Robotics

Practical 6: Obstacle Avoidance Challenge

Andrew Davison
a.davison@imperial.ac.uk

1 Introduction

The final practical of term consists of a challenge competition. This practical will be UNASSESSED, and does not form part of the assessed coursework for Robotics.

We will hold the competition in the final session of term, on Thursday March 7th. The competition will start soon after the start of the lab session at 11:00am. Groups will be called up one by one in box number order for their first attempt, then when all groups have had a turn there will be a second round where every group will have another attempt.

In each round, you must be ready to go when I call your team number or you will miss your turn! We will score both of your attempts, but only the better of the two will count for the competition. Prizes will be awarded for the best teams.

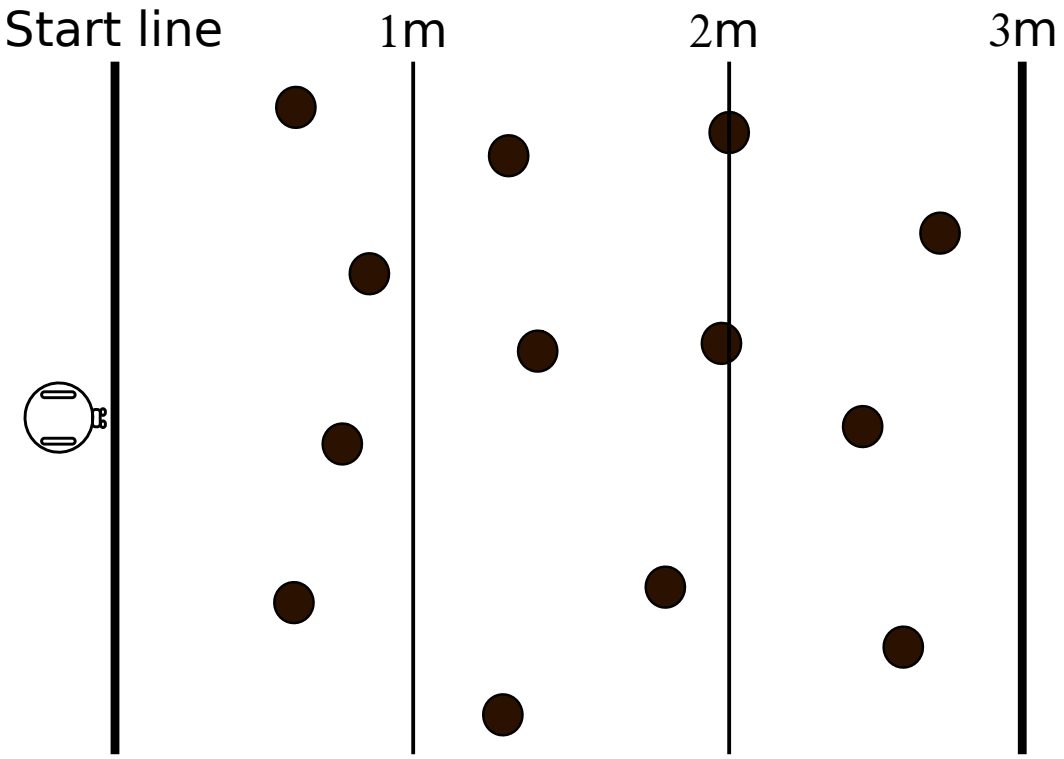
2 Camera Set-Up and Calibration

If you want to take part in the challenge and do not yet have a Raspberry Pi camera, come and see the TAs and they will give you one and help you to connect it to your kit. You then should follow the steps in the Practical 5 sheet, which has information and example code for capturing images from the camera and displaying them in the web interface, how to do some simple image processing, and how to calibrate your camera to make accurate metric measurements of the planar positions of objects in front of the robot.

https://www.doc.ic.ac.uk/~ajd/Robotics/RoboticsResources/questions5.pdf

3 The Goal

The aim of the challenge this year is to navigate across an area filled with obstacles. Your robot will start with no knowledge of the positions of the obstacles and will have to locate them with its camera and plan a path to advance as far and as quickly as possible.



The robot will start behind a start line marked on the floor, facing forwards, and there will be a parallel goal line 3m ahead. We will mark intermediate lines at 1m and 2m. The course will be at least 3m wide and the robot should not attempt to deliberately move outside of this width. The robot needs to advance all the way across the course without hitting any obstacles, and ideally do this as fast as possible

The obstacles we will use are the same red disks which we have given you for calibrating your robots' camera. On each run, disks will be placed by us randomly to fill the space in front of the robot. You can assume that the closest obstacles in front of the robot at the start will be at least 30cm away from the start line. There will be fairly wide gaps between the obstacles, with at least 40cm between any pair, which will be big enough for any of your robots to comfortably fit through, though narrower robots may have a small advantage.

The floor surface in this course is the standard lab **carpet**.

4 Methods

We will be judging purely the performance of the robots against the criteria in Section 5 and the choice of robot design and algorithms is up to you. However, here we will explain the most obvious way to approach the task.

We suggest a purely local planning approach similar to the Dynamic Window Approach. Back in Lecture 2 I demonstrated and provided the code for the implementation of this method in Python in a simulated environment in:
`http://www.doc.ic.ac.uk/~ajd/Robotics/RoboticsResources/planning.py`
This simulation is somewhat close to the competition setup, and simulates the robot having a forward looking sensor such that it only becomes aware of obstacles when they are in front of it and within a cone shaped regions, and then plans local motions to move towards a target location while avoiding obstacles. The robot moves by controlling its v_L and v_R left and right wheel velocities, so your real robot should also use velocity control and on each timestep adjust the velocity demands it sends to the two wheels. You should use `MAXVELOCITY` and `MAXACCELERATION` values which fit the range

within which you think you robot can move smoothly.

