# PacBot – RFID-Based Location Tracking for Robotics

Liang-Ting Jiang and Alex Takakuwa

***Abstract*— In this project, we utilize the building-scale RFID readers deployed in the Paul G. Allen Center for Computer Science & Engineering, including hundreds of RFID readers. We implement a particle filter to estimate the location of RFID tags in the building based on the tag read events, and apply the filter on the continuous domain without the need of predefined discrete blocks on the map. To evaluate the performance of the particle filter tracking we compare the error between the location of a localized robot (the ground truth location) and the filtered tag location on the static map. We tested and evaluated several sthinsmodels and motion prediction models, and analyzed effects of filter update rates. Results show our system enables fine-grained RFID-based location tracking with a typical average error of 2.9 meters in a trial on the complete run of a trial on the 4th-floor of the building.**

## I. INTRODUCTION ANR RELATED WORK

Commercial and user-centered applications increasingly use RFID technology to track the locations of people and objects, particularly in the indoor environment in where GPS localization is not applicable. Building location-aware application based on realtime RFID sensor data is challenging since the RFID tag reads are usually noisy, sparse in time, and discrete in space (constrained to where the antennas are deployed). Wireless indoor localization techniques other than RFID have also been proposed, for instance, WiFi-based [1] localization. However, RFID-based localization still has many advantages among others: low cost battery-free tags, the ability to associate virtual objects with physical objects, privacy control, and etc.

In this work, we focus on using the noisy, sparse, and discrete sensor data read from the building-scale RFID readers to estimate the location of the RFID tags. The original RFID Ecosystem infrastrucure in the UW CSE Building is deployed by the Database Research Group to provide a living laboratory for long-term, in-depth research in applications, databases, privacy, security, and systems [3]. Letchner et al. [2] proposed to use Markovian stream, the output of inference on an hidden Markov model (HMM), to infer the distributions of the location over time. The underlying inference technique for HMM is particle filtering. The discrete location candidates $L = \{L_1, L_2, ..., L_n\}$ are defined on the map, and the probability on each locaion is found by dividing the number of particles on the locaion and the total number of particles.

In this work, we aim to extend the particle filtering algorithm by applying RFID-based location tracking. In particular, the main goals are:

Liang-Ting Jiang is with the Department of Mechanical Engineering, University of Washington, Seattle, WA. jianglt@uw.edu

Alex Takakuwa is with the Departments of Computer Science and Engineering, University of Washington, Seattle, WA. alextaka@uw.edu

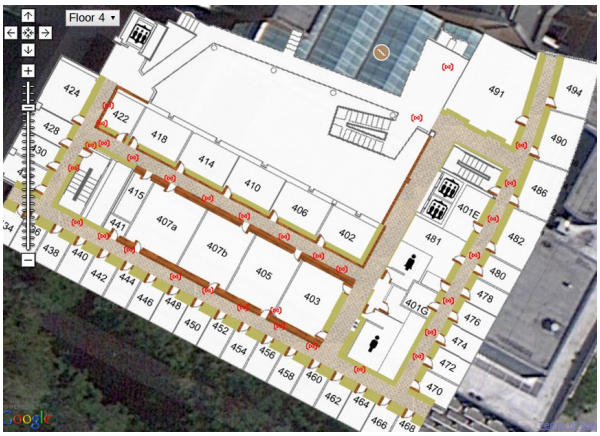


Fig. 1. RFID antenna deployment map on the 4th floor of the UW CSE building.

- Apply a **particle filter** to achieve fine-grained RFID-based location tracking in a **continuous** coordinate without dividing the spaces into predefined locations.
- Combine with **robot localization** technique to enable quantitative evaluation of the location tracking accuracy.
- Study and evaluate several **sensor models** and **motion prediction models** for the particle filter.

