



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Intelligent Robotics A.Y: 2023/24 Group 12

Assignment 1

Prof.
Emanuele Menegatti

Students:
Matteo Villani : matteo.villani@studenti.unipd.it
Stefano Trenti : stefano.trenti@studenti.unipd.it
Gabriel Taormina : gabriel.taormina@studenti.unipd.it

Link

1. Repository Bitbucket
2. Google Drive folder

1 Description

The goal of the project is to get the robot Tiago from the initial point Starting Pose to point Pose_B by navigating through the environment that includes two rooms, corridor and obstacles. Once Tiago reaches the Pose_B, it must recognize the obstacles, cylindrical in shape, from his surroundings and indicate how many there are and their location in the room.

Tiago can reach Pose_B in three different ways:

- The first involves using the motion control law that we developed to reach Pose_B directly, thus never using the Navigation stack (1). [EXTRA-POINT]
- The second involves using a motion control law that we developed to cross the corridor and then using the Navigation stack to reach Pose_B (2). [EXTRA-POINT]
- The third involves using the control law that is already there by taking advantage of the Navigation stack (3)

2 Structure

We implement an Action Client/Server structure:

- Client: the action client receives the input from the user by command line, by providing:
 - The x, y coordinates regarding location and yaw regarding orientation. This information will go to create Pose_B that corresponds to our goal.
 - Which of the three possible motion control laws the user want to use choosing between 1, 2, 3.

Once the goal is obtained, the client sends the information to the server. The action client also implements callbacks to the feedback of the action server, and prints the task's current status in the terminal.

- Server: the action server executes all the tasks.
 - First it acts as a client for the move action, used to reach our goal, sending the information to the dedicated server that will process the robot's movement, in accordance with the mode chosen as the motion control law to achieve the goal (1, 2, 3).
 - Once the goal is reached, it subscribes to the topic /scan to get the information regarding the surroundings area. This information will then be used for locating objects and counting them.
 - When all the information is obtained, the server returns it to the client indicating the location (in the "base laser link" reference frame) and quantity of objects in the room

3 Objects detection

Our approach takes into account the following assumption: the objects to be detected should be detached from the wall and cylindrical in shape (previously agreed with the tutors). Information obtained from the laser scan present on Tiago was used to detect obstacles.

The approach is as follows:

- Using the distance data associated with a certain angle obtained from the laser scan, we calculate the derivative (understood as the difference of two consecutive values: $[i] - [i-1]$) to detect discontinuities. This works like an edge detector in image processing, detecting the right edge discontinuities in the laser scan since the scan is counter clockwise. Such discontinuities, in fact, highlight the presence of a different object, since the difference of two consecutive points will be a very small value if they belong to the same element (e.g., the wall). Saving the obtained values in a vector having the length equal to the laser data vector.

- Once we have obtained the vector of discontinuities, we check vector values and remove ones that are too small, choosing a threshold value of 0.5 m proved to be a good value to distinguish objects and walls. In this way, we get a vector where most of the values are zero, except for the indexes where we detect the right-edge of an object.
- Then this vector was used as a guideline to create another vector of only the data points that correspond to objects, this was accomplished by taking the original distance value (corresponding to the right-edge index of object) and keeping the following distance values until they are within a threshold in a counter-clockwise sweeping motion. We then obtain a vector with all zero values but the distance values corresponding to the various objects.
- To find the center of the objects, knowing it is of a cylindrical shape, we choose to take the initial, halfway, final points of the singular objects and interpolate a circle. From the interpolated circle we then obtain the center point coordinates with respect to the `base_laser_link` frame.

This approach also ignores all points too close to the sensor, such as parts of the robot itself. The detection was tested in various position on the map and was found to be successful in detecting the visible object and obtaining the correct center-point values. Some of these test result were uploaded as screenshots in the Google Drive folder.

4 Extra Points - Motion Control Law

For the extra points part of the assignment the ask was to implement a motion control law to reach a goal point that directly process the sensor data and generate velocity command for the robot in the `mobile_base_controller/cmd_vel` without using the `move_base` stack. To achieve this, the idea was similar to the potential fields algorithm but implemented in a reactive and radial form. We computed two vectors, an attractive force vector and a repulsive force vector which are then summed to create the total force vector. This vectors have a size equal to a radius $2\pi \times \text{angle_increment}$ to obtain a vector size of 1088. In this way the robot X-axis direction is the halfway point of the array, the points after this are the positive angles $+\theta$ and the ones before are the negative angle $-\theta$.

