UNIVERSITY OF PADUA

INFORMATION ENGINEERING DEPARTMENT (DEI)

MASTER'S DEGREE IN COMPUTER ENGINEERING

Report Assignment 2

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/1b5M-d4pXXdlHBrVRtp62zrt9C6iMgcvV/view?usp=sharing

Contents

1 Introduction

Our solution to the assignment uses a series of servers and functions that provides basic actions of the robot. Those functions are then used in the main function in node A to carry out the required tasks. Other than that, there are a series of constant waypoints that the robot follows in the correct order, depending on the objects that it has to pick up and place. The main sections of the

project are:

- The detection servers (node C)
  - Obstacle extractor
  - Pose table detector
  - Poses detection
  - Poses transformer
  - Table objects detection
- The manipulation servers (node B)
  - Head movement
  - Torso lifter
  - Pick-Place
- The node A and its functions
  - Ids to Waypoints
  - Move robot
  - Lift torso
  - Move head
  - Object detector callback
  - The main function

2 Node C: detection

In this node there are the classes and servers that do the detection of varius kind of objects:

- Obstacle extractor employs Split And Merge or Iterative End Point Fit algorithms. Additionally, there are parameters that control the behavior of the obstacle extraction process. The class provides methods for processing points and extracting segments and circles, as well as merging them based on certain conditions. The overall process involves detecting and processing groups of points, forming segments, and eventually circles, with options to merge and filter them based on specified criteria. These operations are the same as the one used for the first assignment.
- Pose table detector function captures the image of the environment and, by performing a series of thresholds and point extraction operations, determines the mode (biggest, smallest, or middle) based on the specified color (blue, green, or red). It then sorts the circles received in the request based on their x-coordinates and, finally, returns the requested circle based on the calculated mode.
- Poses detection publisher method captures AprilTag detections, send it to Poses transformer and publishes the results on a specified topic. Additionally, it establishes a ROS service server for transforming poses between specified frames.
- Poses transformer method is responsible for transforming a given pose using a ROS service. Upon receiving a transformation request, uses the TF2 library to look up the transformation and applies it to the provided pose.

- Table objects detection provides methods for detecting and handling objects on a table and detects the table itself. The constructor initializes an action server, and various methods are implemented for moving the robot's head, capturing images and interacting with detected objects. The startDetection method orchestrates the entire detection process, including pointing to the table, colored objects, and other obstacles.

3 Node B: manipulation

In this node there are the classes and servers that do the detection of varius kind of objects:

- Head Movement: this class is dedicated to controlling head movements. It contains 3 main methods, the one to initialize the various parameters including the intrinsic parameters of the camera for the ROS topic. The one for the movement of the head along the Z axis. Which allows the robot to get up to start scanning the surrounding environment. And the last one to move the head along the designated trajectory based on different cases.
- Pick Place manage pick-and-place actions of the system. There are a series of functions to:
  - Define a collision object for every object detected on the table to assure a collision-free trajectory during the movements of the arm
  - Move the arm in a default pose automatically
  - Let us attach or detach an object from the robot hand
  - Move the arm to the right position needed to pick up a given object (the final position depends on the type of object that has to be picked up)
  - Open or close the hand of the robot
  - Move the arm to the right position needed to pose a given object (the final position depends on the type of object that has to be picked up)
  - The callback function that uses all the others in the correct order based on the request of the client
- Torso Lifter is a class dedicated to the movement of the torso. It receives a request to raise the torso to a specific position and then returns feedback when the action is completed.

4 Node A: main

The node A contains the main function that uses all the others servers, classes and functionalities. The main calls the human node to get the order of ids of the objects to pick up, then converts them in a list of poses where the robot must go to pick up them.
For each object the main tells to the robot to:

1. Go to the corresponding pose (pass through a waypoint if needed, based on the object to pick up)
2. Lift the torso
3. Get the poses of the objects on the table
4. Reset the head position
5. Pick the object
6. Lower the torso
7. Move through some waypoints (red and blue objects need an extra waypoint to make the robot go around the table)
8. Once the robot is in fron of the 3 cylinders it can detect the right one (if the extra flag is active) or choose it automatically with constant positions (if the extra flag isn't active)
9. Go in front of the cylinder

10. Lift the torso
11. Place the object
12. Lower the torso
13. Go to waypoint 5 (a default waypoint near the table)
14. repeat for each item

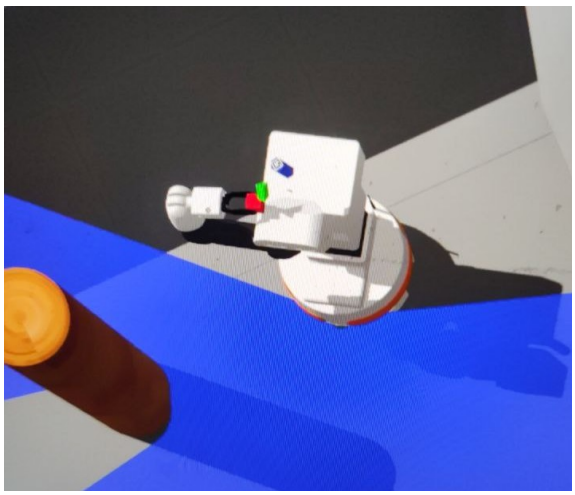
A series of functions allow the main to do complex tasks (like making the robot go to a point) with only one instruction, this makes it very high-level.

5 Final Notes

1. We didn't use our first assignment directly since it wasn't implemented well enough for this second assignment and because we needed to reuse some inner parts (like the circle detection) which we have re-implemented adapting them to the new context.

2. We had a lot of problems with VLab and it was our only way of compiling and testing the project. The problems often stopped us from working to the project leading to a solution that doesn't always work (for example sometimes the robot hits some obstacles).

3. We had some difficulties (that were then solved) with the pick/place routines (see the image). They lead us to a potentially more efficient solution that picks up all the 3 objects at the same time by carrying them on the robot's head.



The robot with all the 3 object picked up at the same time.