## II. SYSTEM DESIGN AND ALGORITHM

### A. System Design

The location tracking system consists of two subsystems: the RFID Ecosystem, and the particle filter, as shown in Figure 2. The RFID Ecosystem consists of 44 RFID readers, and 161 antennas deployed in the building. In our experiment, we tested our system on the 4th floor of the building. Figure 1 shows the deployment of the antennas on the 4th floor of the building. When an antenna reads a tag, it reports the tag read $z_i = (x_i, y_i)$ at the location of the antenna to the centralized server in realtime. The server gathers the tag reads from all the readers in the building for a given time and publishes the accumulated tag reads $\mathbf{Z} = \{z_i\}, i = 1..k$, at a fixed rate. An example of tag reads published by the server looks like: $\mathbf{z} = \{(16, 20), (28, 30)\}$

### B. Algorithm: Particle Filter

We use an apporximate inference based on the particle filter for HMM to infer the distribution of the tag location as the posterior probabilities represented by a set of samples on the map. The particles are randomly initialized on the free space of the map. The belief of the current states is defined

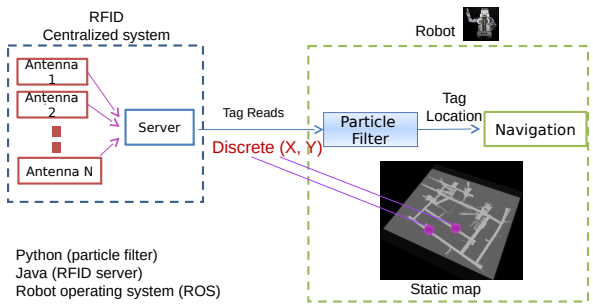


Fig. 2. System architecture.

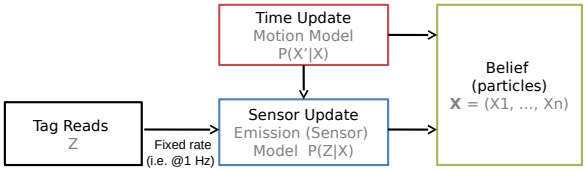


Fig. 3. Particle filter.

as $B(X_t) = P(X_t|Z_{1:t})$. Everytime a tag read comes in to the filter at fixed rate controlled by the RFID server, the particle filter does a two-step update on the belief:

(1) **Motion prediction update** (time update): use a motion prediction model to move each particle, given its location in the current belief.

$$B'(X') = \sum_x P(X'|x)B(x) \tag{1}$$

(2) **Observation update**: use sensor (emission) model to weight each sample with its likelihood on the belief, given the observation.

$$B(X_{t+1}) \propto P(Z|X)B'(X_{t+1}) \tag{2}$$

Figure 3 shows the procedure of the filter updates at each time stamp. In the next section, we will evaluate the effects of using different motion prediction models and sensor models on the tracking accuracy in detail.

### C. Location Estimation

Given the current belief, we determine the estimated location of the tracked tag by computing the average of the coordinate $(x_i, y_i)$ over all the particles in the belief.s

$$X_e = \frac{\sum_i (x_i, y_i)}{N} \tag{3}$$

where $N$ is the number of particles.

## III. EVALUATION

### A. Evaluation Metrics

In order to measure the tracking accuracy, we put a RFID tag on a robot and use our system to track its location $X_e$. The robot localized itself on the static map using an adaptive KLD-sampling Monte Carlo localization approach [4], which is provided as a third-party package in ROS. After the localization, the robot knows its own coordinate on the map $X_r$. Using this as the ground truth data, we compute the Euclidean distance between the ground truth coordinate and the estimated location:

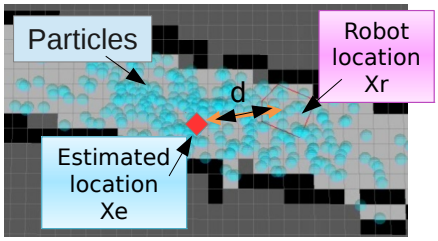


Fig. 4. Distance measurement $d$ between the groundtruth (robot) location $X_r$ and the estimated location $X_e$ from the particle filter.

$$d = euclidean(X_e - X_r) \tag{4}$$

We then compute the tracking error $e$ as the averaged distance over each data collection trial:

$$e = \frac{\sum_i d_i}{N_u} \tag{5}$$

where $N_u$ is the number of filter updates in the trial. In the following subsection, the error metric will be used to compare the accuracy using different filter parameters including the update rates, motion prediction modeles, and sensor modeles.

