Assignment 2

DEADLINES:

- **Winter Session**: January 20th 2025, 23.59 CET
- **Summer Session**: June 10th 2025, 23.59 CET
- **Fall Session**: August 22nd 2025, 23.59 CET

In this assignment, you have to implement a routine that lets Tiago **pick and place** some objects from one table to another.

In the simulated environment, there are 9 objects on a pick-up table: three red cubes, three green triangles, and three blue hexagons (Figure 1). Each one of them has an apriltag, and the IDs go from 1 to 9. Tiago must pick at least three of them, no matter the color, the shape, or the ID, and place them on the empty table on the right.

The objects **must be placed on a straight line** of equation `y = mx+q`, whose coefficients `m` and `q` are provided by service `/straight_line_srv` on request. The **reference frame** of the line can be found on the placing table, which has an **apriltag** on a corner (**ID 10**). The apriltag's reference frame is the line's reference frame. The distance between the objects on the line is up to you, as well as their orientation, but consider you need to find a position within the table surface.

Tiago must be in a suitable position for picking and placing the objects, not too close to nor too far from the tables. Find these **docking positions and set them as global values**.

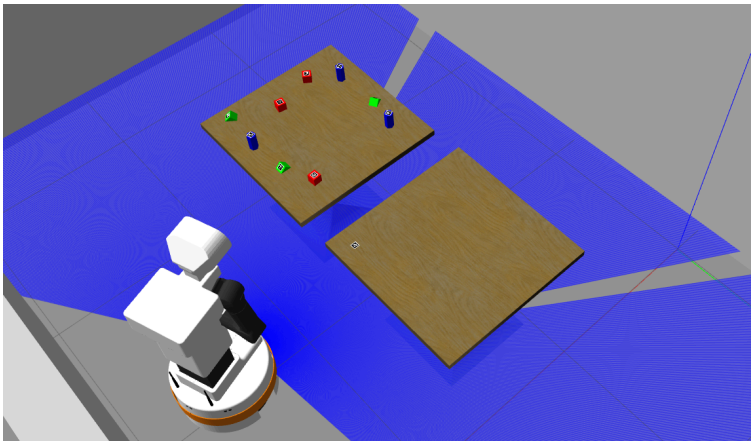


Figure 1

You have to develop a pick-and-place routine, **exploiting the MoveIt! library.**

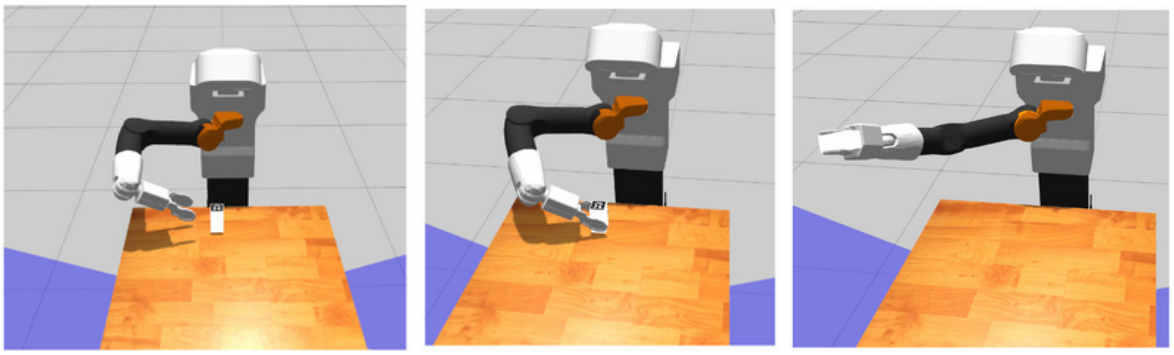
SUGGESTION:

We suggest the following pipeline to solve the assignment:

1. Navigate in front of the pick-up table. 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 you detect (and want to pick) and for the table. For the hexagonal prism, you can create a collision object in the shape of a cylinder; for the triangular prism, you can use a parallelepiped. The collision object helps you (i.e., the module MoveIt) to generate a collision-free trajectory. Traditionally, the collision object is created a little larger than the real object to ensure safety. (more info for collision objects is below)
4. Assign an initial configuration to 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 2a).
    a. **N.B:** This is true if you grasp the object from the top. 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 on its side, i.e., the target position.
6. Complete the grasping through a linear movement. This sub-sequence from 4-6 should help you implement the correct grasping of the objects (Figure 2b).
7. Remove the target object from the collision objects.
8. Attach virtually the object to the gripper (use arm_7_link) using the appropriate Gazebo plugin: Gazebo_ros_link_attacher (See the section **Instruction to start the homework**).
   (This is a trick to avoid singularities in the physical engine of the GAZEBO simulator, which is not good at simulating the friction and so it might happen that the robot grasps the object and the object slips away from its fingers.)
9. Close the gripper.
10. Come back to the previous target position through a linear movement of the arm.
11. Move the arm to an intermediate pose (Figure 2c).
12. Move the arm to a secure pose to avoid collisions before navigating to the destination (e.g., fold the robot's arm close to the robot's body).
13. Navigate to the correct position for placing the object.
14. Place the object on the top of the table, in a position (x,y) belonging to the line **y=mx+q** found at the beginning, with respect to the reference frame of the apriltag on the corner of the table.
15. Open the gripper.
16. Detach the object via the Gazebo plugin.

