"Exploration of Unknown Environments"

COMP0037 Assignment 2
Simon Julier (s.julier@ucl.ac.uk), Dan Butters (daniel.butters.16@ucl.ac.uk), Julius Sustarevas (julius.sustarevas.16@ucl.ac.uk)
Version: 25st February, 2019

Overview

Assignment Release Date: Tuesday 25th February, 2019
Assignment Submission Date: 12:00 Friday 17th March, 2019
Weighting: 30% of module total
Final Submission Format: each group submits *three* things - a zip file (which contains the source code implemented to tackle this coursework), a separate report in PDF format and a video file. The name of the zip file will be `COMP0037_CW2_GROUP_n.zip`, where n is the name of your group. Similarly, the name of the PDF will be `COMP0037_CW2_GROUP_n.pdf`. Finally, a video will be submitted with the name `COMP0037_CW2_GROUP_n.[ext]`. You may use a variety of video formats, but the video must be playable by VLC.

Assignment Description

In this assignment you will investigate how a mobile robot explores an unknown environment. In particular, we will look at the factory-like environment encountered so far. We assume that the robot operates in two phases. In the first phase, the robot constructs a static model of the environment. This is used in the second phase to carry out a set of tasks. This coursework concentrates on the first phase.

There is an unmarked familiarization phase which is to get you familiar with the code.

This coursework has five main parts:

1. Implement a reactive planner to allow the robot to compensate for uncertainty as it completes its mission. (80 marks)
2. Implement an exploration system which will attempt to explore an environment. (260 marks)
3. See what happens when you put parts 1 and 2 together. (20 marks)
4. Explore the properties of the information content of a map for information-theoretic path planning. (80 marks)
5. A video describing your solution.

As explained below in the materials section, you are provided with reference code which implements some of the techniques in the lecture. These include a mapping system, a (deliberately very inefficient) exploration system, and the hooks to show where reactive planning will be implemented. The code also includes a reference implementation of a Dijkstra planner.

The results of the first three parts will be presented in a single report. This will be divided into three sections according to the assignment above. The rubric gives the allocation of marks within each part separately. **The report should be no more than 15 pages, including all diagrams and appendices.**

Please note that the video is a check to reduce the prevalence of contract plagiarism. It *must* be submitted and must obey the minimum requirements which are described below. **Failure to do so means the marks from the coursework will be assigned a value of zero.**

In the breakdown below, we assign "marks" to each question.

Part 0: Familiarization - Mapping System (Unmarked)

This is a familiarization exercise to get you up and running with the new system. It is also an opportunity for you to finetune the system, and to bring in your code from CW1 and use it if you wish.
Follow the instructions in "Materials Description" to download and install the code to support CW2.

Run:

```
roslaunch comp0037_cw2 mapping_prespecified_waypoints.launch
```

This will launch the planning and mapping system using a known search map of the environment, and a set of waypoints that the robot will drive to in order to full model the environment. For path planning, the robot uses my implementation of Dijkstra.