### B. Motion Prediction Model

We test motion prediction models with the particle filter at different filter update rates

*1) Random Movement Model:* The random movement model is based on the movement model for RandomGhost - given a particle position, we perform a time update that moves it randomly in all directions. However, because we are working in a continuous space, we model movement with a Gaussian, rather than a direction and associated probability. The result is a distribution of possible positions, from which we resample our particles.

*2) Velocity Model:* There are, however, some problems with the random movement model. Namely, the motion model is meant to track 'ghosts', which in our case are meant to be humans walking around the 4th floor with RFID tags. We notice that a random movement model at each timestep does not accurately reflect actual human movement patterns. As such, we adopt a simple model that would more accurately reflect humans' tendency to continue to move the way they have been moving. Our basic model is as follows:

- With $80\%$ probability, the agent will maintain it's velocity (speed and direction) from the previous time step
- With $20\%$ probabiltiy, the agent will move randomly
- If maintaining it's velocity sends an agent off the map, we instead move randomly with $100\%$ probability.

TABLE I

TRACKING ACCURACY COMPARISON WITH DIFFERENT MOTION PREDICTION MODELS AND FILTER UPDATE RATES (N = 300). SENSOR MODEL: MULTIMODAL GAUSSIAN ($\sigma^2 = 0.5m$)

| Update Rate (Hz) | Motion Model | Mean Tracking Err (m) |
|---|---|---|
| 10 | Random ($\sigma^2 = 0.1m^2$) | 2.434 |
| 1 | Random ($\sigma^2 = 0.1m^2$) | 4.115 |
| 1 | Random ($\sigma^2 = 1m^2$) | 2.481 |
| 1 | Velocity ($\sigma^2 = 0.1m^2$) | 2.543 |

*3) Effects of Motion Models:* We found the tracking performance is correlated with the combination of the random movement variance $\sigma^2$ and the filter update rates. Slower update rates require a larger variance $\sigma^2$ to give enough moving freedom for each of the particles. However, applying a velocity model helps the particles spread out effectively enough to provide accuracy similar to the random model with ten times the refresh rate, or ten times the variance, or ten times the variance. We did not include a test with high variance and update rate for the velocity model. Table I shows a comparison of the combinations of the parameters tested on a trial moving through the entire 4th floor of the building.

### C. Sensor Model

Here we test three sensor models with the particle filter at sensor update steps. If a reader detects an RFID tag, we mark it as active, and it will submit both it's location, and the number of reads per time step to the centralized server.

*1) Multimodal Gaussian:* This sensor model is represented as a simple Gaussian distribution $N(\mu, \sigma^2)$ centered directly underneath the active reader. To build this distribution, we iterate through each active reader and weight a Gaussian distribution $N(\mu, \sigma^2)$ by multiplying by the number of reads. This way, when a reader registers more reads, it indicates a higher probability that a tag is beneath the reader. After iterating through all active readers, we normalize our probabilities. If multiple readers read a tag, we can see that the distribution will be a milt-modal Gaussian, with peaks at each of the antenna locations.

*2) Center of Mass Gaussian:* Logically, a multi-modal distribution doesn't make sense. In a two reader situation, if a tag registers reads from both readers, it seems more likely that the object is between the two readers than directly underneath one or the other. In the center of mass Gaussian, when a tag is read by multiple readers, we shift the Gaussian so that it is centered around the center of mass (COM) of all the antennas. To find the COM, we weight the location $(x, y)$ of each active reader by the number of reads, and calculate the coordinates of the center of mass, which will be closer to readers which register more reads.

