Homework 2

DEADLINE: 20th January

Suppose a human user has requested one object. This object has to be detected on a table, among other *n* objects. The pose of the markers on the top of these objects can be obtained from the AprilTag library. The Apriltag library detects the markers through their ids. Each marker has a specific identifier.
The markers are attached on top of each object. Thus, the detected pose is the pose of every marker. Finally, the shape of every object is known (see Figure 1).

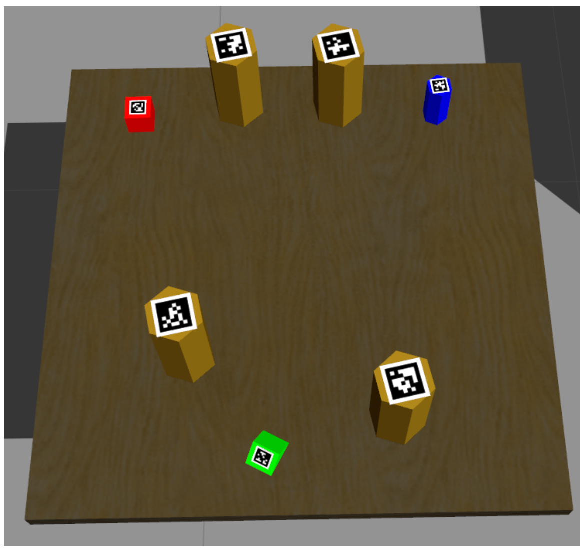
Figure 1

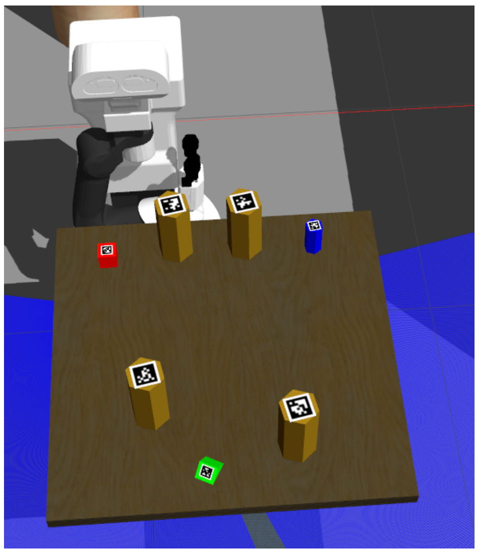
Figure 2

This homework aims to implement a MoveIt! routine for **one object**. Thus, you have to develop a routine to transport the object from the table (Figure 2) to its final destination in accordance with its colour (Figure 3). **Only one among the red cube, the green triangle and the blue hexagon must be taken and placed on the corresponding cylindrical table** (see Figure 3). The gold hexagons in the table (Figure 1 and Figure 2) are obstacles and Tiago must not collide with them (and with the table) while grasping the required targets.

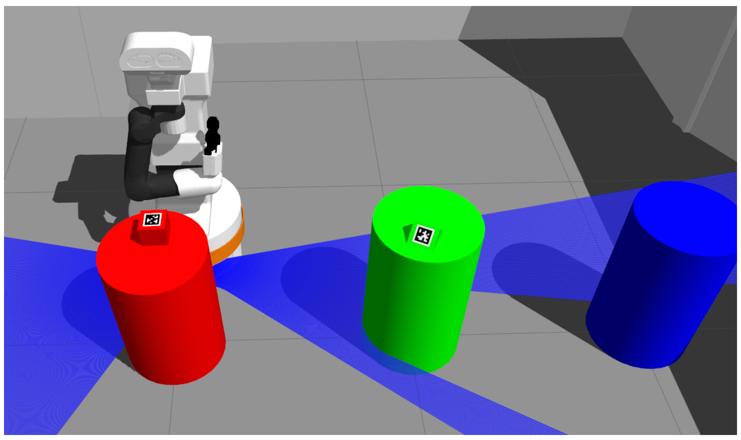
Figure 3

The whole environment is shown in Figure 4.

