Homework 1

Release: 01/23/2023 Due: Mon. 02/02/2023, 11:59 PM

- You are **allowed** to consult any external resources, but you must cite them. You are also **allowed** to discuss with each other, but you need to acknowledge them. Members of the same team can submit the same copy; however, please mark down how each one contributed to each of the problem.
- Your submission includes a PDF file containing the solutions and results (including the .ipynb file export); and a zip file containing your source code.
- This homework is worth 10/100 of your final grade.
- This homework itself contains 40 points.
- It is highly recommended that you begin working on this assignment early.

Problem 1. Set up and play with SAPIEN and ManiSkill2 (16pts).

In the following homework, we will employ SAPIEN (`https://sapien.ucsd.edu`), a physical simulator, and ManiSkill2 (`https://sapien.ucsd.edu/challenges/maniskill/2022/`), a SAPIEN-based robot manipulation task suite, as our playground.

- Please follow the Google Colab tutorial here (clickable link) to install and play with SAPIEN and ManiSkill2
- Documentations of SAPIEN can be found at `https://sapien.ucsd.edu/docs/latest/index.html`
- Please answer the questions in the tutorial, export the tutorial to PDF, and combine with the solution to other problems and submit.
- Please fill in the Feedback Form at the end of the tutorial carefully. We will grade by how informative your feedback is.

Problem 2. Rotation (6pts).

Let $p := (1+i)/\sqrt{2}$ and $q := (1+j)/\sqrt{2}$ denote the unit-norm quaternions. Recall that the rotation $M(p)$ is a 90-degree rotation about the X axis, while $M(q)$ is a 90-degree rotation about the Y axis. The quaternion that lies halfway between p and q is:

$$\frac{p+q}{2} = \frac{1}{\sqrt{2}} + \frac{i}{2\sqrt{2}} + \frac{j}{2\sqrt{2}}$$

1. Calculate the norm $|(p+q)/2|$ of that quaternion, and note that it is not 1. Find a quaternion r that is a scalar multiple of $(p+q)/2$ and that has unit norm, $|r| = 1$, and calculate the rotation matrix $M(r)$. Around what axis does $M(r)$ rotate, and through what angle (say, to the nearest tenth of a degree)? (1pt)
2. What are the exponential coordinates of p and q? (1pt)

3. We use $[\omega]$ to represent a skew-symmetric matrix constructed from $\omega \in \mathbb{R}^3$:

   (a) Build the skew-symmetric matrix $[\omega_p]$ of p and $[\omega_q]$ of q, and derive their rotation matrices. (1pt)

   (b) Using what you have above to verify that $\exp([\omega_1]+[\omega_2]) = \exp([\omega_1])\exp([\omega_2])$ does *not* hold for exponential map in general (Note: the condition for this equation to hold is $[\omega_1][\omega_2] = [\omega_2][\omega_1]$). Therefore, composing rotations in skew-symmetric representation should not be done in the above way. (1pt)

4. Double-covering of quaternions:

   (a) What are the exponential coordinates of $p' = -p$ and $q' = -q$? What do you observe by comparing the exponential coordinates of $(p, -p)$ and $(q, -q)$? Does this relation hold for any quaternion pair $(r, -r)$? If it does, write down the statement and prove it. (1pt)

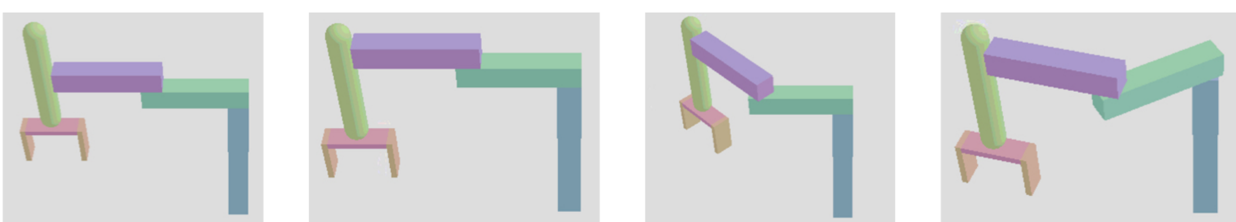
   (b) Note that the above property of quaternions is called *double-covering*. When designing a neural network to regress a quaternion output, can you use the L2 distance between the ground truth quaternion and the predicted quaternion? Why and why not? (1pt)

Problem 3. Forward Kinematics (16pts).

