Voronoi Tracking: Location Estimation Using Sparse and Noisy Sensor Data

Lin Liao, Dieter Fox, Jeffrey Hightower, Henry Kautz, and Dirk Schulz

Deptartment of Computer Science & Engineering
University of Washington
Seattle, WA

Abstract

Tracking the activity of people in indoor environments has gained considerable attention in the robotics community over the last years. Most of the existing approaches are based on sensors which allow to accurately determine the locations of people but do not provide means to distinguish between different persons. In this paper we propose a novel approach to tracking moving objects and their identity using noisy, sparse information collected by id-sensors such as infrared and ultrasound badge systems. The key idea of our approach is to use particle filters to estimate the locations of people on the Voronoi graph of the environment. By restricting particles to a graph, we make use of the inherent structure of indoor environments. The approach has two key advantages. First, it is by far more efficient and robust than unconstrained particle filters. Second, the Voronoi graph provides a natural discretization of human motion, which allows us to apply unsupervised learning techniques to derive typical motion patterns of the people in the environment. Experiments using a robot to collect ground-truth data indicate the superior performance of Voronoi tracking. Furthermore, we demonstrate that EM-based learning of behavior patterns increases the tracking performance and provides valuable information for high-level behavior recognition.

1 Introduction

Over the last years, the estimation of the location of people in indoor environments has gained increased attention in the robotics community [5; 13; 9]. This is mainly due to the fact that knowledge about the positions and motion patterns of people can help mobile robots to better interact with people, as stated in [1]. Most existing approaches to people tracking rely on laser range-finders [5; 1; 13] or cameras [9]. A key advantage of these sensors is their location accuracy. Unfortunately, they do not provide information about the identity of people. Recently, especially the ubiquitous computing community has started to equip indoor environments with networks of sensors that are capable of providing information about a person's location and identity [8]. Such sensors, however, have the disadvantage that they provide only relatively coarse location information. In addition to being corrupted by noise, such sensors provide measurements at low frame rates only.

Even though id-sensors do not allow accurate position estimation, they can be used to keep track of a person's location at a more abstract level such as which room or hallway she is in. Such discrete, abstract location information additionally provides an ideal representation for learning patterns in a person's long term behavior. In contrast, pattern discovery in continuous space trajectories often requires supervised learning methods [1].

Based on these observations, we introduce a novel approach to estimating the locations of people using sparse and noisy sensor data collected by id-sensors. The key idea of our approach is to track the locations of people on Voronoi graphs [3], which allow us to naturally represent typical human motion along the main axes of the free space. The estimated trajectories on the Voronoi graph help us to bridge the gap between continuous sensor data and discrete, abstract models of human motion behavior. In contrast to existing localization approaches using discrete, abstract representations [14; 3], our method provides accurate estimates along the continuous edges of the graph. We introduce a two-level approach in which a particle filter uses a switching state space model for human motion, and the Voronoi graph guides the particles through the high-level transition model of the graph structure. We additionally show how to apply EM to learn typical motion patterns of humans in a completely unsupervised manner. The transition probabilities learned from real data significantly increase the tracking quality of the approach. Our experiments show that tracking on the Voronoi graph significantly outperforms particle filters in the unconstrained state space (arbitrary position and orientation).

This paper is organized as follows. In the next section, we will derive Voronoi tracking starting from the general Bayes filter. Then, in Section 3, we show how to learn the parameters of the tracking model using expectation maximization. Before concluding in Section 5, we show experimental results supporting the superior performance of Voronoi tracking.

2 Voronoi Tracking

We define a Voronoi graph $G = (V, E)$ by a set V of vertices v_i and a set E of directed edges e_j. Figure 1(a) shows the Voronoi graph representing the Intel Research Lab Seattle, our indoor testing environment. Note that this graph results from manual pruning of the original Voronoi graph [3] and that for clarity only the undirected version of the graph is shown.

