Robotics

Week 6 Practical: Place Recognition

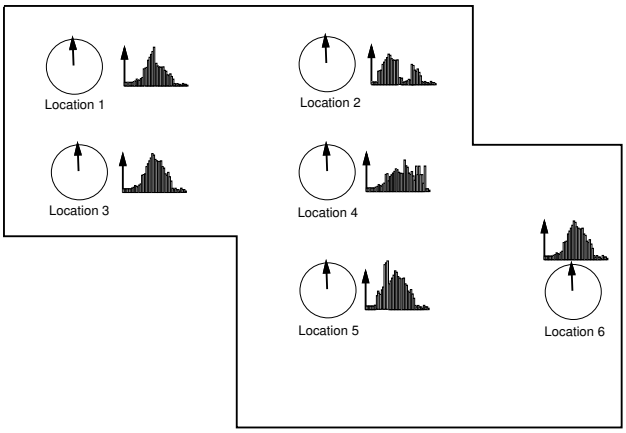
Andrew Davison
ajd@doc.ic.ac.uk

1 Introduction

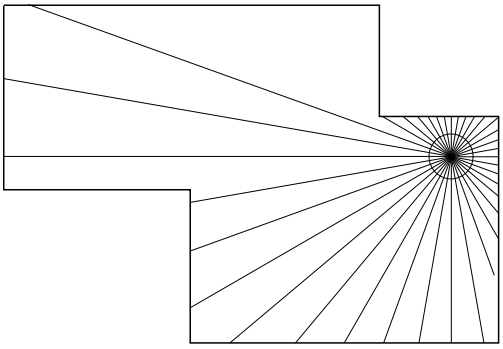
This week we will try a learning and recognition method for global localisation which is able to solve the 'kidnapped robot problem' if the robot is placed randomly in one of a finite number of previously learned locations. Your robots will need to make circular scans to gather a sweep of sonar depth measurements which characterise a particular location. In a training phase, the robot learns like this the appearance of several locations and saves its results in terms of a *signature* or *descriptor* for each one. In a later testing phase, the robot is put randomly in a location, makes a scan and then must decide whether it recognizes any of the learned locations depending on whether a signature of the new scan matches any of the learned ones well enough.

This week's practical is UNASSESSED. There are several tasks laid out in the following sections, and it will be very useful for your understanding of the examinable material in the course to attempt them, but we will not have a formal assessment session and achievement in this practical will not count towards your overall coursework mark for Robotics.

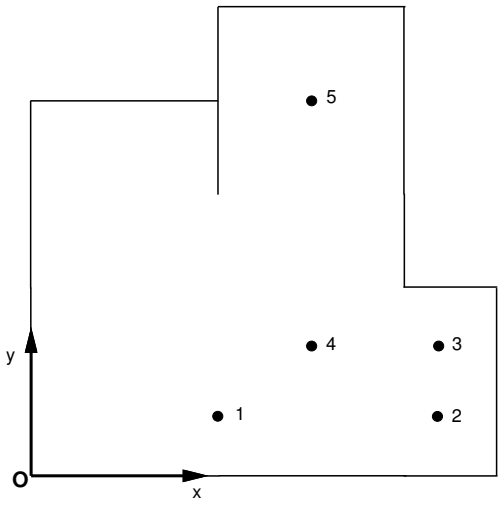
2 Place Recognition



The basic mechanism we will use for place recognition is a scan of sonar depth measurements obtained from spinning the sonar on the spot and taking measurements at regular intervals. The array of measurements generated is saved as a 'signature' of this location, and will hopefully be unique within the environment.



The experimental environment for this practical will be the same enclosed area we used for Monte Carlo Localisation. The training points for relocalisation (for parts 4.2 and 4.3) will be first five numbered waypoint locations 1, 2, 3, 4, 5 from the MCL practical. These are marked on the course.



One issue is that your sonar sensors may max out and not be able to measure the farthest distances. However, since they will do this in the same way during both learning and recognition you will be able to ignore this and just record the raw depth measurements from the sensor for comparison.

