UNIVERSITY OF LINCOLN

Lincoln School of Computer Science

Assessment Component Briefing Document

Title: CMP3103/CMP9050M Autonomous Mobile Robotics, Autonomous Mobile Robotics (M), Assessment Item One – Robotics Practical Coursework	**Indicative Weighting CMP3103M: 40%** **Indicative Weighting CMP9050M: 30%**

Learning Outcomes:

On successful completion of this component a student will have demonstrated competence in the following areas:

- [LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments

Requirements

This assessment item is split into two main tasks: "Group Robot Tasks" and "Escape the Maze", both detailed below.

Note: The "one-stop shop" for resources relating to this assessment component and the workshops in CMP3103M is https://github.com/LCAS/teaching/wiki/CMP3103M.

1. "Group Robot Tasks" (Criterion 1)

Your first task (relating to Criterion 1 "Group Robot Tasks" in the CRG, 20% of the assessment item one mark) consists of continuous engagement with a total of four workshop tasks you work on as a group of 3-4 students, demonstrated successfully on a real Turtlebot robot and in simulation.
The tasks will be made available to you at the beginning of each workshop session at the latest, and are to be demonstrated to a member of the delivery team by the week after the latest (this gives each team 2 workshop sessions to program, test and demonstrate each task). Demonstrating later than the defined deadline results in a reduction of the merit mark by 50%. for this individual task. This is *group work*, so it is fine to work together on the task, but only students who are present at the demonstration of the group work, and can who answer questions about it, will be awarded the marks. Absent group members will have to demonstrate the work individually, under the same rules as outlined above, ie, will only obtain half the marks if demonstrated later than the week after the workshop task has been officially released. The individual tasks are available from https://github.com/LCAS/teaching/wiki/CMP3103M.

2. "Escape the Maze" (Criterion 2 & 3)

Your second task (relating to Criterion 2 and 3 in the CRG, total of 80% of the mark for assessment item one) is to develop a reactive robot behaviour, programmed in Python using ROS (the Robot Operating System), that enables a robot to find a goal in a maze, utilising the robot's sensors (LIDAR, Odometry and camera). This assessment is entirely assessed in simulation, and *not* on the real robot, but you are encouraged to also try out your implementation on the real robots available. As part of this task, you must submit an *implementation* (criterion 2) and a *video* (criterion 3).

Implementation (Criterion 2)

Your task is to implement a behaviour that enables the robot in simulation to find the goal square (green), hidden in a maze. The maze features dead ends, and a few deadly traps (marked as red squares). You need to utilise the robot's sensors and actuators to safely navigate the robot in the maze, looking for the green goal while avoiding bumping into obstacles and driving onto any red squares. Success is defined as being on the green square. Your objective is to find the green square as quickly as possible. The maze also comprises visual (blue) clues that your robot *can* use to find the way to the goal quicker, but you are not required to make use of those cues.

For the development and demonstration of your software component, you will be provided with a simulation environment (called "Gazebo"). The required software is installed on all machines in the Labs and instructions to install the required software (Ubuntu 16.04LTS amd64 and ROS) on own computers are provided, though the delivery team cannot provide individual support with home installation. The simulated environment comprises a maze with tall walls, limiting the robot's ability to see beyond. Your robot starts from a predefined position and must try to find the goal completely autonomously. You will be provided with a number "training mazes" in simulation, available to you throughout the workshops, allowing you to develop your innovative solution. Your solution should be flexible enough to solve all provided mazes and also work in others that resemble them in terms of size, structure and complexity, but with varied positions of the walls, to assess the generality of your approach. Hence, your approach will have to be flexible and be able to work in an unknown environment. However, the colours of traps and clues will be the very same in all mazes.
You may choose any sensors available on the robot to drive your robot behaviour (you must not add more or other sensors).
The minimum required functionality consists of a simple reactive behaviour, allowing in principle to navigate a maze. For a good pass, the behaviour should be able to successfully (without hitting walls or falling into traps) make progress towards the goal. Further extensions are possible to improve your mark in this assessment, for instance making use of the additional clues provided to make the search more efficient.
The software component must be implemented in Python and be supported by use of ROS to communicate with the robot. The code should be well-commented and clearly structured into functional blocks, also employing OOP principles in Python. Your implementation needs to be submitted via blackboard, with the source code containing good comments, and appropriate references to all code and solutions re-used from previous work or 3rd parties.

