PROJECT REPORT

Junaid Ahmed Ansari

Robotics Research Center, KCIS

International Institute of Information Technology, Hyderabad

Contents

Chapter 1

Safe and feasible frontier detection

In this report, we explain our approach of estimating safe and feasible directions (frontiers) for the robot to explore, discuss the underlying assumptions, and present results for the same.

1.1 Introduction

From a broad perspective, our approach mainly comprises of three steps: obstacle segmentation, computing safe and feasible explore directions, and smoothing the results to avoid unnecessary changes in the robot's trajectory. We operate on point cloud data and accept point cloud and disparity images as input. In the case of disparity images, the corresponding point cloud is internally computed using the calibration parameters of the stereo camera. The choice of input can be controlled by the input_mode parameter, and can be toggled on-the-fly. For a complete reference on all the control parameters, please refer.

In the following sections, we first introduce a brief algorithm of our approach followed by a detailed explanation of the key steps involved in the process.

```
Step 1: Read point cloud:
  Based on the input_method {point cloud | disparity image} :
    point_cloud = {incoming point cloud | point cloud computed from the incoming di

Step 2: Segment all the obstacles:
  Based on obstacle_segmentation_method
  {camera height | road plane segmentation}:
    obstacles_points = all points {above road | that do not satisfy the road plane
```

```
Step 3: Extract all possible safe and feasible explore directions:
  a: Estimate gaps between the obstacles:
  b: Extract the safe gaps
  c: Compute safe and feasible explore directions

Step 4: Smooth the explore directions:
  a. compute safe angle range for all gaps in current frame
  b. smooth directions:
    For all gaps:
      If previous_explore_direction(gap) IN current safe_angle_range(gap)
           current_explore_direction(gap) = previous_explore_direction(gap)
         Else
           use the newly estimated explore direction
         End if
       End for
```

In the following section, we shall discuss and explain each step of the algorithm, and the requirement for the same.

1.2 Discussion

1.2.1 Obstacle Segmentation

Obstacle segmentation is the most important step of our algorithm as its results govern the outcomes of the subsequent steps, thereby affecting the overall robustness and performance of the algorithm. Inaccuracies in this step might lead to generation of headings pointing to unsafe areas. For this step, we have explored two different techniques: (1) camera height based and (2) road plane segmentation based. In both the techniques we assume that the camera is mounted with negligible roll and pitch (or they are known) and the camera's height from the road is a priori known. The segmentation method can be controlled by obstacle_segmentation_method parameter [7], on-the-fly.

1.2.1.1 Height Based Obstacle Segmentation

In this technique we segment the points as obstacles or navigable by looking at their height from the road. It is a simple and fast approach, as given the camera height, a point can be qualified as an obstacle point by only considering its Y coordinate (see equation (1)). However, the simplicity comes with the assumption that the road is considerably planar and parallel to the camera's optical axis i.e. the Z-axis, for at least up to the distance considered for segmentation.

1.2.1.2 Road Plane Based Obstacle Segmentation

This method is based on segmenting road plane and qualifying points as road points or obstacles. It is an elegant approach towards segmenting obstacles as, unlike height based approach, it only assumes the road plane to be approximately planar and not parallel to optical axis.Figure 1.1 shows the result of road plane based obstacle segmentation.

Segmenting road plane: To segment the road plane and compute its normal, we first extract a volume of points from the point cloud that mostly comprise of road points; we use camera's height as a prior to extract such a volume. A plane model is fit to the volume to estimate its normal (eq. 2.1). Equation 2 is essentially a least square formulation and hence can be easily solved using SVD as shown in the following algorithm. we use PCL's (Point Cloud Library) plane model segmentation support which robustly fits a plane equation to the points using RANSAC and SVD.

$$argmin \sum_{i=1}^{N} ((\mathbf{p}_i - \bar{\mathbf{p}})^T \mathbf{n})^2 \tag{1.1}$$