N.B.: We know that Gazebo has many problems with objects' attachment. Thus, you can attach the object only in RVIZ and complete the task. In this case, you can manually remove the object from Gazebo.

If you have problems with open/close and attach/detach, do not worry, leave these routines in your code, even if they are not performing properly. In the evaluation, we will consider if they are placed in the right place of the pipeline and not if they generate strange behaviors of the objects. Please, specify in the report which problems you encountered anyway.



a b c

Figure 2: Example of the pick routine.

MANDATORY:

Your code must be implemented using the following structure:

- Create a node (A) that sends a request to the /straight_line_srv using the service Coeffs.srv. The latter will provide coefficients m and q of the straight line you need for the placing routine. Implement here the placing routine.
    - Remember that when you choose the point (x,y) belonging to the line, it is defined with respect to the ref. frame of the apriltag of ID 10. For placing the object you need to use the TF to transform the pose from this frame to the robot frame.
- Implement two nodes (B, C) to handle the detection of the objects and their manipulation.
    - Remember that 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.
    - Once you find the object's position in node B, send it to C and implement the MoveIt routine for manipulation.

Communication between these nodes can be developed using messages, service, or Action/Service infrastructure. You choose the strategy you think is most appropriate.

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

Extra points n. 1 (1.5 points)

Substitute the global docking positions with a routine that uses the laser data to find the two tables and implements a docking routine.
In this case, you have to specify in the report that you also implemented this optional solution.

Extra points n. 2 (1.5 points)

Implement a routine to detect the color of the objects and pick up those that have the same.
In this case, you have to specify in the report that you also implemented this optional solution.

Instruction to start the homework

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

In this assignment, you will need the following:

- 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

Some of them were required for the first assignment. Thus, you probably already installed them. Check it out! In case you do not have them, please retrieve the instructions for installation (Local and VLAB) in the first assignment.

The tiago_iaslab_simulation repository that you cloned in your ws for the first assignment **MUST BE** updated in the following way:

- **Re-clone** the tiago_iaslab_simulation repository

```
$> cd ~/catkin_ws/src
$> git clone
https://github.com/AnnaPlt/tiago_iaslab_simulation.git
```

- To remap to a new repository it can be helpful remove the old git blob:

```
> rm -rf tiago_iaslab_simulation/.git
```

Then:

- Go inside the workspace/src and create a new package

```
$> cd ~/catkin_ws/src
$> catkin_create_package assigment2_package_name
```

- Build and source

```
$> cd ~/catkin_ws
$> catkin build
$> source ~/catkin_ws/devel/setup.bash
```

- Start the simulation and MoveIt:

```
$> roslaunch tiago_iaslab_simulation start_simulation.launch
world_name:=iaslab_assignment2
```

- AprilTag:

```
$> roslaunch tiago_iaslab_simulation apriltag2.launch
```

- Navigation stack:

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

- get_straightline_node:
  `$> rosrun tiago_iaslab_simulation get_straightline_node`

NOTE: given the complexity of the pipeline, we advise you to prepare a single launch file that calls all the necessary nodes and other launch files. This will also speed up the development. Have a look at the ROS documentation if you don't know how to do it.

How to submit your solution

To submit your solution, you **must** do the following:

1. Setup your group repository

Use the same repository you set up for the Assignment 1. The repository must contain only the packages you developed for the assignments. Please, **use different packages for Assignment 1 and 2**.

2. Start coding

a. **Push your code (with good comments)** into your group's git repository. Code explainability will be part of the evaluation criteria.
b. We encourage you to commit the code as you write it and not in one block
c. **Add a README.md file** which contains the necessary commands to correctly execute and test your code (e.g. *roslaunch* and *rosrun* commands). Make sure the instructions are working properly. **The first lines of the README.md must contains the following informations**:

   GROUP XX
   Member1's name, email
   Member2's name, email
   Member3's name, email (if any)

   You can create a new README.md inside the Assignment 2's package or update the previous README.md, just **clearly distinguish instructions to run Assignment 2 nodes**.
d. We suggest you to test your code step by step in a new clear workspace, cloning the code from your repository in order to be sure that your code is executable by third parties.
e. **Make sure your code is compiling before submitting**. Not compiling codes will be penalized in the final grade.

