

Particle Filtering Homework Project

October 12, 2006

Sandra Mau
Brian Ziebart
Ellie Lin

Problem

The problem we are trying to address is localization in a mapped location based on laser range data using a particle filter. Specifically, we use Monte Carlo Localization, a localization algorithm based on particle filters.

The following sections will detail how the algorithm works, show the results as well as provide a discussion of the issues we took into consideration. We used the Wean Hall data set for testing.

Approach

This section describes how each of the major components of our particle filter works and interacts with one other.

Particle Filter

The main algorithm lies in this function. It involves an initial setup step, which involves loading the map from file, setting the number of random particles and initializing them, and setting up the plot and movie and initializing other variables needed for later.

Next, odometry and laser readings are iteratively read in from the logs. Since laser readings are more sporadic than odometry readings, if a series of odometry readings are read then the particles are only pushed through the motion model without resampling. If laser readings follow odometry readings, then the particles are first passed into the motion model, then the propagated particles that are returned are passed into the measurement model to obtain sensor probabilities (in the form of log likelihood) of each particle. Once all the particles have gone through the models, we resample from them with the probability of drawing a sample proportional to that particles likelihood of being a correct particle. We iterate until all the robot log data has been read.

Motion Model

The motion model uses the odometry information from the robot log files to pass in the reading of the previous location as well as the current location (set u_t). The difference between the previous and current position is taken to get a change in (x,y,θ) . The particle (x_{tm1}) is propagated forward using this odometry information. We then randomly sample for a new particle given by the distribution $p(x_t | u_t, x_{tm1})$. This distribution is assumed normal with mean at the propagated location and standard deviation set based on experiments. The new particle is returned.

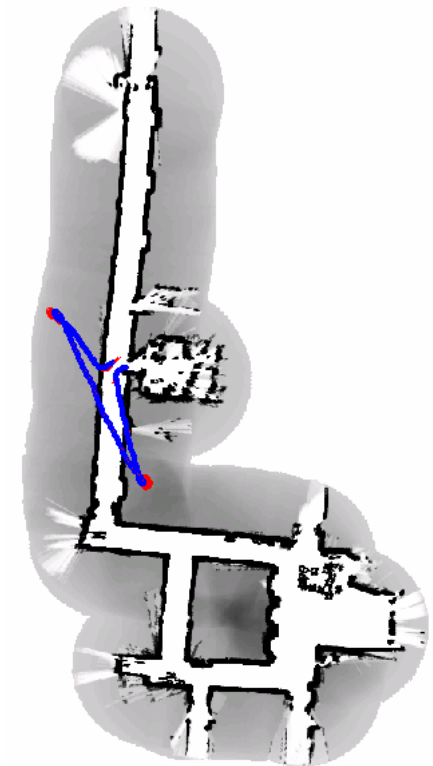
Sensor Model

We project beyond wherever is the assumed laser termination cell for each laser reading of each particle to also capture any obstacles farther away that may have influence on the probability distribution. The probability of the laser terminating in the n^{th} cell is the product of the probabilities of passing through the first $n-1$ cells times the probability of being block in the n^{th} cell. We smooth this function by projecting the probability distribution onto itself scaled down and slightly offset in each direction, and then apply a sliding window smoother with window of 9. This ends up looking crudely like a Gaussian in shape for distinct walls. We then add probability uniformly across the distribution to allow the laser a chance of missing the object and terminating beyond the wall or encountering a dynamic obstacle. We of course normalize this to be a probability distribution again.

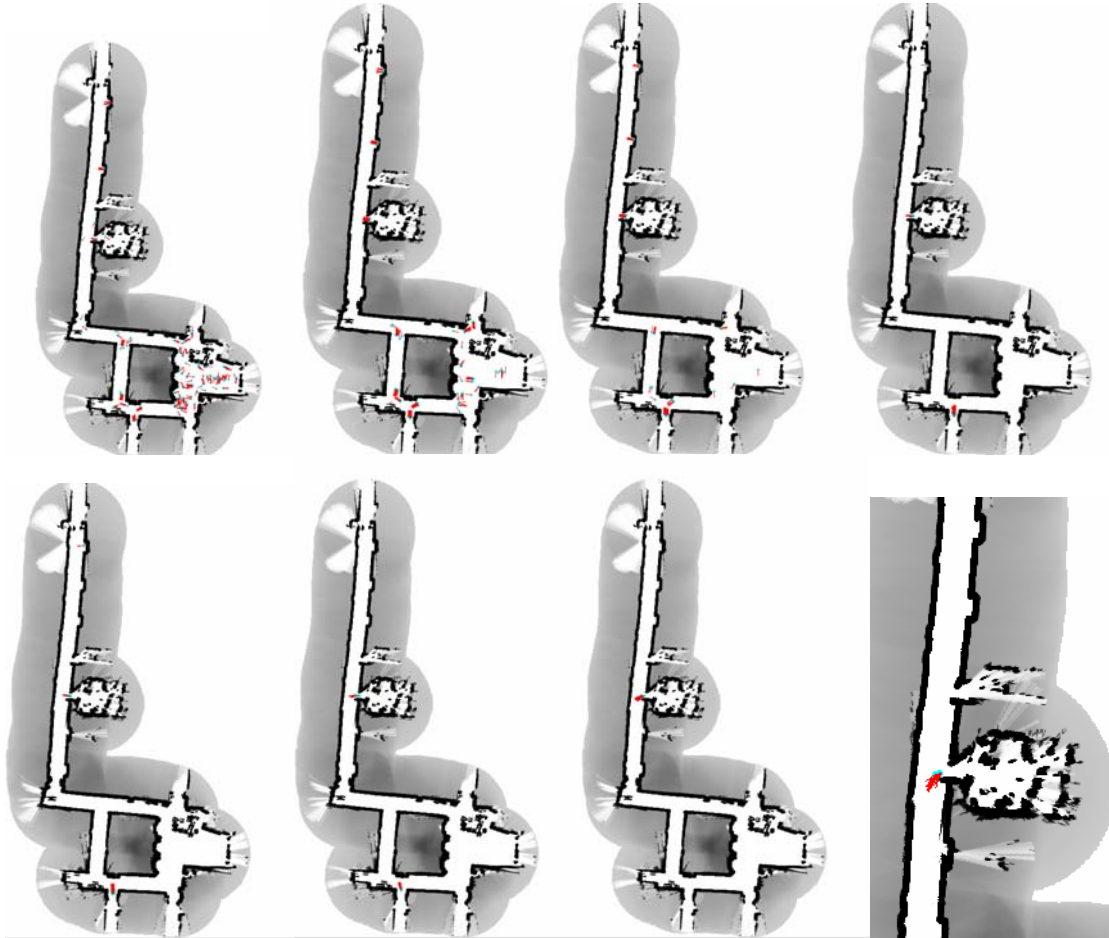
The large number of sensor readings spread the likelihood of any two particles far apart in absolute terms. This is bad for resampling, as the same particle will end up being predominantly resampled. We could smooth our laser termination probability distribution more to help counter this, but instead we do a combination of randomly down-sampling the laser readings by basing the likelihood on only every N^{th} laser measure (also for computation efficiency), and we also divide the log likelihood by a constant (which has a similar effect – it can be thought of as randomly choosing which readings to down-sample).