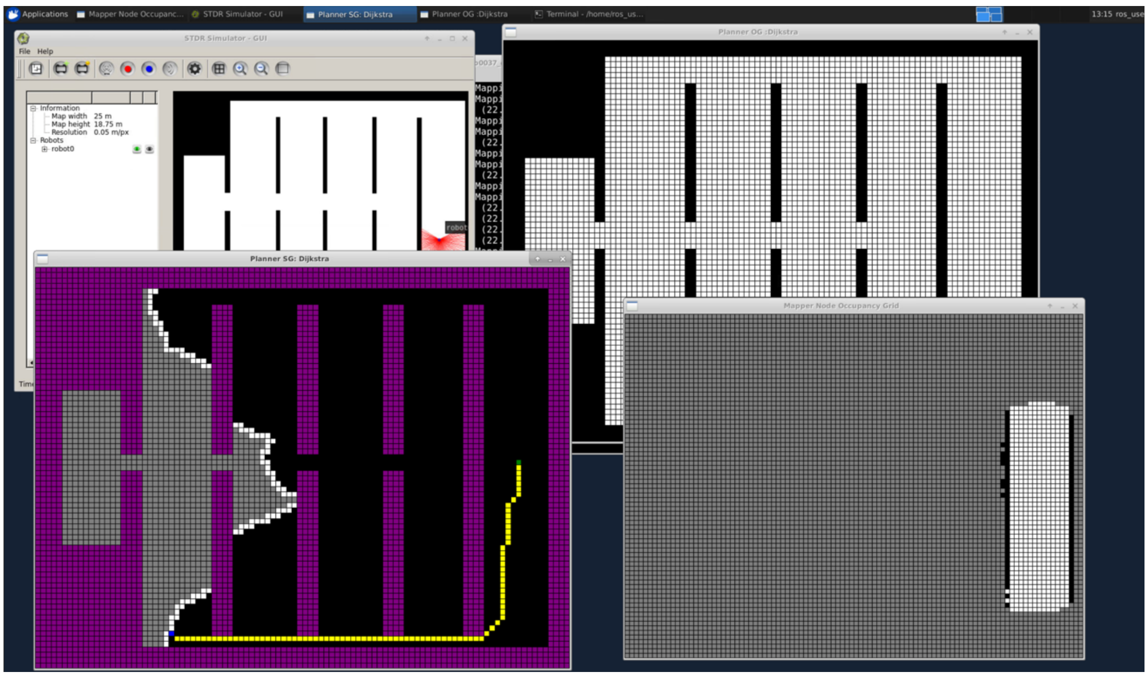

Figure 1: Docker output for familiarization task.

The code will open four windows shown in Figure 1.

1. *Planner OG:* This is the occupancy grid (OG) the planner has available. Recall that in the OG each cell stores the probability that the cell is occupied. 0 is empty, 1 is blocked and an intermediate value means “not sure”. In this familiarization task, this is exactly the structure of the environment. Therefore, the cell probabilities are either 0 or 1. We relax this assumption later
2. Planner SG: This is the search grid (SG) the planner uses. We create this from the OG using the thresholding strategy described in the lectures. Once this grid has been created, the planning algorithms (A*, etc.) use it in planning.
3. Mapper Node Occupancy Grid: This is the view of the world the mapping system builds. Initially it is all empty (grey) but will fill in as the robot moves with cells. You will see artefacts on occasion which are caused by things such as numerical rounding issues and timing mismatches due to networking. These should not cause any substantial difficulty for the coursework.
4. STDRGUI.

The robot will drive to a set of prespecified waypoints, causing the mapping system to update as it goes.

Optional tasks you can look at. Note these can be carried out at any time and you might want to look at some of them (e.g., 2. and 3.) after you have tackled other parts of the coursework first.

1. You should be able to take your A* implementation from CW1 and use it directly in the code. See the file `comp0037_reactive_planner_controller/scripts/planner_controller_node.py`. The code uses my (not very good) implementation of Dijkstra.
2. You should be able to look at your robot control strategy and see about porting it over. The main thing to note is that, unlike CW1, for CW2 there are additional API hooks to stop the robot driving if a collision is detected. If you compare

the code in `comp0037_reactive_planner_controller/src/controller_base.py` and `comp0037_reactive_planner_controller/src/ move2goal_controller.py`, you should able to see what changes are needed.

3. Different robot speeds can affect the quality of the mapping. Explore the effect of vehicle drive speed on the quality of map used, and identify a suitable choice. One way to do this would be to use the keyboard-based method to enter robot odometry in `comp0037_example`. If you choose to do so, note that it has not been modified to account for any changes required to make the mapper run correctly.

