2012 International Conference on Indoor Positioning and Indoor Navigation, 13-15th November 2012

Hidden Markov Model-based 3D Path-matching using Raytracing-generated Wi-Fi Models

Nicolai Viol, Jó Ágila Bitsch Link, Hanno Wirtz, Dirk Rothe, and Klaus Wehrle
Communication and Distributed Systems
RWTH Aachen University
Aachen, Germany
Email: {viol, bitsch, wirtz, rothe, wehrle}@comsys.rwth-aachen.de

***Abstract*—We propose an efficient approach to probabilistic 3D indoor path-matching and localization based on Wi-Fi-signal measurements using Hidden Markov Model-based (HMM) algorithms. Given a 3D model of the building, we derive high-resolution emission probabilities and transition probabilities from raytracing-generated Wi-Fi signal propagations. Therefore we use both the generated signal-strength values and the geometric information of the 3D model. Based on the emission and transition probabilities and a sequence of Wi-Fi signal measurements provided by the client, the HMM-based algorithm computes the most probable path through the building.**

We qualitatively compare our HMM-based algorithms with state-of-the-art Particle Filter-based (PF) algorithms for both offline and online localization accuracy using both simulated and real-world Wi-Fi measurements. In simulation, HMM outperforms PF by up to 40% regarding path accuracy and also performs significantly better in finding the current position. Using real-world measurements, HMM computes the user's path 34% more accurately compared to PF while both perform equally well in online localization. The evaluation further shows that a low-effort 3D model of a building using only five different materials for walls and doors is sufficient to achieve an average localization error below 2m.

Our HMM-based approach provides accurate Wi-Fi localization in real-time using commodity hardware and performs especially well in path-matching. This makes it also a viable tool for applications like geofencing or for offline analysis to identify error-prone locations and measurements.

I. Introduction

Indoor localization based on Wi-Fi has been a fertile research topic in the last decade. Compatible devices are cheap and the infrastructure it needs is almost ubiquitous, in many public and private environments. Still, the actual deployment of Wi-Fi-based indoor localization is limited. The main cause for this lies in the calibration and maintenance effort required to keep the system accurate. Furthermore, the accuracy for specific applications such as geofencing and other location-based services might not be enough.

In this work, we address these points. To reduce the initial calibration effort, we make use of already available 3D models of a building. Especially in the context of trade fairs and office buildings, there are detailed floor and exhibition plans, or office plans available. In case those are not available, we can make use of legally mandated evacuation plans, from which we can generate a 3D model of a floor. Using this approach, we can significantly reduce the overhead for manual calibration that other Wi-Fi fingerprinting based schemes require.

978-1-4673-1954-6/12/$31.00 © 2012 IEEE

Once we have this high-resolution signal data available from the ray tracer, we evaluate the performance of different 3D localization mechanisms. Aside from the more common Least Mean Square Error (LSME)-based methods, we look at Particle Filters (PF) and particularly at Hidden Markov Models (HMM). We propose an optimized HMM approach applying a Viterbi decoder and state pruning to achieve real-time results in the highest resolution with commodity hardware (Intel core i7 CPU). We show that while we achieve performance comparable to the related work during online localization, we can improve the performance for offline applications like tracking and geofencing by up to 40%.

The remainder of the paper is structured as follows: After an analysis of the related approaches, both in terms of propagation modeling and in Wi-Fi Localization, particularly fingerprinting and HMM-based approaches, we discuss the framework of our system. We focus on the raytracer and its calibration, as well as the localization component. We follow with a detailed algorithmic description of our HMM-based and our Particle Filter-based approaches. In our evaluation we support our performance claims.

Contributions

The main contributions of this work are:

- **HMM-based offline and online localization:** We propose efficient offline and online 3D localization using an HMM-based path-matching algorithm using only commodity and inexpensive Wi-Fi hardware, namely a mobile device and access points.
- **Low-effort raytracing:** We use a low-effort and easy to adapt 3D radio propagation model based on raytracing. The derived high resolution signal volumes and 3D model map to finely distinguishable paths.
- **Good performance of HMMs in a variety of cases:** We show HMM outperforms PF and LMSE especially in the offline variant, on both synthetic and real-world data. The real-world results are comparable to other high-effort approaches.

II. Related Work

We divide our related work into two main categories: Propagation Modeling, which can be further subdivided into empirical, semi-empirical and physical models, while Wi-Fi Localization may be categorized into fingerprinting, model-base, and sequential approaches. We will see that, through the novel combination of several of these approaches, we can significantly reduce the effort required for Wi-Fi-based indoor localization while preserving good accuracy.

A. Propagation Modeling

Various signal propagation modeling approaches have been published in the last decades. While their initial purpose was the optimization of wireless communication coverage and performance, they receive new interest in their ability to improve wireless localization systems, either through predicting a localization accuracy at different locations, or by serving as a basis of a localization technique itself.

We divide radio propagation models into empirical, semi-empirical (or analytical), and physical models. We can derive a basic empirical model purely from manual measurements. The model captures the signal propagation very exactly. However, this accuracy comes at the price of a very high manual effort with respect to resolution and consistency.

Semi-empirical models make use of path-loss functions to compute the signal propagation. These functions are parametrized by empirical measurements. While basic semi-empirical models only consider the distance to a source, more advanced systems take different materials and environments into account. A prominent example is the Wall Attenuation Factor (WAF) model used within the RADAR localization system [1]. Parameters are very environment-specific and the model needs to be re-calibrated in each new scenario.

In contrast, physical propagation models simulate the physical nature of radio signals by accurately modeling different physical effects like attenuation, reflection, diffraction, or refraction. This requires an accurate model of the environment including detailed material parameters. Depending on the scenario, the additional overhead associated with the creation of such a high-accuracy model does not pay off and makes it necessary to find the best tradeoff between effort and accuracy. These models are prominently computed using a raytracer [2]–[5].

El-Kafrawy et al. [6] performed an extensive analysis on the applicability of different types of signal propagation models to Wi-Fi localization. They show that raytracing models in general and 3D-raytracing in particular are able to provide more accurate signal propagation models compared to simpler models and achieve significantly better localization results. This makes raytracing models a viable basis for Wi-Fi localization, even though they need longer computation times and more detailed models compared to other propagation prediction schemes.

B. Wi-Fi Localization

Fingerprinting-based Wi-Fi localization, also referred to as range-free localization, typically uses a previously generated fingerprinting database containing the AP signal levels at various locations of the scene to determine the location of a user. In contrast to range-based approaches, fingerprinting does not use explicit distance estimation to a set of anchor-points and is therefore more accurate in indoor scenarios where signals are massively affected by many objects. Thus, the required infrastructure and client hardware for fingerprinting can be simpler and less expensive.