In the following we phrase the problem of location estimation on Voronoi graphs as a special case of Bayesian filtering, on which virtually all probabilistic location esti-
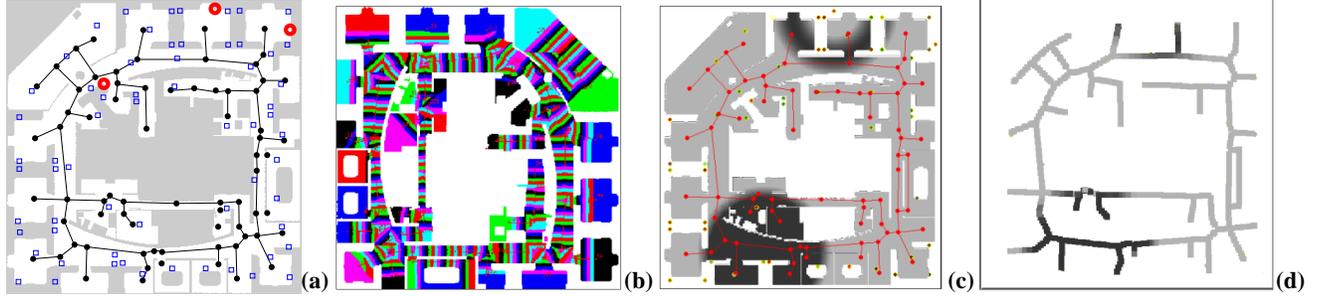
Figure 1: *Voronoi graphs for location estimation: (a) Indoor environment along with manually pruned Voronoi graph. Shown are also the positions of ultrasound Crickets (circles) and infrared sensors (squares). (b) Patches used to compute likelihoods of sensor measurements. Each patch represents locations over which the likelihoods of sensor measurements are averaged. (c) Likelihood of an ultra-sound cricket reading (upper) and an infrared badge system measurement (lower). While the ultra-sound sensor provides rough distance information, the IR sensor only reports the presence of a person in a circular area. (d) Corresponding likelihood projected onto the Voronoi graph.*

mation approaches are based. We will start by describing general Bayes filters for the full continuous state space, and we will then show how to project the different quantities of the Bayes filter onto the topological structure represented in a Voronoi graph.

2.1 Bayesian Filtering on a Voronoi Graph

Bayes filters estimate posterior distributions over the state x_t of a dynamical system conditioned on all sensor information collected so far:

$$p(x_t \,|\, z_{1:t}) \propto p(z_t \,|\, x_t) \int p(x_t \,|\, x_{t-1}) \, p(x_{t-1} \,|\, z_{1:t-1}) dx_{t-1} \quad (1)$$

Here $z_{1:t}$ is the history of all sensor measurements obtained up to time t. In the context of location estimation, the state typically describes the position and velocity of the object in $\langle x, y, \theta \rangle$-space. The term $p(x_t \,|\, x_{t-1})$ is a probabilistic model of the object dynamics, and $p(z_t \,|\, x_t)$ describes the likelihood of making observation z_t given the location x_t and a map of the environment.

Let us now describe how to implement the recursive Bayes filter for Voronoi graphs. We represent the state x_t of an object by a triple $x = \langle e, d, m \rangle$, where e denotes the current edge on the graph, d indicates the distance of the object from the start vertex of edge e, and $m \in \{$stopped, moving$\}$ indicates the current motion state of the object. A straightforward implementation of the observation model for Voronoi graphs would be to simply compute the likelihood of observations for positions on the graph. However, this is not a valid approach since it does not consider the fact that the Voronoi graph projects the 3d state space onto the one-dimensional graph G. Hence, to compute the likelihood of an observation z given a position x on the graph, we have to integrate over all 3d positions projected onto x:

$$p(z|x) = \int_{\nu \in \mathcal{S}(x)} p(z \mid \nu) \, p(\nu \mid x) \, d\nu, \quad (2)$$

Here, $\mathcal{S}(x)$ denotes the set of all positions ν projected onto x. In the limit, $\mathcal{S}(x)$ are the states on the line from x to the closest objects defining the Voronoi graph. In our implementation of the sensor model, we discretize positions on

the graph, which results in location *patches* $\mathcal{S}(x)$, as illustrated in Figure 1(b)–(d).