Part 1: Implement Reactive Planner (80 marks)

The goal is to implement a reactive planner for the robot which uses feedback from the mapping system to validate if the current planned path can be successfully executed and the goal can be reached. If the plan cannot be executed, the planner should be run to create a new path. (Because of the complexity, you do not need to implement a local planner).

For testing, two launch scripts are available to launch the reactive planner:

```
roslaunch comp0037_cw2 part_1_1_reactive_planner.launch
roslaunch comp0037_cw2 part_1_2_reactive_planner.launch
```

The first sends a set of goals the robot should be able to reach, the second is a single goal the robot *cannot* reach.

- Describe what is meant *by a reactive planning system*. Your description should include a block diagram which explains the main components. **(10 marks)**

- Implement your reactive planning system. Describe your implementation. Present results on the two launch scripts above. Your results should include the maps constructed at the point where the reactive system is finished. **(60 marks)**

- Reactive planners tend to be very inefficient in their operation, both in terms of planning initially where to move, and the computational costs associated with replanning. Several approaches for improving performance – including more efficient global planners and the use of local planners – were discussed in the lectures. Describe one approach for improving the performance. You do not have to implement the algorithms. **(10 marks)**

Part 2: Implement Frontier-Based Exploration System (260 marks)

The goal of the first part of the coursework is to describe and implement a frontier-based planning system to guide the robot to fully explore a factory space. The exploration system replaces the boss component by an explorer which dynamically creates the waypoints. The exploration system should keep running until all it has mapped all reachable parts of the map.

In the first part of this coursework, the planner controller has full knowledge of the map and so, when presented with a candidate waypoint, it can directly determine if the robot can reach the planned waypoint.

To run the reference system, use the launch file:

```
roslaunch comp0037_cw2 part_2_exploration_mapping.launch
```

This includes a (very inefficient) exploration algorithm. Explain how this reference algorithm works. Characterise the performance of the existing exploration system in terms of total distance travelled, total angle turned, total time for completion (use ROS time) and number of waypoints that have to be generated.

- Define what is meant by a *frontier*. Explain how it used in exploration, and describe two methods for identifying frontiers, and two heuristics for choosing which heuristic . **(40 marks)**

- Explain how the frontier-based provided by us in the code works. Evaluate its performance and discuss where the limitations lie. To evaluate the performance, you should consider: the total time required to explore the map and the coverage (the percentage of cells which were detected by the exploration system). **(20 marks)**

- Choose one frontier detection algorithm and one heuristic for choosing the next point. Provide a justification of your choice. Implement your choice. Explain your implementation. Present the results in terms of the total time required to explore the and coverage**. (200 marks)**

Part 3: Putting it All Together (20 marks)

In this part, you will run your exploration algorithm with your reactive planning algorithm. Unlike part 2, where the planner had knowledge of the complete map and could validate whether a goal is reachable or not, the system here will only have access to the map which is being built at run time. Note that tuning a system can take a long time; the small number of marks here is because you should just run the system directly and see what it does.

```
roslaunch comp0037_cw2 part_3_reactive_exploration_mapping.launch
```

- Present results on the launch scripts above. Your results should include the maps constructed at the point where the reactive system is finished as well as information about coverage. It is possible that the system will not completely map the environment. If so, provide an explanation of what you believe the cause might be. **(20 marks)**

Part 4: Information-Theoretic Path Planning (80 marks)

Information Theoretic approaches pose exploration as an optimization problem. Implementing the full method is fairly difficult and *you will not need* to implement the approach here.

- Describe the information theoretic approach to path planning and some of the approaches used to implement it. Your description should include a definition of what the cost measure is and the mathematical formulation of the approach. **(20 marks).**

- Modify the mapping node that it computes the entropy of the map roughly once every 5s of simulation time. Description the equations you used to compute the entropy, and your implementation. **(20 marks)**

- Evaluate the performance of the base line exploration algorithm and another exploration algorithm by plotting how the entropy changes over time for different algorithms. Your comparison should be presented on the same graph. Discuss your results and suggest which algorithm you think might be most effective. (**40 marks)**

Part 5: Video Submission

Submitting a video is mandatory and must obey the following instructions. If you do not do this, then you will receive no marks for this coursework.