3. Submission

1. **You are allowed to push the code in the repo until the submission on Moodle**. Later commits will not be considered for the evaluation.
2. To get the maximum grade, you have to submit it within the 20th January 2025 at 23:59 (CET). Later submissions will be penalized.
3. **Prepare a brief PDF report (max. 2-3 pages)** that explains your solution. The report is supposed to explain the high-level ideas, strategies or

algorithms you implemented to solve the assignment. **The first lines of the report must be like**:

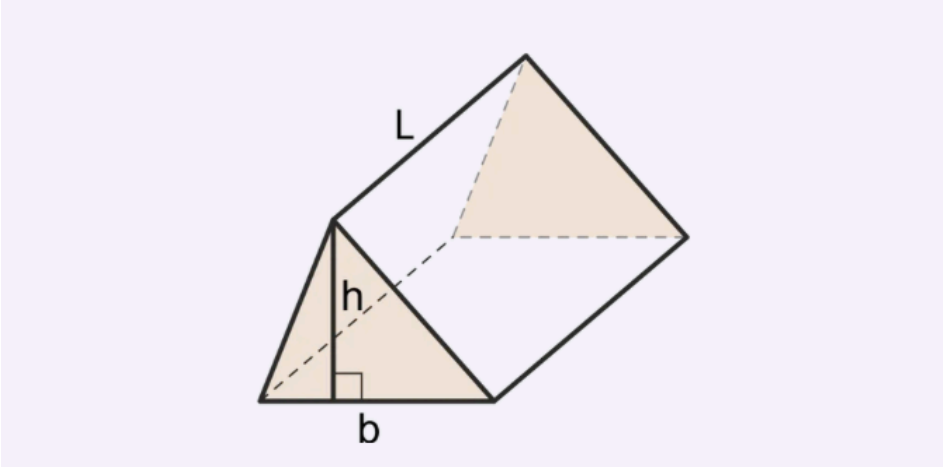
```
GROUP XX
Member1's name, email
Member2's name, email
Member3's name, email (if any)
LINK TO THE REPOSITORY
LINK TO ADDITIONAL MATERIAL (see point 5)
```

4. **Prepare a short video** (e.g., screen recording) that shows the execution of your software by the robot. This is necessary in case we cannot easily run your code.
5. **Report and video must not be placed in the repository**. We suggest you create a Google Drive folder, upload the video or any additional material and share with us the link to the folder inside the report.
6. We will open a submission to Moodle. You will be able to attach the report to your submission (be sure that it contains all the needed information such as a link to the repo, link to the video, additional info, etc.). **The name of the submission must be as follows: GroupXX_A2**. Please submit only once per group, one member can submit for everybody.

Additional Information

SHAPES OF OBJECTS:
You can exploit the following information for developing your code, especially for building the collision objects:

Apriltag ID	Shape of the objects	Measures
1,2,3	hexagonal prism	h: 0.1, r: 0.05 m
4,5,6	cube	l: 0.05 m
7,8,9	triangular prism*	b: 0.07, h: 0.035, L: 0.05 m



TABLES:
pick-up and place **tables** have dimensions: h: 0.9, l: 0.9. you can represent them as a cube. Remember to build the collision objects a little bit larger than real ones.

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 3)

