Mobile Robotics, Perception: Line extraction based on range data

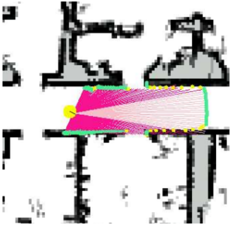
Material based on the book Autonomous Mobile Robot 2nd Ed. (Siegwart, Nourbakhsh, Scaramuzza) [AMR]; Chapter 4.7

Summary

- Introduction to line extraction
- Probabilistic line fitting [Chapter 4.7.1.1]
- Split and merge [Chapter 4.7.2.1]
- Line Regression [Chapter 4.7.2.2]
- RANSAC [Chapter 4.7.2.4]
- Hough transform [Chapter 4.7.2.5]

Line extraction from range data

- **Goal**: Extract geometric features from range data
  - e.g, point cloud
- Most features are geometric features
  - e.g, lines, corners, ...
- We focus on line
  - used for various tasks, e.g. laser scan matching in localization



Laser data (Source [PR])

Line fitting

◇ Geometric feature fitting: comparing and matching **measured data** against a pre-defined **description** of the feature

◇ Usually this is an overdetermined problem (number of data exceeeds number of template's parameter)

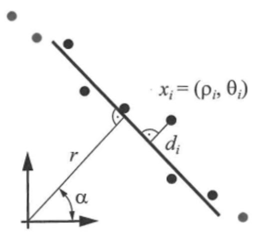
◇ Data are noisy

◇ Optimization problem: find parameters that minimize the discrepancy from data (Least-Squares estimation)

Probabilistic line fitting

- **Goal**: fit a line to a set of (noisy) measurements
- n **range** measurements in polar coordinates (ρ_i, θ_i)
- Consider the line perpendicular to (r, α)
- $d_i = \rho_i cos(\theta_i - \alpha) - r$
- Sum of squared errors $S = \sum_{i=1}^{n} (\rho_i cos(\theta_i - \alpha) - r)^2$
- we want to find r and α that minimize S

$$\frac{\delta S}{\delta r} = 0 \quad \frac{\delta S}{\delta \alpha} = 0$$



Line fitting in the least-square sense (Source [AMR])

Algorithms for line extractions

- main issues:
  - how many lines ?
  - which point belongs to which line ?
  - how can we estimate the line model parameter ?
- Algorithms we will consider
  - Split and Merge
  - Linear Regression
  - RANSAC
  - Hough transform

Split and Merge

- Split data-points to achieve better fit, merge to minimize number of lines

Data: Set of points S, Threshold th
Result: Set of lines
$L \leftarrow S$;
while L **is not empty do**
    $S_i = \text{pop}(L)$;
    $ln = \text{fitLine}(S_i)$;
    $d_p = \text{maxDist}(S_i, ln)$;
    if $d_p > th$ **then**
        Split S_i in S_{i1} and S_{i2};
        $L \leftarrow \text{push}(S_{i1})$;
        $L \leftarrow \text{push}(S_{i2})$;
    end
end
Merge collinear segments

Line Regression

◇ based on a sliding window of size N_f, at every step fit a line to N_f point; at the end check for merge.

Data: Sets of points S

Result: Set of lines

Initialize sliding window of N_f points;

foreach w **sliding window of** N_f **point do**

  fit a line to the points in w;