In this problem, we will build up a toy robot arm in SAPIEN and calculate the forward kinematics for it. The problem is based on the Google Colab notebook at here.

1. **Build up a simple robot. (5pts)**

   In the notebook, we provide an example to create a simple robot arm. As shown in Fig. 1, we use some simple primitives (i.e., box and capsule) to represent links. For simplicity, the robot arm only contains three movable joints (two revolute joints and one prismatic joint). You need to fill in the blanks (marked as 'FILL_ME_P' or 'FILL_ME_Q') in the function *create_robot* to set up the kinematics structure so that outputs of a test function match our expectation.



(a) (b) (c) (d)

Figure 1: A simple robot arm with different joint positions.

   Specifically, function *test_FK* takes the robot arm and a set of joint positions as input and will print link poses under that configuration. In the *main* function, there are 5 test cases. If you correctly build up the robot arm, the outputs for the first four test cases should match the results shown in Fig. 2; and the visualizations should also match Fig. 1.

   We don't provide the result for the last test case, and please report your output in your PDF submission. We will grade this problem based on your code and your result for the last test case.

```
Test with qpos:  [0, 0, 0] .
Link base's pose is:  Pose([0, 0, 0], [1, 0, 0, 0]) .
Link link1's pose is:  Pose([0, 0.8, 1.7], [1, 0, 0, 0]) .
Link link2's pose is:  Pose([0, 2.4, 2.1], [1, 0, 0, 0]) .
Link link3's pose is:  Pose([0, 3.6, 2.1], [1, 0, 0, 0]) .
Link end_effector's pose is:  Pose([0, 3.6, 0.85], [1, 0, 0, 0]) .
Link left_pad's pose is:  Pose([0, 3.05, 0.5], [1, 0, 0, 0]) .
Link right_pad's pose is:  Pose([0, 4.15, 0.5], [1, 0, 0, 0]) .
----------------------------------------
Test with qpos:  [0, 0, 0.7] .
Link base's pose is:  Pose([0, 0, 0], [1, 0, 0, 0]) .
Link link1's pose is:  Pose([0, 0.8, 1.7], [1, 0, 0, 0]) .
Link link2's pose is:  Pose([0, 2.4, 2.1], [1, 0, 0, 0]) .
Link link3's pose is:  Pose([8.34465e-08, 3.6, 1.4], [1, 0, 0, 0]) .
Link end_effector's pose is:  Pose([8.34465e-08, 3.6, 0.15], [1, 0, 0, 0]) .
Link left_pad's pose is:  Pose([8.34465e-08, 3.05, -0.2], [1, 0, 0, 0]) .
Link right_pad's pose is:  Pose([8.34465e-08, 4.15, -0.2], [1, 0, 0, 0]) .
----------------------------------------
Test with qpos:  [0, 0.7853981633974483, 0.7] .
Link base's pose is:  Pose([0, 0, 0], [1, 0, 0, 0]) .
Link link1's pose is:  Pose([0, 0.8, 1.7], [1, 0, 0, 0]) .
Link link2's pose is:  Pose([0.565685, 2.16569, 2.1], [0.92388, 4.56194e-08, 0, -0.382683]) .
Link link3's pose is:  Pose([1.41421, 3.01421, 1.4], [0.92388, 4.56194e-08, 0, -0.382683]) .
Link end_effector's pose is:  Pose([1.41421, 3.01421, 0.15], [0.92388, 4.56194e-08, 0, -0.382683]) .
Link left_pad's pose is:  Pose([1.0253, 2.6253, -0.2], [0.92388, 4.56194e-08, 0, -0.382683]) .
Link right_pad's pose is:  Pose([1.80312, 3.40312, -0.2], [0.92388, 4.56194e-08, 0, -0.382683]) .
----------------------------------------
Test with qpos:  [-0.5235987755982988, 0.7853981633974483, 0.7] .
Link base's pose is:  Pose([0, 0, 0], [1, 0, 0, 0]) .
Link link1's pose is:  Pose([-0.4, 0.69282, 1.7], [0.965926, -3.08536e-08, 0, 0.258819]) .
Link link2's pose is:  Pose([-0.592945, 2.15838, 2.1], [0.991445, 1.55599e-08, 0, -0.130526]) .
Link link3's pose is:  Pose([-0.282362, 3.31749, 1.4], [0.991445, 1.55599e-08, 0, -0.130526]) .
Link end_effector's pose is:  Pose([-0.282362, 3.31749, 0.15], [0.991445, 1.55599e-08, 0, -0.130526]) .
Link left_pad's pose is:  Pose([-0.424712, 2.78623, -0.2], [0.991445, 1.55599e-08, 0, -0.130526]) .
Link right_pad's pose is:  Pose([-0.140011, 3.84875, -0.2], [0.991445, 1.55599e-08, 0, -0.130526]) .
```

