Week 6 Lab Notes

A. Localisation of the Robot (Dead Reckoning)

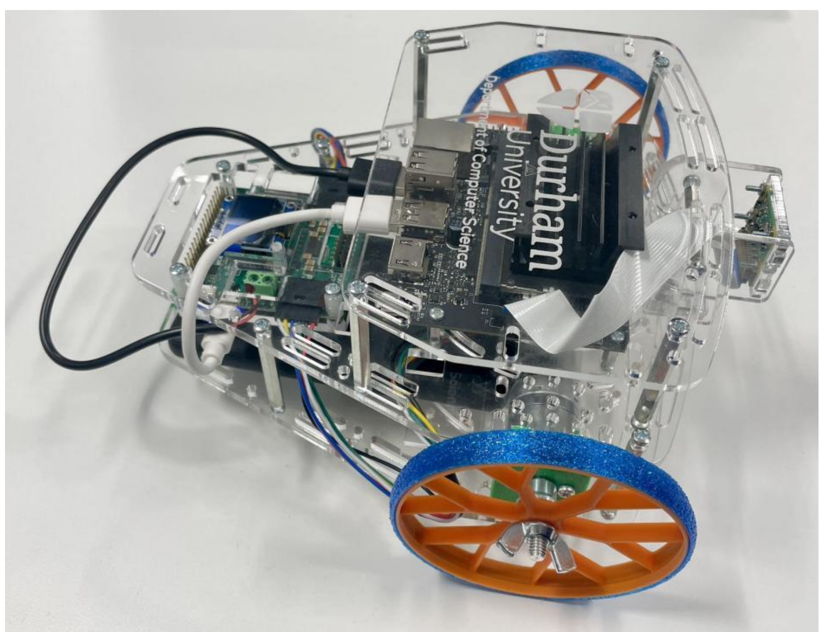
Objective

The purpose of this lab is to understand how to implement a motion-based linearisation technique for mobile robot.

Learning outcomes

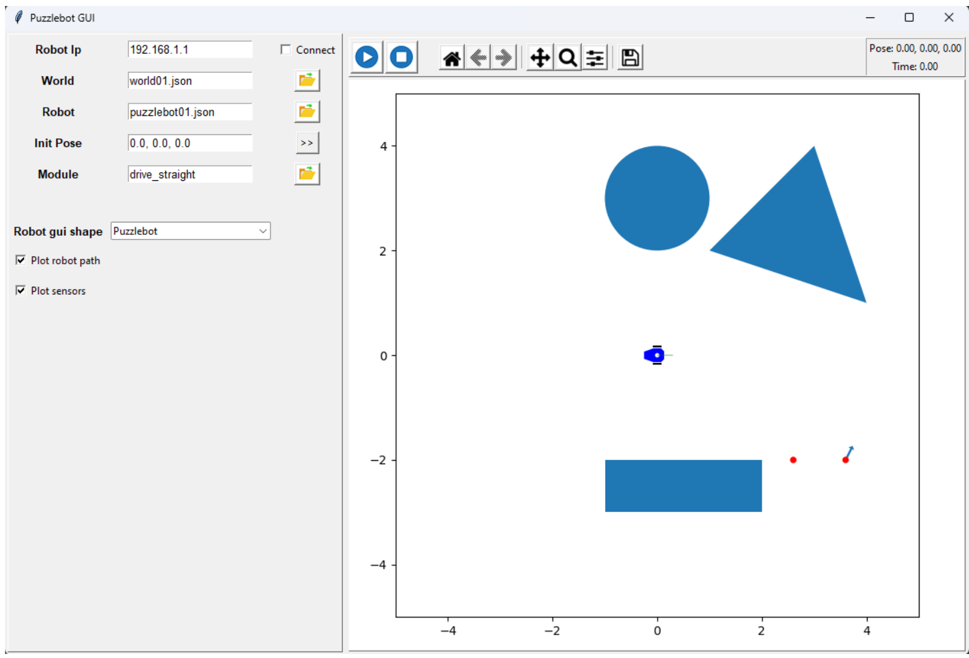
At the end of this exercise, you should be able to:

- Implement a Dead Reckoning Localisation for a two-wheel mobile robot and navigate the robot along a predefined trajectory.
- Assess the performance of the robot under the motion-based localisation.
- Observe and analyse the performance when the robot performing a continuous task.

