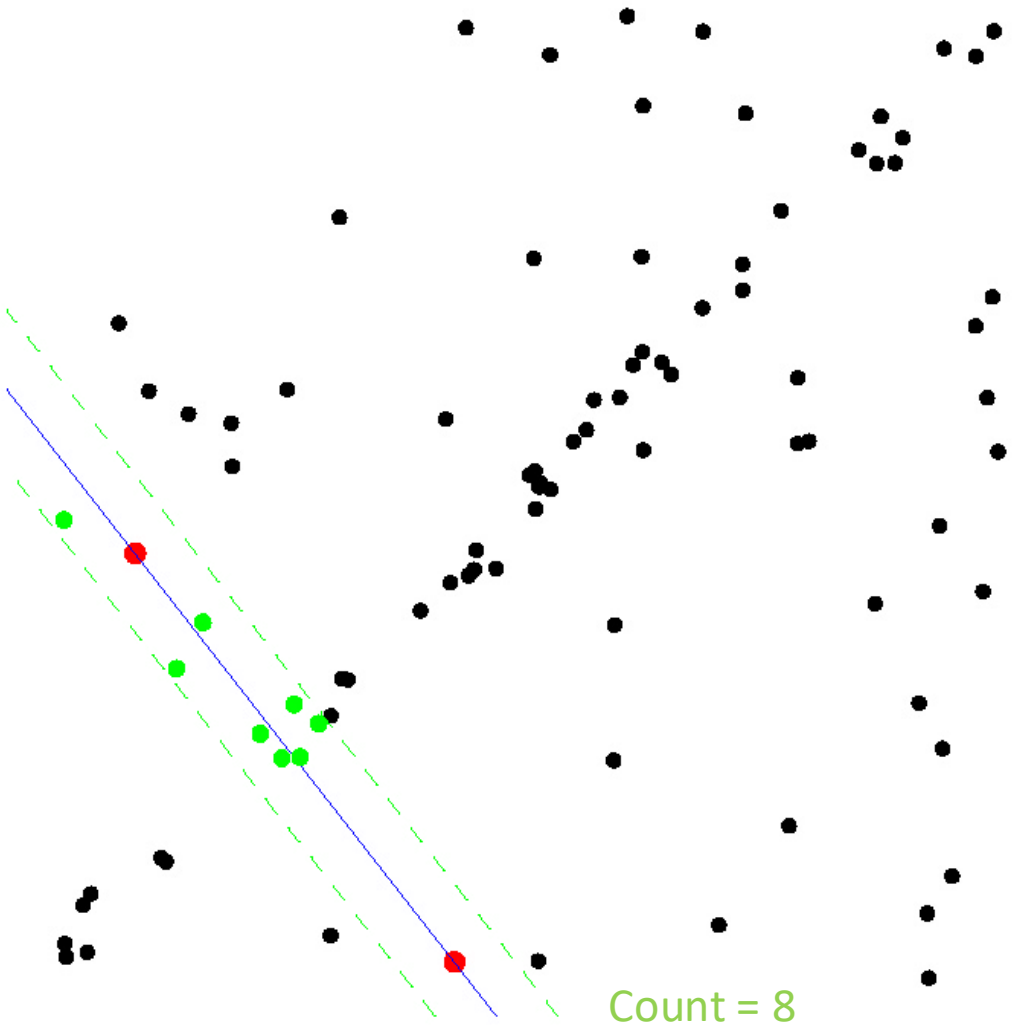
- Select a sample of two points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point

Line Extraction – RANSAC



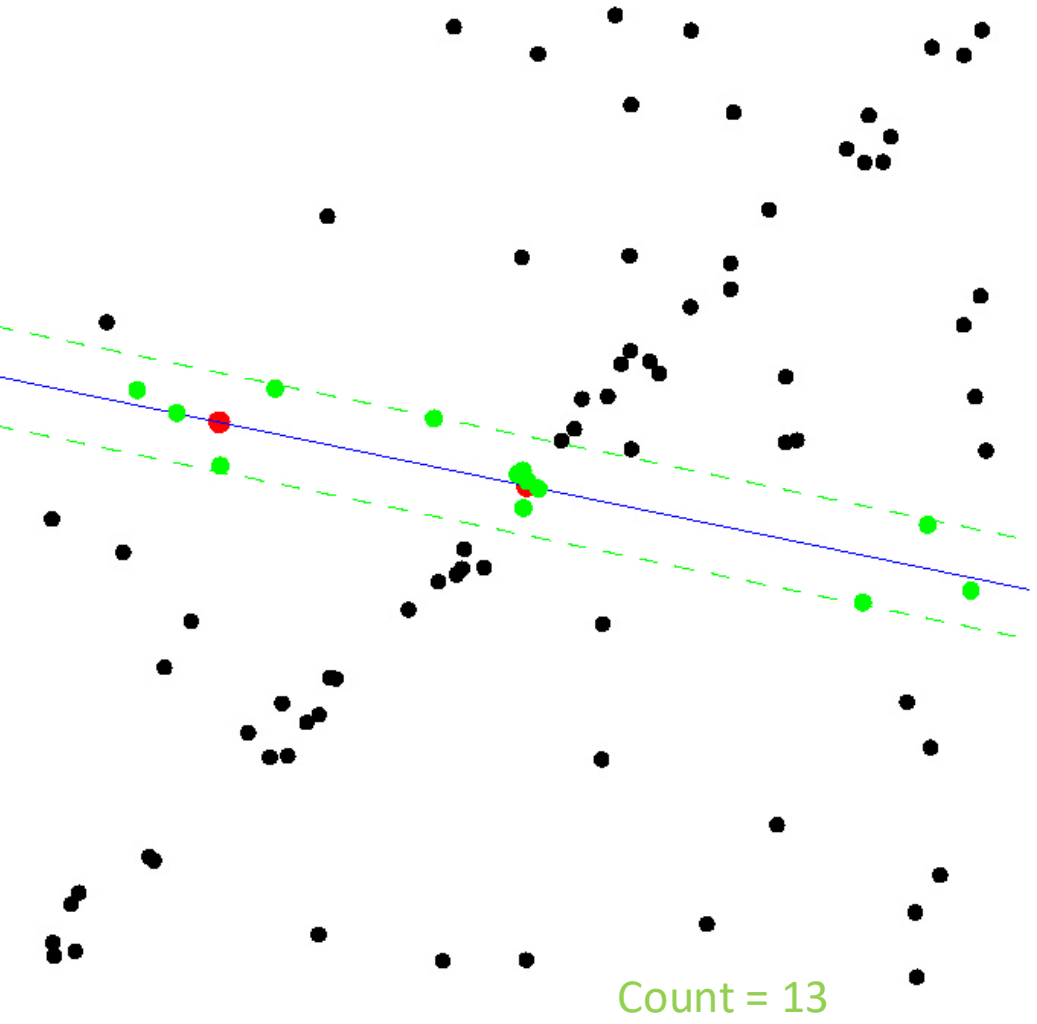
- Select a sample of two points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis

Line Extraction – RANSAC



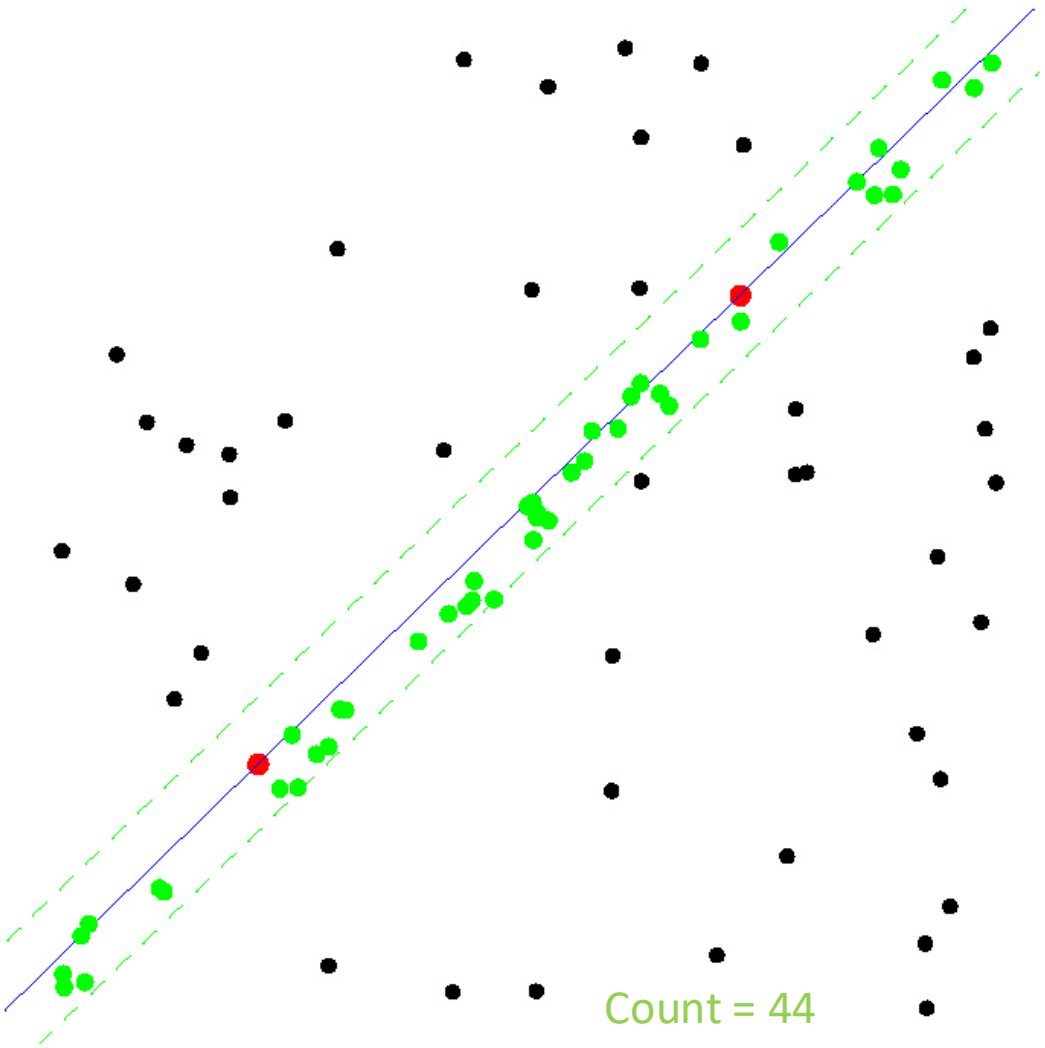
- Select a sample of two points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
- Repeat sampling

Line Extraction – RANSAC



- Select a sample of two points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
- Repeat sampling

Line Extraction – RANSAC



- Select a sample of two points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
- Repeat sampling

Set with the maximum number of inliers obtained after *k* iterations

Line Extraction – RANSAC

How many iterations does RANSAC need? ***k* =?**

- Ideally: check all possible combinations of **2** points in a dataset of ***N*** points
- Number of all pairwise combinations: ***N(N-1)/2***
  - computationally infeasible if ***N*** is too large.
    example:
    10’000 points to fit a line through a need to check all 10’000 x 9’999/2 = **50 million** combinations!
- Do we really need to check all combinations or can we stop after some iterations?
  - Checking a **subset** of combinations is enough **if** we have a **rough** estimate of the **percentage of inliers** in our dataset
- This can be done in a probabilistic way

Line Extraction – RANSAC

How many iterations does RANSAC need? k **=?**

- N: total number of data points
- w: number of inliers / N
  - w is the fraction of inliers in the dataset, the probability of selecting an inlier-point out of the dataset
- p: the desired probability of success
- The number of RANSAC iterations needed is

$$k = \frac{\log(1-p)}{\log(1-w^2)}$$

- **Example:** if we want a probability of success p = 99%, and we know that w = 50% → k = 16 iterations
  - dramatically fewer than the number of all possible combinations!

