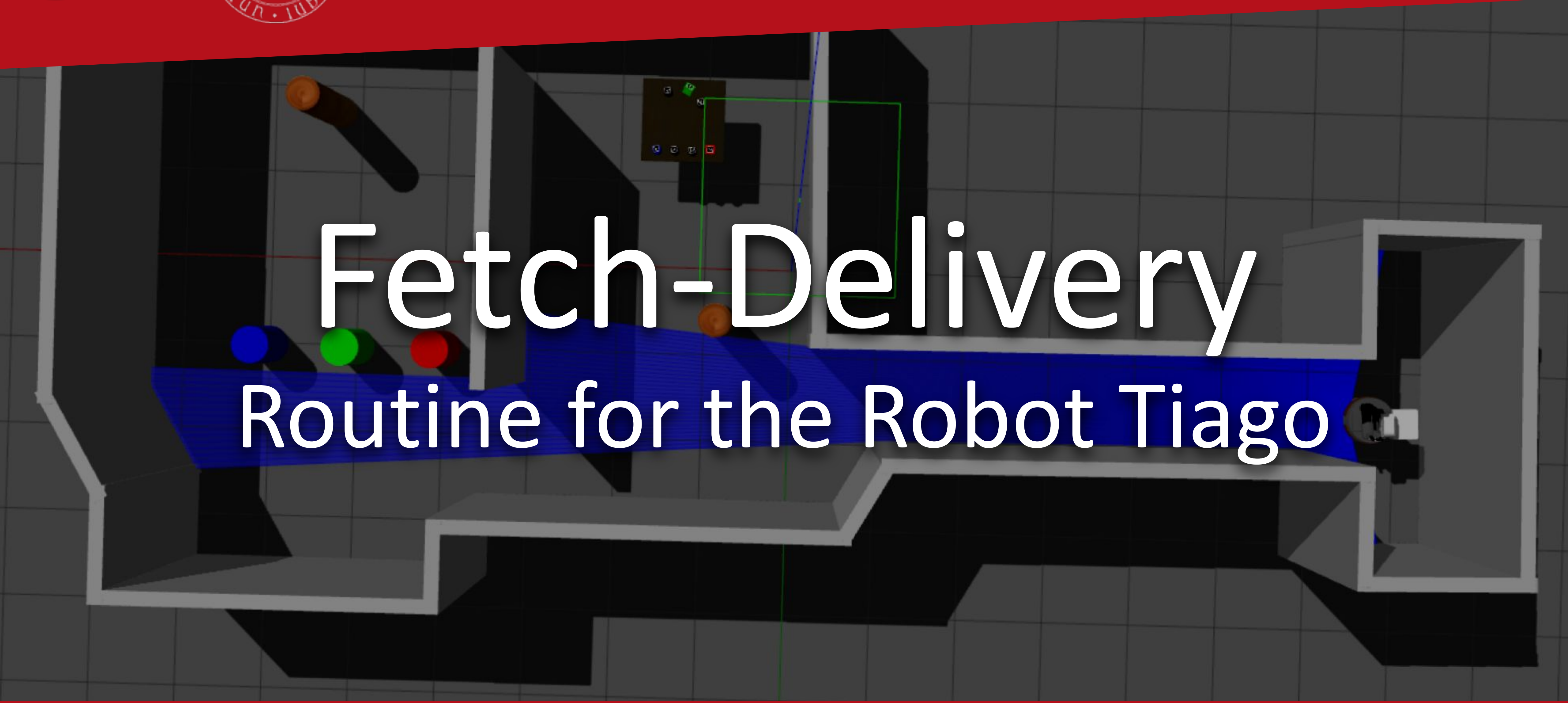
Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