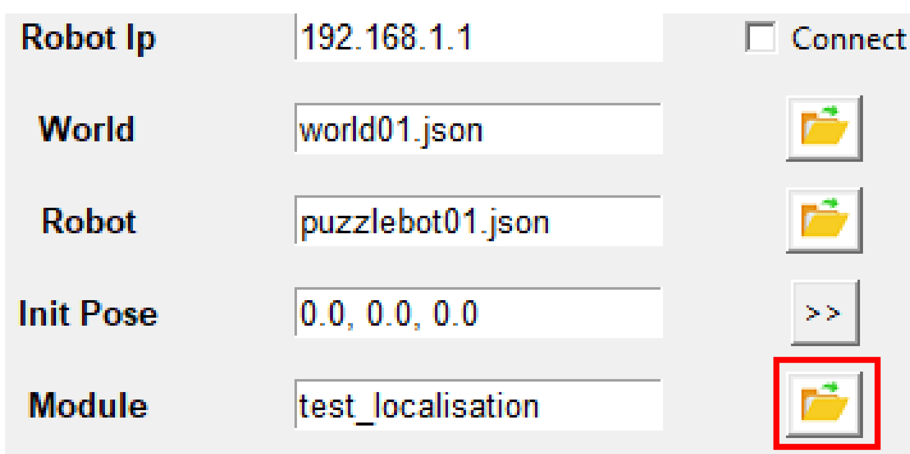


Task 1: Preparation

- Download ‘`test_localisation.py`’ and ‘`dead_reckoning.py`’ files from Ultra, and put them into the “`my_examples`” folder.
- Open the GUI by running puzz_gui.py



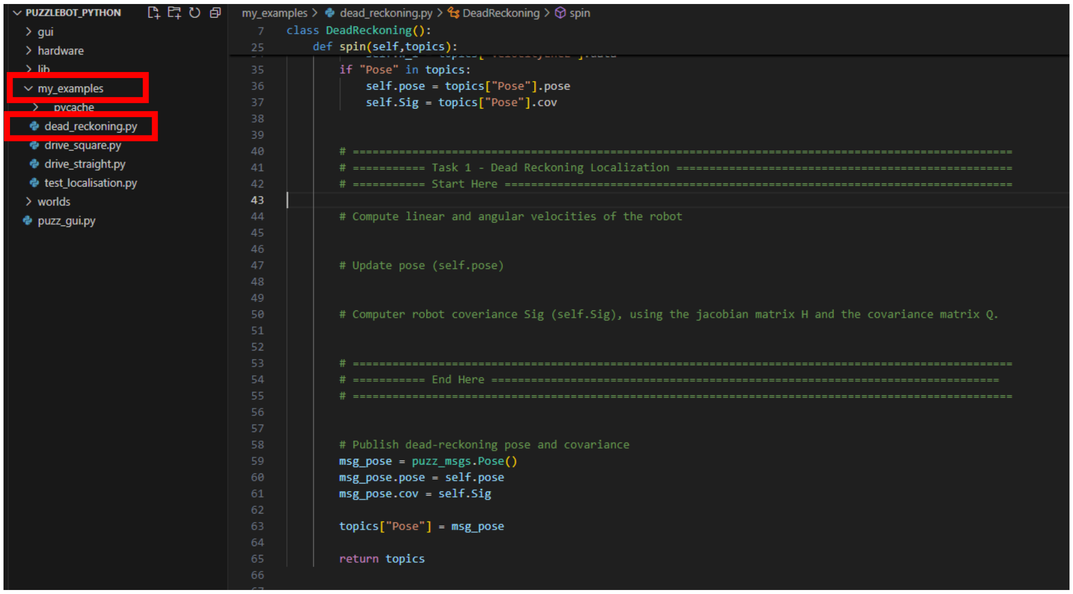
- Select ‘**Module**’ with ‘`test_localisation.py`’. Note, there is a basic control program at this file for testing ‘`dead_reckoning.py`’.