Algorithm: *Algorithm for fitting a plane using orthogonal distance regression and singular value decomposition*

```
1.  $\bar{p} = \frac{1}{N}(\sum_{i=1}^{N} p_i)$
2.  $A = [p_1 - \bar{p}, p_2 - \bar{p}, ..., p_N - \bar{p}]$
3.  $[USV^T] = SVD(A)$
4.  $\mathbf{n} = U(:,3)$ i.e. normal is given by $3^{rd}$ column of $U$
```

Qualifying points as road or obstacles: Points are qualified as road points or obstacles by testing if they satisfy the road plane equation or not, respectively (see eq. 2.2 and 2.3).

$$p_i \in obstacles \iff \{(\mathbf{p}_i - \bar{\mathbf{p}})^T \mathbf{n}\} > road\_plane\_tolerance \qquad i = 1,2,3...N \tag{1.2}$$

$$p_i \in road \iff \{(\mathbf{p}_i - \bar{\mathbf{p}})^T \mathbf{n}\} \leq road\_plane\_tolerance \qquad i = 1,2,3...N \tag{1.3}$$

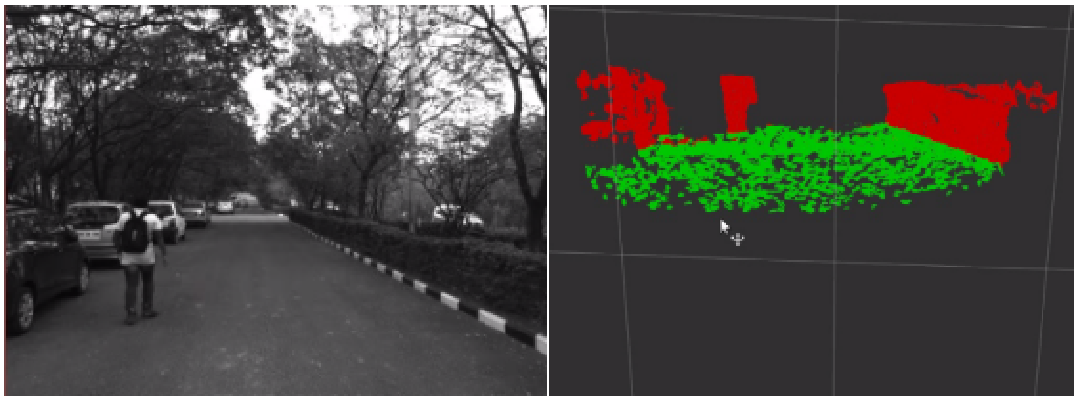


Figure 1.1: Road plane and obstacle points rendered in green and red color, respectively

Although we have both methods available for the user to choose and can also be switched in run-time, we have noticed that for real-time performance on a decent computing platform, the height based method is the adequate approach. This is because the road segmentation based method requires a substantial amount of points on the road for it to robustly fit a plane which is generally not available when working with real-time stereo matching algorithms like BM. However, if we have good computing facility on which algorithms like SGBM can work in real-time, then road segmentation based method would be the better option for reasons already stated above.

1.2.2 Safe and feasible explore direction estimation

In this step we compute a set of all possible safe and feasible heading directions which the robot can choose to perform the exploration. These directions point to different safe gaps computed between the obstacles. An explore direction's safety and feasibility is determined by its corresponding gap's width, spatial location, and orientation.

1. **Extracting gaps:**For this process we represent the obstacle points in polar coordinates. In this system, an obstacle point is at a rho distance from the origin making an angle theta with the positive X-axis, in the anti-clockwise direction. To extract the gaps, we first generate an angular occupancy strip - a one dimensional array which encodes the presence of obstacle for every angle in the range 0 to 180 degrees, and then finding all connected free cells in the strip. Figure 1.2 illustrates the complete process of extracting gaps.

