6ELEN018W - Tutorial 2 Exercises

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

Create a 2D rotation matrix using the Python Robotics toolbox. Visualise the rotation using trplot2. Use it to transform a vector. Invert it (using the `np.linalg.inv()` function) and multiply it by the original matrix. What is the result? Reverse the order of multiplication. What is the result?

Exercise 2

1. Write a Python function which accepts a single argument representing an angle θ in radians and returns a matrix in the form of list of lists with the rotation matrix about this angle θ.
2. Confirm the correctness of your implementation by calling the equivalent function in the Python Robotics toolbox which calculates the rotation matrix.
3. Extend your function so that it accepts a second argument which is a string. If the string is `'deg'` it calculates the rotation matrix in degrees, otherwise if the string is `'rad'` it calculates the rotation matrix in radians. For any other value of the second argument the function returns an error.

Exercise 3

Repeat the previous exercise (2) but this time you need to use a `numpy` array instead of a list of lists. You can learn how to create such arrays by studying the following link:

`https://numpy.org/doc/stable/reference/generated/numpy.array.html`

Exercise 4

Write a Python function which accepts 2 arguments corresponding to an angle θ (in degrees) and a list $(t_x t_y)$ corresponding to a translation vector. The function should return the 3×3 homogeneous transformation for that angle and translation.

Exercise 5

Create a 3D rotation matrix with angle π about the z axis using the Robotics toolbox. Visualise the rotation using `trplot`.

Exercise 6

Write a Python function which accepts a two arguments an angle θ and an axis ('x', 'y' or 'z') and returns the 3D rotation matrix with that angle with respect to the axis. The returned type should be a `numpy` array.

Test the correctness of your implementation by using the equivalent Robotics toolbox function.

Check the correctness of your implementation using the Robotics toolbox. The Robotics toolbox can calculate the composition of 3D homogeneous transformation using for example:

```
T = transl(2, 0, 0) @ trotx(pi / 2) @ transl(0, 1, 0)
```

where `transl(x, y,z)` creates a translation with x, y, z units respectively, `trotx` creates a relative pose corresponding to the given argument angle with 0 translation.

The above expression represents a translation along the x-axis by 2 units, followed by a rotation of 90° about the x-axis, followed by a translation of 1 unit along the new y-axis.

Exercise 7

Define a Python function which accepts 2 arguments corresponding to a coordinates vector (in `numpy` array type) and a rotation matrix (also in `numpy` array type).

The function should return the new coordinates of the original coordinates vector after its transformation according to the rotation matrix.

Test the correctness of the implementation by using the appropriate Python Robotics toolbox function.