One of the first papers published presenting a fingerprinting approach is [1]. By manually measuring the signal values of the available APs at various reference points, Bahl et al. achieve a localization accuracy in the range of 2 m to 3 m.

More advanced approaches in matching client measurements to the stored fingerprints are able to further reduce the error below 1 m [7]. However, the performance of Wi-Fi fingerprinting is affected by dynamic objects and changes in the environment. Hence, the main problem of fingerprinting is the huge effort associated with the creation and maintenance of an up-to-date fingerprinting database. As a consequence, model-based approaches are proposed that uses different propagation model techniques to reduce the initial effort, whereby raytracing models have shown to achieve the best results [6].

Further improvement in terms of localization accuracy and robustness is achieved by sequential localization approaches. They exploit the sequential characteristic of Wi-Fi measurements to match the individual measurements to a sequence of locations using mobility models and thereby ensuring more natural movements. Evaluating a sequence of measurements comes at the cost of higher computational complexity and thus requires efficient algorithms like Particle Filters (PF) [8], [9] or Hidden Markov Models (HMM) [10]–[12]. These approaches are prominently used in robotics and are typically combined with additional sensors, e.g., compass, step-detection, or accelerometer which refine the mobility model through activity and mobility detection. Although HMM-approaches achieve best results they demand higher computation times. Therefore, they are typically used only with low-resolution states [10]. In contrast, PF approaches work very efficiently due to smart random sampling but the parameters need to be calibrated carefully with respect to the expected error of the measurements for each scenario. Otherwise, PF-based approaches are prone to fail.

This observation motivated us to investigate an efficient HMM-based localization approach based on high-resolution raytracing models.

III. Framework

The localization framework consists of two main components, the Radio Propagation Component (RPC), and the Localization Component (LC).

The RPC is responsible for modeling radio propagation from different access points. Using a raytracer allows us to produce a fingerprinting database (i.e., 3D signal volumes) with a much higher spatial resolution compared to pure empirical methods while being able to adapt to environmental changes. However, we still use raw measurements derived from a client device to calibrate our model parameters.

The LC uses the propagation model and the 3D model provided by the RPC as the main source of information to perform localization. A mobile client is now able to send a sequence of Wi-Fi measurements from the scene to the LC. This sequence is then processed by the LC and the client gets a sequence of locations in return. We propose a Hidden Markov Model-based path-matching algorithm in an offline and an online variant. We compare the results with corresponding offline and online Particle Filter path-matching variants, as well as with a naive Least Mean Square Error-based fingerprint matching scheme.

A. Radio Propagation Component

To compute the radio propagations of the APs in the environment we use the Photon raytracer [13]. The generated radio propagation is a 3D voxel volume with a predefined resolution, also called signal volume. We use signal volumes with a resolution of $0.2\,\text{m} \times 0.2\,\text{m} \times 0.2\,\text{m}$. The Photon raytracer takes the following data as input:

1) A polygon mesh of the scene geometry: We created the mesh model manually in Blender using floor plans and few manual measurements to gather the heights. While this is not the focus of this paper, other tools exist that automatically derive these models from floor plans. Also, trade fair planning tools directly provide us with the necessary information.

2) Material parameters for surfaces: Using an optimizer, we found a set of optimal material parameters to use in our scenario. This process is time-consuming, but can be completely automated, using the reference values described in Section V-B. Using optimal parameters allows us to investigate the influence of model granularity on the localization accuracy without artificially reducing our accuracy beforehand.

3) AP position, power value and channel: We create a different configuration for each signal source.

4) Voxel resolution: The voxel resolution may have an arbitrary size. However, higher resolutions come at the cost of higher processing times. Voxel sizes below the radio wavelength do not make sense, as effects like interference are not considered in the Photon raytracer. Therefore, a higher resolution does not provide us with more information. A voxel size of about $0.2\,\text{m}$ provides us with a good trade-off between raytracing time and propagation accuracy. We can also reduce the resolution later using voxel aggregation in a postprocessing step, if needed.

B. Localization Component

A mobile client localizes itself by periodically scanning for available APs. The scan result includes the BSSID and the MAC address of each discovered AP, as well as the received signal strengths. The localization component (LC) preprocesses these samples and forwards them to a localization engine. For our HMM-based approach, this preprocessing provides sample interpolation to avoid missing samples. More details about this process are presented in Section IV-A3.

We implemented three main localization engines, an HMM-based, a PF-based, and an LMSE-based engine. The HMM- as well as the PF-based engine provide, an offline and an online variant, depending on the exact use case. A localization engine takes the preprocessed signal sequence and combines it with the radio propagation volumes and the 3D mesh from the RPC.

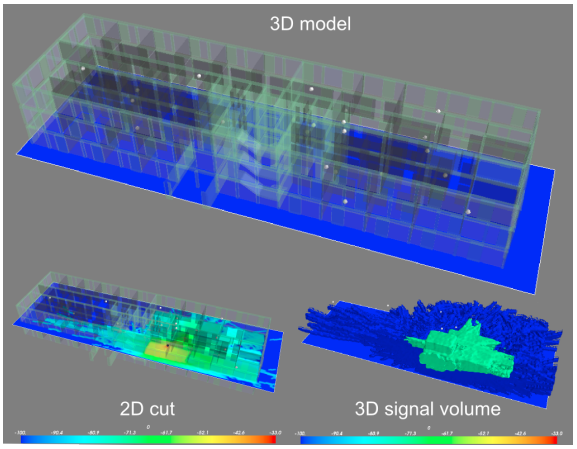

Fig. 1. Visualization of the 3D model and the generated signal volume in both a 3D view and a 2D cut.

In this context, we weigh a candidate location by the difference between the signal sample and the radio propagation volume. The 3D mesh refines the location estimation, e.g., by identifying blocked and impossible locations. In addition, a mobility model further restricts possible locations. We present the details of this mobility model in Section IV-A2.

As a result the LC returns a sequence of locations representing the most probable path of the mobile client with respect to the sample sequence and the applied localization engine.

IV. Localization Algorithms and Design

We define the localization problem as follows: For a given sequence of Wi-Fi measurements, i.e., a sample sequence, a sequence of locations is determined that (1) matches best the signal values derived from the pre-computed propagation volumes and that (2) follows the mobility model of a mobile user in a building. Hence, the localization problem is analog to a tracking or path-matching problem.

Additionally, we require the sample sequence to be geographically correlated, that is, consecutive samples are taken fast enough, so that the possible movement of a user is restricted to short distances with respect to the scene size. More formally, for each measurement x_i performed at time t_i a predecessor x_{i-1} performed at time t_{i-1} exists with $dt = t_i - t_{i-1} < \epsilon$ and $\epsilon \times \text{maxMovingSpeed} = \text{moveDistance} << \text{SceneSize}$, where the measurement x_i is a D-dimensional vector holding the signal strength readings of D APs measured at time i.