- The attractive force vector was computed by first obtaining the index of the bin corresponding to the Goal heading angle, then the vector was compiled by assigning a value of 1 to this index and a linearly decreasing value for all other more distant indexes. This means the closer you are pointing to the goal heading angle the higher attractive force value.
- The repulsive force vector was instead computed using the LaserScan message from the `scan` and by setting the repulsive force to 0 if nothing was detected or was out of the laser sight and setting it to $1 - g \times \text{range}[i]$ if anything was detected, where g is the potential repulsive gain that can be tuned.
- The total force vector was then computed summing the two previous force vectors.

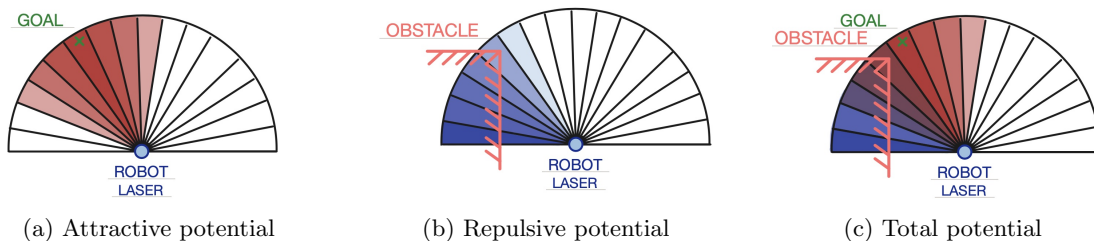


Figure 1: Visual representation of radial potential field computation

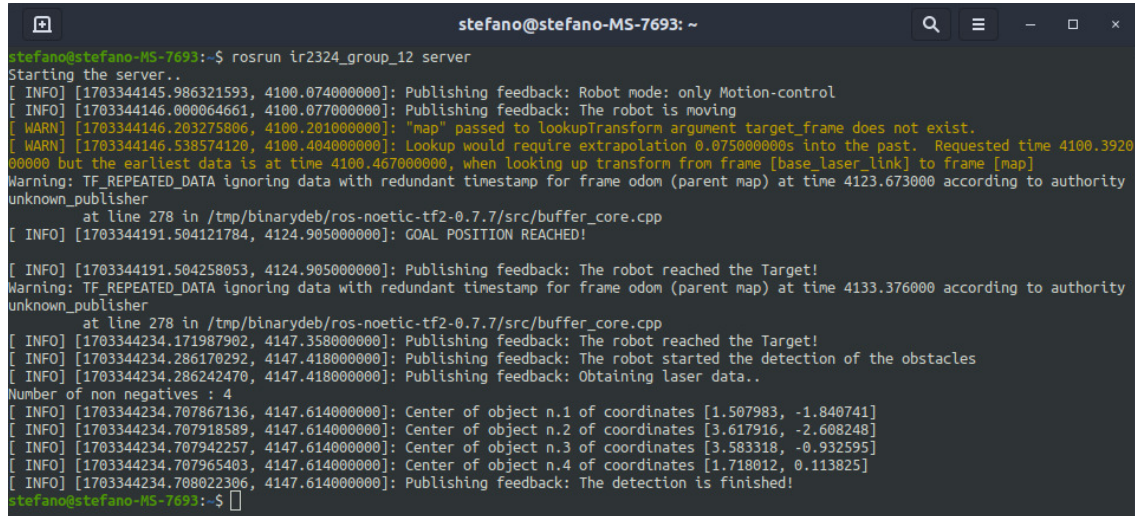
From the total force vector we can select the index corresponding to highest potential which is then used as the angular direction for the robot to follow to reach the goal and avoid collisions. To improve this rudimental approach many steps of pre-processing were implemented. The LaserScan data was smoothed within the 8 nearest neighbours so that the fall-off from object corners was

gradual, this is needed because the robot thinks itself as a point and may believe to be able to pass close to a corner without hitting it. Then an obstacle detection function was implemented which takes a cone in front of the robot as detection area where, if anything is detected inside this area, the repulsive force for the corresponding bins are incremented depending on the free space left. Also a fail-safe function was implemented so that if anything was detected in a "danger zone" (which is a semicircle of radius 0.5 from the robot and $\pm 70^\circ$ from the X-axis) the robot will rotate until it clears this obstacle. After obtaining the safe best heading angle direction the velocity commands are published in the appropriate topic as a `geometry_msgs::Twist` message which contains the linear velocity in the X-axis and the angular velocity in the Z-axis.