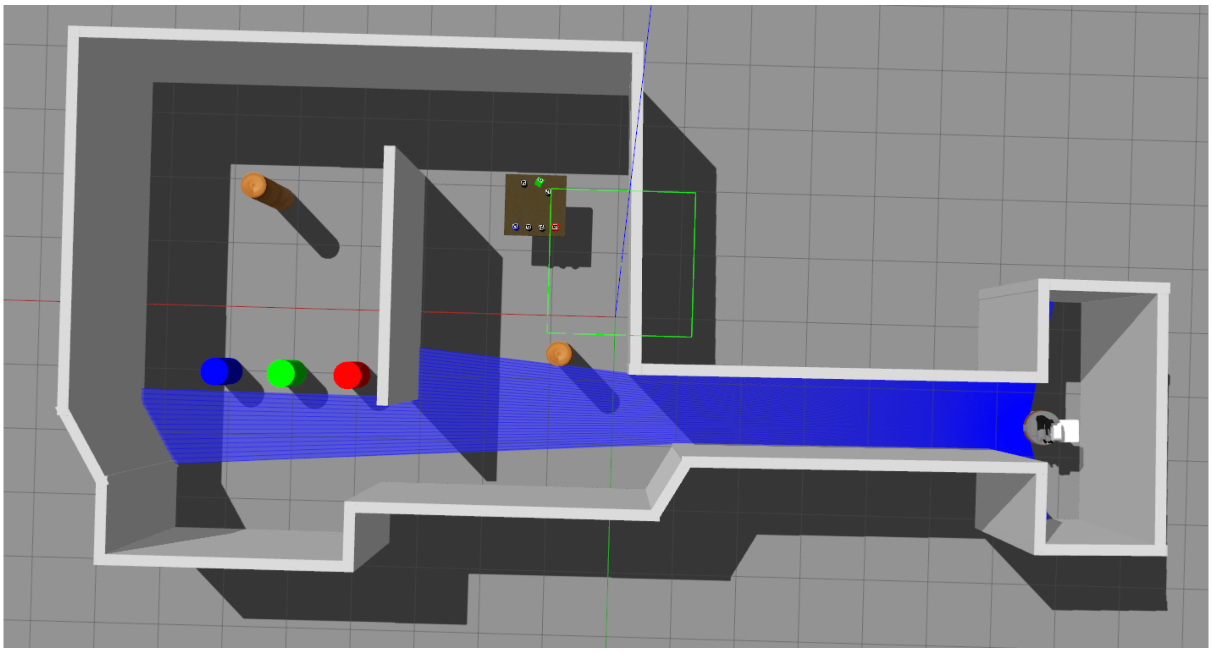

Figure 4

To navigate in the environment and transport objects you must use the navigation module you developed in the previous homework.

SUGGESTION:

We suggest the following pipeline to solve the homework:

1. Navigate in front of the table with the navigation stack (Figure 2). The position in front of the table is up to you.
2. Use the AprilTag ROS package and Tiago's camera to detect the objects in front of Tiago.
3. Define a collision object for every object that you detect and for the table. For the hexagons, you can create a collision object like a cylinder. The collision object helps you (MoveIt) to generate a collision-free trajectory. Traditionally, to ensure safety, the collision object is created a little larger than the real object.
4. Assign an initial configuration to the Tiago's arm. With MoveIt you can move only the arm or both torso and arm. (see Tiago Tutorial and MoveIt Tutorial)
5. Make the arm move in a target position. The target position is set at *n* cm above the marker position (align the gripper with the centre of the marker itself) (Figure 5a).
    a. **N.B:** This is true if you make a grasp from the top of the object. But, especially for the hexagon, you can grasp it from the side. In this case, knowing the hexagon position, you can generate a grasping point, i.e. target position, on its side.
6. Through a linear movement complete the grasping. This routine from 4-6 should help you to ensure the correct grasp of the object (Figure 5b).
7. Remove the target object from the collision objects.