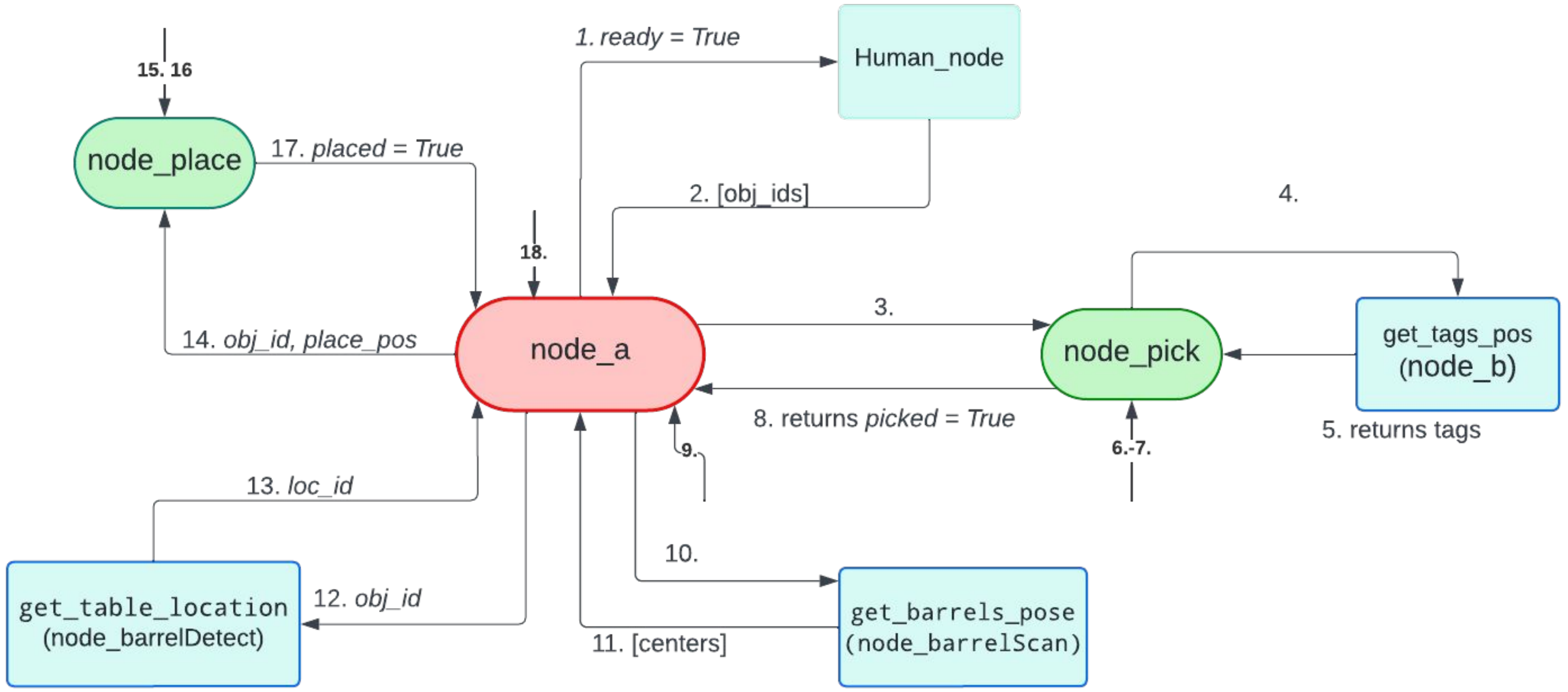


Fetch-Delivery

Routine for the Robot Tiago

07 Feb 2024

PROGRAM STRUCTURE



🟥 Central Node

🟩 Action Server

🟦 Service

ROUTINE

For each of the 3 IDs the following **Routine** is accomplished by Node A:

1. Human node $\longrightarrow$ **ID sequence**
2. Node Pick + Node B $\longrightarrow$ **Object picked**
3. Node barrelScan + Node barrelDetect $\longrightarrow$ **Target table** [AUTOMATIC DOCKING ROUTINE]
4. Node Place $\longrightarrow$ **Object placed**