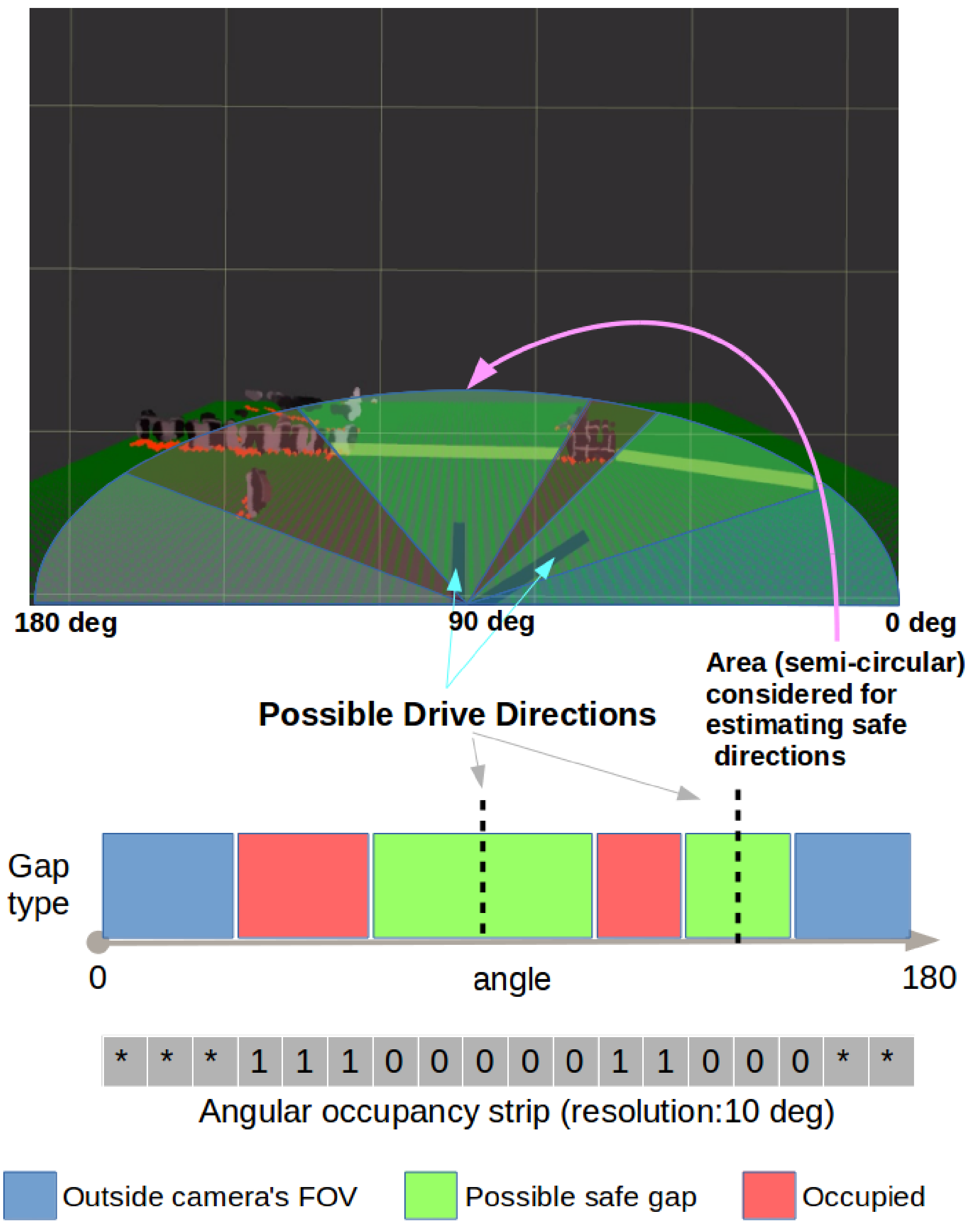

180 deg
90 deg
0 deg
Area (semi-circular) considered for estimating safe directions
Possible Drive Directions
Gap type
0
angle
180
* * * 1 1 1 0 0 0 0 0 1 1 0 0 0 * *
Angular occupancy strip (resolution:10 deg)
Outside camera's FOV
Possible safe gap
Occupied

Figure 1.2: safe gap extraction and the corresponding heading direction estimation

2. **Safety check:** A gap and hence its corresponding heading directions is qualified as safe if the gap width is greater than or equal to a threshold width, typically the robot's width including some tolerance.