In your video, you will discuss your solution for each part you attempted for the coursework above. Each video will feature all the members of the team. Each member of the team will be expected to speak for at least a minute, and the overall video should run between 3 and 10 minutes. (Please note that there is *no* assumption that longer is better. Rather, producing a very compact short video can take a very long amount of time, which is not the intent of this task.)

The video should include the following:

- An introduction to the team. Each team member must introduce themselves on camera.
- For each part of the coursework attempted, explain your solution. You do not need to produced a polished presentation. It is sufficient to highlight parts of the report or code using a mouse cursor and to discuss the presented text. Simply reading out the text of the solution is not sufficient. Instead, you should discuss your solution and identify things such as where the challenges lie.
- For each part of the coursework not attempted, provide a brief explanation of why this was the case.

For the group, video taken with a handheld camera (such as a mobile phone) with the built-in microphone is sufficient. However, please ensure that video and audio quality is clear. For capturing solutions / code there are a variety of methods. My colleagues recommend Snagit (https://www.techsmith.com/screen-capture.html) because it offers a free month-long evaluation period, it is easy to switch between webcam and screen recording, and includes some minimal video editing. Please have a look at the tutorial via https://www.techsmith.com/tutorial-snagit-how-to-document-process-video.html#transcript.

There are a variety of ways to encode videos. There are only two technical requirements. First, the video must be playable by VLC 3.0.5 or later (https://www.videolan.org/vlc/releases/3.0.5.html). (VLC is supported on Linux, Mac and Windows. It is typically plays a much wider range of media than, say, the Windows Media Player or the QuickTime Player.) **Second, the video must be less than 160M in size to permit upload to Moodle.**

Materials Description

All the software provided to support this module is on github. This consist of a catkin workspace which you will initialize and run on your machine.

Installation

The code is in the branch `cw2`. The safest approach is to create a new `catkin_ws` and run:

```
cd <your_cw2_catkin_wt clone https://github.com/UCL/comp0037.git
cd comp0037
git checkout cw2
```

If this was successful, you should see:

```
Switched to a new branch 'cw2'

ls -l
-rw-r--r--   1 ucacsjj  staff  1512 19 Feb 08:19 LICENSE
drwxr-xr-x   6 ucacsjj  staff   192 19 Feb 08:20 comp0037_cw2
drwxr-xr-x   5 ucacsjj  staff   160 19 Feb 08:20 comp0037_example
drwxr-xr-x   8 ucacsjj  staff   256 19 Feb 08:20 comp0037_explorer
drwxr-xr-x   5 ucacsjj  staff   160 19 Feb 08:20 comp0037_launch
drwxr-xr-x  10 ucacsjj  staff   320 19 Feb 08:20 comp0037_mapper
drwxr-xr-x   8 ucacsjj  staff   256 19 Feb 08:20 comp0037_reactive_planner_controller
drwxr-xr-x   8 ucacsjj  staff   256 19 Feb 08:20 comp0037_resources
drwxr-xr-x   5 ucacsjj  staff   160 19 Feb 08:20 comp0037_the_boss
drwxr-xr-x   5 ucacsjj  staff   160 19 Feb 08:20 comp0037_time_server
drwxr-xr-x  13 ucacsjj  staff   416 19 Feb 08:19 stdr_simulator
```

You should then be able to do:

```
cd ../..
catkin_make
source ./devel/setup.bash
```

The code only runs in full ROS mode.

Detailed Description

The software consists of the following packages. Packages in italics are unchanged from coursework 1.

