CSC C85 – Embedded Systems
Project # 1 – Robot Localization

The goal of this project is to apply the ideas we have discussed in lecture to a real-world robot localization task. You will be working with Lego NXT robots, and you will have to find ways to work around the constraints and limitations enforced by the NXTs characteristics while implementing a probabilistic localization algorithm similar to particle filtering.

Learning Objectives:

You will strengthen your understanding of particle filters as they apply to the problem of robot localization

You will apply ideas acquired in lecture regarding sensors, noise, and nose management.

You will learn to handle limitations in the hardware and software available to you as an embedded systems developer.

You will design a mobile robot for a specified task, and think about how to configure the mobile elements and sensors so as to best solve the task at hand.

You will learn to write robust code that is able to handle testing conditions different from those you may have foreseen.

Skills Developed:

Designing, building, and programming a mobile robot to solve a specified task.

Dealing with real-world sensor and actuator behaviour. Working with uncertain inputs.

Implementing algorithms on platforms that have limited computational resources.

Reference material:

The handout for Lab #2 detailing particle filters.

Your lecture notes on robot localization and sensors.

Acknowledgements:

Project definition, implementation, testing, and documentation by Kevin Lee, Summer 2012.

The Little Lost Robot:

You have taken on an important assignment to build a fully automated robot to locate and dispose of hazardous material transportation. After a long and arduous design and verification process, your robot is ready for roll-out.

On the test day, the robot will be dropped onto a realistic simulation of an accident involving the release of a hazardous substance. The robot's task is to locate a pile of dangerous muck and transport it to a safe location. However, on the test day you find that GPS localization is off-line and the robot's receiver seems to be inoperative. Your little robot is now lost.

Being a careful designer, you have installed a backup system on the robot that will allow it to use landmarks in the environment to locate itself, navigate toward the hazardous muck, and proceed to the safe disposal location. Will your robot find its way? or is it destined to become a pile of rusting junk in some lost corner of the test area?

Project Overview:

You will be in charge of:

* Designing and building a mobile robot that will successfully navigate the map we provide, reaching two specified locations, one for the hazardous material and one for the safe storage facility.

* Make appropriate use of the NXTs sensors to gather information about the environment as the robot travels.

* Implement a particle-based localization algorithm similar to the particle filters you coded for Lab #2, but adapted to this specific problem and the computational limitations of the NXT platform.

* Testing and validating that your robot works

* Writing a short report detailing your design choices, your solution, and your observations on the behaviour of your robot.

Setup

Your robot will be operating within a world consisting of colored regions. The ***map*** colors are Meaningful:

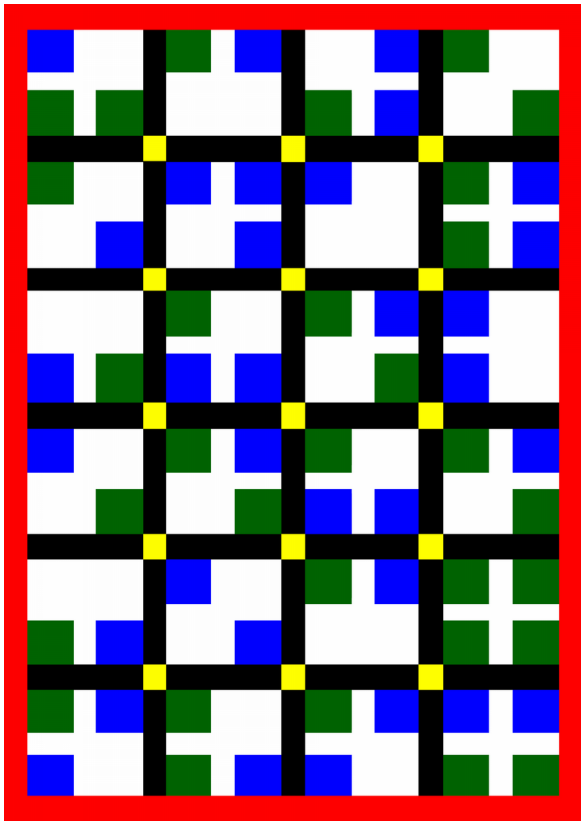
* Yellow - corresponds to street intersections

* Black – indicates streets. Your robot ***can only travel along streets***.

