COM SYS Communication & Distributed Systems

RWTH AACHEN UNIVERSITY

Indoor Localization of Mobile Devices Based on Wi-Fi Signals Using Raytracing Supported Algorithms.

Diploma Thesis

Dirk Rothe

RWTH Aachen University, Germany

Chair for Communication and Distributed Systems

Advisors:

Dipl.-Inform.	Nicolai Viol
Prof. Dr.-Ing.	Klaus Wehrle
Prof. Dr.	Leif Kobbelt

Registration date: 2011-08-22

Submission date: 2012-02-22

I hereby affirm that I composed this work independently and used no other than the specified sources and tools and that I marked all quotes as such.

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, den 22. Februar 2012

Abstract

This thesis focuses on the localization problem adapted to the constraints of a raytracer simulated signal distribution for Wi-Fi capable mobile devices in indoor scenarios. The localization problem is defined as predicting the most probable locations for an observed sequence of Wi-Fi signal strength readings. An accurately performing solution is of high interest because Wi-Fi signals can be observed cheaply due to an already widespread deployment of Access Points. For an efficient analysis of the problem, a framework is implemented that combines the raytracing, the localization and evaluation components. Based on this framework, it is investigated whether the raytracing tool provides an effective basis for an accurate Wi-Fi localization system. Furthermore, the performance of a Hidden Markov Model, a Particle Filter and a Nearest Neighbour based localization approach are evaluated on automatically trained raytracer models. Therefore, a representative corpus of location annotated signal measurements is assembled and subsequently employed for a thorough investigation of the algorithm properties with respect to tracking the device in scenes of various complexity. The trained Wi-Fi signal strength predictions diverge in average by $4dBm$ from the real measurements. Under these predictions, the tracking algorithms reach a localization accuracy of about $1.5m$ on pathways and degrades up to $4m$ in complex scenes like stairways.

Acknowledgments

I wish to thank my supervisor Nicolai Viol and Prof. Dr. Wehrle for the opportunity to work on the fascinating subject of this thesis. Without their guidance, their support and especially the challenging and constructive discussions, the thesis would not have been completed with a satisfying result.

Beside my supervisor, I want to thank my team at semantics for allowing me to dedicate my time fully onto the presented topic. Without their efforts, this thesis would not have been possible.

I'm also grateful for all the help of my friends: Mirjam, Kadir, Albert, and Chrissi to proofread the text and dig out all the invisible inconsistencies.

Last, but not least, I must thank my wife Christine for her support and endless patience during the last six months.

Contents

1

Introduction

The problem of determining the location of a person or an object is an ancient one. Many different methods were employed over the recent centuries. For example, the navigation of ships has been supported by referencing to the celestial map of stars, lighthouses or even by transportable devices known as sextants. More recently, tracking the location of vehicles is primarily done with the support of satellite based systems. The first deployed of these systems is the well known GPS system. From air planes over ships to cars, nearly every modern vehicle today is able to determine its position with an accuracy down to a few meters. But the need for localization solutions is not just confined to vehicles of transportation services. For example, due to the now ubiquitous availability of powerful mobile computing devices, the realization of personalized context- and location-aware applications has become an active field of research. But the natural habitats of human individuals, the indoor environments, are dark zones for the signals of the GPS satellites.

The lack of a comparable efficient indoor localization method motivates the research activities into alternative localization systems that are specifically adapted to these environments. Therefore, indoor localization solutions have been based on various information sources that reflect the constraints of the different use-cases. Whereas a hypothesized domestic robot can be specifically designed to carry multiple sensors as optical cameras, ultra-sound or infra-red devices, this degree of freedom is not given for the localization of human beings. There, the sensors need also to be unobtrusive which can be ensured by sensing signals of communication networks.

This thesis will focus on the signals of IEEE 802.11 wireless networks as the primary source of information to approach the localization problem. The important advantage of Wi-Fi, in contrast to other technologies, is the inexpensive hardware and the already dense deployment of Wi-Fi Access Points (APs) in urban areas. For example, at the RWTH Aachen University it is most likely to be in range of at least 5 APs across the campus side [1]. Widespread interest into these signal-strength based

[1] Although it has to be noted, that RWTH is an university with a strong technical background, and thus probably a site with a high saturation of APs. But by interpolating the history it can also be expected that the density of deployed Wi-Fi infrastructure still increases.

localization solutions has been induced by the RADAR [2] system developed at Microsoft Research at the year 2000. The system uses the received signals strengths of a number of APs and an analytical model for the impact of an obstacle on the signal strength to determine the position of a mobile device with respect to a 2D floor map. From the structure of the RADAR system can be concluded that the problem formulation has two major aspects:
What is the distribution of the signal strengths and how is this information processed algorithmically?

1.1 Radio Propagation

The first aspect relates to the nature of the Wi-Fi signals and rises the following questions. How are the signals distributed in the localization space? How are they propagated from the AP source? These questions lead to the concept of radio propagation models. These models can be specified at different levels of complexity but they have in common, that they allow a prediction of the Wi-Fi signal distribution over the targeted areas. This prediction can then be used to drive the decision process that leads to a localization result.

Consequently, the generation of an accurate radio propagation model was the first focus of this thesis. The primary source for the investigated propagation model is the so called PHOTON raytracer [23] that was developed recently by Arne Schmitz at the chair of I8 of the RWTH. The performance of the GPU-driven raytracer, with respect to radio signal propagation, was in the first place examined for urban outdoor environments, but it is designed for the general application to arbitrary indoor and outdoor environments. A basic evaluation of the model capabilities for an indoor scenario was conducted in an earlier work by Schmitz [24].

In order to simulate the propagation of the AP emitted radio signals accurately, the raytracer has to be configured with parameters that relate to the physical properties of the involved entities. The first entity is the AP that is basically configured to be an isotropic radio sender with a scalar antenna gain. The other simulation relevant entities compose the structure of the building and can basically be understood as material annotated scene geometry. Thus, the scene geometry is a mandatory prerequisite for the raytracer and the material parameters have to be determined independently.

The target indoor scenario for the evaluation of this thesis is the UMIC office building with four levels and a size of $15m \times 60m \times 9m$. The 3D geometry of the building was modelled by using the software Blender[2]. 10 different types of materials were defined and accordingly attached to the mesh model. Further details on the model properties and the materials are described more formally in the evaluation chapter 6.1.1.

The parameters of the materials, specifically the coefficients controlling the rate of absorption and reflection of the given building are assumed to be unknown in order to perform the raytracing simulation. To acquire these parameters, a training technique based on evolutionary concepts, more precisely Genetic Algorithms, is devised and

[2]The open source toolkit Blender is freely available at `http://www.blender.org/`.

implemented to distribute the computational demanding search for the unknown parameters over an array of GPU-nodes. The procedure has been determined to yield adequate material parameters for the simulation of the signal distribution over the $2 \cdot 10^6$ voxel[3] resolution of the $8100m^2$ volume for UMIC scene.

1.2 Localization Algorithms

The second topic of the thesis deploys these radio propagation models for the design, implementation and analysis of different localization algorithms. The algorithms exploit the information provided by the propagation models and the available knowledge about the nature of the environment. For example, information about the structure of the building is already available in the form of the scene geometry used for the PHOTON raytracer. From this geometry, for example the knowledge about unreachable zones in the location space can be derived. The algorithms were chosen by studying the related literature to this topic and by applying prior knowledge of Bayesian pattern recognition principles to this field of research. The first of the three analysed algorithms is a nearest neighbour based technique, named Least Mean Squared Error (LMSE), that was also used by the mentioned RADAR system. The second is based on Hidden Markov Models (HMM) and the third one uses a Particle Filter (PF) based approach.

All three techniques can be inferred from the Bayesian decision theory but only the HMM and the PF are conceptually related. The primary difference between the LMSE and the HMM/PF is rooted in its model assumptions that ignore the sequential nature of the tracking problem. With respect to the scope of this thesis, the tracking problem is defined as follows: *Given a sequence of RSSI readings the optimal sequence of locations has to be found.* It is reasonable to assume that adjacent RSSI readings are related due to constraints imposed by the physical world. Or spoken in the terms of probability theory: Temporal adjacent readings are not *statistically independent*. The HMM and PF based approaches presented in this thesis make explicit use of these dependencies, whereas the LMSE does not and thereby retains a simple structure [4].

The HMM and PF technique exploit the additional information contained in the time driven sequentiality of the tracking problem. Both model the concept of movement from one location to another in successive steps. In terms of probability theory: They assume a *conditional probability* for moving to the location s under the condition to come from location s' which should be denoted as $p(s|s')$ [5]. Furthermore, both make use of the radio propagation model to relate an RSSI measurement vector to a location in space. This can be understood as the conditional probability to receive the RSSI vector x at the location s. This probability is denoted as $p(x|s)$. And finally, they combine these two probabilities iteratively for all measurements of the observed sequence to predict the most probable sequence of positions. So where are the differences, why care for both?

[3] A 3D pixel, a discretized volume of space.

[4] The simplicity of the LMSE makes it a valuable tool to evaluate the quality of radio propagation models with regard to the localization problem.

[5] A location is understood to represent an abstract state from the search space of possible locations, therefore it is denoted as s instead of l.

In the HMM case, the locations are assumed to be discrete and enumerable. Therefore, the possible combinations of these locations, the probable solutions to the tracking problem, are enumerable, too. Since these are a huge number of possible location sequences, the HMM model has to make assumptions that restrict the *search space* to a tractable size. Only depending on the quality of the assumptions the algorithm predicts the most probable location sequence of all possible solutions.

The PF makes use of a continuous location space thus avoiding the error induced by the coarseness of an eventual discretization of the space. Instead of *searching* for solutions by enumerating the search space, the algorithm *generates* solutions that are elements of an infinite solution space. Whereas the HMM uses the mentioned conditional probabilities to assign probabilities (or scores) to *all* candidate sequences, the PF generates a *subset* of all solutions by simulating the progress of the modelled stochastic process by sampling from the conditionals. Due to this properties, the PF technique is a member of the Markov chain Monte Carlo methods.

1.3 Framework and Implementation

For the realization of this thesis, a localization framework was designed and implemented to solve the identified problems. The framework handles the training and simulation of the radio propagation models and uses the results as a foundation for the application of the localization algorithms. The final results of the localization algorithms are then either used to predict the current location with the track history, during the online stage, or are later processed for an offline evaluation.

The system is driven by a central Server process that communicates with the producers and consumers of the different data streams over HTTP service interfaces. A prominent producer is given by the mobile devices that push their collected RSSI readings for processing at the Server process, and subsequently consume the results during online tracking. Another producer/consumer is the implemented Fat Client used for visualizing and debugging the localization system with an OpenGL based data analysis toolkit which is especially suited for the 3D nature of the simulated environment. The third component, that is interfaced with the Server over HTTP, are the GPU-nodes that are employed for training the free parameters of the radio propagation models.

The mobile devices and the GPU-Nodes are the simplest of these four primary components, as they are only responsible for high-level I/O. The device with $sensors \rightarrow HTTP$ and the GPU-Node with $HTTP \rightarrow processor \rightarrow HTTP$. Each of the two components consist therefore only of around 100 lines of code. On the contrary, the Server is the most complex component and it depends on a number of subsystems that are responsible for the different tasks of the localization problem. The most relevant subsystems are the Simulator, the Optimizer and the Localizer which are therefore briefly described.

The Simulator is responsible for the organization of the simulation of radio propagation models. It responds to requests for propagation models by dispatching the configuration of the model parameters in a job enclosure to the available GPU-Nodes and returns a job specific result. Many thousands of these requests are queued by

the Optimizer during the training of the model parameters. The Optimizer implements the genetic algorithm approach to the optimization problem. The resulting optimized propagation models are employed by the Localizer component which uses the stored RSSI values as the primary information source for solving the localization problem. Under the assumptions of this thesis, the localization problem is given by sequence of RSSI readings arriving from the mobile device. These readings are subsequently processed by the Localizer through the application of a localization algorithm. Three different algorithms are available, the HMM, the PF and LMSE implementation.

The implementation of the framework is based on the Python programming language. Since Python is very popular in the research communities it has a wealth of third-party libraries that are suited to support the scientific topic of this thesis. The core implementation of the localization algorithms is written in a dialect of Python called Cython [3]. This was necessary due to the slow[6] Python runtime with respect to needs of number crunching algorithms. Cython is a Python-to-C compiler, which enables the prototyping of algorithms in native Python followed by a transformation into an efficient C representation. The transformation is supported by providing type annotations and using dedicated data structures in the form of NumPy arrays. These multi-dimensional NumPy array types provide the basis data structures for the radio propagation models and the localization algorithms.

These design decisions have lead to a flexible framework that can be easily extended if needed. For example, switching from the PHOTON raytracer to another simulator for radio propagation can be accomplished by simply adapting the current PHOTON-specific driver script and ensuring a similar 3D voxel representation of the simulated signal strengths. Furthermore, using the Python/Cython/NumPy stack has lead to fast and memory efficient localization algorithms which has made the evaluation of the system convenient.

1.4 Evaluation

The last part of this thesis is dedicated to the evaluation of the designed, implemented and now presented localization framework. After a thorough description of the conditions under which the experiments of the evaluation were conducted, the first steps of the evaluation will investigate the quality of the PHOTON generated radio propagation models.

It will be analysed whether the proposed training process with genetic algorithms leads to propagation models that can adapt to multiple device classes. The other objective of this part of the evaluation is given by the question how much granularity on the 3D geometry level is needed for the PHOTON raytracer to produce propagation models that represent a good estimate of the unknown true signal distribution. For these tasks that relate to search for the unknown material parameters as training corpus is needed for the optimization algorithm. Such a corpus was collected for 4 different devices with RSSI readings from 100 locations of the UMIC building.

[6] Actually, Python is quite fast for most of the common use-cases in software engineering. Therefore, the ratio of Python/Cython code over the implementation of the framework is about 10:1.

After evaluating aspects relating to the nature of the propagation models, the other major part of the evaluation will focus on the performance of the localization algorithms. The three described algorithms, the HMM, the PF and the LMSE, will use the resulting propagation models from the first part of the evaluation to solve the localization problem on an evaluation corpus consisting of sequences of location annotated RSSI readings.

These sequences represent measurements from eight differently defined paths of various complexity. For example, the most demanding one with respect to the localization problem is a path upward through the stairways over three levels of the building. For each of two different Android devices, an Iconia tablet and a Nexus smartphone, 160 sample paths were taken. This leads to an evaluated distance of about $8000m$ during the real world evaluation.

But before analysing the results on these real world measurements, a synthetic set of measurements with 320 samples over the eight paths will be employed under different noise conditions. This idealized environment will help to evaluate the differences between the HMM, the PF and the LMSE with respect to their algorithmic nature.

Afterwards the evaluation will be finalized by using results of the radio propagation evaluation and the experience from the synthetic evaluation for interpreting the observations that are made in the real world evaluation. It will be seen, that the promising results from the synthetic evaluation are not directly mappable to the localization in natural environments.

Furthermore, the localization algorithms are employed on the results of the radio propagation evaluation that relate to the granularity of the 3D geometry. It will be seen, how complex the scene needs to be modelled to derive a propagation model from the PHOTON raytracer that leads to acceptable localization error rates. In the last part of the evaluation, it will additionally be investigated how well the framework generalizes over more than one device. A propagation model that has only seen measurements from one device will be evaluated on the other. The results of this experiment seem promising.

1.5 Outline

The structure of this thesis is given as follows: After this introduction, the concepts used in the framework and needed for understanding the localization algorithms are explained in the background chapter 2 which is followed by the related work chapter 3. In chapter 3, comparable approaches to the localization problem found in the literature will be investigated. By building on the foundations lain in the background chapter, the design chapter 4 is structured. There, all major components of the framework and their interactions are described comprehensively. An overview of the implementation is given in chapter 5 which is finally followed by the thorough evaluation of the system presented in chapter 6. In the last chapter 7 conclusions will be drawn and an outlook into further research activities will be given.

2

Background

In this chapter the background to the two main topics of this thesis is presented. These are radio propagation models and localization algorithms. A radio propagation model is used to compute the propagation of Wi-Fi signals and lays the foundation for the localization methods developed in this thesis. Therefore, the challenges of radio propagation will be discussed in general and different technologies to compute the signal propagation of Wi-Fi terminals are presented. Thereby, the raytracing technology is discussed in more detail because it is able to compute most accurate propagations and is therefore used for the further work of this thesis. Furthermore, a technique for automatic training of the material parameters, thereby enabling the generalization to unknown scenes, will be provided. The technique of choice is optimization with Genetic Algorithms. Finally, the two main error measures for evaluating the quality of the model are defined.

The chapter continues by introducing the two different variants of the localization problem: Positioning and Tracking. Furthermore, the possible sources of information, that can be exploited by a localization algorithm, are described. A focus was placed on the signal strength information given by RSSI values that are receivable with Wi-Fi capable devices. For the tracking problem, additional information sources are presented in the form of mobility models. For both sub-problems, the corresponding error measures will be defined.

After describing the basic ideas of simple positioning approaches in 2.2.1, this chapter will end with a detailed presentation of more sophisticated models. These are the Hidden Markov Model and the Particle Filter, as both of them are more suited to the tracking problem and have therefore been evaluated in this thesis.

2.1 Radio Propagation Model

In this thesis, two main approaches for modelling the propagation of radio signals are distinguished by the following reasoning: Propagation models are used to construct

an accurate signal strength map (SSM). A SSM represents the distribution of RSSI values from an AP over the space of an indoor scene. Therefore, the two approaches are distinguished by which means these RSSI values are obtained. The first approach, named *empirical radio propagation model*, is based on the technique to collect a significant amount of real world RSSI measurements, so that the propagation model can afterwards predict RSSI estimates for arbitrary locations. It is crucial for this approach to gather enough information about the interesting zones of the scene. Furthermore, the data should preferably be collected homogeneously, for example, by applying a grid to the location space. It can be seen, that depending on the resolution of this grid, the construction of an empirical propagation model can be a laborious undertaking. Additionally, this approach becomes even more expensive if changes in the environment happen, for example by relocating APs or reorganisation of furniture. Such changes make a recalibration of the propagation model mandatory.