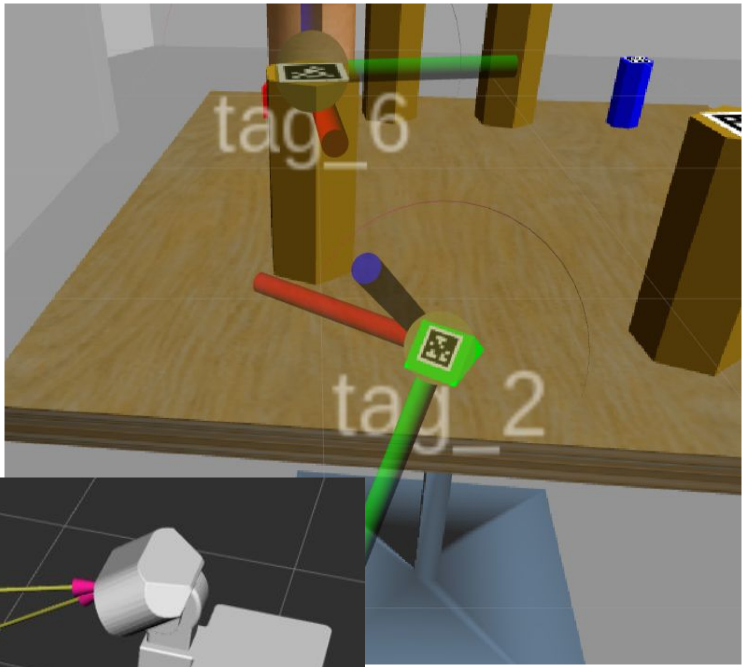
OBJECT POSE DETECTION

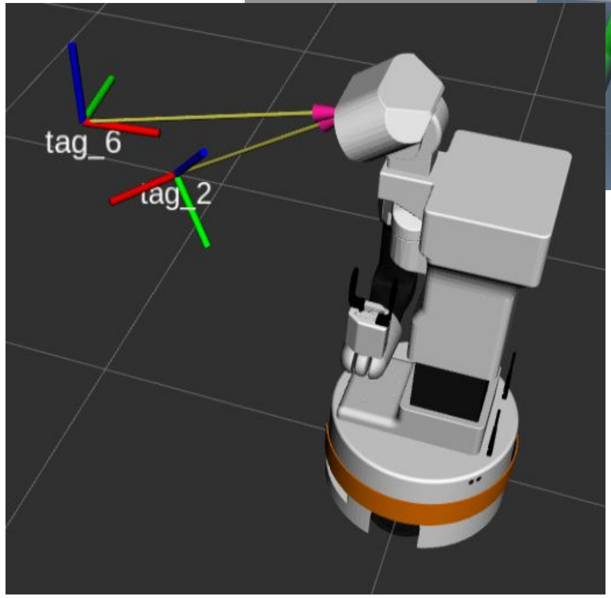
AprilTag detection

The aprilTag pose detection was implemented as a service `"get_tags_pose"`:

- **Tilts** Tiago’s head toward table with custom `moveHead()` function
- **Waits** for the `apriltag_ros::AprilTagDetectionArray` message from the `"/tag_detections"` topic
- **Converts** the poses of detected tags from `"xtion_rgb_optical_frame"` to `“map”` reference frames
- **Tilts** back head

