CS 6341 Robotics Homework 4

Professor Yu Xiang

November 6, 2025

In this homework, write down your solutions for problems 1 and finish the coding problem 2. Upload your solutions and code to eLearning. Our TA will check your solutions and run your scripts to verify them.

Problem 1

(4 points)

Rigid-body dynamics in an arbitrary frame. Exercise 8.3 in Lynch and Park, Modern Robotics.

Steiner's theorem says that, the inertia matrix $\mathcal{I}_q$ about a frame aligned with $\{b\}$, but at a point $q = (q_x, q_y, q_z)$ in $\{b\}$, is related to the inertia matrix $\mathcal{I}_b$ calculated at the center of mass by

$$\mathcal{I}_q = \mathcal{I}_b + \mathbf{m}(q^{\mathrm{T}}qI - qq^{\mathrm{T}}), \tag{1.1}$$

where I is the 3×3 identity matrix and $\mathbf{m}$ is the mass of the body.

(1.1) Show that the following equation is a generalization of Steiner's theorem.

$$\mathcal{G}_a = [\mathrm{Ad}_{T_{ba}}]^{\mathrm{T}}\mathcal{G}_b[\mathrm{Ad}_{T_{ba}}]. \tag{1.2}$$

(1.2) Given the dynamic equation for a single rigid body in the body frame

$$\mathcal{F}_b = \mathcal{G}_b\dot{\mathcal{V}}_b - [\mathrm{Ad}_{\mathcal{V}_b}]^{\mathrm{T}}\mathcal{G}_b\mathcal{V}_b \tag{1.3}$$

and Eq. (1.2), prove that

$$\mathcal{F}_a = \mathcal{G}_a\dot{\mathcal{V}}_a - [\mathrm{Ad}_{\mathcal{V}_a}]^{\mathrm{T}}\mathcal{G}_a\mathcal{V}_a. \tag{1.4}$$

Problem 2

(6 points)

ROS programming, grasping of a block.

In this problem, you will use moveit to enable a robot to grasp a cube in ROS2. **You can directly use Ubuntu, or Docker or virtual machine to install ROS2 according to your own set up**. Refer to the ROS2 Jazzy page if needed:

`https://docs.ros.org/en/jazzy/`.

For the following steps, start with your ROS2 workspace from the previous homework.

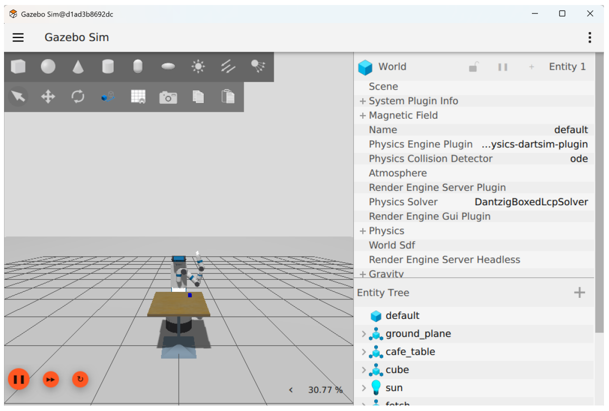
(2.1) I made some changes to the following repositories. **Please update your code and then rebuild your workspace.** You can either git clone the new code, or do a git pull for each repo.

- Git clone the source code to the src folder of your ROS workspace (use the master branch): git clone `https://github.com/IRVLUTD/panda_gz_moveit2.git`
- Git clone the source code to the src folder of your ROS workspace (use the main branch): git clone `https://github.com/IRVLUTD/pymoveit2`

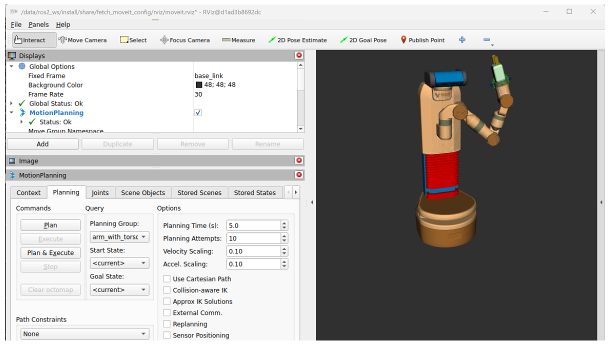
(2.2) Launch Fetch Gazebo Simulator by following the steps in Homework 2. Use terminator to start multiple windows. Make sure that you do not see any error from the terminal; sometimes some controllers may not start correctly, so you can restart it.

- `ros2 launch panda fetch.launch.py`

You shall see the Gazebo environment as in Figure 1.



Gazebo sim interface



Rviz interface

Figure 1: Two windows after launching the Fetch Gazebo simulation

(2.3) Up until now, you shall have Gazebo and Moveit running. **To verify your setup is correct, you can use the motion planning panel in Rviz to try some planning tasks.** In this coding assignment, you need to use the pose of the demo cube computed from Homework 2 and moveit for inverse kinematics from Homework 3 to grasp the demo cube.

