CS 6341 Robotics Homework 3

Professor Yu Xiang

November 6, 2025

In this homework, write down your solutions for problems 1, 2 and finish the coding problem 3. Upload your solutions and code to eLearning. Our TA will check your solutions and run your scripts to verify them.

Problem 1

(3 points)

Forward Kinematics. Exercise 4.12 in Lynch and Park, Modern Robotics.

The RRPRRR spatial open chain of Figure 1 is shown in its zero position (all joints lie on the same plane). Determine the end-effector zero position configuration M, the screw axes $\mathcal{S}_i$ in {0}, and the screw axes $\mathcal{B}_i$ in {b}. Setting $\theta_5 = \pi$ and all other joint variables to zero, find T_{06} and T_{60}.

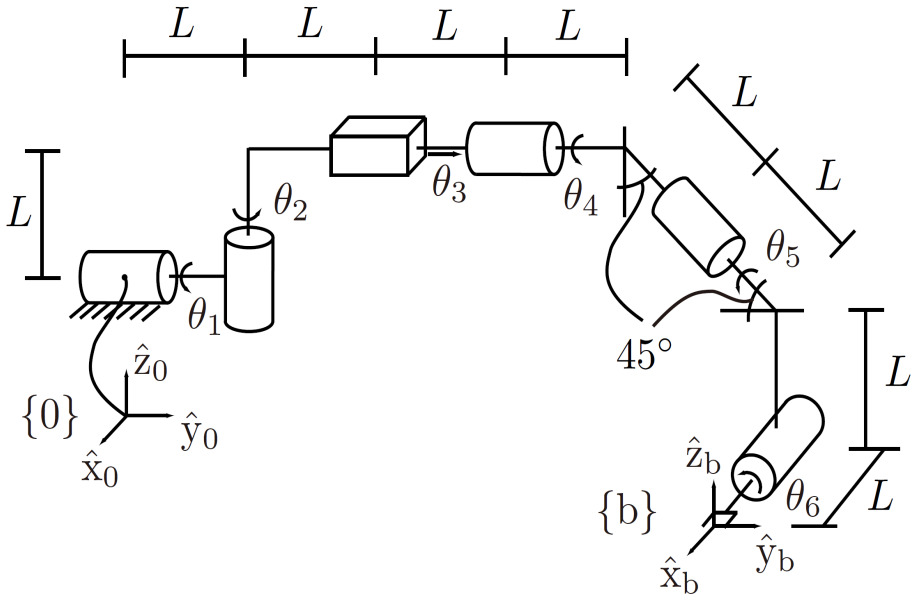


Figure 1: An RRPRRR spatial open chain.

Problem 2

(3 points)

Velocity Kinematics. Exercise 5.11(a) in Lynch and Park, Modern Robotics.

The spatial 3R open chain of Figure 2 is shown in its zero position. Let p be the coordinates of the origin of {b} expressed in {s}. In its zero position, suppose we wish to make the end-effector move with linear velocity $\dot{p} = (10, 0, 0)$. What are the required input joint velocities $\dot{\theta}_1$, $\dot{\theta}_2$ and $\dot{\theta}_3$?