Note that there are several parts of the DWA planning simulation code that you will not need: especially everything to do with Pygame, simulating the random positions of landmarks, setting up the display, wraparound, etc. The important parts which you should use from the code which actually implement the DWA algorithms are the `predictPosition` and `calculateClosestObstacleDistance` functions and the main planning loop where the programme selects possible actions from `vLpossiblearray` and `vRpossiblearray`. Try not to get confused by the rest of the code and just strip it out.

One difference between that simulation and the challenge setup is that in the simulation the robot is trying to reach a specific (x, y) goal location, whereas in the challenge we have just asked the robot to move forward as far as possible. If the x coordinate of your frame is the initial forward direction, then the robot's 'benefit' term should try to maximise x increasing and you can modify the program to reflect this.

The other obvious difference is sensing, which must now become real rather than simulated, and I expect that you will use your camera, with a calibrated ground plane homography to obtain estimates of the locations of the obstacles the robot can see relative to its current position. Once an object has been observed once, its global coordinates can be estimated based on the camera measurement and the robot's position, and added to an obstacle list stored by the robot. I would assume that the obstacle list should be estimated in the world frame, so that the robot will need to transform all measurements into the world frame based on its current position estimate from odometry.

The robot needs to keep moving forward, and will not have a known map of the area to localise against, so I would assume that you will rely on odometry to keep a running estimate of the robot's location. Since the challenge is to keep moving forward, and the sideways position of the robot doesn't matter in the challenge, the need for accurate odometry is reduced, and the main aspect is to keep a fairly accurate orientation estimate so that the robot knows which direction is forwards.

A few more things you may want to think about:

- What is the best position and angle of camera for this task? A camera which looks downwards at a high angle will capture good information about nearby obstacles, but will not be able to see far ahead, while if the angle is more shallow you will see further but perhaps with less accuracy.
- Consider the speed of the vision loop which captures information from the camera, and how this relates to the speed of the planning loop. Do these loops need to run at the same rate? How fast can they run anyway?
- Note that if you are displaying images in the web display, this can cause the rate of the vision loop to be quite slow and variable due to the need to send images over the network — it is probably hard to run the camera reliably at more than 1 or 2Hz while displaying images, and it will be important to display images while debugging your program. But you may be able to turn that off when things are running well and get the vision loop to be much faster — probably at least 10Hz.
- How fast can your robot move while still behaving accurately, and at what safe distance should it aim to avoid obstacles to be sure of no collisions?

You can do whatever you want with your program, including changing the image processing code or planning algorithm if you want to.

5 Judging

We will judge the robots as follows, based on the **best** run that your robot achieves from the **two attempts allowed**. Note that we will go through all the groups to have one attempt each first, and then do a second round for everybody so there will be a little time to adjust your robots in between, though you must be ready when your group is called. Bear in mind that we will move the obstacles just before each run so that you are not able to program in their locations in advance.

Judging will be quite simple. Your robot should aim to get as far forward from the start line as possible without hitting any obstacles. If it hits an obstacle, we will measure the distance of that obstacle from the start line and that will be your robot's score. If your robot successfully crosses the whole arena without hitting any obstacles, then we will stop the clock when it crosses the finish line at 3m, and the robot's score will be its time. Note that it does not need to stop or do anything special after crossing the 3m line; probably it will just keep moving forward and we will have to pick it up.

So if no robots make it all the way to 3m, the winner will be the one which travelled furthest before hitting an obstacle. If at least one robot get to 3m, the winner will be the fastest one.

We we judge that an obstacle has been hit if any solid part of your robot (wheel, body or sensors) passes over any part of the obstacle — i.e. imagine that each disk represents a tall vertical cylinder of the same cross section. (Cables 'hitting' an obstacle is OK.)

So: accurate obstacle avoidance and continued forward progress is the most important thing, and you should first focus on enabling your robot to cross the whole arena without any collisions while travelling slowly. If it is doing this reliably, then you can try speeding up your robot.

Remember that you will get two attempts, so maybe you could try changing your robot's parameters between runs, to try something safer on the first run and more risky on the second.

There will be prizes for the best two teams in the competition, and we may also award another prize just for the most interesting robot (in terms of algorithm, design or capability), even if that robot does not do very well in the competition!

6 Finish and Tidy-Up

Straight after the competition, we need to collect up the robot kits and tidy up the lab. Please:

- Disassemble your robot. Into your numbered plastic box please pack the same items that it contained at the start of the course.
  - Raspberry Pi and BrickPi, with camera if you have one (these can be left together in their perspex case)
  - SD Card
  - Battery
  - Y-cable
  - Charger
  - HDMI cable
  - 3 motors
  - 1 sonar
  - 2 touch sensors

  - 3 Lego motor/sensor cables

- Return your plastic box to us. We will sign these back in and I won't give any groups a coursework mark until you have returned your box!
- Return all other Lego parts to the drawers.
- If you have tape measures, scissors, sticky tape, pencils, etc., please return them to us.
- Any other robotics bits you find around the lab, please return them to us.
- Please help us to have a general tidy up of any Lego or other Robotics bits which might be lying around the lab. Please throw away any waste paper you may have, and if possible remove any tape or other things that are stuck to the floor. Thank you.

Acknowledgements

Thank you to Riku Murai, Marwan Taher and the other TAs on the robotics course for massive help with preparing this practical.