Due to the expensive nature of the empirical approach, the research in this areas has been focused on the alternate idea to create the sought RSSI distributions artificially by reasoning about the rules of radio propagation. Therefore, the class of these models is named *analytical radio propagation models*. The most basic one is given by assuming an idealized free-space environment and the corresponding quadratical signal power loss with respect to the distance between the current location and the sender. This radio propagation model is called *ideal path-loss model* and has therefore been ranked lowest in figure 2.1. The first adaptation to the environment conditions is done in the *general path-loss model* by assuming a linearly elevated quadratical power loss. The linear coefficient has to be determined empirically and can be assumed to be higher for scenes with more Non Line Of Sight (NLOS) than LOS conditions as more obstacles in the scene lead to a higher probability of signal absorptions.

Therefore, the analytical models needs to be adjusted with empirical estimated parameters as well. These unknown parameters of the analytical propagation models are called *free parameters*. Another parameter driven analytical model has been described in the RADAR system [2]. The presented model is the so called *Wall Attenuation Factor* (WAF) model. The basic assumption is given by an assumed constant signal decay at each obstacle intersection on the straight line between the sender and the simulated location in space. The model is easily enhanced to simulate different types of obstacles, and will therefore be called multi-material WAF. The free material parameters of the multi-material WAF have also to be found empirically. And the only physical effect, that the WAF simulates is an absorption of the signal

An alternative approach, the *dominant path model* [1] adds the simulation of signal reflection. The change of direction of the signal at a material intersection is computed and the signals on the dominant paths (i.e. the signals with the maximum power) are traced until exhaustion. Their aggregated information of the traced paths will be used as the basis for the SSM.

As a general rule, more sophisticated radio propagation models can be obtained by simulating more of the physical effects that influence the propagation of radio signals. Such effects are especially found at the transition boundaries of optical media, for example between air and solid material. The following physical effects can be considered:

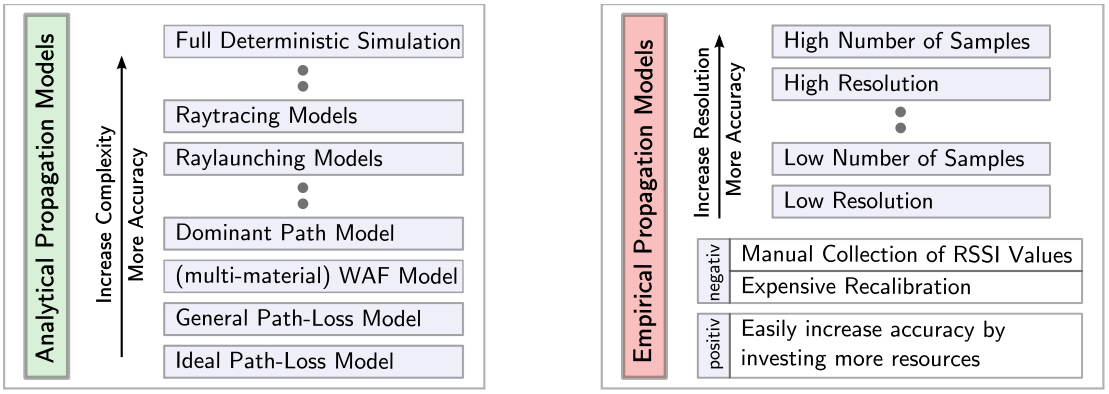

Figure 2.1 This figure shows the hierarchy between analytical radio propagation models of different complexities on the left side. On the right side, the other major approach, the empirical propagation model, has been highlighted.

Absorption: The rate of power loss at a media intersection, as some of the energy of the signal is transformed into other energy forms as for example heat.

Reflection: The change of direction of a signal at an interface between two different media. The angle of the incoming direction equals the angle of the outgoing direction with respect to the surface normal (see figure 2.2). Only a part of the signal is reflected, the remaining part of the signal follows the rules of the next effect.

Refraction: The change of direction at a media intersection for the unreflected part of the incoming signal. The unreflected part of the signal enters the new optical medium, and depending of the frequency of the underlying wave of the signal and the refractive index of the medium, the direction of the signal changes.

Diffraction: If the underlying wavelengths of the simulated signals are in the magnitude of the physical obstacles, the diffraction effect leads to a change of directions in the form of bending around the corners of these objects.

Interference: The phenomenon, that the overlapping waves lead to new wave forms. These superimpositions can either increase or decrease power of the signal.

Scattering: Small obstacles in the size of particles and rough surfaces induce a noisy reflection of the signal.

Polarization: The orientation of the underlying waves for the radio signal influences the absorption rate at different material types.

A simulation of the radio propagation, obeying to a selection of these effects, can be obtained by raytracing algorithms. A requirement for such a raytracer simulation is the knowledge of the material parameters controlling the absorption rate and the change of direction due to reflection, refraction and diffraction. The combination of these material parameters controls the rate of signal strength decrease at media intersections and the amount of multipath effects due to reflection and diffraction.

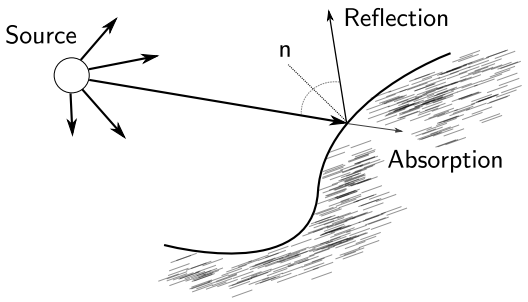

Figure 2.2 The two optical effects reflection and absorption occur at the transitions between materials with different optical densities. They can be simulated with raytracing algorithms. The employed PHOTON raytracer, described in 2.1.1, is configured to model only these two effects.

Furthermore, an initial signal strength for a simulated ray is needed. A simple model of these optical effects, as primarily used in this thesis, is defined by a numerical value for rate of reflection and the rate of absorption for each material and one additional parameter for an initial signal strength of each AP class. The values of these parameters are assumed to be initially unknown and have to be trained. Therefore, the number of trainable parameters n_{free} for the employed model is:

$$n_{free} = 2n_{mat} + n_{apcls}$$

with n_{mat} as the number of different materials and n_{apcls} as the number of different AP classes.

A strategy for finding these free parameters of the model consists of comparing real world measurements at different locations with the corresponding simulation results of the raytracer and use the minimum averaged error over all locations as the target for an optimization algorithm. Of the different parameter optimization methods, that were evaluated, Genetic Algorithms, described in 2.1.2, have shown to be most effective.

2.1.1 PHOTON Raytracer

The PHOTON raytracer [23] that is used[1] for the radio propagation model represents a deterministic approach to model radio propagation with concepts from geometrical optics. The signal of a radio wave is modelled as a single particle, called PHOTON, that travels on a straight line until intersecting with another physical medium. At these intersections the physical effects reflection and absorption are simulated. Even though it is possible to simulate behaviour of radio propagation that is more complex than diffraction, this has not been used in the presented setup. Instead a simpler form of Bidirectional Reflectance Distribution Function (BRDF) is configured by the reflection parameter $\beta \in [0, 1]$ and an absorption parameter $\alpha \in [0, 1]$. The

[1] During the preparation phase of the thesis a fast 2D raytracer has been implemented that was able to simulate 60 raytracing frames per second with basic transmission and reflections rules. It would probable be possible to use this implementation to simulate dynamic effects as opening of doors or the body shadow of moving people.

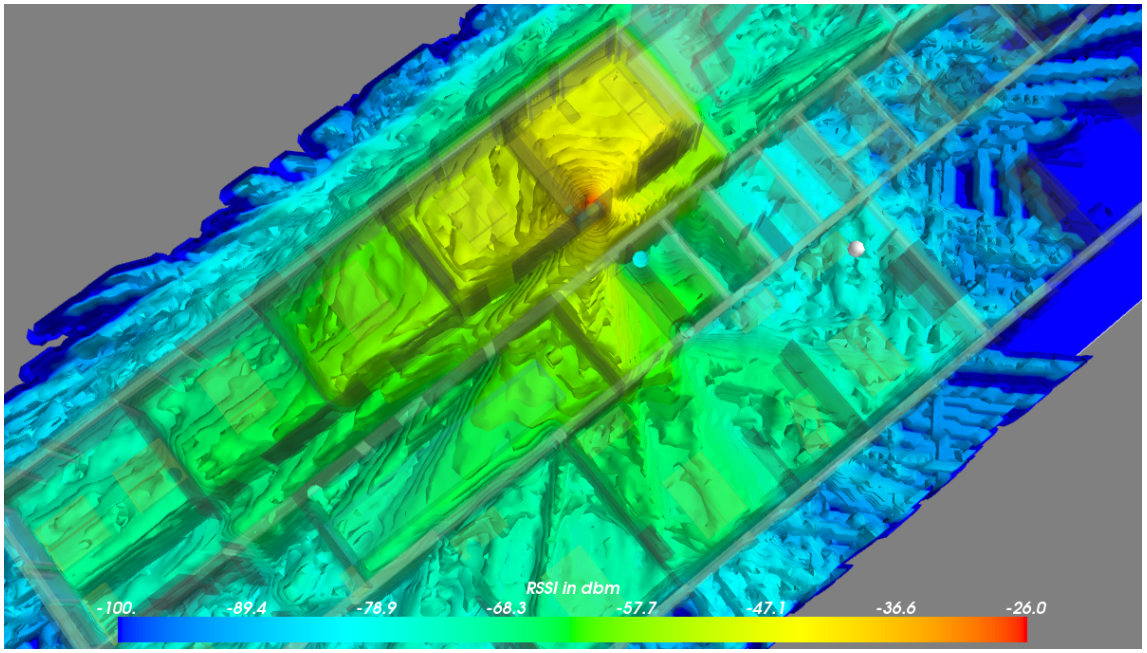

Figure 2.3 Signal Strength Map for one AP from a raytracing generated radio propagation model. This is an example of the output from the PHOTON raytracer for an AP of the UMIC scene on a voxel size of $0.2m$

evaluation of these parameters works as follows if N is the number of PHOTONs hitting a material:

1. $(1-\beta)N$ PHOTONs are absorbed, therefore βN PHOTONs remain for transmission.
2. $(1-\alpha)\beta N$ PHOTONs are refracted and continue to travel at the same direction and the remaining $\alpha\beta N$ PHOTONs are reflected and travel in the inverted input direction.

After simulating the paths of all PHOTONs by applying the BRDF recursively, the paths are drawn into the 3D space as voxelized lines. These lines are further smoothed through an anti-aliasing step. The sum of the individual power values of all overlapping lines determines the resulting power value of a voxel therefore handling multipath effects as in [11]. A final smoothing step is given by applying a three dimensional Gaussian filter.

The raytracer has been developed for the simulation of radio waves emitted by GSM base stations in outdoor intracity scenarios. Experiments have shown an average error between measured and predicted signal strength within $6-8dBm$.

The free parameters of the PHOTON raytracer can be estimated by Optimization Algorithms. The author of the raytracer has successfully employed the Levenberg-Marquardt algorithm. But in the context of this thesis, Genetic Algorithms have shown lower errors in the range of $3-5dBm$ for the UMIC indoor scenario.

Another interesting possibility of the PHOTON raytracer is the modelling of antenna patters by using Spherical Harmonics [22]. Spherical Harmonics can be compactly

represented by a set of coefficients for the corresponding functional forms. These coefficients lead to another set of free parameters, and therefore a larger search-space for the Optimization Algorithms. Due to the increased search-space, more training data in form of manual measurements are needed to offset for a phenomenon commonly referred to as Curse of dimensionality. This phenomenon describes the statistical problems that arise when the volume of the high-dimensional space increases so fast that the training samples become sparsely distributed. Therefore, this technique was not employed in this thesis for the sake of simplicity.

2.1.2 Optimization with Genetic Algorithms

By using an Optimization Algorithm, it is possible to find a set of parameters for a model that minimizes a given cost function. If the cost function evaluates the error or the "quality" of the model, the found parameters are optimal with respect to the cost function. A set of n_{free} parameters can also be understood as an element of a n_{free}-dimensional search-space in $\mathbb{R}$. The cost function that is given by a n_{free}-parameter controlled raytracer run can be assumed to be non-linear and non-differentiable due to the recursive nature of the involved algorithms. Furthermore, it can be safely assumed, that the function is non-convex leading to multiple local optima.

Of the three evaluated heuristics: Minimum Least Squares, Simulated Annealing and Genetic Algorithms, the last one was capable to generate the best results with respect to the reached optimum. In a Genetic Algorithm a set of parameters is called a candidate solution (in the search-space) or simply referred to as an organism. The search for the best set of parameters, also referred to as the fittest organism, is an iterative procedure. The procedure starts with an initialization step where a predefined number of organisms are created randomly. Depending on prior information it is sensible to seed organism in regions of the search-space where optima are more probable. In the context of the given optimization problem, it makes sense to use prior knowledge of the estimated power levels of common APs.

After the initialization, the fitness of each organism of the population is evaluated by calculating the result of the underlying cost function. In the present use case this means a full raytracer run over all APs and the aggregation of the error at all measured locations. Then, the main iteration of the algorithm starts by choosing a proportion of the population for breeding by using the magnitude of the fitness of the organisms as the selection criteria. Breeding leads to new organism and therefore to the exploration of the search-space. New organism are breeded by choosing a pair of parent organisms and crossing over their genes (the n_{free}-parameters) by selecting genes randomly from each parent. Additionally to this random selection, a random mutation of the genes can also be applied to allow for a deeper exploration of the search-space.

For each breeded organism the fitness will then be evaluated and used to select a predefined number of the fittest children as the new population for the next iteration step. Different termination conditions can be chosen, like a maximum number of generations/iterations or a minimum needed change toward a minimum cost target. Through this evolutionary inspired selection process the convergence to an, at least local, optimum of the cost function is guaranteed.

2.1.3 Error

The error for a radio propagation result will be determined by comparing the corpus of measurements m with the simulation results x for each AP configured with n_{free} .

$$RPE(n_{free}) = \frac{1}{N_{aps}N_{locs}} \sum_{a=1}^{N_{aps}} \sum_{l=1}^{N_{locs}} ||m_{a,l}, x_{a,l}||$$

where $x_{a,l}$ is the simulation result for AP a and location l. Likewise $m_{a,l}$ is a collected measurement for AP a and location l. N_{aps} is the number of available APs and N_{locs} the size of the training corpus. For $||m, d||$ the l_1-norm, the absolute delta $|m - s|$ is used. Therefore, the Radio Propagation Error (RPE) is simply the averaged error over all collected measurements at the different locations in units of dBm.

Another variant of this error that is used in literature is the RMS-RPE. This error uses the rooted averaged euclidian distance, or the l_2-norm, as a metric and is given by:

$$RMS - RPE(n_{free}) = \frac{1}{N_{aps}N_{locs}} \sqrt{\sum_{a=1}^{N_{aps}} \sum_{l=1}^{N_{locs}} (m_{a,l}, x_{a,l})^2}$$

The unit of this error is also given by dBm.

2.2 Positioning

The task of Positioning is defined as determining the physical position of a stationary device by using information received by the sensors of that device. This information, extracted from some measured signals, is obviously required to be related to the that position for relevancy. The measured signals are usually obstructed by noisy effects that come from various sources. The performance of a positioning system is defined over the error that is given by the distance between the real position and the estimated one. A well performing positioning model will therefore have to compensate these noisy effect for minimizing the position error

The source of information that is exploited for positioning in this thesis is given by measuring the Received-Signal-Strength-Indicator (RSSI) values of available APs with Wi-Fi capable devices. The RSSI value is a measure of the magnitude of the electromagnetic field at some physical location. The field is emitted by the antennas of an AP with a known location. The RSSI value is expressed in dBm which represents the remaining power of the emitted electromagnetic field in relation to the reference unit of one milliwatt. And explicitly by:

$$x = 10log_{10}(1000p)$$

if p is the power at the source of the electromagnetic field given in watt and x is the measured RSSI value.

Alternate interesting sources of information that are exploited by positioning systems and that were analysed in recent research [10] are:

- Time Of Arrival: TOA based methods deduce the distance between transmitter and receiver by comparing the timestamp of a packet, originating at the transmitter with the local timestamp of the receiver. Prior knowledge of the speed of the transmitted signal combined with the timestamp difference can be used for estimating the covered distance of the signal. A source of error is introduced by asynchronous clocks and NLOS conditions that lead to multipath effects.

- Time Difference Of Arrival: Methods based on TDOA use the difference of two TOA measurements emitted by signals at exactly the same time at different APs. By using only this difference the requirement of a synchronized clock between the different transmitting APs and the receiving device can be dropped. But errors induced by timestamp affecting NLOS conditions remain.

- Angle Of Arrival: AOA based methods rely on measuring the angle of the incoming signal at the receiver with directional antennas. A source of error is induced by NLOS conditions leading to receiving signals of the same AP from different directions. And another error source in the probable incomplete knowledge of the orientation of the receiver. The requirement of directed antennas at the receiver excludes the use of the commodity WLAN hardware that is currently available.

Using the RSSI value as the primary source of information for the positioning system makes a good radio propagation model mandatory. Two large sources of errors are expected. At first, there is the error originating in the unpredictable measurement behaviour or other noisy effects of the Wi-Fi capable devices. And the other class of errors originates in an inadequate modelling of the radio propagation. By using a raytracing generated radio propagation model, the reduction of such errors was a major focus in the presented approach. Especially the inherent modelling of NLOS conditions makes a raytracing approach promising.

Without a raytracer, one has to resort to approximate the dampening effects of walls by introducing an attenuation parameter that determines the magnitude of dampening at a material intersection. Such an attenuation parameter would be highly material dependent. In the raytracer approach, the corresponding modelling is represented by the interaction of the α and β material parameters of the employed basic BRDF.

Another source of noise with a high impact on the RSSI values is given by the shadow effect of the human body. Radio waves with 2.4GHZ are easily absorbed by materials with a high proportion of water. Furthermore, it is expected that location aware devices are attached or very close to the owner of the device, therefore boosting this dampening effect.

