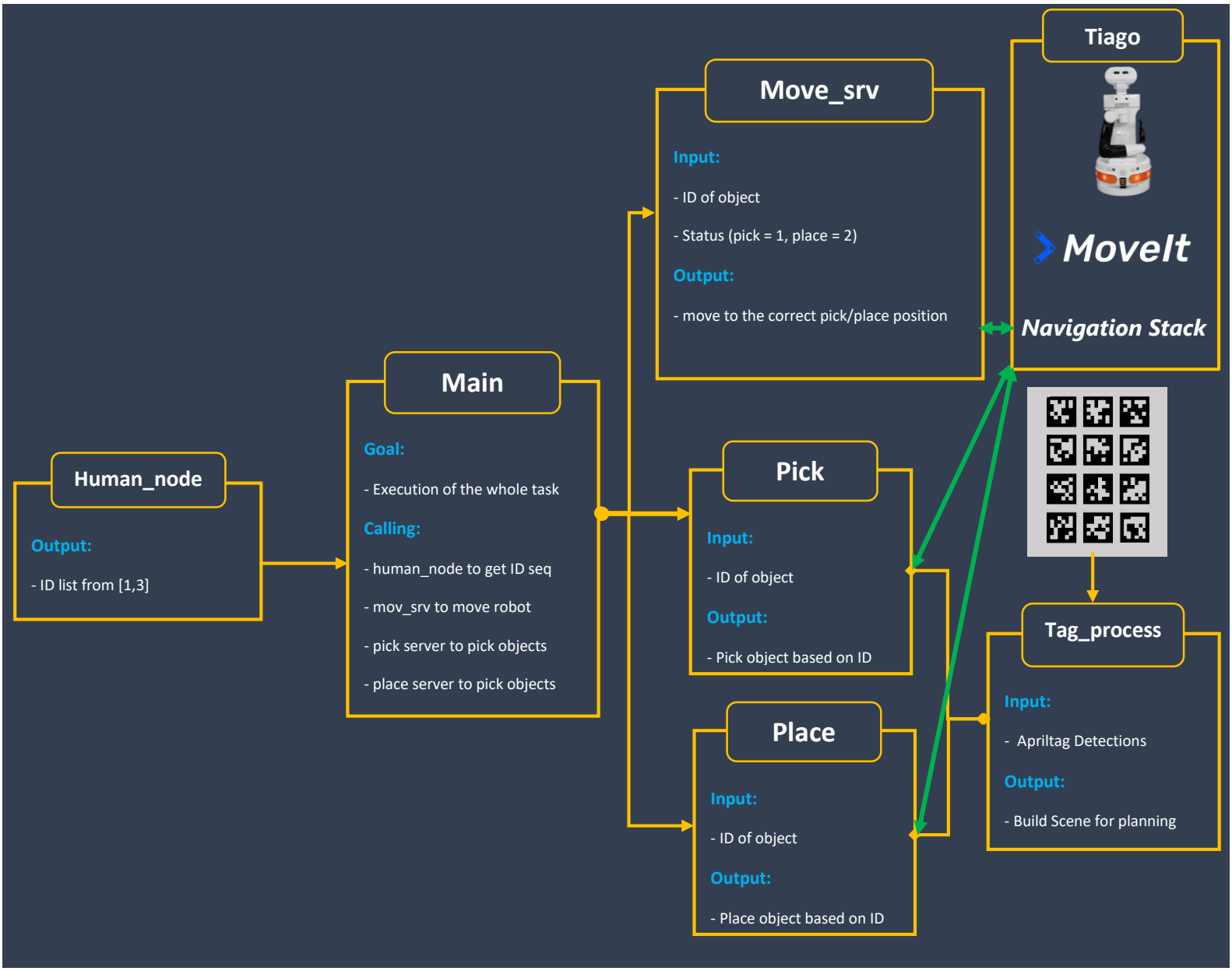
Assignment (2) Intelligent Robotics

Mohamed Abdelwahab, Victor Miguel, Onur Akman

1 Problem Definition

The problem that we are addressing is the “Pick and Place” of three target objects. A blue hexagonal prism, a green triangular prism, a red cube on a table among some yellow hexagonal obstacles each of those objects have tag that assigns an ID to it. The objective is, after receiving a sequence of ID’s from human node, detect and pick objects from a cluttered table, and bring them to a cylindrical table with the same color. To solve this, routines were implemented on ROS based on navigation, joint and pose control for a robotic arm, and object detection based on tags.

