1. Differential Drive Mobile Robot Labs

1.1 Installing The Required Software

This section will guide you through the steps required to install the software needed for the Lab.

1.1.1 Installing Jupyter Notebook

A Jupyter Notebook is an interactive computational environment which combines code execution, documentation, and visualization in a single web-based interface. Users can create and run code cells to perform calculations, manipulate data, and generate plots, while interspersing explanatory text, equations, and multimedia content in Markdown cells.

To install Jupyter Notebook using the Python package manager `pip`, you can follow these steps:

1. **Open a Terminal or Command Prompt**: Depending on your operating system (Windows, macOS, or Linux), open a terminal or command prompt with administrative privileges.
2. **Update** `pip`: It's a good practice to ensure that `pip` is up to date. You can do this by running the following command:
   ```
   $ pip install -upgrade pip
   ```
3. **Install Jupyter Notebook**: Once `pip` is up to date, you can install Jupyter Notebook using the following command:
   ```
   $ pip install jupyter
   ```
4. **Verify Installation**: After the installation is complete, you can verify that Jupyter Notebook is installed correctly by running:
   ```
   $ jupyter notebook
   ```
   This command should start the Jupyter Notebook server and open a new web browser window displaying the Jupyter Notebook dashboard.

1.1.2 Installing Python Libraries in Visual Studio Code

In this section, we'll guide you through the process of installing NumPy, Matplotlib, and the Robotics Toolbox for Python (roboticstoolbox-python) in both Visual Studio Code and PyCharm.

Open Visual Studio Code

Launch Visual Studio Code on your computer.

Create or Open a Python Project

Either create a new Python project or open an existing one in Visual Studio Code.

Open the Terminal

Inside Visual Studio Code, open the integrated terminal.

Install NumPy and Matplotlib

In the terminal, run the following commands to install NumPy and Matplotlib using `pip`:

```
$ pip install numpy
```

```
$ pip install matplotlib
```

Install Robotics Toolbox for Python

If you want to install the Robotics Toolbox for Python (roboticstoolbox-python), you can use `pip` as well:

```
$ pip install roboticstoolbox-python
```

Verify Installation

You can verify that the libraries are installed correctly by creating a Python script and importing them. For example:

```
import numpy as np
import matplotlib.pyplot as plt
import roboticstoolbox as rtb
```

If no errors occur, the libraries are successfully installed.

1.2 Cloning the PR_LAB1 Repository

In this section, we'll guide you through the process of cloning the "PR_LAB1" repository from GitHub onto your computer. This repository contains materials related to a laboratory exercise or project. Make sure you have Git installed on your computer before proceeding.
Follow these steps to clone the repository:

1. **Open a Terminal or Command Prompt**: Depending on your operating system (Windows, macOS, or Linux), open a terminal or command prompt.
2. **Navigate to the Desired Directory**: Use the 'cd' command to navigate to the directory where you want to store the repository. For example:
   ```
   $ cd Documents
   ```
3. **Clone the Repository**: Run the following command to clone the "PR_LAB1" repository from GitHub. Replace '<your-username>' with your GitHub username:
   ```
   $ git clone https://github.com/pere-ridao-rodriguez/PR_LAB1.git
   ```
   If you have GitHub authentication set up, you won't need to provide credentials. If not, GitHub may prompt you to enter your username and password.

4. **Verify Cloning**: Once the cloning process is complete, navigate to the cloned repository's directory using the 'cd' command:
   ```
   $ cd PR_LAB1
   ```
   You should now be inside the cloned repository. You can check the contents to verify that everything has been cloned correctly.

That's it! You've successfully cloned the "PR_LAB1" repository onto your computer. You can now work on the contents of the repository or use them for your laboratory exercise or project.

1.3 Lab 0: Pose Compounding

This exercise explains how to define a robot pose in 3 Degree Of Freedom (DOF) as well as how to compose different poses using the direct and the inverse compounding operations.

