CS148 – Project 2
Monte Carlo Localization

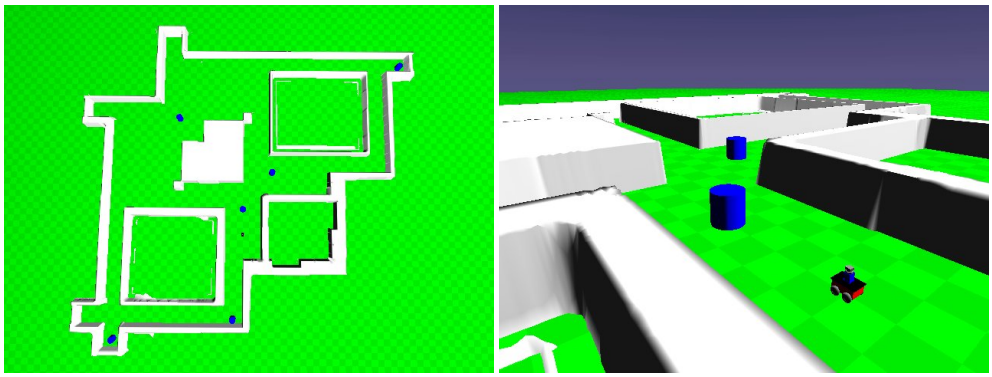

Figure 1: (left) A birdseye view of the Gazebo world (right) A view from Gazebo

1 Dates

- Project demonstration in class: October 26, 2006
- Project writeups due: October 30, 2006, electronic handin by 10pm

2 Introduction

The purpose of this project is to implement a localization procedure that will enable a robot client to infer the robot's position given a map of the environment and data from onboard sensing. The ability to localize will enable your Roomba Pac-Man to find food pellets and power-ups more effectively. To make this implementation more manageable, this project will be implemented entirely in simulation. Project 3 will involve extending the work in this project to the Roombas, so it is very important to get the basics now.

3 Specification

For this project, your robot client will localize using the robot's blobfinder and bump sensor on a provided map. For this task, you will be using a particle filter for estimating the probability distribution across all possible robot locations, expressed in 3 dimensions as an x,y location and an angular heading. In the particle framework, your estimate of the state of the robot will be extracted from the localization posterior distribution.

At a high level, the particle filter is specified as follows. To determine its position at time t, the robot will keep a set of hypothetical estimates on the robot's pose, each expressed as a 3-dimensional vector $X_{i,t} = (x, y, \theta)$. Each hypothesis is considered a "particle." The particle filter then evaluates how likely each particle hypothesis is based on the robot's current sensory information Z_t. In comparing sensory information, the filter weights each particle based on its

"likelihood." These particles and their weights form the localization "posterior," from which a decision about the robot's pose can be estimated. At the next time-step, a new set of particles are generated (or "resampled") based on the weights of the particles at previous time-step (the "prior") and their expected movement from odometry (the "motion model"). This new sample of particles is then evaluated for their likelihood, continuing the localization loop. The filter continually runs, estimating robot pose as long as the robot is running. The following procedure details this localization procedure:

- (1) Randomly distribute equally-weighted particles throughout the environment, serving as the initial particle distribution $P(X_0|Z_0)$.
- (2) Update each particle given the odometry of the robot as a product of the prior $P(X_{1:t}|Z_{1:t})$ and its expected movement $P(X_{t+1}|X_t)$. Be sure to do this in the coordinate frame of the particle hypothesis and not the robot's local coordinates.
- (3) Using this distribution as a weighting factor, randomly sample a new set of particles. One approach to this is to normalize (between 0 and 1) the weights of all the particles and build a step graph (or cumulative sum) out of those new weights. You can then resample from the step graph using a random number generator. The result will be a new set of particles that has more particles around the previous set's likely points.
- (4) Evaluate the likelihood of each particle $P(Z_{t+1}|X_{t+1})$ given the sensory input of the robot. To do this you should compare the robot's sensors (blobfinder and bump) to what each particle would hypothetically receive if it were the correct pose.
- (5) Estimate the most likely pose of the robot given the distribution of the particle filter. There are many approaches to this such as: taking the average of all the particles, taking the most likely particle or a combination of the two. Think about what makes sense.
- (6) Repeat steps 2-4 as the robot drives through the world feeding the new set of particles from step (4) into step (2)

You should pay particular attention to your likelihood function, that is, the probability of observations given a robot's location (matching the expected reading from the map to the actual reading). The sensor readings you will be using may include bump sensor, blob finder, and odometry data. You can think of the map as a function `map(x,y,theta)` that returns the color expected by the map at a location x,y.