8. Attach the object to the gripper (use arm_7_link) both through MoveIt! and through the Gazebo plugin.
9. Close the gripper.
10. Come back to the previous position through a linear movement.
11. Move the arm to an intermediate pose (Figure 5c).
12. Move the arm to a final secure pose to navigate to the destination.
13. Navigate to the correct destination (Figure 3). Also, in this case, you can decide the position in front of the cylindrical table and use the navigation module to navigate.
14. Place the object on the top of the table.
15. Open the gripper.
16. Detach the object via Moveit! and the Gazebo plugin.

N.B: We know that with Gazebo there are many problems with the attachment of objects. Thus, you can attach the object only in RVIZ and complete the task. In this case, in Gazebo, you can remove the object manually.
If you have problems with open/close and attach/detach, write in your code these routines, but they may not work. However, it is important for the evaluation that they are placed in the right place in the pipeline!

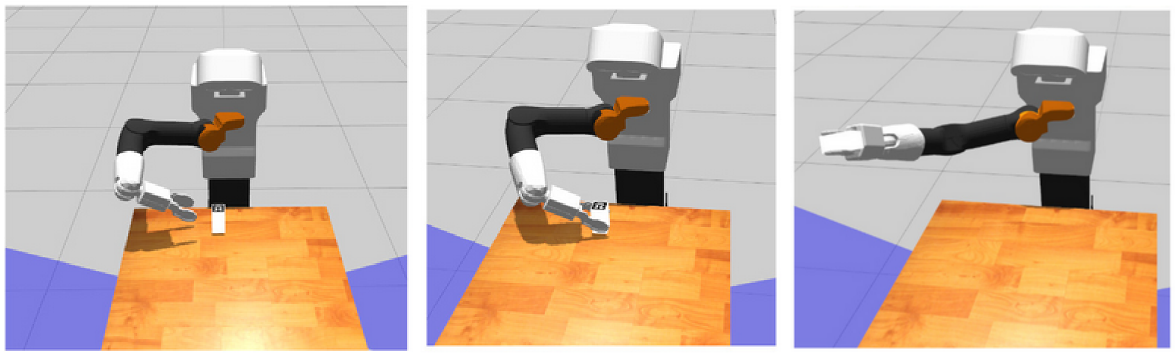
a b c

Figure 5

MANDATORY:
Your code must be implemented using the following structure:

- Create a node (A) that sends a request to the human_node using a service. The latter is a node that simulates a human who asks the robot which object to pick up. Thus, the human_node returns the object's ID that Tiago needs to pick. We provide the file .srv and the human_node that handles the request and sends the response.
- Use your node (A) to call the Action Server of HW1 and navigate to a position in front of the table.
    - N.B: Since you have only three pickable known objects, you can configure 3 global positions in the front of the table to help you to detect the effective position of the objects on the table (Figure 6).
- Implement two new nodes (B, C) to handle the manipulation of the objects and the detection. The communication with these nodes can be developed using messages, service or Action/Service infrastructure. You choose the strategy you think is most appropriate.

    - We suggest you implement in a node (B) the detection. Using the AprilTag, you know the pose of the objects with respect to the camera reference frame. Then, using the TF you have to transform the pose from the camera frame to the robot frame.
    - With the other node (C), we suggest you implement the MoveIt routine.
- **Optional:** Use the laser to implement your docking routine in front of the cylindrical table to stop Tiago.

N.B: Take advantage of the modularity and scalability of ROS.

