16-831 Project 3: Robot Localization

Alex Brinkman, Shivam Gautam, Tushar Agrawal

October 26, 2016

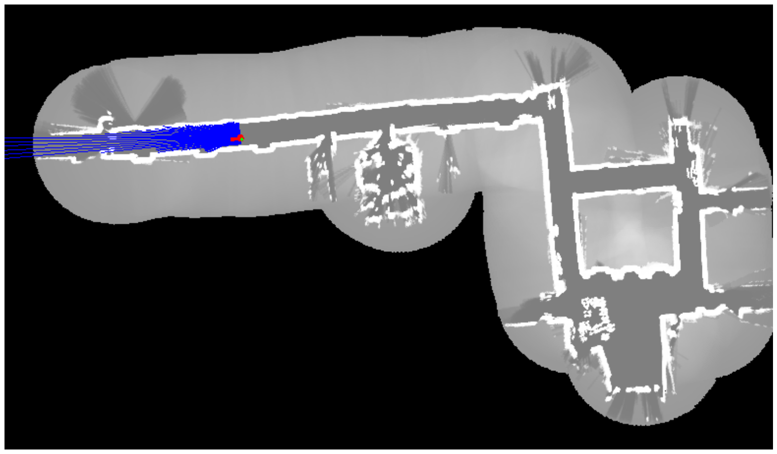


Figure 1: Visualization of the Particle Filter after Convergence

1 Approach

We started by first developing the modules in isolation to facilitate future integration. We started by independently developing the sensor model, motion model, ray tracing, and graphics engine, and log-streaming modules first. We performed unit testing on each to verify they were working properly, relying heavily on the graphics to validate the complex modules. Once the modules were build up, we integrated the modules and debugged the interfacing issues and added importance resampling. Since the modules were tested independently, this was relatively painless. Finally, we refined the pipeline to improve runtime and tuned the particle filter parameters to find a good trade-off between convergence speed and correctness. Figure 1 depicts the final outcome of the particle filter.

2 Implementation

We implemented the particle filter in C++ and used a private git repo in our workflow. All modules were implemented as stand alone classes and invoked in executable scripts. The final integration took the individual classes and encapsulated the modules while exposing configuration parameters to ease tuning. We used GNUplot-iostream to visualize 1D data like the sensor model and the SFML library to visualize the map, particles, and laser readings.

2.1 Sensor Model

The sensor model consists of a superposition of an exponential decay function, Gaussian distribution, uniform distribution, and max value. The Gaussian distribution mean is placed at the expected range measurement with a tunable variance. Since the we wanted to sum the log-likelihood of each of the 180 range readings, the uniform distribution of the model is 1 so for any range measurement this provides a log-likelihood of at least zero. The decay function plays a small role in the overall shape of the function. The Gaussian was tuned first by using reasonable values for gain factor and standard deviation. We found that once we set the initial values to something reasonable, we did not have to tune this model much to get the performance we desired. Figure 2 shows the final sensor model used in our tuned model and Figure 4 shows the final parameters.

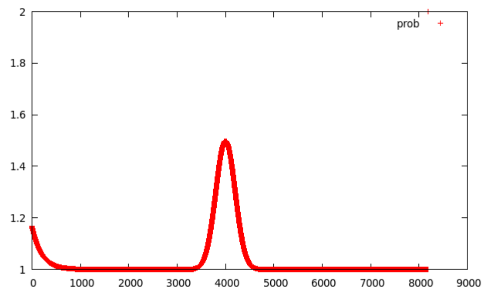
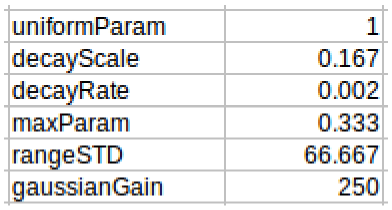


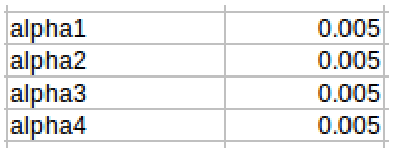
Figure 2: Sensor Model for reference reading for 4000cm, X-axis is cm



uniformParam	1
decayScale	0.167
decayRate	0.002
maxParam	0.333
rangeSTD	66.667
gaussianGain	250

Figure 3: Final Sensor Model Parameters

2.2 Motion Model



alpha1	0.005
alpha2	0.005
alpha3	0.005
alpha4	0.005

Figure 4: Final Motion Model Parameters

2.3 Ray Tracing

2.4 Importance Resampling

3 Results

4 Future Work

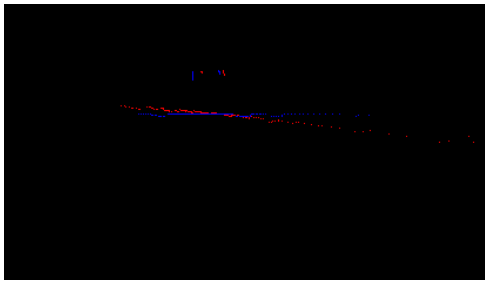


Figure 5: Ray Tracing(Red) Validation against Logged Data(Blue)