Improved Indoor Positioning Using the Baum-Welch Algorithm

Noha El Gemayel *, Javier Schloemann †, R. Michael Buehrer†, Friedrich K. Jondral *
* Communications Engineering Lab, Karlsruhe Institute of Technology (KIT), Germany
{noha.gemayel, friedrich.jondral}@kit.edu
† Wireless @ Virginia Tech, Blacksburg, VA, USA
{javier, buehrer}@vt.edu

***Abstract*—In this paper, we examine the exploitation of individual patterns of behavior to enhance indoor positioning of pedestrians. We make use of the fact that, due to habits and needs, a person is likely to be in some locations more often than others. For example, at their work or in their home, a person is likely to spend more time in some rooms than in others. Therefore, it seems natural to take advantage of established behavior patterns when performing indoor localization in an effort to improve accuracy. Such improvement is particularly beneficial during emergencies, where location inaccuracies may lead to life-threatening delays in response times. In this work, habitual behavior is modeled and learned using a hidden Markov model. It is shown that, applying the Markov model for location estimation results in more accurate estimates when compared to using a standard particle filter with odometry information. Additionally, transition probabilities as well as position error distributions do not need to be known *a priori* since they can be learned using the Baum-Welch algorithm. Results show how the Baum-Welch algorithm can even learn the distributions of biased estimates. On the other hand, it is shown how user feedback can help accelerate the learning process, while guaranteeing good parameter estimation accuracies.**

***Keywords*—indoor positioning, hidden Markov models, Baum-Welch**

I. Introduction

Geolocation using RF signals, both indoors and outdoors, has been an active area of research for over 30 years [1]. In recent years, the accuracy of indoor geolocation has become increasingly important due to the widespread use of cellular phones in placing emergency calls. In fact, the Federal Communications Commission (FCC), citing the fact that a majority of E911 wireless calls come from indoor locations, recently published a Third Notice of Proposed Rulemaking concerning accuracy requirements for indoor cell phone calls [2]. In this notice, the commission proposed that cellular phone service providers must locate callers within 50 meters in the horizontal plane and provide vertical (z-axis) data within 3 meters of accuracy in 80 percent of indoor calls within five years of the effective date of adoption of the rules. Some service providers are quite pessimistic about the chances of current cell-phone localization techniques to meet these new requirements [3]. This motivates increased investigation of the indoor geolocation problem, especially for emergency response, since small positioning errors can lead to long life-threatening delays in help. In this paper, our objective is to improve indoor positioning by learning and exploiting a user's established habits regarding their motion inside a building.

A. Motivation and Previous Work

Indoor positioning systems can be based on GPS, RFID, cellular networks, UWB, or WLAN [4]. Motivated by the emergency response scenario, we present a technique that can be applied by cellular systems to enhance indoor positioning without requiring additional hardware. Specifically, we use a hidden Markov model (HMM) (either in the cellular location server or in the phone itself) to model the location behavior of people indoors. HMMs have been used previously to enhance indoor positioning. In [5], the Markov state is modeled as a joint state of position and line-of-sight (LOS)/ non-line-of-sight (NLOS) propagation. Based on a UWB network, the observations used are the power delay profiles or received signal strengths at the available wireless access points. The motion was modeled as relative motion resulting from a known driving process. In [6], the rooms and hallways of a building are modeled as states of an HMM. The transition probabilities are assumed equally likely for all possible transitions. IEEE 802.11 signals are used and the network is trained by recording received signal strength histograms at each state over 24 hours. [7] presents a technique where a person is observed and, considering and learning their intentions, an HMM is automatically derived. Laser and visual data are used to update the model parameters. [8] presents an automatic HMM training by using a robot that collects WiFi and ultrasound data at different parts of the building. The transition matrix is defined equally for all possible transitions, meaning that no pattern of behavior is considered in that model.

It is well known that the main challenge for indoor positioning systems using range measurements is NLOS signal propagation. In order to deal with this challenge, several different methods have been presented. For example, [9] presents a method for identifying and mitigating NLOS using experimental data while [10] models the error using information about the environment.

