6ELEN018W - Tutorial 3 Exercises

Reminder: Setting up the Python Robotics Toolbox on your Personal Computer

To set up the Robotics toolbox on your personal machine you must have a working version of Python installed.

Once you have this run either of the following commands (the second is only for Anaconda installations of Python). You can run these inside a Python virtual environment (the way you create and activate this, depends on the operating system of your machine) or the main installation of Python on your computer:

- `pip install rvc3python`
- `conda install rvc3python`

The above installation provides a command line tool `rvctool` which can be started executing its name. The tool starts a Python command line and it automatically imports all necessary modules for the toolbox.

Alternatively, you can start your own Python command line or IDE (e.g. JupyterLab) and import all the necessary modules before using the toolbox:

```
import math
import numpy as np
from scipy import linalg, optimize
import matplotlib.pyplot as plt
from spatialmath import *
from spatialmath.base import *
from spatialmath.base import sym
from spatialgeometry import *
from roboticstoolbox import *
from machinevisiontoolbox import *
import machinevisiontoolbox.base as mvb
```

You will need to import these modules in every program you write.

Exercise 1

Using the Python Robotics toolbox compute the homogeneous transformation which calculates the pose of the end-effector for a robot manipulator having 4 revolute joints.

The calculation should be done in symbolic form. The joints angles are q_1, q_2, q_3, q_4 and the lengths of the corresponding links are a_1, a_2, a_3, a_4.

Exercise 2

Consider the problem of a 2-joint robot arm discussed in the lecture.

Write a Python function which accepts 4 arguments, the two joints angles and the lengths of the 2 links.

1. Implement your function so that it returns the homogeneous transformation which describes the end-effector pose without using the Python Robotics toolbox. Use a `numpy` array for the return type.
2. Call the appropriate Python Robotics toolbox to do the equivalent computations and check that your two implementations give the same results.
3. Write a Python function which returns the x, y coordinates of the end-effector with respect to the world frame.

Exercise 3

A robot manipulator consists of 3 links having lengths $a_1 = 2, a_2 = 3, a_3 = 4$. The robot has rotated its joints at angles $(q_1, q_2, q_3) = (\frac{\pi}{2}, \pi, \pi)$.

Write some Python code which calculates the location of the hand (i.e. the end-effector) of the robot, in order to grasp an object.

Exercise 4

Consider the problem of a 3-joint robot arm discussed in the lecture.

Write a Python function which accepts 9 arguments, the three joints angles q_1, q_2, q_3, the angular velocities (rates of change) of the 3 joints $(\dot{q_1}, \dot{q_2}, \dot{q_3})$ and the lengths of the 3 links.

Your function should return the Jacobian matrix which can be used for the calculation of the velocity of the pose of the end-effector. Your implementation should NOT use the functions included in the Python Robotics toolbox.

Exercise 5

Consider the problem of a 3-joint robot arm from the previous Exercise 3.

Write a Python function which accepts 7 arguments the 3 joints angles, the lengths of the 3 links and a list with the desired velocity $(\dot{x}, \dot{y})$ for the end-effector. The function calculates and returns the required angular velocities of the 3 joints $(\dot{q_1}, \dot{q_2}, \dot{q_3})$ in order to achieve the desired velocity in the end-effector.

Hint: If you need to calculate the inverse of a matrix you can use the `numpy.linalg.inv()`. Make sure that you import `numpy` first.

Exercise 6

Show that the homogeneous transformation matrix for a 1-joint 2D manipulator (described in Lecture 2) corresponds to a translation a followed by a rotation θ and NOT to a rotation θ followed by a translation a.

Hint: Use the Robotics toolbox `trot2()` and `transl2()` functions.

Exercise 7

Using `SymPy` (without using the Robotics Toolbox, use the equivalent Python matrices) derive the equations which calculate the velocity of the End-Effector for a 2-joint robot, i.e. prove that the velocity of the end-effector is given by equations (6) and (7) in the Lecture slides.

Hint: Study the last slide of the lecture on the differentiation of an expression including an unknown function.