The motion model $p(x_t \,|\, x_{t-1})$ has to take into account that the objects are constrained to moving on the graph. Here, we adopt a switching state model approach based on the assumption that objects either do not move at all ($m_t =$ stopped), or move ($m_t =$ moving) [1] with a velocity v governed by a Gaussian process with mean v variance σ_v^2. Additionally, have to consider motion along an edge of the Voronoi graph as well as motion from one edge to another. Let us first describe the simpler case of motion on one edge, which we denote $\hat{p}(x_t \mid x_{t-1})$:

$$\hat{p}(x_t \,|\, x_{t-1}) = p(m_t = \text{move} \,|\, m_{t-1}) \cdot \mathcal{N}\left(\frac{x_t - x_{t-1}}{\Delta t}; v, \sigma_v\right)$$
$$+ p(m_t = \text{stopped} \,|\, m_{t-1}) \cdot \delta(x_t, x_{t-1}) \quad (3)$$

Here, $\delta(x_t, x_{t-1})$ is the Dirac delta function which is 1, if $x_t = x_{t-1}$ and 0 otherwise. In Section 3 we will show how to use data to learn $p(m_t \,|\, m_{t-1})$, the switching between motion states. $x_t - x_{t-1}$ denotes the distance function on the Voronoi graph. It can be defined as

$$x_j - x_i = \begin{cases} d_j - d_i & \text{if } e_i = e_j \\ |e_i| - d_i + d_j & \text{otherwise} \end{cases} \quad (4)$$

The term $|e_i| - d_i + d_j$ computes the distance to the end of edge e_i plus the distance d_j on the next edge. For clarity, we restrict the definition to the case of motion between neighboring edges e_i and e_j, but it can easily be extended to the general case of multi-edge motion steps.

To compute the probability of motion from one edge to another, we assume that the Voronoi graph is annotated with transition probabilities $p(e_j | e_i)$, which describe the probability that the object transits to node e_j given that the previous node was e_i and an edge transition took place. Without prior knowledge, this probability is distributed uniformly over all neighboring edges of e_i. Combining this graph transition model with the single edge motion model

---

[1] While this model might seem too simplistic, our experiments indicate that it yields good results and the extension to more complex motion models is possible.

given in (3), we get the model for motion on the graph:

$$p(x_t \mid x_{t-1}) = \begin{cases} \hat{p}(x_t \mid x_{t-1}) & \text{if } e_t = e_{t-1} \\ p(e_t \mid e_{t-1})\,\hat{p}(x_t \mid x_{t-1}) & \text{if } e_t \neq e_{t-1} \end{cases} \quad (5)$$

This finalizes our description of how general state space Bayes filters can be projected onto Voronoi graphs. In the next section we will describe how to implement this graph-based model using particle filters.

2.2 Particle Filter Based Implementation

Particle filters provide a sample-based implementation of general Bayes filters [6; 4]. The key idea of particle filters is to represent posteriors over the state x_t by sets S_t of n weighted samples:

$$S_t = \{\langle x_t^{(i)}, w_t^{(i)} \rangle \mid i = 1, \ldots, n\}$$

Here each $x_t^{(i)}$ is a sample (or state), and the $w_t^{(i)}$ are non-negative numerical factors called *importance weights*, which sum up to one. Just like Kalman filters, particle filters apply the recursive Bayes filter update to estimate posteriors over the state space. The basic form of the particle filter updates the posterior according to the following sampling procedure, often referred to as sequential importance sampling with re-sampling (SISR, see also [4]):

Re-sampling: Draw with replacement a random sample $x_{t-1}^{(i)}$ from the sample set S_{t-1} according to the (discrete) distribution defined by the importance weights $w_{t-1}^{(i)}$.