2 Project Architecture



3 Code Implementation

Our implementation is based on getting the list of IDs then iterate over them to pick and place the object associated to that ID. Our coded is divided into **5 nodes**, we choose to separate pick and place to follow ROS convention that each node has only one main functionality. Additionally, we define the global variables that needed tuning as **parameter server** for easy access and modification.

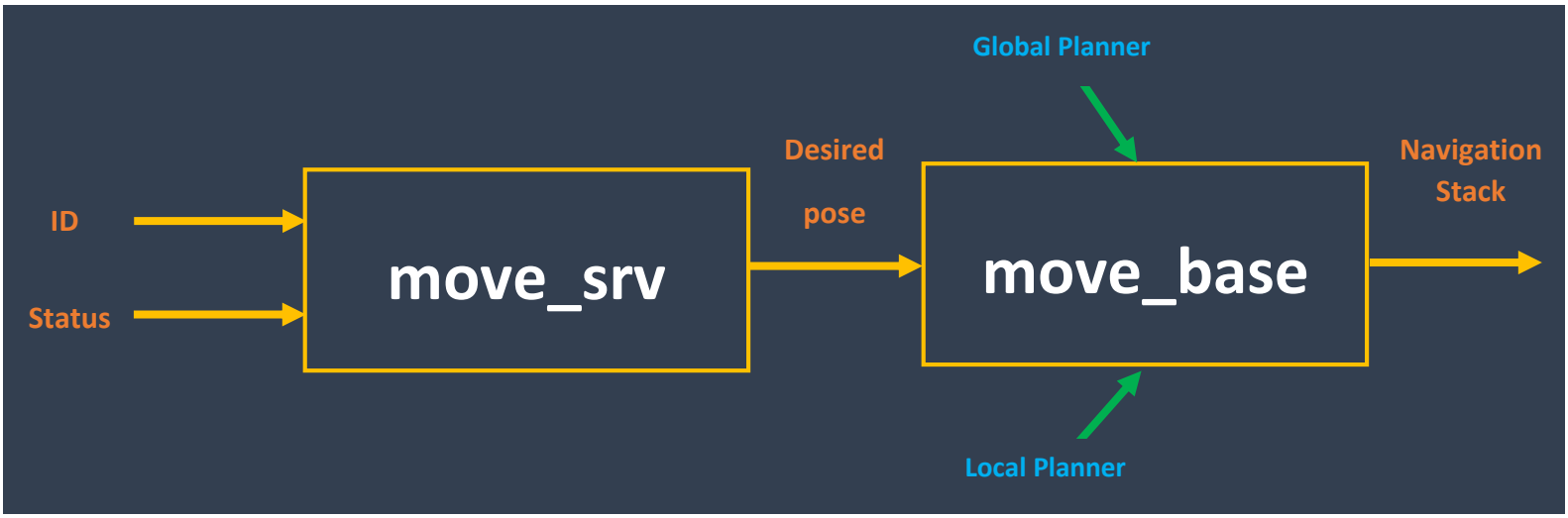
3.1 main Node

This node calls all action servers that we created and coordinates the execution of the various steps:

- Get human input (IDs list)
- Move to table to pick object
- Build the planning scene for objects, table and obstacles
- Move to placing position
- Build the planning scene for placing cylinders
- Place the object to the center of the cylinder
- Do the same for next object

3.2 Navigation Task

Following client-server approach, we first got all global picking and placing poses that simplify the picking task for the robot from parameter server. Then, the client will only need to receive (ID, status = pick/place) to activate navigation to the correct place.