B. Contributions

The mentioned techniques are all built on a specific positioning system with defined measurements (e.g., received signal strength at access points, visual data, power delay

The first author gratefully acknowledges the support of the Karlsruhe House of Young Scientists (KHYS) for the research visit at Virginia Tech and would like to thank Wireless @ VT for hosting the visit.

profile, etc.) and rely on a defined hardware setup. The listed methods are also based on the relative motion behavior of a person (i.e., in one room) and not on a pattern of behavior based on the different rooms and hallways of the building. In this work, we consider habitual behavior (i.e., that some transitions are more likely than others), which is a key difference between our work and that of [6] and [8]. We propose a hidden Markov model that is not restricted to cellular systems and that can be applied to many indoor positioning systems. The presented method improves indoor positioning performance without requiring any additional hardware in the building or in the user device. By introducing transition probabilities to model user movement between rooms based on patterns of user behavior, we will show that the estimation accuracy can be greatly enhanced. Furthermore, the transition probabilities (i.e., the behavior patterns) as well as the error distributions can be *learned* by applying the Baum-Welch (BW) algorithm. As an example, we use a positioning system based on ranging measurements, and model the main problem of NLOS propagation. In order to mitigate position estimation errors introduced due to NLOS propagation, the BW algorithm learns the positioning error distribution. Finally, we show how user feedback can help accelerate the learning process and prevent the proposed algorithm from converging to locally, but not globally, optimum solutions.

II. System Model

A. Motion Model

We model the behavior of a pedestrian inside a large building, for example at the workplace or in an apartment building, using a Markov model with user-specific transition probabilities. The Markov states are the rooms or hallways, or in the case of large rooms/halls, portions of them. Normally, a user has a high probability of staying in some states (e.g., an office) and a very low probability of staying in other states (e.g., a hallway).

As an example of the state transition probabilities, consider the five numbered states in Fig. 1, where states $1, 2$ are the entrance of the building, states $3, 4$ are hallways and state 5 is an office. An example transition matrix for these five states is:

$$\mathbf{A} = \begin{bmatrix} 0.1 & 0.9 & 0 & 0 & 0 \\ 0.05 & 0.05 & 0.9 & 0 & 0 \\ 0 & 0.2 & 0.05 & 0.75 & 0 \\ 0 & 0 & 0.4 & 0.1 & 0.5 \\ 0 & 0 & 0 & 0.05 & 0.95 \end{bmatrix}. \quad (1)$$

Assuming a known floor plan of the building, it is known which state transitions are possible and which are not, but the state transition probabilities are not necessarily known *a priori*. Additionally, we assume that it is known when and where the person enters the building, e.g. by using GPS or some other outdoor localization method. Using this information, the positioning system knowns when to initialize the indoor tracking algorithm as well as the initial state probabilities of the user. Using the transition matrix, the movement of a person between rooms is modeled as a Markov chain. The actual position of the person *within* a state (room/hallway) is modeled using a probability distribution, e.g. here, uniform. How the building is divided into different states is a system design parameter that depends on the positioning scenario, the required accuracy and the available computing capacity. If the aim is to find the room where a person is located, then defining each room as one state would be reasonable.

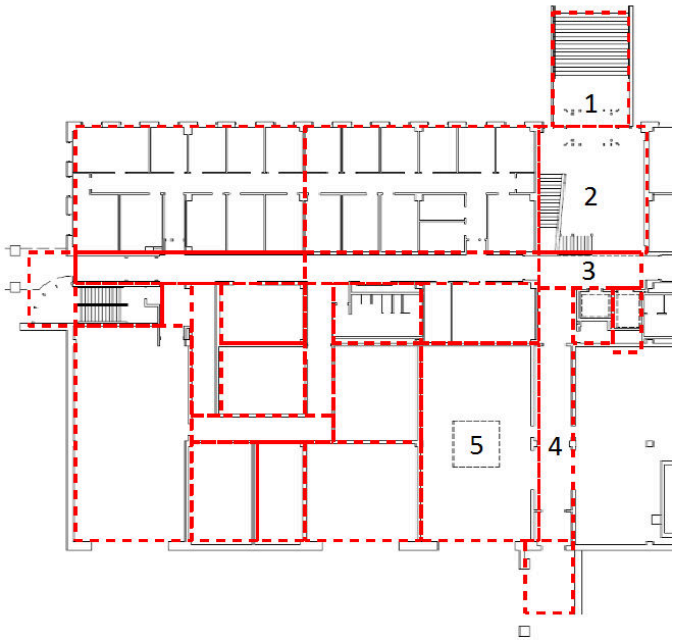

Fig. 1. Durham Hall floor plan with the 27 states that were used for the simulations marked by red rectangles and 5 states picked for illustration.