Sampling: Use $x_{t-1}^{(i)}$ to sample $x_t^{(j)}$ from the distribution $p(x_t \mid x_{t-1})$. $x_t^{(j)}$ now represents the density given by the product $p(x_t \mid x_{t-1})p(x_{t-1} \mid z_{1:t-1})$. This density is the so-called *proposal distribution* used in the next step.

Importance sampling: Weight the sample $x_t^{(j)}$ by the importance weight $p(z_t \mid x_t^{(j)})$, the likelihood of the measurement z_t given the state $x_t^{(j)}$.

Each iteration of these three steps generates a sample drawn from the posterior density. After n iterations, the importance weights of the samples are normalized so that they sum up to 1. It can be shown that this procedure in fact approximates the Bayes filter update (1), using a sample-based representation [4].

The application of particle filters to location estimation on a Voronoi graph is rather straightforward. The resampling step does not have to be changed at all. Sampling the motion update has to be done according to (3)–(5). To do so, recall that the state x_{t-1} contains the position on the graph along with the motion state m_{t-1}. Let us begin with (3). We first sample the motion state m_t with probability proportional to $p(m_t \mid m_{t-1})$. If $m_t = $ moving, then we randomly draw the traveled distance d from the Gaussian distribution given in (3). For this distance d, we have to determine whether the motion along the edge results in a transition to another edge. If not, then $d_t = d_{t-1} + d$ and $e_t = e_{t-1}$. Otherwise, $d_t = d - |e_{t-1}| + d_{t-1}$ and the next edge e_t is drawn with probability $p(e_t \mid e_{t-1})$. Otherwise, if the motion state $m_t = $ stopped, the position x_t

is set to be x_{t-1}. After these sampling steps, the resulting states are distributed according to $p(x_t \mid x_{t-1})$. The importance sampling step of the particle filter is implemented by weighting each sample proportional to the projected observation likelihood as given in (2).

To summarize, we described how to perform recursive Bayesian filtering by projecting the general Bayes filter onto Voronoi graphs, followed by a sample-based implementation. In the next section, we will describe how to learn the motion patterns of individual people.

3 Parameter Learning

One of the key applications of graph-based location estimation is the collection of long-term data so as to learn behavior models of individual people. Learning parameters for a specific person not only increases the accuracy of tracking, but also allows us to understand different motion patterns for different people. As a first step in this direction, we will now describe how to learn the parameters of our Voronoi model using data collected by an object moving through the environment. The parameters Θ of the model consist of the transition probabilities on the Voronoi graph, $p(e_i \mid e_j)$, the switching parameters of the motion model, $p(m_t \mid m_{t-1})$, and the Gaussian motion parameters (v, σ_v^2).[2] To learn these parameters from data, we need an estimate of the robot's location at each point in time. Unfortunately, these positions are not directly observable but have to be estimated from the sensor data. To solve this problem, we apply the EM (Expectation-Maximization) algorithm [12], which is the most widely used approach to solving learning problems with missing features. EM is an iterative algorithm which has an E-step and an M-step at each iteration. In a nutshell, each E-step estimates the trajectory of the person through the environment. This is done using the Voronoi tracking approach along with the model parameters learned in the previous iteration of EM. The particle trajectories of the E-step are used to generate counts for transitions on the graph. These transition counts are then used in the M-step to update the model parameters.

