

Robotics Coursework

(2024-2025)

1. Coursework introduction

This coursework includes three tasks. Task 1 is to apply a motion-based localisation technique to the mobile robot. Task 2 is to develop a motion control strategy based on the robot model and the developed localisation method. Task 3 is to test the effectiveness of the proposed motion control algorithm using a real mobile robot. For detailed descriptions and tasks, please refer to the individual parts.

You should submit your completed assignments via Blackboard Ultra. The deadline for your submissions is at **2 pm** on **Monday 13th of January, 2025**. Each of you should include the following four files in your submission:

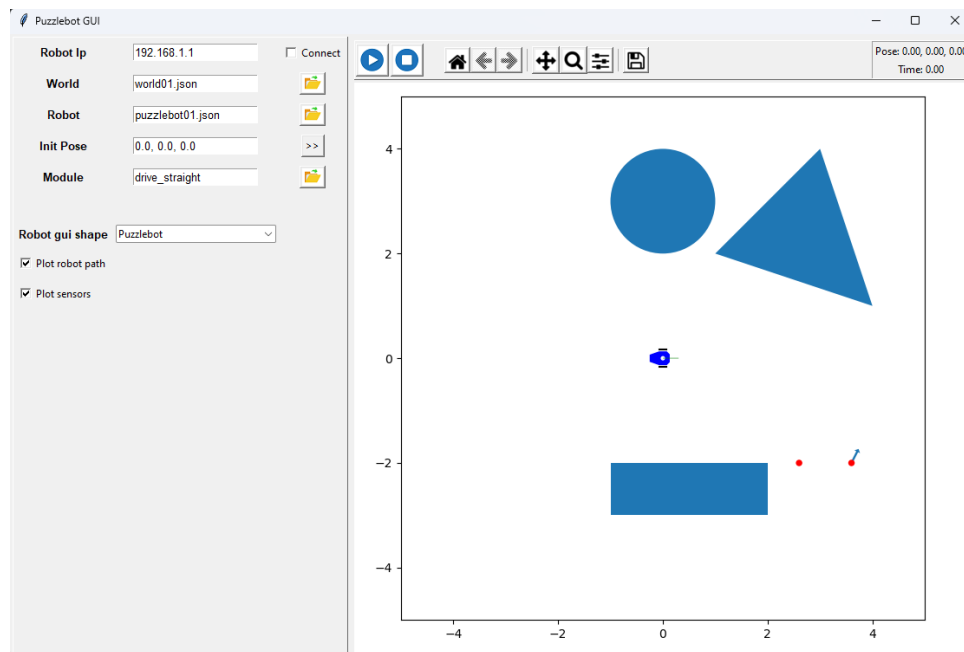
- 1) The report (PDF format)
- 2) The localisation file (`dead_reckoning.py`)
- 3) The motion control file (`driveToGoal.py`)
- 4) The video of the real-robot experiment.

For the report, the **maximum** number of pages is **6** and the **minimum** font size is **11pt** (Only use Arial, Calibri, Times New Roman). Please put each task in a separate section and describe the method and the reasons for choosing such approach. If there are tuning parameters, describe how to tune them and what values to choose. If necessary, put the results in the form of diagrams, figures, tables, etc. You should also include discussion, reflection, conclusion, and recommendation about the results. Please note that, providing a collection of results and figures without detailed explanations and reflections will correspond to low marks!

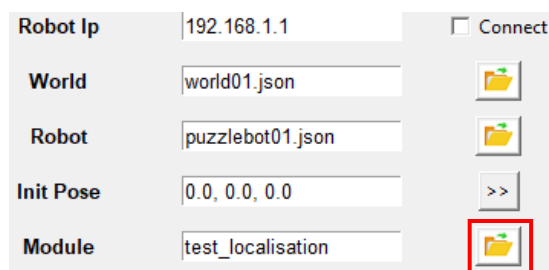
2. Tasks

Preparation (0 marks).

- Download 'test_localisation.py', 'dead_reckoning.py' and 'driveToGoal.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 1 (30 marks): 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.

```


7 class DeadReckoning():
25     def spin(self, topics):
35         if "Pose" in topics:
36             self.pose = topics["Pose"].pose
37             self.Sig = topics["Pose"].cov
38
39
40         # ===== Task 1 - Dead Reckoning Localization =====
41         # ===== Start Here =====
42
43         # Compute linear and angular velocities of the robot
44
45
46         # Update pose (self.pose)
47
48
49         # Computer robot covariance Sig (self.Sig), using the jacobian matrix H and the covariance matrix Q.
50
51
52         # ===== End Here =====
53         # =====
54
55         # Publish dead-reckoning pose and covariance
56         msg_pose = puzz_msgs.Pose()
57         msg_pose.pose = self.pose
58         msg_pose.cov = self.Sig
59
60         topics["Pose"] = msg_pose
61
62         return topics
63
64
65
66

```

The following parameters are used in the program.