3. **Feasibility check:** A gap, despite being of considerable width, can still prove to be infeasible because of its spatial location and the orientation. For e.g. a gap of near vertical orientation with its corresponding heading being approximately in the range of 70 – 110 degrees might just be a false positive resulting due to the occlusion caused by the obstacle in the near end of the gap. Another scenario where a safe gap might be infeasible is when it's orientation is steep with a negative slope in either of the two quadrants in addition to being spatially close to the robot. In this case, it could require the robot to make unnecessary and non-trivial maneuvers. Therefore after qualifying the gaps as safe perform the feasibility check and reject the infeasible ones. See Figure 1.3.

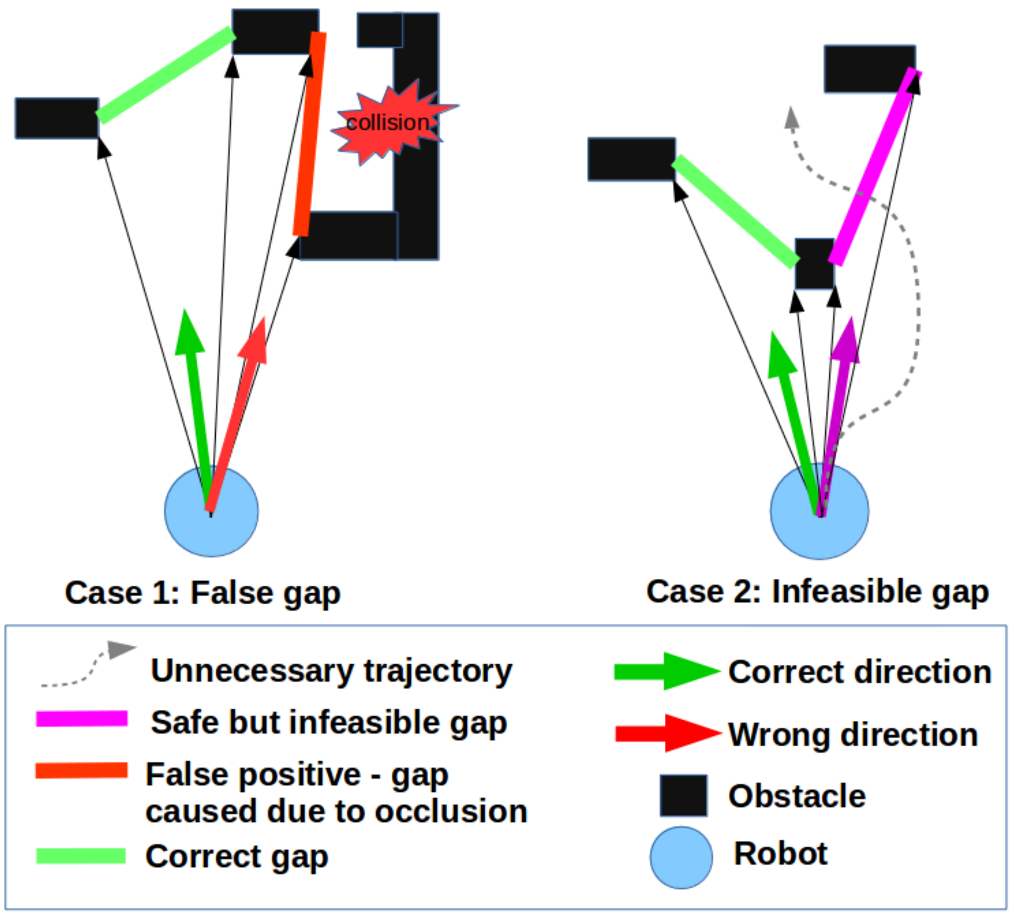


Figure 1.3: Rejecting infeasible explore directions

Therefore After the three stage validation process we generate a set of safe and feasible heading directions for the robot to perform exploration. The heading directions point to the center of of their corresponding gaps.

