COMP3631 Project

You are asked to implement a program that controls a simulated Turtlebot to find and identify a Cluedo character in an environment. In the environment, there will be two rooms. Your robot needs to enter the "green room", which has a green circle near its entrance, and identify the character in the room. The second room will be a "red room", with a red circle near its entrance. Your robot should **not** go into this room. **You will work on this project as a group.**

Your program will be tested using the Turtlebot Gazebo simulation in several different worlds. Your program will be given a map of the environment (you will learn about what a map is, how to build it and how to use it, in Lab Session 4). Your robot will be placed at a start point, which will be the same for all groups. You will be given (x,y) coordinates of the entrance points of the two rooms in the map.

One room will have a red circle on the wall near its entrance, and the other a green circle. The green/red circles on the walls will be visible from these entrance points, but not necessarily from a direct angle. (You might or might not need to move your robot around the entrance points to have a better view of the circles. You are recommended to experiment with different positions of the circles and robot, build a robust program, and report your findings in your report).

Your robot will need to enter the room with the green circle on the door. You will be given the (x,y) coordinates of the center points of both rooms. There will be a Cluedo character in the green room and a Cluedo character in the red room. We know that the "murderer" is the Cluedo character in the green room, **not** the one in the red room. We just need the robot to go into the green room and tell us who s/he is. Your robot, therefore, will have to find the Cluedo character in the green room and report the identity of the character. In your group's GitLab repo, we will provide you with a set of images of different Cluedo characters and their names. Your robot will need to identify which one is in the green room.

We will also provide you with an example Gazebo environment, and associated map, and an example input file (Please see the associated file **ExampleWorld.pdf** to see how to use the example world). These will be just examples; the actual shape of the environment, shape of the rooms, exact size/position of the green/red circles, and the position/identity of the Cluedo character may change. Your program should be robust to such changes.

This project has two components:

- The Python program
- The Written Group Report

In total, this project corresponds to 40% of your module grade. 10% of this will come from the Python program and 30% will be based on your group written report. Details of the Python Program and the written report are below.

Python Program

Deadline: 17:00 on Friday, 29 April 2022. Your GitLab group repo will be locked past this deadline, and you will ***not*** be able to push any new code/changes, therefore, make sure to push your latest code before the deadline.

Submission: You will write a Python program to perform the task, which should be pushed to the GitLab repo of your group by the above deadline.

You can collect 10 points in total, according to the following rules.

- Your program will be tested over 5 different worlds, with different difficulties.
    - 2 worlds of easy difficulty.
    - 2 worlds of moderate difficulty.
    - 1 world of hard difficulty.
- For each world, you can gain at most 2 points:
    - **Character screenshot (1 point):** When your robot thinks it saw the image of the Cluedo character, it should save a snapshot of the camera image with the filename "cluedo_character.png". The character must be completely contained within the saved image. If an image with this name is saved and it does not show the correct character from the green room, you will get -1 penalty point (including if you take a screenshot of the wrong character in the red room or take a screenshot of an empty wall). Please make sure that you use "cluedo_character.png" as the filename.
    - **Character identification (1 point):** Your program must then identify the correct character in the green room, by printing out the character name into a text file with the filename "cluedo_character.txt". If a file with this name is created, but includes a wrong character name, you will get -1 penalty point. Please make sure that you use "cluedo_character.txt" as the filename.

Testing procedure: During the test, we will use your group's code that has been submitted to your git repository before the deadline. Your program should ***strictly*** run according to the following process:

- We should be able to run your complete solution with a *single* python file, named *main.py*. We should not be required to run anything else or a Python program with a different name.
- Your program should run *without* requiring us to pass any parameters to it.
- Your solution should work within the ros.simg Singularity image. We should **not** be required to install any system-wide software or upgrade any ROS component.
    - The only exception is Python libraries installed as a user. These are commands like *pip install --user some_library*. Please note, **not** system-wide Python libraries, which require root permissions (for example **not**: sudo pip install some_library or sudo apt-get install some_library)!
    - If you require the installation of third-party Python libraries, you should include a requirements.txt file at the root of your repository. You can get one by running pip freeze > $HOME/catkin_ws/src/requirements.txt (running this command within Singularity). We should be able to install all the dependencies using that requirements.txt file (using pip install –r requirements.txt). You do **not** need a requirements.txt file if you do **not** use any third-party Python libraries.
    - If your solution fails to run out-of-the-box and requires modifications to work (for example due to an error with third-party library or because of a hard-coded path) on our machine, you will get –3 penalty points from your overall demo grade (so the max you could get after the changes is 7).