* Green, Blue, White – Indicate buildings with different landmarks

* Red – Map boundaries. Your robot must remain within the boundaries of the map at all times.

Setup (continued)



The image at the left shows the map your robot will be operating in.

You will be provided with a printout of the map on which to do testing.

The robot starter code already sets up the equivalent internal representation of the map in the NXT's memory. Read the comments in the starter code for details.

Your robot will be placed at ***an arbitrary location, facing a random direction along a street.***

Your task is to use the NXT's sensors and your knowledge of robot localization to write a probabilistic localization routine that will enable the robot to determine its location.

Once your robot has determined its location, It must proceed to the specified target and destination locations. These are also provided with the map.

Upon successful completion, your robot will play a tune, emit a facetious remark, or do a happy dance.

Starter kit

Each team will be provided with:

* A printout of the map at a scale suitable for testing. Take good care of it and bring it for testing on project review day.

* A complete NXT set. You will take this set with you, and be responsible for keeping it in good working condition and safe as long as it is in your care.

* Starter code that provides a framework on which to build your algorithm. The starter code will also load the map representation into memory. Read the code for details.

Your Tasks

1) Design and build the mobile robot

* You are free to design and build the robot as you see fit. But you should carefully consider the following:

  - It should be able to navigate the streets easily (i.e. should be small enough to comfortably fit in the map).

  - It will have to make sharp (90 degree) turns at intersections

  - It will have to either navigate backwards easily, or perform 180 degree turns If it reaches the map's boundary.

  - It will need to have the colour sensor placed in such a way that it can scan the map for streets, intersections, building colours, and the boundary.

* Beyond the above, use your imagination. ***We will evaluate your robot in terms of design, mobility, and appropriate use of sensors.*** Particularly cool designs will earn extra credit.

2) Implementation and testing of the localization algorithm

For simplicity, you will only need to determine ***which intersection*** the robot is at.

This means that while the robot is moving along a street, you do not need to update the robot's ***belief*** about its location.

A) Once a robot hits an intersection, it will do the following:

* Scan the intersection to determine the building colours at the 4 corners of the street. Buildings will be blue, green, or white, and you must design a motion pattern that results in the robot scanning these colours in a ***predictable***, and ***reliable*** pattern. It is essential that you know in which order the building colours were scanned, so you can check against the map data.

  The map data gives you building colours around each of the intersections in the map, starting with row 0, column 0, then row 1, column 0, up to row 4, column 0. Next is row 0, column 1, and so on until row 4, column 2.

  Colour data is listed clockwise starting with the North-East corner, so for the intersection at row 0, column 0, you will see 6 2 6 3 which corresponds to white, blue, white, green. Check a couple other intersections to make sure you fully understand how building colours are stored.

Your Tasks

2) Implementation and testing of the localization algorithm (cont...)

(Once a robot hits an intersection, it will do the following: cont...)

* Update the robot's belief about its potential location based on the readings you acquired.

Our grid contains 5x3 potential locations where the robot could be.

We will use a probabilistic algorithm to determine robot location, this means you have to keep track of the probability the robot is located at a given Intersection.

However, this is not all. You also need to know ***which direction*** the robot Is moving. So in practice, you need to keep track of 4 particles at each intersection corresponding to the probability that the robot is at that specific intersection and is moving in one of 4 possible directions.

Q1) What is the total number of probabilities your robot has to keep track of?

Initially the robot has no idea where it is, or which direction it is facing.

Q2) What should be the initial value of the probabilities your robot is keeping track of?

You now have to deal with the following problems:

* Noise – sensor readings are noisy and can be unreliable. You need to implement some strategy to deal with this and ensure you get accurate colour readings most of the time.

Q3) Describe your strategy for dealing with sensor noise, and explain why it should work and how you tested it.