B. HMM-based User Mobility Model

Starting from the previously described movement model, we introduce the hidden Markov model. The HMM with N states representing rooms, hallways, or parts of both is characterized by the following parameters:

- $\mathbf{A}$: $N \times N$ transition matrix; $[A]_{ij} = a_{ij}$ is the transition probability from state i to state j. Note that some elements are known to be zero (since they represent non-adjacent locations), but otherwise the matrix is assumed to be unknown in advance, unless stated otherwise.
- $p_i(\mathbf{o}_t)$: conditional output probability, where $\mathbf{o}_t$ is the observed quantity (here $\mathbf{o}_t \in \mathbb{R}^2$ is the position estimate obtained from a standard position estimator). $p_i(\mathbf{o}_t)$ is the probability of observing $\mathbf{o}_t$ at time t conditioned on being in state i.
- π_i: initial probability of being in state i.

Estimating the Markov state is equivalent to estimating the room or the part of the room where the node (e.g., the mobile device or in the emergency scenario, the cellular user) is located. Therefore, in this analysis, accuracy is quantified through the room-level (or partial room-level) accuracy measure, or the probability that the estimated state is the true state. This is different from most studies, which emphasize the estimation accuracy in meters. Since here we are ultimately only concerned with room-level accuracy, we find error probability to be a more useful metric.

C. Observation Errors

The observations used in the HMM are position estimates that result from applying a positioning algorithm to available measurements (e.g., time of arrival, received signal strength, time difference of arrival, etc.). Thus, the model does not depend on the available measurements or the underlying system, but rather on the expected error distribution of the estimated positions. Since the model is built upon the outcome of the position estimation, it can be regarded as a higher layer estimation and can be applied in various systems.

In this paper, we use time difference of arrival (TDoA) measurements as an example and apply a least squares algorithm [11]. In the two-dimensional case, an observation at time t is: $\mathbf{o}_t = [\hat{x}, \hat{y}]^T$ and its corresponding true position (generated as described in Sections II-A and II-B) is: $\mathbf{x}_t = [x, y]^T$. The position estimates experience errors due to imperfect TDoA measurements. Assuming L TDoA anchors and defining, without loss of generality, anchor 1 as the reference, the estimated range difference between this reference anchor and anchor l, $\hat{d}_{l,1}$, can be modeled as [10]

$$\hat{d}_{l,1} = v \cdot \hat{\tau}_{l,1} = d_{l,1} + b_{l,1} + n_{l,1}, \quad l = 2, ..., L, \tag{2}$$

and

$$d_{l,1} = d_l - d_1, \tag{3}$$

where d_l, d_1 are the distances from the anchors l, 1 to the node (e.g., the cellular user), $d_{l,1}$ is the true range difference between the anchors, v is the propagation speed, $\hat{\tau}_{l,1}$ is the estimated TDoA, $b_{l,1}$ is a bias due to multipath propagation and $n_{l,1}$ is zero-mean Gaussian noise with power σ_{n}^2. Assuming that the biases do not change when the environment does not change, we model the resulting range biases as position dependent, meaning that each point in each state has fixed range biases. To be able to learn these biases, we simplify the model by partitioning the states into smaller sections, which we call *cells*, and define a fixed bias vector per cell c. Note that this should not be confused with the cells of a cellular system. The set of cells is defined as: $\mathcal{C} = \{(i,j) : i \in \{1, 2, ..., N\}, j \in \{1, 2, ..., M_i\}\}$ where N is the number of states and M_i is the number of cells in state i, which depends on the size of the area corresponding to a particular state. Fig. 2 shows an example of the cell structure for a state, defined by the squares. The cell size is another system design parameter depending on the environment as well as on the accuracy requirement. The smaller the cell size, the higher the accuracy.