*3) Trained Emperical Sensor model:* However, the previous two models use a stock Gaussian distribution, which is not built by experimental data. To improve upon those two models, we use ground truth data to build representative distribution based on the number of reads reported by an active reader. The trained empirical sensor model maps a number of reads to a Gaussian distribution $N(\mu, \sigma)$ of distances from the reader. Figure 5 shows the sensor model trained from 1237 sensor data points. The data reflects the underlying logic - as tag reads increase in number, the probability that a tag is near the reader is very high. More subtly, we can see that as tag reads decrease, the mean increases slightly, in addition to increases in variance.

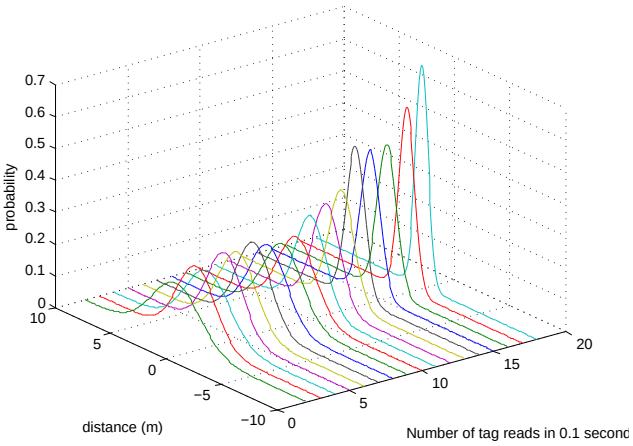


Fig. 5. The emperical sensor model trained from 1237 sensor data points.

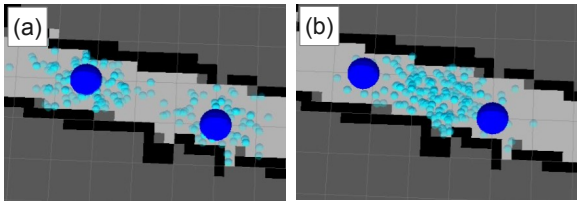


Fig. 6. Comparison of the distribution using different sensor models when a tag was read by two antennas: (a) using the multimodal gaussian model. (b) using the trained emperical sensor model. The light blue dots are particles, large dark blue dots are readers.

*4) Effect of Sensor Models:* The effect of the using different sensor models is best displayed when a tag is read my multiple antennas. Figure 6 shows an example of a tag read my two antennas. While using the multimodal gaussian model, the distribution after sensor updates represents a bimodal distribution, while using the trained empirical sensor model results in the desirable distribution centered between the two antennas. Such a distribution translates well to $> 2$ readers as well.

When tested on the trial moving through the entire 4th floor, the performance difference is not as obvious as that from the motion model, as shown in Table II. Because the tag moves quickly across the entire floor, the robot does not spend much of it's time in readable areas, especially in areas where it activates multiple readers. As such, measurable performance benefits of improved sensor models are minimal. However, the trained sensor model does provide improvement when we reduce the scale from the entire floor to a smaller size only including a couple readers, as shown

TABLE II

**Whole Floor:** TRACKING ACCURACY COMPARISON WITH DIFFERENT SENSOR MODELS (N = 300), RANDOM MOTION MODEL ($\sigma = 0.1m$).

| Sensor Model (Standard Deviation) | Mean Tracking Error (m) |
|---|---|
| Multimodal ($\sigma = 0.1m$) | 2.434 |
| Center of Mass ($\sigma = 0.1m$) | 2.526 |
| Trained Dist. ($\sigma = 0.1m$) | 2.681 |

TABLE III

**Small Section:** TRACKING ACCURACY COMPARISON WITH DIFFERENT SENSOR MODELS (N = 300), RANDOM MOTION MODEL ($\sigma = 0.1m$).

| Sensor Model (Standard Deviation) | Mean Tracking Error (m) |
|---|---|
| Multimodal ($\sigma = 0.1m$) | 1.414 |
| Center of Mass ($\sigma = 0.1m$) | 1.370 |
| Trained Dist. ($\sigma = 0.1m$) | 1.360 |

in Table III.