end

Merge collinear segments

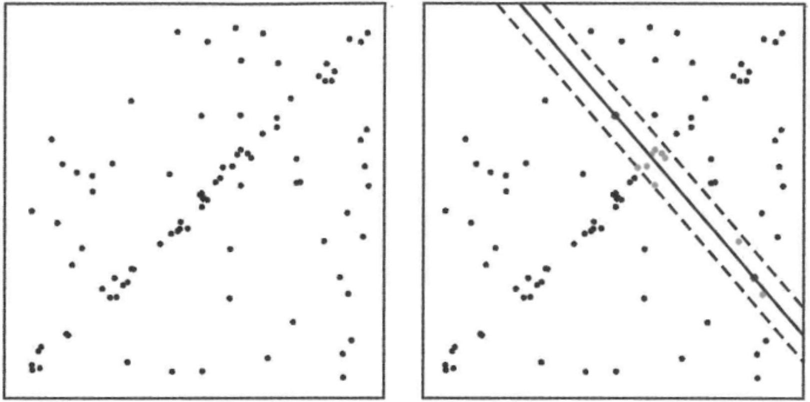
RANSAC

- **RANSAC** = RANdom SAmple Consensus
- Generic and robust fitting algorithm for models in presence of outliers
- **Goal**: identify **inlier** which satisfy a predefined model
- typical applications in robotics
  - line/plane extraction from 2D or 3D data
  - feature matching
  - structure from motion (image correspondence)
  - ...
- **Iterative** and **non deterministic**
  - the more iteration the higher the chance of removing outliers
- **Non determinism** $\Rightarrow$ results differ between runs

RANSAC: algorithm

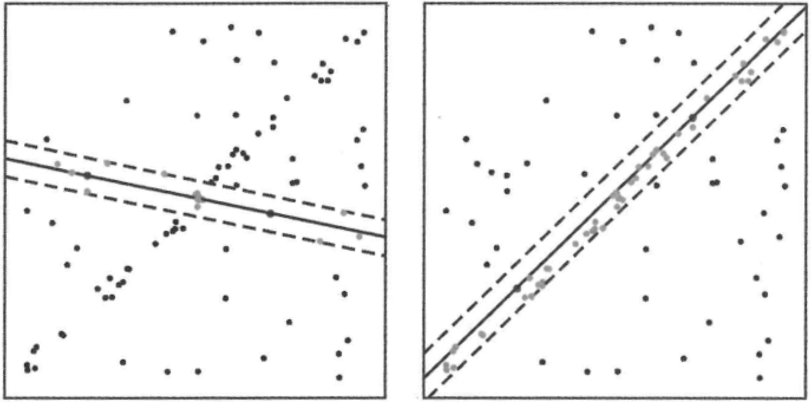
Data: Sets of points S, number of iterations k, threshold th
Result: one line
repeat
randomly select two points (p_1, p_2) from S;
fit a line l through (p_1, p_2);
Compute distance d_i of all other points in S from l;
InlierSet $\leftarrow$ all points for which $d_i < Th$;
Store the InlierSet
until **maximum number of iterations k reached**;
Return the line with maximum number of inlier as a solution;

RANSAC: example I



Visualization of the RANSAC algorithm applied to line extraction in 2D (Source [AMR])

RANSAC: example II



Visualization of the RANSAC algorithm applied to line extraction in 2D (Source [AMR])

RANSAC: notes

◇ We can not know the maximum number of inliers, how do we set k ?

- check all possible combinations of 2 points over n
- $\frac{n(n-1)}{2}$
- not feasible for large n

◇ We can exploit a rough estimate of the percentage of inliers to compute k using probability

RANSAC: determining k

- w percentage of inliers (number of inliers/n)
  - w = P(selecting an inlier-point out of the dataset)
- Let p = P(selecting a set of points free of outliers)
- **Assumption**: the 2 points to estimate a line are selected independently
  - w^2 = P(both selected points are inliers)
  - $1 - w^2$ = P(at least one of these two points is an outlier)
- Let k = number of RANSAC iterations executed so far
  - $(1 - w^2)^k$ = P(RANSAC never selects two points that are both inliers)
  - $1 - p = (1 - w^2)^k$