* Determine agreement - Given the sensor readings, you have to determine whether there is agreement between what the robot sensed and what the map says the robot should see at a given intersection. ***Here you need to account for the different possible directions the robot may be facing.***

* Update probabilities – Based on the amount of agreement (you need to define what that means) you have to update the probabilities for each particle.

Your Tasks

2) Implementation and testing of the localization algorithm (cont...)

(Once a robot hits an intersection, it will do the following: cont...)

(You now have to deal with the following problems: cont...)

***** Update probabilities – cont...

You have to decide ***how*** and by ***how much*** to change the probabilities for each particle. But keep in mind the following: ***you must have a reason for the update strategy*** you choose. The updates should account for factors such as your estimate of how reliable the sensor readings are.

The update should depend ***exclusively*** on the current measurements. You should not use historical data of previous grid readings.

After the update, you ***must*** ensure the sum of probabilities over all particles is ***1.0***.

Q4) Describe your probability update function in detail and explain how you defined 'agreement' as well as how you Determined the amount by which to update probabilities.

At this point, the robot has finished processing this intersection.

Two things could happen:

* Some particle with a specific direction has emerged as a ***clear winner*** over the rest and you can conclude the robot has found itself.

If this is the case – Make your robot emit a distinctive sound, or emit a clever remark. Go to step ***C)*** 'completing the mission'

Otherwise:

B) Move

* The robot is still not sure of its location. It must keep exploring the environment.

* Have your robot continue moving along its current direction.

* If the robot hits the boundary of the map it must turn back. It will end up at the same intersection it just scanned. Update particle directions accordingly.

Your Tasks

2) Implementation and testing of the localization algorithm (cont...)

(Move ... cont...)

* If the robot does not reach the boundary, you will reach a new intersection. ***Before you process this intersection you must update the particle positions to account for the move the robot just made.***

  Q5) Describe in detail how you update the probabilities to account for robot motion.

* Once you updated the probabilities to account for motion, go back to step ***A)*** and process the new intersection.

C) Completing the mission

* Your robot now knows where it is with some certainty.

  Q6) Explain your method for deciding when the robot has achieved correct localization.

* Your robot must proceed to two designated locations (stated at the end of the map data file) in the correct order.

  The robot must do this without wandering around the map (it already knows where it is, so no more intersection scanning and probability updates should happen). The robot should not attempt to go outside the map or hit a boundary at this point.

* Once you have reached the second specified target the robot's mission is complete.

  Have your robot do a happy dance, play a song, or emit a facetious remark. ***Bonus marks for cool behaviour upon completion of the mission.***

3) General comments

As stated above, your robot's main loop will consist of

1) Motion,
2) Motion probability update
3) Scanning an intersection
4) Agreement probability update

Your Tasks

(3) General comments... cont.)

You will find you need to modify the following functions in the starter code:

```
move_robot()
scan_surroundings()
move()
sense()
main()
move_to()
change_direction()
```

Be sure to read and understand all the comments in the starter code before you start tweaking it.

Comment on the following:

Q7) Briefly explain your robot design choices, including sensor placement and robot mobility considerations.

Q8) What parameters did you find affect the most the ability of the robot to determine its location?

Q9) Can external factors (illumination in the room, for example) affect the ability of the robot to find itself?

Q10) On average, how many intersections does your robot need to visit to achieve successful localization?

Q11) Are there locations in the map that make localization harder? (i.e. does it matter where your robot starts its journey? Why?)

Hand the answers to all questions in this handout on a separate page with your team's name and each member's name and student number.

No handwritten answers please.

How to fail in this project...

1) Ignore sensor noise and sensor behaviour

Simply assume that the colour sensor is perfect regardless of how you placed it on your bot. Assume that your program can just read the colour values and use them 'as is'. Also, assume your bot's sensor doesn't need shielding for stray light.

2) Ignore noise and inaccuracies in robot motion

Assume that a command to drive forward at certain speed for a specified amount of time always produces the exact same amount of displacement in exactly the same direction. Similarly assume turning always works perfectly.

Assume battery charge doesn't matter. Assume wheels don't slip on the surface and threads don't get stuck.

Assume the map is always perfectly flat and always exactly the same size.

