WiFi Localization and Navigation for Autonomous Indoor Mobile Robots

Joydeep Biswas
The Robotics Institute
Carnegie Mellon University
Pittsburgh PA 15213
Email: joydeepb@cs.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh PA 15213
Email: veloso@cs.cmu.edu

***Abstract*— Building upon previous work that demonstrates the effectiveness of WiFi localization information per se, in this paper we contribute a mobile robot that autonomously navigates in indoor environments using WiFi sensory data. We model the world as a WiFi signature map with geometric constraints and introduce a continuous perceptual model of the environment generated from the discrete graph-based WiFi signal strength sampling. We contribute our WiFi localization algorithm which continuously uses the perceptual model to update the robot location in conjunction with its odometry data. We then briefly introduce a navigation approach that robustly uses the WiFi location estimates. We present the results of our exhaustive tests of the WiFi localization independently and in conjunction with the navigation of our custom-built mobile robot in extensive long autonomous runs.**

I. Introduction

Successful localization and navigation is of utmost importance for task-driven indoor mobile robots. In the past, a variety of approaches to localization have been explored, including using sonar [1], laser scanners [2] and vision [3].

With the increasing prevalence of wireless LAN, "WiFi," considerable work has been done on using signal strength measurements from WiFi access points for localization. Approaches based on wireless signal strength propagation models have been proposed [4], but due to the complex interactions of WiFi signals in particular in indoor environments, data-driven signal map based methods have been found more suitable [5]. Wifi-based localization has been shown to be successful in different scenarios [6], [7], [8], [9] in terms of its ability to support humans or robots to identify their locations.

Our work is motivated by our indoor visitor companion robot, CoBot (Fig. 1). CoBot is a custom built robot[1] with a four-wheel omni-directional drive.

We first present a WiFi signal strength based localization algorithm that uses a parametric graph based representation of the environment, similar to [8]. In our case, the graph is composed of the discrete points for which we collect the WiFi signal strengths, and the perceptual model as well as the location hypotheses are constrained to lie strictly on the graph. Our WiFi localization algorithm uses odometry data and a particle filter to represent the location belief, and proceeds in four phases to compute the new particles,

[1]Thanks to Mike Licitra, who designed and built the robot.

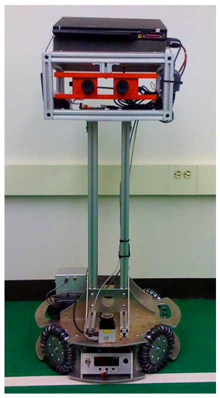
Fig. 1. CoBot, our mobile robot (aimed at being a visitor companion).

namely *to predict* based on the robot's own actions, *to update* based on the WiFi-sensed data, *to constrain* based on physical constraints of the domain, and *to resample* to account for sensory and model mismatches. We then contribute a navigation algorithm that utilizes the location and uncertainty estimates of our WiFi localization algorithm to robustly navigate a robot autonomously in an indoor environment.

We extensively evaluate our approach in one floor of a university building with a series of interconnected hallways. The resulting CoBot is able to localize itself accurately, avoid static and dynamic obstacles, and navigate to any arbitrary location on the map from any other arbitrary location. This we show with experimental results from successful long autonomous runs of the robot.

In summary, the contributions of this paper include:

- Discretely sampled WiFi - graph representation of the robot environment
- Continuous perceptual model of the location of the robot generated from the discrete samples
- WiFi localization algorithm using this perceptual model and constraining the location hypotheses to the graph
- Navigation algorithm that uses location and uncertainty estimates of the localization algorithm
- Extensive results of the combined approach tested on an actual robot

The paper is organized as follows: Section II presents the map representation and acquisition. Section III describes the WiFi-based localization algorithm and corresponding experimental evaluations. Section IV introduces the overall navigation approach. We then present our empirical evaluation of the combined localization and navigation in Section V, and Section VI concludes the paper.