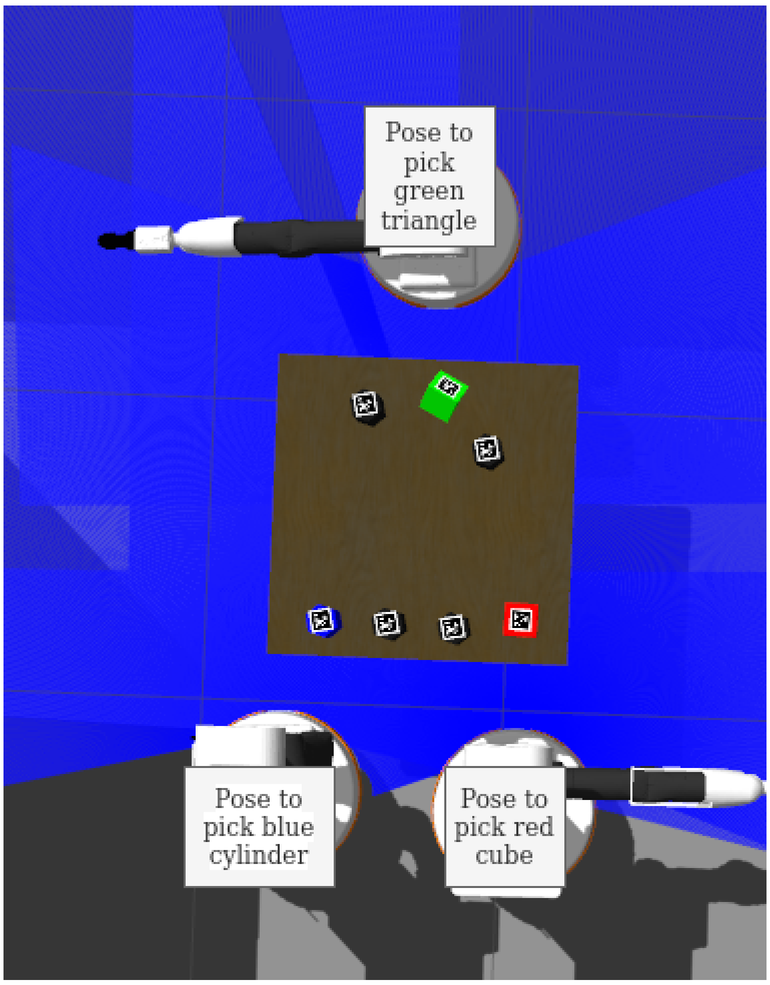

Figure 6

Extra points

What if the user wants Tiago to bring him more objects? You have to generalize your Pick&Place routine! In this homework, to get extra points, we ask you to generalize the routine to carry all three objects (red cube, green triangle and blue hexagon). The sequence to be executed will be given to you using the service.
In this case, you have to specify in the email and in the report that you also implemented this solution.

Instruction to start the homework

Follow these instructions to install the environment and start the homework:

- Clone the following repositories in your workspace (src folder):
    - Apriltag: https://github.com/AprilRobotics/apriltag.git
    - Apriltag_ros: https://github.com/AprilRobotics/apriltag_ros.git
    - Gazebo_ros_link_attacher: https://github.com/pal-robotics/gazebo_ros_link_attacher.git
- We push the homework2 files in a new branch called hw2. Thus, you have to pull the repo and, if you want, merge it into master

```
$> git pull
https://username@bitbucket.org/iaslab-unipd/intelligentroboti
cs_group_XX.git
$> git checkout origin name_branch
```

- Start the simulation and MoveIt:

```
$>  roslaunch  tiago_iaslab_simulation  start_simulation.launch
world_name:=ias_lab_room_full_tables
```

- AprilTag:

```
$> roslaunch tiago_iaslab_simulation apriltag.launch
```

- Navigation stack:

```
$> roslaunch tiago_iaslab_simulation navigation.launch
```

- Human Service node:

```
$> rosrun tiago_iaslab_simulation human_node
```

Instruction to submit your solution

To submit your solution you **MUST** do the following:

- Push your code into your group bitbucket repository;
- Prepare a small pdf report (max. 2 pages) that explains your solution;
- The report, or the readme file in the repository, **must** contain the necessary commands to correctly execute and test your code (e.g. roslaunch and rosrun commands);
- Prepare a short video (e.g. screen record) that shows the execution of your homework. This is necessary in case we cannot easily reproduce your code;
- Send us an email that confirms the submission and attach all the necessary information (e.g. report, link to the repo, additional info, etc.).
- Optional: if you use Docker you can also submit the docker image. Thus, you have to give us the name of the repository and tag or the link of your repository in Docker Hub.

Additional Information

APRILTAG: We provide an example of the AprilTag marker (Figure 7) and the set of IDs and corresponding frame_ids (Table 1).