Video (Criterion 3)

You are also required to submit a *short video* (of 2 minutes maximum duration, no more than 20Mb total size, MP4 format) of the simulation running your implementation in action. This video should have a *voice over*, but should you not have the equipment to record audio, you may alternatively just add a short *textual description* of the video content in MS Word (docx) format and submit it alongside your video. We suggest you use any screen recording tool you are familiar with to record the video, and then compress it to a suitable quality and size, but we also provide some best practices and tools to record and compress a video in Ubuntu 16.04 at https://github.com/LCAS/teaching/wiki/Recording-your-screen-on-Ubuntu-16.04. You may use other tools, but you have to make sure the video is compressed to less than 20Mb and in the correct format (MP4). You may even just use a mobile phone to film your screen with your implementation running and talk over it. Note, however, we *cannot accept a link to a video* shared on YouTube or anywhere else online. You must submit the video file.

Your video (with voice over or in the text description) should cover a presentation of exemplary performance of your implementation (screen recording of the simulation with your implementation running), a brief description of your overall system design, including details of the distinctive features of your perception and control algorithms, and some reflection on your implementation's performance. You make move the robot manually in the simulation during the recording or post-edit your video to show different situations you want to highlight. Make sure you present all your hard work briefly in the video recording.

Please follow carefully the instructions given in the lectures and workshops on the requirements for the video. You will be marked on both the functionality of your implementation, and the quality of your video presentation as detailed in the CRG.

Useful Information

This assessment is an overall individually assessed component (with a group assessment criterion 1). Your work must be presented per the Lincoln School of Computer Science guidelines for the presentation of assessed written work. Please make sure you have a clear understanding of the grading principles for this component as detailed in the accompanying Criterion Reference Grid. If you are unsure about any aspect of this assessment component, please seek the advice of a member of the delivery team.

Submission Instructions

The deadline for submission of this work is included in the school hand-in dates on Blackboard. Note that the deadline indicated is for the submission of both your <u>implementation</u> and your <u>video</u>, for the "Escape the Maze" task. There is no Blackboard submission for the group tasks (criterion 1).

You should submit *<u>ONE</u>* zip file, containing both your <u>implementation</u> and your <u>video</u>. The structure in your zip file should be as follows:

```
\
    \video      (this directory to contain your video)
    \src        (this being the src/ tree of your catkin workspace, or, if you have just a single
                 Python file with your implementation, this single file should go in here)
```

Your submission must contain <u>all</u> your developed code in the `src/` folder of the zip file, but *<u>no</u>* <u>compiled artefacts</u>. Do *<u>not</u>* submit the entire catkin workspace, just the complete `src/` folder if you indeed created your own catkin packages, or just the relevant Python files with your implementation. It is a highly recommended practise to create the zip folder for the submission and then try to recreate a running system solely from the content of that zip file. This way, you can have confidence that your markers will also be able to recreate your system. If your system requires more detailed instructions to get launch it than just locating a single executable Python file or a single `roslaunch` file, then it is required for you to add a `README.md` file in the `src/` folder of your zip file, with clear instruction to your marker on how to start your implementation. If your solution just consists of a single Python file or has one single launch file in the submission, then this is not needed.

The complete submission, including all developed source code and video must <u>not exceed 30Mb</u> in total (size of compressed ZIP file).

Note: This is an *individual assessment* and you must not plagiarise others' code. While it is acceptable to consult with your peers, your solution (functionality and source code) <u>must</u> be solely your own, any code fragments sourced from research or the internet most be cited correctly in the source code using comments. Submitted source code is automatically checked for plagiarism and you will have to answer to concerns regarding originality of your work during your viva. *Your implementation of navigating the maze is <u>not</u> group work!*

DO NOT include this briefing document with your submission.