1.2.3 Smoothing the explore directions

Due to the motion of robot and presence of dynamic obstacles, the position and orientation of gaps keep changing. It leads to continuous changes in explore directions which can lead to instability in the robot's trajectory and to avoid it we smooth the heading directions. Smoothing is a two step process:

1. **Compute safe angle range for gaps:** To perform smoothing, we introduce the notion of safe angle range associated with every safe gap. A safe angle range is essentially a range of heading angles which the robot can choose to safely move through the corresponding gap. Safe angle range for a gap is computed by reducing the gap width from either side by an $offset$ (typically half the robot's width) and then computing all heading directions possible in the $reduced\ gap$ (see Figure 1.4).

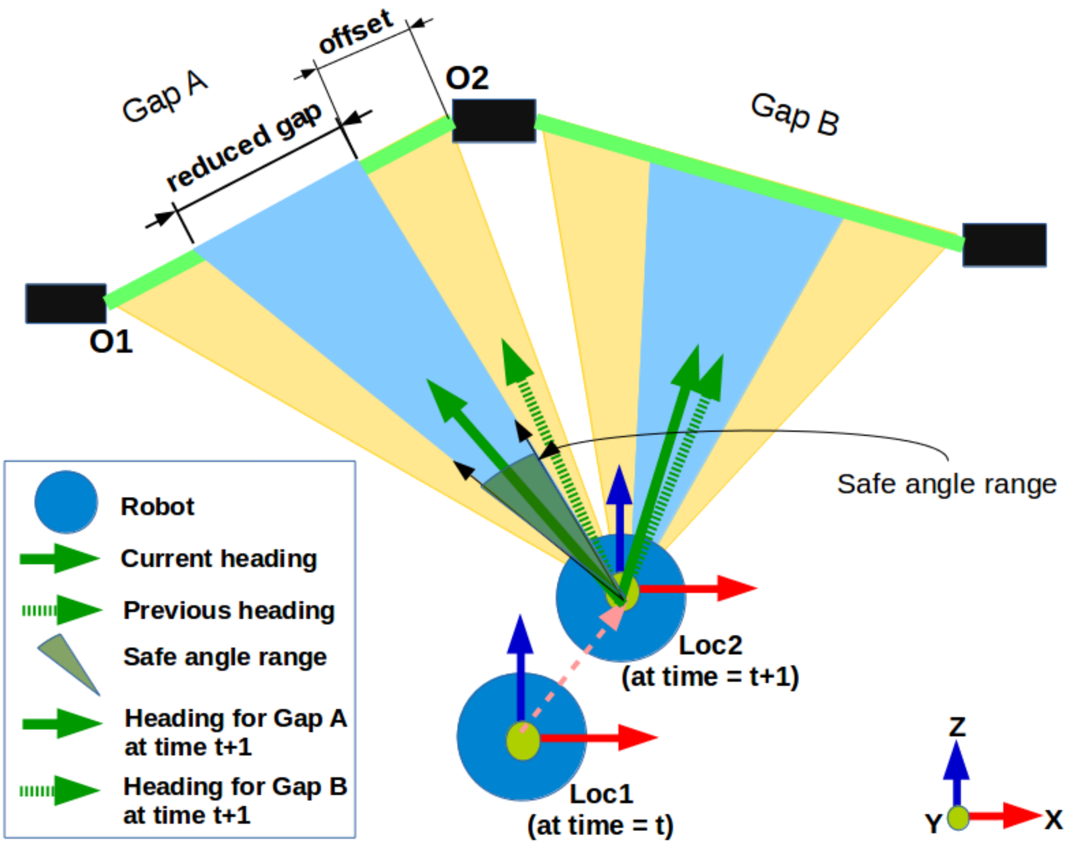


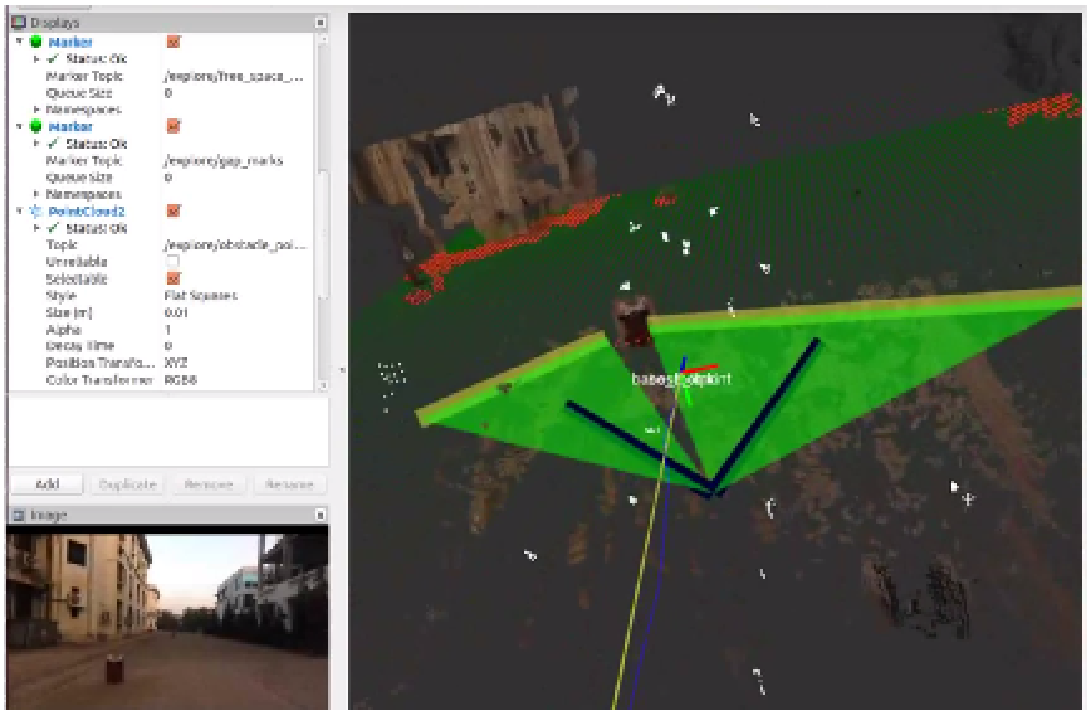
Figure 1.4: Rejecting infeasible explore directions

2. **Smoothing the explore directions:** The safe angle range helps us decide whether there is a crucial need of change in the heading direction for a safe gap. The heading directions are only changed if the previous heading directions are out of bound of the safe