Figure 2: Expected output of the first four test cases.

Hints: 1. There are multiple ways to fill in the blanks, and you only need to ensure the resulting robot's behavior matches Fig. 1 and Fig. 2; 2. Understand the toy car example provided in *tutorial/create_articulation.py*; 3. See the comments in the notebook.

2. **Forward kinematics. (5pts)** For the robot arm you just created, we want to calculate the forward kinematics. Specifically, given a set of joint angles $\theta = [\theta_1, \theta_2, \theta_3]$, the pose of the link 'end_effector' can be represented as a function $T(\theta)$. Here, we use a $SE(3)$ matrix to represent a pose. Please calculate the expression of $T(\theta)$.

   Hints: 1. $T([0.1\pi, 0.2\pi, -0.3]) = \begin{bmatrix} 0.588 & 0.809 & 0 & 2.112 \\ -0.809 & 0.588 & 0 & 2.697 \\ 0 & 0 & 1 & 1.15 \\ 0 & 0 & 0 & 1 \end{bmatrix}$; 2. You can also check your results with the function *test_FK*.

3. **Reachable workspace. (2pts)** The set of positions (in $\mathbb{R}^3$) which can be reached by the end-effector with some choice of joint positions is called the *reachable workspace*:

$$W_R = \{p(\theta) : \theta \in Q\} \subset \mathbb{R}^3$$

   where Q is the configuration space and $p(\theta) : Q \to \mathbb{R}^3$ is the position component of the forward kinematics map $T(\theta)$. Please describe the reachable workspace for our robot arm, where $Q = [0, 2\pi) \times [-\frac{\pi}{2}, \frac{\pi}{2}] \times [-1, 1]$.

Hints: 1. You can describe the reachable workspace with figures and expressions; 2. In general, it will be challenging to analyze the reachable workspace if the robot is complex, but it's doable in our cases.

4. **Point velocity. (4pts)** Denote the center of the "left_pad" link as q. The coordinate of q in the spatial frame (i.e., base link frame) is denoted as q^s, while the coordinate in the end effector frame is denoted as q^e. Given the forward kinematics $T_{s\to e}(\theta)$, we can transform the coordinates by $q^s = T_{s\to e}(\theta) q^e$. Please first write down the expressions for both q^e and q^s.

   The velocity of point q in the spatial frame is given by $\mathrm{v}_q^s = \dot{q}^s$, where $\dot{q}^s$ indicates the derivative of q^s with respect to time. It is also possible to specify the velocity with respect to the (instantaneous) end effector frame, which is given by $\mathrm{v}_q^e = T_{s\to e}^{-1} \mathrm{v}_q^s$. For a specific joint position $\theta = (-\frac{\pi}{6}, \frac{\pi}{6}, \frac{1}{2})$ and joint velocity $\dot{\theta} = (1, 2, 1)$, please calculate v_q^s and v_q^e.

Problem 4. Contribution description (2pts).

Please write down team member names and describe the contribution of each here.