Group25_A1

David Petrovic, Federico Luisetto, Umberto Salviati

December 2023

Abstract

The aim of this project is to move TIAGo robot in a predefined environment to a certain position of the map. When the robot is in the correct position and orientation the detection of the objects starts, giving as output the position of the objects found.

1 Group information

GROUP 25
David Petrovic, david.petrovic@studenti.unipd.it
Federico Luisetto, federico.luisetto@studenti.unipd.it
Umberto Salviati, umberto.salviati@studenti.unipd.it
`https://bitbucket.org/fede-luis-ir/ir2324_group_25/src/master/`
`https://drive.google.com/drive/folders/1kel8kBC7010t5D5SY_CqPlxilI8Glt-I?usp=sharing`

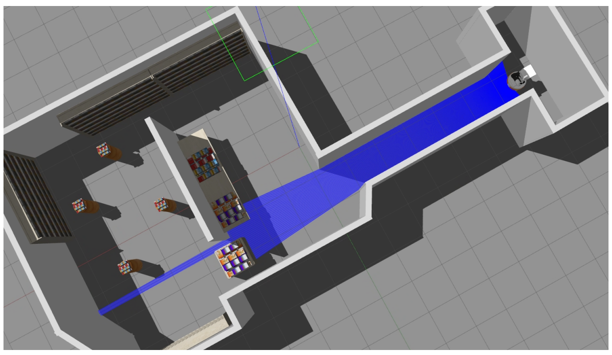
2 Introduction

In this project we have to implement a routine that permits TIAGo to navigate inside the environment from the starting pose (Fig. 1a) to the destination pose (Fig. 1b). The destination position and orientation have to be inserted by the user from the command line. Once the destination pose is reached the robot has to detect the movable objects (the cylindrical tables visible from that pose) and return the positions of the latter, excluding the static obstacles of the map (e.g. walls, shelves, ...).

3 Movement

The project is implemented with the following structure:

- the client: action client that receive as input the final pose that the robot has to reach `(X,Y,angle in degrees)` and sends it to the server through the goal message (of custom type `pose.msg`).
- the server: action server that receives the goal from the client and by using `move_base` the goal pose is passed to the robot. After the robot has reached the goal pose, by subscribing to the `scan` topic, the server calculates the positions of the movable obstacles and sends a vector of positions (of custom type `positions.msg`) to the client as result of the action.


(a) Starting pose

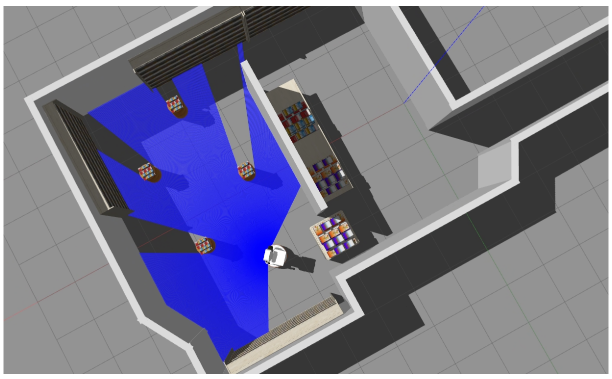

(b) Final position and orientation

Figure 1: Environment

4 Object detection

As stated before, the data used to execute the object detection derives from the `scan` topic.
To detect objects using LaserScan data, we initially perform data segmentation. To this purpose, we introduce the finite difference (the approximation of the derivative):

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

as the limit definition of the derivative:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{\Delta x}$$

in our discrete set. In the Δx of the derivative formula, we used the `angle_increment` provided by the laser scanner. By employing an empirical threshold, determined by examining the derivative of the data [2], we found that a threshold of 50 was appropriate. This threshold enables us to identify the discontinuities in our vector.

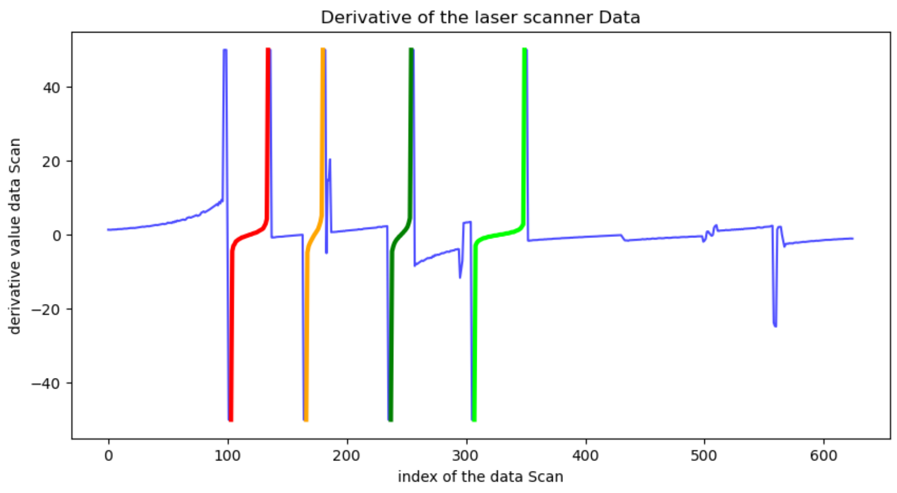


Figure 2: Finite difference of a scan(similiar to Fig1b) where each obascle is colored differently

The following step was to detect circles. In this phase, we assume that the wall is composed of straight lines, while the circular lines represent our movable obstacles. After converting the segmented data into cartesian coordinates (Fig.3a), we distinguish between straight lines and circles. During this transformation, each segment is treated individually, ensuring that each semicircle always has concavity upwards.

