Homework 4

Due Date: Sunday, 8 December 2024

Acknowledgement: These problems are adapted from "Principles of Robot Autonomy I", CS237A/EE260A, Stanford University. We thank them for making these course materials available to us.

Part A: Image Filtering

Problem 1

Given an image $I \in \mathbb{R}^{m \times n}$ represented as an m x n matrix, each element $I(i,j) \in [0, 255]$ denotes the grayscale value of a pixel and the indices I = 0, …., m – 1 and j = 0, …., n – 1 denotes the position of the corresponding pixel. The **correlation operator** $G = F \otimes I$ produces an image G whose pixels G(I, j) are a sum of the original image I's pixels weighted by the entries in a given filter $F \in \mathbb{R}^{k \times l}$. Specifically given a filter $F \in \mathbb{R}^{k \times l}$, the output image $G \in \mathbb{R}^{m \times n}$ is defined pixelwise as,

$$G(i,j) = \sum_{u=0}^{k-1} \sum_{v=0}^{l-1} F(u,v) \,.\, \bar{I}(i+u, j+v),$$

Where $\bar{I} \in \mathbb{R}^{(m+k-1) \times (n+l-1)}$ is the original image, I, padded with zeros along it's edges. In this problem consider the image $I \in \mathbb{R}^{3 \times 3}$ and its corresponding zero-padded image $\bar{I} \in \mathbb{R}^{5 \times 5}$

$$I = \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}, with\ \bar{I} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 4 & 1 & 0 \\ 0 & 8 & 5 & 2 & 0 \\ 0 & 9 & 6 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Manually calculate the output image $G \in \mathbb{R}^{3 \times 3}$ as a result of $G = F \otimes I$ when the filter F is:

a) $F = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

b) $F = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

c) $F = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Briefly describe what the filter in part (c) is doing to the image.

d) Let, $F' = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$. What is the difference in functionality of filter F' and F of part (c)?

e) $F = \frac{1}{16}\begin{bmatrix} 1\,2\,1 \\ 2\,4\,2 \\ 1\,2\,1 \end{bmatrix}$

Briefly describe what the filter in part (e) is doing to the image.

f) $F = \frac{1}{9}\begin{bmatrix} 1\,1\,1 \\ 1\,1\,1 \\ 1\,1\,1 \end{bmatrix}$

What is the difference in functionality of the filters in part (e) and (f)?

Problem 2

a) Show that we can write correlation as a vector dot product,

$$\underline{G(i,j) = f^T t_{ij}}$$

Where f is the vector representation of the filter and t_{ij} is the vector representation of the neighborhood patch of the image.

b) Implement the correlation operation in Matlab or Python which takes a filter F and a grayscale image I and outputs G, where $G = F \otimes I$. Specifically, implement it using the dot product result of part (a), otherwise it will be too slow. Remember to implement zero-padding to maintain correct sizes.

Test your code using the provided image "parrot.png" and the filters in problem 1.

Part B: Line Extraction

Problem 3

In this problem, you will implement a line extraction algorithm to fit lines to Lidar range data. Consider the overhead view of a typical indoor environment shown in Figure 1.

A mobile robot needs to explore and map this using a 2D lidar sensor, which casts equally spaced laser beams from the center of the robot to form a 360° view. The first step in mapping is to extract meaningful information from the range measurements. Line extraction is a common technique used to fit a series of straight-line segments to range data in an attempt to define the border of objects in the environment.

Line Fitting

A range scan describes a 2D slice of the environment. Points in a range scan are specified in a polar coordinate system with the origin at the location of the sensor. It is common to assume that the noise on measurements follow a Gaussian distribution with zero mean, some range variance and negligible angular uncertainty. In polar coordinates the equation of a line is given by equation 1,

$$\frac{r}{\rho} = \cos(\theta - \alpha) \qquad (1)$$

Where $-\pi < \alpha \leq \pi$ is the angle between the x-axis and the shortest connection between the origin and the line, the perpendicular distance r. See Figure 2.