In practice, this means if we restrict the user speed to $3\,\text{m/s}$, we should collect a sample at least every $1\,\text{s}$, at a movement state resolution of $0.6\,\text{m}$. Using this sequential characteristic and the knowledge about previous locations, we can reduce errors as they are often caused by temporary disturbances, such as signal shadowing by people or other dynamic effects. We can also use this information to improve localization in regions where due to limited AP coverage only inexact or limited data is available.

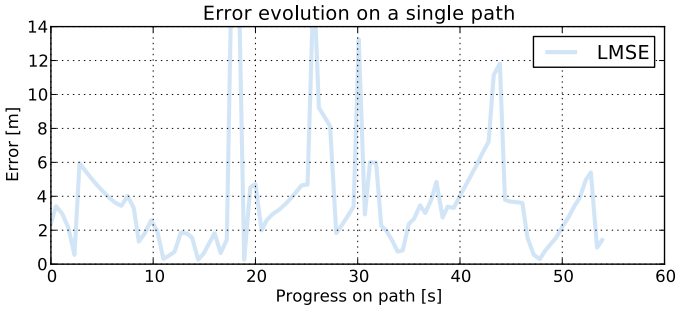

Fig. 2. Evolution of the LMSE error for a selected path. False measurements in error-prone regions lead to extreme localization outliers.

To quantify the effects of applying mobility models we compare our approach to a classic naive LMSE-based approach, such as RADAR [1], which does not consider any mobility constraints and relies only on the radio propagation model.

a) Least Mean Square Error (LMSE): LMSE compares the RSSI reading of the D-dimensional signal sample x with all location annotated measurements y_s of a database using the Euclidean distance and returns the location s, which fits best. The LMSE decision rule $r_{\text{LMSE}} : x \to s$ is defined as:

$$r_{\text{LMSE}}(x) = \underset{s}{\operatorname{argmin}} \left[\frac{1}{D} \sum_{d=1}^{D} (x_d - y_{sd})^2 \right]$$
$$= \underset{s}{\operatorname{argmin}} \sum_{d=1}^{D} (x_d - y_{sd})^2$$

An example of a location sequence derived with LMSE is shown in Figure 2. Localization outliers larger than 12 m, e.g., at time 18 s, result in large jumps through the scene. Path-matching algorithms eliminate these jumps by following the natural movement of a user.

Both the HMM-based and the PF-based localization algorithms make use of the mobility model. We distinguish between online and offline localization variants depending on whether we derive the location as we are measuring the signal samples or want to derive the most probable sequence of locations ex post facto.

b) Online vs. offline localization: The online variant returns the currently most probable location. It reduces the error by eliminating impossible paths and avoiding jumps caused by faulty measurements or limited signal data availability. However, in regions with limited signal data, it might still lead to inaccuracies. Online localization is typically used for real-time applications, e.g., in navigation systems, where a user needs to know the current position just in time.

In contrast to this, the offline variant returns the complete path of a user ex post facto. As we have perfect future knowledge for a given sample, we can take this into account when calculating the path a user took through the environment. Hence, we first calculate the best final location and then backtrack through the most probable previous locations. This allows us to significantly reduce errors at intermediate locations. The offline approach can be used to determine an accurate path of a mobile agent, for instance in geofencing applications. Also, the results from offline tracking can help identify error-prone regions, e.g., by comparing offline with online results.

A. HMM

Our HMM-based localization approach is based on a first-order Hidden Markov Model. Each location is modeled as a state $s \in S$. We denote the transition probability from state s to state s' by $p(s|s')$. The current state is not directly observable. However, each state has a given emission probability $p(x|s)$ for a set of observable signal measurements x from an access point. We model these measurements as a multivariate Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with independent components. Using the Viterbi decoder, we can efficiently extract the sequence of locations, i.e., the hidden states, from the sequence of signal measurements.

To find the best sequence of states s^T for a given sample sequence x^T measured in the time interval T, the Viterbi algorithm is based on the Bayes decision rule, i.e., the problem of finding the maximized joined probability:

$$s^T = \underset{s^T}{\operatorname{argmax}}\, p(x^T, s^T)$$
$$= \underset{s^T}{\operatorname{argmax}} \prod_{t=1}^{T} p\left((x^t|s^t)|(x^{t-1}|s^{t-1})\right)$$

This can be further simplified. The HMM requires the emission probabilities $p(x|s)$ to be conditional independent. Without loss of generality, each signal sample is measured or modeled at any location individually and does not depend on previous or future samples, i.e., the samples are a multivariate Gaussian distribution. Thus, the requirement for the emission probability is fulfilled. Furthermore, in a first-order HMM each state in a sequence s_1^t only depends on the previous state. Applying these to the formula above results in:

$$s_1^T = \underset{s_1^T}{\operatorname{argmax}} \prod_{t=1}^{T} p(s^t|s^{t-1}) \cdot p(x^t|s^t)$$

To derive the most probable sequence of states s_1^T for a given observed sample sequence x_1^T, the Viterbi algorithm makes use of a dynamic programming approach. Q denotes the dynamic programming matrix with $|S|$ rows and $|T|$ columns. We initialize the calculation by giving each state the same probability:

$$\forall s \in S : Q(0, s) := 1/|S|$$

Then, the Viterbi algorithm is defined by the following recursive equations:

$$Q(t, s) := \max_{s'} \left[p(x_t, s|s') \cdot Q(t-1, s') \right.$$

As we are only interested in the sequence of states, and a recursion step is only dependent on the previous column in Q, we store the decision about which s' was chosen in a back-pointer array:

$$B(t, s) := s' = \underset{s'}{\operatorname{argmax}} \left[p(x_t, s|s') \cdot Q(t-1, s') \right]$$

Therefore, we can derive the most probable sequence of states by taking the most probable location from the last column of Q and then backtracking through B.

The computational complexity for evaluating Q at all time frames T is $O(T \cdot S \cdot N)$ where S is the number of states and N is the number of transitions at each state. The memory requirements for the Viterbi algorithm is proportional to the size of the back-pointer array and is $T \cdot S$.

As we are multiplying many small numbers, we can run into numerical problems when probabilities become very small. To address this problem, we perform the probabilities in negative logspace. This allows us to transform the search for a maximum probability into a very efficient search for minimum cost that only requires additions. Applying the transformation to negative logspace to the Viterbi recursion $Q(t, s)$ leads to:

$$\begin{aligned} Q(t,s) &:= \min_{s'} \left[-log(p(x_t, s|s')) + Q(t-1, s')\right] \\ &:= \min_{s'} \left[Q(t-1, s') - log(p(x_t|s)) - log(p(s|s'))\right] \end{aligned} \tag{1}$$

Commonly, the emission probabilities ($p(x_t|s)$) and transition probabilities ($p(s|s')$) evaluated by the Viterbi decoder are estimated during a training phase using the Baum-Welch algorithm [14]. For such a training, we need a set of location annotated signal measurements that covers all possible states, i.e., an empirical propagation model. In our approach, we simulate such measurements using the radio propagation model of the raytracer.