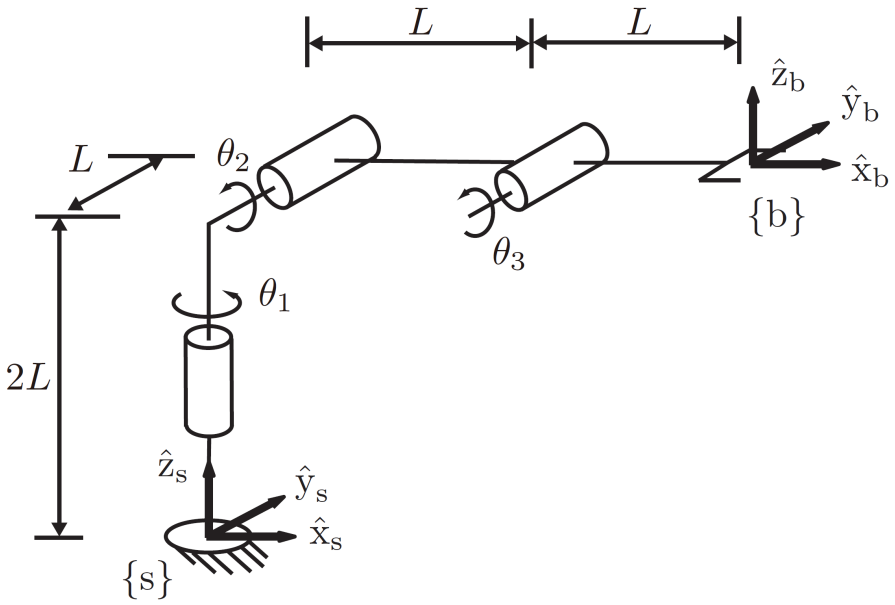


Figure 2: A spatial 3R open chain.

Problem 3

(4 points)

ROS2 programming, Moveit2 planning scene, forward kinematics and inverse kinematics.

In this problem, you will learn forward kinematics and inverse kinematics 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:

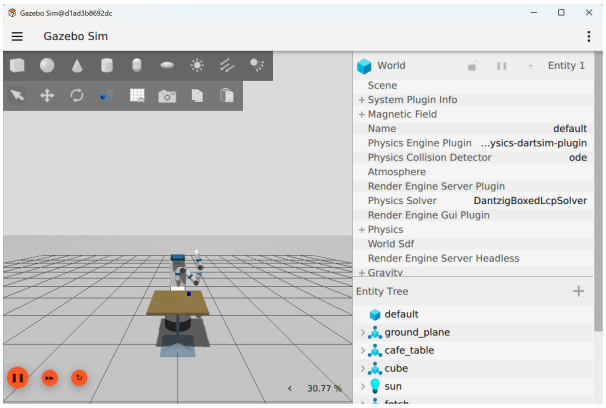
`https://docs.ros.org/en/jazzy/`.

For the following steps, start with your ROS2 workspace from Homework 2.

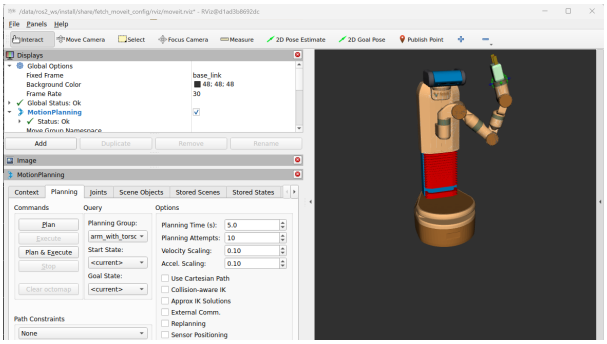
(3.1) Install and launch Fetch Gazebo Simulator by following the steps in Homework 2. Use terminator to start multiple windows.

- `ros2 launch panda fetch.launch.py`

You shall see the Gazebo environment as in Figure 3.



Gazebo sim interface



Rviz interface

Figure 3: Two windows after launching the Fetch Gazebo simulation

(3.2) Up to now, you shall have the Gazebo and Moveit running. In order to use python for motion planning, we need another package named pymoveit2:

- Git clone the source code to the src folder of your ROS workspace (use the main branch): git clone `https://github.com/IRVLUTD/pymoveit2`
- Build your ROS workspace again to include this package. Source your workspace after building it.

  Make sure that the worksapce is correctly built without seeing any error from the terminal.

(3.3) In this is coding assignment, you need to use the pose of the demo cube computed from Homework 2 to set up a planning scene in Moveit, and use pymoveit2 for forward kinematics and inverse kinematics.