- `comp0037_cw2`: This package contains launch files, which are used to automate starting up ROS nodes, together with the scenarios (maps + lists of goals).
- *`comp0037_example`: This is the move the robot example you encountered in Lab Exercise 02.*

- `comp0037_explorer`: This module will be responsible for running the exploration phase of the robot. It will be responsible for identifying frontiers, planning, etc.
- *`comp0037_launch`: Example launch scripts for starting up stdr.*
- `comp0037_mapper`: This module runs the mapping algorithm which takes odometry and other data and produces new updated maps of the environment. It issues two types of maps: the full updated occupancy grid, and the delta occupancy grid, which only contains non-zero entries where the occupancy grid cell values have changed.
- `comp0037_reactive_planner_controller`: This forms a similar function to the planner controller from course work 1. However, it has been modified to provide hooks to support real-time feedback.
- *`comp0037_resources`: These are resources such as simple rooms.*
- *`comp0037_the_boss`: Sends waypoints for the robot to drive to from a prescribed list.*
- *`comp0037_time_server`: This can be used to "speed up" and "slow down" the simulation time. By default it runs at real time (1s simulation = 1s real world time).*
- *`stdr_simulator`: This is a locally modified version of the stdr simulator which fixes a couple of bugs.*

Removed packages:

Note the following two have been removed:

- `comp0037_cw1`: This package contains launch files, which are used to automate starting up ROS nodes, together with the scenarios (maps + lists of goals). It is not needed for cw2.
- `comp0037_planner_controller`: The reactive planner controller introduces many small changes and trying to maintain full backwards compatibility was too difficult.

You will mainly be working with `comp0037_reactive_planner_controller` for Part 1, `comp0037_explorer` for Parts 2 and 3.

Getting Help

You are encouraged to use the assignment discussion forum to help when you get stuck. Please check the forum regularly and try and answer each other's questions. I will be checking the forum as often as I can and will be answering any questions that have remained unanswered. Please note that if you have questions about coursework, please post them to the forum directly rather than emailing me. The reason is that all students will be able to see the reply.

Submission Format and Structure

As noted above, each group will submit three things:

- Your copy of the ROS workspace you used to tackle this coursework. This is a zip file which contains This should be the `catkin_ws/src` directory. Please *do not* include the `build` or `devel` subdirectories – these include very large intermediate files which can only be used on your local machine. Please name this file `COMP0037_CW2_GROUP_n.zip`, where `n` is the code letter of your group.
- Your report which describes your analysis and results. This will be have the name `COMP0037_CW2_GROUP_n.pdf`.
- Your video. This will be submitted with the name `COMP0037_CW2_GROUP_n.[ext]`, where `ext` depends on what format / software you used to create your video.

Please note that, if you do not conform to the submission guideline, a 10% penalty on your overall mark for the coursework will be imposed. This includes not using the requested filename format.

Marking Guidelines

All reports will be marked against the marking rubric, which is downloadable from the course's Moodle page. The mark weighting for each section is as follows:

1. Algorithms Description (30%)
   How well are the different algorithms described and contrasted?

2. Implementation, Analysis and Presentation of Results (40%)
   How thorough is the analysis and discussion of the results? How well do they compare between the different algorithms and with properties of those algorithms? Are the algorithms well-supported by quantitative evidence?

3. Format, structure, referencing and clarity of writing (10%)
   Is your report well laid out and does the write-up follow a clear structure? Have you included any references to show background research/reading? Is your writing free from spelling, punctuation, and grammatical errors?

4. Challenge Problems (20%)
   Additional marks for the challenges – 10% on description, 5% on analysis and 5% on performance gains.

Submission and Feedback

The deadline for submission is **12:00 (midday) on Tuesday 17th March, 2020**. As noted in the lecture, late penalties will start to accrue if any element of the coursework has a receipt time of 12:01 or later on Tuesday 17th March. Therefore, please do not leave submission until the last minute.

We will release a video on **Tuesday, 24th March** (after the five day limit for late submission) which will describe our model solutions and expected results.