II. Map Definition And The Learning Phase

A. Map Definition

We denote the Map of the building as $\mathbb{M}$, with $\mathbb{M} = \langle V, E, A, \mathbf{M}, \mathbf{D}\rangle$ where V is a set of vertices, E a set of edges to connect them, and A the set of WiFi access points in the environment. $\mathbf{M}$ and $\mathbf{D}$ are matrices representing the WiFi signal strength means and standard deviations across the map. Each vertex $v \in V$ corresponds to a unique location in the building, and is defined by $v = \langle l\rangle$. Here $l = (x, y)$ denotes the cartesian coordinates of the vertex. Edges $e = \langle v_a, v_b, \Gamma\rangle, e \in E$ indicate that there is a navigable path between vertices v_a and v_b. For every edge, the width of the corridor in that section and the length of the edge are represented as $\Gamma = \langle width, length\rangle$. Element $\mathbf{M}_{ij}$ of $\mathbf{M}$ is the mean WiFi signal strength (in dBm) of access point a_j as measured from vertex v_i. Similarly, element $\mathbf{D}_{ij}$ of $\mathbf{D}$ is the observed standard deviation of the WiFi signal strength of access point a_j as measured from vertex v_i.

B. Learning Phase: Data Collection and Offline Processing

Matrices $\mathbf{M}$ and $\mathbf{D}$ are enumerated during the "Learning Phase", which needs to be performed once. The Learning Phase starts with a manual definition of a "skeleton graph" of the map, where the longest straight line segments in the building are defined as the edges of the skeleton graph. Once this is done, a path is planned to traverse every edge of the skeleton graph. The robot then follows this path, and along each edge, it stops at regularly inter-spaced sample locations of a user-defined maximum spacing, and defines a new vertex. Thus, each edge of the skeleton graph is split up into multiple smaller edges. At each sample location (which is also a vertex in the newly generated complete graph), the robot collects WiFi signal strength readings for a predetermined (and user-defined) duration. During this process, the robot uses odometry alone for estimating its localization, and errors in localization are corrected manually using a graphical user interface. Thus, the skeleton graph is used to generate the complete graph, and the matrices $\mathbf{M}$ and $\mathbf{D}$ are populated.

III. WiFi Localization

Our WiFi Localization algorithm uses Monte Carlo Localization [10] with Bayesian filtering to maintain a set of hypotheses of the robot location in real time. As part of the Bayesian filter, we need a perceptual model which can be used to calculate the probability of making a particular signal strength measurement S at a location l, $P(S|l)$.

A. Estimating the WiFi Signal Strength Map, And The Perceptual Model

As described in Section II, the WiFi signal strength mean and standard deviations of every access point are measured from each vertex $v \in V$. Using this data, we model the WiFi signal strength mean and standard deviation as being piecewise linear along the graph, with Gaussian noise. Let l be a location on the edge e ($e = \langle v_i, v_j, \Gamma\rangle$) of the graph, between vertices v_i and v_j. Let $v_i = \langle l_i\rangle$, $v_j = \langle l_j\rangle$. Let M^l denote the vector of mean signal strengths of every access point as seen from location l. The component M^l_k of vector M^l is the mean WiFi signal strength of access point a_k ($a_k \in A$) as seen from location l. Similarly, let D^l denote the standard deviations of the signal strengths of each access point as seen from location l, with D^l_k being the standard deviation of the signal strength of access point a_k as seen from location l. Hence, the linearly interpolated mean signal strengths vector $M^l = [M^l_1 \ldots M^l_{|A|}]$, and the standard deviations vector $D^l = [D^l_1 \ldots d_{|A|}]$ at location l are given by:

$$M^l_k = \frac{\|l - l_j\|M_{ik} + \|l - l_i\|M_{jk}}{\|l_i - l_j\|} \quad (1)$$

$$D^l_k = \frac{\|l - l_j\|D_{ik} + \|l - l_i\|D_{jk}}{\|l_i - l_j\|} \quad (2)$$

During the operation of the robot, let $S = [S_1 \ldots S_{|A|}]$ be a WiFi signal strength observation set, where S_i is the measured WiFi signal strength of access point i. Hence, the probability of making this observation S from a location l, ignoring unobserved access points (*i.e.* $S_i : S_i = 0$) is given by:

$$P(S|l) = \prod_{i=1, S_i \neq 0}^{i=|A|} \left(\frac{2\epsilon}{\sqrt{2\pi} d_{xi}} \exp \frac{(S_i - M^l_i)^2}{2(D^l_i)^2} \right) \quad (3)$$

Fig. 2 shows the mean signal strength of one access point (a_1) for different vertices $v_i = \langle l_i\rangle, l_i = (x_i, y_i)$ across the map. Fig. 3 shows a sampling of the values of $P(S|l)$ for $S = [-100, -59]$ from Access Points 1 and 2, for various locations l across the map.