- In practice, we need only a rough estimate of w. There are more advanced variants of RANSAC

Line Extraction – RANSAC

RANSAC is not only for line extraction

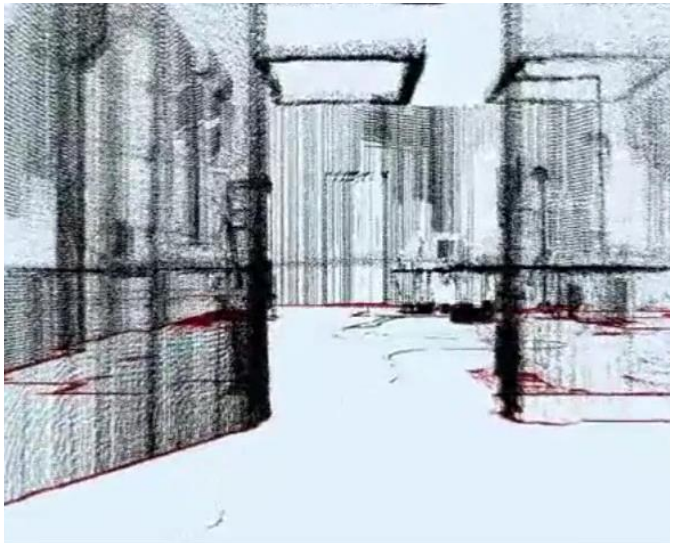
```
Given:
    data – a set of observations
    model – a model to explain observed data points
    n – minimum number of data points required to estimate model parameters
    k – maximum number of iterations allowed in the algorithm
    t – threshold value to determine data points that are fit well by model
    d – number of close data points required to assert that a model fits well to data

Return:
    bestFit – model parameters which best fit the data (or nul if no good model is found)

iterations = 0
bestFit = nul
bestErr = something really large
while iterations < k {
    maybeInliers = n randomly selected values from data
    maybeModel = model parameters fitted to maybeInliers
    alsoInliers = empty set
    for every point in data not in maybeInliers {
        if point fits maybeModel with an error smaller than t
             add point to alsoInliers
    }
    if the number of elements in alsoInliers is > d {
        % this implies that we may have found a good model
        % now test how good it is
        betterModel = model parameters fitted to all points in maybeInliers and alsoInliers
        thisErr = a measure of how well betterModel fits these points
        if thisErr < bestErr {
            bestFit = betterModel
            bestErr = thisErr
        }
    }
    increment iterations
}
return bestFit
```

wikipedia

Line Extraction from a Point Cloud

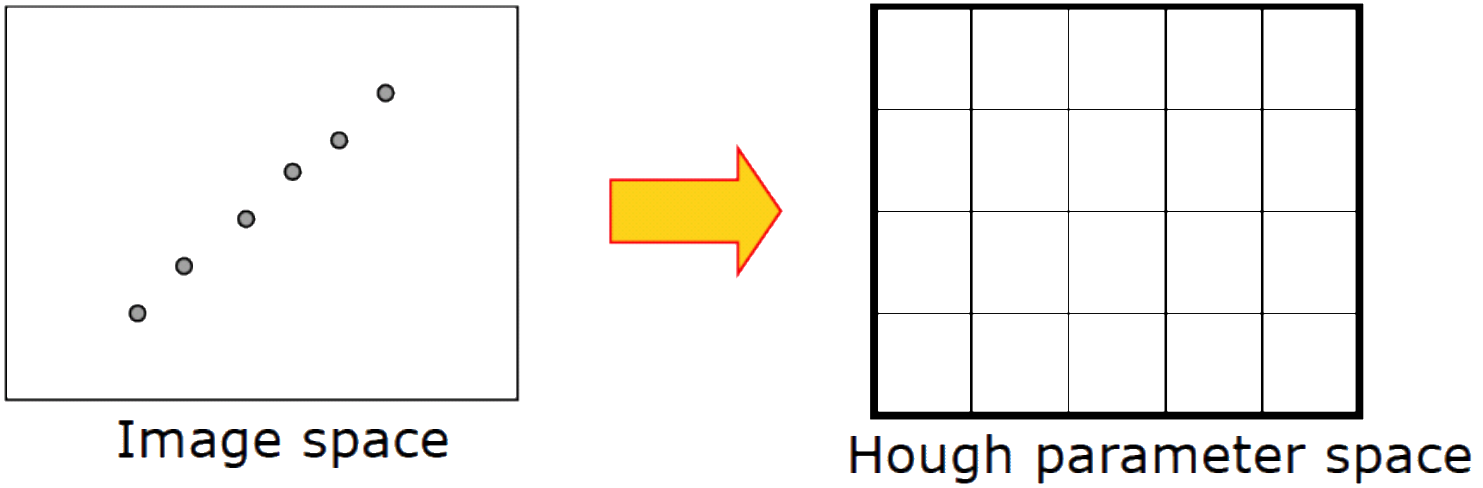
- Extract lines from a point cloud (e.g. range scan)
- Three main problems:
  - How many lines are there?
  - **Segmentation**: Which points belong to which line?
  - **Line Fitting/Extraction**: Given points that belong to a line, how to estimate the line parameters?

- Algorithms we will see:
  - Split-and-merge
  - Linear regression
  - RANSAC
  - Hough-Transform



Line Extraction – Hough-Transform

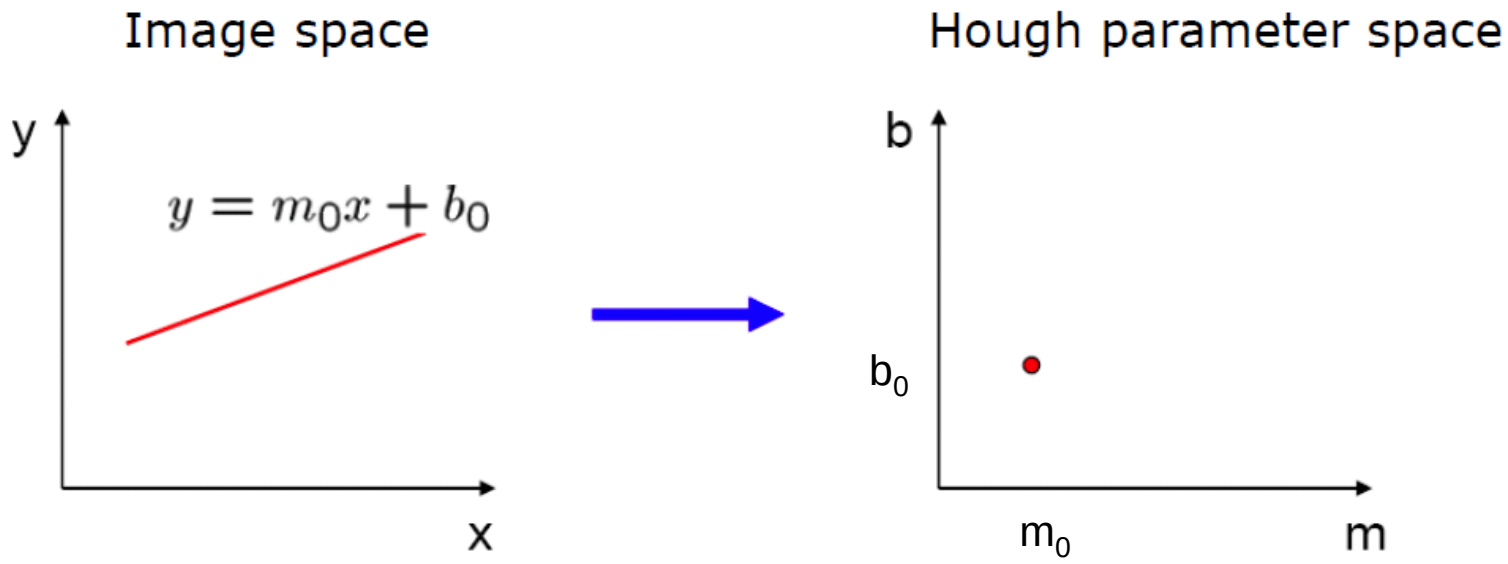
- Points vote for plausible line parameters
- Hough-Transform: maps image-space into Hough-space
- **Hough-space**: voting accummulator, parametrized w.r.t. line characteristics



