THE UNIVERSITY OF SYDNEY

MTRX3760 - Project 1
Hello, Turtlebots!

This project contributes 10% towards your final mark. You must complete it in **groups of four or five** students, details below. Total Marks: 100.

Project 1 is due Friday of Week 9 before midnight. Late assignments will be subjected to the University's late submission policy unless accompanied by a valid special consideration form. Plagiarism will be dealt with in accordance with the University of Sydney plagiarism policy. **Lab submissions will not be accepted more than a week after the due date.**

Submissions will be assessed based on the following components. Incomplete submissions will have severely reduced marks.

- **Report**: Each group will submit a .pdf report using the class Canvas site. The report can be very simple, and needs only directly address the questions laid out in the assignment.

  The front page of your report should include the SIDs and tutorial sections of all members of your group. Do not include names on the cover page to comply with the University's anonymous marking policy.

  Code appendix: The appendix of your report **must** contain a **printout** (in text format) of all source code you wrote for the assignment. File header comments and proper formatting will be critical to making this section readable.

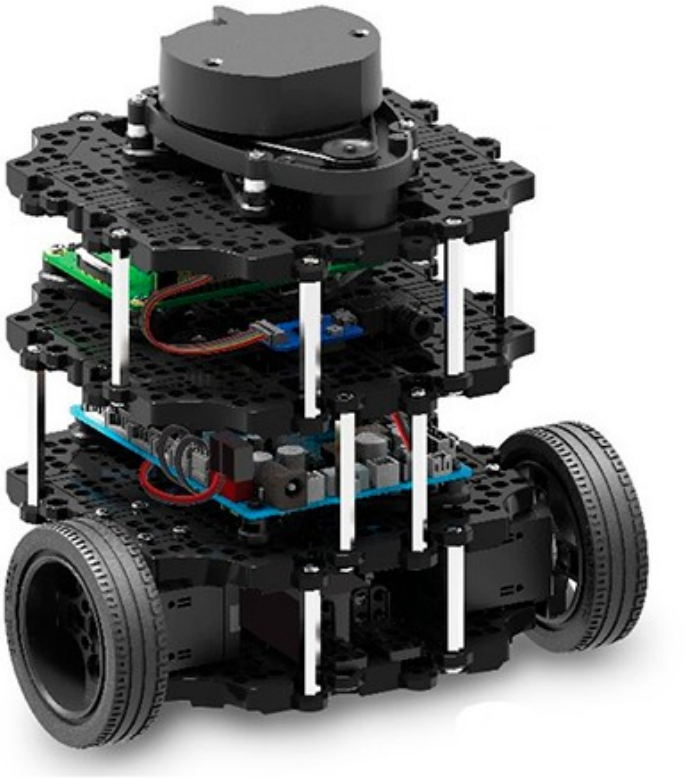
- **Code**: Submit your code (only code you wrote, and no binaries) in a .zip file via the canvas site.

- **Video:** Submit a brief video showing your turtlebot solving a maze. The video need not be elaborate, narrated or subtitled. The video is strictly to provide a record of the turtlebot correctly achieving the task described for this project.

<u>Group Work</u>
You **must** complete this project in a **group of four or five** students.

- Each group must nominate a leader who will submit the project to canvas for the group; only one group member should submit the work
- All group members must be in the same lab section
- Projects by larger groups will be graded with higher expectations

THE UNIVERSITY OF SYDNEY



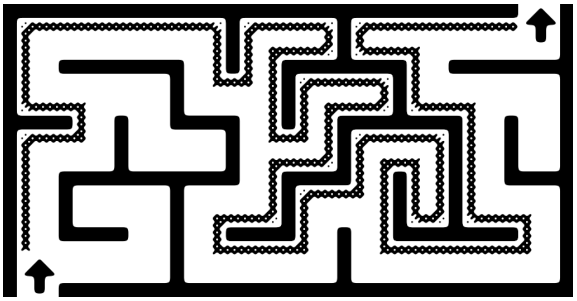
Turtlebot Setup

Refer to the accompanying “MTRX3760 Turtlebot User’s Guide” for important safety details and getting started guide for the Turtlebot 3 mobile robotic platform.