2.2.1 Techniques

There are different positioning techniques that have been developed by using the mentioned information sources. One of the simplest techniques is called Proximity Sensing that uses only the identity of the transmitter instead of any distance or angle

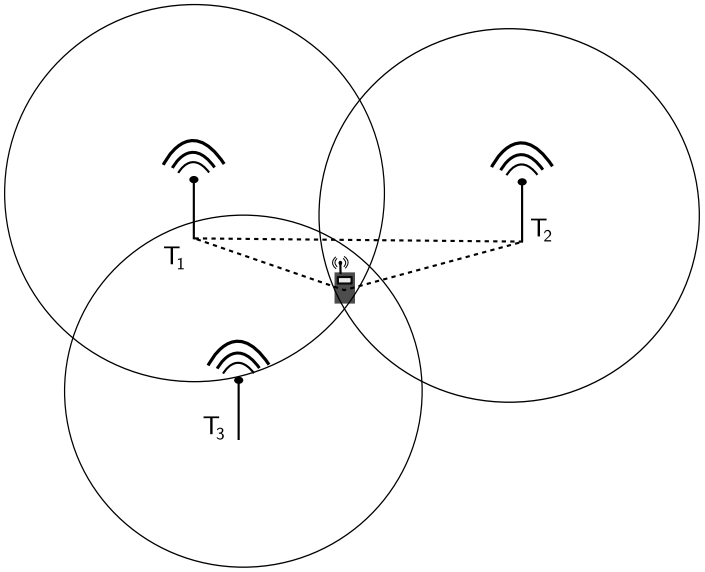

Figure 2.4 Positioning with triangulation techniques. The length of edges of the triangle is determined by ratio of the RSSI values of between the transmitters T_1 and T_2.

related measure. The position of the receiver is assumed to be the position of the transmitter. If multiple transmitters are available, a choice has to be made between them. This choice is guided by using the maximum signal strength, and therefore introducing a range related information.

Explicit usage of range or angle related information is made by techniques that rely on triangulation for positioning. An example of such an approach is called Lateration. Lateration uses the ratio between the RSSI values of two transmitters for finding the two locations in a 2D space that have the property to be the third point of triangle that includes both transmitters as the other points. The triangle is defined by the property that the variable edges between the transmitter and the unknown location of the receiver have the same ratio as the RSSI values. The approach can also be generalized to either multiple transmitters or other information sources like TDOA, then called Hyperbolic localization, or AOA, then called Angulation.

Another group of techniques can be gathered under the topic of fingerprinting. These approaches rely on knowledge of the surrounding of the transmitters. These knowledge has either to be empirically collected or analytically modelled and has to be stored for efficient access during the localization phase. In the localization process the collected information is compared with the stored data and the best matching location is chosen.

The empirical construction of a fingerprinting map makes it mandatory to rebuild this map if the environment changes. By employing analytical models, like raytracing generated signal strength maps, it is easier to adapt to environment changes. But the initial costs of such models can be higher. For example in the case of the raytracer, a 3D Map containing probably unknown materials is needed.

2.3 Tracking

Tracking is the generalization of the Positioning problem. Whereas Positioning is defined as a stationary localization problem, Tracking drops the immobility constraint by allowing the receiving device to move over time. The simplest approach for Tracking is therefore the sequential execution of a Positioning algorithm with disregard to any structural dependency between the information at different timestamps. But doing so, does surely yield an inferior localization result, as the sequential nature of the tracking problem is a source of valuable information. Prior knowledge like the maximum walking speed, that is usable as a constraint on the maximum distance between two successive positioning results, can easily be exploited.

It is also required to incorporate this source of information in order to offset for the much larger search-space that is given by the Tracking problem. The search-space for the Positioning problem is linearly dependent on the resolution and the size of the modelled space. In the worst case that is a high resolution 3D space as used in the UMIC scene, with around $2 \cdot 10^6$ solutions representing cubes with edge size $20cm$. In contrast, the solutions for the Tracking problem are sequences of locations with an additional measurement specific resolution that determines the length of that sequence. This length T has an exponential impact on the size of the search-space. An input sequence of signal vectors x with T timeframes, represented as x_1^T, leads to a solutions sequence s_1^T. And if the representation of the space is made of S disjunct positions, this would induce S^T possible solution sequences. Consequently, a brute force search, probably computationally tractable for the singular positioning problem, has to be excluded as an algorithmic attempt for the Tracking case.

There are two approaches to model the state space of possible locations. Either the space is assumed to be rasterized or segmented into ”spaces” of interest with some resolution factor for adjusting the granularity, or space is assumed to be dense with real values for the two or three possible dimensions. The first approach leads models based on Markov chains like Hidden Markov Models (HMM). Since such models have a finite number of states, the computation of a solution involves making decisions between different states by relying on the evaluation of their properties. A major part of this thesis studies different aspects of HMM based models.

In contrast, in the second approach a position, given in real values, is updated by some function configured with prior knowledge of the environment or of the behaviour of the moving person. The evaluation of this function results in the next-best real valued position. Examples of this approach can be found in the form of Particle Filters or in the different forms of Kalman Filters.

2.3.1 Mobility Models

The different approaches for tackling the sequential nature of the tracking problem have in common, that they use prior information of the shape of the environment or prior knowledge of the rules that a moving device has to obey. There are multiple sources for extracting such information.

A deterministic mobility model can be employed, if the speed and acceleration of a moving device are available. Combining such information with the laws of physics,

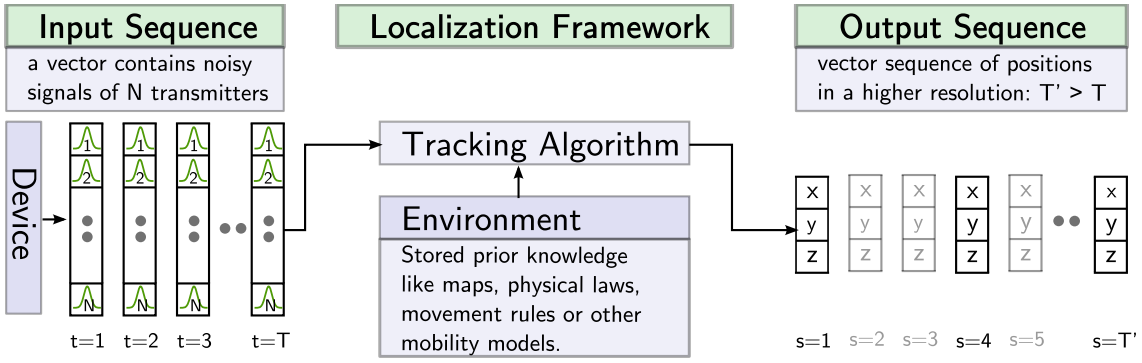

Figure 2.5 The Tracking problem consists of finding the best output sequence $s_1^{T'}$ of positions for an input sequence of measured signals. Both sequences are not required to have the same length due to probable further preprocessing steps.

especially the Newtonian laws of motion, gives valuable hints for finding the best solution for the tracking problem. Tracking vehicles in an outdoor scenario is a classical example for a case where the corresponding required runtime information is also available from the built-in sensors. On the other hand in indoor scenarios, it is less likely that the availability of information as speed or acceleration can be presumed.

Another class of mobility models are derived from stochastic processes. A very simple one is given by the Brownian motion or more general the Random walk model. Both models simulate movement by stepping forward in a randomly chosen direction. The Brownian motion model assumes an asymptotically small step-width approaching zero. These models can easily be enhanced to use other probability models instead of simple uniform or normal ones. This can be done by incorporating the history or the predecessors of states in the sequence and leads to the formalism of Markov chains.

The information of direction is already used in a first order Markov chain, but for the evaluation of direction changes higher order Markov chains are needed. For example in the presented localization system, the direction information is used in a first order HMM to penalize movements into directions with less free space. The degree of free space at a given location and a given direction can be retrieved from the 3D scene.

This leads to the so called Geographical-restriction models. These models derive rules for the transition from one state to another by extracting information from an available map of the environment. Such a rule is given by disallowing the transitions into regions of space, that are blocked either directly or indirectly by obstacles. A further modelling of temporal variations in these restrictions can be used to mark obstacles as active or inactive. An obvious use-case is given in the form of time controlled door locks.

Another simple model, driven by prior geographical information, is the so called Pathway mobility model for vehicle navigation. A map containing a grid of streets and intersections is used to model movement restricted to the streets with a simple decision rule for the crossings. At each crossing the probability of moving forward is 0.5 whereas the movement to the left or the right is 0.25. The constraints of such

a model can easily be projected into an indoor scenario, but here, also allowing a change in the opposite direction should also be considered.

The last class of mobility models rely on exploiting the behaviour of multiple synchronously moving devices. Therefore, they are called Group mobility models. It is for example plausible to assume, that vehicles on a street that are near to each other, have correlated speed and direction properties. The same holds true for swarms of animals or human movement on crowded places.

2.3.2 Error

The error of a Tracking result s_1^T can be given in the form of the Root Mean Square Tracking Error (RMS-TE). The RMS-TE between the correct sequence of positions r_1^T and the estimated sequence s_1^T is defined as:

$$RMS - TE(s_1^T, r_1^T) = \sqrt{\frac{\sum_{t=1}^{T}(||r_t, s_t||)^2}{T}}$$

with $||r_t, s_t||$ as the 2D or 3D euclidian distance between the *real* positions r_t and s_t. The performance of a localization system can be given by evaluating the error over multiple tracking attempts stored in an evaluation corpus C.

$$RMS - LE(C) = \sqrt{\frac{1}{N}\sum_{c=1}^{N}\frac{1}{T_c}\sum_{t=1}^{T_c}(||r_{c,t}, s_{c,t}||)^2}$$

where T_c is the length of a sample from the corpus. This error can also be evaluated for the positioning problem by assuming that $T_c = 1$ for all pseudo-paths of the corpus. In literature a variant of this error is also given by using the simpler l_1-norm:

$$LE(C) = \frac{1}{N}\sum_{c=1}^{N}\frac{1}{T_c}\sum_{t=1}^{T_c}|r_{c,t} - s_{c,t}|$$

This error should be referred to as the averaged Localization Error evaluated on a collected corpus of tracked paths. All three error variants are given in the unit meter.

2.4 Bayesian Pattern Recognition

For understanding the approaches to the localization problem, that are presented in the following sections, a general understanding of the basic principles of Bayesian inference is needed. Therefore, a short introduction into this very popular approach to the problem of machine learning and pattern recognition is given.

In Bayesian inference, the Bayes theorem is used to compute how the degree of belief in a proposition changes due to available evidence. In the context of RSSI information based localization, the proposition is: "The device is there." with the evidence: "It has received these RSSI readings". Since such a proposition is inherently a *decision* for a *state*[2] in a however modelled environment, it will be represented by s.

[2]Another convention for formalizing the concept of the proposition is given by the notion of a *class* that will be decided upon.

The simplest state space is defined by the 2-state case: "The received email text is either spam or ham". In this minimal example, the previously mentioned *evidence* is a representation of the text in a machine processable form. Such a form of the evidence is commonly called *Feature Vector* and should be denoted as x.

For the "decision making process", that should be called Decision Rule, it would be beneficial to have a measure of the quality of the different possible decisions, the different possible states under the feature vector represented evidence. In Bayesian inference this measure is given by the joint probability between the state s and the feature vector x with:

$$p(x, s)$$

which evaluates under the definitions of probability theory to a scalar value in the range [0..1]. Furthermore, if x and s are conditionally independent, the joint probability can be factored into the two conditional forms that are essential for the Bayes theorem:

$$p(x, s) = p(x|s)p(s) = p(s|x)p(x)$$

Under the Bayes theorem, the interpretation of the probabilities $p(x|s)$, $p(s)$ and $p(s|x)$ is indicated by their commonly used names. $p(s|x)$ is called the *posterior probability* as it represents the belief that the state *follows* the evidence given by x. $p(s)$ is the *prior probability* and represents an evidence independent knowledge about the probability how often the state s will be observed. And finally, $p(x|s)$ is the *state-conditional probability* that represents the probability to observe the evidence x under assumption that the environment is in state s. $p(x)$, the probability to observe a specific evidence, has not been given a dedicated name as it will be unimportant for the sought Decision Rule.

The Decision Rule should obviously lead to a high quality decision. This should therefore be the decision with the *maximum* joint probability. By this reasoning, the Bayes Decision Rule $r_{bayes} : x \to s$ is defined as:

$$r_{bayes}(x) = \underset{s}{\operatorname{argmax}}\, p(x, s)$$

which factors according to the laws of conditional probabilities into:

$$\begin{aligned} r_{bayes}(x) &= \underset{s}{\operatorname{argmax}}\, [p(s|x)p(x)] \\ &= \underset{s}{\operatorname{argmax}}\, [p(s|x)] \quad , since \ \underset{s}{\operatorname{argmax}} \ is\ independent\ of\ p(x) \end{aligned}$$

This is an intuitive result. A decision that is based on the posterior probability leads to the same result as using the joint probability. But it is still unknown how to obtain the posterior $p(s|x)$. Therefore, the Bayes theorem will be used again:

$$p(s|x) = \frac{p(x|s)p(s)}{p(x)}$$

Inserting the factored posterior into the Decision Rule:

$$\begin{aligned} r_{bayes}(x) &= \underset{s}{\operatorname{argmax}} \left[\frac{p(x|s)p(s)}{p(x)} \right] \\ &= \underset{s}{\operatorname{argmax}}\, [p(x|s)p(s)] \quad , since \ \underset{s}{\operatorname{argmax}} \ is\ independent\ of\ p(x) \end{aligned}$$

This is not an intuitive Decision Rule any more, but probability models for $p(x|s)$ and $p(s)$ can be *learned* from the environment. The prior $p(s)$ is a discrete PDF, due to the discrete nature of the states[3], that can be determined by simple counting of the occurrences of s.

Modelling the state-conditional distribution $p(x|s)$ is more complicated. The vector space of x, the numerical representation of the evidence, can have manifold shapes. It can be of categorical nature leading to a discrete PDF, or, if it represents physical measurements it will become a continuous one. The analytical choice of the form of the PDF, whether it is a multi-variate Gaussian, a mixture density or another complex distribution, can be termed: *to apply model assumptions*. If the prior analysis has lead to a $p(x|s)$ that is modelled according to the true nature of the environment, the *free parameters* of the model need to be determined. Similar to learning the structure of the prior $p(s)$, the parameters of $p(x|s)$ can be learned from the environment. But due to the coupling of s and x, special *state-annotated* evidence-data is needed. Ignoring the problem of gathering this data, parameter estimation techniques like Maximum-Likelihood can then be applied to the set of training samples. If $p(x|s)$ is modelled as a Gaussian, this results in estimating the mean and the variance.

Summarizing the results: The Bayes Decision Rule conducts a *search* for the state s with the maximum posterior probability $p(s|x)$ for an observed feature vector x. The Bayes Decision Rule is therefore a function with input given by some measured evidence x leading to the output of the most probable state s of the environment. Instead of directly evaluating the posterior probability $p(s|x)$, the prior $p(s)$ and state-conditional $p(x|s)$ are employed as they can be learned from the environment. $p(s)$ can be easily learned by counting. And for $p(x|s)$, suitable model assumptions must be chosen and the free parameters of the model need to be trained.

If these concepts are applied to the positioning problem with RSSI measurements, this leads to the following example model:

1. A state s is an enumerable region of space, a location.

2. The feature vector x is the jointly received vector of RSSI values for different APs.

3. $p(s)$ is the probability to be in a specific location. In a geographically-restricted mobility-model, $p(s)$ would be zero for unreachable regions.

4. $p(x|s)$ is the probability to receive the measurements x at the location s. $p(x|s)$ can be modelled as a multi-variate Gaussian, with a mean vector that represents the anticipated AP-specific RSSI values at the location s. Assuming equal noise over all APs, a signal variance of around $5dBm$ will be chosen.

5. The AP-specific means of $p(x|s)$ will be obtained from a radio propagation model.

[3]It is possible to assume a continuous state space as well, but this has not been done for simplicity.

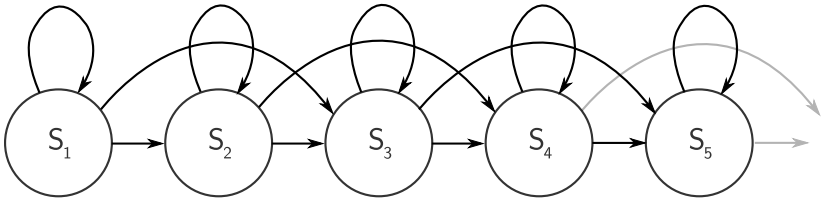

Figure 2.6 Markov chain with a $(0, 1, 2)$ transition model. The choice of the future state depends only on the present state.

For a new RSSI vector observation x the Bayes Decision Rule is used to decide for the most probable location s that explains the observation. This means evaluating the posteriors $p(s|x)$ for *all* all locations, and selecting the location s with $\max(p(s|x))$. Due to the unavailability of a direct form of $p(s|x)$, the maximization is carried out over the known prior $p(s)$ and the state-conditional $p(x|s)$.

If the prior $p(s)$ is assumed to be constant for all s, then this will lead to the LMSE based localization approach that will be described in section 2.7.

2.5 Hidden Markov Models

The Hidden Markov Model approach to the localization problem leads to the first algorithm that is based on the principles of Bayesian inference. But before the nature of the state-conditional is discussed, the formalism of the Markov Chain is introduced to derive the source for the prior probability.

The process of movement through space can be modelled as a Markov chain. Each possible discrete position in space, their number depends on the rasterization resolution, translates to a state in the Markov chain. A Markov chain is a sequence of states in a stochastic process where the Markov property holds. The Markov property refers to the memorylessnes of the process, that is given by the constraint that a future state depends only on the present state and ignores all other preceding history.