Implemented in `node_b`



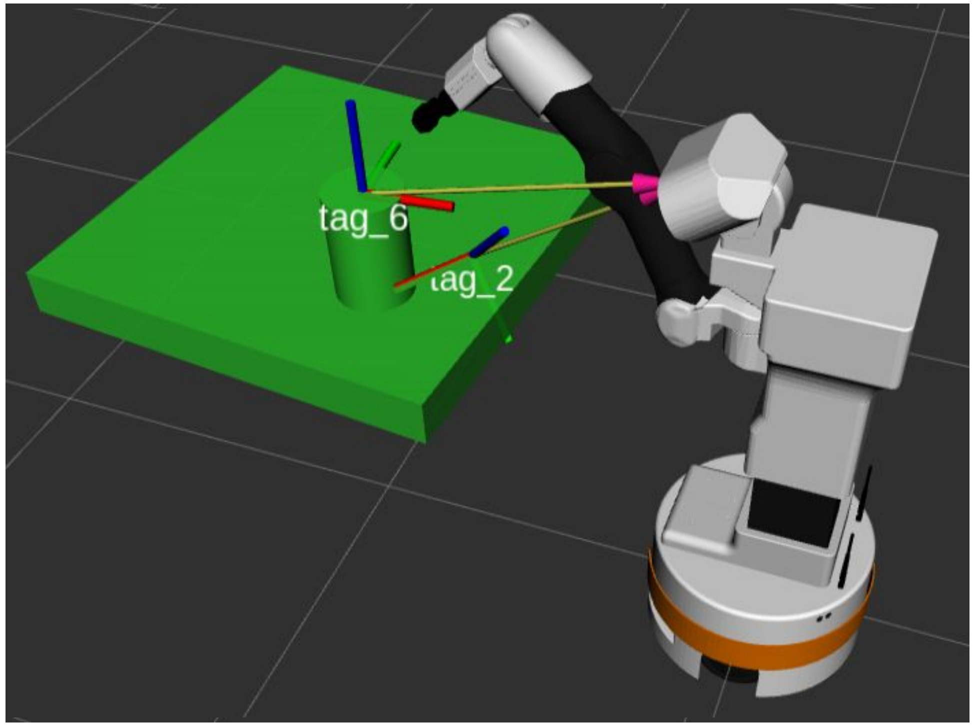


COLLISION OBJECTS

Pick Phase

Collision Objects (c.o.) are created as to avoid our robot arm crashing into objects, the steps are:

- **`node_pick`** calls the srv and obtains tag's **poses**
- Creates a hard-coded c.o. for the table
- Creates **cylindrical c.o.** for all tags with ID not corresponding to the **goal pickObj**
- Creates **custom c.o.** (cylindrical or cubical) depending on which object we are picking
- Add c.o.'s to `moveit::planning_interface::PlanningSceneInterface`
- After the approach phase is finished, **removes** c.o. of the object to be picked from the interface



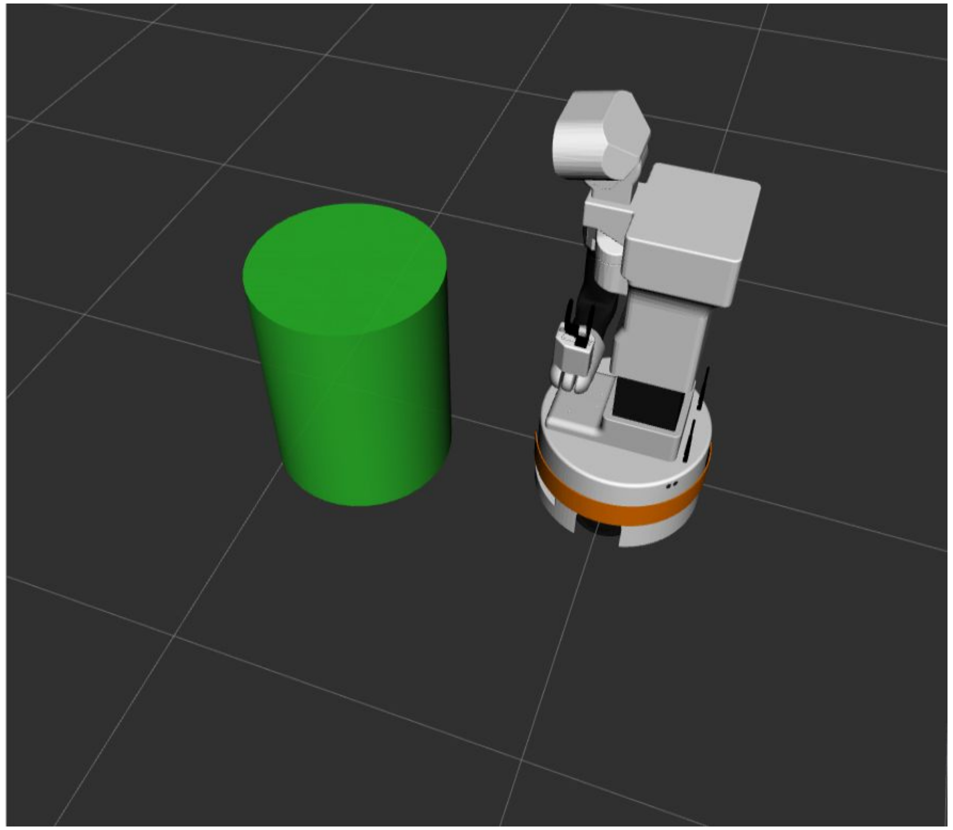
Implemented in `node_pick`

COLLISION OBJECTS

Place Phase

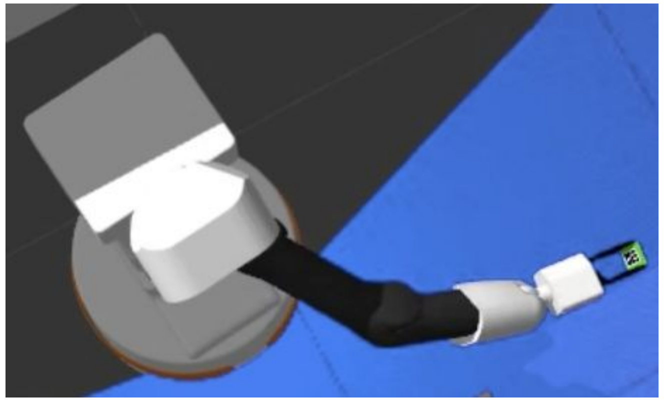
For the place phase we need the c.o. for the table where the object have to be placed:

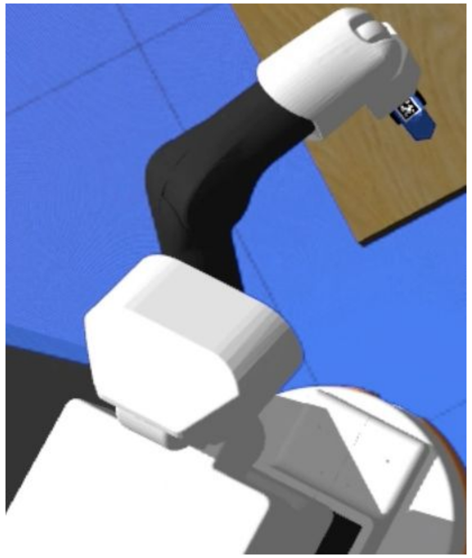
- **position** of the table is given as the **goal** of the place action function from `node_a`.
- Creates cylindrical c.o. with size slightly larger than the barrel/table and add it on the `moveit::planning_interface::PlanningSceneInterface`
- After the place phase is finished, removes c.o. of the table from the planning_scene_interface

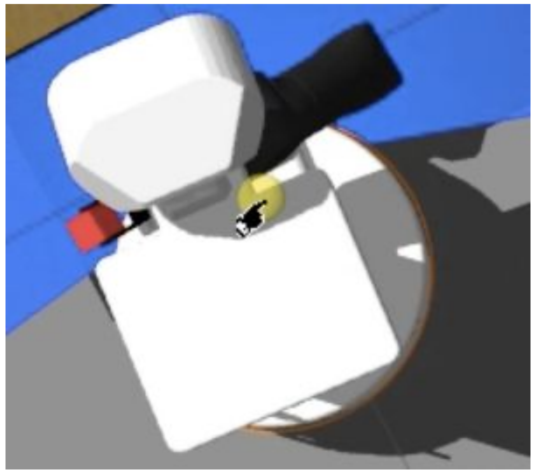


Implemented in `node_place`

PICK PHASE


Object ID 1


Object ID 2


Object ID 3

After obtaining the **poses of the various object to be picked**, using the positions and orientations with respect to the reference frame `”map”` the pick phase begins…

Implemented in `node_pick`

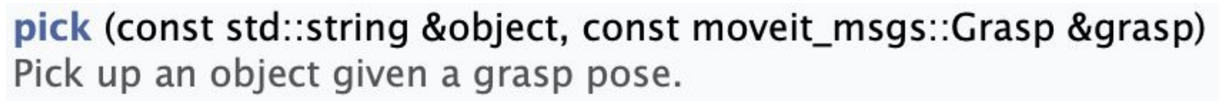
PICK PHASE

The pick phase consists of the following **subphases**:

- **Positioning** the arm so that it is easier to grip the object
- **Grasp** of the object
- **Attach** the object to the gripper and close it
- **Positioning** of the arm equal to the first point
- Placement of the arm in a **safe configuration for movement** within the environment

Two different functions were developed for picking: one for the red and green objects that takes advantage of the **pick()** function found in the MoveGroup Class and an **ad hoc** one for the blue object by making a plan for the **move_group** associated with the arm .

(pick from MoveGroup Class)



pick (const std::string &object, const moveit_msgs::Grasp &grasp)
Pick up an object given a grasp pose.