1) Emission Probabilities: In our scenario, the location annotated signal measurements are the signal volumes generated by the radio propagation component. As we assume the signal strength to follow a multivariate Gaussian distribution where all components are independent, we obtain a vector of the mean signal strength from each of the D Access Points for each position s. Unfortunately, it is not possible to derive the variance of the signals at individual locations from the signal volumes easily, because it is influenced by multiple effects that are not yet taken into account by the radio propagation component, e.g., interference effects, or shadowing effects by moving people. Thus, we assume a constant variance.

Accordingly, an input feature vector x for a time frame t is D-dimensional, i.e., x represents a list of stochastically independent RSS readings from D Access Points. Using a Gauss distribution, we can then define $p(x|s)$ as:

$$\begin{aligned} p(x|s) &= p(x_1, ..., x_d, ..., x_D|s) = \prod_{d=1}^{D} p(x_d|s) \\ &= \frac{1}{\prod_{d=1}^{D} \sqrt{2\pi\sigma_{sd}^2}} \exp\left[-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_d - \mu_{sd}}{\sigma_{sd}}\right)^2\right] \end{aligned}$$

As we assume the variance σ_{sd} to be constant for all locations and access points, we can further simplify this definition, by replacing the variance dependent terms by constants:

$$p(x|s) = \frac{1}{C_1} \exp\left[-\frac{C_2}{2}\sum_{d=1}^{D}(x_d - \mu_{sd})^2\right]$$

As we are operation in negative logspace, we further simplify the term:

$$-log(p(x|s)) = \frac{C_2}{2}\sum_{d=1}^{D}(x_d - \mu_{sd})^2 + log(C_1)$$

Inserting this into Equation 1, we are only interested in the minimum. Therefore, we can drop the constants C_1 and C_2 which leads us to the following recursion assignment for the Viterbi algorithm:

$$Q(t,s) := \min_{s'} \left[Q(t-1, s') + \sum_{d=1}^{D}(x_{td} - \mu_{sd})^2 - log(p(s|s'))\right]$$

Thus, we can simplify the Gaussian modeled emission probability to a distance calculation between x_{td} and the generated μ_{sd} components. This is equivalent to the LMSE distance calculation.

2) Transition Probabilities: We use the transition probabilities $p(s|s')$ to define a mobility model. We only allow state transitions that follow the natural movement of a user. Our mobility model is a (5,5,3)-transition model. This means, that we allow transitions to locations up to a distance of 2 steps in the x and in the y direction and of 1 step in z direction. Including the standing still case, this includes 5 possibilities in x and y direction and 3 possibilities in z direction. Thereby, we restrict the user to move primarily in the horizontal plane. In total, we have 75 possible transition per state.

It is easy to define other mobility models, but this simple model matches our observation of natural movement in an office environment and presented a good compromise between freedom of user movement and computational effort. Assuming a state resolution of $0.2\,\text{m}$, this results in a maximum transition distance of $0.6\,\text{m}$ in 3D space or a maximum distance of $1.8\,\text{m}$ assuming a state resolution of $0.6\,\text{m}$.

Next, we define the discrete probability distribution for the 75 transitions for each state. The model ensures that the probability distribution satisfies the normalization constraints defined as $\sum_{s \in S} p(s) = 1$ and $\sum_{s' \in S_{(5,5,3)}} p(s|s') = p(s)$. We distribute the probability of $p(s)$ evenly over all three possible transition distances of 0, 1, and 2:

$$p(0-\text{transition}) = p(1-\text{transition}) = p(2-\text{transition}) = 1/3$$

Then, we define the individual transition probabilities for the 75 transitions by using the environmental information from the 3D model combined with assumptions about a pedestrian user. To correlate the states with the 3D model effectively, we rasterize it into 3D cubes of the same resolution as the state space.

Transitions across different materials are easy to observe. We define them using the following assumptions about the movement of a user:

1) A user cannot walk in or through walls or furnitures, i.e. blocking elements. For all states s and s' that intersect with a blocking material (e.g., a wall), we define $p(s|s') = 0$. Special care need to be taken for the 2-transitions, because they possibly allow to jump through blocked states. To avoid this impossible transition, we check if there exists an unblocked path with a maximum length of two between s and s'. If not, we set $p(s|s') = 0$.
2) A user cannot fly, i.e., we assume a maximum height of $2\,\text{m}$ above ground. Therefore, state transitions that

lead to states that are more than 2 m above a blocking element are assigned $p(s|s') = 0$.

3) A users movement is assumed to follow straight paths most of the time. Hence, a transition towards free space is more likely than, e.g., a transition in front of a wall. Transitions with more free space are emphasized while transitions with less free space are weakened.

Note that assumption 1 and 2 already led to reasonable good results. Assumption 3 added only minor improvement and can be considered a fine-tuning.

3) Sample Interpolation: The maximum speed this model allows depends on the frequency of the samples taken from the client, because we allow only a maximum transition across two locations and the location states have a fixed size. Hence, the maximum physical distance is $0.6m$, $1.2m$, or $1.8m$ per sample, respectively.

The sample frequency depends on the client ability of performing the Wi-Fi scans. Even if we scan at the highest possible frequency, our test hardware only allowed us to take between 1 to 3 samples per second. Thus, for example, a sample rate of 1 sample/s and a state resolution s_{res} of 0.2 m results in a maximum speed of only 0.6 m/s allowed by the model, which is too slow to align with the maximum speed of the user.

To enable the model to allow a maximum speed $v_{\max}$ of at least 3 m/s for all state resolutions, we perform a sample interpolation on the sample sequence in a preprocessing step. The sample interpolation is defined as follows: For each two samples x_i and x_{i+1} taken at timestamp t_i and t_{i+1}, we generate a new sample x_j if $\Delta t > t_{\min}$, with $t_{\min} := s_{\text{res}} \cdot 3/v_{\max} = s_{\text{res}}/1\,\text{m/s}$. We linearly interpolate the signal values of x_j by taking the mean between x_i and x_{i+1} for each AP and the time $t_j = t_i + \Delta t/2$. If Δt is still too big, we perform this operation again.

The linear interpolation is analog to a low-pass filter applied only to the interpolated samples, thus, it does not introduce outliers. However, it does not reduce the overall measuring error of the sample sequence, because the true measurements are not changed.

