METR4202 -- Robotics & Automation

Problem Set 3: Robotic Systems -- **Robotics: Capture the Flag**

Objective

Robotics gets one there (automatically). This project explores the integration of principle elements of a modern robot: kinematics, sensing, motion planning, control, and intelligent decision making. Teams are required to bring these elements together to develop an autonomous robot car that can autonomously navigate through a cluttered environment.

Requirement

This project tasks your team with programming an autonomous robotic vehicle capable of navigating through a cluttered environment and back again. Your team demonstration mark will be based on the robustness of your system, its ability to navigate in environments of increasing complexity and your ability to explain and justify your implementation.

In general, the system should be able to:

- Identify the flag at the end of the course
- Identify any obstacles on the course
- Plan and execute a route to the flag.
- Turn around and come back.

The **demonstration is worth 50%** of your final grade. Project demonstrations with take place **24-25 October 2019**. Team roles must be submitted to Platypus before your team demonstrates.

Teams may **only demonstrate <u>once</u>**.

Individuals will receive the team mark, scaled by their peer assessment factor. Peer assessment factor forms must be submitted prior to the demonstration.

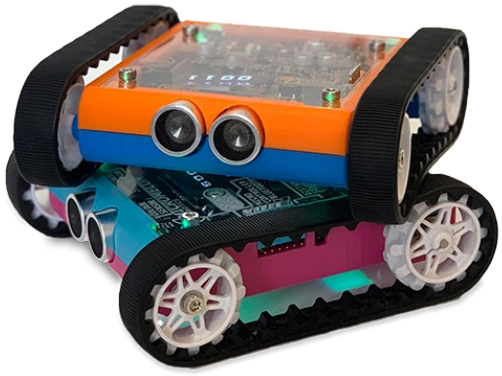
Environment

<u>The Course:</u> The course will be approximately 2m long and 2m wide. There will be two flags placed at each end of the course.

The Flags: The flags will look will be QR codes, see below. They will be mounted on a flag pole approximately 15cm off the ground (to bottom of flag).
(printable version at http://robotics.itee.uq.edu.au/~metr4202/tpl/flag.pdf)

Note: You are permitted to use existing code for locating QR codes in an image.

The Robot: The robot used in this project will be the Micromellon Rover, (https://www.micromelon.com.au/). The rover comes with a camera mounted on top of it. It also has 3 range sensors, mounted on the front and sides, and an IMU. You are free to use any of the sensors in any way you see fit for this project.



Programming Language:
Python will be the preferred programming language for this project as this is the programming environment used on the Micromelon rover. You will still be able to use Matlab.

Obstacles: Coca-Cola cans will be used as obstacles in some tasks, as specified in the task descriptions.

Objective: The rover will be placed in front of one flag, facing the flag at the other end of the field (except for the difficult level where it might not face the flag in the opposite end). The object is to drive the robot across the field, "capture" the opposing flag and then returning to the starting position.

- A flag is "captured" by touching its flag pole with any part of the rover. .
- A rover has returned when is touches the flag at its starting position.

Perception Pipeline

The classic autonomous robotics pipeline is: **Plan—Sense—Control—Act**
More than sensing, the perception is an extensive "domain" of robotics in its own right and is a key part of modelling the problem and the overall system's operation.

In particular for this project, consider the following tasks:

- Image acquisition and capture.
- QR detection (here you are permitted to find and existing code for locating QR codes in an image)
- Use any image features/keypoints? If so how to detect them.
- How (and if) to use Range sensor readings.
- Detection/Segmentation of any obstacles
- Motion Planning.

Rules of Play

Each team will be allocated **20 minutes** to demonstrate the functionality of their robot vehicle. Hardware cannot be touched or moved by a team member once the demonstration commences.

Prior to the commencement of each run, the robot must be positioned in a standard 'start position'. This position must be the same for every run and task. The obstacles will then be placed on the workspace. Runs will be timed from the point that the code is commenced, to the point that the robot returns to its 'start position'.

Teams must run the same code for each task.

The success of a run is quantified by:

1. Speed (time taken to solve and perform the task)
2. The number of collisions with obstacles.

A run is partially successful if the flag is captured but not returned.

The run is considered a *fail* and the robot must start again if:

1. An obstacle is moved more than 2cm as a result of a collision
2. The team decides they wish to reset

There is no penalty for a failure on a single run, other than demonstration time wasted. Teams have **three** lives (three strikes and you're out). A team may continue until they run out of lives (*fail* three times) or run out of demonstration time. At this point, the demonstration is complete and the team's demonstration grade is determined.

Task Descriptions

Task 1:

Your robot must perform the task of **capturing the flag at the opposite end of the field and returning home.** A number (depending of difficulty level) of known obstacles (coca-cola cans) will be placed randomly on the course.

	Easy	**Intermediate**	**Difficult**
Number of Obstacles	No obstacles	5 obstacles	10 obstacles
Maximum Marks	30	30	20

Teams will be marked based on their execution, motion path and speed.

Extra Credits

Extra credits will be awarded to the 3 fastest teams on the intermediate level.
1st – 10 marks
2nd - 3rd – 5 marks

Task 2:

Similar to Task 1, your robot must capture the flag at the opposite end of the field and returning home. The difference is that now the obstacle type is not known anymore. They can be anything books, a shoe, a sandwich...

Marks available: 10

Question and Answer:

Team members will be asked questions about the design and implementation. All team members need to understand and be able to explain all aspects of the design. The team will receive a single, group mark for the Q&A that represents the accuracy and clarity of the responses given. This mark is at the tutor's discretion.

Marks available: 10

Caveats

Some general "reasonable person" rules apply to the code and its execution:

- It is expected that you will use source/version control
- Codes with fixed (predetermined) estimates are not valid (even if the value is correct) -- read "NO HARD CODING"
- Internet access may or may not be present -- the code should assume that it will not have Internet access during execution and thus operate in a self-contained manner. This proviso excludes UQ license servers that may be needed by the program (e.g., Matlab). A "Mechanical Turk" or "phone home" solution is explicitly disallowed.
- Memory space may or may not be cleared between challenges and submissions -- The memory space might be cleared before each function. Thus, if your routines rely on parameters to be exchanged, it should do so by writing to a file. Similarly, if certain variables names (e.g., counters) are used between functions, then be sure to initialize them correctly.
- All source code(s) may be assessed -- Thus, it is requested that it is commented. If custom precompiled codes are used (e.g., mex files), the source code and compilation instructions (e.g., makefiles) should also be submitted.
- Computational and memory resources -- the functions should be able to operate reasonably on a "standard" Laptop/Workstation class computer (such as the UQ EAIT PC Workstations). Judges may terminate execution after 5 minutes.