1.3.1 Step-by-Step Guide

In this section, we'll walk you through the process of completing and executing the "Lab0_compounding.ipynb" notebook using Jupyter Notebook. Follow these steps to work with the notebook:

1. **Start Jupyter Notebook**: Open a terminal or command prompt on your computer and navigate to the directory where the "Lab0_compounding.ipynb" notebook is located. Launch Jupyter Notebook by running the following command:
   ```
   $ jupyter notebook
   ```
   This will open the Jupyter Notebook dashboard in your web browser.
2. **Access the Notebook**: In the Jupyter Notebook dashboard, locate and click on the "Lab0_compounding.ipynb" notebook. This will open the notebook in a new tab.
3. **Navigate the Notebook**: The notebook consists of a series of cells, including Markdown cells for instructions and code cells for calculations. Read the instructions in each cell carefully.
4. **Execute Code Cells**: For code cells, you'll need to run them to perform calculations. To execute a code cell, select it and click the "Run" button in the Jupyter Notebook toolbar or use the keyboard shortcut Shift + Enter. The code will run, and any output or results will be displayed below the cell.
5. **Submit Your Work**: If this notebook is part of an assignment, follow your instructor's guidelines for submission. You may need to export the notebook as a PDF and submit it through the designated platform.

By following these steps, you'll be able to complete and execute "Lab0_compounding.ipynb" in Jupyter Notebook.

1.4 Lab 1.a: Simulation And Dead Reckoning

The `SimulatedRobot` and `DifferentialDriveSimulatedRobot` classes are integral components of a robot simulation framework designed to model and study the localization and mapping algorithms of robotic systems within a virtual environment.

1.4.1 Simulation

SimulatedRobot Class

The `SimulatedRobot` class serves as the foundational element of the simulation framework. It defines a generic robot model, providing attributes and methods for motion modeling, sensor

simulation, parameter initialization, and visualization. This class is designed to be extensible, allowing developers to create specialized robot simulation classes by inheriting from it.

DifferentialDriveSimulatedRobot Class

The `DifferentialDriveSimulatedRobot` class is a specialized extension of the `SimulatedRobot` class. It focuses on modeling the behavior of robots with a differential drive configuration, typically consisting of two independently controlled wheels. This class overrides certain methods and attributes of the `SimulatedRobot` class to tailor them to the specific characteristics of differential drive robots. It provides motion modeling specific to differential drive robots, simulates sensors such as wheel encoders and compasses, and customizes parameterization for accurate simulation.

To Do

In this part of the lab we will complete the simulation code for a 3 DOF Differential Drive Mobile Robot.

1. **Understanding the *SimulatedRobot* base class**:
   (a) Read its documentation in the *"prpy: Probabilistic Robot Localization Python Library"* document [**prpy23**].
   (b) Check the code of the *RobotSimulation.py* file.
2. ***DifferentialDriveSimulatedRobot* class**:
   (a) Read its documentation in the *"prpy: Probabilistic Robot Localization Python Library"* document [**prpy23**].
   (b) Check the uncompleted code provided for this class in the *DifferentialDriveSimulatedRobot.py* file.
   (c) Complete the code of the class by implementing the methods labeled as *"# Todo: To be completed by the student"*
3. **Test the Robot simulation**:
   (a) Program the robot simulation to perform a circular shaped trajectory.
   (b) Program the robot simulation to perform a trajectory shaped as an "8".

1.4.2 Localization

Localization Class

The ‘Localization‘ class serves as the base class for implementing and executing localization algorithms within a simulated robotic environment. This class provides the fundamental structure and methods necessary for simulating and solving the robot localization problem.
A detailed description of the class can be found in [**prpy23**].

DifferentialDriveSimulatedRobot Class

The ‘DifferentialDriveSimulatedRobot‘ class represents a simulated robot with a differential drive motion model. It serves as an extension of the ‘SimulatedRobot‘ class and overrides several of its methods to define the specific motion model and behaviors of a differential drive robot.
A detailed description of the class can be found in [**prpy23**].

To Do

In this part of the lab we will solve the Dead Reckoning Localization of a 3 DOF Differential Drive Mobile Robot.

1. **Understanding the *Localization* base class**:

   (a) Read its documentation in the *"prpy: Probabilistic Robot Localization Python Library"* document [**prpy23**].
   (b) Check the provided code of the *Localization.py* file.
2. **Completing the *DR_3DOFDifferentialDrive* class**:
   (a) Read its documentation in the *"prpy: Probabilistic Robot Localization Python Library"* document [**prpy23**].
   (b) Check the uncompleted code provided for this class in the *DR_3DOFDifferentialDrive.py* file.
   (c) Complete the code of the class by implementing the methods labeled as *"# Todo: To be completed by the student"*
3. **Testing the Robot Dead Reckoning Localization**:
   (a) **Circular Trajectory**: Check the localization results of the implemented Dead REckoning algorithm.

Bibliography

[1] Pere Ridao and Roger Pi. *prpy: Probabilistic Robot Localization Python Library*. October 2023. Available online in the course moodle.