angle range of the corresponding currently generated gap. For e.g. in Figure 1.4, when the robot moved from $Loc1$ to $Loc2$, in the case of $GapA$, its previous heading angle i.e. computed when the robot was at $Loc1$, lies outside the safe angle range of the gap. Therefore, $GapA$ will have an update in heading angle. On the other hand, at time $t+1$, the previous heading angle of $GapB$ is still in its safe angle range which means there is no need to unnecessarily update heading for $GapB$ at time $t+1$. This way we impart stability in the robot's motion by minimally changing the directions. 1.3.

1.3 Experiments and Results

We have attached this module on RTABMAP. Snapshot of the result id shown in figure 1.5.



```
Marker                      ✓
  ✓ Status: Ok
  Marker Topic              /explore/free_space_...
  Queue Size                0
  Namespaces
Marker                      ✓
  ✓ Status: Ok
  Marker Topic              /explore/gap_marks
  Queue Size                0
  Namespaces
PointCloud2                 ✓
  ✓ Status: Ok
  Topic                     /explore/obstacle_poi...
  Unreliable                ☐
  Selectable                ✓
  Style                     Flat Squares
  Size (m)                  0.01
  Alpha                     1
  Decay Time                0
  Position Transfo...       XYZ
  Color Transformer         RGB8
```

Figure 1.5: Blue Lines are navigable Directions for Robot and green area shows the coverage