4) Pruning: To reduce the memory and the computational complexity of the Viterbi decoder, we prune the most unlikely states from the calculation and only keep the top-N states for further processing. This is motivated by the observation, that many of the intermediate hypotheses have a very low probability as they are far away from the actual user position. Thus, they are very unlikely to affect the observation of the real path.

To efficiently retain the list of the top-N hypothesis at each time frame we use the skiplist data structure [15]. A skiplist represents a list of permanently sorted items, i.e., the top hypothesis, with an insertion cost of $O(\log(n))$ during decoding.

However, if pruning is too aggressive, we might remove a hypothesis, i.e., a candidate location prematurely, even though it receives a boost in the next iteration. Hence, the size of N is crucial to the localization performance. We observed that 6% of the total number of states is a good choice. Thus, pruning 94% of the hypotheses, i.e., cutting computation time by a factor of 10, does not adversely affect the results. This allows us to decode a sequence of measurements in real-time an commodity hardware, even at the highest state resolution of 0.2 m.

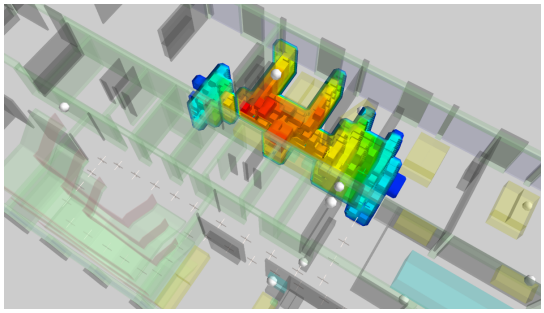

Fig. 3. A 3D plot of 300 unpruned states after the first sample of a sequence was evaluated by the HMM. The colors indicate the individual state probabilities where red indicates the largest and blue the lowest probability.

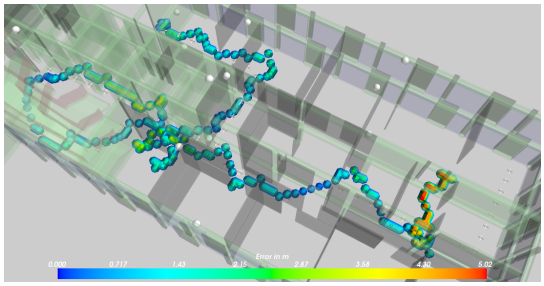

Fig. 4. The HMM localization result plotted in 3D. The path starts in the top left in the first floor and follows the stairs down to the floor beneath and than to the right side along the rooms. The colors indicate the localization error of each sample with red indicating larger errors at the end of the path.

Figure 3 shows a 3D visualization of the top-300 (unpruned) state after evaluating the first time-frame of a sequence. All states are close together concentrating around the location with the highest probabilities in the center of the top results. The result of the evaluation of a complete sequence is presented in Figure 4.

B. Particle Filter

Particle Filters (PF) are solving an analog problem to HMMs. For a given sequence of signal samples x^T, we need to find the most probable sequence of s^T of states with given emission and transition probabilities $p(x|s)$ and $p(s|s')$.

The main difference lies in the evaluation process of the sequence x^T. Instead of evaluating all possible path traversals, the PF approach optimizes this process efficiently by using weighted random choices, i.e., the particle sampling.

We define the sampling process recursively. Let $\{s_t^{(l)}\}$ be a set of samples with cardinality L for a time frame t and $\{s_0^{(l)}\}$ an initial set of samples uniformly distributed across the scene: $p(s_0)$. Then, the sampling weights $\{w_t^{(l)}\}$ are defined by:

$$w_t^{(l)} = \frac{p(x_t|s_t^{(l)})}{\sum_{m=1}^{L} p(x_t|t_t^{(m)})}$$

The weight for a sample is derived from the emission probabilities $p(x_t|s_t)$. The weights satisfy the normalization

constraint $\sum_l w_t^{(l)} = 1$ and are in the range $0 \leq w_t^{(l)} \leq 1$. The posterior probability $p(s_t|x_t)$ is represented by the combination of the samples and the corresponding weight for each sample. The next posterior $p(s_{t+1}|x_t)$ is defined by combining the weights and the transition probabilities:

$$p(s_{t+1}|x_t) = \sum_{l=1}^{L} w_t^{(l)} p(s_{t+1}|s_t)$$

In a last step, new samples $\{w_{t+1}^{(l)}\}$ are generated from this distribution. The weight defines the number of new samples that are generated. The transition probability sets the most probable direction and distance when generating new particles.

During the evaluation, intermediate states are stored in a backtracking table. To derive the offline result, i.e., the intermediate states that led to the most probable final state, we evaluate this backtracking table. The online result is defined as the most probable state at the time frame t.

1) Emission Probabilities: The definition of the emission probabilities $p(x|s)$ follows the HMM design. $p(x|s)$ is a multivariate Gaussian distribution with a mean vector extracted from the radio propagation models. We empirically determined that an assumed signal variance of 5 dBm leads to the best results.

2) Transition Probabilities: The transition probability $p(s|s')$ is modeled as a multivariate zero-mean Gaussian distribution with independent components. The variance of the horizontal components has been set to 5 m and the variance of the vertical axis to 2 m, analog to the distances allowed in the HMM-transitions. We empirically determined both values by testing the PF on location annotated measurements.

Like in the HMM approach, the information of blocked elements is derived from the 3D model. During the sampling phase of the Particle Filter, the candidate samples s, that are either in blocked zones or have blocked voxels on the straight path from the source state s' to s are rejected. To ensure that not too many samples are being rejected, we perform a resampling until a minimum number of total samples is reached. We also limit the resampling to ensure termination of this process.

1×10^5 samples are generated at each iteration of the process, i.e., at each time frame. Further increasing this number did not have any positive influence on the error rate of the PF approach. This results in ≈ 1 sample/m^3 in our evaluation scenario.

V. Evaluation

After introducing the evaluation scenario, the used hardware and the process of performing the Wi-Fi measurements, we present the 3D model we built to drive the raytracer and evaluate the raytracer performance with respect to different model configurations. As the main evaluation, we show a synthetic analysis based on simulated signal values and an extensive analysis on real-world measurements of the performance of the proposed localization approaches.

A. Scene and Model Description

We used the UMIC Research Centre at RWTH Aachen University, a university office building, as our evaluation scenario. The building has a rectangular footprint of about 900 m^2 and a height of about 13 m and consists mainly of uniform office rooms and larger seminar rooms. We had detailed 2D floorplans available, which provided us with basic information on different wall types, doors, windows, and stairways necessary to construct a detailed 3D model.

