CS 6301 Introduction to Robot Manipulation and Navigation Homework 2

Professor Yu Xiang

September 21, 2022

In this homework, write down your solutions for problems 1, 2, 3 and finish the coding problem 4. Upload your solutions and code to eLearning. Our TA will check your solutions and run your scripts to verify them.

Problem 1

(2 points)

Exponential Coordinates of Rotations. Exercise 3.5 in Lynch and Park, Modern Robotics.

Find the exponential coordinates $\hat{\omega}\theta \in \mathbb{R}^3$ for the $SO(3)$ matrix

$$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}.$$

Problem 2

(2 points)

Homogeneous Transformation Matrices. Exercise 3.18 in Lynch and Park, Modern Robotics.

Consider a robot arm mounted on a spacecraft as shown in Figure 1, in which frames are attached to the Earth {e}, a satellite {s}, the spacecraft {a}, and the robot arm {r}, respectively.

(2.1) Given T_{ea}, T_{ar}, and T_{es}, find T_{rs}.

(2.2) Suppose that the frame {s} origin as seen from {e} is $(1, 1, 1)$ and that

$$T_{er} = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Write down the coordinates of the frame {s} origin as seen from frame {r}.

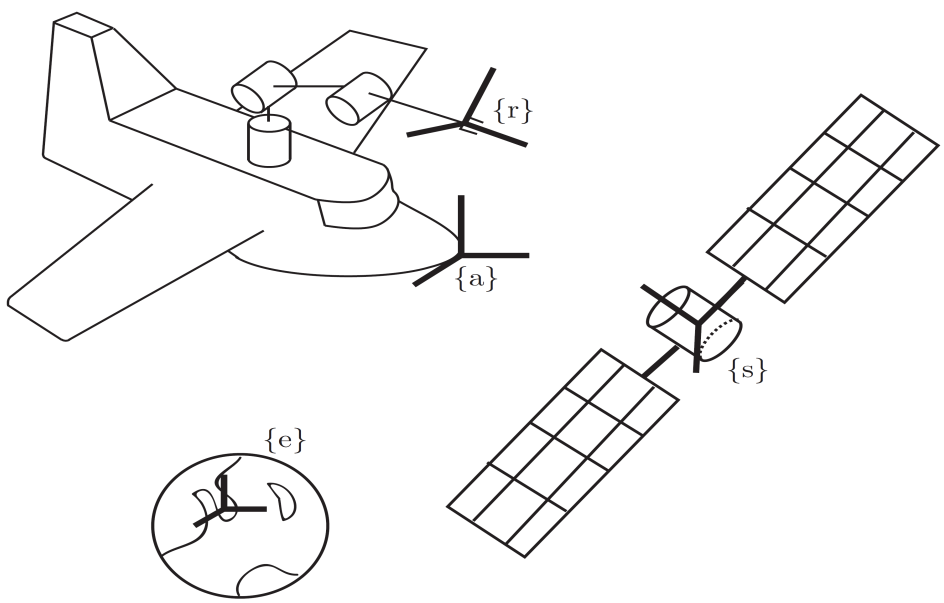

Figure 1: A robot arm mounted on a spacecraft.

Problem 3

(2 points)

Twists and Screw Axes. Exercise 3.27 in Lynch and Park, Modern Robotics.

Draw the screw axis for the twist $\mathcal{V} = (0, 2, 2, 4, 0, 0)$.

(Hint) Convert twist $\mathcal{V}$ to screw axis $\{q, \hat{s}, h\}$ and $\dot{\theta}$, and then draw it.

Problem 4

(4 points)

ROS programming and Transformations.

In this problem, you will learn the homogeneous transformations in ROS. **You can directly use Ubuntu, or Docker or virtual machine to install ROS according to your own set up**. Refer to the ROS wiki page if needed `http://wiki.ros.org/`.

(4.1) Mounting a host folder into Docker if you use Docker. For example, the following command will mount a folder in Windows "C:\data" as a folder "/data" in Docker:

- docker run -it -v C:\data:/data ubuntu:ros

In this way, you can save all your code in the host machine and use them in the Docker environment.

(4.2) Creating a ROS workspace. You can also reuse your workspace from the previous homework. A ROS workspace is a place to store your own ROS packages. Following the link here to create a ROS workspace `http://wiki.ros.org/catkin/Tutorials/create_a_workspace`. You should create the ROS workspace in the mounted folder from the host machine.