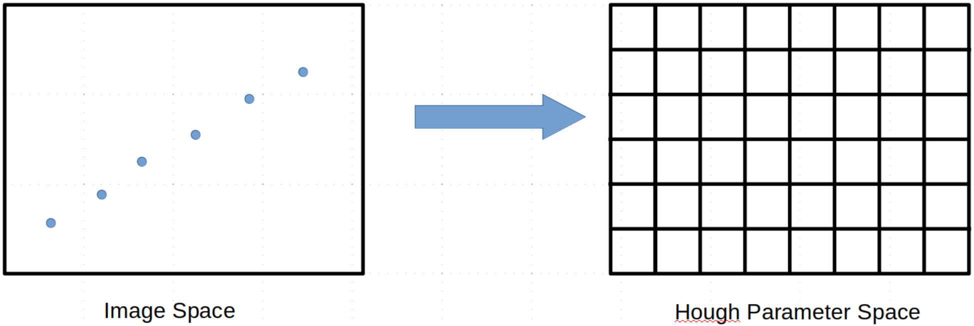
$$k = \frac{log(1-p)}{log(1-w^2)}$$

RANSAC: discussion

- $k = \frac{log(1-p)}{log(1-w^2)}$ can be used to compute k given p and w
  - assume we want a probability of finding a set free of outlier $p = 99\%$
  - assume we estimate our dataset to have a percentage of inlier $w = 50\%$
  - $k = 16$
- If we want to retrieve more than one model (i.e., more than one line) we can re-run RANSAC removing points that have been assigned to lines.
- **Drawback**: no guarantee of optimality
  - number of inliers
  - best fit

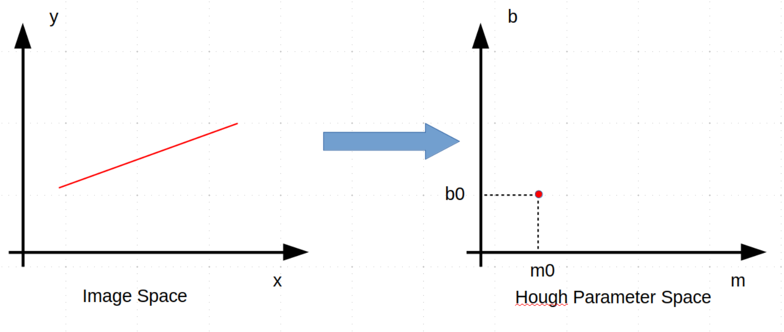
Hough Transform

- Consider parameter space for lines
- points vote for plausible line parameters
- **Hough transform**: maps data-space to Hough-space
- **Hough-space**: accumulates votes for line parameters



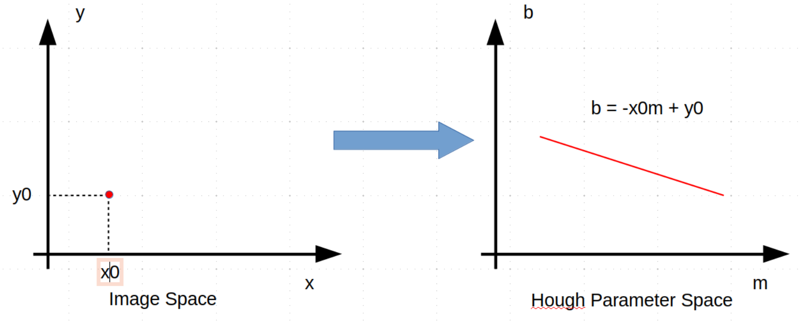
Hough Transform: from lines to points

- Transforming a **line** from the data space we obtain a **point** in the hough-space
- $y = m_0 \cdot x + b_0 \Rightarrow (m_0, b_0)$



