Robotics
Practical 6: Navigation Challenge

Andrew Davison
ajd@doc.ic.ac.uk

1 Introduction

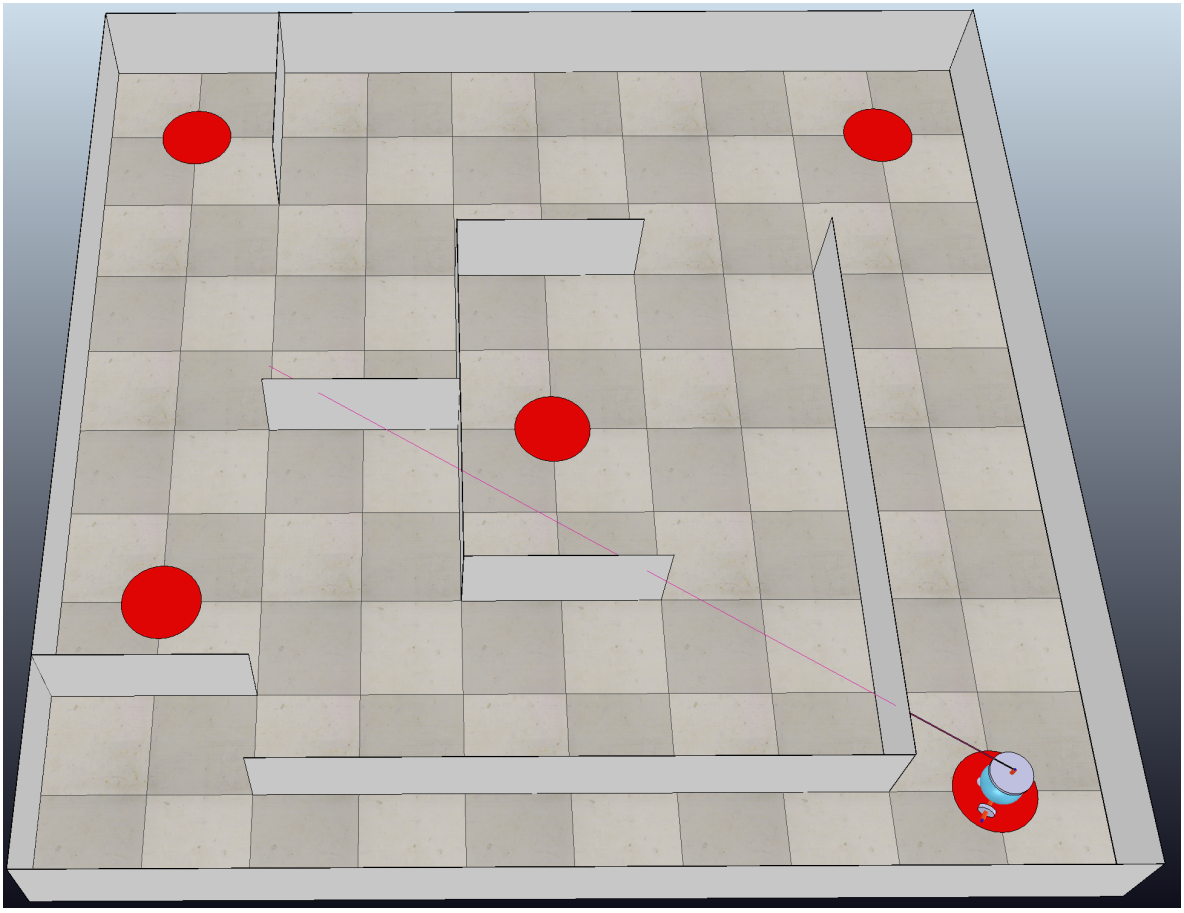
The final practical of term consists of an easy to state navigation challenge, but I will not be giving you specific guidance on how to achieve it. You should use what you have learned in the previous lectures and practicals (or elsewheree) to design robots and algorithms which you believe will be most effective. In this challenge, accuracy and speed are both important. Importantly, we will give points this week purely depending on the performance of your robot based on a tough marking scheme.

This practical is UNASSESSED and will not count towards your coursework mark for robotics, but we will test out the robots in the form of a competition in the final session of term, on **Friday March 4th**. All groups that have achieved what they think is a good solution to the challenge can take part in the competition by submitting a scene file to us before the practical session. **I will confirm next week how you should submit your files.** I and my TAs will then run the submitted scene files one by one while screensharing and we can all watch and judge how well the different groups perform. There will be prizes for the winning groups in three categories: best performance with the standard robot; best performance with a modified robot; and best creativity (more details below).

Since there are no formal assessment marks for this practical, it is up to each group how much effort you put into it, and it is OK if you decide not to take part in the competition. As ever, working on the practical should be very good for strengthening your knowledge of the material in the course before the exam at the end of term, and hopefully taking part in the competition will also be fun!

2 The Goal

Download and run the CoppeliaSim scene file for this practical: `https://www.doc.ic.ac.uk/~ajd/Robotics/RoboticsResources/practical6.ttt` You will see the scene in the picture, which features the **same arena** which we used for the MCL practical (with the same wall positions, and randomly generated bumpy floor), but with some extra features.



First, you will notice the five red disks, which are **GOAL** locations for the robot, for which you are given the coordinates in the scene file.

The robot in the scene is the same as we used for MCL, but you will see that each time you run the scene file it is randomly placed at one of the goal locations, and at a random orientation.