3.1 E-Step:

In the E-step, we update the posterior distribution over the trajectories of the person and compute the expectation of the log-likelihood (see [12; 10] for details). We define:

$$Q(\Theta, \Theta^{(i-1)}) \quad (6)$$
$$= E[\log p(z_{1:t}, x_{1:t} \mid \Theta) \mid z_{1:t}, \Theta^{(i-1)}]$$
$$= \int_{x_{1:t}} \log p(z_{1:t}, x_{1:t} \mid \Theta) p(x_{1:t} \mid z_{1:t}, \Theta^{(i-1)}) dx_{1:t} \quad (7)$$

Here $x_{1:t}$ and $z_{1:t}$ are the sequences of states and observations, respectively. Θ are the parameters of the Voronoi graph-based model we want to estimate and $\Theta^{(i-1)}$ are the estimation thereof at the $i-1$-th iteration of the EM algorithm. The difficulty here is to estimate the second term

---

[2] We do not learn the observation likelihood (2), since it can be extracted directly from the general model of the sensors.

of (7), which is the posterior distribution over state trajectories $x_{1:t}$ given observations $z_{1:t}$ and model parameters $\Theta^{(i-1)}$. To estimate these trajectories, we do particle filtering using the motion and graph transition model with parameter $\Theta^{(i-1)}$. Smoothing of the trajectories is done by performing a forward and a backward filtering pass through the data. Then we multiply the counts resulting from the two distributions at corresponding time slices, which corresponds to the Baum-Welch algorithm widely used for Hidden Markov Models [12]. The resulting estimate is a sample-based approximation for $p(x_{1:t} \mid z_{1:t}, \Theta^{(i-1)})$. More specifically, (7) is approximated by

$$Q(\Theta, \Theta^{(i-1)}) \approx \frac{1}{m} \sum_{j=1}^{m} \log p(z_{1:t}, x_{1:t}^{(j)} | \Theta), \qquad (8)$$

where m is the number of particles and $x_{1:t}^{(j)}$ is the state history of the j-th particle, estimated using the data. For simplicity, we assume that all the particles have equal weight, *i.e.* after they are resampled. It is straightforward to extend our derivation to the case of different weights.

Knowing the state of the particles at the different points in time, it is straightforward to extract information needed for the M-step. The graph transition parameters $p(e_t | e_{t-1})$ can be estimated by counting the number of particles that move from one edge e_i to another edge e_j. To generate such counts, we discretize time into intervals of equal size Δt and sum the number of transitions during each interval. We additionally generate counts for $m_{ijk}(t)$, the number of particles on edge e_i that switch from one motion state m_j into another other motion state m_k. The reason for learning different switching models for each edge is that we want to be able to determine where a person typically stops for extended periods of time. To estimate the parameters of the Gaussian motion model, we store the velocities of all samples that are in the "moving" state. The generation of counts turns out to be very efficient since at each time we only need to save the aggregate information for each edge, not for each particle.

Our approach is in fact very similar to the Monte Carlo EM algorithm [16]. The only difference is that we allow particles to evolve with time. It has been shown that when m is large enough, Monte Carlo EM estimation converges to the theoretical EM estimation [10].

3.2 M-step:

The goal of the M-step is to maximize the expectation we computed in the E-step by updating the parameter estimations. From (8), we have:

$$\Theta^{(i)} = \underset{\Theta}{\arg\max}\, Q(\Theta, \Theta^{(i-1)})$$

$$= \underset{\Theta}{\arg\max} \sum_{j=1}^{m} \log p(z_{1:t}, x_{1:t}^{(j)} \mid \Theta)$$

$$= \underset{\Theta}{\arg\max} \sum_{j=1}^{m} (\log p(z_{1:t} | x_{1:t}^{(j)}) + \log p(x_{1:t}^{(j)} | \Theta)) \quad (9)$$
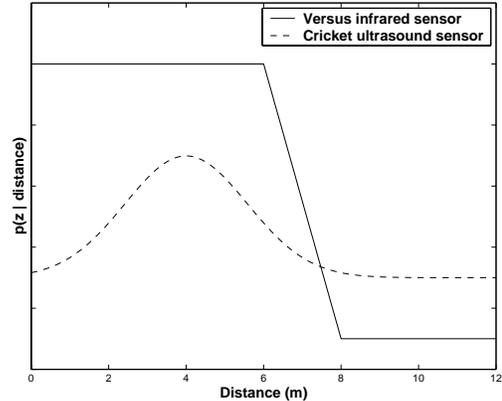