For the 3D model, we defined 11 different object classes which we identified as the most common objects in the building and suspected to have the most effect on signal propagation. These object classes are the *Concrete* and the *LightWalls* class, used for, e.g., different walls, stairs, and floors. Furthermore, we defined three classes for different types of doors, namely the *Doors* class, representing regular wooden doors, the *IronDoor* class, and the *GlassDoor* class. Additionally we defined the classes *OuterBuildingWall* and *GlassWindow* and the two classes *Cupboard* and *Table* to represent the main pieces of furnitures present. Finally, we defined a class *Hardware* to represent electronic devices of a given size, like server racks or large printers, and the class *Railing* to represent the railing at the stairways.

We used the open source 3D modeling framework Blender (http://www.blender.org) to create the 3D models for all objects described by the classes. A 3D visualization of the model is shown in Figure 1.

B. Wi-Fi measurements

As a client platform, we use an Android Tablet, namely the Acer Iconia tablet. To measure the signal strength values of the access points we used the default Android Wi-Fi API using default settings without any specific adaptations. We can initiate Wi-Fi scans easily and the scan result returned is a list of the discovered access points including the required information about the BSSID, and the received signal strength (RSS). We performed two different Wi-Fi scans for the following uses:

1) **Static measurements at 60 distinct and known locations over three floors:** At each location we took measurements for ≈ 30 s to ensure a minimum number of RSS readings per AP. We used these location annotated measurements to calibrate the material parameters of the raytracing model and to evaluate the model performance.
2) **Sequential measurements on 7 different paths:** We made these sequential measurement in forward and backward direction through the building to create a set of sample sequences for the real-world evaluation. Each sequence also includes reference positions on the path that were marked with a time-stamp. We set the reference positions at abort every 5 m and at every turn of the paths. Thus, we were able to calculate all intermediate true positions and compare the localization results to this ground truth. Typically, we received 1 to 3 scan results/s which has proven to be enough for our purpose.

In total, we located 22 APs distributed on three floors inside the building. 7 of these APs were specifically dedicated only

TABLE I
PROPAGATION ACCURACY FOR DIFFERENT MODEL COMPLEXITIES.

Mesh	Basic1	Full1	Basic2	Basic5	Full11
Materials	1	1	2	5	11
Δ $[dB]$	5.7	5.8	4.4	4.2	4.3

to our project, meaning that we could move them as we saw fit. The remaining APs belonged to other projects or are part of the RWTH Aachen University IT infrastructure. Depending on the model of the APs, we discovered minor differences in the transmission powers, so we configured the raytracing for each AP type accordingly.

C. Propagation Model

We computed the signal propagations for the 22 APs using the Photon raytracer. Photon takes the 3D model, the material parameter for the model, an AP configuration, and a voxel configuration as input. The material parameter are crucial to the performance of the raytracer. Thus, we decided compute a optimum set of material parameter instead of relying only on historical data only partially available from literature.

To derive an optimum set of material parameters for all objects we used the location annotated signal measurements from 60 distinct locations for an optimization process based on a genetic algorithm. Typically, 30×10^3 raytracer operations were necessary for the genetic algorithm to terminate and deliver a optimal set for the 3D model.

1) Model Complexity: Making use of the optimization process, we evaluated the influence of the model complexity, i.e., the number of different object classes and material parameter, on the raytracing performance. Table V-C1 summarizes the results of five different model configurations and the corresponding mean delta between the simulated signals and the signals measured at the 60 reference points. *Basic1* includes Concrete and LightWalls classes combined to one mesh and one set of material parameters. *Full1* includes all objects combined to one mesh and one set of material parameters. *Basic2* includes Concrete and LightWalls and two sets of material parameters. *Basic5* includes the five main objects of the scene, Concrete, LightWalls, and the three door classes with a own set of material parameters each. *Full11* is the full set of 11 objects classes with individual material parameters.

We see that the improvement of using more than 2 different object classes and materials is marginal and probably not worth the effort to model these details. The result for the full model is even slightly worse than the *Basic5* result. We also compared the influence of the propagation performance to the localization performance and observed the same results. Real-world localization performed best using the *Basic5* model. Thus, we decided to perform the remaining evaluations presented in this paper on the *Basic5* model.

2) Volume Resolution: The volume resolution and therefore the number of possible states is a crucial factor for the performance of the localization approach. This is especially true for the HMM-based algorithm. Even though the RPC generates signal volumes at the highest reasonable resolution,

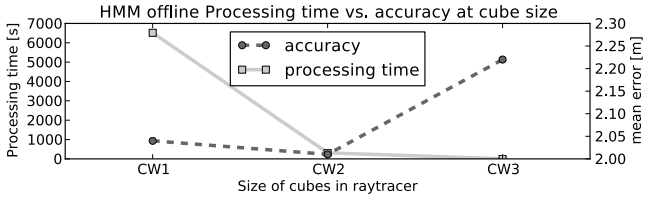

Fig. 5. The tradeoff between computation time and accuracy of HMM path-matching with respect the state resolution, i.e. the cube-width used.

with respect to computational performance we propose a simple solution to effectively reduce the resolution of the signal volumes and thus the number of states, without generating new signal volumes with the RPC. To reduce the signal volume resolution, we aggregate neighboring signal values into a mean signal. Thereby, The target resolution of the reduced volume is defined by the number of neighboring signal values in one dimension that are aggregated, also referred to as the cube-width c_{width}. For example, with $c_{\text{width}} = 3$ the three neighboring signal values in each dimension, i.e. $3^3 = 27$ signal values, are aggregated to one mean value. Thus, the resolution is effectively reduced by a factor of c_{width} and the number of states by a factor of c^3_{width}.

Figure 5 shows the joined graph of the localization error and the computation time versus the cube-width c_{width} of 1, 2, and 3. For a reduced state resolution of $0.4\,\text{m}$ ($c_{\text{width}} = 2$), we achieved the best trade-off between localization accuracy and computation time. Thus, the evaluation of the localization algorithms is performed on a state resolution of $0.4\,\text{m}$, even though the performance was sufficient to deliver real-time results also for the highest resolution of $0.2\,\text{m}$.

The fact that the localization error is about the same for $c_{\text{width}}1$ and $c_{\text{width}}2$ is probably due to the higher number of interpolations required for the lower resolution. For a state resolution $0.2\,\text{m}$ 70 % of the samples are interpolated, while for $0.4\,\text{m}$ only $40\,\%$ and for $0.6\,\text{m}$ about 20% of the samples are interpolation.

D. Path-matching and Localization

In this evaluation, we compare the performance of the offline and online variants of the HMM and the PF approach and the LMSE on (1) synthetic data simulating random error, and on (2) real-world sample sequences.