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

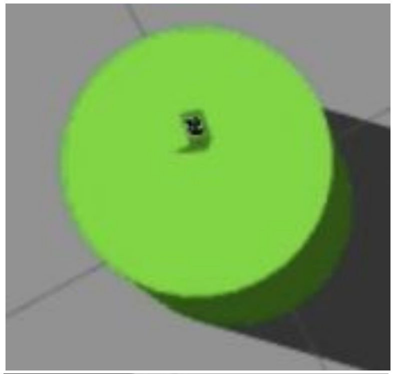
PLACE PHASE

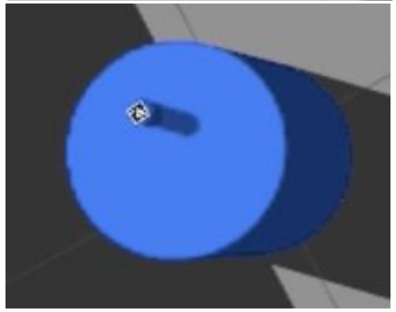
The place phase consists of the following subphases:

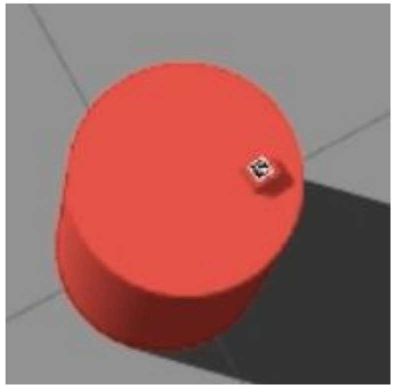
- Positioning the arm so that it is easier to place the object
- Place the object
- Open the gripper and detach the object
- Positioning of the arm equal to the first point
- Placement of the arm in a safe configuration for movement within the environment

In this case the reference frame used for the position and orientation of the object to be placed on the table is `”base_footprint”`.