Such a Markov chain is parametrized by transition probabilities. The transition probabilities form a discrete probability distribution. A transition is the pair $s' \to s$. The conditional probability for a transition, the probability that the future state s follows after the present state s' is given by $p(s|s')$. The normalization constraint of a PDF holds:

$$\sum_{s} p(s|s') = 1, \quad \forall s'$$

A special transition model, allowing only three predecessor states, is the $(0, 1, 2)$-model (see figure 2.6) that is defined by:

$$\sum_{i=0}^{2} p(s+i|s) = 1$$

with $s + i$ representing a state index as the notation s_t is used for indexing over time frames. The $(0, 1, 2)$-model is used for time alignment, that has the goal for

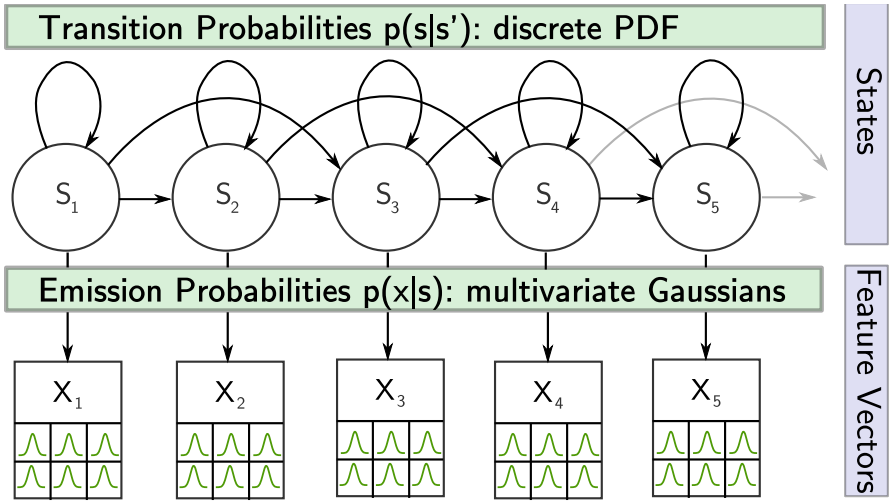

Figure 2.7 In a Hidden Markov Model the states are not directly observable. But the output of the states can be measured as sequence of feature vectors. From these observations the best matching state sequence s_1^T can be recovered.

compensating distortions in the speed of the stochastic process. If the process is only developing slowly, for example a slowly moving localization target, more 0-transitions can be used. In a 0-transition the process remains in the present state. The opposite effect have 2-transitions. They are employed to model accelerated phases of the stochastic process. Elevating the one dimensional $(0, 1, 2)$-model into the three dimensional localization space with six degrees of freedom leads to the rather clumsy notation of $((0, 1, 2)_1, (0, 1, 2)_2, .., (0, 1, 2)_6)$-model. By combining the 0-transitions and counting only the number of predecessors on each dimension this should be called $(5, 5, 5)$-model. The $(5, 5, 5)$-model is parametrized by 125 possible transitions for each state.

In a Hidden Markov Model (HMM), the states of the Markov chain are not directly observable. But the output of the states, also called the emission of the states, is visible. The visible emission x is coupled to the hidden state s through the probability distribution $p(x|s)$. Therefore, by observing a sequence of measurements and relating them to the emission probabilities, the HMM can be used for assigning probabilities to different hypothesized hidden sequences which on their part are lists of locations.

This emission probability $p(x|s)$ represents the state-conditional probability in the Bayesian approach and has to be learned from the environment. In the context of localization, the state s represents a position in space. Therefore, it is understood as a model for the probability to receive a special signal constellation at a given position. Emission probabilities are often modelled as multivariate Gaussians or more complex mixture densities. The chosen model properties for the emissions in this thesis are discussed in the later section 2.5.4, where the simplification of the mentioned Gaussians is of concern.

2.5.1 Decision Rule

Finding the best matching sequence of positions s_1^T for a given sequence of measurements x_1^T is defined under the Bayesian approach by finding the maximum joint probability:

$$s_1^T = \underset{s_1^T}{\operatorname{argmax}}\, p(x_1^T, s_1^T)$$

applying the product rule leads to:

$$s_1^T = \underset{s_1^T}{\operatorname{argmax}} \prod_{t=1}^{T} p(x_t, s_t | x_1^{t-1}, s_1^{t-1})$$

In the HMM formalism, there is no condition on the previous emission vectors. They can be dropped:

$$s_1^T = \underset{s_1^T}{\operatorname{argmax}} \prod_{t=1}^{T} p(x_t, s_t | s_1^{t-1})$$

and there is only a dependency on the previous state (first-order Markov):

$$\begin{aligned} s_1^T &= \underset{s_1^T}{\operatorname{argmax}} \prod_{t=1}^{T} p(x_t, s_t | s_{t-1}) \\ &= \underset{s_1^T}{\operatorname{argmax}} \prod_{t=1}^{T} p(s_t | s_{t-1}) \cdot p(x_t | s_{t-1}, s_t) \\ &= \underset{s_1^T}{\operatorname{argmax}} \prod_{t=1}^{T} p(s_t | s_{t-1}) \cdot p(x_t | s_t) \end{aligned}$$

Therefore, calculating the probability of a sequence s_1^T is reduced to iteratively combining the transition probability of a jump and the emission probability for a measured signal vector x_t at the jump destination. This is a very cheap operation, so $p(x_1^T, s_1^T)$ can be determined in fractions of microseconds.

With this representation, a solution for the maximization can already be found with a brute force approach. The product has to be evaluated for all possible state sequences s_1^T. But with N possible transitions from one state s' into another s, the computational complexity is given by N^T solutions. This makes a brute force search intractable.

2.5.2 Viterbi Algorithm

The Viterbi Algorithm is an efficient dynamic programming algorithm for finding the most probable state sequence s_1^T for an observed vector sequence x_1^T. The Algorithm exploits the memorylessnes of the model that is induced by the Markov property. The Algorithm is defined as a set of recursive equations starting with:

$$Q(t, s) := \max_{s_1^t : s_t = s} \prod_{\tau=1}^{t} p(x_\tau, s_\tau | s_{\tau-1})$$

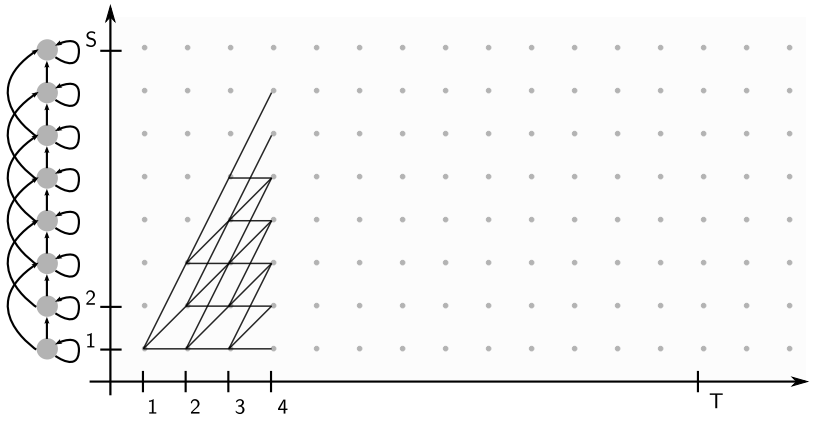

Figure 2.8 The paths through the lattice represent the different possible state sequences s_1^T under the $(0, 1, 2)$-model assumptions. In the visualized model, a sequence has always to start in s_1. In the localization context, where states represent positions, the sequence can start anywhere.

the decomposition of the predecessor states at $t-1$ with:

$$[... \to (s,t)] = [... \to (s',t-1)]\,[(s',t-1) \to (s,t)]$$

leads to:

$$Q(t,s) := \max_{s'} [p(x_t, s|s') \cdot \underbrace{\max_{s_1^{t-1}:s_{t-1}=s'} \prod_{\tau=1}^{t-1} p(x_\tau, s_\tau|s_{\tau-1})}_{Q(t-1,s')}]$$

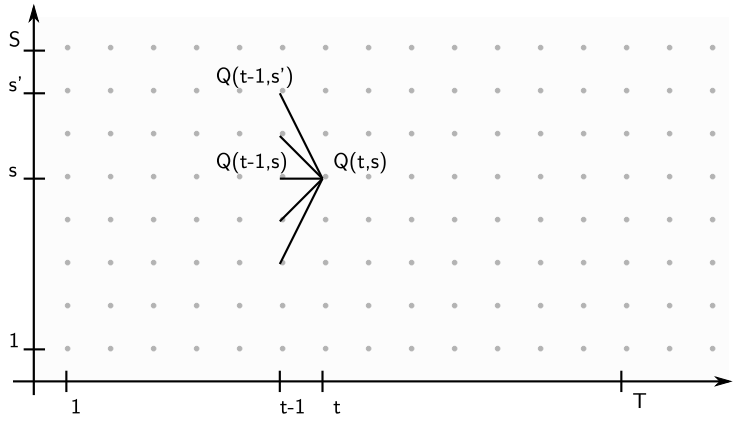

Figure 2.9 The recursion equation $Q(t,s)$ is computed by deciding for the predecessor state s' with maximum $Q(t-1,s')$.

and can therefore be simplified to:

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

Each local decision s' is stored in a backpointer array $B(t,s)$ and is retrievable as follows:

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

The computational complexity for evaluating Q at all time frames T is $T \cdot S \cdot N$ if S is the number of states and N is the number of transitions. The memory requirements for the Viterbi Algorithm are determined by the size of the backpointer array which is given by $T \cdot S$.

The best matching sequence with the maximum joint probability $p(x_1^T, s_1^T)$ is found by evaluation of Q at the final time frame T. The last position s_T of that sequence is decided by:

$$s_T = \underset{s}{\operatorname{argmax}}\, Q(T, s)$$

Starting with s_T, the full sequence s_1^T is reconstructed by recollecting the stored decisions from the backpointer array. This can be formalized by:

$$S(T) = s_T$$

$$S(t) = B(t, S(t+1))$$

so finally:

$$s_1^T = \underset{s_1^T}{\operatorname{argmax}}\, p(x_1^T, s_1^T) = [S(1), S(2), .., S(T)]$$

Assuming large models, as the presented UMIC model, with $S = 2 \cdot 10^6$ possible states and $T = 100$, the backpointer array can become large. By using a 4-byte int32 state space, this leads to 400MB memory usage. Although this can further be reduced by storing only the compact transition indexes $i \in 1..125$ for the $(5, 5, 5)$-model. With only 125 distinct values, a 1-byte int8 datatype is enough, which reduces the memory usage down to 100MB (see 5.2.2 for more details).

2.5.3 Higher Order Models

Incorporating more history in the transition probabilities leads to HMMs of higher order. The presented first order HMM uses only the present state for deciding which future state is the most probable. A second order HMM uses the first state of the history as well. Thus, the next state s is now dependent on two predecessor states instead of one (see figure 2.10). The transition probability is now given by $p(s|s', s'')$. The number of states for a second order HMM grows by the factor of N transitions. For each state, N new states are needed that have a *configured history* of the corresponding transition. The computational complexity is therefore $T \cdot S \cdot N^2$. The memory requirements due to the backpointer array grow to $T \cdot S \cdot N$.

Although the increase in complexity is quite significant, only with higher order models it is possible to model the probability of movements like turning left. It would be plausible to increase the probability of transitions that change directions in crossways and lower them in pathways. So from a computational standpoint, the enhancement seems feasible, but higher order models have not been fully implemented in the thesis.

2.5.4 Logspace

The recursion equation Q of the Viterbi Algorithm multiplies probabilities. Probabilities are defined to be $0 <= p <= 1$. Therefore, the application of many such

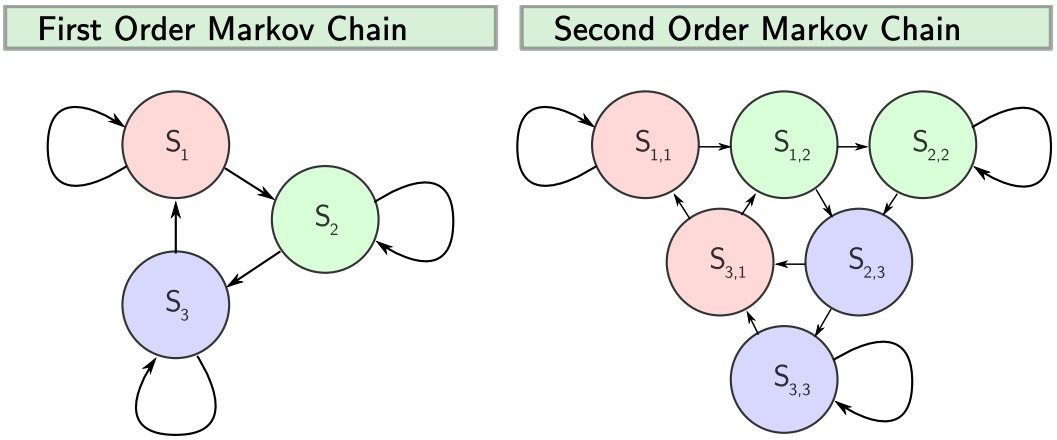

Figure 2.10 The states of a second order HMM carry the history of one predecessor state (first index) and the current state (second index). Due to this contextualization the number of states is increased by the factor 2, the number of transitions. Equal colors represent equal current states.

multiplications leads to numerical underflows. The classical approach to retain numerical stability in the recursions is to transform the probabilities into the negative logspace. Doing so, leads to a reinterpretation of the probabilities to costs. Instead of searching for the state sequence with the maximum probability, the task is now transformed to a search for the minimum costs. It should be noted, that applying the logarithm to a function is an invariant operation with respect to the maximum. Therefore, the decision for the resulting sequence will not be influenced.

So starting with the dynamic programming recursion for the local decisions:

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

and applying the logarithm leads to:

$$Q(t,s) := \max_{s'} [log(p(x_t, s|s')) + Q(t-1, s')]$$

with further interpreting as minimum costs:

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

It can be seen, that the evaluation of $Q(t,s)$ is performed very efficient in logspace. The evaluation of the costs for a candidate state s' is a simple summing of the stored transition probability $-log(p(s|s'))$ and the calculation of the costs of the emission probability.

In the presented system, the emission probabilities are modelled as a multivariate Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with independent components. An input feature vector x for a time frame t is D-dimensional. Therefore, the vector represents a list

of RSSI readings from D Access Points. The RSSI readings are assumed to be stochastically independent. The analytical form of $p(x|s)$ is given by:

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

The next model assumption is introduced in the form of a constant pooled variance for all states and dimensions. This leads to a further simplification of the emission model:

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

transforming the equation with the negative logarithm:

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

insertion into the recursion equation and dropping the constants C_1, C_2 due to the minimization leads to:

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

Thus, in logspace the Gaussian modelled emission probability is simplified to a distance calculation between x_{td} and the stored μ_{sd} components.

2.5.5 Pruning

Pruning, also known as Beam Search, is a heuristic approach to make the Viterbi Algorithm more efficient. The basic idea is to skip the evaluation of $Q(t,s)$ that are determined to be unlikely with respect to the global optimum. Therefore, finding the global optimum, the most probable sequence s_1^T, is not guaranteed any more. Loosing the global optimum will be more probable if the pruning is too aggressive and the number of evaluated states becomes too low.

There are two pruning strategies. The first strategy discards unlikely hypotheses if their probability drops below a threshold that is determined by the probability $Q_{top}(t)$ of the top hypothesis:

$$Q_{top}(t) = \max_{s} Q(t,s)$$

by introducing a factor f, the state hypothesis (s,t) is pruned if:

$$Q(t,s) < f \cdot Q_{top}(t)$$

With this strategy, the number of unpruned states is variable for each time frame t.

The other strategy induces a hard limit N on the number of unpruned states. During the evaluation of $Q(t,s)$ for a given time frame, the top-N states are collected for ensuring their further processing in the next $t+1$ time frame. Due to the simpler memory management, this strategy has been chosen in this thesis.

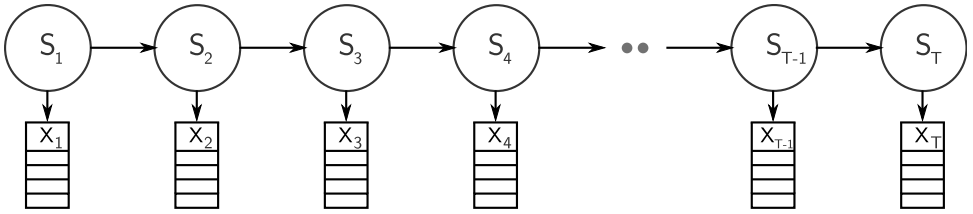

Figure 2.11 The general state space model is the base for the discrete or continuous representation of the sequential nature for a stochastic process. The progress of the stochastic process, a hidden state sequence s_1^T, can only be observed by measuring a sequence of noisy feature vectors x_1^T.

2.5.6 Training

For learning the parameters of the transition and emission probabilities of a HMM, a corpus of state-annotated measurement sequences is needed. If such a corpus is available, inference of the parameters of the stochastic models can be performed with the Baum-Welch algorithm. This algorithm represents the adoption of the basic assumptions of the estimate-maximize algorithm to sequential data.

But during the development of the presented system, no such training corpus for the Tracking problem was available. Therefore, the complex intrinsics of the inference algorithm should not be presented here (see [4] for details).

2.6 Continuous Models

Continuous models are the generalization of models with a discrete state space, like the HMM, to a continuous state space. Therefore, the once discrete PDF for $p(s|s')$ is now modelled as a continuous PDF, often as a Gaussian. But that is the only conceptual difference to the HMM model, as both are based on the state space model of figure 2.11. The memorylessnes, analogue to the Markov property, and the concept of observable emission probabilities are preserved. So the same factorization for the joint probability $p(s_1^T, x_1^T)$ under the first order Markov assumptions can be derived:

$$p(s_1^T, x_1^T) = \prod_{t=1}^{T} p(s_t|s_{t-1}) \cdot p(x_t|s_t)$$

with the difference of a now continuously distributed $p(s_t|s_{t-1})$.

2.6.1 Linear Dynamic System

A Linear Dynamic System (LDS), also known under the name Kalman Filter, is such a continuous state space model that is briefly motivated before continuing with more flexible concept of the Particle Filter.

The motivation for this approach arises from the following practical problem. An unknown quantity s should be measured by a noisy sensor. The measured observation x represents the underlying s distorted by zero-mean Gaussian. If a sequence

of measurements is available in the form of x_1^T, the hidden value of s can be easily estimated by averaging of the measurements.

The problem will again become more complex if the quantity s is allowed to change over time. An obvious approach would be to use some form of historical average of the last measurements to estimate the current true value of s. But how far should that evaluated window of measurements reach back? If the value for s is changing slowly over time, a large window capturing a lot of valuable information is appropriate. But if the value for s is changing fast, this leads to averaging over high variety information and therefore introduce a new form of error. Here, a short moving average window is better. This approach can be made more complicated by introducing a weighed average of the history, probably giving more recent ones more weight. This reasoning relates to the need for state-conditional probabilities $p(x|s)$, now the transitions will be investigated.

The other source of information, the transition probability $p(s|s')$ is modelled by a non-stationary multivariate linear-Gaussian distribution:

$$p(s|s') = \mathcal{N}(s|As', B)$$

where A is the linear transformation representing the movement vector and B the initial mean and variance. The requirement of a linearly changing state sequence is given by the need for integration instead of summing during finding the marginals of $p(x_1^T|s_1^T)$. A similar restriction is enforced on the emission probabilities. Only simple PDFs like Gaussians are allowed for $p(x|s)$. Mixtures densities lead also to problems during integration due to their contained summations.

