IR Project Report - Assignment 1

(EXTRA POINT DEVELOPED)

Group 09		
Boscolo Simone	Casarin Marco	Polato Anna
2026826	2044727	2007061

1. Introduction to the proposed solution

For the Assignment 1 of the IR Project, the routine was developed according to what fig. 1 illustrates. This framework allows to best exploit the ROS features, splitting the tasks between different nodes and keeping them connected one with each other, sharing relevant information. As one can observe in figure 1 four nodes were defined, one client and three servers.

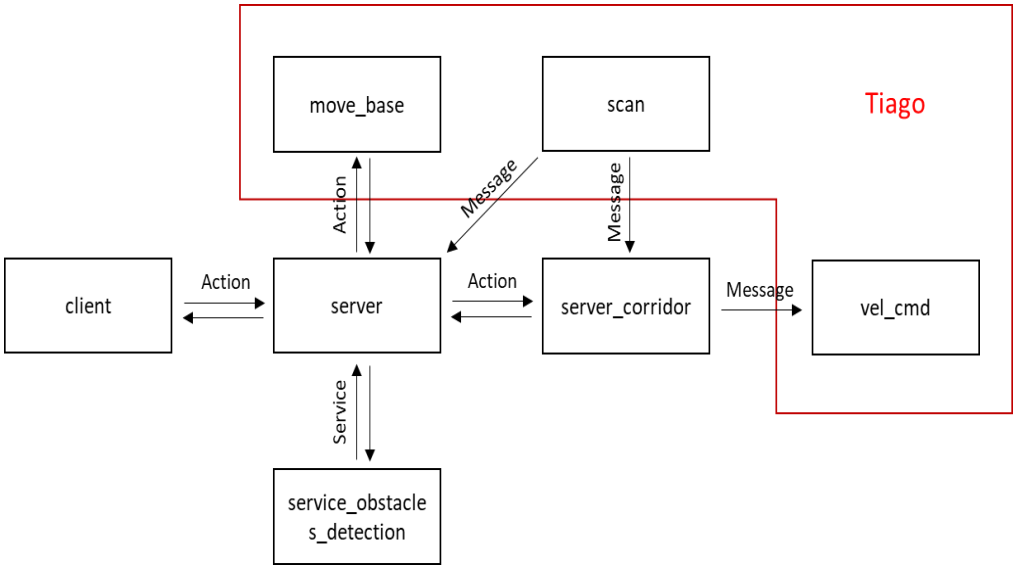

Figure 1 General structure of the project

The client's request can be seen as composed by three parts: while moving Tiago to a desired position, it must be able to get through narrow passages without collisions and then return the positions of the dynamic obstacles once the goal pose is reached. The `server` is the only one communicating with the client, receiving instructions from it and sending back feedbacks and results. It can be seen as a "mediator" because it does not directly solve the above-mentioned tasks but distributes them between other three nodes, whenever their contribution is required. The latter are not linked directly to the client and their intervention fully depends on the mediator server. The links `server-server_corridor` and `server-move_base` were chosen to be based on actions which can give feedback during the navigation. In fact, the feedback from the `server` to the `client` consists in a string that describes the state of the robot. The obstacle detection instead, being more immediate, does not require to be monitored therefore a service is sufficient for the `server-obstacle_det` link. The chosen structure makes the code clear and in case of malfunction of the routine, the problem is isolated to the single node and easier to solve.

2. The Corridor Issue – extra point implementation

To improve the navigation through narrow passages, e.g., the corridor, it was decided to implement a routine in a node that communicates with the main `server` which informs Tiago when to use the corridor routine. The `server` checks every second the content of the laser scanner on the left and on the right side of the robot and, in case of closeness to an object, it sends the proper feedback to the `server_corridor`. The latter is an action server node that predicts the right moving direction based on the orientation of Tiago with respect to the obstacles.
It can be observed from figure 2, from the laser scan two values are extracted, one from left and one from the right of Tiago. The angular rotation around z axis to pass to Tiago is the difference between left and right ranges; in this way, the next orientation of the robot is mathematically always towards to the center of the corridor.

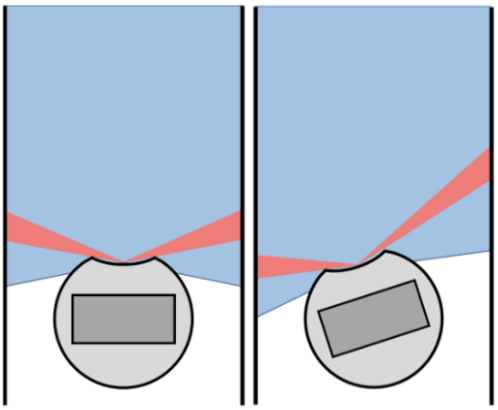

Figure 2 Example of section of the scan ranges adopted to check if the robot is in a narrow space.

