Group25_A2

David Petrovic, Federico Luisetto, Umberto Salviati

January 2024

Abstract

The aim of this project is to implement in an assistance robot (namely a TIAGo by PAL Robotics) to fetch and delivery behavior for everyday objects.

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/1oHUEsZ28TrrXRZDUDC0CUJtkjIuJyRpl?usp=sharing`

2 Introduction

In this assignment, we are required to implement a fetch and delivery behavior for everyday objects in an assistance robot, specifically a Tiago by PAL Robotics. We have developed a routine that scans objects, understands and stores their positions, and then performs a pick-and-place operation. The robot identifies the objects and their final placement locations, subsequently picking up the objects and correctly placing them on designated colored tables. In particular, we implemented our solution using laser data to select the pose for the docking routine in front of the cylindrical tables during the object placement phase.
Our implementation is organised as a "central" node named *manager* from which we control all the execution. Then, there is an apposite service managed by a node called *nodeB* for the detection of the objects on the table. Another node that manages two services for the pick and for the place tasks (*arm_controller*). One node for detecting the order (from right to left) in which the colored tables appear in front of the robot by using colors (*detector_colored_tables*). Finally, we used the nodes from assignment 1 for moving Tiago and the detection of the cylindrical tables (*server* and *findCenter*).

3 Human node and Thiago Movement

The movements of the robot is accomplished by invoking the actionServer from assignment 1, where we transmit the position relative to the map frame as (x, y, θ), with θ representing the rotation about the z-axis.

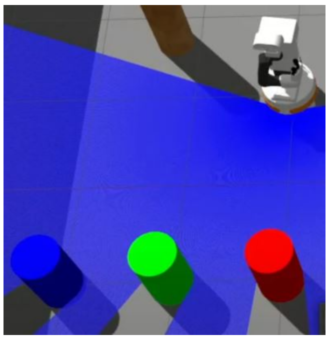
We initiate a call to the human node through the service named `human_objects_srv`. A service client is established for this purpose in the `manager`, which sends a request to the `human_node` and receives the objects IDs in the sequence that the robot must follow during the pick-and-place phase.

4 Position of colored tables

For getting the positions of the colored tables we implemented a two phase method:

1. in the first phase, we move the robot near the tables and from the `manager` we call a service named `detector_colored_tables` controlled by the node with the same name. Once a request is received, this node subscribes to the topic in which Tiago's camera publishes the images and wait for one of them. When this happen, by using Opencv it applies three filters by colors (one for each possible color) for getting three masks of the original image. Then, by using `cv::bitwise_and` it gets for each table, one image with all pixels setted as black exept for the ones corresponding to that table (setted as white). Later, it calculates the mean of the white pixels for each image and then compares the three mean pixels for understand the order in which the tables appear in front of the robot (from right to left).

2. in the second phase we implemented a routine for achieve the bonus points. We used the node `server` from assignment 1 for receiving the positions of the centers of the tables, computed by using `laserScan` and the method for finding the center explained in the previous report. Since the obtained position were not very precise, we decided to do two detections of the colored tables with `laserScan` from different perspectives (as depicted here [1]) and compute the mean values to make the results more robust.


(a) First scan

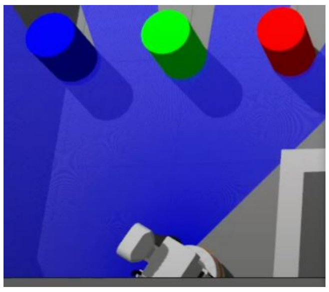

(b) Second scan

Figure 1: Utilize two detections from assignment 1 to enhance robustness

5 AprilTag

Once we receive the IDs of the objects to pick from the `human_node` and found the positions of the colored tables, for each id we move the robot to a fixed position near the table where the objects are located. Then, using an action client that connects to the action server of type `control_msgs::PointHeadAction`, we send the goal position of Tiago's head to enable the robot's camera to focus on the objects on the table.

Subsequently, a service named `detection` managed by `nodeB` is used. From the `manager`, we invoke this service, which, upon receiving a request, subscribes to the topic `tag_detections` and waits for a valid message (until a message with at least one detection arrives). When this occurs, we iterate over all the detections (all the detected April tags from the image published by the robot's camera), and for each of them, we record the information about the ID and the pose (the latter is converted from `xtion_rgb_optical_frame` to `base_footprint` using `tf2`). This information is stored in a vector, which, upon receiving a valid message, is copied into the response of our service.

