Intelligent Robotics - Assignment 2
Group 09

Università degli Studi di Padova
DEI - Department of Information Engineering
Academic year: 2023/24

Andrea Campagnol
andrea.campagnol.1@studenti.unipd.it
Daniele Fabris
daniele.fabris.5@studenti.unipd.it

Link to the repository
Link to the additional material

---

1 Introduction

1.1 Purpose

The purpose of the assignment is to implement a routine that allows Tiago to pick up and place each of three different objects, while moving freely in the environment, avoiding collisions. The sequence of the objects to pick up and place is casual and given by a "human" node. In short, for each object, Tiago must reach the table, detect the object, pick it up, go to the relative placing position, and place it correctly.

1.2 Brief explanation

Our implementation is based on 4 principal nodes:

- `node_A`: the master node that communicates with the human node and coordinates the other nodes.
- `node_B`: responsible for the object detection.
- `node_C`: responsible for the arm and torso motion (pick and place action).
- `server_tiago`: responsible for the Tiago movement in the environment (from `assignment_1` package)

We would like to underline that `assignment_1` has not been changed, which underlines the modularity and the scalability of ROS.

1.2.1 Tiago Motion

The motion in the environment is managed by the `tiago_server` node, the same of the `assignment_1`. Refer to the `assignment_1` report to find the information regarding its implementation.
Tiago has to reach 4 main positions in order to complete its task:

- get out the narrow corridor;
- go in a specified point close to the table (one for each target object to pick);
- go in the second room in a point that allows Tiago to observe all the 3 cylindrical tables (in order to select the correct one);
- go close to the target table to place the object.

As noted in the annotations we have set some way-points, due to Navigation ROS Stack's slowness.

1.2.2 Tag Detection

We have implemented a dedicated node (`node_B`) to solve the AprilTag detection.

The `node_A` communicate with the `node_B` through a Client/Server structure, where the (`node_B`) receives the tag ID of the target object from the (`node_A`) and, then, returns as result the pose of the target object and the other detected objects (if found). In order to reach the goal specified, the following operations are executed:

- moving its torso to rise the point of view of its camera;
- lower its head in order to look at the target object and other close objects over the table;
- identify among the various objects which is our target and save all the poses;
- transform the poses in the right robot frame (`base_footprint`).

1.2.3 Pick and Place

We have implemented a dedicated node (`node_C`) to perform the motion of Tiago's torso and arm in order to pick and place the objects, using a Client/Server structure that connects `node_A` and `node_C`. The `node_A` passes all poses (and other minor) information (received by the `node_B`) to the `node_C` which, using MoveIt, execute different collision free plans to move Tiago's torso and arm in order to pick and place objects.
Actually the picking and placing actions require to consider various factor:

- the definition of a set of collision objects to avoid while performing the motion plans;
- the computation of the approaching, grasping, placing and safe poses, considering some margins;
- the execution of a linear movement in order to reach/leave the grasp pose;
- the opening/closing of the gripper.
- the virtually attachment/detachment of the object.

2 Strategies

2.1 End-Effector Picking Position and Orientation

The target's picking position is automatically computed using the position and the orientation of the AprilTag marker, given by the `tag_detection` topic. We consider some offsets for the position coordinates, while we apply a transformation to the orientation (quaternion), using RPY, and keeping the yaw angle of the orientation the action goal give us (the one of the tag). The picking is computed from above for all of the three target objects.

It should be noted that we save the height of the point where the object is grasped with respect to the surface of the table; this value is going to be used later for a more precise placing.

2.2 EXTRA POINT: Placing Pose

After reaching the position where the cylindrical tables can be seen, we know from the `server_tiago` of `assignment_1` the position of three tables (the center of the cylindrical tables). In order to find the right table, the `node_A` call the topic `/xtion/rgb/image_raw` and retrieve an image, which is analyzed in HSV color.

We respectively apply different masks to isolate the red, blue and green colors in the image, and depending on the order in which the rightmost pixel of the respective color appears (by examining the columns of the image), we order the positions of the colors in the image. Then, depending on the target's we need to place, we identify the position of the color and associate it with the coordinates of the respective cylindrical table.

Furthermore, to move Tiago in front of the cylindrical table, a transformation of the `map` frame coordinates is carried out (with an offset in order to don't collide with the table), while for the positioning of the object

we transform them in the `base_footprint` frame.

2.3 Motion performing

We have implemented many different ways to perform the motion relative to Tiago's head, arm and torso.

2.3.1 Arm

In order to move the arm we use different methods, using the joints or directly the `geometry_msgs::Pose`, all based on the `MoveGroupInterface::Plan` object. Particularly, in the case of motion from the approach pose to the grasping pose and vice versa, we perform a linear motion, computing the Cartesian path.

2.3.2 Head

To move the Tiago's head (make it look down at the table to detect the target and its close objects and to look straight after finish the picking action) we utilize the `/follow_joint_trajectory` in `/head_controller` topic setting the two head joints, and working on the trajectory. In the case where Tiago looks down to detect the target and its close objects, the two head joints' values are passed within the `Detect` action goal, considering each object has its respective height.

2.3.3 Torso

To move the Tiago's torso we utilize the `/follow_joint_trajectory` in `/torso_controller` topic similarly for the head movement, obviously considering the only joint available, to move it up or down.

3 Annotations

- Due to the amount of time that the navigation ROS stack sometimes takes to reach the indicated positions, we have added some way-points in the route, to avoid stall conditions in which the robot moves "randomly".
- The positions near the picking table predefined, each one in front of the target object to be picked.
- The picking table's collision object and the placing tables' ones are defined with the help of determined values.