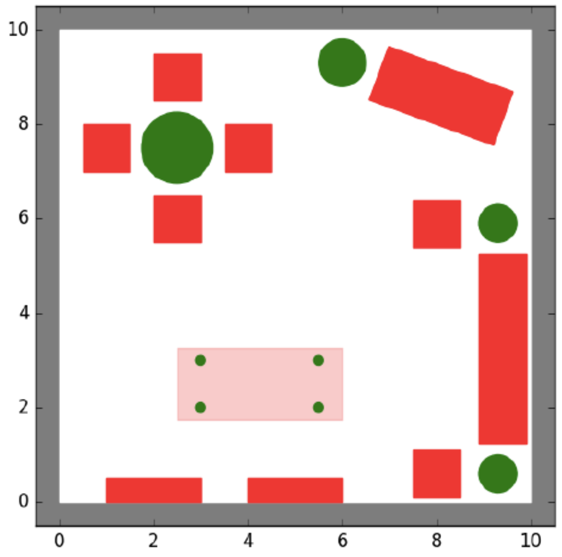

Figure 1

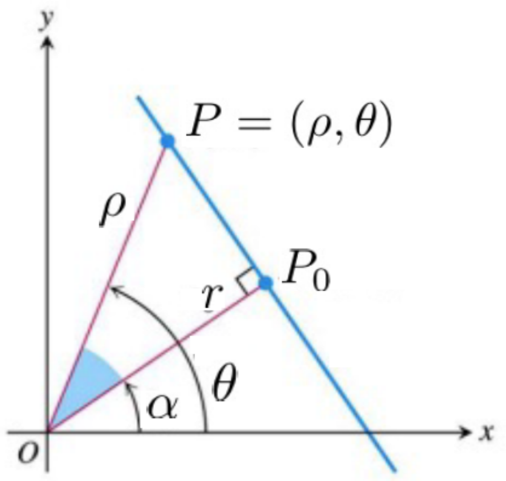

Figure 2

The goal of line fitting in polar coordinates is to estimate the parameters (r, α) given a set of n data points (ρ_i, θ_i). See Figure 3. (r, α) is estimated by minimizing the distance error function given by equation 2,

$$S = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} (\rho_i \cos(\theta_i - \alpha) - r)^2 \qquad (2)$$

The solution of this least squares problem gives the line parameters:

$$\alpha = \frac{1}{2}\text{arctan2}\left(\frac{\sum_i^n \rho_i^2 \sin 2\theta_i - \frac{2}{n}\sum_i^n \sum_j^n \rho_i\rho_j \cos\theta_i \sin\theta_j}{\sum_i^n \rho_i^2 \cos 2\theta_i - \frac{1}{n}\sum_i^n \sum_j^n \rho_i\rho_j \cos(\theta_i+\theta_j)}\right) + \frac{\pi}{2}, \qquad r = \frac{1}{n}\sum_i^n \rho_i \cos(\theta_i - \alpha) \qquad (3)$$

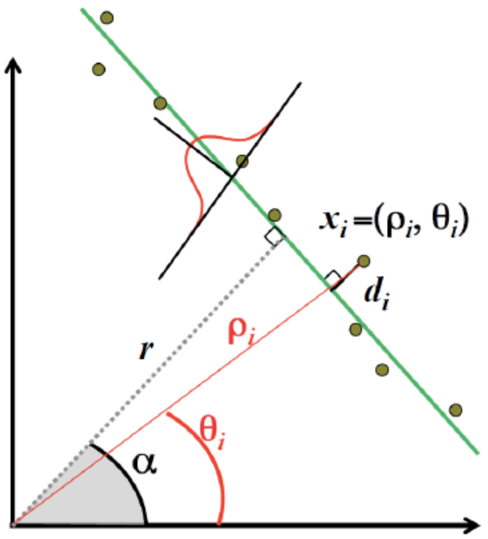

Figure 3

Line Extraction

Many algorithms have been successfully used to perform line extraction (e.g., Split-and-Merge, Line regression, RANSAC, Hough Transform etc). Here, we will implement the Split-and-Merge algorithm which is arguably the fastest but not as robust to outliers as other algorithms. See algorithm 1 below and also included in our class notes.