Figure 2: *Sensor model of ultrasound crickets and infrared badge system. The x-axis represents the distance from the detecting infrared sensor and ultrasound sensor (4m ultrasound sensor reading), respectively. The y-axis gives the likelihood for the different distances from the sensor.*

$$= \underset{\Theta}{\arg\max} \sum_{j=1}^{m} \log p(x_{1:t}^{(j)} \mid \Theta) \qquad (10)$$

Here, (9) follows from the independence condition $p(z_{1:t} \mid x_{1:t}^{(j)}, \Theta) = p(z_{1:t} \mid x_{1:t}^{(j)})$, *i.e.* observations are independent of model parameters if the state trajectory is known. To maximize the parameter set Θ, the M-step essentially converts the frequency counts obtained in the E-step into probabilities. The parameters of the Gaussian motion model are the mean and variance of the velocity values.

To summarize, EM iterates between an expectation step, which estimates the trajectory of the person using particle filtering forwards and backwards through the data set. The trajectories are used to count the transitions of particles on the Voronoi graph, the switching between motion modes, and the velocities of particles. These values are then converted into probabilities during the M-step, which generates a new model estimate. The updated model is then used in the next iteration to re-estimate the trajectory of the person. For the first E-step, we initialize $\theta^{(0)}$ with some reasonable values using background knowledge of typical human motion and a uniform distribution for the outgoing edges at each vertex of the Voronoi graph. The algorithm stops if $\Theta^{(i)}$ and $\Theta^{(i-1)}$ are close enough.

4 Experiments

We evaluated the performance of the Voronoi tracking (VT) approach based on data recorded at the Intel Research Lab Seattle (Figure 3). The office environment is equipped with two different kinds of id sensors: 73 Versus infrared receivers which provide information about the presence of a person in the vicinity of a sensor, and three Cricket ultrasound receivers, which provide identity and distance estimates (see [8] and references therein). Figure 2 shows a model of the uncertainty of the two sensor types. Both sensors suffer from a high probability of false-negative readings, *i.e.* they frequently fail to detect a person.
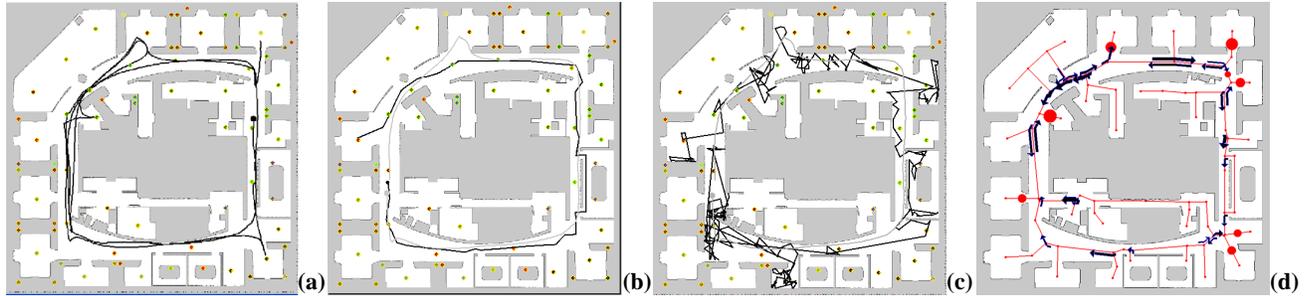
Figure 3: *(a) Trajectory of the robot during a 25 minute period of training data collection. True path (in light color) and most likely path as estimated using (b) Voronoi tracking and (c) original particle filters. (d) Motion model learned using EM. The arrows indicate those transitions for which the probability is above 0.65. Places with high stopping probabilities are represented by disks. Thicker arcs and bigger disks indicate higher probabilities.*
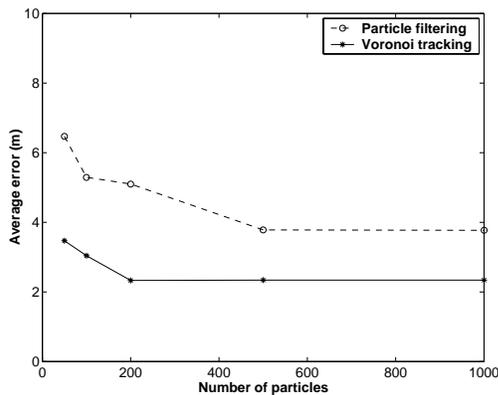