1) Synthetic Error: For the synthetic evaluation, we used the signal volumes from the raytracer and a Gaussian noise function to simulate noisy signal measurements for noise $\sigma \in \{0.0, ..., 15.5\}$dB in $0.5\,\text{dB}$ steps. We defined 16 distinct paths of different length and complexity trough the scene and generated 20 sample sequences x^T_{syn} per path and noise value σ, leading to 320 paths of a total length of about $8\,\text{km}$ per σ value. The synthetic localization errors for all algorithms are presented in Figure 6.

For all values of σ, the HMM offline algorithm achieves the best accuracy and is at least $20\,\%$ more accurate than the competing PF offline approach, in particular for larger values of σ. As expected, HMM and PF outperform the non-sequential LMSE algorithm. Interestingly, the PF offline

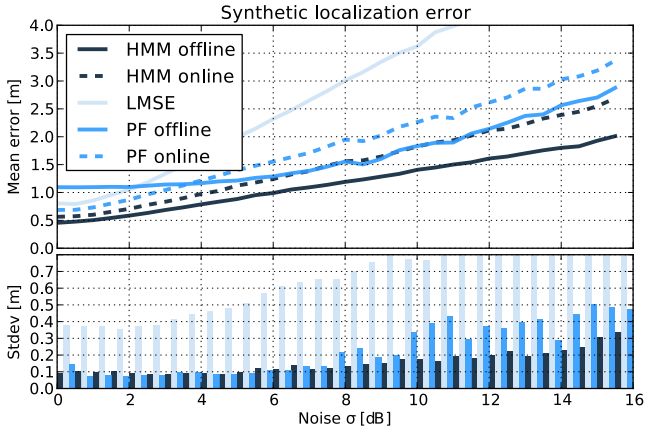

Fig. 6. Synthetic analysis of the HMM (offline and online) performance compared to PF (offline and online) and LMSE algorithms. The graphs show the corresponding mean error and standard deviation for different noise levels.

TABLE II
PHYSICAL PROPERTIES OF THE PATHS AND THE MEASUREMENTS.

Path	Length	Turns	avg. x	avg. APs	# x^T
EG-rooms-1	$40.0m$	12	30.5	8.9	20
OG1-floor-1	$41.9m$	7	40.6	8.4	22
OG1-rooms-3	$14.3m$	9	43.2	9.9	24
OG1-rooms-2	$81.2m$	16	93.3	7.3	21
OG1-rooms-1	$36.3m$	8	40.6	12.1	23
OG1-EG-1	$69.2m$	13	62.3	10.4	26
OG1-EG-2	$59.8m$	11	65.4	6.6	22

variant performs unexpectedly bad for $\sigma < 5$. The reason for this lies in the fixed variance values for the emission and transition probabilities for all values of σ. In turn, this results in a higher error on the backtracking path. The variances were set to $(5, 5)$, i.e., the optimal values we used for the real-world evaluation. This results in an almost constant error for $\sigma < 5$, which is also noticeable in the low standard deviation of the PF offline error. From these results we can conclude that HMM is more robust against random errors of arbitrary size. In the next section, we will show that this assumption is also true for real-world measurements.

2) Real-World Error: For the real-world evaluation, we collected a set of location annotated signal sequences measured on 7 different paths through the building. The paths we chose differ in length and complexity, e.g., the number of turns, rooms, and floors. We measured at least 20 sequences per path to collect a representative data set. The different properties of the 7 paths are presented in Table V-D2. The name indicates the floors (EG, OG1) included on the path. *OG1-EG-1* and *OG1-EG-2* also include a stairway to climb floors. *OG1-floor-1* is just a straight path along the floor, not entering any room. The table also shows the average number of samples taken x, the average number of discovered APs, and the total number of measured sequences for that path.

Figure 7 presents the 50, 65, and 80 percentile error of the offline variants compared to LMSE on individual paths and for all paths. Over all paths, HMM performs at least 34 % better than PF and 49 % more accurate than LMSE. However, the performance varies between the different paths. Figure 8 shows the significant advantage of the offline path-matching

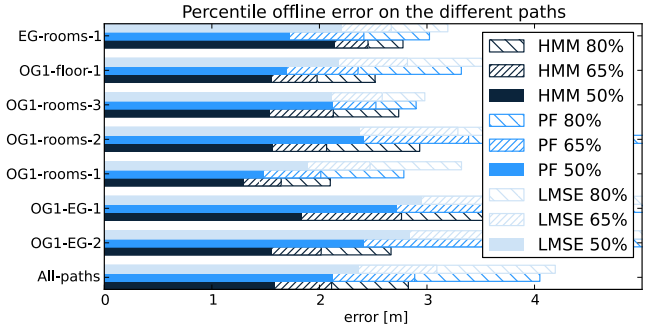

Fig. 7. The 50, 65, and 80 percentile of the offline localization error evaluated on the individual paths and for all paths. 50 % of the real-world offline HMM errors are below $1.6m$, i.e. 34 % more accurate then offline PF and 49 % more accurate than LMSE.

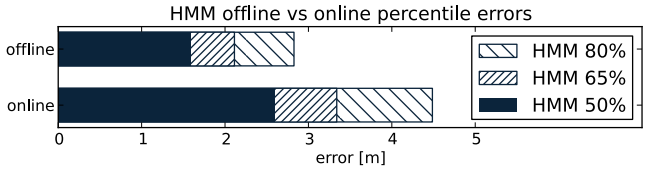

Fig. 8. The 50, 65, and 80 percentile of the HMM offline localization vs. the HMM online localization error over all paths. The offline variant performs up to 60 % more accurate.

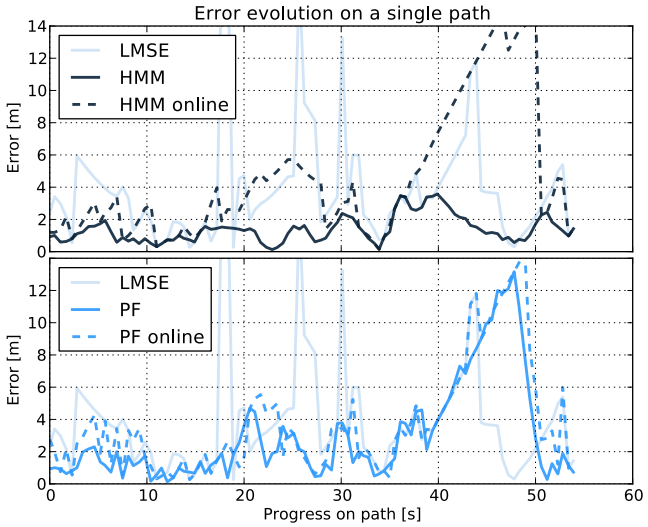

Fig. 9. Evolution of the HMM and PF error for a selected path. The offline variant for both algorithms is able to reduce large errors while HMM performs even better. The LMSE error indicates error-prone regions.

compared to the online variant with respect to localization accuracy. For the 50 percentile the offline HMM performs about 60 % more accurate then offline HMM.