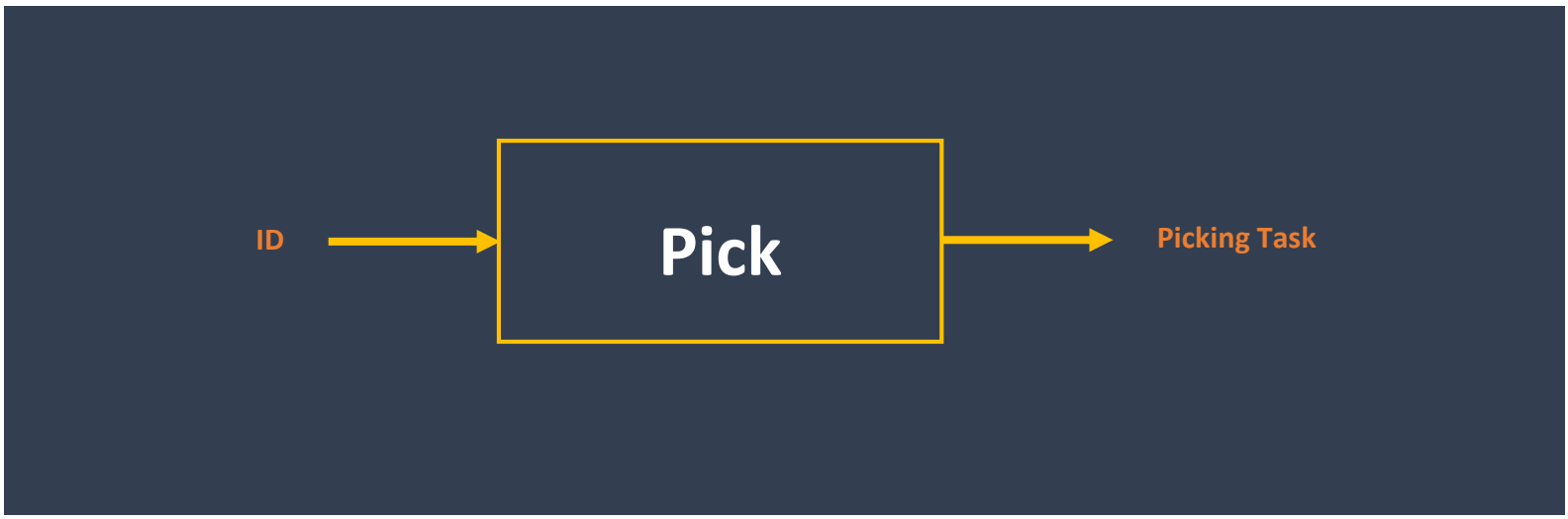


3.3 Object Detection & Planning Scene Building

This node is activated once the robot reaches to the picking/placing position for the object in the list given by the human node, before performing the picking action, detection procedure is followed. This procedure is done with the help of markers, AprilTags attached to the surfaces of the objects. Once the main node publishes that the scene should be prepared for the picking operation, the node that is responsible for the scene starts detection of the tags and adding collision objects for the related shapes to the scene with the help of PlanningSceneInterface. The geometry of objects and obstacles were hidden inside the node, since they are known for us from URDF and Gazebo. We used two topics to start and terminate scene generation, our motivation was not to continuously add the same object to the scene and to avoid changes in poses values and frames from Apriltags.

3.4 Pick Task

This server node is responsible picking object routine from the table.



Given object's ID, the following steps are computed inside the server:

- Lift the robot torso to reach higher position where is easier to pick the objects and see the tags.
- Activate tags process node to construct the scene and then stop.
- Move the arm to a safe pose from which it will start the picking task.
- Retrieve the object from the planning scene.
- Move the arm above the object center position to prepare for picking
- Open the gripper.
- Approach the object through a linear movement.
- Remove the object from the scene to be able to grasp it.
- Attached the object with the Gazebo plugin to avoid dropping.
- Close the gripper.
- Through a linear movement we depart from the target position of few cm.
- Move away from the object through a linear movement.
- Add the object back to the scene and attached to the arm in MoveIt
- Move the arm to the safe position for moving in the room.

3.5 Place Task

This server node is responsible placing object routine on the target cylinders.

Given object's ID, the following steps are computed inside the server:

- Lift the robot torso to reach higher position where is easier to place the objects.
- Get the table and object pose and geometry from the scene, then compute the final pose.
- Move the arm to a safe pose to begin the placing task.
- Move the arm above the target position (center of the cylinder).
- Approach the object through a linear movement.

- Open the gripper.
- Detached the object from both the gripper and remove it from the scene.
- Move away from the object through a linear movement.
- Move the arm to the safe position for moving in the room.

4 Problems

4.1 Navigation

- For certain goals, the generated path would make the robot get into poses where it might get stuck, due to the design of the map. This particular problem was avoided by a well-known and simple solution: by defining a safe position to pass by before proceeding to the final position.

4.2 Pick & Place

- Sometimes Apriltag values change. For this case, we only subscribe only once and build the scene then we deactivate subscribing.
- Sometimes object fall from the placing table after correctly placing it.
- It was difficult to exactly determine the center pose of the pyramid.
- Since the inverse kinematics problem is solved numerically in MoveIt, sometimes it can't find solution for the same problem and same initial conditions.

5 Group Contribution

- Mohamed and Onur implemented the tags_process node and move_srv node.
- Mohamed and Victor implemented the pick node and place node.

6 Conclusion

A remarkable aspect is that by experimentation process, all the hyper-parameters of both the detection, and motion algorithms were tested, and selected. By considering the possible stuck situations, and avoiding them, achieving desired poses with the navigation stack becomes a trivial problem. While the arm motion planning task was challenging both in collision scene building and in picking the object with the correct pose after defining some intermediate points, we managed to solve the pick and place tasks successfully.