These limitations were addressed by the introduction variants of that approach. For example, the extended Kalman filter [14] or the also preseneted Particle Filter, drops the restriction of linearity in the state changes.

For a more detailed introduction of the LDS technique, it should be referred to [21] and [4].

2.6.2 Particle Filter

One of the more recent approaches to the tracking problem, that was also evaluated in this thesis, is given by the Particle Filter (PF) [7]. The model is a response to the problem that estimating the non-linear movement of a person in an indoor localization scenario with the linear Kalman Filter is prone to fail. The concepts of the PF are based on the idea to sample the posterior distribution $p(s_t|x_t)$ directly for a given time frame t. In the HMM formalism, this posterior distribution is approximated by the recursion equation $Q(s,t)$. And due to the enumerability of the discrete states, the Viterbi Algorithm is able to compute $p(s_t|x_t)$ and can thus decide locally for the state s_t with maximum probability. The goal of this procedure can also be described as a marking of the most probable states or locations given an observed measurement sequence.

The same marking of probable states in a continuous space is conducted if $p(s_t|x_t)$ is sampled under the conditions of the observations. By generating enough samples, the best state hypothesis will get the most probability mass. This is in a way an

analogue to the remaining unpruned states of the pruning HMM. The unpruned states at a time frame t determine the new set of unpruned states at $t+1$ by evaluation of a new x_t. In the Particle Filter, the set of samples at a time frame t is equivalently used to generate the new set of samples for $t+1$ with respect to the new observations of x_t.

The sampling process is defined recursively[4]. Let $\{s_t^{(l)}\}$ be set of samples with cardinality L for a timeframe t and $\{s_0^{(l)}\}$ an initial set of samples that are drawn from a $p(s_0)$ distribution. 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 therefore computed from the emission probabilities $p(x_t|s_t)$ that are obtained from the radio propagation models. The weights satisfy the normalization constraint $\sum_l w_t^{(l)} = 1$ and are in the range $0 \leq w_t^{(l)} \leq 1$. They can be understood as a way to explain the importance of each sample under the current observation x_t. The posterior $p(s_t|x_t)$ 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 of t and the transition probabilities $p(s|s')$ to the following mixture:

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

By drawing new samples $\{w_{t+1}^{(l)}\}$ from this distribution, the recursive step is completed. Therefore, the algorithm can be interpreted as to switch between two modes. At first, the weight of the samples under the observation x is determined by using the stored $p(x|s)$. The weight defines the number of new samples that are spawned from this sample. In the context of this thesis: If the sample represents a very probable location for the RSSI readings, the sample location is used as a major source for the next set of samples.
The following set of samples is then generated by spawning the appropriate number of samples at each source into the most probable direction/distance given by the transition probability $p(s|s')$.

2.7 Least Mean Squared Error

Another localization method, that has been prominently used due to it's simplistic principles, is given by the so called Least Mean Squared Error (LMSE). Contrary to the HMM and the PF based approaches, the method ignores the sequential nature of the measurements and is therefore not based on the state space model of figure 2.11. The basic idea is given by comparing the RSSI reading of the D-dimensional measurement vector x with all location s annotated measurements y_s of a database representing the prior knowledge of the distribution. The LMSE uses the euclidian

[4] A more detailed presentation of the PF can be found in [4].

distance for the comparison and decides for the location with the least distance. The LMSE decision rule $r_{lmse} : x \rightarrow s$ is thus defined as:

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

The method is derived from the Bayes Decision Rule that has been introduced in section 2.4:

$$r_{bayes}(x) = \underset{s}{\operatorname{argmax}} \left[p(x|s)p(s) \right]$$

All locations are equally probable, and the calculation is done in logspace:

$$r_{bayes}(x) = \underset{s}{\operatorname{argmax}} \left[log(p(x|s)) \right]$$

The location-conditional $p(x|s)$ is modelled as a Gaussian with independent components: $p(x|s) \sim \mathcal{N}(\mu, \sigma)$. Furthermore, by assuming a constant pooled variance over all locations and removing the constant coefficients of the Gaussian[5], this leads to:

$$\begin{aligned} r_{bayes}(x) &= \underset{s}{\operatorname{argmax}} \left[- \sum_{d=1}^{D} (x_d - \mu_{sd})^2 \right] \\ &= \underset{s}{\operatorname{argmin}} \sum_{d=1}^{D} (x_d - \mu_{sd})^2 \end{aligned}$$

Therefore, under the given assumptions for $p(x|s)$ and if $y_s = \mu_s$ the following equality holds:

$$r_{lmse}(x) = r_{bayes}(x)$$

2.8 Summary

A thorough introduction into the general principles that are employed to approach the localization problem and the radio propagation modelling was given. The Bayesian approach to the pattern recognition problem was introduced and three algorithms, the HMM, the PF and the LMSE technique were derived. For the topic of radio propagation, the basic background information needed for understanding the primary physical effects that influence the RSSI signal distribution were presented. Building on that foundations, a brief overview over the mechanics of employed PHOTON system was given.

[5] A similar transformation into logspace has been done with the HMM emission probabilities $p(x|s)$ in section 2.5.4. A pooled variance, has been used there as well.

The Tracking problem, the search for the best matching sequence s_1^T for a given x_1^T, has been shown to be addressable with a number of different algorithms. The first one shown, was derived from the concepts of Hidden Markov Models. The model assumptions of a HMM, especially the memorylessnes due to the Markov property, make use of efficient dynamic programming algorithms for finding s_1^T possible. The computations in the Viterbi Algorithm were further simplified by moving the probabilities into logspace and assuming a constant pooled variance on the emission probabilities. The final computations contain only summations, memory lookups for stored probabilities and a distance calculation between x_t and the stored RSSI values of the radio propagation model. Since the computations for $Q(s,t)$ are simple, the implemented system can process around $5 \cdot 10^6\, Q(s,t)$ per second on current hardware. The proposed Pruning technique leads to a further speed-up of around one order of magnitude.

The other two presented algorithms are the LMSE and the Particle Filter. The latter is similar to the HMM as it is also derived from the state space model of figure 2.11. Whereas, the LMSE approach ignores the additional knowledge given by the history of the RSSI readings, the HMM/PF respect the sequential nature of the problem by modelling transition probabilities. As becomes observable during the later evaluation of the three algorithms, that incorporating this knowledge about the stochastic process leads to significantly reduced localization error rates.

The Positioning problem will be understood as a special case of the Tracking problem. The same algorithmic designs are used for finding solutions. Although in the worst case, a positioning attempt contains only data from one time frame without exploitable history. In this special case the LMSE approach does represent the most efficient solution. But it can be assumed, that in the presented use-case of Wi-Fi based positioning, the signal stream is easily adaptable to employ sequential data.

3

Related Work

This chapter gives an overview over related research that has been conducted on either modelling the radio propagation of AP transmitters in section 3.1 and over localization algorithms exploiting this information in section 3.2. Since the larger focus of this thesis was placed on the localization algorithms, especially the HMM and PF approach, the latter section is formed more broadly.

An historically important project, especially for RSSI based indoor localization systems, is given by the RADAR [2] framework. This framework combines the basic methodology, the appliance of a radio propagation model (the WAF model) to a localization algorithm (specifically the LMSE), that was also used in this thesis and other related works.

3.1 Radio Propagation

The two major approaches for the radio propagation model are either manually building a database of RSSI readings or using an analytical model for simulating the radio propagation. Historically, the former approach has been used more prominently as it gives a simple procedure to build a map of RSSI values for an indoor scene. The latter, the modelling by reasoning about the physical nature of the radio propagation, is a response to the expensive nature of the labour-intensive manual collection and leads to ray tracing based models of different complexities. A selection of localization systems that have employed analytical models based on 2D or 3D-raytracers are presented here. The following aspects of the systems are highlighted:

1. How detailed is the simulation of the physical effects at the intersection points between rays and scene geometry?
2. Is the transmitter of the signal a simple isotropic model or more complex?
3. What is the source for the 2D/3D-scene geometry?

4. Are the physical properties of the geometry homogeneously modelled or does the system distinguish between different materials?

5. How are these material parameters obtained?

6. If the material parameters were trained from measurements, what was the strategy? What was the optimization target?

7. Which error rates have been reached by comparing the simulation results to real world measurements? How complex were the testbeds?

One of the simplest analytical models for the radio propagation problem has been given by the RADAR system with the so called Wall Attenuation Factor model (WAF). The path-loss of the signal is described as a function of the number of intersecting walls on the straight line between localization hypothesis and the transmitter of the signal. The Wall Attenuation Factor influences the rate of power decay at each intersection and has to be found manually. Different factors for different wall materials are possible although not used. The reduction of the signal strength is empirically determined to be $3.1dBm$ for a wall intersection in the analysed scenario. This model can be understood as a very reduced ray-tracing approach. The line of sight is the only simulated ray and the attenuation factor corresponds to the transmission parameter of the employed PHOTON raytracer. Although the evaluation of RADAR has lead to the conclusion that the fingerprinting based approach reaches better localization accuracy than an also compared analytical model, this has not stopped interest in this area of research.

3.1.1 2D-Raytracer Models

A more complicated model is given by 2D-raytracers. These models are enhanced by adding the simulation of reflection effects at the ray intersection points. 2D-raytracers were investigated in [12] and [13]. The system, presented in [12] uses heterogeneous reflection and transmission coefficients for different materials, similar to the PHOTON system. In contrast, the ARIADNE system [13] does not distinguish between different materials.

In the ARIADNE system a rasterized 2D-floor plan is automatically converted to a 2D-model of the environment. They have employed a 2D-raytracer that simulates the basic physical effects like transmission and reflection. Refraction or scattering have not been simulated. The model ignores rays with a power below a fixed threshold. Similar to the PHOTON raytracer, the individual power values of overlapping rays are combined by simple summing without anticipating interference effects. The free parameters of the raytracer model are given by the antenna gain of a single AP and a transmission and reflection coefficient. Hence, only one material type is considered. An optimization of these free parameters has been done by running a simulated annealing optimizer against the RSSI value of only one measured location. As described in 2.1.2, the presented localization framework has shown better convergence with a genetic algorithm optimizer. But modelling different material parameters and separate antenna gains for different AP classes leads to a higher dimensional search-space and has therefore probably other convergence properties.

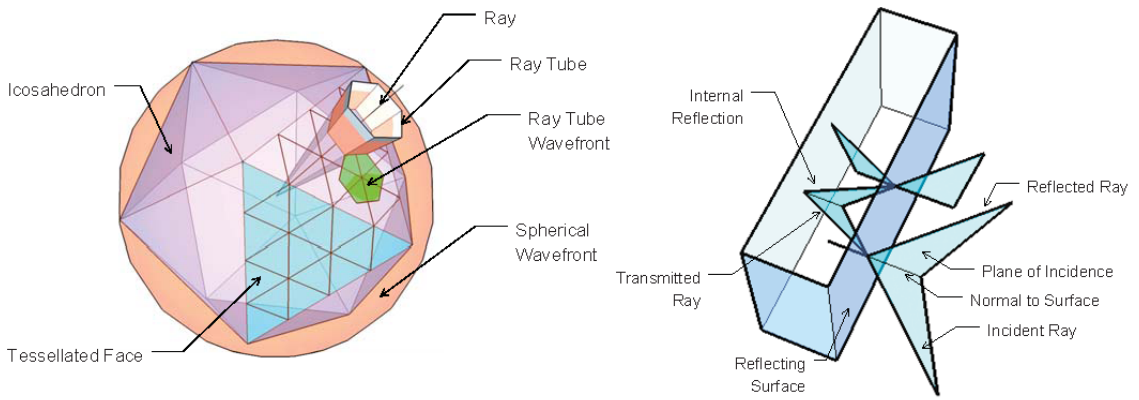

Figure 3.1 Schematics of the AROMA 3D-Raytracer. One the left side: ray-launching for variable antenna patterns and on the right side: material intersection of rays.

A RPE of around $3dBm$ and a RPE-RMSE of $3.5dBm$ has been evaluated[1] by comparing the simulation results of the model against 30 measurements at validation locations. These values are comparable to the results of the PHOTON model. But for using only one initial measurement during training of the free parameters, this is a very good result.

3.1.2 3D-Raytracer Models

One of the earliest works on raytracer generated 3D radio propagation Models can be found in [26]. The described technique was employed more recently in [8]. The presented raytracer simulates the physical effects of reflection and refraction. Different antenna patterns are employed. The 3D-scene is imported from CAD output files or synthesized by interpreting a 2D raster image containing a floor plan. Walls or other anticipated solid structure from the 2D floor plan are extruded into the Z-axis with equal height. Different materials are assigned to the different parts of the 3D-Scene. Instead of applying some training procedure, the material parameters were taken from literature like [30] and [20]. This is an unique approach compared to the other presented systems.

The evaluation of the system has been conducted on a 4-room scenario of around $80m^2$. 3 APs were placed and 3 averaged measurements at 42 locations are afterwards used for validation. A RMS-RPE of $2.28dBm$ is reported. The RMS-RPE degrades to $2.83dBm$, if the raytracer uses only the 2D mode. Comparing the performance over different sources of material parameters, the parameters in [30] have led to the best result.

A sophisticated raytracer driven radio propagation Model is given by the AROMA framework [9]. The implemented 3D-raytracer is able to simulate the physical effects of reflection, refraction and diffraction. The diffraction model is based on

[1]The reported value of $0.65dBm$ MSE has been corrected to $3.5dBm$ to be comparable to the other systems. Restoring the correct values was possible since a known number of 30 locations were used.

the Uniform Theory of Diffraction (UTD). AROMA is the only presented system that models the human body shadow. Detailed antenna modelling is also possible. The different properties of the antenna model that can be configured are given by transmitting power, frequency, maximum gain and radiation pattern. Figure 3.1 illustrates these model assumptions. Configuration values for isotropic and half-wave dipole antenna patterns are predefined. The input for the scene geometry can be given either in Blender or Google SketchUp format. Different material parameters are assigned to objects of the scene. It is unclear how the material parameters are derived, but the system exposes a preconfigured database for commonly used building components. The simulation of the human body shadow relies on the UTD model with the basic assumption that a human body can be represented by the shape of a circular cylinder with radius $0.15m$ and height $2m$.

As testbed for the evaluation, a $80m^2$ apartment room was chosen. 2 APs were placed and measurements at 11 locations were collected. At each location, 60 measurement samples were taken and subsequently averaged. A RPE of $3.2dBm$ and RMS-RPE of $4dBm$ is reported on this environment.

Another advanced 3D-Raytracer model with a 2D fallback mode is presented in thesis of Martin Klepal [15]. The engine simulates the optical effects of reflection, refraction and diffraction. Non-isotropic radiation patterns for the antenna model are also supported. The scene geometry is represented by a voxel space with a voxel edge size of $30cm$ for the $2GHz$ wave frequency. Multiple classes of materials are supported by assigning them to the corresponding voxels. As in the presented localization framework (i4lf), the material parameters are trained by optimization with Genetic Algorithms. The optimization criterion, also called the fitness value, is given by a minimization of a variance value which is also used during evaluation. The author reports that the Conjugate Gradient and Hill Simplex optimization algorithms have shown worse convergence behaviour with respect to the reached optimum. The impact of the number of rays on quality of the simulation is also analysed. The author concludes that using 0.3 - 2 Mio rays leads to a good AP coverage estimation without artefacts due to undersaturated areas. This relates nicely to the experience with the PHOTON raytracer where 0.3 Mio rays lead to usable and 2 Mio to excellent results with respect to the amount of artefacts in undersaturated areas.

An extensive performance evaluation was conducted on a single floor of 4 different office style buildings. The floor area of all sites combined is around $5000m^2$. The four different materials, that are used in all scenes, are given by: light wall, heavy wall, windows, floor. Special care has been taken by collecting the measurements. A wooden stick, that holds the device around $1m$ in front of the operator, was used to minimize the body shadow effect. The number of measurement locations and the corresponding number of received values that were probably averaged are unfortunately not disclosed. The RMS-RPE is reported to be between 3.5 and $4.5dBm$ over the four scenes[2] and corresponds to the PHOTON results as well.

[2]On page 84 of the thesis the author reports a computed mean and variance over the measurement deltas *without* applying the abs() function. This variance is a lower bound for the RMS-RPE. Therefore, the RMS-RPE is assumed to be around $0.5dBm$ higher.

3.2 Positioning and Tracking

The following sections will present literature for the major algorithmic approaches to the localization problem under the restriction of AP generated RSSI readings. An exception has been made for the related work about Particle Filters in 3.2.2. Their indoor localization approaches seems to derive of use-cases from the field of robot navigation and have therefore been historically adapted to other sensor types. The presented radio propagation dependent localization systems use the RSSI information in an *active* way. This means, a mobile device is actively sensing for the information. Recent research has also been conducted on the *passive* approach that is given by measuring fluctuations in the radio propagation with stationary device. Such a method was proposed in [32] and has the advantage to drop the need for special equipment at the mobile localization target in the form of some Wi-Fi-receiver. With exception of the presented system in [13], described in section 3.2.3, all RSSI based systems use a single device for measurements. In [13], three devices are used synchronously.

3.2.1 Hidden Markov Models

In this section, three HMM based localization systems and their evaluation result are presented. The focus is placed on the following model properties and environment configurations that are used for the evaluation aspect:

1. What radio propagation model was employed? Are different materials parameters distinguished if it is an analytical one?
2. How are is the PDFs of the emission probabilities $p(x|s)$ modelled and by which means are they obtained from the radio propagation model?
3. How are the discrete PDFs of the transition probabilities $p(s|s')$ modelled?
4. What is the rasterization ratio of the environment that determines the area or volume that is covered by a state?
5. How many predecessor states are evaluated and finally stored for backtracking to the best state sequence s_1^T?
6. How was the radio propagation model configured for evaluation? How many material parameters were used. How was the model obtained if it is an empirical one?
7. What is the size of the scene, and how are the LOS/NLOS conditions. How many APs were used?
8. What was the evaluation corpus of tracked paths and what is the reported error.

The first HMM based pedestrian localization system, that is analysed here, is reported in [25]. For the radio propagation model an empirical model is used that

is constructed by collecting indoor and outdoor measurements. Additionally to the stored RSSI information, sensor readings of the acceleration and from a compass are also available to the system. The database with the collected measurements was interpolated to obtain the emission probabilities $p(x|s)$ at all states s representing 2D locations on a grid. The database contains mean and variance information that is used to model $p(x|s)$ as Gaussian distributed. The evaluation of this probability for a measurement x contains a smoothing step that is conducted by integrating over the $[-0.5dBm..0.5dBm]$ interval around the measured RSSI value. The transition probabilities $p(s|s')$ are modelled variably and depend on the sensor readings of the accelerometer and the compass by using dead reckoning. All states are possible predecessor states leading to a computational complexity of $T\ cdotS^2$ during the Viterbi Decoding. Finally it is a first order Markov model, as only the direct predecessors influence the decision process.