At each time step t, the actual position is assigned to a cell ($\mathbf{x}_t \in (i,j)$) with a fixed (but random) bias vector ($\mathbf{b}(\mathbf{x}_t) = [b_{2,1}(\mathbf{x}_t), b_{3,1}(\mathbf{x}_t), ..., b_{L,1}(\mathbf{x}_t)]$) associated with it. The resulting range difference measurement is then modeled as

$$\hat{d}_{l,1}(t) = d_{l,1}(t) + b_{l,1}(\mathbf{x}_t) + n_{l,1}(t). \tag{4}$$

The bias elements in each cell can be drawn from any distribution, e.g., from a uniform distribution such that $b_{l,1} \sim \mathcal{U}(0, b_{\max})$ where $b_{\max}$ is the parameter defining the maximum bias value (in meters) or from an exponential distribution such that $b_{l,1} \sim \exp(\frac{1}{b_{exp}})$ where b_{exp} is the parameter defining the expected value (in meters). These two distributions will be used in the simulation section for exemplary purposes, but they can be chosen arbitrarily. The TDoA estimates are used, together with the known positions of the anchors, as inputs to the least squares algorithm described in [11], and the resulting position estimate $\mathbf{o}_t$ is the observed input in the HMM.

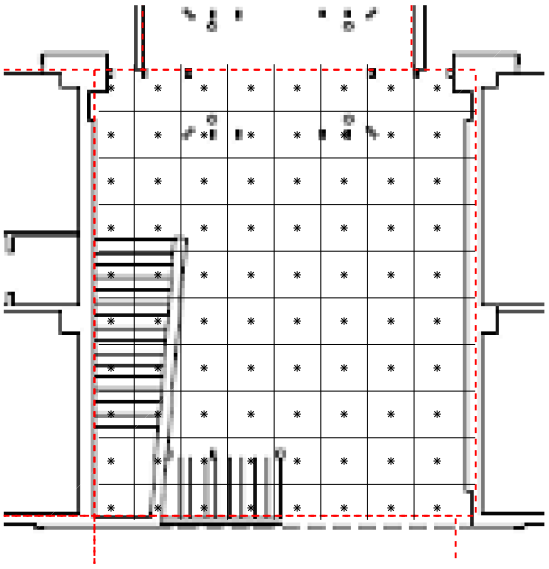

Fig. 2. Example of the cell points marked by ($*$) and the cells defined by the lines for state 2 in Fig. 1

D. Calculating the Conditional Output Probability

Since the observations are continuous, a continuous HMM is used. The probability of observing $\mathbf{o}$ while being in state i is

$$p_i(\mathbf{o}) = \int_{R_i} p(\mathbf{o}|\mathbf{x}) p(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{5}$$

where R_i is the region corresponding to state i and $p(\mathbf{x})$ is the probability of being at position $\mathbf{x}$. To be able to learn the model parameters, the integral in (5) is approximated by summing over M_i cell points in each room, yielding

$$p_i(\mathbf{o}) \approx \sum_{m=1}^{M_i} p(\mathbf{x}_{i,m}) \cdot p(\mathbf{o}|\mathbf{x}_{i,m}). \tag{6}$$

Fig. 2 shows an example of the cell points as well as the cells for state 2 of Fig. 1. While the cells (squares) are used to define the areas with fixed biases, the cell points are used to describe the probabilities of the observations for each state. Since the position estimation error is unknown, we approximate the error distribution of the position estimates using a Gaussian mixture. For a uniformly distributed position within a room and for the error models given in Section II-C, the approximated conditional output probability is:

$$p_i(\mathbf{o}) \approx \sum_{m=1}^{M_i} \frac{1}{M_i} \cdot \frac{1}{2\pi \cdot \sqrt{|\mathbf{\Sigma}_{i,m}|}} \cdot e^{-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu}_{i,m})^T \mathbf{\Sigma}_{i,m}^{-1} (\mathbf{o}-\boldsymbol{\mu}_{i,m})} \tag{7}$$

where $\boldsymbol{\mu}_{i,m}$ and $\mathbf{\Sigma}_{i,m}$ are the mean vector and the covariance matrix of mixture component m of state i, respectively. For

unbiased position estimates, the mean vectors are the cell point coordinates. For biased estimates, the mean vectors would be shifted depending on the fixed biases of the cells.

III. Implemented Algorithms

In this section, three implemented algorithms are described: (i) the Baum-Welch algorithm which is used to learn the HMM model parameters, (ii) the forward algorithm which uses HMM model parameters to estimate state probabilities and (iii) the particle filter with odometry information which was implemented as a benchmark to the presented method.