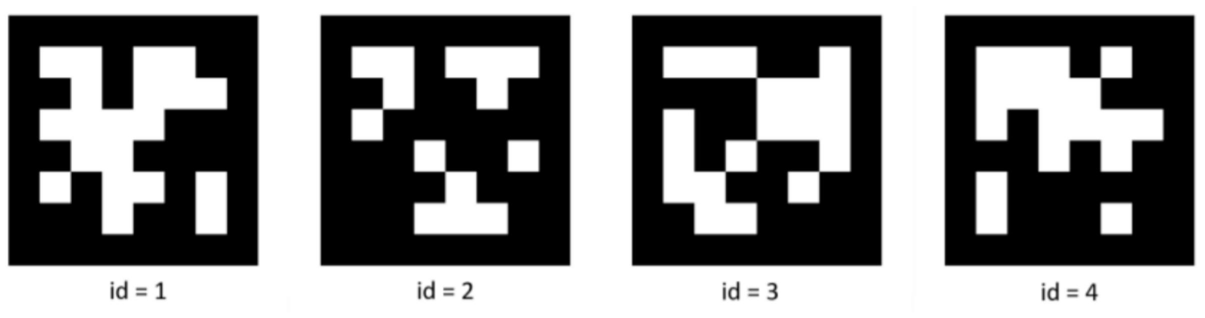

Figure 7

ID	FRAME_ID
1	blue_hexagon
2	green_triangle
3	red_cube
4	gold_obs_0
5	gold_obs_1
6	gold_obs_2
7	gold_obs_3

Table 1

HEAD ACTION: To detect the objects on the table you have to move the head of Tiago. See these two tutorials:

- http://wiki.ros.org/Robots/TIAGo/Tutorials/motions/head_action
- http://wiki.ros.org/Robots/TIAGo/Tutorials/trajectory_controller

Also, rqt_joint_trajectory_controller can be helpful to find the best position of the head.
Run the following command:

```
$> rosrun rqt_joint_trajectory_controller
rqt_joint_trajectory_controller
```

GRIPPER OPEN/CLOSE: To open/close the gripper you can use the gripper controller interface. See http://wiki.ros.org/Robots/TIAGo/Tutorials/trajectory_controller

RVIZ: In RVIZ you can test MoveIt using the MotionPlanning (Figure 8)