An evaluation of the performance of the HMM localizer on synthetic data was conducted. The synthetic RSSI data is generated by assuming a radio propagation model with simple log-normal fading and combining it with white Gaussian noise of $6dBm$. The scene is a $2500m^2$ square area with full LOS conditions. Furthermore, additional synthetic noisy accelerometer and compass information was added to the setup. The state space contains 2500 states with edge size $1mx1m$. 9 APs were evenly distributed. This environment leads to an offline LE of around $2m$ for a not exactly specified sequence of 10^5 measurements.

Additionally an evaluation on two real world scenes was conducted. The propagation model is obtained from previously collected measurements in both cases. Both cases have an unknown number of APs and an unknown rasterization for the state space. The measurements of the three mentioned sensors were taken with a HTC Hero Android smartphone. The first scene is a Christmas market with area of around $2500m^2$ at a time of the day where nearly no visitors lead to destructive body shadow. Due to the many small tourist shops at the market, there are high NLOS conditions. For this outdoor scene an offline averaged LE was evaluated on an unclear number of measured signals. The second scene is a combination of an indoor and outdoor area of around $1200m^2$. An offline LE of around $2m$ for the indoor, and $5m$ for the outdoor part is determined. The combined LE on all 2300 measurements is given by $4m$.

Another HMM based localization system is presented in [27]. An analytical radio propagation model was employed that is based on a multi material WAF model. The emission probabilities $p(x|s)$ are obtained from the WAF model that is enhanced by assuming that attenuation factors are modelled as Gaussians. The noise at the receiver side is also modelled as a Gaussian. Both PDFs are combined to form the final $p(x|s)$. As in [25], the computation of $p(x|s)$ for an instance of the RSSI vector x is done by integrating over the $[x - 0.5dBm..x + 0.5dBm]$ interval. The transition probabilities $p(s|s')$ are modelled by assuming two modes of operation. The first mode is the *movement* mode which is detected by evaluating the variance from the last 10 measurements. If the variance exceeds a preconfigured threshold, this mode is assumed to be given. The other mode can be evidently called *non-movement* mode. The mode configures the maximum walking speed that is used to determine the possible transition origins s' for the current state s. As $p(s|s')$ is modelled as a rasterized Gaussian, the maximum walking speed is projected onto the variance. All

states are possible predecessor states although most of them will have a $p(s|s') = 0$, depending on the mode controlled variance. Also, it is a HMM of first order.

The evaluation was conducted on an indoor office style scene with an area of $600m^2$. The WAF model is configured with 6 different material coefficients. 3 APs were placed to lead to NLOS conditions at the evaluation site. The 2D grid resolution of $0.5mx0.5m$ leads to a state space with cardinality 2400. An averaged LE of around $3m$ for this setup is reported[3].

The HMM based localization system reported in [16] also uses a variant of the WAF model, dubbed RSSI delay profile, for simulating the radio propagation. Similar to [27] the variance of the emission probabilities $p(x|s)$ is influenced by assumed noise on the attenuation factors. For the transition probabilities $p(s|s')$, different discrete two-dimensional PDFs are possible. All states are possible predecessor states. It is a first order Markov model.

Synthetic measurements were generated with the WAF model for an office like scene with an area of $1200m^2$. The noise level of the sythetic data, given by different variances, should represent natural conditions. 4 APs were evenly placed to form a square. p(s|s') is modelled as a conic shaped two-dimensional PDF with maximum probability at $p(s|s)$ that is rapidly decreasing with increasing distance. On this setup, a RMSE-LE of $5m$ is reported.

3.2.2 Particle Filters

Particle Filters (PF) are one of the most recent approaches to indoor localization. Historically, they originate from the field of localization for robotics. In this context, they have the typical sensor information of vehicles in the form of speed available. Two of the presented implementations can use the information of the step length and the step heading and/or RSSI readings for pedestrian navigation. The techniques to process this additional information source in the PF approach seems to originate in the research of robot navigation.

Four implementations will be presented by focusing on the following properties:

1. What is the available information source? If its a radio propagation model, what is the type of it?
2. How is the environment modelled, is it a 2D or 3D model?
3. How is the weighting of the particles defined? How is the state-conditional, the emission probability $p(x|s)$ applied?
4. How are the transitions $p(s|s')$ modelled and how are they used during the sampling of new particles?
5. How is the problem of sample impoverishment handled?
6. On what scene was the evaluation conducted, how many particles were used and what error rates were reached?

[3]The value is derived from the CDF plot as only a median of $2m$ is reported and there seem to be a large number of outliers in range of $> 8m$

A PF for pedestrians, targeted to support rescue operations, that uses only footstep sensor reading is presented in [29]. The sensor reports a step length, fused from an accelerometer and a step heading that is derived from a compass. The environment is 2D based and constrained by information from 2D indoor/outdoor floor plans. The weighting of the particles is handled by a simple zero weight if the particle has crossed a wall and constant otherwise. The transition probabilities are derived from the step heading and step length information and are combined with white Gaussian noise. Therefore, a directional sampling of new particles, comparable to dead reckoning constraints, is employed. The PF was enhanced by an interesting approach to sample impoverishment problem. The basic idea is to track the history of the particles by storing links between the new and the source particles. Therefore, each generation of particles is stored as well. Now, if the degradation of the current generation is detected by a drop in the combined weight of all particles, the particles are traced back into the older sample generations and the source particles of the currently "bad" particles are replaced by new samples. From these "corrected" generations the sampling process is restarted.

The evaluation was conducted on an $2700m^2$ indoor/otdoor scene. A tablet PC, processing the information of XSens MTi motion sensor, was used as the mobile device. Map constraints from a detailed 2D-map were available. 2000 Particles are used. The averaged LE is given by $1.5m$ for the basic PF implementation and an even better $1.3m$ for the variant that escapes sample impoverishment with backtracking

Another PF, that uses the same footstep sensor as the major information source, is presented in [31]. The system does also employ RSSI measurements but, these are only used for the initialization of the particles. The particles are not uniformly displaced over the environment, they are constrained by a rough approximation of the probable localization area. The used radio propagation model for this step is a simple path-loss model without wall attenuation. The system has a notion of the 3D-space for multi-level building by connecting the individual 2D floor maps at junction points and enriching the 2D map polygons with height information. This 2.5D-map is used for rejecting trajectories that lead through impassable regions. The sample spawning is driven by transition probabilities that are modelled similar to [29]. The weighting of particles is also assumed to be zero if a wall is crossed. If it is a valid particle, the weight will be determined by height information for the 2.5D map and the height change that is additionally extracted from the footstep sensor. The transitions are modelled equally to [31] and derive their parameters from the step length/heading information.

The evaluation was conducted on a three floor office style building with a total area of $8725m^2$ that is comparable to the UMIC scene. A hip mounted PC processes the footstep data of a XSens Mtx IMU sensor. Over 6 walks with a duration of $16min$ each, a very accurate RMS-LE of less than $0.7m$ is reported.

In [28], three different nonlinear filters: Fourier density approximation, a gridbased filter and a PF are compared on an analytical and an empirical radio propagation model for 2D scenes. The former analytical model is a multi material WAF model and the latter a manual collected RSSI database. In the analytical case, the weighting of a particle is derived from the output of the WAF model with noise from a two-components Gaussian mixture with the two means of $-7.5dBm$ and $+7.5dBm$. In the empirical case, the stored RSSI are interpolated and used as means for the mix-

ture model. In both cases, the transition probabilities $p(s|s')$ are circular bended one dimensional Gaussians with mean 2.5 and variance 0.3 representing average movement speed of $2.5m$ per sampling step. The problem of sample impoverishment on the particles is not discussed.

An evaluation is conducted on synthetic data from the WAF model. The scene is an office style floor with an area of around $1200m^2$. The WAF model is configured with attenuation coefficients for three different materials. 6 APs are evenly distributed. The emission probabilities are used to generate measurements for one reference path with length $25m$ that visits three rooms in the scene. The reported offline LE is less than $0.8m$.

On the empirical radio propagation model an evaluation with real measurements was conducted. 14 APs were evenly distributed in the same scene. On the same path as above, the reported averaged offline LE is less than $1m$. In both experiments 500 particles were used.

The last investigated implementation of a PF which uses RSSI readings can be found in [5]. The propagation model is an empirical one and is described in [6]. The weighting is based on emission probabilities that are Gaussians with a variance of $5dBm$. The indoor environment is described in a 2D-map that is used to constrain the movements during the particle sampling process. The transition probabilities are further modified in crossway zones to reflect the higher probability to turn in direction. Sample impoverishment is detected by a drop of the combined weights of the particles and is compensated by feeding new uniformly distributed particles into the system.

The evaluation is conducted on a rectangular indoor office scene with an area of $1600m^2$. An averaged LE of $1.9m$ is reached but the number of APs and the length of the path were unfortunately not further defined.

3.2.3 Nearest Neighboor based Approaches

For completeness, an overview of localization systems that ignore the sequential nature of the tracking problem or that are simply only designed for the positioning problem, will be given. These systems use the Mean Squared Error (MSE) as the distance measure between the received RSSI vector and the stored RSSI values for a location. The location with the minimum MSE is selected as the most probable location. Therefore, the employed technique is labeled Least Mean Squared Error (LMSE) and was also derived in the background chapter 2.7.

A problem in the approach of LMSE is given by the result, that largely different locations can have similar MSEs. As a reaction to this problem, enhanced LMSE-based methods were developed. One of them is described in [18] as the so called closeness elimination scheme. Another in [17] where the locations at the top smallest MSEs are combined.

The system described in [8] evaluates a basic LMSE driven localizer that is based on a raytracer generated progation model. The raytracer is configured with different material coefficients that are taken directly from the literature. By using material parameters from [30] the performance of the system is given by a RMS-LE of $2.28m$

for the positioning problem. The chosen scene for the experiments was a $130m^2$ small 3-room floor with three APs. The number of positioning attempts that were evaluated is not documented.

Another framework, that is presented in [13], generalizes the LMSE selection criterion to a clustering based approach. Of the two known clustering strategies k-means clustering and hierarchical clustering, the former is reported to lead to better results. The unknown initialization variable of the k-means strategy is given by the number of target clusters. This number was determined empirically. Furthermore, the cluster history is exploited without formalizing the model to the state space model of Figure 2.11. The basic idea is given by rejecting clusters at the time frame t if the best clusters of $t-1$ are too far away. The reported averaged LE is given by $2.8m$. The contained tracking error is probably an online 2D-error with history constraints.

In [9] a simple localizer, based on the LMSE criterion, is used to test the quality the radio propagation model that is generated by the sophisticated 3D-Raytracer described in section 3.1.2 of this chapter. The evaluation of the localization algorithm is performed by estimating the positions of 20 RSSI vectors that are collected for each one of 11 predefined locations. No dependency between the 20 readings of a single location is assumed. A pretty good LE of $1.61m$ is reported, although the setting is only a smallish single-room scene of $80m^2$ with nearly no NLOS conditions.

3.3 Summary

The following observations were made during the evaluation of the presented literature. There is no defined set of rules for determining the quality of radio propagation models or for comparing the performance of localization algorithms. Various error measures are in use and sometimes the evaluation is conducted on unclear environment conditions. If needed, the presented errors have therefore been normalized to the measures of this thesis, described in 2.1.3 and 2.3.2, for easier comparison. It would be helpful, if the research community could agree on some common set of principles that should be mandatory for the evaluation of both topics. For example, it should be mandatory to report the number of used APs in an evaluation scene for RSSI based localization algorithms. That was not always the case.

4

Design

This chapter introduces the different concepts that were needed for the realization of the localization framework. The core components, their use cases and their individual relationships are presented. The various communication channels that connect these components are described and the motivations for the individual architectural decisions are explained.

4.1 General Overview

The localization framework is build around a central server instance that communicates with different consumers for various purposes over HTTP. The different consumers are composed of mobile devices, browsers, GPU-nodes and fat clients. The different purposes for these communications are localization, measuring, evaluating of results and distributing computational hard problems. The Server combines five major components for approaching the topics of this thesis. These components consists of:

1. The Optimizer, the component responsible for training the radio propagation model through searching for optimal material parameter solutions by comparing raytracer results with a given set of initial measurements.

2. The Simulator distributes raytracing jobs to the available GPU-nodes and collects the processing results. Either in the context of an Optimizer run or for manual inspection of the radio propagation results for the different APs and different material parameters in a debugging environment like the Fat Client.

3. The Localizer component represents the HMM and PF driven algorithms with the associated environment needed for positioning and tracking of mobile devices. Therefore, its also used by the Evaluator for evaluating offline recorded measurements annotated with tracked positions.

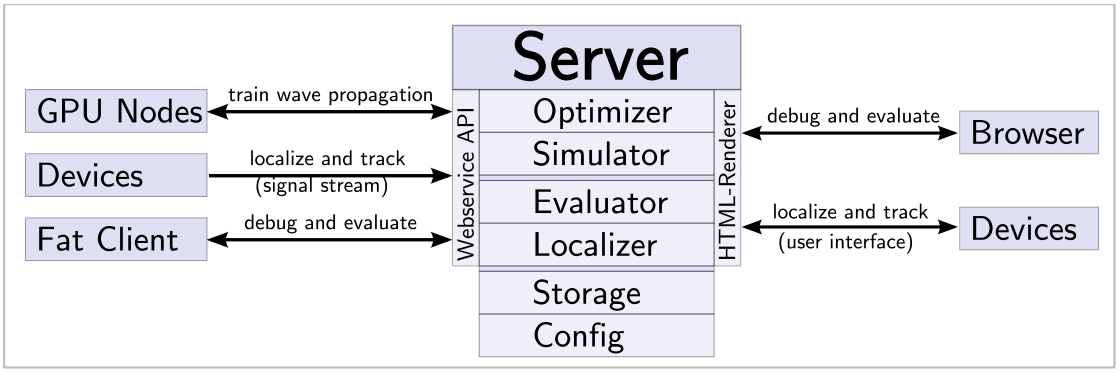

Figure 4.1 Highlevel overview over the localization framework showing the major components, their relationships and the communication channels. The employed communication protocol is always HTTP.

4. The Evaluator is a component that is needed for determining the different errors that arise during the localization process. These errors are aggregated over the collected corpus of paths and therefore used for evaluating the overall system performance under different conditions.

5. The last component can informally be labelled General Infrastructure, as it contains subcomponents suited for data-persistency or configurability, the HTTP-Server, rendering engines and other required application infrastructure.

The primary communication channels between these various components and the consumers are shown in Figure 4.1. Of available consumers there are:

1. The GPU-nodes for running a computational expensive radio propagation simulation with the PHOTON raytracer.

2. The devices, representing a class of Wi-Fi capable hardware like smartphones, tablets, laptops other forms of mobile computing devices.

3. A Fat Client for debugging and evaluating the overall system performance and the generated radio propagation models.

4. Different browsers, either on a desktop PC complementing the Fat Client for debugging and evaluation, or on a device for the interaction with the localization application.

4.2 Radio Propagation

The performance of the indoor localization system is strongly coupled with the accuracy of the estimation of the signal strength for each AP at a given location. Therefore, the first task is to obtain such a signal strength map by the means of a radio propagation model. Instead of using an empirical propagation model that requires extensive manual calibration via collecting measurements, the decision was,

to use the raytracer PHOTON for generating the SSMs. Although the raytracer enables dynamical simulation of the radio propagation for freely placeable APs, there are still some required prerequisites. Initially, a 3D model of the building is needed that should preferably distinguish between materials with different physical properties. For each material, the optical parameters can then be either looked-up or they need to be trained. In the context of this thesis, a method to train these parameters was designed. The main motivation for these efforts was given by the circumstances that although material parameters were found in [30], they were not directly applicable to the presented setup. It is also reasonable to assume that the properties of the used materials in the UMIC building differ from the ones used in other experiments. Furtermore, there exists no unified input format for different raytracers in general and radio propagation simulators in particular. Although probably this hurdle could be overcome by devising some adaptation scheme between different material parameter representations.

4.2.1 Model

The raytracer model consists of a Blender generated 3D-Scene with multiple meshes representing the walls, doors, windows and furniture of the building. Each mesh has a defined material. For each material, there is a reflection parameter α and a transparency parameter β. Additionally, a power parameter is defined that controls the initial strength of the emitted signal for each AP class. APs of the same model are grouped into such an AP class. With α and β as material parameters, the raytracer uses the following BRDF that is described in 2.1.1 for simulating the intersection between rays and materials. Refraction, diffraction and interference are not simulated by the basic BRDF and there is no change of the direction of the ray that passes through the material. For simplicity, the emission of the signal of an AP is assumed to be isotropic. It is possible to employ more complex antenna patters that are based on spherical harmonics. But that would lead to more free, and therefore trainable, parameters instead of the chosen representation by a single scalar power parameter for a class of APs.

The raytracer was successfully employed to simulate the radio propagation of GSM-stations in outdoor scenarios. So it is reasonable to assume, that indoor scenarios can probably be simulated successfully as well. The raytracer is GPU-accelerated and simulates a scene of $3000m^2$ over three Floors in a resolution of two megavoxels and with 10^6 Rays in around 30 seconds on a NVIDIA QUADRO 6000. Therefore, the raytracer produces SSMs with a voxel size of $(20cm, 20cm, 20cm)$. For each AP with a given position and power class such a signal strength map is generated. During evaluation, these maps have to be generated only once given appropriate material parameters for the scene and power/location parameters for the APs. Hence, it is very cheap to adapt the propagation model to a new AP configuration.

4.2.2 Parameter Estimation

For training the free parameters of the raytracer model, the implemented system uses measurements that are collected with Wi-Fi capable devices at predefined positions in the building. After this initialization phase, an evolutionary optimization

process is used to find the parameter combination that minimizes the aggregated error between the propagation simulation result and the collected measurements.

4.2.2.1 Initialization

The measurements of the signal strength for the individual APs are either collected by using the mobile devices that should be used as localization targets later or by some special hardware like the WISPY spectrum analyzer[1]. The measurements are generated by reading the device specific Wi-Fi-APIs and then transmitted over HTTP to the server instance. There, they are stored for each device separately to retain the possibility to adapt for possible device specific distortions.