Parameter	Notation	Description
self.pose	μ_k	Robot pose mean (3x1) $[x \ y \ \theta]^T$ where $x[m]$, $y[m]$ and $\theta[rad]$
self.Sig	Σ_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 .
- Explain the performance of the localisation.
- Discuss the strategy to further improve the performance.

End of Task 1

Task 2 (35 marks): Motion Control

Implement a control algorithm to move the robot from its current position to an arbitrary goal point within the boundaries of the simulated world, following the next steps:

- Open 'driveToGoal.py' file.
- Set the goal point in line 16 and line 17, e.g., $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$.
- Write your code for Motion Control in the allocated section for this task

```


10 class DriveToGoal():
11     def __init__(self):
12         self.w_setR = 0.0
13         self.w_setL = 0.0
14
15     def spin(self, topics):
16         self.dead_reckon.spin(topics)
17         est_pose = self.dead_reckon.pose
18
19         # ===== Task 1 - Motion Control =====
20         # ===== Start Here =====
21
22         # Hint: You will need to calculate the distance and angular error first, and then use these information for your proportional controller design.
23         # Note that, you need to set a condition such that the robot will stop moving once it is close enough to the goal, e.g., 0.1 m.
24         # Besides, considering the motor saturation, the directly generated control signal may be unrealistic, how you can address this issue in your controller design?
25         # In the end, you will need to obtain the desired left and right motor speed to achieve the goal.
26         # Good luck!
27
28         # ===== End Here =====
29
30         # Publish wheel velocities
31         msg_w_setR = puzz_msgs.Float32()
32         msg_w_setL = puzz_msgs.Float32()
33         msg_w_setR.data = self.w_setR
34         msg_w_setL.data = self.w_setL
35
36         topics["VelocitySetR"] = msg_w_setR
37         topics["VelocitySetL"] = msg_w_setL

```

The following parameters are used in the program.

Parameter	Notation	Description
est_pose	μ_k	Robot pose mean (3x1) $[x \ y \ \theta]^T$ where $x[m]$, $y[m]$ and $\theta[rad]$
self.target_x self.target_y	\mathbf{x}_g	Goal point (2x1) $[x \ y]^T$
self.w_setR	ω_r	Right motor speed set point. $[rad/s]$
self.w_setL	ω_l	Left motor speed set point. $[rad/s]$
self.w_max	ω_{max}	Maximum angular speed of both the right and left wheels $[rad/s]$.
self.R	r	Radius of the wheels (0.05 $[m]$)
self.L	l	Robot wheel base (1x1) (0.09 $[m]$)
topics["IsDone"] = True	...	A command to stop the simulator.

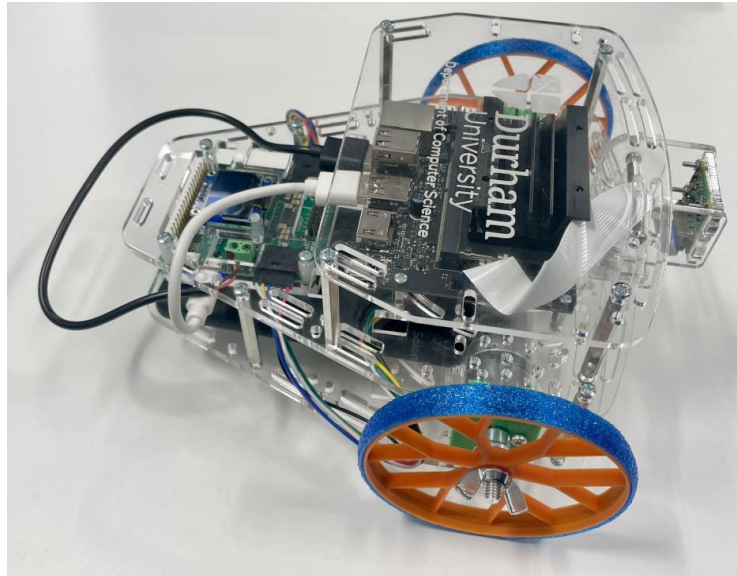
Note that, self.w_setR and self.w_setL are the parameters to be updated by your code.