Results

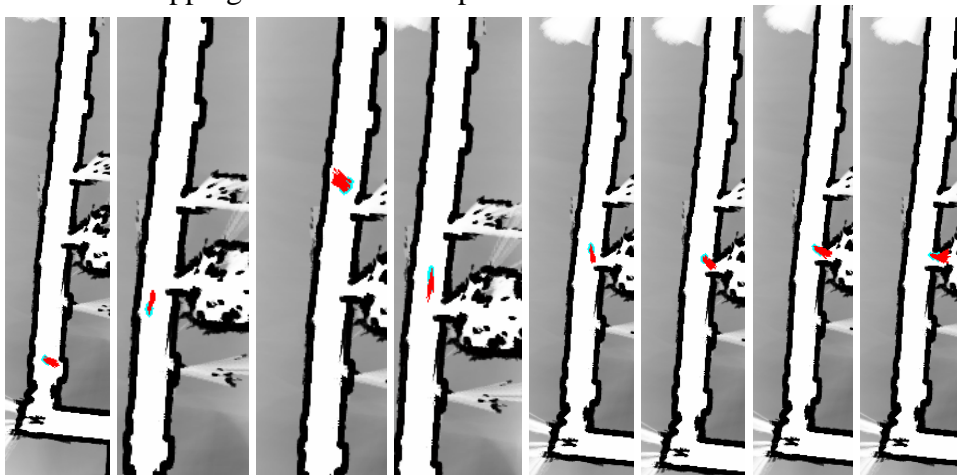
First, we'll show the odometry view of the map (given the correct starting point) without the sensor probability model.



We now show results of our particles over time. The first 8 figures are every 20th time-step of the data. The particles are represented by blue dots and each has a red line associated with it oriented according to its theta value. Notice over time all the particles converge to reside in two plausible starting locations until the robot begins to move, at which point one of those locations becomes more probable than the other.



And then skipping ahead to timesteps of interest



Website for additional resources: <http://www.cs.cmu.edu/~bziebart/robot/>

Discussion

Number of particles

We initially randomly generated 20,000 particles in the areas where the probability of occupancy was 0. The first time we resampled, we reduced the number of particles to 500. After the 100th time step, we reduced the number of particles to 100. We selected the initial number of particles to allow good coverage of the free space and reduced the number of particles keep the runtime of our algorithm reasonable.

Coordinate frames and Matlab

Although we chose Matlab as our implementation language for its ease of visualization, we found that Matlab required us to convert from one coordinate frame to another and spent some time ensuring that our conversions were correct. Specifically, we loaded the occupancy grid as an image, so indexing into our internal map to change the 'color' of an (x,y) value, then displaying the changed image would not be equivalent to plotting a colored point (x,y) directly on the figure. We kept all of our particles in the same coordinate frame as the internal map and did a coordinate frame transform on the particles whenever we plotted them in the figure.

Visualization

We made extensive use of visualization while implementing the algorithm to ensure that it behaved as it should. We visualized the map of Wean as well as the location of the particles at every time step. We additionally visualized the theta value of each particle by drawing a line segment from each particle in the direction of theta. We found this visualization useful while running just the motion model to determine the amount of noise to add in the motion model. We also found it useful for reaffirming that our coordinate frames matched.

Although we did not know the robot's ground truth location in the world, we formed our best guess from watching the video of the robot as it traveled. We speculated that the robot's initial starting location was around the doorway of the large room in the long corridor of Wean, facing the long wall. We visualized the robot's odometry along with the location of laser end points while running through the log from that starting location and orientation and affirmed that our best guess made sense. As a first test of our particle filter, we initialized all of our particles to our best guess starting point and watched to see if the filter behaved as expected.