Perception | Line Extraction
Autonomous Mobile Robots

Margarita Chli – University of Edinburgh
Paul Furgale, Marco Hutter, Martin Rufli, Davide Scaramuzza, Roland Siegwart
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:
  1. Split-and-merge
  2. RANSAC
  3. Hough-Transform
Line Extraction | split-and-merge (standard)

- Originates from Computer Vision.
- A recursive procedure of fitting and splitting.
- A slightly different version, called Iterative end-point-fit, simply connects the end points for line fitting.

Initialise set $S$ to contain all points

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

**Merge**
- If two consecutive segments are close/collinear enough, obtain the common line and find the most distant point
- If distance $\leq \text{threshold}$, merge both segments
Line Extraction | split-and-merge (standard)

- Originates from Computer Vision.
- A recursive procedure of fitting and splitting.
- A slightly different version, called Iterative end-point-fit, simply connects the end points for line fitting.

Initialise set S to contain all points

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

**Merge**
- If two consecutive segments are close/collinear enough, obtain the common line and find the most distant point
- If distance \( \leq \) threshold, merge both segments
Line Extraction | split-and-merge (standard)

- Originates from Computer Vision.
- A recursive procedure of fitting and splitting.
- A slightly different version, called Iterative end-point-fit, simply connects the end points for line fitting.

Initialise set $S$ to contain all points

**Split**
- Fit a line to points in current set $S$
- Find the most distant point to the line
- If distance $> \text{threshold} \Rightarrow$ split set & repeat with left & right sets

**Merge**
- If two consecutive segments are close/collinear enough, obtain the common line and find the most distant point
- If distance $\leq \text{threshold}$, merge both segments
Line Extraction | split-and-merge (standard)

- Originates from Computer Vision.
- A recursive procedure of fitting and splitting.
- A slightly different version, called Iterative end-point-fit, simply connects the end points for line fitting.

Initialise set $S$ to contain all points

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

**Merge**
- If two consecutive segments are close/collinear enough, obtain the common line and find the most distant point
- If distance $\leq$ threshold, merge both segments
Line Extraction | split-and-merge (standard)

- Originates from Computer Vision.
- A recursive procedure of fitting and splitting.
- A slightly different version, called Iterative end-point-fit, simply connects the end points for line fitting.

Initialise set \( S \) to contain all points

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

**Merge**
- If two consecutive segments are close/collinear enough, obtain the common line and find the most distant point
- If distance \( \leq \) threshold, merge both segments
Split-and-Merge | iterative end-point-fit

- Iterative end-point-fit: simply connects the end points for line fitting.
Split-and-Merge | iterative end-point-fit

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

Split and Merge | iterative end-point-fit
Split-and-Merge | iterative end-point-fit

- Iterative end-point-fit: simply connects the end points for line fitting.
Split-and-Merge | iterative end-point-fit

- Iterative end-point-fit: simply connects the end points for line fitting.
Split-and-Merge | iterative end-point-fit

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

Applicable to any problem where the goal is to identify
the inliers which satisfy a predefined model.

Typical applications in robotics are:
line/plane extraction, feature matching, structure from motion, …

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.
RANSAC | how it works
RANSAC | how it works

- Select sample of 2 points at random
RANSAC | how it works

- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
RANSAC | how it works

- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
RANSAC | how it works

- Select sample of 2 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
RANSAC | how it works

- Select sample of 2 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
RANSAC | how it works

- Select sample of 2 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
RANSAC | how it works

Set with the maximum number of inliers obtained within $k$ iterations
We cannot know in advance if the observed set contains the max. no. inliers
ideally: check all possible combinations of 2 points in a dataset of N points.

No. all pairwise combinations: \( N(N-1)/2 \)
computationally infeasible if N is too large.
example: laser scan of 360 points ⇒ need to check all 360*359/2 = 64'620 possibilities!

Do we really need to check all possibilities or can we stop RANSAC after 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
RANSAC | how many iterations?

- \( w := \text{number of inliers} / N \)
- \( N := \text{tot. no. data points} \)
- \( w : \text{fraction of inliers in the dataset} \Rightarrow w = P(\text{selecting an inlier-point from the dataset}) \)

- Let \( p := P(\text{selecting a minimal set of points free of outliers}) \)

- Assumption: the 2 points necessary to estimate a line are selected independently
  - \( ? = P(\text{both selected points are inliers}) \)
  - \( ? = P(\text{at least one of these two points is an outlier}) \)
RANSAC | how many iterations?

- $w := \text{number of inliers} / N$
  - $N := \text{tot. no. data points}$
  - $w : \text{fraction of inliers in the dataset} \Rightarrow w = P(\text{selecting an inlier-point from the dataset})$

- Let $p := P(\text{selecting a minimal set of points free of outliers})$

- Assumption: the 2 points necessary to estimate a line are selected independently
  - $w^2 = P(\text{both selected points are inliers})$
  - $1 - w^2 = P(\text{at least one of these two points is an outlier})$

- Let $k := \text{no. RANSAC iterations executed so far}$
  - $? = P(\text{RANSAC never selects two points that are both inliers})$
RANSAC | how many iterations?

- \( w := \text{number of inliers} / N \)
- \( N := \text{tot. no. data points} \)
- \( w : \text{fraction of inliers in the dataset} \Rightarrow w = P(\text{selecting an inlier-point from the dataset}) \)

- Let \( p := P(\text{selecting a minimal set of points free of outliers}) \)

- Assumption: the 2 points necessary to estimate a line are selected independently
  - \( w^2 = P(\text{both selected points are inliers}) \)
  - \( 1-w^2 = P(\text{at least one of these two points is an outlier}) \)

- Let \( k := \text{no. RANSAC iterations executed so far} \)
  - \( (1-w^2)^k = P(\text{RANSAC never selects two points that are both inliers}) \)
  - \( 1-p = (1-w^2)^k \) and therefore:
    \[
    k = \frac{\log(1-p)}{\log(1-w^2)}
    \]

- In practice we need only a rough estimate of \( w \).
  More advanced variants of RANSAC estimate the fraction of inliers & adaptively set it on every iteration.
Line Extraction | Hough-transform

- Edges **vote** for plausible line locations
- Map image space into Hough parameter space
- Hough space parameterizes coordinate space w.r.t line characteristics
- In practice, it’s a discretized accumulator array (comprising of voting bins)
A line in the image corresponds to a point in Hough space.

What does a point \((x_0, y_0)\) in the image space map to in the Hough space?
A line in the image corresponds to a point in Hough space.

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

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

- At the intersection of: \(b = -x_0m + y_0\) and \(b = -x_1m + y_1\)

\[ y = m_0x + b_0 \]
Hough-Transform | how it works

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

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

- Alternative: polar representation

Each point in image space will map to a sinusoid in the \((\rho, \theta)\) parameter space.
Line Extraction | relative merits

- Split-and-merge: fastest
  - Deterministic & makes 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 RANSAC and Hough-Transform would segment all lines.

- RANSAC and Hough-Transform: more robust to outliers

Courtesy of F. Pomerleau and F. Colas