With this estimate of the perceptual model $P(S|l)$ and the motion model of the robot $P(l_t|l_{t-1}, u_{t-1})$, the localization belief of the robot can be recursively updated.

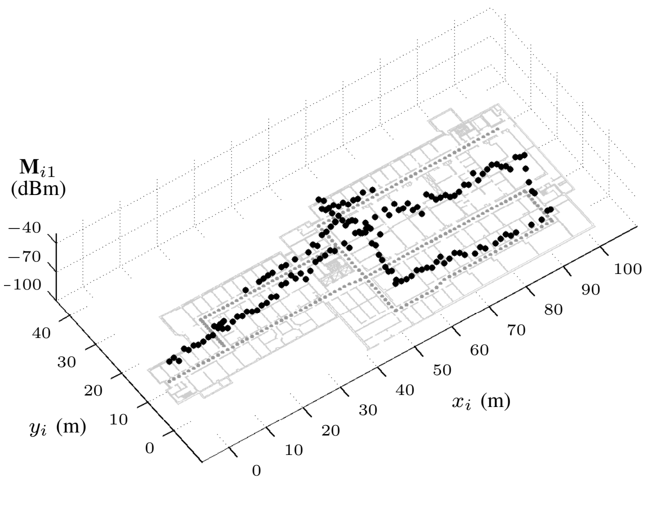

Fig. 2. $\mathbf{M}_{i1}$ (Mean signal strength of Access Point 1) for different vertices $v_i = \langle l_i\rangle, l_i = (x_i, y_i)$ over the map

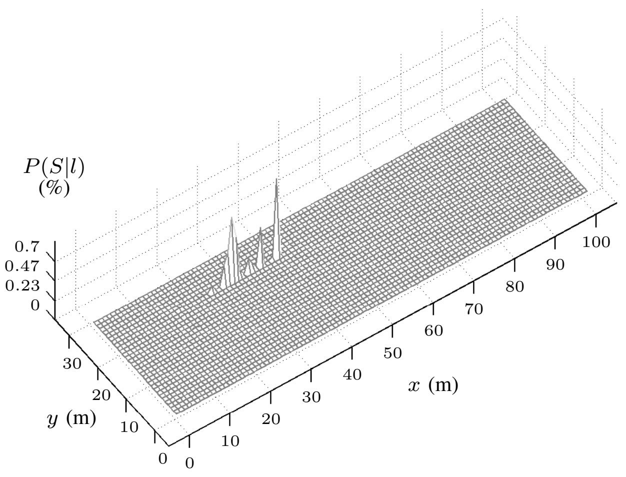

Fig. 3. Probability $P(S|l)$ of making a Signal Strength observation $S = [-100, -59]$, for different locations $l = (x, y)$ on the map

B. Monte Carlo Localization

The recursive Bayesian update for a location belief $B(l_t)$ for location l at time step t using sensor readings S_t and odometry readings u_{t-1} [11] is given by:

$$B(l_t) = \eta P(S_t|l_t) \int P(l_t|l_{t-1}, u_{t-1}) B(l_{t-1}) dl_{t-1} \quad (4)$$

Here, η is a normalization constant, S_t the WiFi signal strength observation at time step t, and u_{t-1} the odometry data between time steps $t-1$ and t. The term $P(l_t|l_{t-1}, u_{t-1})$ is obtained from the motion model of the robot. The term $P(S_t|l_t)$ is given by the perceptual model, eq. 3.