A. The Baum-Welch Algorithm

The description of the Baum-Welch (BW) algorithm as well as the algorithm steps follow the tutorial by Rabiner in [12]. The BW algorithm is an iterative algorithm which searches for the maximum likelihood solution parameters of a hidden Markov model given an observed sequence $\mathbf{o} = [\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T]$. The model parameters that can be estimated using BW are $\boldsymbol{\theta} = (\mathbf{A}, \mathbf{p}(\mathbf{o}), \boldsymbol{\pi})$ with $\mathbf{p}(\mathbf{o}) = [p_1(\mathbf{o}), p_2(\mathbf{o}), ..., p_N(\mathbf{o})]^T$ and $\boldsymbol{\pi} = [\pi_1, \pi_2, ..., \pi_N]^T$. The algorithm requires as input: (i) initial values for the model parameters $\boldsymbol{\theta}_0$ chosen randomly or using some information (e.g., the floor plan of the building), (ii) the defined states and (iii) an observed sequence $\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T$ of T position estimates. The steps of the algorithm are as follows:

1. Compute State Probabilities Using Current Model Parameters $\boldsymbol{\theta}$: To calculate the state probabilities, we first need to calculate the forward and backward probabilities of the sequence $\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T$. The forward probability $\alpha_i(t)$ is the probability of being in state i at time t using the partial observation sequence $\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_t$. It can be computed inductively using the currently estimated model parameters $\boldsymbol{\theta}$ as follows:

- Initialization: $\alpha_i(1) = \pi_i \cdot p_i(\mathbf{o}_1)$.
- Induction : $\alpha_i(t+1) = \left[\sum_{j=1}^{N} \alpha_j(t) a_{ji}\right] p_i(\mathbf{o}_{t+1})$.

The backward probability $\beta_i(t)$ is the probability of being in state i at time t given the partial observed sequence $\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, ..., \mathbf{o}_T$ and the currently estimated model parameters $\boldsymbol{\theta}$. It can be solved inductively as follows:

- Initialization: $\beta_i(T) = 1$.
- Induction: $\beta_i(t) = \sum_{j=1}^{N} a_{ij} p_j(\mathbf{o}_{t+1}) \beta_j(t+1)$.

Using the forward and backward probabilities, the state probabilities are calculated as

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}. \tag{8}$$

2. Re-estimate Model Parameters to $\hat{\boldsymbol{\theta}}$: The model parameters can now be re-estimated to $\hat{\boldsymbol{\theta}} = (\hat{\mathbf{A}}, \hat{\mathbf{p}}(\mathbf{o}), \hat{\boldsymbol{\pi}})$ by

- Calculating the probability of being in state i at time t and state j at time $t+1$ as follows:

$$\xi_{ij}(t) = \frac{\alpha_i(t) a_{ij} p_j(\mathbf{o}_{t+1}) \beta_j(t+1)}{\sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_k(t) a_{kl} p_l(\mathbf{o}_{t+1}) \beta_l(t+1)} \tag{9}$$

- Updating the transition probability by

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \tag{10}$$

- Updating the mean vectors and covariance matrices of the Gaussian mixtures by

$$\hat{\boldsymbol{\mu}}_{i,m} = \frac{\sum_{t=1}^{T} \gamma_{i,m}(t) \cdot \mathbf{o}_t}{\sum_{t=1}^{T} \gamma_{i,m}(t)} \tag{11}$$

and

$$\hat{\boldsymbol{\Sigma}}_{i,m} = \frac{\sum_{t=1}^{T} \gamma_{i,m}(t) \cdot (\mathbf{o}_t - \boldsymbol{\mu}_{i,m})(\mathbf{o}_t - \boldsymbol{\mu}_{i,m})^T}{\sum_{t=1}^{T} \gamma_{i,m}(t)} \tag{12}$$

where

$$\gamma_{i,m}(t) = \gamma_i(t) \cdot \left[\frac{p(\mathbf{o}_t|\mathbf{x}_{i,m})}{\sum_{l=1}^{M_i} p(\mathbf{o}_t|\mathbf{x}_{i,l})}\right]. \tag{13}$$

3. Repeat Steps 1 and 2 until Convergence: The state probabilities can be re-calculated using updated model parameters and the model parameters can be re-estimated using the new probabilities. These steps are repeated until the maximization of the likelihood $p(\mathbf{o}|\theta) = \sum_{i=1}^{N} \alpha_i(T)$ has reached a desired convergence (i.e., when the improvement of the likelihood is negligible from one step to the next).