Maze Solver in Simulation

Simulate a Turtlebot 3 in RViz and Gazebo, including simulated laser and camera, by following the instructions here, up to and including Section 6.2:
https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/

Now implement your own code that gives the turtlebot the ability to autonomously navigate through a complex space. Wall following is a classic approach to solving this kind of problem. The method works like it sounds: a robot chooses to either follow the wall to its right, or to its left, until it reaches the maze exit. Shown here is a left- wall-following robot solving a maze:



[adapted from https://commons.wikimedia.org/wiki/File:Maze01-02.png]

To add wall following capability to your project, locate and modify `turtlebot3_drive.cpp` so that instead of driving around aimlessly, the robot performs left wall following implementing a wall-following algorithm. Hint: start by looking inside `turtlebot3_drive.cpp`.

Autonomous Maze Solver

Once you've established your maze solving approach can work in simulation, you will migrate it to operate on a physical Turtlebot3 platform. Show your tutor your working maze solver working in the lab, and sign out a turtlebot.

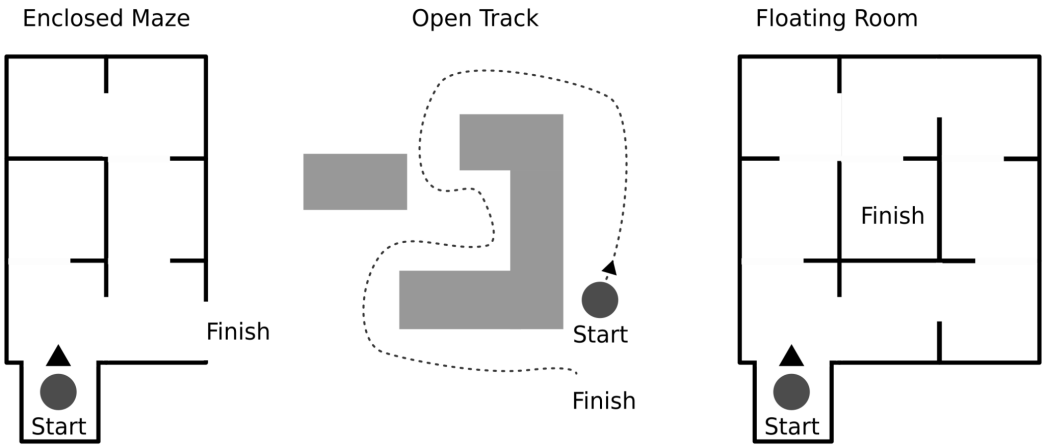
Note that we are entrusting your team with an expensive piece of equipment. We expect the turtlebot and all kit contents to be returned in good working condition. There are marks associated with this in the Major Project.

Simplification: For this project all decision-making may occur on a remote PC, you need not make the code run on the turtlebot.

Testing / Demonstration

It is up to you to decide how to test and demonstrate your system. You may collaborate with other teams in creating test scenarios and demonstration setups. Cardboard, walls, and furniture may be employed to build up maze walls. **Please be mindful to leave the lab in a clean state at the end of your session.**

Carefully consider which type of maze you would like to approach. Some types of maze are easier to solve than others, see three examples below. Solving a more challenging maze will attract extra marks. To get full credit for functionality it must be clearly demonstrated in your video.



When demonstrating your approach, you may make some simplifying assumptions, e.g. that the robot always begins adjacent to a wall to its left or to its right.

Code Refactor / Redesign

Chances are your implementation process for the autonomous wall follower resulted in some less-than-ideal coding decisions. Using git to prevent doing irreversible damage, refactor or redesign your code so that it is a well-structured object-oriented program. Consider how to best split up your code into classes, how best to split your classes into files, and which techniques covered in lecture are appropriate for making this program as clear and maintainable as possible.

Submit only your final, refactored, working code.

Video

Include in your submission a video of your physical (not simulated) turtlebot autonomously solving a maze. Ideally the robot solves the maze autonomously without requiring any intervention.

THE UNIVERSITY OF SYDNEY

Grading

[30 Marks] **Functionality and Testing**

In your report include images depicting your turtlebot **simulation** proving the following:

- The simulation works and includes LiDAR and camera
- Your solution can solve a maze, by showing the trajectory followed by the robot

Screenshots are ideal here. You should not draw the path followed by the robot by hand, make the computer do the work. Include captions briefly describing what each figure in your report is depicting.

In your **video** show the physical turtlebot autonomously solving a maze. For full marks, the robot should be able to solve a maze autonomously without intervention. Up to +10 bonus marks are available for solving more complex maze types, e.g. involving a floating room.

The report, video and code will be assessed for correct functionality.