1) Representation of the Location Hypotheses: The multiple hypotheses of the robot location are sampled and represented by *particles* $p_i = \langle ep_i, d_i, o_i, x_i, y_i, \theta_i, w_i, wc_i \rangle, p_i \in \mathbb{P}$. The number of particles is $|\mathbb{P}|$. Each particle p_i has the following properties:

- ep_i, the edge that the particle is associated with.
- d_i, the projected location of the particle on the edge.
- o_i, the offset of the location of the particle from the edge.
- x_i, y_i the Cartesian location of the particle on the map with respect to a global reference frame.
- θ_i, the orientation of the particle with respect to the global reference frame.
- w_i, the normalized weight assigned to the particle.
- wc_i, the map constrained weight assigned to the particle. This weight is calculated in the *Constrain* step of the Run-Time phase, as described in Section III-C.3.

The properties of the particle are graphically illustrated in Fig. 4.

C. Particle Filter Implementation : Updating the Location Hypotheses

The location hypotheses are updated iteratively when new data is available. The four steps involved in the update are the *Predict* step, the *Update* step, the *Constrain* step and the *Resample* step. The Predict step is executed whenever new odometry data from the robot is available, and updates

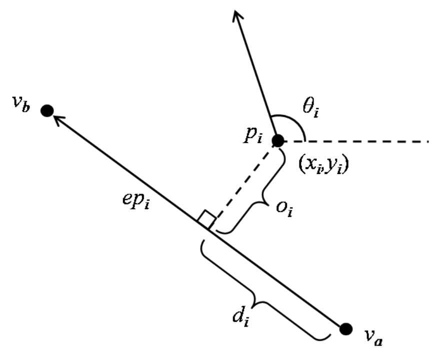

Fig. 4. Properties of a particle p_i: Associated edge ep_i ($ep_i = \langle v_a, v_b, \Gamma \rangle$), Projected location d_i, Offset o_i, Cartesian location (x_i, y_i), Orientation θ_i

the positions and orientations of the particles. The Update step is executed every 500ms when new WiFi signal strength data is available, and updates the weights of the particles. The Constrain step is executed whenever the Predict step is executed, and updates the weights and edge associations of the particles based on map data and constraints. After every Update iteration, the particles are re-sampled in the Resample step.

1) Predict: Given new odometry data of the motion of the robot, for each particle p_i, its properties θ_i,x_i,y_i,d_i,o_i are updated using the motion model of the robot, where the robot's linear and angular motions are modeled as having Gaussian error.

2) Update: To update the weights of the particles based on the latest WiFi signal strength observation, the estimated observation model as described in Section III-A is used. For each particle p_i, the observation probability for that location $P(S|l)$ is calculated using the location of that location in equation 3.

3) Constrain: Following the Update step, the edge association of each particle is re-evaluated, and the map constrained weights computed as outlined by the pseudocode in Algorithm 1. Here, the $thresh$ term is a threshold term, which is set to 0.1. Prior to renormalizing the particle weights, the sum of the updated and constrained weights, wc_{sum} is computed, which is required for the next step, Resample.

4) Resample: After every Update step, the particles need to be resampled. The number of particles to be resampled N is calculated based on Sensor Resetting Localization [12], subject to a maximum N_{max} and minimum N_{min}. In addition to this, particles with weight less than a threshold wc_{min} are also resampled. Algorithm 2 implements this. All resampled particles are re-initialized to location x_i, y_i with probability $P(S|(x_i, y_i))$, based on the latest WiFi signal strength observation.

D. Inference of Location

For inferring the location of the robot based on the particle set $\mathbb{P}$, we perform K-Means clustering of the particles p_i by modifying the algorithm of [13] to take into account the weights wc_i of the particles. The reported location of the robot is then the location of the cluster with the largest weight. During this inference step, we also estimate the