The challenge is to program your robot to achieve the following:

1. Your robot will be placed randomly by the program in one of the five marked goal positions. The robot will be placed precisely in terms of (x, y) at the chosen goal, but at a **random orientation**. Timing will start when your program starts running.

2. The robot needs to work out where it is, then navigate as quickly and accurately as possible to all four of the other four marked goals in any order, and finally to return to the goal it was originally placed at. Note that the fact that the robot must return to where it started means that there is no obvious advantage to starting in any particular goal position.

3. Upon arriving at each of the four other positions, the robot should call the `reachedGoal()` function we have defined, with argument of the index of the goal position (1 to 5) that it thinks it has reached, and it will receive a score for how accurately it has reached that goal.

4. After reaching the four other goals, the robot should return to the goal it started at, and call `reachedGoal()` a final time here. If the robot has successfully reached all of the goals and returned, the timing clock will stop.

5. The orientation of the robot when it reaches goals (including the final one) is not important, only its (x, y) position.

6. A goal will be judged as achieved if the robot's centre is less than 20cm from the goal position, and more credit will be given for stopping within 10cm or 5cm of the goal (see the marking scheme below).

7. If your robot manages to achieve every goal, additional points will be given for its overall time, from program start to when the robot finally returns to the goal it started from.

3 Judging and Scoring

Points will be assigned as follows out of a maximum of 20, based on the run that your robot achieves.

1. Up to **3 points per goal** for stopping accurately at each of the 5 goals: **3 points** for an error of less than 5cm, **2 points** for an error of less than 10cm, and **1 point** for less than 20cm. These errors are judged automatically by the `reachedGoal()` function.

2. Up to **5 points** for the total simulation time taken (from starting the program until the robot returns to its original position and stops). These will be awarded *only if the robot gains at least one point at every one of the five goals for stopping correctly*.

Total time:	0–60 seconds	**5 points**
	60–90 seconds	**4 points**
	90–120 seconds	**3 points**
	120–180 seconds	**2 points**
	180–240 seconds	**1 points**
	slower	**0 points**

This scoring scheme is **tough** but should allow the best teams to shine through.

The winner of the challenge will be the robot which obtains the most points, with a tiebreak on the same number of points being decided by the quickest time.

4 Details

Choice of algorithms is up to you, though the most obvious way to proceed is a combination of a global localisation method (either learned place recognition or perhaps an MCL-based method with suitable particle initialisation) with then normal continuous MCL for localisation as the robot navigates around waypoints you define in the scene. You will presumably have to define a number of waypoints which lie between the goal positions so that the robot can move between them without hitting walls.

You are free to use any other methods you think of, and wall following is another one that might prove useful. In this practical only we will not judge the choice of methods, but simply the performance of the robots, hoping to reward those groups who have used their knowledge to make robots that work really well in practice.

You have the choice of tackling the main challenge in two ways, and we will offer a separate prize for the winning robot in each of these two class:

4.1 Standard Robot Class

One competition will be based on using the standard robot provided which is the same as we used in the MCL practical. You can use the turret controlled by the third motor to point the sonar sensor in different directions as we did in the wall following practical (for instance for scanning to obtain a signature of a location, or for better performance in MCL).

You should keep the 0.1m standard deviation noise which is added to all distance sensor readings which make it behave like a realistic sonar.

Also you should limit the rotational velocity demand you send to any motor (either the driving motor or the turret) to 10 radians/second.

You should keep the simulation running with the standard dt value of 50ms.

Probably your robot will stop at the goals to call the `reachedGoal` function, but this is not required and in principle you could call it while still moving. Note however that you can only call the function one time for each goal so the first time you try will have to count.

All of your control program's interactions with the simulator should be via sensing and actuation commands to read from sensors and set commands to motors, as we have been doing throughout term. i.e. you should not call any 'magic' functions in the simulator API once your robot is running, for instance to read the robot's coordinates directly or to teleport the robot directly to a precise location. We will be able to see your code and check if we have suspicions!

4.2 Modified Robot Class

We will try a second competition between robots that have modified designs, and it will be interesting to see if teams can improve their performance much this way.

The general rules of the competition are the same. The robot must start in a random goal position and at a random orientation, and must navigate to the four other goals and back with the same points for accuracy and time; and navigate without using 'magic' functions. Also keep the simulation time at 50ms please.

The modifications allowed will be limited. You can add up to 10 sensors if you wish, though they should all be of the same 'ray' type as the sonar sensor we use on the usual robot with 0.1m standard deviation noise added to all measurements.

You can change the design and geometry of the robot if you wish, for instance with the goal of moving faster or more accurately, though of course the robot still needs to fit between the quite narrow gaps between walls.

You can use up to 6 motors if you want, but they should all be of the same type as in the main robot, and also move at a maximum rotational velocity of 10 radians/second.

Please get in touch if you have any questions about rules because we may not have thought of everything and can clarify during the week!

4.3 Best Creativity Prize

Besides the prizes for best standard and modified robots, we will have one final prize for what we judge to be the most creative robot submitted. This robot might be a particularly creative solution to the main task in either the standard or modified class, even if that robot does not have the best performance; or a robot which works well while not fitting the rules of either competition; or we could choose to award it

to a robot you come up with which is not trying to solve the challenge but is just doing something else which is interesting within CoppeliaSim.