    - We highly recommend testing your final code on multiple different machines of different group members. If it works in the Singularity image without any changes on one of your fellow member's computer, you will be more confident that it will work on ours as well.
- Your solution should work assuming the following evaluation process:
    - We will run "roslaunch turtlebot_gazebo turtlebot_world.launch" with a world file that we have created beforehand.
    - We will run "roslaunch simulated_localisation.launch map_file:=..." with a map file that we have created beforehand.
    - We will run "roslaunch turtlebot_rviz_launchers view_navigation.launch" to start RViz.
    - We will provide a "2D pose estimate" to localise the robot.
    - We will move the robot around to better localise it in the world.
    - We will put a new “input_points.yaml” for each world under $HOME/catkin_ws/src/group_project/world . You are expected to read the file from there to obtain the room entrance and room center points for each world. You should not expect us to put this file elsewhere.
    - We will run “rosrun group_project main.py”
        - Remember to make executable the main.py script using *chmod +x main.py*
    - We will mark your program according to the tasks your robot is able to perform as described above.
- You will **not** be able to run different scripts/programs at different stages of the test. For example, please do not think that you can run one program to go to the centre of the green room, and there you can run a new different script. This is not allowed. We will run your main.py file and it should complete the task.
- Be careful with using computer-dependent code. For example, using file paths that are absolute and work only on your computer. You should employ the right Python code to read files agnostic to the computer the script is running on.
- Your robot will have **at most 5 minutes per world** to complete the task. If, after 5 minutes of running, your program has not stopped by itself, it will be stopped and the points you have collected up to that point in that run will be your mark for that world.

It is your responsibility to ensure before the deadline that your solution works exactly as described above. We suggest that you go through these commands before the deadline and make sure that your solution is compatible.

The minimum you can get from the total of 5 worlds is 0 (zero) points; in other words, you cannot go negative due to penalties.

Written Group Report

Deadline: 17:00 on Friday, 29 April 2022.

Submission: The report is to be submitted electronically in the VLE as a PDF file. Only one member of a group should submit this group report. (Please see below for the additional individual report, which should be submitted by every member of the group separately.) All code should be submitted into the group’s GitLab repo.

Content: Write up your solution as a group, as if it was a report to a client. This should be **no more than 10 sides**. In particular;

- Include details of the design options you considered and justification of why you chose the particular options you did.
- Describe how you have tested your solution. Give examples of different environments/maps you have created to test your program, and the performance of your program in the environments we have provided.
- Include in your report **images**, **a link to a video** and **data** to demonstrate how your solution works or fails. Outline and discuss the limitations of your proposed approach. Suggest scenarios where it might not work.
- State any OpenCV/ROS codes you have used that are not part of the standard distribution.

Markscheme:

Design (16 points): Marks will be awarded for:

- Justification of decisions and general knowledge of possible methods
- Is the design structured well? Are different sub-tasks identified well? Are all of them identified? Are they integrated well?
- Was efficiency in mind during the design?
- Was robustness in mind during the design? Likelihood of working in a wide range of environments and images (other than those provided)?
- How novel is the design?

Implementation and Results (6 points): Marks will be awarded for:

- Efficiency/accuracy of reaching to the rooms, use of planning and search methods.
- Accuracy of identification of the room colors and cluedo character.
- Testing and analysis of performance, whether successful or unsuccessful.
- Use of concrete evidence (numbers, figures, tables, and diagrams) to evaluate performance.
- Video/image examples of performance.

Real robot test (4 points): Marks will be awarded if you test your program on a real Turtlebot:

- Describing what needed to be done to make the program work on the real robot.
- Describing how the robot performed in the real world.
- A link to a video showing a real robot attempt.

Writeup (4 points): Marks will be awarded for:

- Clarity of presentation of solution and results [N.B. Large chunks of code with no explanation are unlikely to gain high marks!]
- Discussion of the strengths and weaknesses of the system presented
- Presentation
- Use references to credit the resources you used, if any.

We expect *all* members of a group to contribute to the solution and attend group meetings. We reserve the right to investigate the contributions of a student to the project solution and the report and adjust individual's marks according to his/her contribution. If you think that a member of your team is not contributing to the solution or/and to the report, please contact us.