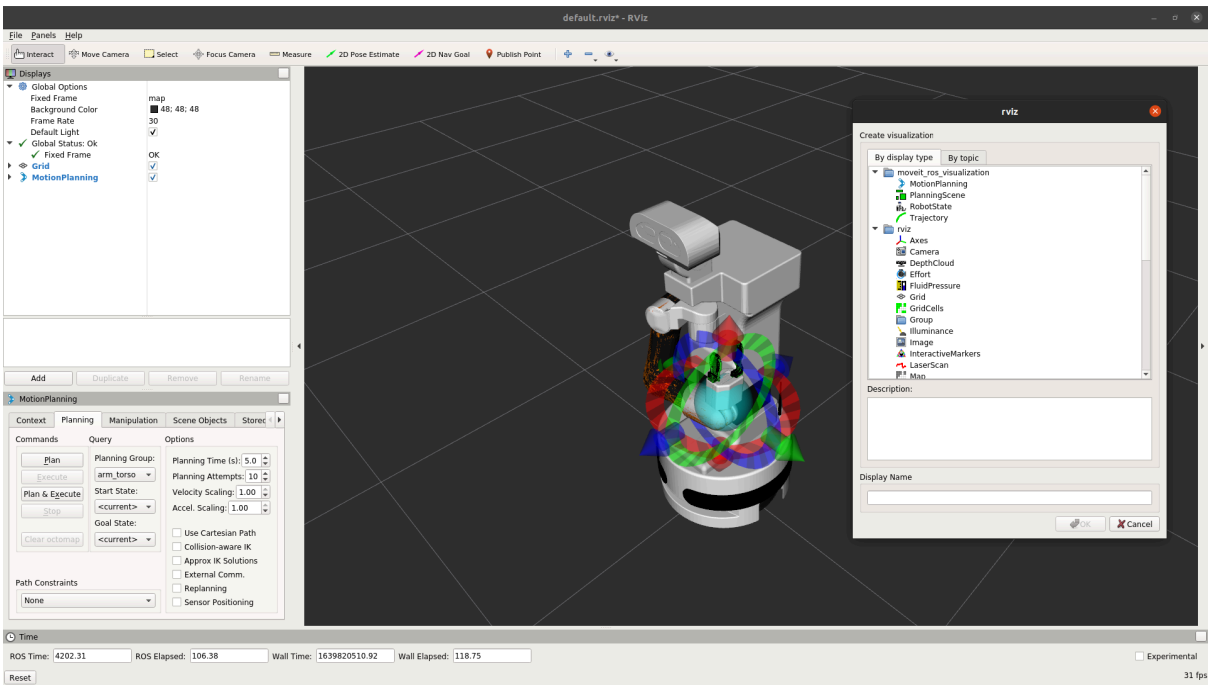


Figure 3

TF: Using rqt you can see the TF Tree (from terminal digit rqt, then in the GUI search plugins-visualization-TF Tree) (Figure 4)

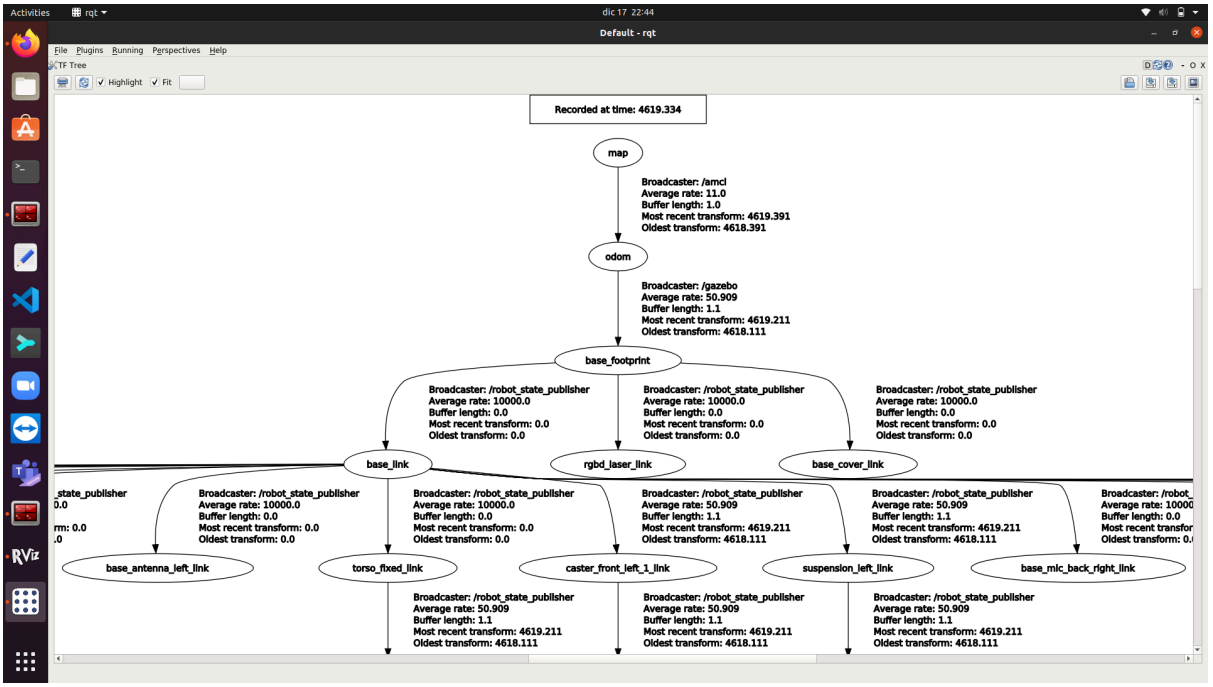


Figure 4