1. P. Hough, Machine Analysis of Bubble Chamber Pictures, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959
2. J. Richard, O. Duda, P.E. Hart (April 1971). "Use of the Hough Transformation to Detect Lines and Curves in Pictures". Artificial Intelligence Center (SRI International)

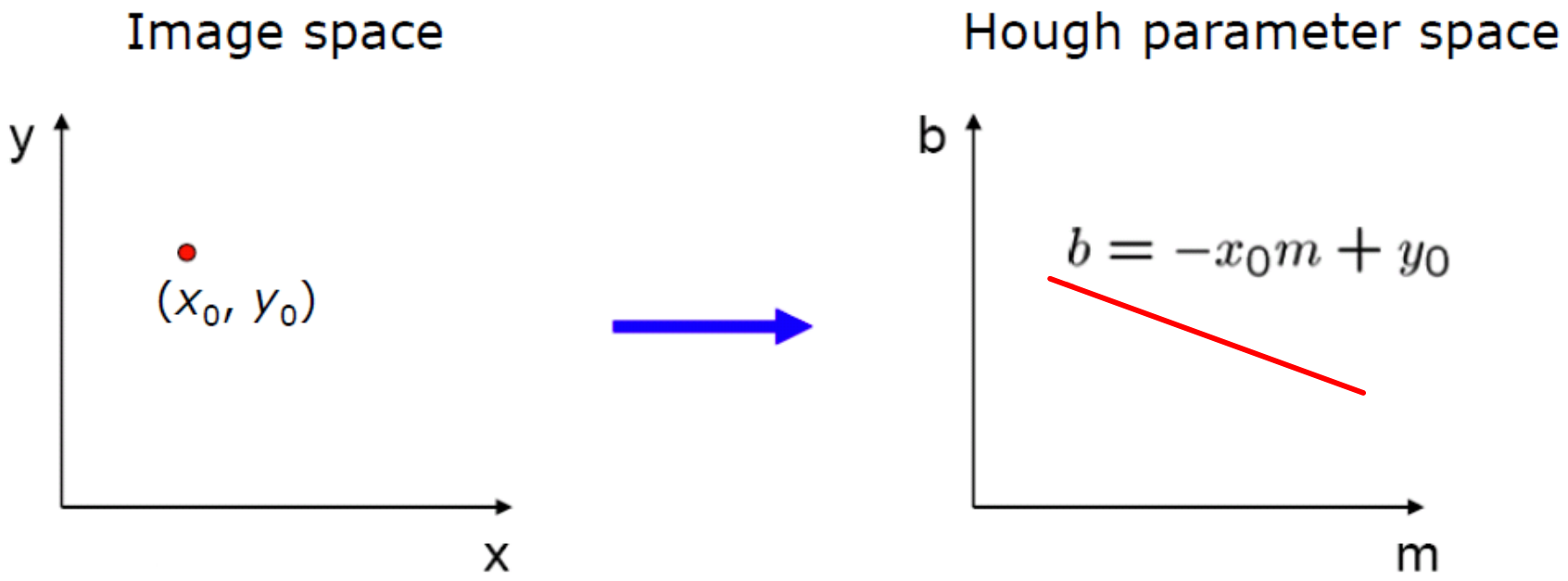
Line Extraction – Hough-Transform

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



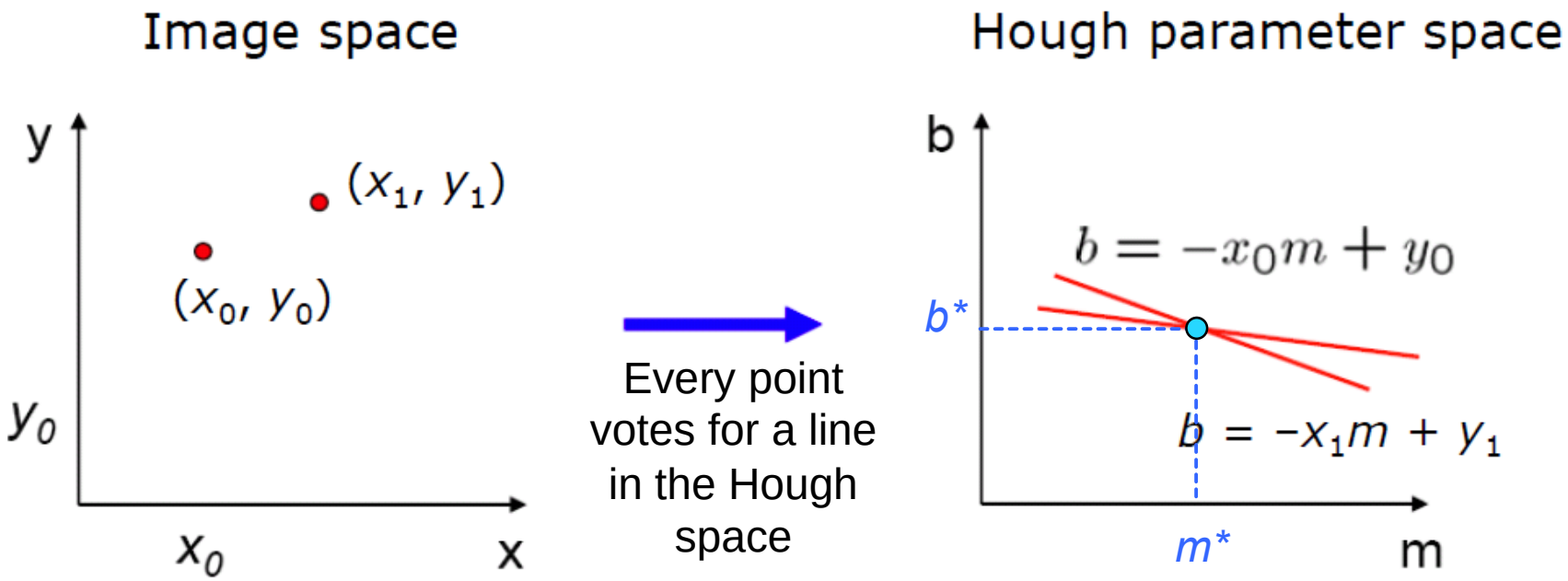
Line Extraction – Hough-Transform

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