Task 2: Localisation

Implement a Dead Reckoning Localisation algorithm following the next steps:

- Open ‘`dead_reckoning.py`’ file.
- Write your code for localisation in ‘`dead_reckoning.py`’ file in the allocated section for this task.

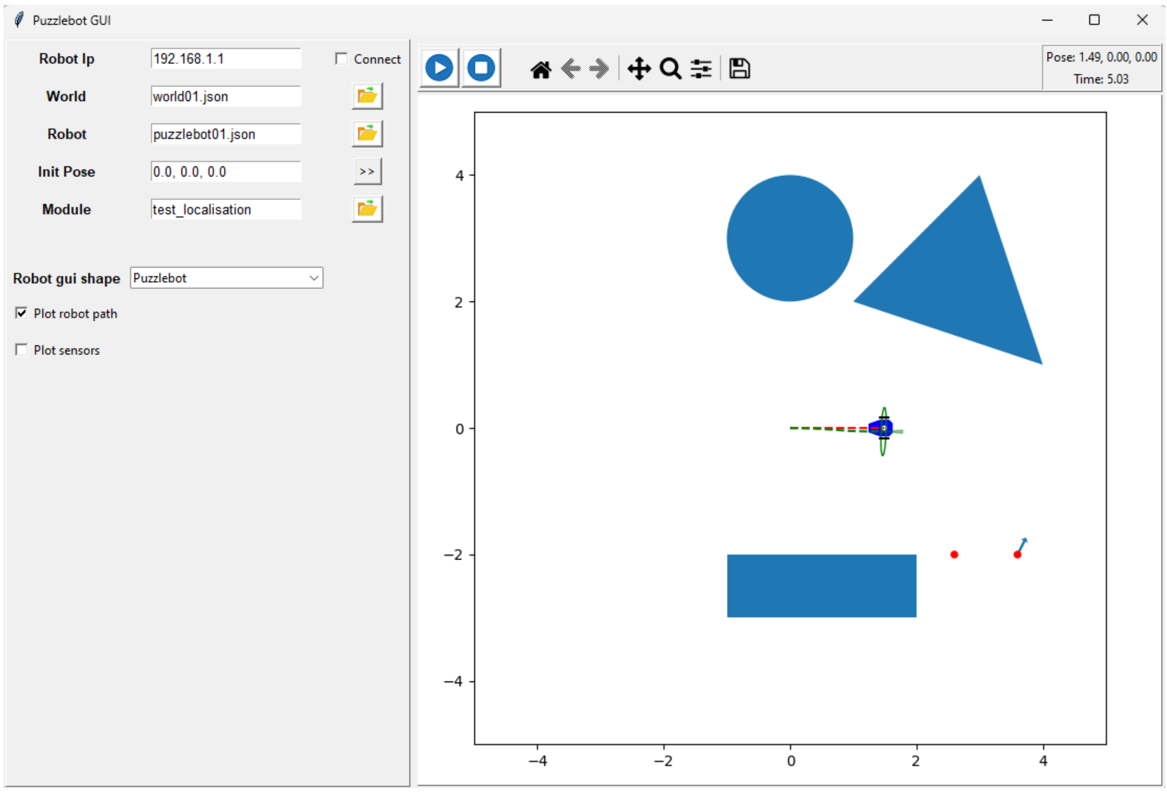


The following parameters are used in the program.

Parameter	Notation	Description
self.pose	$\boldsymbol{\mu}_k$	Robot pose mean (3x1) $[x\ y\ \theta]^T$ where $x[m]$, $y[m]$ and $\theta[rad]$
self.Sig	$\boldsymbol{\Sigma}_k$	Robot pose covariance (3x3)
dt	Δt	Sampling time (1x1) in seconds $[s]$
self.w_l	ω_l	Left motor encoder reading $[rad/s]$
self.w_r	ω_r	Right motor encoder reading $[rad/s]$
self.R	r	Radius of the wheels (0.05 $[m]$)
self.L	l	Robot wheel base (1x1) (0.09$[m]$)
self.k	$k_r = k_l$	Error associated with computing the angular velocity for each wheel

Note that, self.pose and self.Sig are the parameters to be updated by your code

- Run your code using GUI by clicking the icon .
- You will see the robot covariance like the following figure if your codes are correct.



[End of Lab]