Algorithm 1 'Constrain' Algorithm

```
1:  $wc_{sum} \leftarrow 0$
2:  for $i$ = 1 to $|\mathbb{P}|$ do
3:      Let $p_i = \langle ep_i, d_i, o_i, x_i, y_i, \theta_i, w_i, wc_i \rangle$
4:      $fr \leftarrow d_i$/(length of edge $ep_i$)
5:      if $fr > 1 - thresh$ or $fr < thresh$ then
6:          Find edge $e$ best associated with $p_i$
7:          if $e! = ep_i$ then
8:              $ep_i \leftarrow e$
9:              Calculate new $d_i$
10:             $o_i \leftarrow 0$
11:         end if
12:     end if
13:     $width_i \leftarrow$ width of edge $ep_i$
14:     if $|o_i| > width_i$ then
15:         $wc_i \leftarrow w_i * \exp(-(abs(o_i) - width_i/2)^2)$
16:     else
17:         $wc_i = w_i$
18:     end if
19:     $wc_{sum} \leftarrow wc_{sum} + wc_i$
20: end for
21: Renormalize weights $wc_i$ such that $\sum_{i=1}^{i=|\mathbb{P}|} wc_i = 1$
```

Algorithm 2 'Resample' Algorithm

```
1:  $N \leftarrow 0$
2:  Sort Particles $p_i$ in increasing order of $wc_i$
3:  for $i$ = 1 to $|\mathbb{P}|$ do
4:      Let $p_i = \langle ep_i, d_i, o_i, x_i, y_i, \theta_i, w_i, wc_i \rangle$
5:      if $wc_i < wc_{min}$ then
6:          $N \leftarrow N + 1$
7:      end if
8:  end for
9:  $N \leftarrow N + |\mathbb{P}| * \min(\frac{N_{max}}{|\mathbb{P}|}, \max(\frac{N_{min}}{|\mathbb{P}|}, wc_{sum}/\kappa))$
10: for $i$ = 1 to $N$ do
11:     Draw $(x_i, y_i)$ with probability $P(S|x_i, y_i)$
12:     $wc_i \leftarrow P(S|x_i, y_i)$
13:     $w_i \leftarrow P(S|x_i, y_i)$
14:     $o_i \leftarrow 0$
15:     Find edge $e$ best associated with $p_i$
16:     calculate $d_i$ for particle $p_i$ on edge $e$
17: end for
```

"uncertainty" (σ) of the location hypothesis as the weighted standard deviation of all the particles in the cluster with the maximum weight. The "confidence" (c) of the location estimate is estimated as the weight of the cluster with the maximum weight.

E. Orientation Estimation

The orientation of the robot could be estimated using the particle filter too, but this would increase the dimension of the hypothesis space, and the number of particles would correspondingly increase. Instead, we use a mixture of dead reckoning with global wall orientation data, similar to the method described in [14]. The property θ_i of each particle p_i is allowed to evolve using the robot's motion model when no wall is detected by the robot. When walls are detected by the robot (using the LIDAR sensor), the property θ_i of every particle p_i is updated using global wall orientation information and the location of the particle.

F. Experimental Evaluation

We evaluate the performance of our WiFi localization algorithm based on a number of parameters.

1) Convergence: In this test, the robot was stopped at 20 randomly chosen locations on the map, and the particles initialized with equal weights randomly distributed across the map. Fig. 5 shows a plot of the mean uncertainty σ in localization with time. It took a mean of 8s for the location uncertainty to converge to less than 1m.

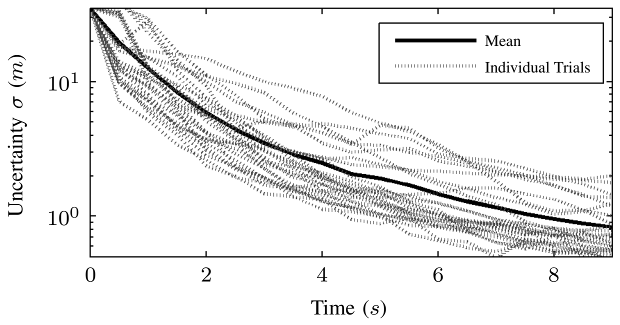

Fig. 5. Uncertainty σ *vs.* t: Convergence of Location Estimates. Black, solid: Mean of all trials. Grey, dotted: Individual trials.

2) Accuracy: In this test, the robot was again stopped at 20 randomly chosen locations on the map, and the particles initialized with equal weights randomly distributed across the map. The location estimates were allowed to converge till the uncertainty of localization σ stopped decreasing. The reported location estimates were then compared to the ground truth. Table I sums up the results of this test.

