UNIVERSITY OF PADUA

INFORMATION ENGINEERING DEPARTMENT (DEI)

MASTER'S DEGREE IN COMPUTER ENGINEERING

Report Assignment 1

Prof.
Emanuele Menegatti

Students of Group 19
Francesco Agostini : francesco.agostini.5@studenti.unipd.it
Andrea Felline : andrea.felline@studenti.unipd.it
Andrea Pietrobon : andrea.pietrobon.4@studenti.unipd.it

Bitbucket Repository:
bitbucket.org/unipd-projects/ir2324_group_19/

Video example:
drive.google.com/file/d/1OVGCOG_ivf0-LZL4KkG6Co00eFIXfU/view?usps̄haring

Contents

1 Introduction

This report aims to provide a comprehensive overview of the functionality and methodology employed in our solution for the given assignment. Specifically, we will delve into the following key aspects:

- The structural framework of our solution, detailing the client-server.
- The strategy adopted to navigate through narrow corridors utilizing laser data for effective pathfinding.
- Our approach to detecting obstacles, encompassing the identification of walls and circular movable obstacles within the environment.

2 Client-Server Comunication

The communication between client and server is managed using an action.
The client asks the user to insert a pair of coordinates (the point B), the orientation that the robot should have at the end and asks also if he wants to read the feedbacks of the server or not.
The final orientation of the robot should be inserted as an angle in radians with respect to the initial orientation that the robot has (when the simulation starts the robot is facing straight to the positive section of the X axis).
The client will then send a personalized message as a goal to the server, which will read it, reformat it and send it to the robot on the `/move_base/goal` topic.
While the robot is moving the server will read the feedbacks and the laser scans. It will try to detect the cylinders from the laser scan values and will send some partial results as feedback to the client. This feedback will contain the current position of the robot (taken from the feedback of the robot itself), a list of cylinders that are currently in the field of view of the robot and the largest list of cylinders that the robot has ever seen at the same time along the path.
When the robot reaches the destination (point B sent by the client) it sends a result message to the server, which will, in turn, send a result to the client. The server-client result will contain the list of cylinders that the robot is seeing in his final position and the largest list ever seen along the path.
Depending on the position given, it may happen that the robot never gets to see all the cylinders at once, so it may not return a complete list.

3 Movement

The movement of the robot is managed through the `move_base` action system of the robot. Each single topic (goal, feedback, and result) is read separately for convenience.
The server receives the orientation of the robot as an angle in radians with respect to the original orientation of the robot, but it should send it as a Quaternion. To achieve this, it uses an object of type `tf2::Quaternion` and the function `Quaternion::setRPY()`.

The position and the quaternion are sent to the robot on the `/move_base/goal` topic. The robot then automatically starts moving towards the given position and orientates itself with the provided angle once it reaches it.

While the robot moves, it sends its current position on the topic `/move_base/feedback` (along with other information), and when it reaches the goal, it sends a message on the topic `/move_base/result`.

4 Detection

The server, upon sending the goal to the robot, also initiates the process of reading the messages that the robot sends on the topic `/scan_raw`. On this topic, the robot transmits a list of distances obtained from a scan of the half space in front of it: the scan extends from an angle of **-1.9199 radians** to **1.9199 radians**, where **0 radians** represents the direction directly in front of the robot, resulting in a field of view slightly larger than **180 degrees**.

The server analyzes these distances and groups them: a group signifies a continuous surface, thus forming a new group whenever a distance significantly differs from the previous one. A continuous set of points within a group is defined as a profile, with each point in a profile not differing from its neighbors by more than **0.5 meters**.

Each profile undergoes scrutiny to ascertain whether it resembles a semi-circle, akin to what would be seen if a cylinder were present. The assessment of a profile's semi-circular nature involves several criteria:

- The distance between the nearest point to the first and last points of the profile should be similar (difference $<$ **0.1**).
- The midpoint between the first and last points, presumed to be the circle's center, should share a similar distance with the nearest point.
- The radius (i.e., the distance from the center to one of the profile points) should approximate the expected radius of the cylinders being detected (approximately **0.17 meters**).
- If these conditions are met, a stronger but slower check is performed to further validate the profile. This additional check examines every point's distance from the center, ensuring similarity to the circle's radius.

Upon satisfying all conditions, the profile is labeled as a circle; otherwise, it is marked as unknown. The list of profiles identified as cylinders is saved and transmitted with the feedback. If the current list's number of elements is greater than or equal to the largest previously seen list, it replaces the previous record. This approach aids in updating the list, accommodating any changes in cylinder positions that may occur while the robot continues detecting.

Updating the larger list, even if the current one holds the same number of elements, ensures adaptation in case any of the cylinders are repositioned during the robot's detection process. If the robot re-observes all the cylinders simultaneously, it updates the list with the revised positions.

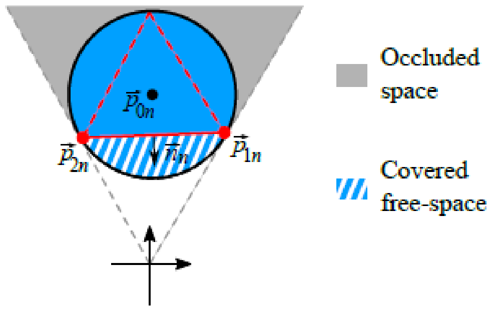
5 Final Notes

1. It was required to reflect the status of the robot in the feedback. In our program, the server sends feedback only when the robot is moving and detecting, so there is only one type of status and one type of feedback message. The feedback content still reflects the status of the robot since it contains the current position of the robot (indicating movement and position updates) and the lists of cylinders (indicating detection).

2. In the video, we can observe occasional false positives in the feedback of the first goal sent, but memorizing the largest group of cylinders helps the server to rectify these errors. It would be highly improbable for the algorithm to produce multiple errors simultaneously. Additionally, different values for the position, orientation, and feedback flag yield various outputs, demonstrating the robot's correct behavior in response to these parameters.

3. In the folder `./Other documentations/assignment_1`, you will find the file `refman.pdf`, which provides further explanation of the steps to execute in order to start the program and a more comprehensive documentation of the code.
Additionally, in the folder `./Other documentations/assignment_1/html`, you will find the file `index.html`, which contains an interactive representation of the documentation. It is recommended to review both sources for a complete understanding of the work done.

4. As highlighted in the image (see Figure 1), we are aware that our method, although functional, might have limitations if the robot approaches objects too closely. Currently, the navigation system prevents the robot from getting too close to objects. However, if it were to get too close, the identified centers of the objects could be misaligned or even, in extreme cases, not be correctly identified if below a certain threshold.



Example Image of the space around the objects.

5. We mainly used github to manage our versions, you can see all the commits and the steps of our work on https://github.com/AndreaFel/IRProject