Algorithm 1: The Split-and-Merge algorithm for line extraction

```
function SplitLinesRecursive ($\theta, \rho, a, b$);
Input  : $\theta \in \mathbb{R}^N$, $\rho \in \mathbb{R}^N$, start index $a \in \mathbb{N}$, end index $b \in \mathbb{N}$
Output: $\alpha \in \mathbb{R}^M$, $r \in \mathbb{R}^M$, line indices $i \in \mathbb{N}^{M\times 2}$
$\alpha, r \leftarrow$ FitLine($\theta_{a:b}$, $\rho_{a:b}$);
if $b - a \leq$ MIN_POINTS_PER_SEGMENT then
    return $\alpha, r, (a, b)$;
end
$s \leftarrow$ FindSplit($\theta_{a:b}$, $\rho_{a:b}$, $\alpha$, $r$);
if $s$ is not found then
    return $\alpha, r, (a, b)$;
end
$\alpha_1, r_1, i_1 \leftarrow$ SplitLinesRecursive($\theta$, $\rho$, $a$, $a + s$);
$\alpha_2, r_2, i_2 \leftarrow$ SplitLinesRecursive($\theta$, $\rho$, $a + s$, $b$);
return $(\alpha_1, \alpha_2), (r_1, r_2), (i_1, i_2)$;
```

Three data files are provided, rangeData_<x_r>_<y_r>_<n_{pts}>.csv, each containing range data from different locations in the room and of different angular resolutions, where <x_r> is the x-position of the robot in meters, <y_r> is the y-position and <n_{pts}> is the number of measurements in the 360^o scan. Figure 4 illustrates the three data sets.

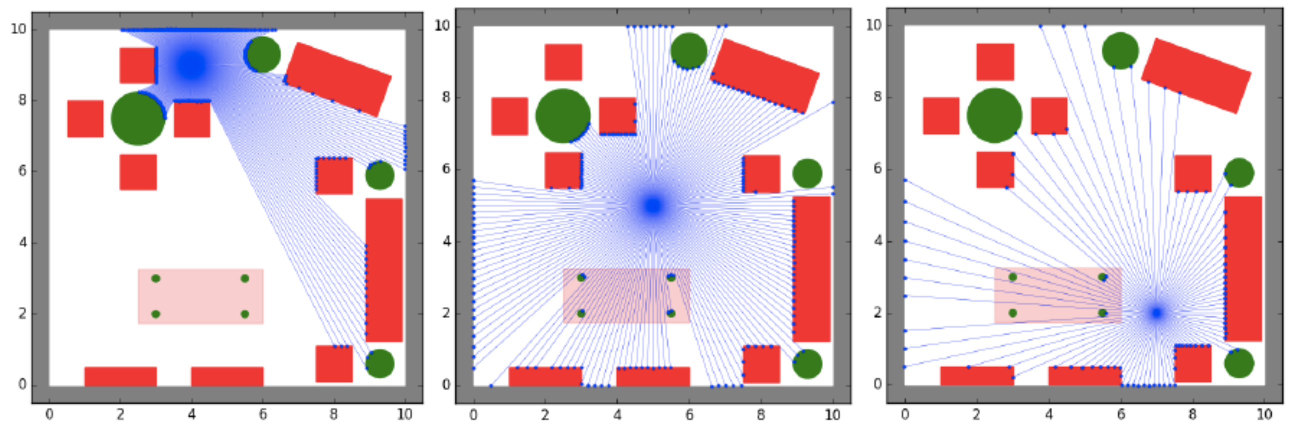

Figure 4: Lidar range data for three different locations in the room and three different resolutions, corresponding to rangeData_4_9_360.csv, rangeData_5_5_180.csv, rangeData_7_2_90.csv, respectively.

a) Implement the Split-and-Merge algorithm. Test your algorithm using each of the three data sets.

There are four suggested parameters to control segmentation:

- LINE_POINT_DIST_THRESHOLD: The maximum distance a point can be from a line before the line is split
- MIN_POINTS_PER_SEGMENT: The minimum number of points in a line segment
- MIN_SEG_LENGTH: The minimum length of a line segment
- MAX_P2P_DIST: The maximum distance between two adjacent points in a line segment

Note: There is not one correct answer to this problem. Different implementations of the algorithm may produce different lines. The best results will smoothly fit the actual contours of the objects in the room and minimize the number of false lines (e.g., lines that jump between objects in the room).

b) Submit three plots showing the extracted lines for each of the data sets. Include your segmentation parameter choices for each plot.