## IV. REFLECTIONS

We were particularly surprised by how difficult it can be to implement a real world system. There were many factors that we were not able to understand or account for going into the project. Issues with robot network connectivity were particularly frustrating and derailed a number of data collection attempts. Furthermore, our final goal was to get the PR2 robot to chase RFID tags by instructing it to move to a position. Given how the robot moves currently, this seemed feasible, but without knowledge of the algorithms that choose a route and instruct movement, debugging unexpected behavior proved too difficult. In particular, a bug that caused the robot to stop and start required restructuring of a number of algorithms in order to guarantee, with computational efficiency, that any point we can potentially send to the robot would be 'in bounds' on the map. Unable to get consistent performance out of the robot, we did not manage to get a reliable working game of PacMan on the fourth floor. However, having real world test data verify our intuition and our algorithms was very rewarding. Certainly, however, we would have spent more time early on just getting the robot moving around so as to ameliorate some of the more difficult and time intensive challenges involving data collection later on.

## V. CONCLUSION AND FUTURE WORK

We have successfully applied a particle filter to track locations of RFID tags on the fourth floor.

There are a number of expansions to this project which represent avenues for future work. As far as data collection and verification, more data and different environments/situations in which to train the robot would be helpful. Not only for providing more statistical significance to our findings, more data would allow us to train the robot more effectively, possibly by detailing the read profile for each RFID reader on the floor so that we could localize. Also, we noticed that refining the accuracy of sensor updates didn't prove to be nearly as useful as increasing frequency of updates, or applying velocity models. As such, we could also build and implement an Unscented Kalman Filter (UKF), which may give better performance at the cost of some accuracy. However, in this particular case, the increase in performance may allow further increases in the frequency, and it would be interesting to compare the effectiveness of UKF in comparison to our Particle Filter. Lastly, our Pacman project is simply using the greedyAgent from previous Pac-Man projects. Developing a POMDP that could optimize the agent's behavior in a partially ovservable space. Changing estimates and actions based on observations of a human's behavior over time would be an interesting project once the robot is up and effectively chasing down ghosts. We also would need to implement a way for the PR2 to construct a plan to attack multiple 'ghosts' without changing routes too much. Furthermore, a way to have the robot chase and successfully eliminate ghosts would be a fun project, as it should be able to differentiate humans who are playing the game from humans who are not, via a special hat/shirt of some sort.

## VI. APPENDICIES

The video of an example experiment is online at:

http://youtu.be/fPGdqy2C3sA

Liang-Ting Jiang is responsible for the designing and implementing the system, robot integration, and data collection. Alex Takakuwa is responsible for the design of the sensor and motion models, implementation, and verification. The code base for the particle filter has been adopted from the Berkeley Pac-Man Projects originaly developed by John DeNero and Dan Klein. The code base for robot navigation and localization has been adopted from a ROS package. We used the PR2 robot from UW Sensor Systems Lab.

## REFERENCES


[1] K. Chintalapudi, A. P. Iyer, V. N. Padmanabhan, Indoor Localization Without the Pain, in *MobiCom'10*, September 2024, 2010, Chicago, Illinois, USA.

[2] J. Letchner, C. Re, M. Balazinska, M. Philipose, Challenges for Event Queries over Markovian Streams, in *IEEE Internet Computing Magazine*, vol. 12(6), pp. 30-36, 2008.

[3] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska, G. Borriello, Building the Internet of Things Using RFID: The RFID Ecosystem Experience, in *IEEE Internet Computing Magazine*, vol. 13(3), pp. 48-55, 2009.

[4] D. Fox, Adapting the sample size in particle filters through KLD-sampling, in *nternational Journal of Robotics Research (IJRR)*, vol. 22, 2003.