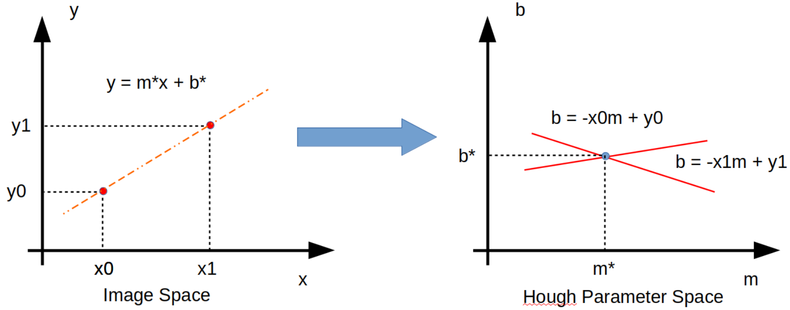
Hough Transform: from points to lines

- Transforming a **point** from the data space we obtain a **line** in the hough-space
- $(x_0, y_0) \Rightarrow b_0 = -x_0 \cdot m + y_0$



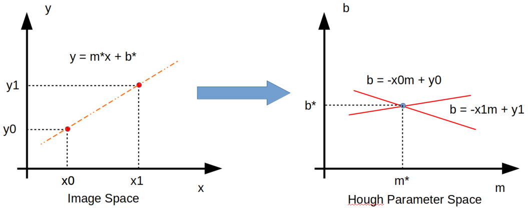
Hough Transform: multiple points

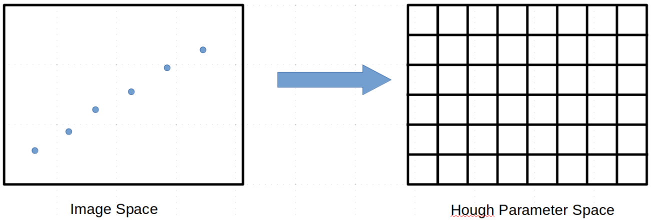
- **data-space**: line that contains both (x_0, y_0) and (x_1, y_1)
- **hough parameter space**: intersection of $b = -x_0 \cdot m + y0$ and $b = -x_1 \cdot m + y1$



Hough Transform: voting

- Each point in data space votes for line parameters in Hough space





Hough Transform: issues with (m, b) space

- unbounded parameter domain
- how to represent lines that are aligned with axes
- can use **polar representation** ρ, θ

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

- every point in the data-space maps to a **sinusoid** in the (ρ, θ) parameter space

Hough transform: algorithm

Data: Sets of points S, Number of rows N_r and columns N_c for accumulator, Threshold th

Result: one line

Initialize accumulator H to all zeros;

for point $p = (x, y) \in S$ **do**

  foreach θ **do**

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

    $H(\theta, \rho) += 1$;

    Store point p;

  end

end

$(\theta^*, \rho^*) = \arg\max H(\theta, \rho)$;

if $H(\theta^*, \rho^*) > th$ **then**

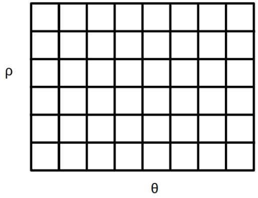
  Inliers = all points that voted for cell $H(\theta^*, \rho^*)$;

  ln = fitLine($inliers$);

  Return ln;

end

H: accumulator array (votes)



Comparison of line extraction algorithm

- empirical evaluation of widely used algorithms Nguyen et. al IROS 2005
  - N number of points in dataset (e.g., 722)
  - S number of line segments extracted (e.g., 7 on average)
  - N_f sliding window size (e.g., 9)
  - N_{tirals} number of trials for RANSAC (e.g., 1000)
  - N_R, N_C number of rows, columns for the Hough accumulator (e.g., $N_R = 401$, $N_R = 671$)

	Complexity	Speed (Hz)	False Positive	Precision
Split and Merge	$N \cdot logN$	1500	10%	+++
Line Regression	$N \cdot N_f$	400	10%	+++
RANSAC	$S \cdot N \cdot N_{\text{trials}}$	30	30%	++++
Hough transform	$S \cdot N \cdot N_C + S \cdot N_R \cdot N_C$	10	30%	++++