Outline

• Filters for feature extraction
• Point-feature extraction
• Line extraction algorithms
Line extraction

• Supose that you have been commissioned to implement a lane detection for a car driving-assistance system
• How would you proceed?
Line extraction

- How do we extract lines from edges?
Two popular line extraction algorithms

• Hough transform (used also to detect circles, ellipses, and any sort of shape)
• RANSAC (Random Sample Consensus)
Hough-Transform

• Finds lines from a binary edge image using a voting procedure
• The voting space (or accumulator) is called Hough space

Let \((x_0, y_0)\) be an image point.

We can represent all the lines passing through it by \(y_0 = mx_0 + b\).

The Hough transform works by parameterizing this expression in terms of \(m\) and \(b\):

\[b = -x_0m + y_0\]

This is represented by a line in the Hough space.
Let $(x_0, y_0)$ be an image point.

We can represent all the lines passing through it by $y_0 = mx_0 + b$.

The Hough transform works by parameterizing this expression in terms of $m$ and $b$: $b = -x_0m + y_0$.

This is represented by a line in the Hough space.
Hough-Transform

- How do we compute the line \((b^*, m^*)\) that contains both \((x_0, y_0)\) and \((x_1, y_1)\)?
  - It is the intersection of the lines \(b = -x_0m + y_0\) and \(b = -x_1m + y_1\)

\[
\begin{align*}
\text{Image space} & \quad \text{Hough parameter space} \\
(x_0, y_0) & \quad b^* \\
(x_1, y_1) & \quad b = -x_0m + y_0 \\
\end{align*}
\]
Hough-Transform

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

\[ b = -x_1 m + y_1 \]
\[ b = -x_0 m + y_0 \]
Hough-Transform

• Problems with the \((m, b)\) space:
  – Unbounded parameter domain
    • \(m, b\) can assume any value in \([-\infty, +\infty]\)
Hough-Transform

• Problems with the \((m, b)\) space:
  – Unbounded parameter domain
    • \(m, b\) can assume any value in \([-\infty, +\infty]\)

• Alternative line representation: polar representation \(\rightarrow\) Bounded parameter domain! Can you tell what the bounds are?

\[\rho = x \cos \theta + y \sin \theta\]
Hough-Transform

- Each point in image space will map to a sinusoid in the \((\rho, \theta)\) parameter space

\[
\rho = x \cos \theta + y \sin \theta
\]
Hough-Transform

- Each point in image space will map to a sinusoid in the \((\rho, \theta)\) parameter space

\[
\rho = x \cos \theta + y \sin \theta
\]
Hough-Transform Algorithm

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 $(\theta, \rho)$ where $H(\theta, \rho)$ is a local maximum

4. The detected line in the image is given by: $\rho = x \cos \theta + y \sin \theta$
Examples
Examples

Hough Transform
Examples
Examples

features

votes
Problems: Noise

- Effects of noise: peaks get fuzzy and hard to locate
- How to overcome this?
  - Increase bin size (decrease resolution of the Hough space); however, this reduces the accuracy of the line parameters
  - Smooth the Hough image with a box or Gaussian filter; why?
Problems: Outliers

- How would Hough work with this type of data?
RANSAC (RAndom SAmple Consensus)

• RANSAC has become the standard method for model fitting in the presence of outliers (very noisy points or wrong data)
• It can be applied to line fitting but also to thousands of different problems where the goal is to estimate the parameters of a model from the data (e.g., camera calibration, structure from motion, DLT, homography, etc.)
• We will see many examples of RANSAC applications in the next lecture
• Let’s now focus on line extraction

RANSAC
RANSAC

- Select sample of 2 points at random
RANSAC

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

- 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

- 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

• 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

• 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

Set with the maximum number of inliers obtained within $k$ iterations
RANSAC

How many iterations does RANSAC need?

• Ideally: check all possible combinations of 2 points in a dataset of \( N \) points.

• No. all pairwise combinations: \( N(N-1)/2 \)
  \( \Rightarrow \) computationally unfeasible if \( N \) is too large.
  example: 1000 edge points \( \Rightarrow \) need to check all 1000*999/2= 500'000 possibilities!

• Do we really need to check all possibilities or can we stop RANSAC 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
RANSAC

How many iterations does RANSAC need?

- \( w := \text{number of inliers}/N \)
  - \( N := \text{total number of data points} \)
    - \( w \) : fraction of inliers in the dataset \( \Rightarrow w = P(\text{selecting an inlier-point out of the dataset}) \)

- 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 \( k := \text{no. RANSAC iterations executed so far} \)
  - \( (1-w^2)^k = P(\text{RANSAC never selected two points that are both inliers}) \)

- Let \( p := P(\text{probability of success}) \)
  - \( 1-p = (1-w^2)^k \) and therefore:

\[
k = \frac{\log(1-p)}{\log(1-w^2)}
\]
RANSAC

How many iterations does RANSAC need?

• The number of iterations $k$ is

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

• Knowing the fraction of inliers $w$, after $k$ RANSAC iterations we will have a probability $p$ of finding a set of points free of outliers

• Example: if we want a probability of success $p=99\%$ and we know that $w=50\% \iff k=16$ iterations – these are dramatically fewer than the number of all possible combinations! As you can see, the number of points does not influence the estimated number of iterations, only $w$ does!

• In practice we only need a rough estimate of $w$. More advanced variants of RANSAC estimate the fraction of inliers and adaptively update it at every iteration
RANSAC

1. Initial: let $A$ be a set of $N$ points
2. repeat
3. Randomly select a sample of 2 points from $A$
4. Fit a line through the 2 points
5. Compute the distances of all other points to this line
6. Construct the inlier set (i.e. count the number of points whose distance $< d$)
7. Store these inliers
8. until maximum number of iterations $k$ reached
9. The set with the maximum number of inliers is chosen as a solution to the problem
RANSAC: applications

- **RANSAC** = **RAN**dom **SA**mple **C**onsensus.

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

- Generally applicable algorithm 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** ⇒ the probability to find a set free of outliers increases as more iterations are used

- `RANSAC` is **non-deterministic** ⇒ results are different between runs