Figure 4: *Localization error for different numbers of samples.*

To generate data for which ground truth is available, we equipped a mobile robot with two Versus badges, a Cricket beacon, and additionally with a Sick laser range-finder. Our experiment is based on a 35-minute-long log of sensor measurements received while driving the robot through the environment. The robot moved at an average velocity of 30 cm/s and was stopped at designated resting places. The laser range-finder allowed us to accurately estimate the path of the robot using the map shown in Figure 1(a).

As an initial test, we determined the average localization error when using un-trained VT (see video) *vs.* a particle filter (PF) representing locations in the complete free space of the environment. The resulting error on the complete log was 2.34m for VT and 3.78m for PF. This result is extremely encouraging since it indicates that the projection onto the Voronoi graph does not only result in a good approximation of the PF, but yields *better* performance than the PF. Figure 3(b) and (c) show typical maximum likelihood trajectories using VT and PF, respectively [3]. The graphs clearly demonstrate the superior performance of the VT technique [4]. We also compared the tracking perfor-

---

[3]These paths were generated from the trajectory (history) of the most likely particle at the end of the run. For clarity, only part of the trajectory is shown

[4]Please visit www.cs.washington.edu/homes/liaolin /Research/voronoi_tracking.html for videos on using both techniques.

mance for different sample set sizes. The results are given in Figure 4. It becomes clear that VT makes very efficient use of particles and it is able to track well using only 200 particles.

Next, we evaluated the learning performance of VT. We split the complete data log into a training set of 25 minutes and a test set of 10 minutes. We trained the VT model using the EM algorithm described in Section 3 on the training set and determined the tracking error on the test data using the motion model learned at each iteration. The results are summarized in Table 1.

As can be seen, the algorithm converges after only 3 it-

EM Iteration	Avg. Tracking Errors (m)	Error Reduction after Learning
before learning	2.63	-
1	2.27	13.7%
2	2.05	22.1%
3	1.84	30.0%
4	1.79	31.9%
5	1.76	33.1%

Table 1: *Evolution of test error during EM learning.*

erations of EM, and using the learned model increases the tracking accuracy significantly. This shows that the Voronoi graph is able to extract the correct motion patterns from the training data. This fact is supported by Figure 3(d), which visualizes the motion model learned after 5 iterations. The model correctly reflects the path and resting locations of the robot.

5 Conclusions and Future Work

We have presented a novel approach to tracking the location of people in indoor environments. The technique is able to robustly estimate a person's location even when using only sparse, noisy information provided by id-sensors. The key idea of our approach is to use a particle filter that is projected onto a Voronoi graph of the environment. The resulting model is a two-level technique, where, on the lower level, the particle filter estimates the location of people along the continuous edges of the graph. At the next level, the particles generate transitions on the discrete Voronoi

graph. These two levels are highly connected, since transitions on the Voronoi graph help to predict the motion of particles through the graph structure. On the other hand, particle motion helps to estimate and learn the discrete transition model of the graph. Learning is achieved by EM, which concurrently estimates the model parameters at both levels.

Using data collected by a mobile robot, we demonstrate that our approach has two key advantages. First, it is by far more efficient and robust than particle filters which estimate the location of people in the complete free space of an environment. Second, the Voronoi graph provides a natural discretization of human motion, which allows us to learn typical motion patterns using expectation maximization. We show that the EM-based learning of behavior patterns indeed increases the performance of the approach.