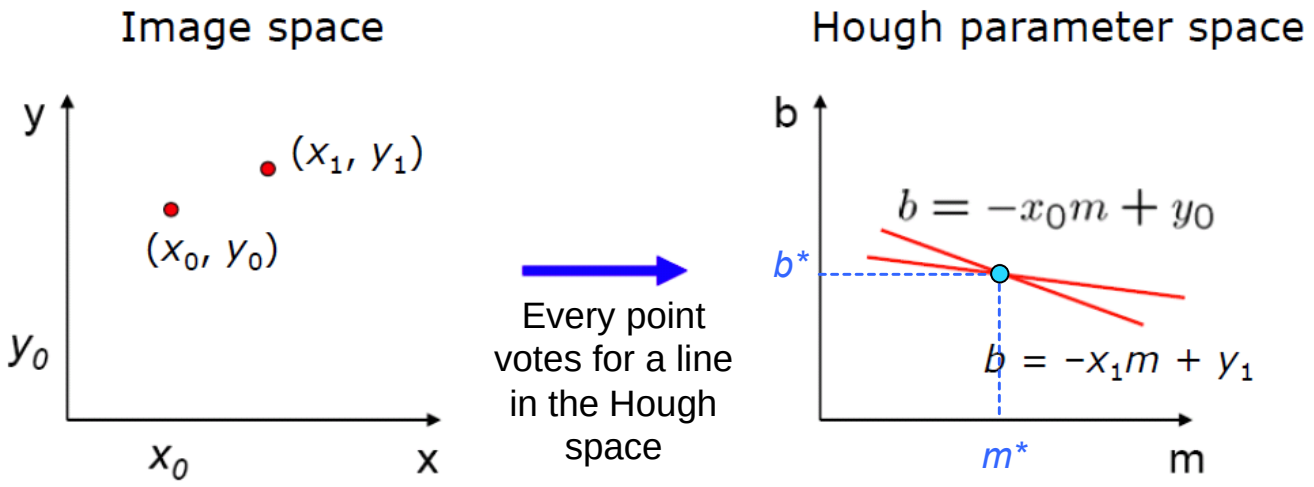


Line Extraction – Hough-Transform

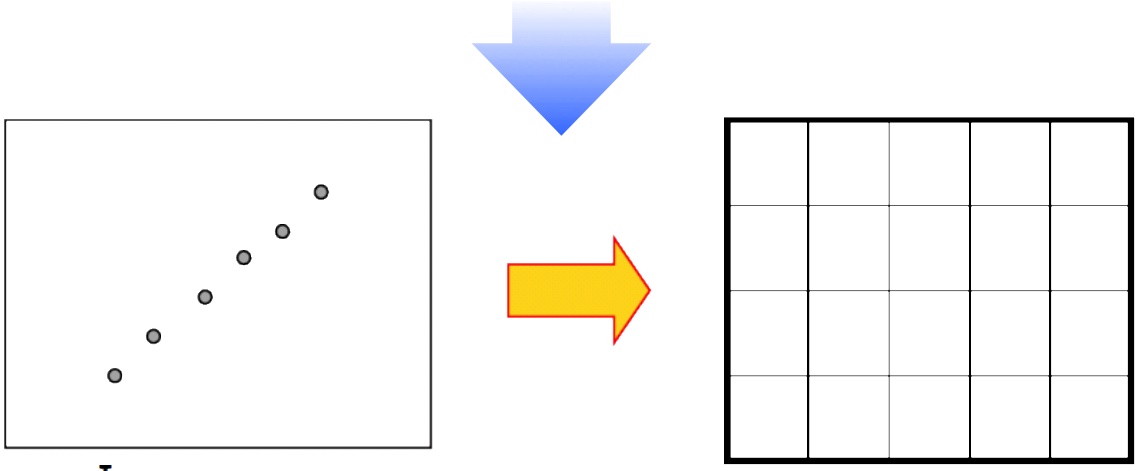
- Where is the line that contains both (x_0, y_0) and (x_1, y_1)
  - It is the intersection of the lines
    $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$



Line Extraction – Hough-Transform



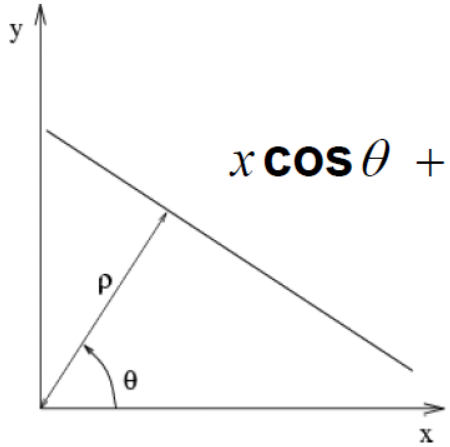
- Each point in image space votes for line-parameters in Hough parameter space



Line Extraction – Hough-Transform

- Problems with the *(m,b)* space:
  - Unbounded parameter domain
  - How to represent vertical lines?