B. The Forward Algorithm

The forward algorithm (FW) is the actual *state sequence* estimation algorithm. Unlike the BW algorithm, which aims at estimating the *HMM model parameters*, the forward algorithm estimates the state sequence using given HMM parameters (known or learned through the BW algorithm) and an observation sequence as discussed in III-A.1. The estimated state at time t is the state with the highest calculated forward probability.

C. The Particle Filter with Odometry Information

As a benchmark to our proposed method and to compare our model with the state-of-the-art, we employ a particle filter (PF) which is provided with the same range measurements as the aforementioned algorithms in addition to (noisy) odometry information. The estimated odometry information at time t is modeled as

$$[\hat{\Delta}x, \hat{\Delta}y]_t^T = \mathbf{x}_t - \mathbf{x}_{t-1} + \mathbf{w}_t, \quad \mathbf{w} \sim \mathcal{N}\left(0, \frac{\sigma_o^2}{2}\mathbf{I}\right) \tag{14}$$

The odometry information is modeled as the straight-line distance between the true positions at time t and $t-1$ plus a zero-mean estimation error $\mathbf{w}$. Using the odometry information, the known odometry error model, the TDoA estimates, as well as the TDoA error model, the particle filter was implemented following [13].

IV. Results

For the simulations, we used a simplified floor plan of part of Durham Hall at Virginia Tech shown in Fig. 1. We defined 27 states on that plan, marked by the dashed red lines, and performed the simulation by generating a Markov chain from these states using a transition matrix generated similar to (1). For the TDoA position estimation, four anchors were placed on four corners around the building. The cell size for generating the fixed biases as well as for calculating the output probability, was set to $1m \times 1m$.

For each generated position, the observed position estimate, which is the input for the different algorithms, was generated following these steps:

1) Calculating the true TDoAs using the four anchors.
2) Adding the bias vector of the cell of the true position.
3) Adding white Gaussian noise to the biased TDoAs.
4) Estimating the position from the erroneous TDoAs using the algorithm by Chan & Ho [11].

After generating the observations from the true states, we ran different algorithms to show their performances: (i) the forward algorithm (FW) given the true model parameters (i.e., true transition matrix, true mean vectors and covariance matrices of the observations), (ii) the forward algorithm with estimated model parameters using the BW algorithm, and (iii) the particle filter using noisy odometry information (PF). To initialize the BW algorithm, we used the floor plan of the building to set the zero-transition probabilities in the transition matrix. All non-zero transition probabilities were assumed equally likely. The mean vectors were the center points of the cells and the covariance matrix was

$$\mathbf{\Sigma} = \begin{bmatrix} 2500 & 0 \\ 0 & 2500 \end{bmatrix}, \tag{15}$$

assuming high positioning errors as initial values. The chosen performance criterion was the probability of finding the correct state since this is sufficient for most indoor positioning scenarios and is equal to the output of the forward algorithm. The parameters of the simulations were the range estimation error (σ_{n} from (2)), the particle filter odometry error (σ_{o} from (14)) and the bias parameters described in Section II-C ($b_{\max},\ b_{exp}$). We show the error behavior as the number of BW runs increases. Each BW run corresponds to running the BW algorithm up to convergence using one observation sequence.

A. Comparing the Forward Algorithm to the Particle Filter

Fig. 3 compares using the PF with odometry information to the FW with known transitions and known mean vectors and covariance matrices of the observations. This result of the FW algorithm can be seen as an upper bound to using learning algorithms for these parameters. The result shows the effect of increasing ranging error standard deviation on both algorithms for different odometry errors. As a comparison, the dashed curve shows the performance of the position estimate $\mathbf{o}_t$ that was given to both algorithms (i.e., the least squares estimate using TDoA measurements). Using the HMM, the probability of estimating the right state is greater than 80% even with a ranging error standard deviation of 35 m. The FW outperforms the particle filter even for an odometry error standard deviation