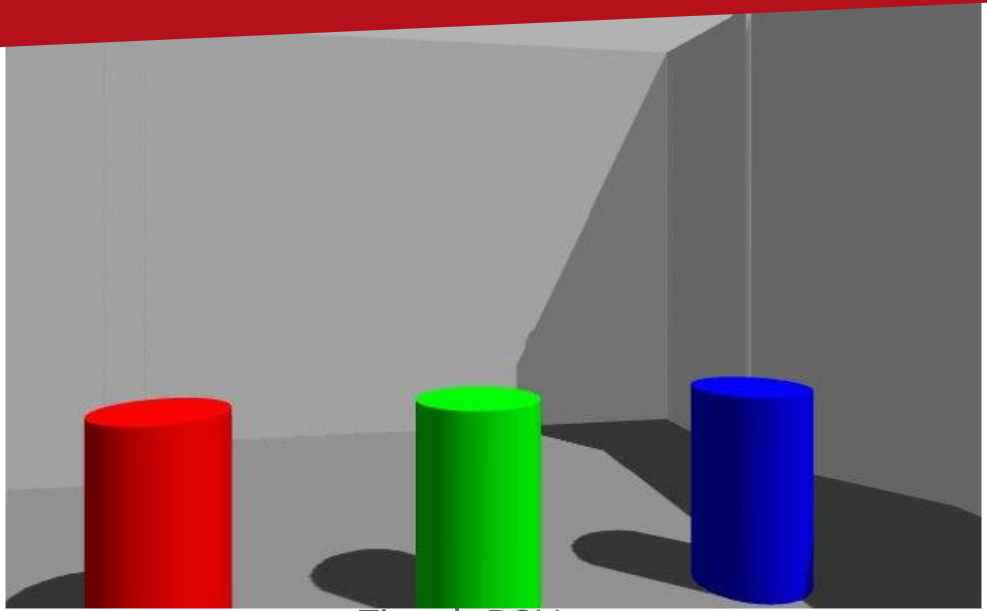
Implemented in `node_place`


Object ID 1


Object ID 2


Object ID 3

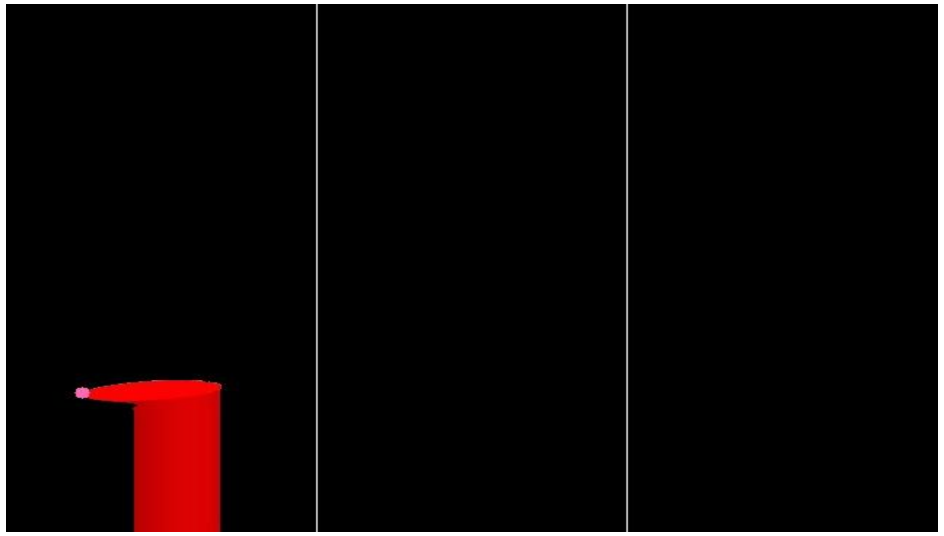
AUTOMATIC DOCKING ROUTINE


Tiago's POV

PROBLEM: Recognize which is the correct table, thus **define** the correct target position.

SOLUTION:

1. Robot moves into table room
2. Looks towards the coloured tables (Tiago's POV)
3. **Threshold** the image to **segment** the right table
4. The first non-black point defines the **target position** (in terms of left/center/right)

Segmented image


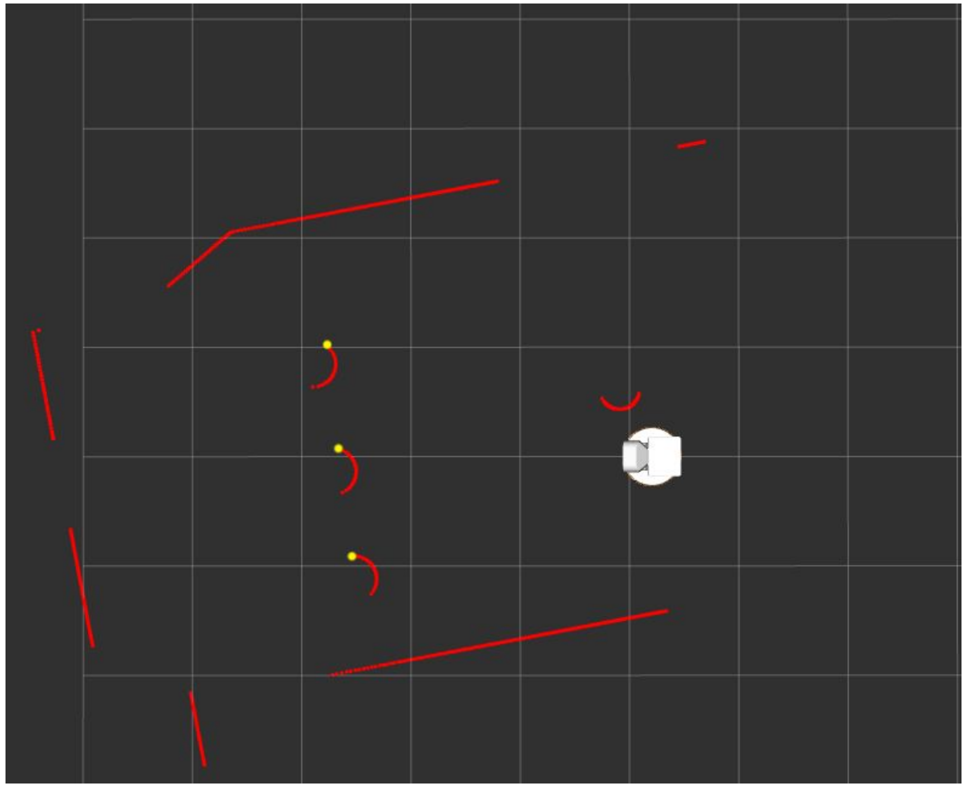
AUTOMATIC DOCKING ROUTINE

5. Using the same **object detection algorithm** as in assignment 1, we obtain the **center points** of the three tables

6. Use the result of the image segmentation to determine which of the three is the **correct table**

7. **Move the robot** to the obtained center table position minus a delta (in front of the table)

finally start the **place phase…**



Implemented in `node_a`