The focus of this project is localization. You are not expected to implement a functional Pac-Man client in this project. Rather you should use a client that will simply wander around its environment (such as your Lab 1 client), and produce estimates of its position.

4 Support

4.1 Map

The environment from which you will localize your robot is a mockup of the CIT 5th floor. We have provided a Gazebo world file (proj2.world), Player configuration (proj2.cfg), and image raster map (proj2.ppm) of the 5th floor for you to use. Proj2.ppm was used to generate the gzb file in proj2.world. You may notice that the ppm and the gazebo world differ slightly with regard to the filling in of unreachable areas. These areas were not filled in for the .gzb file so as to reduce the amount of rendering done by wxgazebo. Fiducials added in the .world file, and corresponding blue dots were placed in the ppm file.

4.2 Support Code

For this project we have included some code that you may find helpful in testing and debugging you code. All of this code is written in C++. You are free to use whatever language you like, but the support code will only be in C++. There are three classes that we have included for you:

- The TruthProxy class allows you to query Gazebo for the true location of your robot. While this information **MAY NOT** be used for writing your localization client, it can be used to help determine the accuracy of your client. You will need to add the following line to your Makefile to get the TruthProxy and Image class to compile:

  ```
  'pkg-config --cflags gazebo' 'pkg-config --libs gazebo' -lIL
  ```

- The BumperProxy class is a wrapper we wrote around the Laser Proxy. The Pioneer 2ATs that we have been working with simulation do not have a bump sensor available to them, so we wrote this class that uses a laser to simulate a bump sensor. It uses the same function calls as the BumperProxy in Player, so you will be able to port your code directly to the Roombas for project 3.

  You may have noticed in previous labs that Gazebo and the camera have issues with clipping when you get close to an object. This is important to bear in mind when writing your client.

- The Image class that allows you interact with images. Most noticeably it allows you to read and write to and from a file. The map that your robot uses to localize is an ASCII ppm image. If you haven't already played with creating your own world you can use the GIMP to edit PPMs (save them as ASCII, and delete out the line the GIMP adds) and create worlds using GZBuilder (covered in lab1). You may want to change the format that your program outputs the images as. Binary PPMs offer the fastest method, while TIFFs are smaller in size (good if you have a limited amount of space in your directory).

- The DrawWorld class will allow you to visualize a vector of (x,y) locations via the Image class.

Feel free to modify the Image Class and the DrawWorld Class to your likings, and we welcome you explore different options for visualization if you like. You should note that coordinate frames of Gazebo and of the ppm are not the same. The Image class references the top-left corner of the ppm image as (0,0), while Gazebo's origin is in the bottom-left corner of the ppm image.

5 Project 2 Deliverables and Grading

Project 2 involves two main deliverables, a demonstration of your localization client (in class on 10/26) and electronic submission of your work (project writeup, source code, and other materials). In particular, your project writeup should address design choices with regards to your particle filter, such as number of particles, likelihood function, and position estimate. Please refer to the course missive for details about electronic submission and project writeup format. Addendums to electronic submission:

- the writeup should be named "¡login¿_mcl.pdf" (example, cjenkins_mcl.pdf).
- other materials, such as a video of your client in action, should be put in a subdirectory named "materials/"

Both your handin and your actual robot will determine your grade. Your implementation and writeup will be scored in the following manner:

Implementation		Writeup	
Particle filter	20%	Thesis/motivation	5%
Likelihood function	15%	Approach	15%
Resampling procedure	5%	Evaluation	10%
Localization estimates	10%	Discussion	15%
Exploration mechanism	5%	Conclusions	5%

6 References

[1] M. Isard, A. Blake, "CONDENSATION – conditional density propagation for visual tracking", Int. J. Computer Vision, 29, 1, 5–28, 1998,
http://www.robots.ox.ac.uk/ ab/abstracts/ijcv98.html