CS 519 F1/10 Autonomous Racing - LiDAR Lab

Pseudo Code:

Input: lidar scan msg
Output: lidar_gaps, best_gap

Parameters:

ζ = 0.1m (threshold distance b/w 2 consecutive measurements)
Δ = 0.5m (minimum width for a cluster to qualify as a gap)
λ = 1.0m (minimum depth for a cluster to qualify as a gap)
$\boldsymbol{\beta}$ = 0 (criteria for selecting best gap
  $\boldsymbol{\beta}$ = 0 → best gap based on max depth
  $\boldsymbol{\beta}$ = 1 → best gap based on max width)

Algorithm:

1) Initialize *lidar_gaps ← NULL*
2) Initialize *current_gap ← NULL*
3) Initialize *prev_measurement ← scan_msg.ranges[0]*
4) for (i = 1; i< size(scan_msg.ranges); i++)
    a) If | scan_msg.ranges[i] - prev_measurement | < ζ
        i) current_gap.ranges ← add(scan_msg.ranges[i])
        ii) current_gap.depth ← update_depth(current_gap)
    b) else
        i) current_gap.width ← calculate_width(current_gap)
        ii) If current_gap.width >= Δ && current_gap.depth >= λ
            (1) lidar_gaps ← add(current_gap)
            (2) lidar_gaps.max_depth ← update_max_depth(current_gap)
            (3) lidar_gaps.max_width ← update_max_width(current_gap)
        iii) re-initialize *current_gap ← NULL*
    c) prev_measurement ← scan_msg.ranges[i]
5) if $\boldsymbol{\beta}$ = 0
    a) best_gap_idx ← max_depth_idx(lidar_gaps)
6) else if $\boldsymbol{\beta}$ = 1
    a) best_gap_idx ← max_width_idx(lidar_gaps)
7) best_gap = lidar_gaps[best_gap_idx]
8) radius = best_gap.ranges[size(ranges)/2]
9) angle = scan_msg.angle_min + (best_gap.start_idx + size(ranges)/2)*angle_increment
10) x_pos = radius*cos(angle)
11) y_pos = radius*sin(angle)
12) z_pos = 0.0
13) gap_center = publish(x_pos, y_pos, z_pos)
14) all_gaps = publish(lidar_gaps)

Explanation

The lidar_gaps variable is a dynamic memory allocation variable used to store all the gaps found per scan, and the current_gap is the variable to handle those gaps. The algorithm begins from line 4. For every measurement present in the scan, we check if the difference between two consecutive measurement is less than a threshold (i.e., ζ = 0.1m). If so, we add the measurement to the current_gap ranges, and update the depth of the current gap (lines 4.a).

If the difference is greater than the threshold, we hypothesize the current_gap cluster to be a valid gap and calculate the width of it using the law of cosines (line 4.b.i). We then filter out erroneous gaps by passing the depth and width of the gaps to a thresholding filter (i.e., Δ = 0.5m & λ = 1.0m in line 4.b.ii). Upon passing through the filter, valid gaps are added to the lidar_gaps variable, and the gaps, which have the maximum depth, and maximum width are calculated (lines 4.b.ii.2 - 3). The current_gap is re-initialized for handling the next potential gap.

Depending on the criteria metric (i.e., $\boldsymbol{\beta}$ = 0), best gap among all the lidar_gaps is chosen and the corresponding x,y position w.r.t to the lidar sensor is calculated by converting the polar coordinates to cartesian coordinates (lines 7-12).

Parameters

There are 4 important parameters used in this algorithm:

$\zeta \rightarrow$ dictates the threshold b/w 2 consecutive measurements. This parameter controls the splitting up of clusters, which in turn produces potential gaps. We have set $\zeta \leftarrow$ 0.1m.

The potential gaps hypothesized previously are validated by 2 parameters, Δ & λ. Δ controls filters out the gaps that are too small for the car to impenetrate, and λ is responsible for filtering out the gaps that are too close to the car (e.g., distinguishing walls from actual gaps). We observed that setting $\Delta \leftarrow$ 0.5m & $\lambda \leftarrow$ 1.0m worked well for us.

$\boldsymbol{\beta} \rightarrow$ dictates the selection criteria for choosing the best gap. The best gap from the pool of lidar gaps can be selected based on maximum depth (i.e., $\boldsymbol{\beta}$ = 0), or maximum width (i.e., $\boldsymbol{\beta}$ = 1). We observed that selecting a gap based on maximum depth gave better results because, i) it gave the car more time to plan (leads to potentially faster drive), and also filters out walls (those pass the previous two filters, accidentally), as walls usually have larger width and smaller depth.

Novel extension: Another potential selection criteria is a hybrid weight-based metric, where we could calculate a new metric $\boldsymbol{\gamma} \leftarrow$ **k1*width + k2*depth**, where k1, k2 are weights, and k1 + k2 = 1. And, by tuning k1 and k2, we could gain fine control over the selection of gaps.