- Run your code using GUI by selecting '**Module**' with '`driveToGoal.py`' and clicking the icon .
- Test your algorithm for different scenarios with different goal points.
- Explain the performance of your control strategy.

End of Task 2

Task 3 (20 marks): Real-Robot Experiments

Apply the motion control algorithm developed in Task 2 to a real robot, Puzzlebot.




You should follow the next steps to apply your control strategy to the robot:

- Check the IP address in the GUI and the robot IP address that is shown in the screen of the real robot. If they are different, change the IP address in the GUI to be the same as the one shown on the robot screen.

Robot Ip	192.168.1.1	<input type="checkbox"/> Connect
World	world01.json	
Robot	puzzlebot01.json	
Init Pose	0.0, 0.0, 0.0	
Module	driveToGoal	

- Click the checkbox 'Connect'

Robot Ip	192.168.1.1	<input checked="" type="checkbox"/> Connect
World	world01.json	
Robot	puzzlebot01.json	
Init Pose	0.0, 0.0, 0.0	
Module	driveToGoal	

- Run your code using GUI by clicking the icon .
- Explain the performance of your control strategy and discuss the difference between simulation and real experiments.
- Record a video of the real robot moving towards the desired goal position.

End of Task 3

3. How to submit?

You will submit four files:

- 1) `dead_reckoning.py`
- 2) `driveToGoal.py`
- 3) A .pdf file. In this file, you present your answers for Task 1-3 (**maximum 6-page**).
- 4) A video file, only one of the following should be used: mpeg, mp4, mpg, (**maximum 1 min**).

Important: You must compress all files into a single **.zip** file, and name your .zip file as “**YourName_YourID.zip**”, make sure to replace “YourName” with your own name and “YourID” with the ID given to you by the University.

For example, JohnDavidson_cgab36.

Please upload your compressed file to Ultra for submission. The total size of your .zip file should not exceed **20MB**.

The submission deadline is at **2 pm** on **Monday 13th of January 2025**.

4. Collaboration policy

You may discuss your work with anyone, but you must complete your work yourself independently and comply with the University rules regarding plagiarism and collusion (<https://www.dur.ac.uk/learningandteaching.handbook/6/2/4/1/>). Your submissions will be assessed for collusion and plagiarism.

5. Feedback Sheet

Task 1 – Localisation (30 marks)

Content	Mark
Plot robot estimated path and covariance ellipsoid	5
Explain and reflect why the real robot path is different from the estimated path (What are the problems with localisation? Why? Real system expectations? Is the behaviour expected? How does it relate to literature/theory? etc.)	10
Show, explain and reflect on the effect of the noise and the covariance unbounded propagation over time (Does the covariance propagate equally when the robot moves straight vs when it turns and why? Tests used to prove this? Conditions, constraints, real system expectations? Is the behaviour expected? How does it relate to literature/theory?, etc.)	10
Propose a potential solution for a better localisation performance and justify why it will help. The implementation is not required.	5

Task 2 – Motion Control (35 marks)

Content	Mark
Plot robot estimated path and covariance ellipsoid	5
Explain the method you used to achieve the desired motion control goal. Equations and some details are expected.	10
Explain and show the parameter tuning methodology and the advantages and disadvantages of the controller used (Tuning methodology? Which tests were used to tune the parameters? Constraints? Acceptance conditions? Comparisons, advantages and disadvantages of the control method? Is the control affected by nonlinearities? How? etc.)	10
Testing the algorithm for two different scenarios (different goals, plots for each case).	5
Propose different approaches to enhance the performance and reflect on the robot behaviour (Why we expect better performance? Comparisons, discussions, improvements, theoretical analysis) The implementation is not required.	5

Task 3 – Experiments (20 marks)

Content	Mark
Plot real robot trajectory observed from simulator	5
Explain the performance of the real-robot experiment. What are the differences between real-robot experiments and the simulations? How do you improve the performance of the mobile robot by considering real-world uncertainties and disturbances?	10
A good quality video record of the experiment showing the smooth movement of the robot in the real-world environment. Full 5-mark will be deducted from overlength videos!	5

Others (15 marks)

Content	Mark
Report presentation, language, format and clarity	10
Correctness and clarity of the codes	5