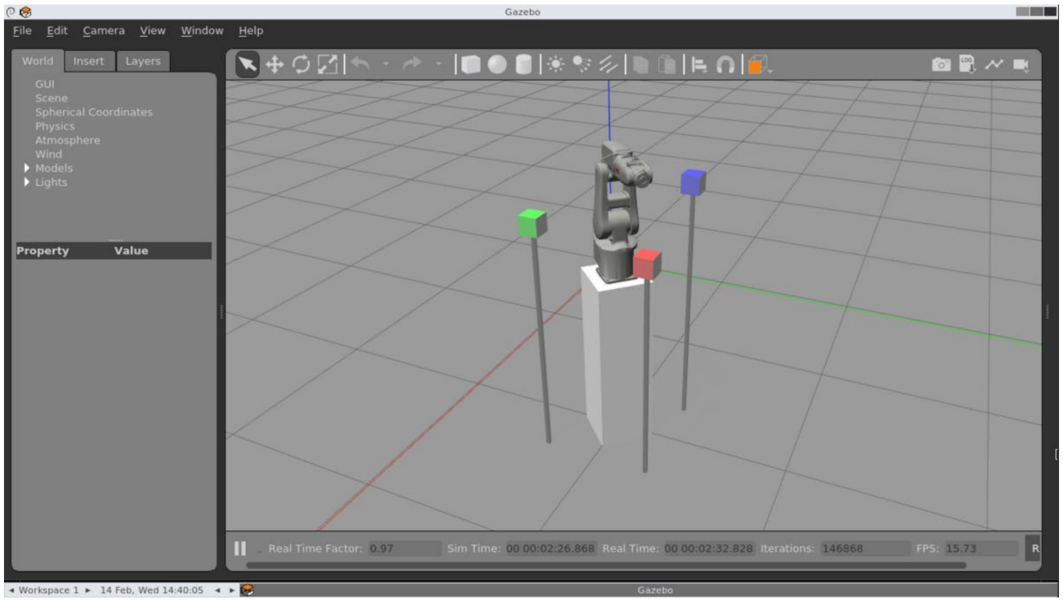
ARC 380 / CEE 380 – Introduction to Robotics for Digital Fabrication
Mondays & Wednesdays 11:00am-12:30pm | Arch Bldg N107 / ECL
Princeton University | School of Architecture | Spring 2024

Professor: Arash Adel | Assistant-in-Instruction: Daniel Ruan



Assignment 3: Introduction to ROS

Due: Tuesday, February 20, 11:59pm

Instructions:

Clone the course repository from https://github.com/ADRLaboratory/arc380_s24.git and work in the assignments/Assignment3/ARC380_Assignment3.py Python file.

Part 1 – Quaternions (3 pts total)

In the first part of the assignment, you will be completing functions for working with quaternions. Basic tests have been provided at the end of the file, and this is the only portion of the code that will be Autograded.

a. Complete the function 'quaternion_to_rotation_matrix' that converts a quaternion (as a NumPy array, in the format [x, y, z, w], where w is the real part) into a 3x3 rotation matrix. **(1 pt)**
    - Hint: You can easily search for formulas on how to do this online. Just make sure you note the convention they are using (e.g., [x, y, z, w] vs [w, x, y, z] or [q1, q2, q3, q4], where q1 is the real part).

b. Complete the function 'rotation_matrix_to_quaternion' that converts a 3x3 rotation matrix into a quaternion (as a NumPy array, in the format [x, y, z, w]). **(2 pts)**
    - Hint: There are also references for this online. One example, which has example code in Java and C++: https://www.euclideanspace.com/maths/geometry/rotations/conversions/matrixToQuaternion/

Part 2 – Manipulating the Robot in Gazebo (7 pts total)

In the second part of the assignment, you will build up towards manipulating an ABB IRB 120 arm in Gazebo to knock down three colored blocks.

Open two docker terminals (`docker exec -it arc380-ros bash`) and launch Gazebo in one terminal first, then MoveIt in the other, both from the arc380_assignment3 package. The Gazebo application will open the world with the colored blocks loaded, and MoveIt will launch without the Rviz GUI.

```
roslaunch arc380_assignment3 gazebo.launch
```

```
roslaunch arc380_assignment3 moveit.launch
```

a. Complete the function 'get_block_positions' which subscribes to the '/gazebo/model_states' topic and returns the position (x, y, z) of each block as a list of NumPy arrays. The subscriber code has been provided for you, to demonstrate how you can retrieve a single message from a topic. The block names are 'block_red', 'block_green', and 'block_blue'. **(1 pts)**

   - Hint: Examine the documentation for the gazebo_msgs/ModelStates message: http://docs.ros.org/en/jade/api/gazebo_msgs/html/msg/ModelStates.html
   - The 'name' key stores a list of the Gazebo model names, and their poses are stored in 'pose' in the same order.

b. Complete the function 'get_robot_pose', which subscribes to the '/gazebo/model_states' topic and returns the position (x, y, z) and orientation (as a quaternion, in the format [x, y, z, w]) of the robot root. The name of the robot model is 'abb_irb120_3_60'. **(2 pts)**

   - Hint: You can copy and modify the subscriber code from the previous part.

c. The block positions that you retrieved from Part 2.a. are located in the Gazebo world frame, so they need to be transformed into the robot frame. Complete the function 'coordinate_xform_world_to_robot_frame' which retrieves the robot position and orientation from Part 2.b. and returns a coordinate transformed point. **(1 pt)**

d. Now that you have the block positions in the frame of the robot, you need to generate a path trajectory. Examine the code for the function 'move_robot_to_waypoints' (you don't need to complete any code here). This function takes in a list of waypoints, calls the '/compute_cartesian_path' service, then executes that trajectory using the action client '/joint_trajectory_action'. **(0 pts)**

e. Complete the function 'form_cartesian_path_msg', which is called by 'move_robot_to_waypoints'. Most of the message has been provided, and you need to complete msgs['waypoints']. **(2 pts)**

   - Hint 1: Read the documentation on 'moveit_msgs/GetCartesianPath' here: http://docs.ros.org/en/noetic/api/moveit_msgs/html/srv/GetCartesianPath.html
   - Hint 2: You will need to provide orientation values here. You will likely need to try different orientations to reach every point.

f. Complete the function 'knock_over_blocks', which ties everything in this part together. Uncomment the function at the bottom of the file to execute it in Gazebo, and record a successful attempt at knocking every block to the ground. Use the 'reset_gazebo_world' function to reset after each attempt. **(1 pt)**

   - Hint: You may need to add additional waypoints to complete your trajectories successfully. The current motion planning isn't very intelligent, and only moves the robot in straight paths.

Submission guidelines: Upload your completed Python file and simulation recording to Gradescope. Do not change the name of the Python file (keep it as "ARC380_Assignment3.py"). You have unlimited submission attempts up until the due date, however, it is recommended that you test your code first with the provided tests before submitting. The Autograder will utilize the same tests plus a few hidden ones to grade Part 1.

You will receive full points for Part 2 if all the blocks are knocked down successfully. Partial credit will be graded on code correctness for each function as outlined above.