3 Some Code to Help You

The file available from `http://www.doc.ic.ac.uk/~ajd/Robotics/RoboticsResources/place_rec_bits.py` contains useful bits of Python code for this practical which you may want to use as the basis for your programs. In particular, it shows you how to write and read simple signature files from Python. You will need to save signatures to file during learning and read them in during recognition.

Either function `learn_location()` or `recognize_location()` should be uncommented to generate a program which will either learn one new location and save it to file, or try to recognize the robot's current location by comparing against the saved signatures. Clearly you should generate separate executable programs for both of these.

Function `learn_location()` gives the basic procedure for learning and saving a location signature. You will have to create the function `characterize_location(ls)`. This function will have the job of spinning the sonar, gathering measurements and saving them in an instance of class `loc_sig` which saves the signature for one place, consisting of `NO_BINS` depth measurements. You will see that

there is code to automatically save the new signature to a file with a filename which is generated not to clash with locations previously learned.

Function `recognize_location()` is the procedure for recognizing a new location, consisting of taking a new scan via `characterize_location(ls)` and checking in turn against the saved signatures. You will have to write the code to do the comparing!

Note that once you create files on the Pi, you will be able to see and delete them manually if you need to from the web interface.

4 Objectives

4.1 Circular Scan and Measurement

Write a program which spins your sonar carefully and precisely through a calibrated full rotation of $360°$, records depth measurements at regular intervals (which could be of $1°$ or more) and plots the results graphically in the web graphical interface as a set of radial lines relative to the centre of the screen in the style of the figure above. This program should work anywhere — the robot wouldn't need to be in the map.

This spinning could be achieved with a fixed sonar by actually rotating the whole robot through a full turn on the spot, having carefully calibrated its rotational movement as in previous practicals. An alternative is to mount the sonar sensor on the top of the robot using the third motor in your kits such that the sonar can be spun through a whole rotation horizontally. The encoder on this motor can then be used to control the sonar's rotation precisely without fear of drift. You can either use position mode and move the motor in small steps, or possibly in velocity mode, moving continuously and making sonar measurements at regular encoder intervals. You will also need to be careful with the cable connected to the sonar to make sure that it doesn't get tightly wound up as the motor rotates.

Note that this week the robot won't actually need to drive anywhere; just capture a sonar scan from a fixed position. You will want to experiment with angular increment of your measurments to see what size of signature is really needed to characterise a location.

4.2 Learning Signatures and Recognizing Location with One Orientation

Write a learning program such that when the robot is placed at a location, it makes a circular scan, records a signature and then saves it to a file. *Train* your robot by placing it in all of the five unique numbered locations (1, 2, 3, 4, 5) in the MCL map and saving a signature. In each case, make sure your robot is oriented to point along the x axis when you start the scanning movement. Then write a recognition program which captures a new signature when the robot is placed in a test position, checks this against all of the five saved locations and reports either a good match to one of them or no good match. See the lecture notes this week for the straightforward correlation test (sum of squared differences) which can be used to compare the test signature with the learned ones. In this test, the robot will always be placed with its orientation exactly pointing forward along the x axis of the map so you will not have to calculate rotation.

The coordinates of the five locations are:

1. (84, 30)
2. (180, 30)
3. (180, 54)

4. (138, 54)
5. (138, 168)

These are the same as first five waypoints from last week's MCL practical.

4.3 Efficient Location Recognition at Any Orientation

Write a program which can recognize which of the five learned locations the robot is in regardless of what orientation it is placed at in the testing stage, and is also able to report the angular difference between itself and the x-axis.

To achieve this, you will have to see (as in the lecture notes) which shift of the test signature gives the best matching score against the learned locations. Clearly, if the robot is in the same place but just rotated then it should record the same scan values but shifted (with wrap-around). You need to find which learned location, with which shift, gives the best match score overall.

You will probably find the performance of your programs to be quite slow due to all the tests required if you search through all possible locations and shifts exhaustively, so for extra speed you should implement a rotationally invariant signature for each location using the depth frequency histogram I explained in the lecture notes. This should quickly be able to find the correct location and the concentrate on testing to find the angular shift.

Acknowledgments

Thank you to Adrien Angeli who developed the original version of this practical.