- Alternative: polar representation



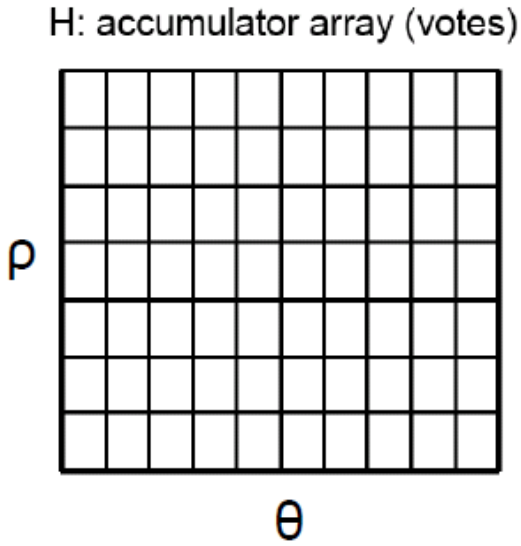
$$x\cos\theta + y\sin\theta = \rho$$

Each point in image space will map to a **sinusoid** in the (ρ,θ) parameter space

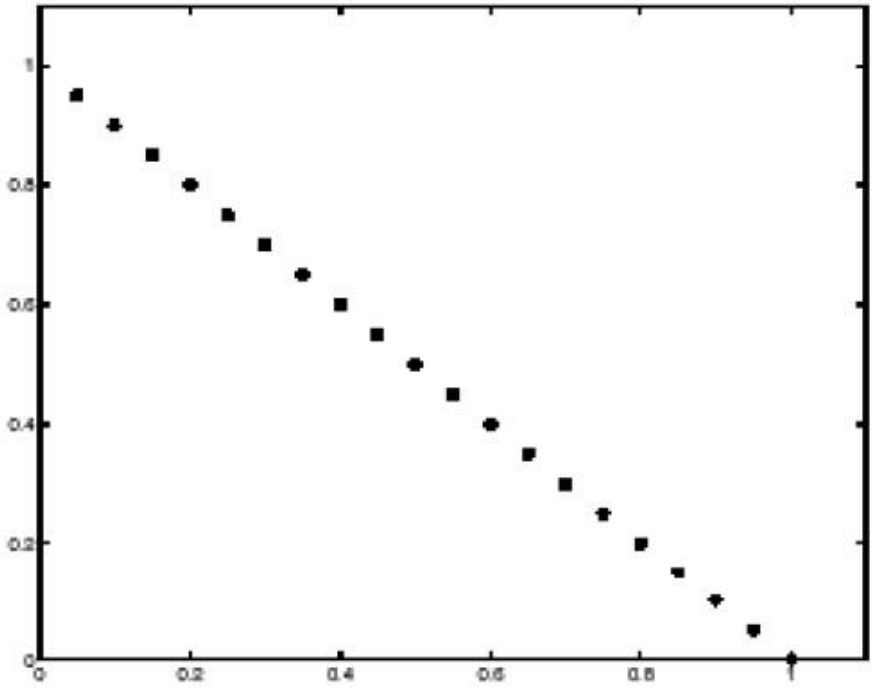
Line Extraction – Hough-Transform

1. Initialize accumulator H to all zeros
2. for each edge point (x,y) in the image
    for all θ in [0,180]
        Compute $\rho = x\,cos\theta + y\,sin\theta$
        $H(\theta, \rho) = H(\theta, \rho) + 1$
    end
end
3. Find the values of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
4. The detected line in the image is given by:

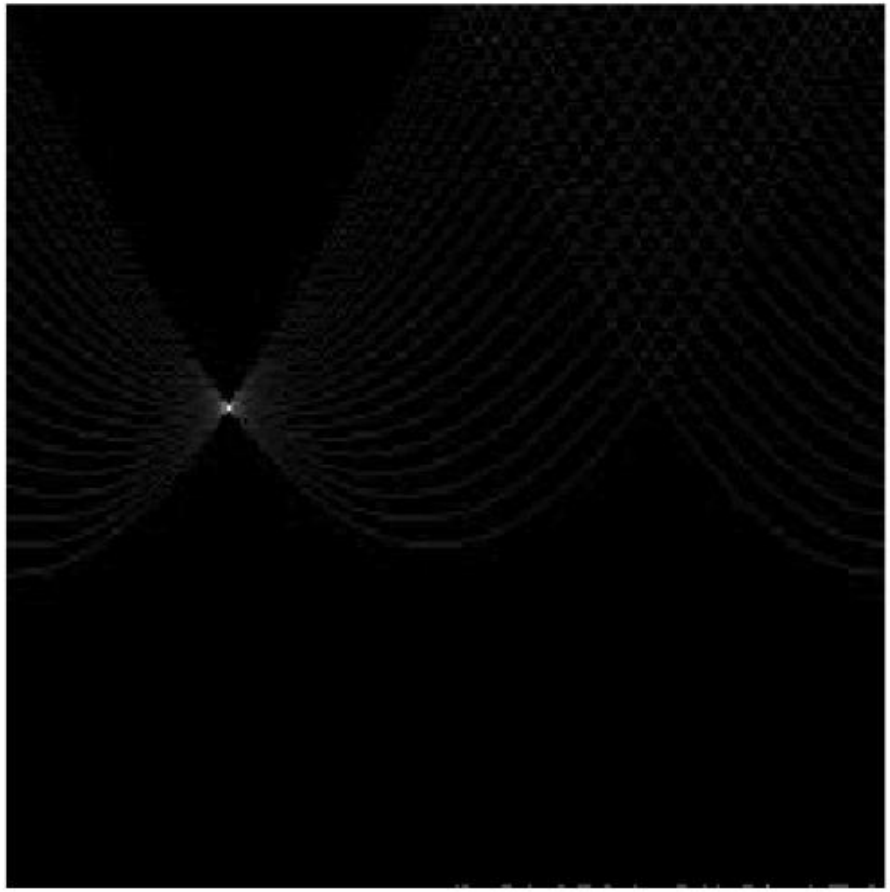
$$\rho = x\,cos\theta + y\,sin\theta$$



Line Extraction – Hough-Transform: Examples



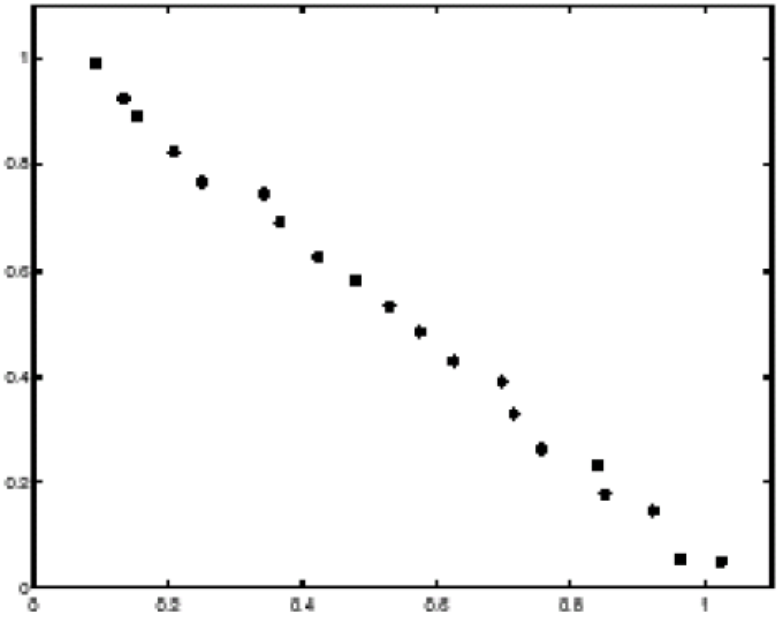
features



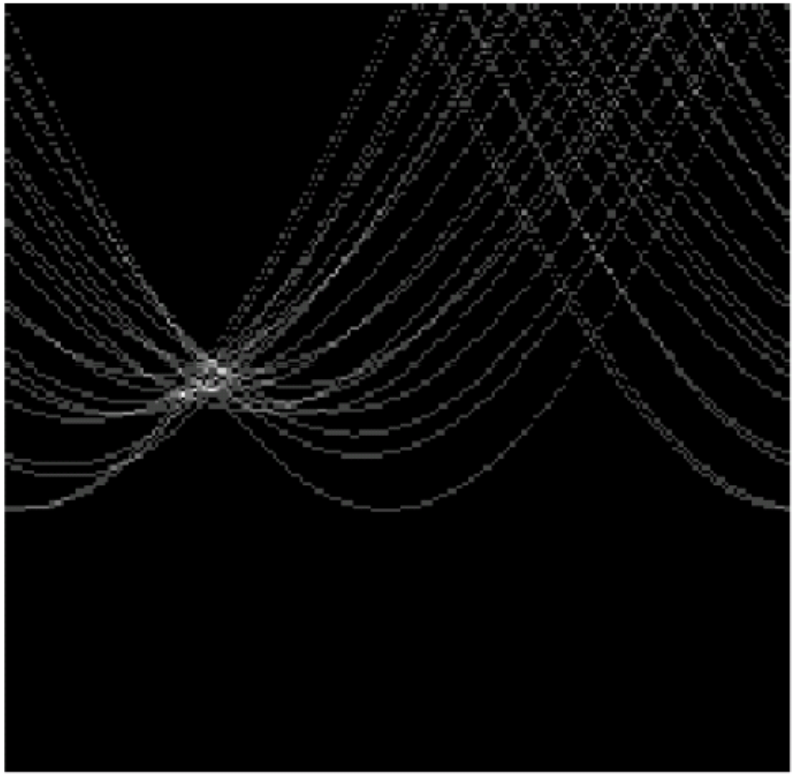
votes

Line Extraction – Hough-Transform: Examples

Effect of Noise:



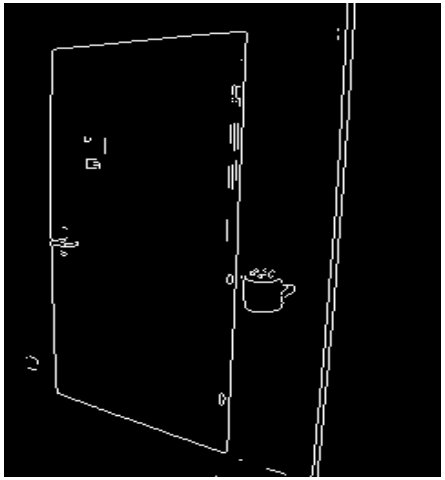
features

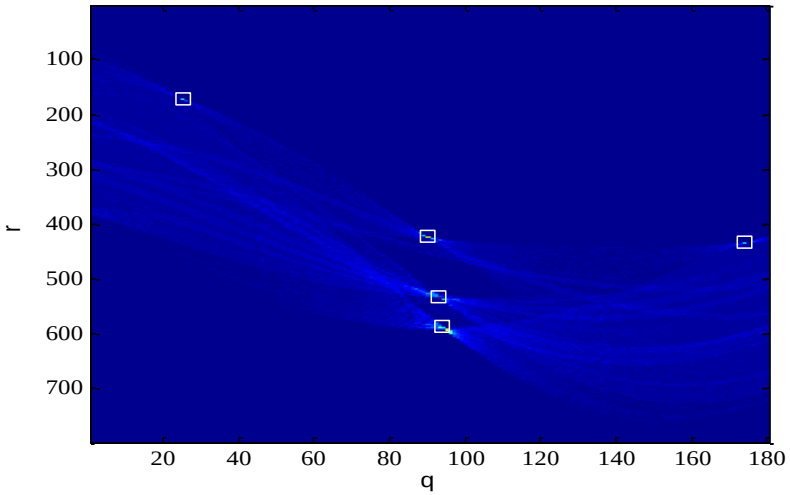


votes

- Peak gets fuzzy and hard to locate

Line Extraction – Hough-Transform: Examples





Hough Transform

Line Extraction – Comparison

- Split-and-merge, Incremental and Line-Regression: **fastest** – best applied on **laser scans**
- Deterministic & make use of the **sequential ordering** of raw scan points (: points captured according to the rotation direction of the laser beam)
- If applied **on randomly captured points** only last 3 algorithms would segment all lines.
- RANSAC, Hough-Transform and EM produce greater precision --> more robust to outliers

	Complexity	Speed (Hz)	False positives	Precision
Split-and-Merge	$N\,logN$	1500	10%	+++
Incremental	$S\,N$	600	6%	+++
Line-Regression	$N\,N_f$	400	10%	+++
RANSAC	$S\,N\,k$	30	30%	++++
Hough-Transform	$S\,N\,N_C + S\,N_R\,N_C$	10	30%	++++
Expectation Maximization	$S\,N_1\,N_2\,N$	1	50%	++++