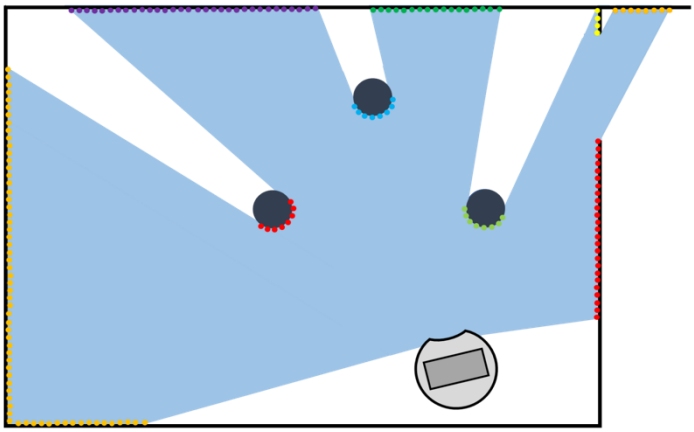

Figure 3 Example of a laser scan detection of objects' points and relative clustering: each color represents a cluster.

3. Obstacle Detection and Tracking

The idea for the detection of the dynamic obstacles (i.e., the cylinders) consists in clustering laser scan points based on distances and then process each cluster to check whether the shape of its points fits a circumference or not. The routine keeps adding detected points to the same cluster if they are "close enough" to the last added one; A new cluster is defined whenever there is a gap between subsequent points, that is, they differ by a length greater than a threshold defined as the double of the average distance of all consecutive points in the scan. Figure 3 shows how the clustering works.
Once the points are clustered into groups, each one is tested to fit a circumference and a line:

- **Circumference fit test**: considering the angle range of the cluster, the equation of the circumference passing through the first, the last and the point in the middle is found. Then it is computed the following normalized score s_c:

$$s_c = \frac{1}{z_c} \qquad \text{where} \qquad z_c = \frac{\sqrt{(\sum_{i=1}^{n} (d_{cp_i} - r)^2)}}{n}$$

  where *r* is the radius of the circle, d_{cp_i} the distance from the point p_i to the center of the circle and n the number of points in the cluster.
  If the points fit well a circle, z_c will be a low value, hence s_c high value; Higher the score, higher the probability that the cluster fits a circumference (Figure 4).

- **Line fit test**: similarly, a score s_l is used to quantify the probability that each cluster fits the line passing through the first and the last points in the range angles (i.e., it is a wall). Then the score is no more computed using the distance from point p_i to the circumference, but the distance from point p_i to the line.

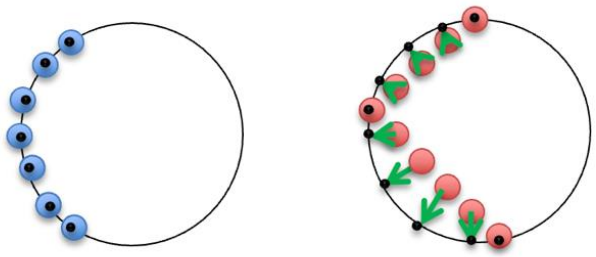

Figure 4 Example of computation of the score Sc given two clusters of points (blue and red). The green arrows represent the distance of points from the circumference. If the cluster is circle shaped (blue) the distances are close to zero, then the score is high.

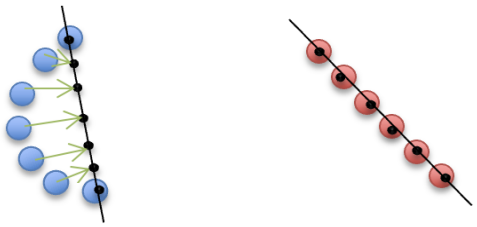

Figure 5 Example of computation of score Sl given two clusters (blue and red). The green arrows represent the distance of points from the line. If the cluster is circle shaped (blue) the distance are greater than zero, then the score is low.

Lastly, cylindrical objects are found as following: all clusters with high line score s_l are discarded and only the remaining ones with high circular score s_c are selected as cylinders. Then, as result of the action, we return the length of the radius and the coordinates of the center of the cylinder with respect to the base link reference frame.