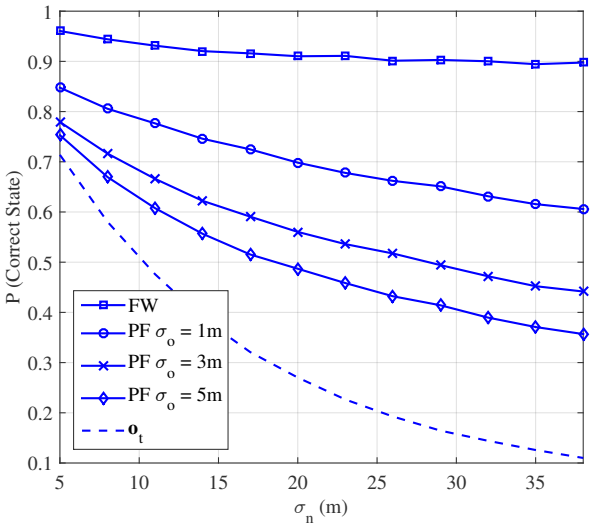

Fig. 3. Performance of the forward algorithm with known model parameters vs. the particle filter with known odometry error distribution over increasing σ_{n}.

of only one meter. It is important to note here that the results highly depend on the transition matrix. The further away we are from uniform probabilities (the more we move towards deterministic behavior), the better the results using the hidden Markov model and hence, the more robust the localization performance is against errors.

B. Learning the Biased Error Distribution

Fig. 4 shows the effect of learning the distributions of the biased observations, assuming known transition probabilities. The biases were generated randomly either using a uniform distribution or an exponential distribution. Each BW run was performed using a sequence of 200 estimated positions. The results show that the BW algorithm is able to accurately approximate the error distributions of the observed position estimates using the Gaussian mixtures presented in (7) after only one run. Additionally, it can be seen that, given the same mean ranging bias, exponentially distributed ranging biases lead to poorer performance than uniformly distributed biases, possibly due to the unbounded nature of the ranging errors with the exponential model.

C. Learning the Model Parameters with User Feedback

A limitation of the BW is that it searches for a *local* maximum of the likelihood function. Whenever the system has a complicated distribution, as in our model, the algorithm may converge to a local maximum, thereby limiting performance. This is the case when trying to estimate all model parameters in the described scenario, given random or uniform initial parameters. This problem can be mitigated using feedback. Feedback in this context means providing information about the true state, supplied by the user, who is aware of his/her position, at random times. More specifically, this feedback could be given by a cellular user simply stating whether or not he/she is in the estimated room. Using this feedback, the model is trained without having to run tests and perform measurement campaigns for each user. Fig. 5 shows the effect of

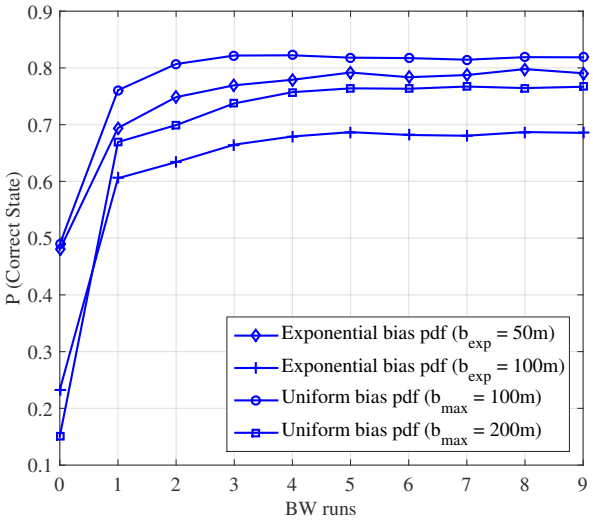

Fig. 4. Performance of the forward algorithm with a known transition matrix and an unknown biased observation error. The results are plotted over Baum-Welch runs that were performed every 200 Markov steps.

