6ELEN018W - Tutorial 8 Exercises

1 Revision of Lecture Slides

Go through all of the Lecture 4 slides, making sure you understand them and run the described Python code. Make sure that you understand what you are doing or otherwise ask your tutor.

2 Pre-defined Robot Models in the Python Robotics Toolbox

Run in the Python toolbox (`rvctool` or simply import the relevant Python modules in your environment) the necessary Python statements to see all the pre-defined robot models in `URDF` format.

1. Create an instance of one of them. Choose one with 6-joints.
2. What are the pre-defined configurations for this robot?
3. Calculate the forward kinematics for a pre-defined configuration.
4. What is the location of the end-effector for this specific configuration?
5. Add a new configuration to the robot with your own defined joint angles.
6. Calculate the forward kinematics for this specific configuration.
7. What is the location of the end-effector for this specific configuration?
8. Plot the robot for your own configuration.
9. Calculate the Jacobian for your own configuration.

3 3-DOF Robot Calculations

A 3-DOF robot has joint angles q_1, q_2, q_3 and link lengths a_1, a_2, a_3.

Use the Python Robotics Toolbox to construct a robot using a chain of robot links (e.g. see the `Link2, ERobot2` classes in the lecture slides). Then using the Robotics toolbox:

1. Find the Jacobian of the robot for a specific configuration.
2. Find the inverse Jacobian of the robot for that configuration.
3. For a specific point in time (no loops required) can you calculate (in Python) the velocities of the joint angles that need to be applied by the actuators, in order to achieve a specific velocity of the end-effector?

4 Controlling the Velocity of the End-Effector of a Robot

Consider the Puma360 robot which is built-in the robotics toolbox as a model.

In Slide 23 of the lecture we calculated the instantaneous required angular velocities $(q_1, q_2, q_3, q_4, q_5, q_6)$ that need to be applied in order to achieve the desired velocity of $[0, 0, 1, 0, 0, 0]$.

We did this as:

```
>>> p560 = models.ETS.Puma560()
>>> p560.addconfiguration("qn", [0, math.pi/4, math.pi, \
                                 0, math.pi/4, 0])
>>> J = p560.jacob0(p560.configs["qn"])

>>> nu = np.array([0, 0, 1, 0, 0, 0])  # desired target v of end-effector

>>> np.linalg.inv(J)@nu  # angle velocities to be applied
```

However, after applying the calculated angular velocities for a small amount of time (time step=0.01) Δt the new angles of the joints become:

```
q = q + q*0.01
print(f'q_new:{q}')
```

which calculate to:

```
q_new:[0, 0.79325215, 3.17300858, 0, 0.79325215, 0]
```

and these are different than the previous q in the last time step, therefore the Jacobian has changed and the calculation of the required angular velocities is not valid any more!

- Write a Python program which keeps calculating the new Jacobian based on the new q values at each time step, in order to achieve the desired velocity for the robot hand after a number of time steps.

Hint: Look at Slide 22 of the Lecture.