(4.3) Install and launch Fetch Gazebo Simulator. We need to install fetch_gazebo from the github source due to version issues. Follow the steps:

- Git clone the source code to the src folder of your ROS workspace:
  git clone –branch gazebo11 `https://github.com/fetchrobotics/fetch_gazebo.git`
  We need to use the gazebo11 branch for ROS noetic.
- Build your ROS workspace by catkin_make. You may need to install the following two packages if you see missing package erros: *ros-noetic-robot-controllers* and *ros-noetic-rgbd-launch*. Use apt install.
- Start terminator with multiple windows. For Docker users, you need to start X server as we did for Rviz in Homework 1.
- Use one terminator window. Verify your Fetch Gazebo installation by
  roslaunch fetch_gazebo simple_grasp.launch according to `http://docs.fetchrobotics.com/gazebo.html`.

  I encountered an error on 'python command is not found' when I ran the command in docker. In this case, create a symbol link for python under /usr/bin. If the command is correct launched, you will see the Gazebo environments as in Figure 2.

(4.4) Visualize information with Rviz. Use another terminator window to start Rviz with command: *rosrun rviz rviz*. We need to keep the Gazebo running. Follow the steps:

- In Global Options, change Fixed Frame from map to base_link.
- Click the Add button to add a Robot Model.
- Click the Add button to add an Image. Change the image topic to "/head_camera/rgb/image_-raw".

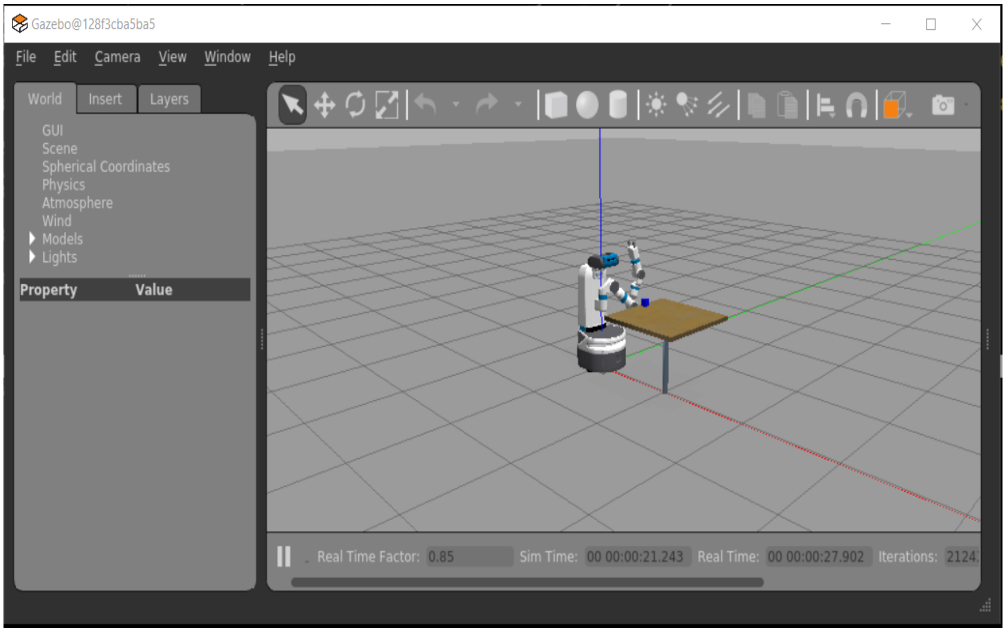

Figure 2: A Fetch Gazebo Interface.

- Click the Add button to add a TF. These are frames of different robot links.

After these steps, you should see a Rviz window as in Figure 3.

(4.5) Compute the pose of the demo cube in the Fetch base_link frame.

Download the visualize_block_pose.py file from eLearning. This python script first queries the pose of the cube in the Gazebo environment and then publishes the pose to a tf (`http://wiki.ros.org/tf`) for visualization in Rviz.

Finish the implementation of the get_pose_gazebo() function in the python script. Then you can run the python script to visualize the computed cube pose in Rviz. Figure 4 shows the demo cube pose in Rviz if your implementation is correct.

You might need to install the transforms3d package by *pip install transforms3d*.