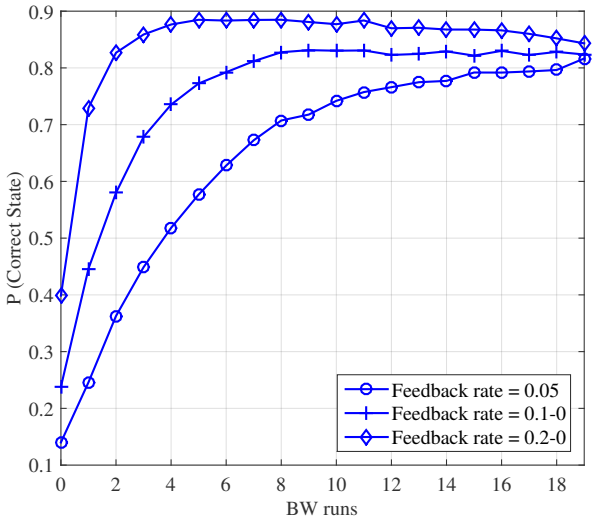

Fig. 5. Performance of the forward algorithm with unknown transition matrix and unknown biased observation error over Baum-Welch runs that were performed every 500 Markov steps with user feedback.

learning all model parameters with the help of user feedback. Compared to learning the observation error distribution with a known transition matrix, learning all model parameters at once requires longer observation sequences in order to obtain accurate results. Therefore, the BW algorithm was applied on observation sequences of length 500 rather than 200 in the last result. We use different feedback rates for the simulations. A rate of 0.05 means that the user chooses to provide his/her state in 25 out of 500 time steps. As a comparison, we also show two simulations with descending feedback starting at 0.1 and 0.2 and ending at 0. The results demonstrate that feedback is needed mainly at the beginning and that higher feedback rates at the beginning accelerate the learning curve. The decrease in accuracy of the two descending feedback curves is due to the fact that the information fed to the algorithm decreases with each run.

V. Conclusion

This paper shows how the habitual patterns of a person inside a building can be used for improving indoor localization, which is particularly helpful in applications such as indoor emergency response. By modeling these patterns using a hidden Markov model, not only can we use algorithms like the forward algorithm to obtain improved position estimates, but we can also learn the transition probabilities of the user as well as the parameters of the positioning error distribution by applying the Baum-Welch algorithm. If the algorithm is applied regularly, we can even adapt to changing environments by re-estimating the model parameters. For future work, additional parameters can be considered in the motion model (e.g., time). Additionally, an experimental test can be conducted to verify the presented results.

References

[1] R. Zekavat and R. M. Buehrer, *Handbook of Position Location: Theory, Practice, and Advances*. Wiley, 2012.

[2] Federal Communications Commission, "Wireless E911 Location Accuracy Requirements: Third Further Notice of Proposed Rulemaking," *PS Docket No. 07-114*, February 21, 2014.

[3] CTIA - The Wireless Association, "Reply Comments of CTIA in the matter of Wireless E911 Location Accuracy Requirements." PS Docket No. 07-114, July 14 2014.

[4] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, pp. 1067–1080, Nov 2007.

[5] C. Morelli, M. Nicoli, V. Rampa, and U. Spagnolini, "Hidden Markov Models for Radio Localization in Mixed LOS/NLOS Conditions," *IEEE Transactions on Signal Processing*, vol. 55, pp. 1525–1542, April 2007.

[6] A. Haeberlen, A. Rudys, E. Flannery, D. S. Wallach, A. M. Ladd, and L. E. Kavraki, "Practical robust localization over large-scale 802.11 wireless networks," in *in Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, pp. 70–84, 2004.

[7] G. Cielniak, M. Bennewitz, and W. Burgard, "Robust localization of persons based on learned motion patterns."

[8] M. Ocaa, L. M. Bergasa, M. n. Sotelo, R. Flores, D. F. Llorca, and D. Schleicher, "Automatic training method applied to a wifi+ultrasound pomdp navigation system.," *Robotica*, vol. 27, no. 7, pp. 1049–1061, 2009.

[9] S. Marano, W. Gifford, H. Wymeersch, and M. Win, "NLOS identification and mitigation for localization based on UWB experimental data," *IEEE Journal on Selected Areas in Communications*, vol. 28, pp. 1026–1035, September 2010.

[10] A. Conti, M. Guerra, D. Dardari, N. Decarli, and M. Win, "Network Experimentation for Cooperative Localization," *IEEE Journal on Selected Areas in Communications*, vol. 30, pp. 467–475, February 2012.

[11] Y. Chan and K. Ho, "A simple and efficient estimator for hyperbolic location," *IEEE Transactions on Signal Processing*, vol. 42, pp. 1905–1915, Aug 1994.

[12] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, Feb 1989.

[13] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 425–437, Feb 2002.