3) Use open-loop control

Assume you can use timed motion commands such as *turn left for 2 sec.* to position the sensor and robot where you want them to be. Don't use sensor readings, after all, the colours in the map are just decorative.

Also assume the map is always exactly flat and the same size.

4) Form over substance

Design the best-looking bot ever that, sadly, can not move because you used the motors to drive awesome-looking crushing-grips-of-doom.

5) Start as late as possible

You can design and build your bot, write the particle filter software, test and debug, and make it work in just one afternoon... right?

Evaluation

Project evaluation will take place at the Embedded Lab

* You will do a brief presentation for everyone, explain your design, and demonstrate its operation '***live***', on the same map we gave you for testing.

* Marking will be based on how well your robot manages to solve the task we gave you.

  However:

  - We do not expect perfection! Things happen in practice with robots, they don't always work as they should.

  - We will pay attention to your design, your answers to questions we will ask while you present, and whether or not different components of the solution are present.

  - The answers to questions in the report are also worth a certain amount of marks toward your project grade.

  - We will ask questions to both team members and expect both to be able to answer any of our questions.

* ***We will make a short video of your presentation/robot demo*** for reference as we review the marks for the entire class. These videos ***will not be posted***. They are exclusively to be used for marking purposes.

* ***In addition to the written answers, submit your NXT code electronically on mathlab by the deadline. Compress your code onto a single .tgz file.***

```
submit -c cscc85f15 -a P1 -f NXTlocalization_teamname.tgz
```

Only one file should be submitted per team.

Finding Help

For issues related to the NXT and NXT programming, starter code, map data format, and so on, please contact your TA during tutorial or in office hours.

For questions regarding the localization algorithm, particle filters, probabilities, or your own ideas about how to solve the problem. Contact your course instructor anytime by email, or drop by at my office.

Your TA can also help you with the localization algorithm design and implementation.

We are happy to help, so please do not hesitate to come to us with questions.

Project Timeline

Phase 1 – Sep. 21 – Sep. 25:

During this week – ***Design your bot and implement the robot design that will be used for the localization task***

Monday Sep. 21 – Tuesday Sep. 22 : Last chance to pick up your NXT kit.

Friday Sep. 25 – First progress check – full team attendance is <u>recommended</u> (this is worth 10% of the project grade)
10:30am-12-30pm at IC 402 – Embedded Lab

Each team must demonstrate:

- Built robot able to move around the map and read colours (e.g. make a different beep for each different colour as it walks on the map)
- Explain their design and any cool features thereof
- Of particular importance, explain sensor placement and operation

Phase 2 – Sep. 28 - Oct. 2:

During this week – ***Work on implementing routines to have your bot***
**** Detect intersections***
**** Orient itself to follow a street***
**** Do right-angle turns from one street onto a perpendicular one at the intersection***
**** Perform an intersection scan and read all 4 colours around it (produce a different sound for each colour to show it reads the correct values)***

Friday Oct. 2 – Second progress check – full team attendance <u>recommended</u> (this is worth 25% of the project grade)
10:30am – ***12:30pm*** – IC 402 – Embedded Lab

Phase 3 – Oct. 5 – Oct. 9:

During this week – ***Implement the particle-filter for robot localization as described in the handout***
**** Your robot should be able to find itself and get to the specified goals having started at any of the locations in the map***

Friday Oct. 9 – Project Review – full team attendance is <u>mandatory</u> (this is worth 50% of the project grade)

9 am until done – IC 402 – Embedded Lab – schedule will be arranged by Doodle poll

Feedback (for yourselves and us) – complete and hand in for bonus marks.

Do you feel this project allowed you to understand the following ideas:
(please answer yes, no, somewhat, and comment on what if anything could have improved your learning experience).

1) The main steps of a robot localization algorithm

2) Dealing with real-world sensors

3) Dealing with real-world motors and robot motions

4) Using probabilities to keep track of ***belief*** about a robot's position

5) Managing uncertainty (e.g. in the quality of sensor readings, or accuracy of robot motions)

6) Do you feel confident you could implement a more complex particle-filtering algorithm for localization given enough time?

7) Any other comments regarding your experience with this project?