6 Pick and place

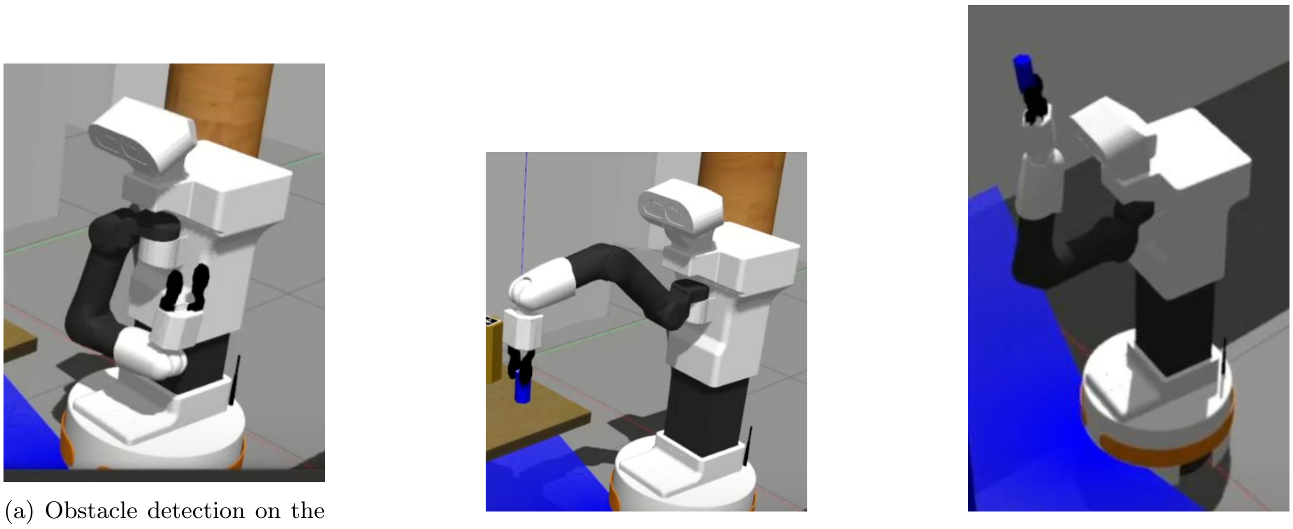
The pick and place problem is implemented and performed by `arm_controller` and it is called by `manager` using the `pick` and `place` services.

6.1 Pick

When the service request is created the objects detected (Fig. 2a) on the table is added in the `detection` vector.

The pick phase is done in these steps:

1. from the request we extract:
   - the pose and the ID of the object to pick (goal).
   - the pose of all the objects passed to build the collision objects (goal and obstacles).
2. add in the collision objects the ones detected and the squared table.
3. from the goal pose we compute the pre-grasp pose adding an offset in z-axis and we move the arm in that pose.
4. remove the object to pick from the collision objects usign its id.
5. move to goal pose (Fig. 2b), close the fingers and attach the object to the `arm_7_link` by using `link_attacher_node`.
6. move to the post-grasp pose, obtained by adding another offset with respect to the pre-grasp pose.



(a) Obstacle detection on the table

(b) Goal pose

(c) Safe pose

Figure 2: TIAGo poses

7. move to an intermediate pose and to the safe pose (Fig. 2c) before moving TIAGo to another position on the map.

8. delete all the collision objects from the planning scene.

6.2 Place

After moving the robot to the correct place pose in the map, the service request is created by adding the ID of the object to place and the position of the cylindrical table for the specific object.
The place phase is done in these steps:

1. from the request we extract the position of the table and we add it to the collision objects of the planning scene.

2. we move the arm to the place position (only moving it in the x axis and just above the table in the z axis).

3. open the fingers and detach the object using `link_attacher_node`.

4. move the arm to the safe position again and return in the first room for the next object.

5. delete all the collision objects from the planning scene.

7 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, Federico focused more in implementing a pick-and-place routine, David in assembling everything and coding the color and objects detection and Umberto helped with the detection of the tables and the debugging.