Value	Mean	Minimum	Maximum
Localization Error	0.7m	0.2m	1.6m
Localization Uncertainty	0.6m	0.2m	0.9m
Convergence Time	11s	7.4s	16.5s

TABLE I
ACCURACY OF WIFI LOCALIZATION

3) "Incidental" Observations: While the robot is performing various tasks, it is reasonable to expect that it would drop WiFi Signal Strength readings from some access points. We wish to investigate the impact of dropped signals on the localization accuracy in this experiment. To do so, the robot is stopped at a fixed location on the map, and is allowed to collect at least one signal strength measurement from all WiFi access points in range. Next we selectively and deliberately drop signal strength measurements from all permutations of the access points to simulate dropped signals. Fig. 6 shows the results of this test. A total of 15 access points were accessible from the location, and even with 9 dropped signals (6 visible access points), the mean error in localization is less than 2m.

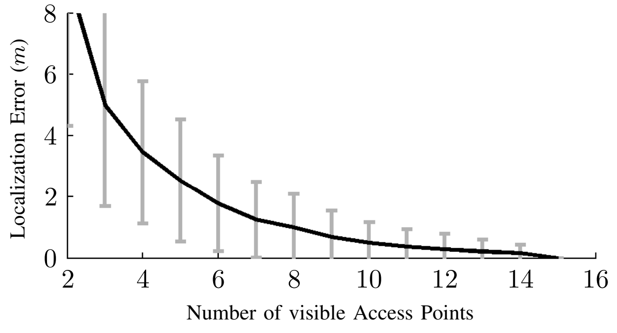

Fig. 6. Localization Error *vs.* Number of visible Access Points

IV. Navigation

We desire the navigation algorithm of the robot to be capable of producing smooth and uninterrupted robot motion. In order to do so, we simplify the robot motion such that it can be split up into the following action primitives:

1) MoveDownCorridor(d, v_s, v_f) : Moves in a straight line between start vertex v_s and end vertex v_f, traveling a maximum distance d.
2) IntegratedTurn$(direction)$: Takes the next available turn (as sensed by the LIDAR) in the desired direction.
3) InPlaceTurn(ϕ) : Turns the robot by ϕ radians in place.

Algorithm 3 outlines the "Next Maximal Action" Navigation Algorithm which generates these action primitives from the robot's current location, given a destination location. It internally calls the following subroutines:

- GetLocation(x,y,σ) : Returns current location (x,y) of the robot, and the uncertainty of localization, σ.
- ComputePolicy(V,E,v_d,Dist,π) : Computes global policy π for destination v_d
- ComputePath(Path,Policy,x,y) : Accepts the current location (x,y) of the robot, and generates a path (Path) to follow in order to reach destination v_d based on the given policy (Policy)
- PathProject(Path,x,y,$v_{current}$,ϵ) : Accepts the current location (x,y) of the robot, and returns the vertex $v_{current}$ ($v_{current}$ $\in$Path) that is closest to the robot's location. Also returns the error of projection as ϵ
- Execute(a,$actionFail$) : Executes action primitive a and sets failure indicator $actionFail$ when execution of a fails.

V. Experimental Results

Our WiFi localization and navigation algorithms were tested using CoBot, deployed on the third floor of Wean Hall at Carnegie Mellon University.

A. Parameters Of The Tests and Test Methodology

The graph representation of the map built during the Learning Phase had 223 vertices and 222 edges. There were a total of 106 unique WiFi access points. The particle filter used 500 particles that were initialized with random orientations and locations, and equal weights.