To differentiate actual data from the straight line, we apply a function study to determine the concavity of a function. Since a semicircle resembles a parabola, its derivative will be zero at the center, before that it will be negative and after the center there will be an increasing slope that leads to an inflection point. This time, the finite difference was calculated using the actual Δx.

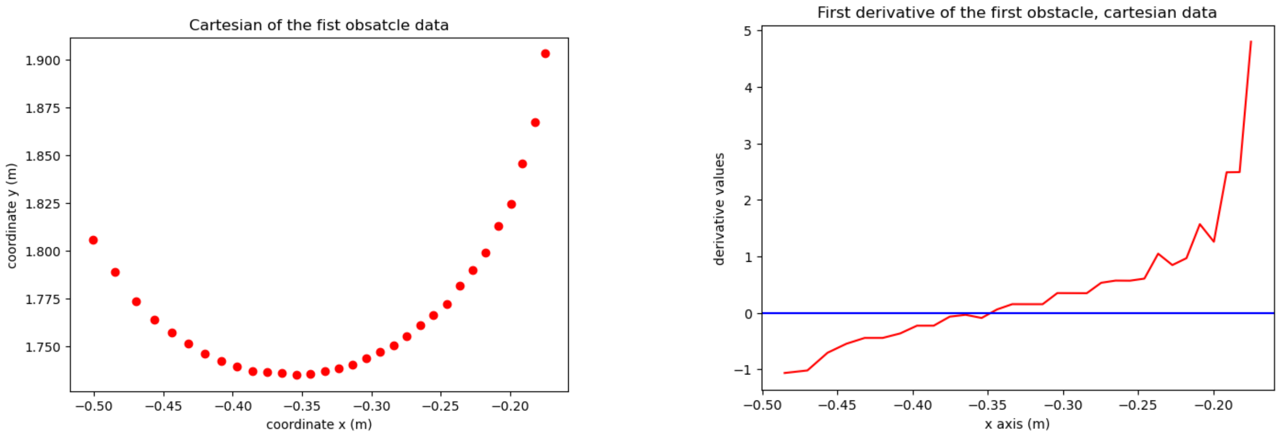
The best approach now should be to calculate the second derivative, but in this case is challenging due to noise in the data (even though we smoothed the data using a window approach). Thus, we simplify the approach by examining the graph of the first derivative. Before the zero point, the derivative should be negative, and after the zero point, it should be positive (Fig 3b). It is evident that occluded obstacles will not be detected in our approach, as we require more than half of the obstacle to be visible for detection. Finally, to clean up the data, segments with insufficient data are discarded.

After the positions of the lasers that touch the moving obstacles surface are found, in each cluster of points the server find the center of the obstacle using four of these points to create two segments. In the middel of these segments pass one perpendicular (to the segment) line. The intersection of those two lines is the center of the circle (Fig. 4).

4.1 Output

As output the server sends to the client one of the following:

- a list of the positions of the moving objects with respect to the map reference frame, if it is possible.
- a list of the positions of the moving objects with respect to the robot reference frame otherwise.



(a) Cartesian coordinate of an obstacle

(b) First derivative of the obstacle graph

Figure 3: Cartesian coordinate and derivative of an obstacle (color matched with Fig. 2)

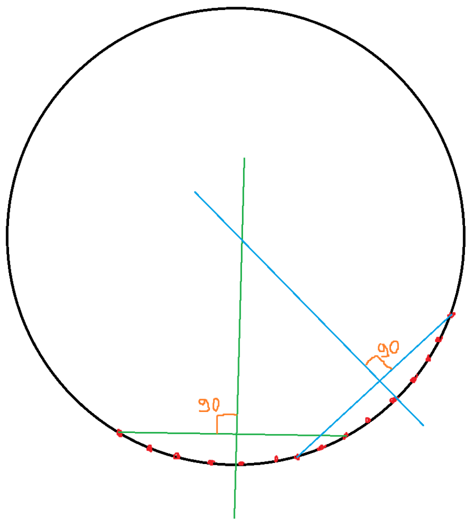


Figure 4: Sketch of center-finder algorithm

5 Conclusions

To conclude, we can say that we have worked almost together during all the assignment on videocall so everybody contributes more or less in the same measure. To be more precisely, Umberto was more focus in the detection part, Federico in implementing a way to compute the centers of the circles and David in assembling all the codes.