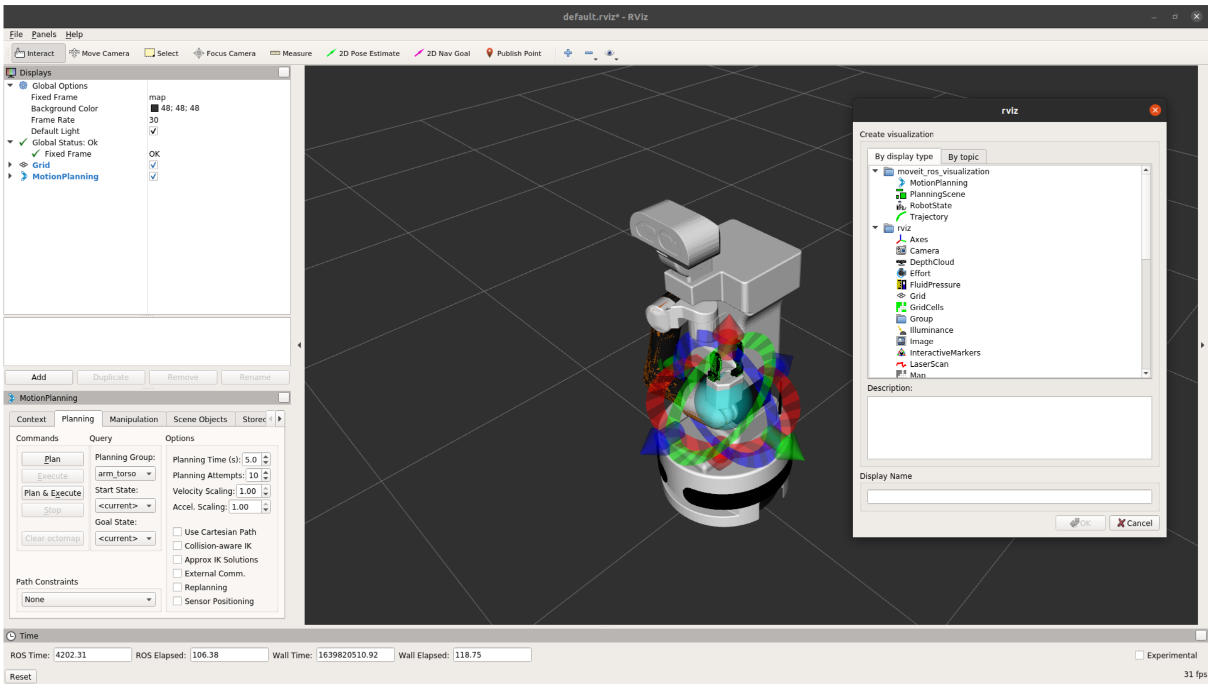

Figure 8

TF: Using rqt you can see the TF Tree (from terminal digit rqt, then in the GUI search plugins-visualization-TF Tree) (Figure 9)

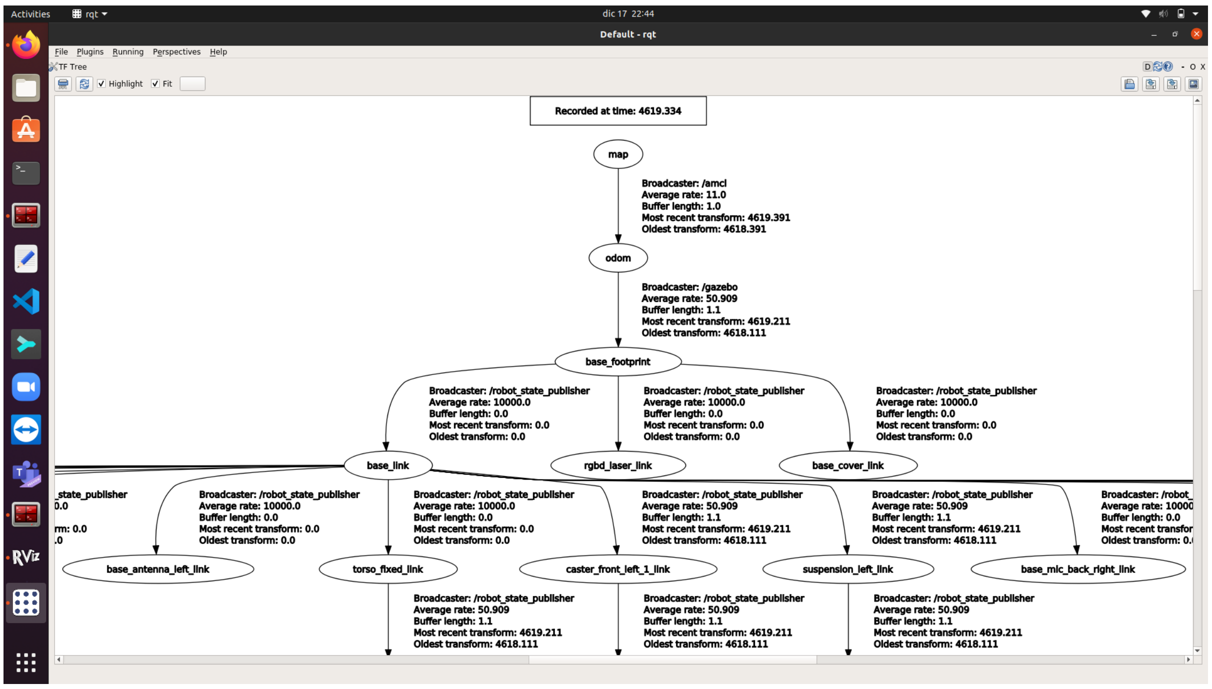

Figure 9