Since the readings of all APs can be measured parallel at each location, the procedure is not time consuming. In the UMIC scenario measuring for 30 seconds at 60 locations has been shown sufficient for running the parameter optimization.

4.2.2.2 Optimization

The parameter optimization is driven by a genetic algorithm on the Optimizer component. A set of free parameters constitute an organism in the context of the genetic algorithm. The fitness of an organism is evaluated by running raytracer simulations for all APs over the geometry of the scene with the corresponding parameters of that organism. This can be a time consuming task since a representative raytracer simulation needs more than 30 seconds on a current GPU/CPU and convergence of the cost function is reached by evaluating at least 3000 organisms. In the analysed UMIC scenario with 22 APs, this leads to a serial processing duration of around 22 days. Through exploiting the parallelizability in the cost/fitness function and the parallel nature of the breeding phase, the duration of such an optimization run can be reduced nearly linearly, as long as $N_{gpus} << N_{pop} \times N_{aps}$, by employing GPU-nodes. These nodes are accessed through the Simulator component that will distribute them accordingly.

The nodes are designed as lightweight processes that are hard wired to run the PHOTON raytracer and execute some job specific code on the results before pushing them back to the Simulator component. The job specific code is transferred with the job package to the node. This allows to bring new types of jobs into the system by only updating the Server deployment. The security risk imposed by transferring the executed result inspection code over HTTP, should be minor as the nodes pull the jobs from a single preconfigured host. The design of the system can easily be extended to allow automatic spawning of nodes but no real efforts have been undertaken in that direction.

The motivation for choosing such an approach was given by the circumstances that during the duration of this thesis the RWTH had deployed a new GPU-Cluster consisting of around 50 NVIDIA QUADRO 6000 nodes. By using these resources a full optimizing run containing 30000 simulations with a duration of 30 sec each,

[1] Although measurements with the WISPY were taken with the goal to derive the free raytracer parameters, the evaluation of the results were inconclusive and have thus been removed from the scope of this thesis.

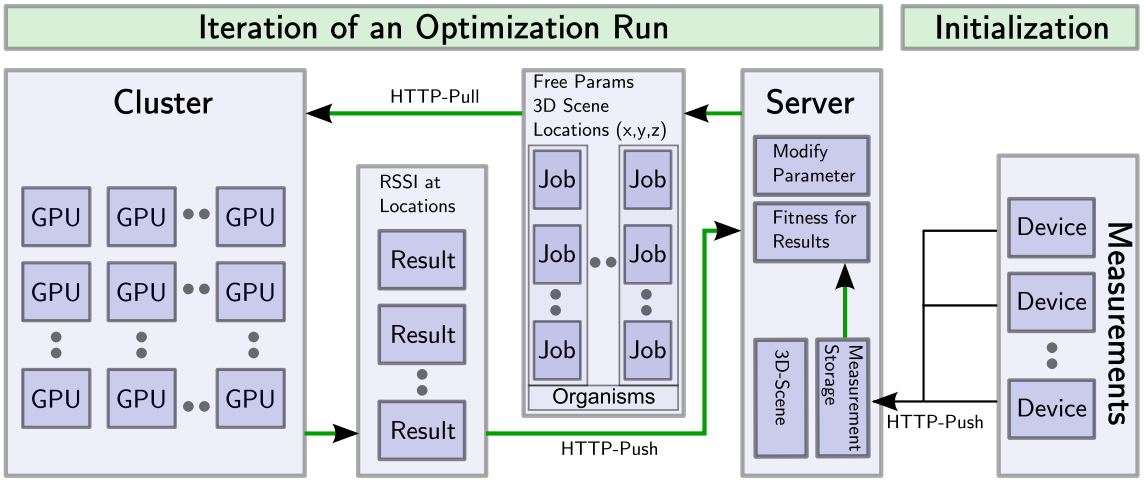

Figure 4.2 Schematics for decentralized parameter training on cluster. The main loop of an optimization iteration is emphasized with fat green lines and consists of interactions between the Optimizer, the Simulator and the GPU-nodes on the cluster. Attached is also the data flow for the initialization phase when measurements of different devices at different locations are collected in an indoor scenario.

completes overnight instead after multiple weeks. Without these resources, one would have to sacrifice either resolution at the raytracer level by decreasing the number of simulated rays or reduce the number of included APs. It can be expected, that the prepared infrastructure will also be useful during training of the localization models [2].

With the described mechanics, evaluating a population of organisms leads to spawning of $N_{pop} \times N_{aps}$ jobs that are pulled by scripted worker GPU-nodes, located in the cluster, over HTTP. A job is defined by a set of material parameters, a set of power parameters for each AP class, a 3D-model and a list of locations in the 3D-scene. The worker node executes the raytracer with the given input data and evaluates the resulting SSM at the defined locations. The list of RSSI values of each location is pushed back as a result for a given job-ID to the optimizer. If all jobs, the results of the individual APs, of an organism are available to the Optimizer, the fitness of the organism is calculated by using the RPE as a measure. The Optimizer tracks the organism with the best fitness, so that if a new optimum is reached, the system stores the parameter set for later use. Convergence is assumed, if either a predefined generation count is reached or the change of fitness between two successive optima is less than 1 percent of the absolute fitness value.

4.2.3 Device Specific Adaptation

Although the scale of the measured RSSI readings over the different devices can be interpreted as a dBm value which represents the dampening of the emitted signal, these readings are not normalized over device specific properties like deviations in the

[2] But as the computational requirements of the tracking algorithms can be handled by a modern multicore CPU and no real big training corpus of tracked measurements was available this has not been fully explored.

antenna gain. Thus, the possibility was introduced to adapt or normalize over these properties with some function that accepts raw RSSI values as input and results in normalized RSSI values for output. The configuration of this function can be learned during the optimization process or alternatively later in a postprocessing step with an optimization algorithm that compares device specific measurements to the radio propagation results.

Since such an adaptation is device specific, a prerequisite for the optimization is the availability of separate measurements with known locations for each device or at least for each different device class that share a similar hardware configuration. Instead of using manual collected and therefore costly measurements, it can also be possible to extract these location annotated measurements from the localization results of an accurately working system. Such a system needs a form of confidence measure for distinguishing between good and bad localization results and is exemplary for the method of reinforced learning.

If individual Device Specific Adaptation is not used due to the presented limitations, a simple global adaptation of the raytracer results is applied. The simple adaptation is a mapping of simulated RSSI values that are smaller than $-100dbm$ into the range $-100dbm$ to $-90dbm$ since all evaluated devices share $-100dbm$ as an upper limit for reported RSSI readings.

4.3 Positioning and Tracking

It has been explained in the background chapter that positioning and tracking can use the same set of algorithms, since the former is understood as a special case of the latter. Both work on a sequence of measurements that collapses to a single measurements at the start of a positioning or tracking attempt. Although it is possible to differentiate between the two cases if additionally sensorial input from the devices could be delivered, for example in the form of acceleration information[3], this was not evaluated in this thesis. Therefore, both use-cases employ the same component, referred to as Localizer, see context in figure 4.1, in an identical configuration for solving the associated localization problem.

The core of the localization system employs the chosen algorithmic backend that is given either in the form of a Viterbi Decoder for the HMM based approach or the sampling algorithms for the PF. The Viterbi Decoder processes prepared sequences of RSSI readings by executing the Viterbi Algorithm on a HMM. The emission and transition probabilities are parametrized with data managed by the Environment component. The same principles are used to drive and configure the PF algorithms.

The RSSI readings for all available APs are collected at each device by querying the local Wi-Fi-APIs in a configurable interval and pushing them to the Server by using the Webservice-API. The Server collects these RSSI readings as a sequence of measurements that are further normalized in a signal preprocessing step (see figure 4.4) before fed into the Viterbi Decoder for the HMM model or into the PF decoder.

[3]It would be reasonable to expect that such information could be used for manipulating the transition probabilities. In the simplest case that means distinguishing between moving and standing.

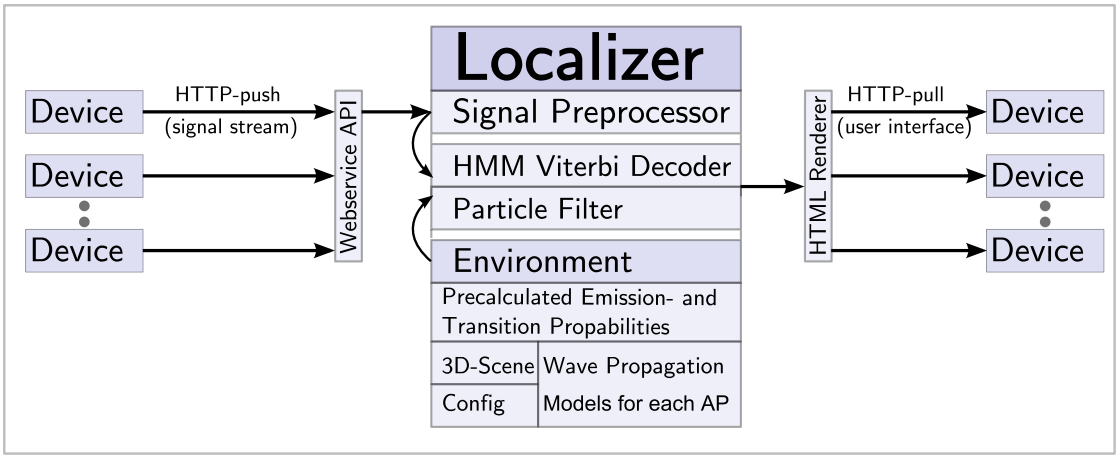

Figure 4.3 Overview over the different subcomponents of the Localizer and the interactions with the devices. The Localizer can either use a HMM, a PF or LMSE for the core algorithmics.

Collecting measurements and evaluating them has been done asynchronously during evaluation of the system. This means, that the measurement sequences were stored to form an evaluation corpus that was afterwards analysed. In the online tracking mode, the measurements are synchronously processed and the incrementally reported sequence of locations is temporarily stored. The devices will pull their corresponding localization result by accessing the Webservice-API and visualize them in a browser.

4.3.1 Hidden Markov Model

The localization problem under the HMM approach is given by finding the most likely sequence of hidden states in a HMM for the observed sequence of measurement. A hidden state of the HMM represents the unknown location of the device for a given time frame. The observed emission of such a state is given by a set of RSSI readings for that time frame.

There are a large number of possible location sequences. If the number of measurements is given by T and the number of possible locations or hidden states is given by S then there are S^T possible location sequences that are possible localization results. At the utilized raytracer resolution in the UMIC scene with a voxel size of $20cm$, the rasterization of the 3D-space leads to $2 \cdot 10^6$ Voxels that represent the maximum resolution of the hidden state space . So with $T = 10$, this leads to around 10^{63} possible localization results. The number of hidden states can be reduced by combining them into larger cubes leading to a state space with a lower resolution. This can be done by combining a number of adjacent voxels for each dimension to a hidden state with a larger edge size. But a brute force search for the most likely sequence of 10 hidden states combined of 3 voxels with an edge size of $60cm$ still needs to inspect 10^{48} candidate sequences. And that remains an intractable computational problem.

Therefore, the problem is reduced to a first-order Markov Model and the Viterbi algorithm is utilized for finding the most probable state sequence efficiently. By using a first-order Markov Model the Viterbi algorithm processes S states for each

time frame $t \in T$. At each state s_t all possible predecessor states are visited. It is reasonable to limit the number of possible predecessor states to the surrounding area of a given state. So by restricting the transitions between states to the neighbourhood of the next two states in both horizontal directions and the next state in the vertical direction this leads to 75 predecessor states and is called a $(5, 5, 3)$ transition model. These constraints limit the number of needed computations to the tractable polynomial complexity of $75 \cdot T \cdot S$. In the UMIC scene, around 60k states with edge size of $60cm$ are defined. Thus, a sequence of $T = 10$ measurements leads to $45 \cdot 10^6$ needed computations, that are processed very quickly, as the implemented decoder is able to compute $5 \cdot 10^6$ states per second on an Intel i7-QuadCore@3Ghz.

4.3.1.1 Parameter Estimation

The parameters of the HMM, that are evaluated by the Viterbi Decoder component, are given by the emission- and transition probabilities. These parameters are estimated during a training phase and are cached in the Environment component. The default HMM parameter training method is given by the Baum–Welch algorithm. But for such a training, a corpus of location annotated RSSI readings is needed. That corpus needs to contain training data for all possible states, representing locations in the 3D-space, so that the practicality of this approach can be excluded. Therefore, the parameters of both probabilities are acquired from other sources.

The combined SSMs over all APs is understood as a generative model for the emission probabilities thus these parameters can be obtained easily. But for the transition probabilities, no such convenient data source is at hand. The only available source of information, which can be exploited for modelling constraints on the movement between states in the 3D-space, is the 3D-model that is already used for driving the raytracer. At least the most critical information for modelling the transitions, the restriction which states are impassable due to blocking material, are retrieved from the building structure. Incorporating such knowledge is used to reject path hypothesis that move through walls, floors or other forbidden zones.

4.3.1.2 Emission Probabilities

The model parameters for the emission probabilities $p(s|x)$ are obtained from the stored radio propagation Models. In the UMIC scenario a hidden state is configured to represent the neighbourhood of 3 adjacent voxels for each dimension. The emission probability is modelled as a multivariate Gaussian with a constant pooled covariance matrix $\Sigma = I$. The N_{aps}-dimensional mean vector for $p(s|x)$ is obtained by calculating the arithmetic mean of the RSSI values from the 27-voxel-neighbourhood for each AP dimension. Although the variance of these voxel-neighbourhoods can be calculated this does not represent any conclusive information source. The real variance of the received RSSI values is driven by multiple effects like multipath propagation, body shadowing and other probably location dependant noise. None of these effects are captured in the radio propagation model[4], so the model drops the variance altogether. By transforming the Gaussians with $\Sigma = I$ into logspace,

[4]Simulating the effect of multipath radio propagation on the variance could be enabled by the possibility to dissect the raytracer results into the different recursion depths that are triggered for

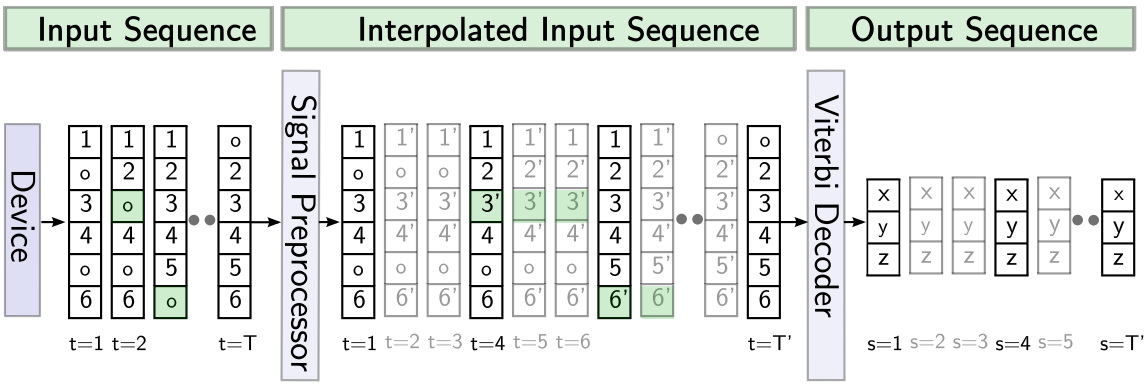

Figure 4.4 Signal preprocessing recovers missing signal components (marked as green circles) by using information from adjacent signals and injects the needed additional interpolated signal vectors (grey) to allow a maximum walking speed of $3ms^{-1}$ under the constraints of the $(5, 5, 3)$ model.

as described in 2.5.4, the task of the Viterbi Decoder to evaluate $p(s|x)$ becomes a simple euclidean distance[5] calculation between the stored mean vector and an input RSSI vector.

Special attention is given to the problem of missing signals in the input vector. Although signal preprocessing mitigates this problem by interpolation of missing AP components in the sequence of measurement vectors (see figure 4.4), there can still be missing AP components at the head or the tail of the sequence. If missing signals are simply adjusted to a value of $-100dbm$ this leads to undesirable high deltas during the distance calculation for the neighbouring states of the correct location. There, in the case of a missing signal, the corresponding AP values are larger than $-100dbm$. With larger simulated RSSI values for such an AP at these states, the effect becomes stronger. The states near the correct location get an artificially elevated distance and are in a sense "pushed away". Empirical investigations have lead to the best results by fully skipping the component, although assigning a distance of zero seems to be a form of inappropriate reward for this case.

4.3.1.3 Transition Probabilities

The transition probabilities $p(s|s')$ are the key instruments for modelling the sequential nature of the input signal. They have a discrete probability distribution that is stored in a large table containing number of states times the number of transitions entries. The model ensures that the normalization constraints of the discrete PDF given by $\sum_{s \in S} p(s) = 1$ and $\sum_{s' \in S_{(5,5,3)}} p(s|s') = p(s)$ are satisfied. Furthermore, the probability mass of $p(s)$ is divided evenly over the three jump widths. One third is reserved for the 0-jump and the other two thirds are given to the 1- and 2-jump.

every material intersection. Overlapping signal information of different recursion depths could then be interpreted as a result of multipath radio propagation.

[5]The Viterbi Decoder can also be configured to use other distance functions. Pretty much equivalent results in terms of errors were archived with the computational less expensive manhattan distance (l_1-norm) restricted to the max 3 signal deltas. The restriction to the top 3 deltas anticipates the greater weight of component distances of l_p-norms with $p > 1$

Therefore, all three jumps are equal probable. If additional sensory acceleration information become available in a future upgrade of the system, these jumps can be differently weighted.

The first take on the sequential nature of the problem, is made implicitly by choosing the $(5, 5, 3)$-transition-model. This can be interpreted as setting $p(s|s') = 0$ for all s' that are not included in the $(5, 5, 3)$ neighbourhood, effectively forcing the resulting path hypothesis to resemble a chain of locations that skips at most one state. The modelled jump width of $(0, 1, 2)$ is the classical approach to allow a time alignment of the received emissions to the hidden states. In other terms, the model can adapt to different speeds of movement. Standing at a position, is by that reason modelled as a series of 0-transitions, whereas faster movement leads to series of transitions with a higher probability of skipping states by using a jump width of 2.