Submission guideline: Upload your implemented visualize_block_pose.py file and screen captures for (4.3), (4.4) and (4.5) to eLearning.

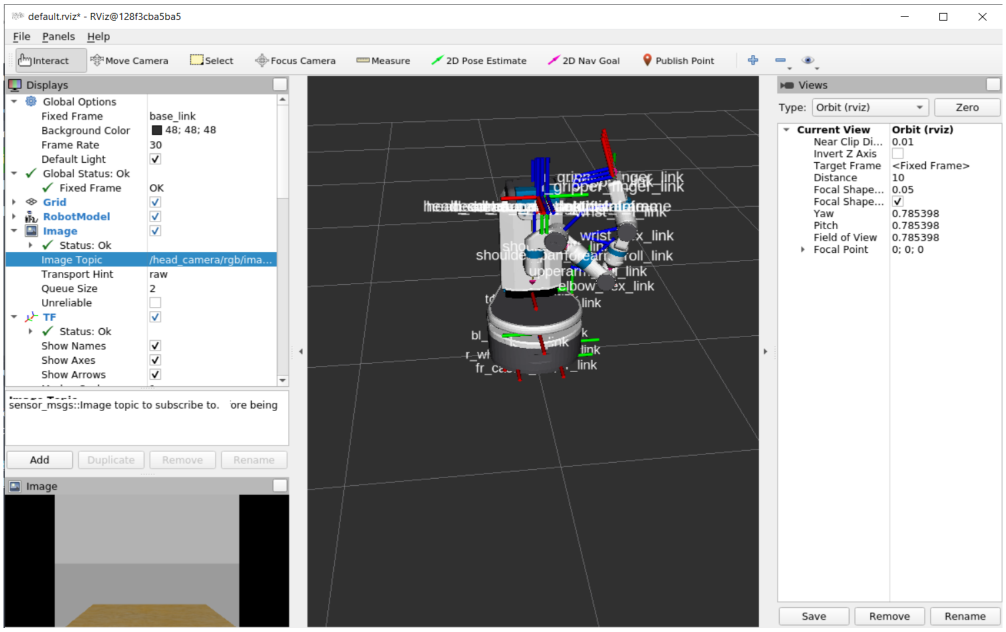

Figure 3: The Rviz Interface.

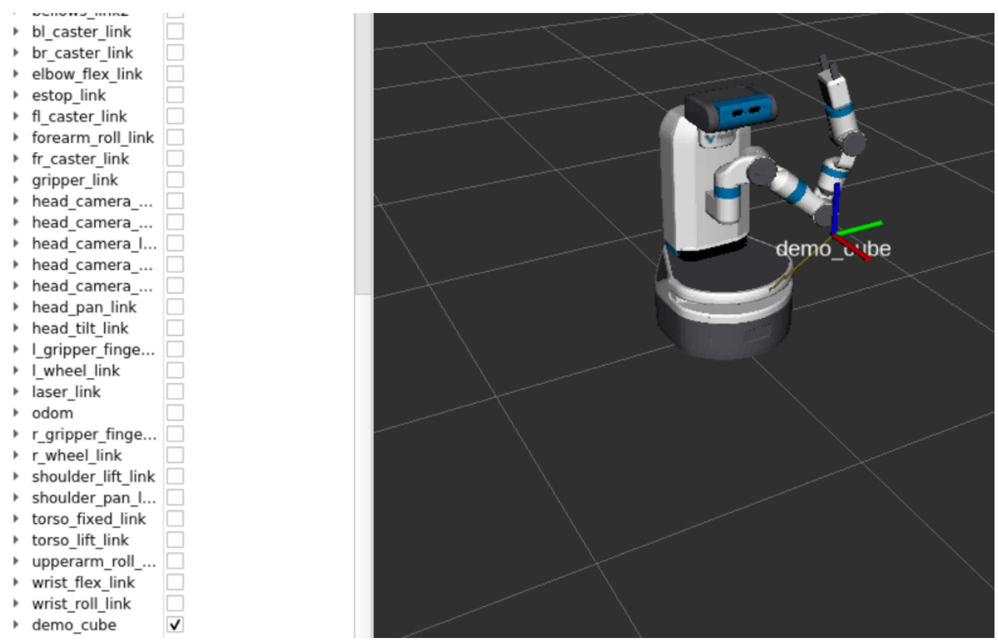

Figure 4: The cube pose in Rviz.