The work presented here is related to tracking a person's or vehicle's location using outdoor GPS data. Most state of the art GPS tracking systems project GPS readings onto graph-based maps describing streets, walkways *etc*. Recently, several researchers suggested using Kalman filters to integrate GPS sensor readings over time [15; 2]. A disadvantage of these approaches is that they only represent unimodal distributions, which is insufficient for very noisy sensor information. Furthermore, at vertices of the Voronoi graph, belief distributions are clearly not Gaussian and we suspect that even mixtures of Gaussians are not well suited to represent very uncertain distributions on Voronoi graphs. We recently applied our technique successfully to the task of learning the motion patterns of a person using data collected by a GPS sensor [11]. For example, the approach is able to learn where a person typically gets on or off a bus.

Despite these very encouraging results, there are still warrants for future research. First, we will collect long-term data from people working in the environment. This data will be used to estimate the parameters of the model for each person. Long-term data logs will allow us to determine whether the model is able to detect different motion patterns for the different people. So far, our Voronoi model only captures first-order transitions. This can only be the first step towards high-level pattern learning. For example, most navigation patterns depend on the time of day and the (potentially far away) goal of the person and not only on the previous position. We intend to extend our technique to hierarchical dynamic Bayesian networks (DBN) [7]. We expect these networks to be able to use information provided by the Voronoi graph to learn long term activities at higher levels of the hierarchy.

Despite these limitations, we believe that this is a first step towards unsupervised learning of behavior patterns.

Acknowledgments

References

[1] M. Bennewitz, W. Burgard, and S. Thrun. Using EM to learn motion behaviors of persons with mobile robots. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[2] P. Bonnifait, P. Bouron, P. Crubillé, and D. Meizel. Data fusion of four ABS sensors and GPS for an enhanced localization of car-like vehicles. In *Proc. of the IEEE International Conference on Robotics & Automation*, 2001.

[3] H. Choset. *Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph*. PhD thesis, Caltech, 1996.

[4] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo in Practice*. Springer-Verlag, New York, 2001.

[5] A. Fod, A. Howard, and M.J. Mataric. Laser-based people tracking. In *Proc. of the IEEE International Conference on Robotics & Automation*, 2002.

[6] D. Fox. Adapting the sample size in particle filters through KLD-sampling. *The International Journal of Robotics Research*, 22, 2003.

[7] Z. Ghahramani. *Lecture Notes in Artificial Intelligence*, chapter Learning Dynamic Bayesian Networks, pages 168–197. Springer-Verlag, 1998.

[8] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *Computer*, 34(8), 2001. IEEE Computer Society Press.

[9] E. Kruse and F. Wahl. Camera-based monitoring system for mobile robot guidance. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[10] R. Levine and G. Casella. Implementations of the Monte Carlo EM algorithm. *Journal of Computational and Graphical Statistics*, 10, 2001.

[11] L. Liao, D. Patterson, D. Fox, and H. Kautz. Inferring high-level behavior from low-level sensors. In *Proc. of the International Conference on Ubiquitous Computing*, 2003.

[12] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*. IEEE, 1989. IEEE Log Number 8825949.

[13] D. Schulz, W. Burgard, and D. Fox. People tracking with mobile robots using sample-based joint probabilistic data association filters. *International Journal of Robotics Research (IJRR)*, 22(2), 2003.

[14] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proc. of the International Joint Conference on Artificial Intelligence*, 1995.

[15] R. Thrapp, C. Westbrook, and D. Subramanian. Robust localization algorithms for an autonomous campus tour guide. In *Proc. of the IEEE International Conference on Robotics & Automation*, 2001.

[16] G. Wei and M.A. Tanner. A Monte Carlo implementation of the EM algorithm and the poor mans data augmentation algorithms. *Journal of the American Statistical Association*, 85, 1990.