Algorithm 3 "Next Maximal Action" Navigation Algorithm

```
procedure NAVIGATE($V$,$E$,$v_d$)
    Dist,$\pi$,$v_{current}$,$x$,$y$,Path,pathProgress $\leftarrow$ null
    $\epsilon, \sigma, T \leftarrow 0$
    $actionFail \leftarrow$ false
    COMPUTEPOLICY($V$,$E$,$v_d$,Dist,$\pi$)
    GETLOCATION($x$,$y$,$\sigma$)
    COMPUTEPATH(Path,$\pi$,$x$,$y$)
    PATHPROJECT(Path,$x$,$y$,$v_{current}$,$\epsilon$)
    pathProgress $\leftarrow$ Path(0)
    while $v_{current} \neq v_d$ do
        $a \leftarrow$ next action primitive from $\pi(v_{current})$
        Compute $V_a$ for action primitive $a$
        EXECUTE($a$,$actionFail$)
        GETLOCATION($x$,$y$,$\sigma$)
        PATHPROJECT(pathProgress,$x$,$y$,$v_{current}$,$\epsilon$)
        if $\epsilon > \epsilon_{max}$ or $actionFail =$ true then
            $T \leftarrow 0$
            while $\sigma > \sigma_{max}$ and $T < T_{max}$ do
                Halt $T_{halt}$ seconds
                $T = T + T_{halt}$
                GETLOCATION($x$,$y$,$\sigma$)
            end while
            COMPUTEPATH(Path,$\pi$,$x$,$y$)
            PATHPROJECT(Path,$x$,$y$,$v_{current}$,$\epsilon$)
            pathProgress $\leftarrow$ Path(0)
        else
            pathProgress $\leftarrow$ pathProgress + $V_a$
        end if
    end while
end procedure
```

In order to test the localization and navigation system, we ran a series of experiments, each of which were conducted as follows. At the start of each experiment, a list of 12 random locations over the map were generated such that no two successive locations were from the same corridor, and the robot had to autonomously navigate to these locations sequentially.

B. Results

Each experiment lasted for an average of 28 minutes, and covered an average distance of 818 meters. In total, this experiment was repeated 8 times, adding up to over 3.5 hours of robot navigation time, and a total path length of over 6.5 km. Out of these runs, for 2 of these runs, while the robot navigated between these locations, its true location was manually recorded periodically to use as ground truth for comparison with the localization estimates.

Fig. 7 shows a trace of the path followed by the robot during one of the experiments. This trace was reconstructed using the localization estimate of the WiFi localization algorithm. Fig. 8 shows the evolution of the robot's uncertainty σ over the first ten minutes of this experiment. During all the 8 experiments, the robot encountered a total of 33 action