The linear velocity was computed as: $v * (\pi - \text{maxPotAngle}) / (\pi * \text{frontRange} / 2)$ where v is the linear velocity gain, maxPotAngle is the chosen angle with max potential and frontRange is the laser detection in front of the robot. This results in a dynamic linear velocity that let the robot travel fast when nothing is in front of it and slow down when it has to do a turn. The angular speed was instead computed as: $(\text{maxPotAngle} * \text{angleVelGain}) / \pi$ where angleVelGain is a tunable gain value for the angle velocity.

5 Results

In all three modes 1, 2, 3 once the goal is reached, the detection of the number of obstacles is correct and equal to four. Even when testing the detection where an obstacle was completely occluded three objects were correctly detected. The position of the centers of the cylindrical objects calculated by us corresponds to the correct one displayed through Rviz. This was tested on different goals and was successful in all cases.



```

stefano@stefano-MS-7693: ~
stefano@stefano-MS-7693:~$ roslaunch ir2324_group_12 server
Starting the server..
[ INFO] [1703344145.986321593, 4100.074000000]: Publishing feedback: Robot mode: only Motion-control
[ INFO] [1703344146.000064661, 4100.077000000]: Publishing feedback: The robot is moving
[ WARN] [1703344146.203275806, 4100.201000000]: "map" passed to lookupTransform argument target_frame does not exist.
[ WARN] [1703344146.538574120, 4100.404000000]: Lookup would require extrapolation 0.075000000s into the past. Requested time 4100.3920
00000 but the earliest data is at time 4100.467000000, when looking up transform from frame [base_laser_link] to frame [map]
Warning: TF_REPEATED_DATA ignoring data with redundant timestamp for frame odom (parent map) at time 4123.673000 according to authority
unknown_publisher
at line 278 in /tmp/binarydeb/ros-noetic-tf2-0.7.7/src/buffer_core.cpp
[ INFO] [1703344191.50421784, 4124.905000000]: GOAL POSITION REACHED!
[ INFO] [1703344191.504258053, 4124.905000000]: Publishing feedback: The robot reached the Target!
Warning: TF_REPEATED_DATA ignoring data with redundant timestamp for frame odom (parent map) at time 4133.376000 according to authority
unknown_publisher
at line 278 in /tmp/binarydeb/ros-noetic-tf2-0.7.7/src/buffer_core.cpp
[ INFO] [1703344234.171987902, 4147.358000000]: Publishing feedback: The robot reached the Target!
[ INFO] [1703344234.286170292, 4147.418000000]: Publishing feedback: The robot started the detection of the obstacles
[ INFO] [1703344234.286242470, 4147.418000000]: Publishing feedback: Obtaining laser data..
Number of non negatives : 4
[ INFO] [1703344234.707867136, 4147.614000000]: Center of object n.1 of coordinates [1.507983, -1.840741]
[ INFO] [1703344234.707918589, 4147.614000000]: Center of object n.2 of coordinates [3.617916, -2.608248]
[ INFO] [1703344234.707942257, 4147.614000000]: Center of object n.3 of coordinates [3.583318, -0.932595]
[ INFO] [1703344234.707965403, 4147.614000000]: Center of object n.4 of coordinates [1.718012, 0.113825]
[ INFO] [1703344234.708022306, 4147.614000000]: Publishing feedback: The detection is finished!
stefano@stefano-MS-7693:~$