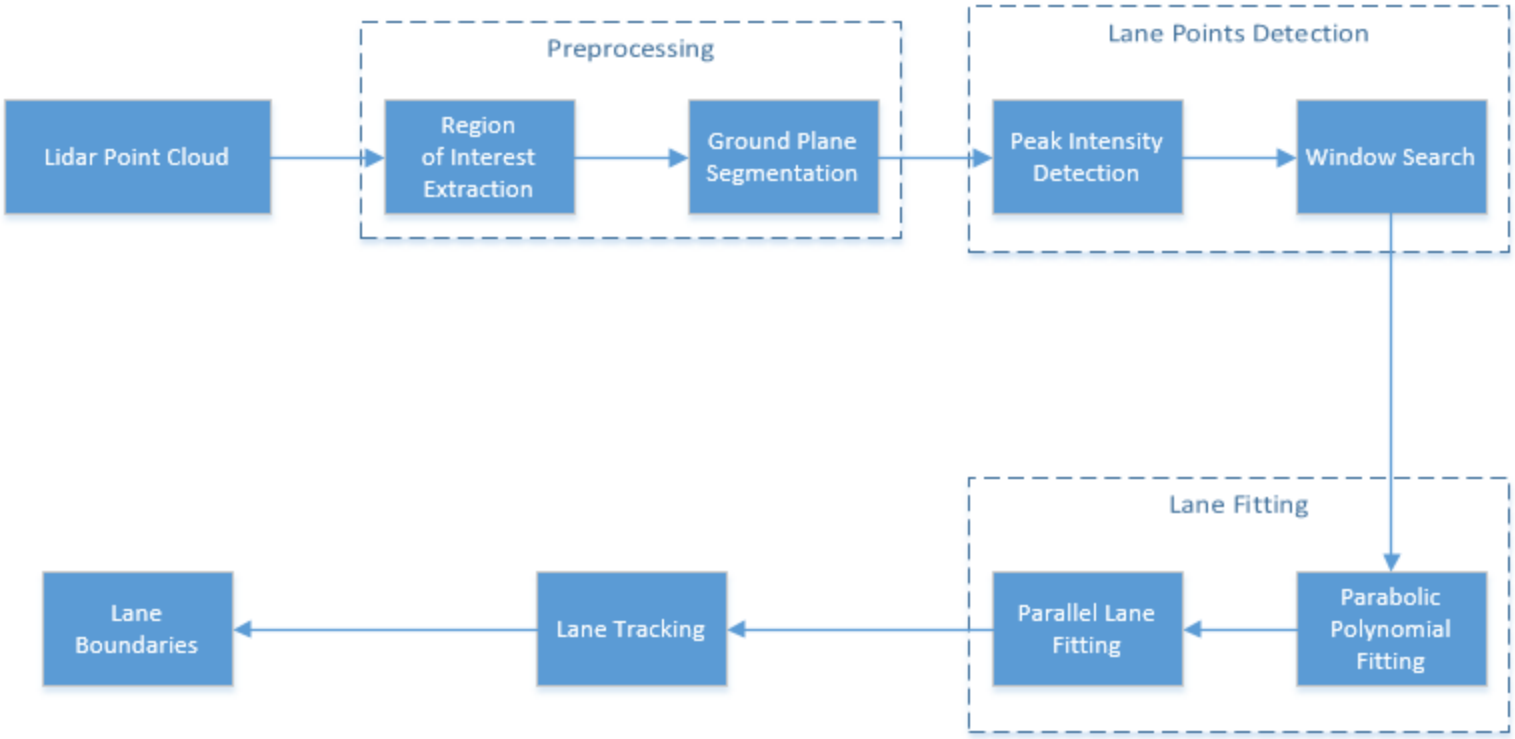
N : no. points considered
N_f : no. points in window
S : no. line-segments to be found
k : no. iterations
N_C, N_R : no columns, rows of the accumulator array

Comparison by [Nguyen et al. IROS 2005]

Lane Detection in 3D Lidar Point Cloud

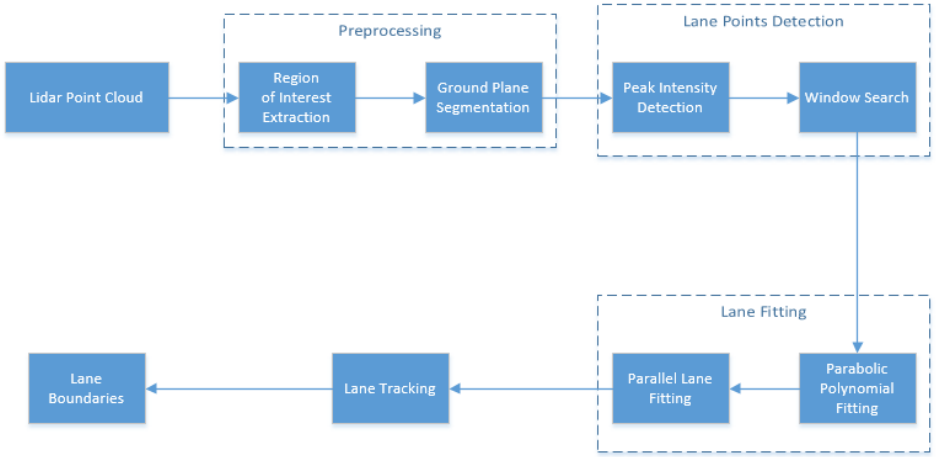
- The advantages of using Lidar data for lane detection are:
  - Lidar point clouds give a better 3D representation of the road surface than image data, thus reducing the required calibration parameters to find the bird's-eye view
  - Lidar is more robust against adverse climatic conditions than image-based detection
  - Lidar data has a centimeter level of accuracy, leading to accurate lane localization

Example:Lane Detection in 3D Lidar Point Cloud



Lane Detection in 3D Lidar Point Cloud

- Lane detection in Lidar involves detection of the immediate left and right lanes, also known as ego vehicle lanes, w.r.t. the Lidar sensor. It involves the following steps:
  - Region of interest extraction
  - Ground plane segmentation
  - Peak intensity detection
  - Lane detection using window search
  - Parabolic polynomial fitting
  - Parallel lane fitting
  - Lane tracking



Examples:

Lidar Toolbox:

- https://www.mathworks.com/help/lidar/
- What is Lidar:
  - https://www.mathworks.com/discovery/lidar.html

Examples:

- https://www.mathworks.com/help/lidar/examples.html
- Line extraction:
  - https://www.mathworks.com/help/images/ref/houghlines.html
- Lane detection in 3D Lidar point cloud
  - https://www.mathworks.com/help/lidar/ug/lane-detection-in-3d-lidar-point-cloud.html
- Curb detection and tracking in 3D Lidar point cloud
  - https://www.mathworks.com/help/lidar/ug/curb-detection-in-lidar-point-cloud.html

More Examples

- Detect and track vehicles using Lidar data
  - https://www.mathworks.com/help/vision/ug/track-vehicles-using-lidar.html
- Ground plane and obstacle detection using Lidar
  - https://www.mathworks.com/help/driving/ug/ground-plane-and-obstacle-detection-using-lidar.html
- Build a map from Lidar data
  - https://www.mathworks.com/help/driving/ug/build-a-map-from-lidar-data.html
- https://github.com/kaist-avelab/K-Lane

- Thank you!