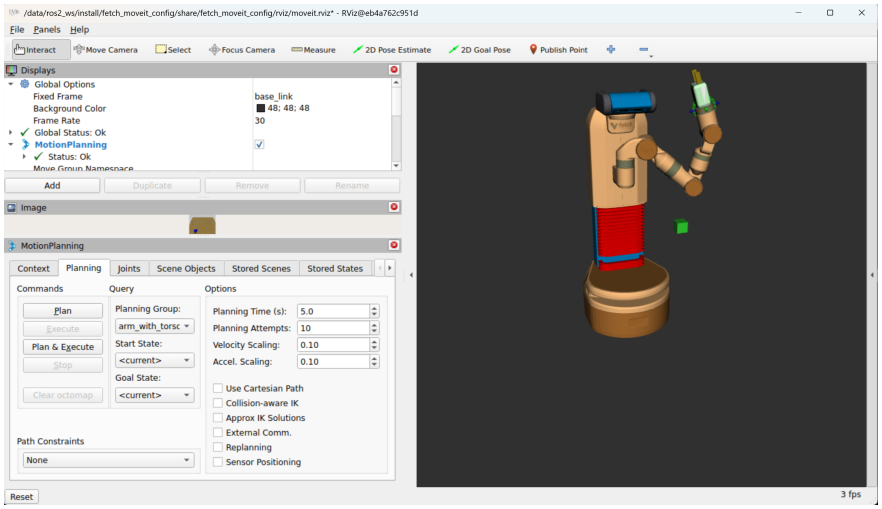


Figure 4: Rviz visualization of the planning scene.

```
Forward kinematics succeeded. Result: geometry_msgs.msg.PoseStamped(header=std_msgs.msg.Header(stamp=builtin_interfaces.msg.Time(sec=0, nanosec=0), frame_id='base_link'), pose=geometry_msgs.msg.Pose(position=geometry_msgs.msg.Point(x=0.13971315719934574, y=0.41790482951941804, z=1.381362123065168), orientation=geometry_msgs.msg.Quaternion(x=-0.4933132692357978, y=0.6356270110471862, z=-0.3640800945555554, w=-0.4691119330752887)))
---------------------------
wrist_roll_link in base_link: pos=(0.1397, 0.4179, 1.3814), quat=(-0.4933, 0.6356, -0.3641, -0.4691)
---------------------------
Inverse kinematics succeeded. Result: sensor_msgs.msg.JointState(header=std_msgs.msg.Header(stamp=builtin_interfaces.msg.Time(sec=0, nanosec=0), frame_id='world'), name=['l_wheel_joint', 'r_wheel_joint', 'torso_lift_joint', 'bellows_joint', 'head_pan_joint', 'head_tilt_joint', 'shoulder_pan_joint', 'shoulder_lift_joint', 'upperarm_roll_joint', 'elbow_flex_joint', 'forearm_roll_joint', 'wrist_flex_joint', 'wrist_roll_joint', 'l_gripper_finger_joint', 'r_gripper_finger_joint'], position=[-0.06410934876935521, -0.0588302010799024 6, 0.37998618324873973, 0.38000004998231085, 5.062122897280595e-10, 0.9082704004523408, 1.3199999996581728, 0.7000001379375981, 1.0533048814707107e-09, -2.0000000159871134, -7.055169842471455e-09, -0.5700001554737 66, 1.9394982202304754e-09, 0.05, 0.05], velocity=[-1.7641708788262989e-09, -1.6364876259666872e-09, -2.499216111839786e-14, 9.884792250946638e-12, 2.4352197233561135e-21, 7.892310199472674e-11, -1.0547129581254668e-13, 4.7200443423559e-14, -1.5515526011330646e-18, -1.8296979261019607e-13, 3.931079908207208e-18, -5.2354550816174905e-14, -2.1645926967083645e-18, -3.2322237283720226e-16, 1.6156931122727125e-16], effort=[])
---------------------------
torso_lift_joint 0.3799862
shoulder_pan_joint 1.32
shoulder_lift_joint 0.70000017
upperarm_roll_joint 1.0533049e-09
elbow_flex_joint -2.0
forearm_roll_joint -7.05517e-09
wrist_flex_joint -0.5700002
wrist_roll_joint 1.9394981e-09
joint distance: 0.0
```

Figure 5: Output from the FK and IK solver in pymoveit2.

Download the planning_scene_block.py file from eLearning. This python script first queries the pose of the cube in the Gazebo environment as in Homework 2. Then it creates a planning scene using Moveit and adds the cube block into the planning scene. Finally, it computes forward kinematics and inverse kinematics of the robot using pymoveit2.

Finish the implementation of the TODOs in the python script. Then you can run the python script. We need to see a green block in the planning scene as in Figure 4. Figure 5 shows the output from the script. You need to see the computed IK solution is close to the current joint coordinates of the robot.

Submission guideline: Upload your implemented planning_scene_block.py file and the screen capture of the planning scene similar to Figure 4 to eLearning.