[30 Marks] **Code Quality and Development Process**

Revision Control: Include in your report your git commit history as reported by:
`git log --oneline`
Note: copy/paste the text, screenshots are unacceptable

Teamwork: List in brief bullet-point form the contributions of each team member to the project.

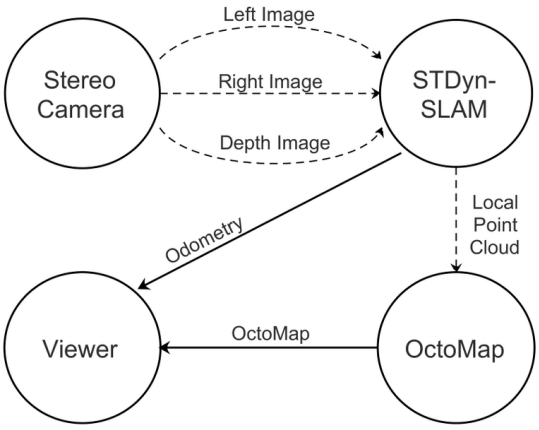
The code, revision history, and report will be assessed for code quality and development process.

THE UNIVERSITY OF SYDNEY

[40 Marks] **Design**

ROS Node Design Diagram: Draw a design diagram depicting how your program is split across ROS nodes. Each node should be in a circle, with arrows between nodes labelled with the key message(s) passed between them, like a typical top-level system diagram. Note that this is a ***design*** document and so rtq_graph or other automatically generated ***diagnostic*** **diagrams will not be accepted**.

The following shows an example, ignore the distinction between solid and dashed lines.

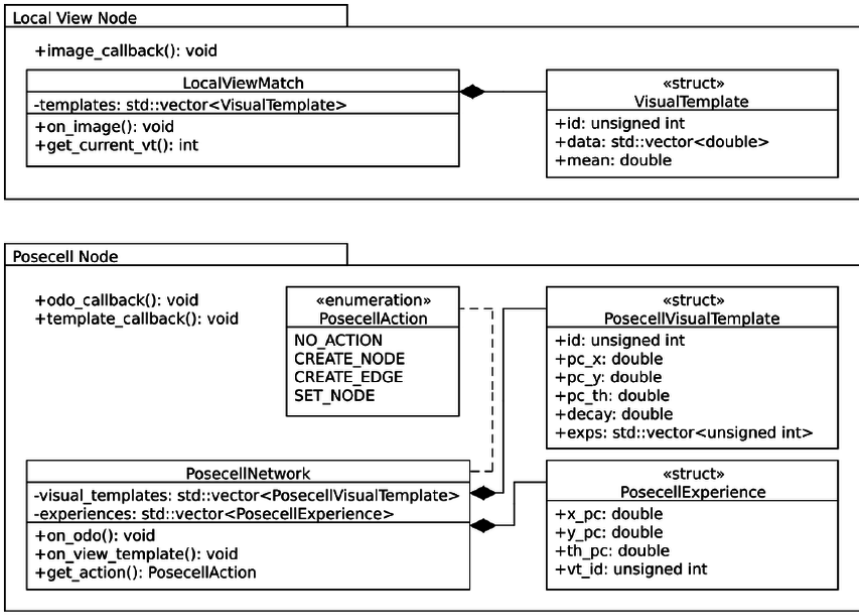


From Esparza et al "The STDyn-SLAM: A stereo vision and semantic segmentation approach for SLAM in dynamic outdoor environments", 2020.

UML Class Diagram: In your report provide a UML class diagram for the parts of your design that you programmed. You need not include member variables and functions.

Hints:

- Do include relationships between the classes you programmed and classes in ROS.
- In this unit we have emphasised composition (has-a), inheritance (is-a), and association (knows-a) relationships. These are *static* relationships, a car *always* has wheels, a student is *always* a person. Focus on getting these relationships right in your design and depicting them in your UML diagram.
- UML class diagram relationships don't exist between separate executables. Below is an example of a UML diagram for a program split across multiple nodes in ROS.



Adapted from Ball et al. "OpenRatSLAM: an open source brain-based SLAM system", 2013

The code and report will be assessed for design quality including following the design principles taught in lecture.

Self-Assessment

[+5 Bonus] In your report estimate the level of achievement of your submission. Show estimates for all component grades as well as the total. If your estimated total is within 5 marks of the correct score you will be awarded 5 bonus marks. Ignore late penalties when estimating.