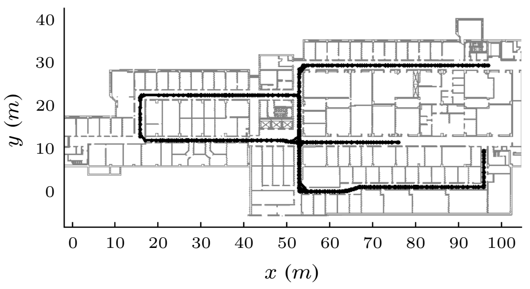

Fig. 7. Trace of the path traversed by the robot

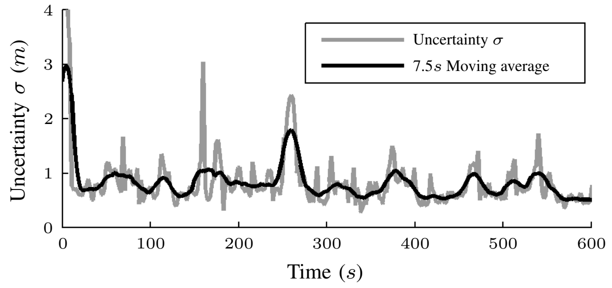

Fig. 8. σ *vs.* t: Uncertainty of localization with time

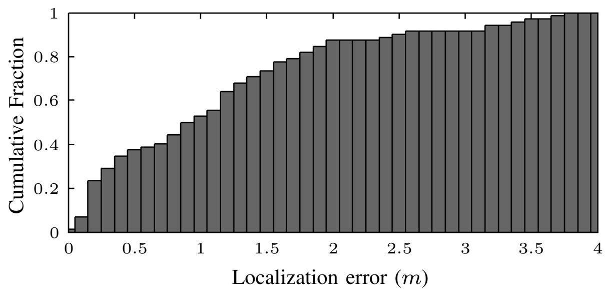

Fig. 9. Cumulative histogram of localization error

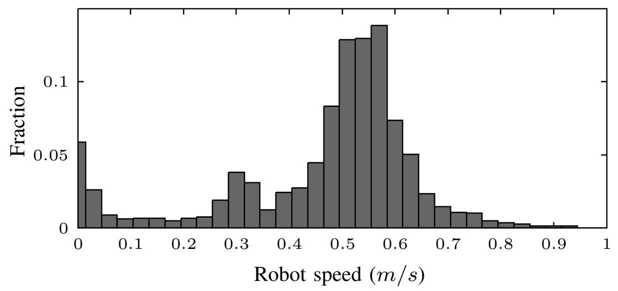

Fig. 10. Histogram of the speed of the robot during the experiments

failures, but autonomously recovered from all of them. Fig. 9 shows a cumulative histogram of the location accuracy of the robot for the 2 experiments with the manually annotated ground truth data. The robot's localization error is a mean of $1.2m$, and the error is less than $1.8m$ for greater than 80% of the time. It is worth noting that while the robot is in motion, the localization has more errors compared to when the robot was stopped (Section III-F). This can be attributed to odometry errors, latency in signal strength readings, and unobserved signal strengths. Fig. 10 shows a histogram of the robot's speed during the experiment. The largest peak around $0.57m/s$ corresponds to the straight line speed (the variations being due to obstacle avoidance), and the smaller peak around $0.3m/s$ is the turning speed of the robot.

VI. Conclusion

In this paper, we introduced an algorithm using WiFi signal strength measurements for the localization of an indoor mobile robot on a map as represented by a graph. The data collected during the Learning Phase of the algorithm was used to generate a perceptual model for the robot's location hypotheses, which along with odometry data and map constraints constituted our localization algorithm. We introduced our "Next Maximal Action" navigation algorithm, and demonstrated the simultaneous functioning of the localization and the navigation algorithms through extensive testing: a total traversed distance of over 6.5km and a total robot navigation time in excess of 3.5 hours.

Acknowledgments

This work was partially supported by the Computational Thinking Center at Carnegie Mellon, and by the Lockheed Martin, Inc. under subcontract 8100001629/1041062, and by Intelligent Automation, Inc, under subcontract 654-1. The views and conclusions contained in this document are those of the authors only.

References

[1] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of robotics and automation*, 3(3):249–265, 1987.

[2] J.S. Gutmann and C. Schlegel. AMOS: Comparison of Scan Matching Approaches for Self-Localization in Indoor Environments. In *Proceedings of EUROBOT*, volume 96, page 61, 1996.

[3] R. Sim and G. Dudek. Learning and evaluating visual features for pose estimation. In *Proceedings of the Seventh International Conference on Computer Vision (ICCV99)*, 1999.

[4] O. Serrano, J.M. Canas, V. Matellan, and L. Rodero. Robot localization using WiFi signal without intensity map. *WAF04, March*, 2004.

[5] M. Ocana, LM Bergasa, MA Sotelo, J. Nuevo, and R. Flores. Indoor Robot Localization System Using WiFi Signal Measure and Minimizing Calibration Effort. In *Proceedings of the IEEE International Symposium on Industrial Electronics*, pages 1545–1550, 2005.

[6] P. Bahl and V. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *IEEE infocom*, pages 775–784, 2000.

[7] Y.C. Chen, J.R. Chiang, H. Chu, P. Huang, and A.W. Tsui. Sensor-assisted wi-fi indoor location system for adapting to environmental dynamics. In *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 118–125, 2005.

[8] B. Ferris, D. Haehnel, and D. Fox. Gaussian processes for signal strength-based location estimation. In *Proceedings of Robotics: Science and Systems*, August 2006.

[9] S. Zickler and M. Veloso. RSS-Based Relative Localization and Tethering for Moving Robots in Unknown Environments. In *IEEE International Conference on Robotics and Automation*, 2010.

[10] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 1322–1328, 1999.

[11] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. *Sequential Monte Carlo Methods in Practice*, pages 499–516, 2001.

[12] S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 1225–1232, 2000.

[13] P.S. Bradley and U.M. Fayyad. Refining initial points for k-means clustering. In *Proceedings of the 15th International Conference on Machine Learning*, pages 91–99, 1998.

[14] S. Thrun, A. Bucken, W. Burgard, D. Fox, T. Frohlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. *AI-based Mobile Robots: Case studies of successful robot systems*, 1998.