```

Figure 2: Server

```

stefano@stefano-MS-7693: ~
stefano@stefano-MS-7693:~$ roslaunch ir2324_group_12 client 11.1 1.3 -1
Starting the client..
[ INFO] [1703344142.497891202, 4098.051000000]: Waiting the 'task Server' to start..
[ INFO] [1703344142.687137965, 4098.189000000]: 'task Server' started, sending goal.
Choose the robot's behaviour mode:
'1': only Motion-control
'2': Half Motion-Control, half Navigation
'3': only Navigation
1
[ INFO] [1703344146.148091179, 4100.177000000]: [Feedback]: Robot mode: only Motion-control
[ INFO] [1703344146.148149135, 4100.177000000]: [Feedback]: The robot is moving
[ INFO] [1703344191.605271893, 4124.966000000]: [Feedback]: The robot reached the Target!
[ INFO] [1703344234.362340704, 4147.454000000]: [Feedback]: The robot reached the Target!
[ INFO] [1703344234.362420001, 4147.454000000]: [Feedback]: The robot started the detection of the obstacles
[ INFO] [1703344234.362466547, 4147.454000000]: [Feedback]: Obtaining laser data..
[ INFO] [1703344234.808231657, 4147.655000000]: [Feedback]: The detection is finished!
[ INFO] [1703344236.182370984, 4148.418000000]: Assignment 1 task finished: [SUCCEEDED]
Printing results..
[ INFO] [1703344236.182432301, 4148.418000000]: [Result] Obstacle [1]: (1.507983, -1.840741)
[ INFO] [1703344236.182449740, 4148.418000000]: [Result] Obstacle [2]: (3.617916, -2.608248)
[ INFO] [1703344236.182467049, 4148.418000000]: [Result] Obstacle [3]: (3.583318, -0.932595)
[ INFO] [1703344236.182483606, 4148.418000000]: [Result] Obstacle [4]: (1.718012, 0.113825)
stefano@stefano-MS-7693:~$

```

Figure 3: Client

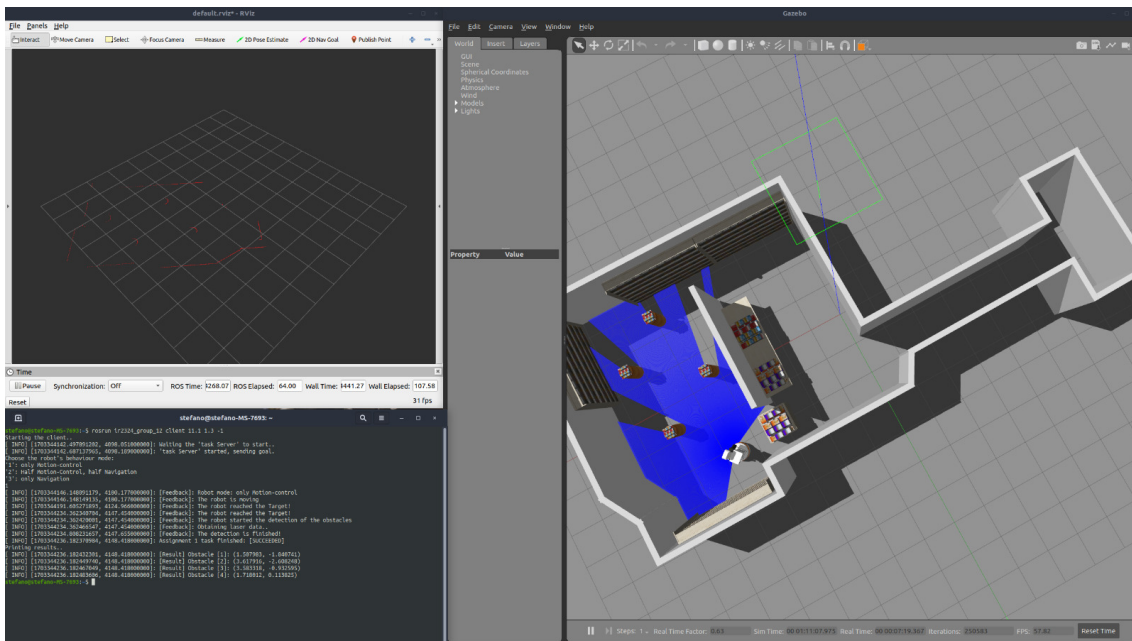


Figure 4: Rviz + Client + Gazebo