Intelligent Robotics course

Assignment 2 Report - Tiago Pick and Place

Prof. Emanuele Menegatti

Group 23: Simone Bastasin, Andrew Zamai & Alessandro Manfè

31st January 2022

1 ROS nodes architecture

We started assignment1 report stating that ROS allows to write well structured, modular and easily extendable code. If it is true that we showed the nice properties of modularity and structure in the assigment1 we can, through this assignment2, show the nice **properties of extendibility and re-usability**. This assignment, in fact, lays its foundations on the navigation nodes developed for assignment1. In particular, in order to make Tiago move, what was required to do, was simply to make the new nodeA connect to our previously defined Proxy and send the goal poses in a similar way the Client node was doing. We did not have to change the previous code, but simply make the new nodeA communicate with the Proxy through ClientProxy.action.

Two new nodes have been defined, as required by the specifications, that are: **NodeB** and **NodeC**, other than the already mentioned **NodeA**. The former performs the object pose detection via AprilTagDetection, the latter implements the pick or place routine, depending on the specified goal. We have decided to implement **NodeB as a service**, since the object tags detection is relatively fast and is a blocking point: until the detection is not finished the robot cannot do anything next. Differently, we implemented **NodeC as an action server**, since both the pick and place routines are relatively long, thus we may want to receive periodical info about the task current status or we may like to preempt the current execution and send another goal.

In Fig. 1 the ROS nodes architecture we designed.

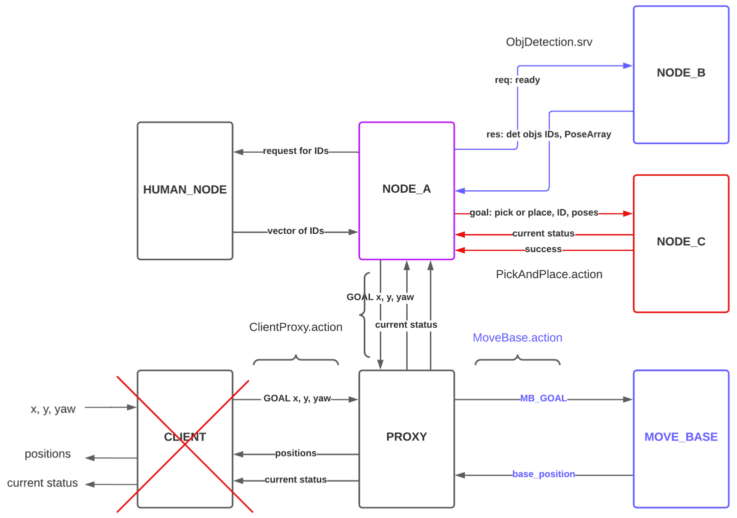

Figure 1: ROS nodes architecture.

2 Work flow

We have above presented the ROS nodes architecture we came up with and the motivations that led us to it. We now briefly describe how the system works, also giving some reasons for the implementation choices we made:

1. nodeA requests humanNode, implemented as a service, to provide the order in which to perform the routines for picking and placing objects from the table to the respective coloured cylinders;
2. Tiago has now to move to the table in front of the first object to pick. We have **assumed an a-priori pose in front of the table** depending on the ID of the object to pick, which Tiago can reach before performing objects detection and pick. The assumption is not on the pose of the objects but on the fact that the objects will not be moved far away from the current simulation neighborhood: even if the object is slightly moved around, the nodeB detection is still able to detect the the tag of the objects in front of it and recover the pose of the object to pick, as well as the poses of the obstacles to be avoided near it. As said, the navigation relies on the Proxy node implemented in assignment1. We had to place few way-points in order to not make Tiago collide with the cylindrical obstacle in the way to the table. Making Tiago pass through the way-points is as easy as simply sending subsequent pose goals to Proxy.
3. before calling nodeB to perform the obstacle poses detection, Tiago is required to **move the head down** in order to look straight at the table. This was achieved by sending a goal to /head_controller /point_head_action. The correct values were found by modifying the look_to_point tutorial in order to print the values when the camera was perfectly capturing the table surface. We make also move the Tiago torso before, in order to not hit the table during the pick routine.
4. nodeA now sends a request to nodeB acting as a service to perform the object poses detection. To do so nodeB subscribes to *tag_detections* topic, waiting for a single message. The detected tag poses are now in the reference frame of the camera, that is *xtion_rgb_optical_frame*. We are required to perform a transformation of this poses from this reference frame to a reference frame suitable for the pick and place routine, that we identified to be *base_footprint*. It is important to observe that we **cannot assume a fixed transformation matrix** but, since the reference frame is fixed to the head and the head can freely move in time with respect to base_footprint, we need to perform such transformation by obtaining the transformation matrix at that time through *TF* package.
5. the poses of the detected tag objects are now sent to nodeC action server as part of the goal definition, together with the ID of the object to pick or place and a boolean variable indicating the routine to perform, whether pick or place. The pick and place routines follow what the suggestion points recommend. One thing that is relevant to be mentioned is the angle we defined for the pitch pose of link 7, that we identified to be $\pi/8$, the yaw depends instead on the object orientation to be correctly grasped.

3 Metrics and conclusions

All the code was written by 6 hands using Simone's and Alessandro's PCs and all members contributed equally to the work. The video in the Google Drive folder shows the robot performing the pick and place routines in the order requested by human node.