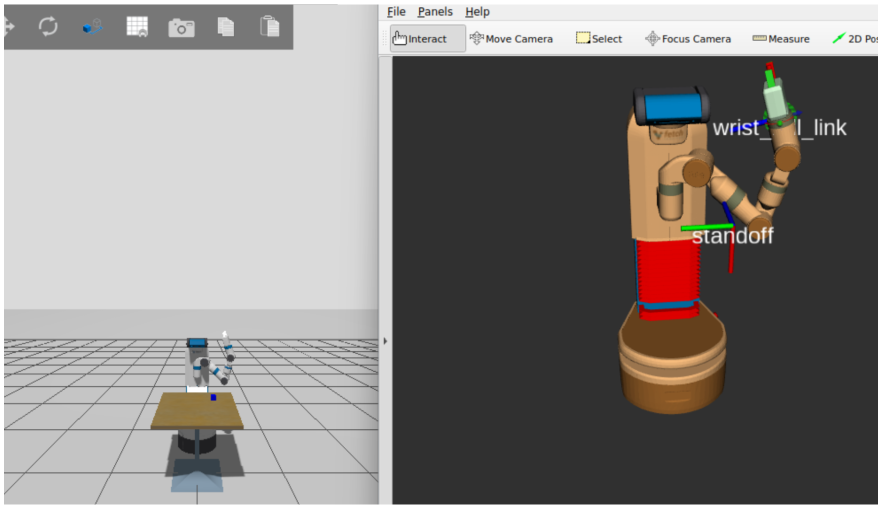


Figure 2: Visualization of the standoff pose for grasping the cube.

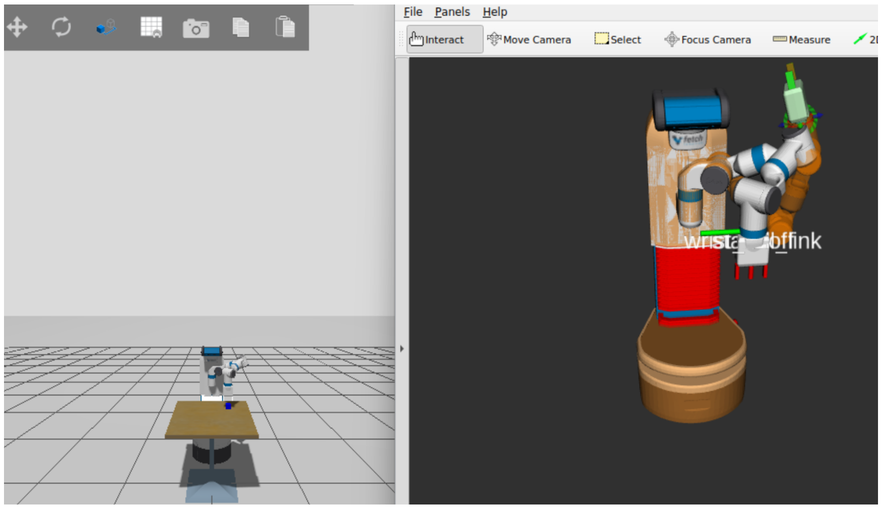


Figure 3: Visualization of the robot reaching the standoff pose.

Here is the outline of the steps:

- Step 1: Figure out the end-effector pose for the standoff pose for grasping the cube, which is the end-effector pose before approaching the cube. In our case, the end-effector link is the wrist roll link as illustrated in Figure 2.
- Step 2: Compute inverse kinematics to obtain the joint position to reach the standoff pose.
- Step 3: Use moveit for planning and control the robot to reach the standoff pose, as illustrated in Figure 3.
- Step 4: Compute the grasping pose from the standoff pose, and then use moveit to plan to the grasping pose, as illustrated in Figure 4.
- Step 5: Close the fingers and the move to the original joint position, as illustrated in Figure 5.

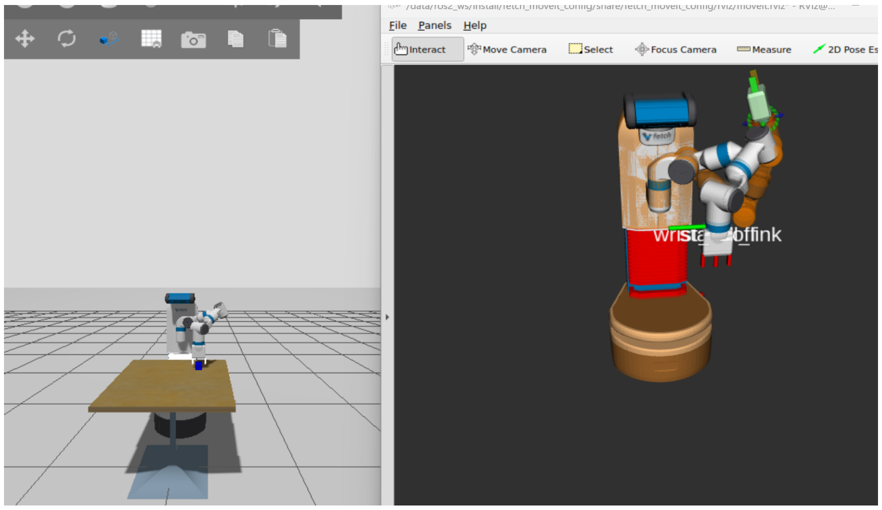


Figure 4: Visualization of the robot reaching the grasping pose.

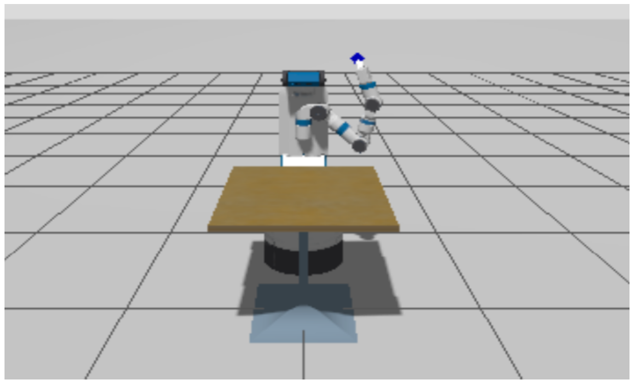


Figure 5: Visualization of the robot grasping the cube.

Download the grasp_block.py from eLearning. Finish the implementation of the TODOs in the python script. Then you can run the python script to verify your implementation.

Submission guideline: Upload your implemented grasp_block.py file and the screen capture of Gazebo after running the script to eLearning.