We selected a specific sample sequence from the OG1-EG-2 path to analyze the behavior of each algorithm in more detail. Figure 9 shows two plots of the error evolution of this sample sequence measured along the path. The top figure compares HMM (offline), HMM online, and LMSE, the bottom figure PF, PF online and LMSE.

The LMSE error is a viable indicator for error prone samples. We see that certain samples cause extreme outliers of up to 10 m and more. As expected, HMM and PF are able to eliminate such outliers effectively. Unfortunately, the online

approaches in some cases tend to stick to wrong paths or to particular locations and only recover back to the true path after some time while performing worse then LMSE during this time. This behavior can be seen in Figure 9 between time-frame 35 and 55. Thus, the average error of both online results is about the same as LMSE, but there are significant fewer extreme outliers. This is also true for the PF offline variant, because the errors become too high and the sampling process was not able to find optimal new states in time. In contrast to HMM, PF sacrifices the knowledge about the joined probability of the complete evaluated sequence and only considers the local probability at each intermediate state. The HMM offline variant eliminates the outliers effectively.

A comparison to the errors of the other sequences taken on this path showed that this errors are not caused by random measuring errors, but occur at predominantly the same locations. For example, the crucial region between time-frame 35 and 55 in figure 9 can be identified as the stairway which tends to be particularly error prone due to low AP coverage (< 4 APs) and weak signal strength measurements ($< -80\,\mathrm{dBm}$). Additionally, our mobility model is probably not handling movements through the stairways optimally.

E. Discussion

We took a low effort approach to collecting client measurements. Taking special care in the measuring process w.r.t. the orientation of the device, shadowing effects by people, or time variance, we expect that we can significantly improve the localization performance [1], [3], [16]. We assume that such improvements will affect all algorithms equally and would therefore not change the overall difference between the presented algorithms.

Furthermore, sequential algorithms like PF and HMM could make explicit use of additional information from motion sensors to further optimize the transition probability distributions and the localization results respectively [11]. We did not yet make use of such sensors, because we focused on a pure Wi-Fi based approach. However, the integration of this information in our system looks straight-forward while at the same time, most modern devices such as smartphones are already equipped with both, Wi-Fi and motion sensors, making these systems interesting for further investigation.

Furthermore, we did not yet validate the optimized material parameters in other environments and only did a quick test concerning the stability of our model w.r.t. AP relocation. First results are encouraging. However, we did not evaluate them yet in more detail.

The gain of HMM path-matching compared to online localization offers the opportunity to use the offline variant to optimize the emission and transition probabilities in error-prone regions, e.g., at the stairways, and thereby improve the online results. This form of model training need further investigation in future work.

VI. Conclusion

We proposed a HMM-based Wi-Fi localization approach using raytracing-generated 3D propagation models. We evaluated the performance in both, an offline and an online variant and compared the results with state-of-the-art PF algorithm and a naive LMSE. We show HMM performs best in offline mode on real-world data as well as on synthetic data. Both mobility model supported algorithms (HMM and PF) outperform LMSE using synthetic data. This underlines the general feasibility of path matching as compared to simple fingerprinting.

The performance optimization of the Viterbi decoder and the pruning allows us to perform HMM offline and online localization in real-time on commodity hardware in a high-resolution state space. This enables our approach for a wide range of applications and for real-time localization.

As we showed that the level of detail from a typical floor plan is enough for a reasonable performing localization system, we are confident, that we can further simplify the deployment of Wi-Fi based localization systems with only minimal additional manual calibration effort.

References

[1] P. Bahl and V. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. Ieee, pp. 775–784, 2000.

[2] A. Hatami and K. Pahlavan, "Comparative statistical analysis of indoor positioning using empirical data and indoor radio channel models," in *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, vol. 2. IEEE, pp. 1018–1022, Janurary 2006.

[3] Y. Ji, S. Biaz, S. Pandey, and P. Agrawal, "Ariadne: a dynamic indoor signal map construction and localization system," in *Proceedings of the 4th international conference on Mobile systems, applications and services*, ser. MobiSys '06. New York, NY, USA: ACM, pp. 151–164, 2006.

[4] A. Eleryan, M. Elsabagh, and M. Youssef, "Aroma: Automatic generation of radio maps for localization systems," *CoRR*, p. 15, 2010.

[5] M. Klepal, "Novel approach to indoor electromagnetic wave propagation modelling," Ph.D. dissertation, Czech Technical University, 2003.

[6] K. El-Kafrawy, M. Youssef, A. El-Keyi, and A. Naguib, "Propagation modeling for accurate indoor WLAN RSS-based localization," in *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*. IEEE, pp. 1–5, 2010.

[7] M. Youssef and A. Agrawala, "The horus location determination system," *Wirel. Netw.*, vol. 14, pp. 357–374, June 2008.

[8] C.-H. Chao, C.-Y. Chu, and A.-Y. Wu, "Location-constrained particle filter human positioning and tracking system." in *SiPS'08*, pp. 73–76, 2008.

[9] H. Wang, A. Szabo, J. Bamberger, D. Brunn, and U. D. Hanebeck, "Performance comparison of nonlinear filters for indoor WLAN positioning," in *Information Fusion, 2008 11th International Conference on*. IEEE, pp. 1–7, June 2008.

[10] Q. Chen, D. Lee, and W. Lee, "Rule-based wifi localization methods," in *Embedded and Ubiquitous Computing, 2008. EUC'08. IEEE/IFIP International Conference on*, vol. 1. IEEE, 2008, pp. 252–258.

[11] J. V. Seitz, "A hidden markov model for pedestrian navigation," *Positioning Navigation and Communication*, no. 7th Workshop, pp. 120–127, Mar. 2010.

[12] A. Ladd, K. Bekris, A. Rudys, L. Kavraki, and D. Wallach, "Robotics-based location sensing using wireless ethernet," *Wireless Networks*, vol. 11, no. 1-2, pp. 189–204, 2005.

[13] A. Schmitz and L. Kobbelt, "Efficient and accurate urban outdoor radio wave propagation," *Electromagnetics in Advanced Applications (ICEAA)*, pp. 323–326, September 2011.

[14] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[15] W. Pugh, "Skip lists: a probabilistic alternative to balanced trees," *Commun. ACM*, vol. 33, no. 6, pp. 668–676, June 1990.

[16] H. Lemelson, S. Kopf, T. King, and W. Effelsberg, "Improvements for 802.11-based location fingerprinting systems," in *Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International*, vol. 1, pp. 21 –28, July 2009.