By utilizing the $(5, 5, 3)$-model, a maximum movement speed is also introduced. Two consecutive signal vectors are only mapped to states that are at most $1.2m$ away assuming a states represents a cube with edge size $0.6m$. So in the case of gaps in the received data stream, accompanied by movements exceeding such distance thresholds, the model will not be able to compensate. Therefore, the sequence of measurements is interpolated to ensure, that there are enough time frames t to cover a distance that is needed by an assumed maximum walking speed of $3ms^{-1}$ (see figure 4.4). Instead of such seemingly artificial restrictions it would alternatively be possible to increase the maximum jump width by employing a $(7, 7, 7)$- or even larger model. But since this leads to a cubic increase in the number of allowed and therefore processable transitions for each state, this is computational expensive. Thus, the chosen $(5, 5, 3)$-model seems to be a good compromise between adaptivity and computational efforts.

The limitation of the maximum jump width to 1 in the vertical direction reduces the number of allowed transitions from 125 to 75 and is justified with the observation, that natural walking movements are primarily in the horizontal 2D-plane. The remaining adaptivity in the vertical dimension has proven to suffice for handling tracks containing significant parts of stairway zones[6].

The next source of information that is exploited to adjust $p(s|s')$ for the remaining transitions defined by the $(5, 5, 3)$-model is the 3D-Scene. If the 3D-cube, represented by either s or s', intersects with a face of a 3D-object that has a material flagged as blocking, then $p(s|s') = 0$. For transitions with a jump width of two, it is additionally checked if there exists an unblocked path of at most two steps between the source and the destination state of the jump. If this is not the case, then $p(s|s') = 0$. This removes the possibility to jump over blocked cubes. See example 5 in figure 4.5.

If one of the both states is more than $2m$ above cubes with blocking material, then $p(s|s') = 0$. This removes the possibility to fly. And the last condition for $p(s|s') = 0$ is given by states, that are located above cubes containing material flagged as impassable. This removes the possibility to walk over furniture like tables but only if the 3D-model exhibits detail at this level.

After applying these constraints, the probability mass for the transitions will then be evenly distributed over all $p(s|s') \neq 0$. This results in an already reasonable

[6] Although the author guesses that using the elevator that is present in the UMIC scene, will probably break this approach.

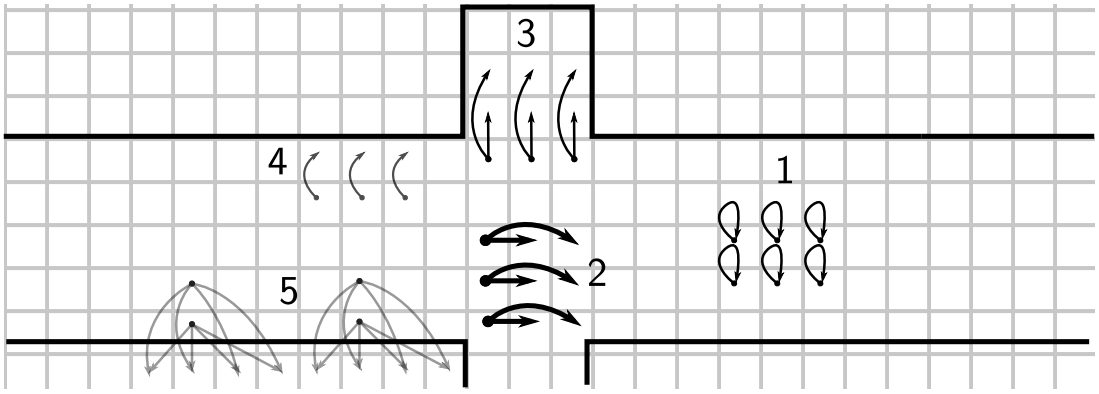

Figure 4.5 Different cases for $p(s|s')$ at the junction point of a pathway: 1) 0-jump with constant $p(s|s') = \frac{p(s)}{3}$. 2) Elevated $p(s|s')$ due to long line of sight. 3) Medium line of sight. 4) Reduced $p(s|s')$ due to short line of sight. 5) $p(s|s') = 0$ due to blocking material.

performing model that excludes most of the invalid path hypothesis and complies with the patterns of natural walking.

The last approach to fine tune the transition model, is given by adjusting $p(s|s')$ to the length of the line of sight in the direction defined by the jump from s to s'. More free space in jump direction leads to elevated $p(s|s')$ whereas less free space, i.e. standing before a wall, gives a penalty on $p(s|s')$. Employing this technique supports the natural moving direction in long pathways at the cost of paths that lead directly before furniture or tables. See examples 2, 3 and 4 in figure 4.5.

4.3.1.4 Pruning

The viterbi algorithm is made more efficient by pruning unlikely hypothesis from the search space. This is motivated by the observation, that a lot of the intermediate hypothesis have a very low probability as they are terminal states of paths that are far away from the real position. This induces a large number of mismatching RSS readings leading to very low emission probabilities. A problem, that can arise by removing unlikely hypothesis from the search space, is encountered if the pruning is to aggressive and removes hypothesis that are unlikely in the some time frame t but will become later the most probable. In this case, the best hypothesis, the best matching path to the measurements, is not found and the quality of the result degrades. It was sufficient to retain only the top 6% hypothesis of the search space, which are around 3000 in the UMIC scene, without loosing relevant hypothesis. Although the parallelizability of the Viterbi Algorithm is negatively affected, as there is more shared data needed, leading to more critical sections, a speedup of around factor 10 is archived.

A crucial data structure for an efficient implementation of the pruning viterbi decoder is a skiplist (William Pugh, 1990), which is needed to keep track of the top N hypothesis during the evaluation of a time frame. The skiplist represents a list of permanently sorted items, here the top hypothesis, with insertion cost of $O(log(n))$ during decoding. Although the data structure can be implemented to allow for

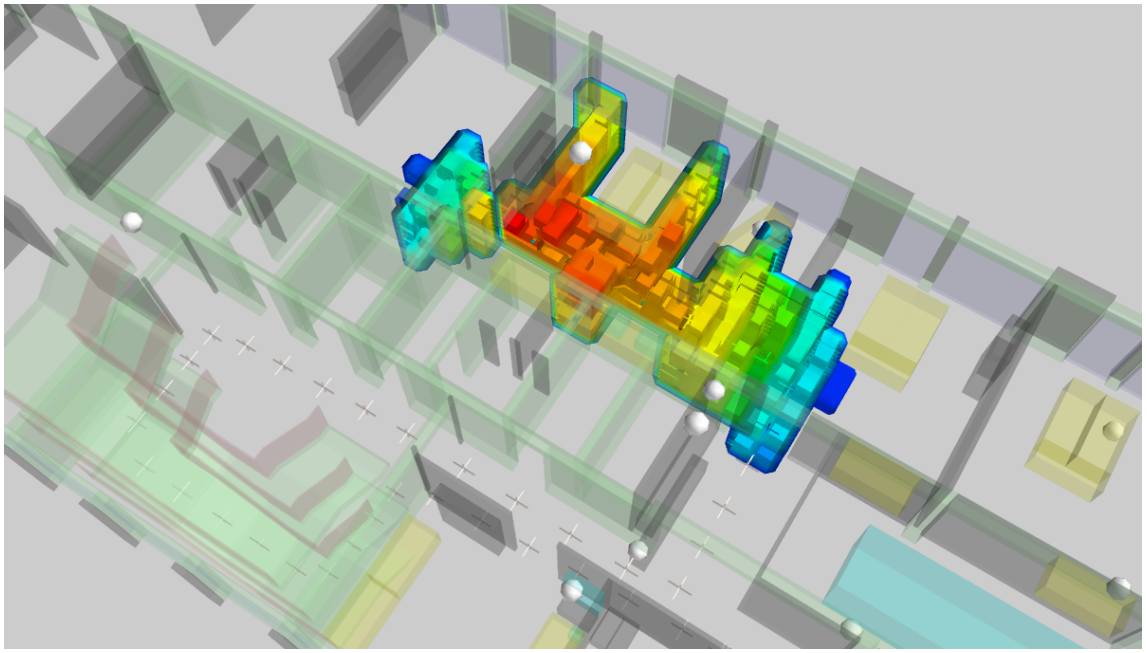

Figure 4.6 Top 300 unpruned states at the beginning of a location tracked measurement run visualized by the white cross symbols. The path starts on the first floor and ends on the ground floor by passing the stairways.

concurrent unlocked insertions this has not been done in the presented decoder due to time constraints.

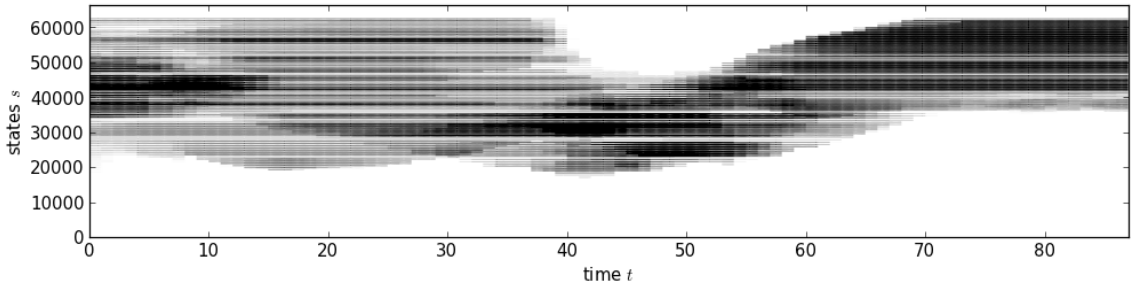

Figure 4.7 Distribution of 500 unpruned from 66000 total States raveled into one dimension over all timeframes of a decoding run. Darker areas indicate more unpruned states that have to be processed at the corresponding 3D location.

The addition of pruning to the Viterbi Decoder makes it feasible to decode a sequence of measurements in real-time on a resolution of $0.20cm$ leading to $2 \cdot 10^6$ states on commodity hardware. Although only a third of this resolution ($0.60cm$ leading to only $\frac{1}{27} \cdot 2 \cdot 10^6$ states due to the cubic nature of the problem) was employed on the designed models, as further detail has not yet brought any better evaluation results, having such a computational reserve is promising. Especially, if possible enhancements like further complex models such as HMMs of higher order are assumed. Introducing for example second order HMMs leads to an expansion of the search space by the number of possible transitions, therefore also to a speed reduction of factor 75 with the chosen $(5, 5, 3)$ model. It is expected, that this is still computational tractable on current hardware.

4.3.1.5 Result Sequence

The Viterbi Decoder can return three different result sequences. The first is the best sequence s_1^T for all given measurements x_1^T. This sequence represents the result of an offline run. Furthermore, the algorithm returns the sequence containing the best state hypothesis s_t for a time frame t *during* the viterbi decoding and represents the results of an online run. By averaging the position over the top N location hypothesis for a time frame t, the third result sequence s_1^T is defined and called the averaged online result. The offline s_1^T has a higher probability of correctness than the online variants since the latter has less information to rely on, as the future measurements are excluded.

4.3.2 Particle Filter

The localization problem can also be approached by modelling the states as continuous variables in the state space model (Figure 2.11). Such an approach is given by the Particle Filter. As in the HMM approach, the most likely sequence of states s_1^T for an observed measurement sequence x_1^T has to be found. But instead of searching this sequence by efficiently enumerating and evaluating the possible hypotheses, the sequence is generated by an iterative sampling process.

Although a basic assumption of the PF are continuous states, the system maps them into a discrete voxel space. Therefore, most of the data-structures from the HMM design can be reused. Voxels with $40cm$ edge size have shown to represent a resolution for reaching optimal results.

4.3.2.1 Emission Probabilities

The parameters of the emission probabilities $p(x|s)$ are similar to the HMM design. $p(x|s)$ is a Gaussian with a mean vector that is extracted from the radio propagation models by averaging over the RSSI values of corresponding voxel groups. The PF does not use the simplified logspace approach of the HMM model, and does therefore not reduce the emission probabilities to distance based cost values. The full gaussian is evaluated and a signal variance of $5dBm$ has been empirically determined to lead to good results. Missing components of the input signal vector x_t have been interpolated from neighbouring signals as described in the related HMM chapter 4.3.1.2.

4.3.2.2 Transition Probabilities

The transition probability $p(s|s')$ is modelled as a multivariate zero-mean Gaussian with independent components. Three dimensions of the Gaussian represent the three axis in free space. The variance of the horizontal components has been set to $5m$ and the variance of the vertical axis to $2m$. The reduced variability in the vertical axis can be justified by the lower probability to move in that direction. Both values have been empirically determined by testing the PF on location annotated measurements.

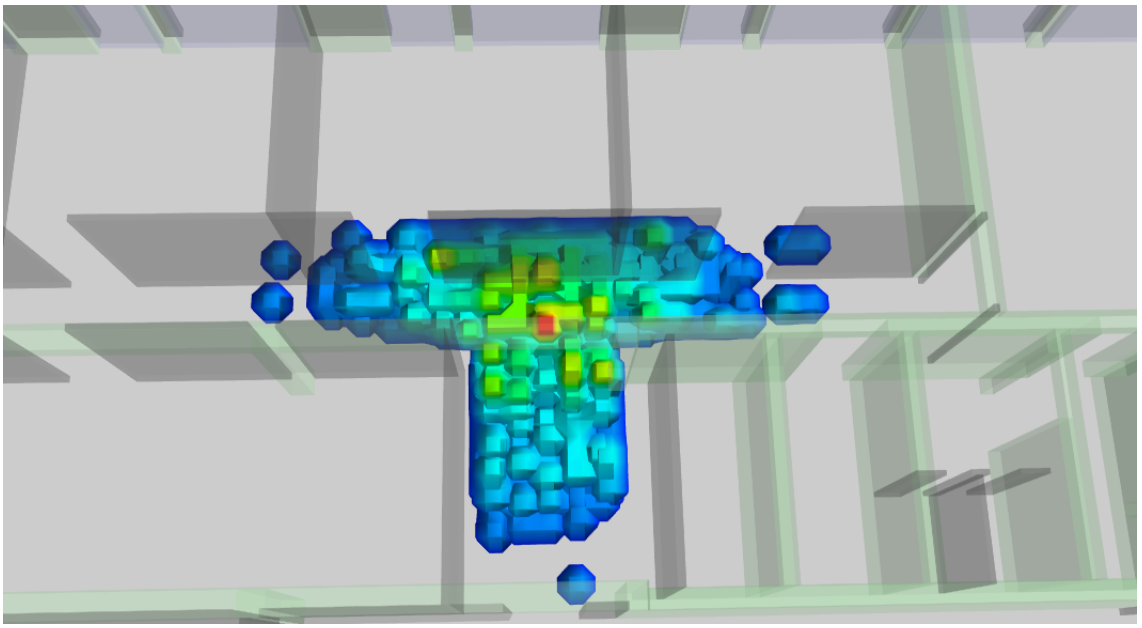

Figure 4.8 Scene geometry restricted sampling of the transition probability $p(s|s')$ during the driving process of a Particle Filter. Here, 1000 valid destination states s were generated after 2184 trial samplings. The mean of the Gaussian, the source state s', is located at the red voxel.

As in the HMM approach, the information of blocked zones can be derived from the 3D-scene. 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 to the source state s', are rejected. A result of this sampling process on a crossing is visualized at Figure 4.8.

10^5 samples are generated for each iteration of the driving process representing a single time frame. Further increasing this number has not shown to have any positive influence on the error rate of the PF. This relates to around 1 samples for each m^3 during the initial uniform displacement on the $8900m^3$ volume of the UMIC scene. The samples are drawn in a round-robin scheme from a pre-calculated pool of Gaussian values. Since the constraints of the 3D-scene lead to rejections of samples, and therefore a resampling from the pool, a hard limit on the number of resamplings has been imposed. Thus, the algorithm is able to proceed timely in pathological cases where most samples are rejected.

4.3.2.3 Sample Impoverishment

A problem that arises in the PF algorithm is the degeneration of the current set of states into a region of space that has no sufficient probable connection to the real location. The process is stuck in that mispredicted region and cannot recover to the true region by moving through the sampling process. This does especially occur in the presented PF, if the 3D-scene restrictions are applied. A problematic zone is given by the stairways where the possible path has the form of a corkscrew. If the sampling process ”misses” the path trough the stairs, when the device moves from one floor to another, the degeneration case is imminent.

The case is detected by a drop of the combined emission probabilities $\sum p(x|s)$ of all samples by several orders of magnitude. This sum is always available, as it is used for the weight renormalization described in 2.6.2. A possible strategy to resolve this situation is given by feeding new uniformly distributed particles into the state space. Another strategy that works well on the presented system is given by simply deactivating temporarily the geometry constraints. After recovery of $\sum p(x|s)$, the constraints will then be reactivated.

4.3.2.4 Result Sequence

The Particle Filter returns three different result sequences. As for the HMM Viterbi Decoder, the first sequence s_1^T evaluates all x_1^T and represents an offline run. This sequence is reconstructed after processing all T time frames that uses information stored in a backtracking table. This table stores for each time frame t and each state s the state s' that was responsible for spawing s. The best s_T is used as the starting state to reconstruct the sequence by iteratively following the state linking backwards.

The second sequence s_1^T that is returned, is given by the particles that have the highest emission probability $p(x_t|s_t)$ for each time frame t. As in the HMM approach, this stands for an online result as it does not depend on the remaining future measurements $x_{t+\{1..T-t\}}$.

The third sequence s_1^T is also an online result but it contains the locations that are the geometric means of the top N particles ordered by $p(x_t|s_t)$. The averaging is more expensive since a sorted representation of the particles is needed. But by utilizing the skiplist data-structure, that is used also in the HMM pruning technique, described in 4.3.1.4, keeping track of the best particles is performed efficiently. This result sequence leads to lower localization errors than the non-averaged variant.

4.4 Devices

No special attention is given to design of the device software. After starting the localization service, it enters a simple loop that pushes the last RSSI readings over HTTP to the Server. For reading RSSI values from the available Wi-Fi-APIs, the service chooses a device specific implementation. Currently two such implementations exists, one for linux/libpcap based devices and another one for android based devices. Localization results can then be analysed with a browser to fetch various reporting and visualization pages from the server.

4.5 Fat Client

The Fat Client is used for debugging and evaluating the different aspects of the localization framework. It is build on a GUI toolkit that embeds the VTK - Visualization ToolKit for analysing 3D-data from different sources. Furthermore, the Fat Client embeds an interactive scripting environment in the form of a Python interpreter for

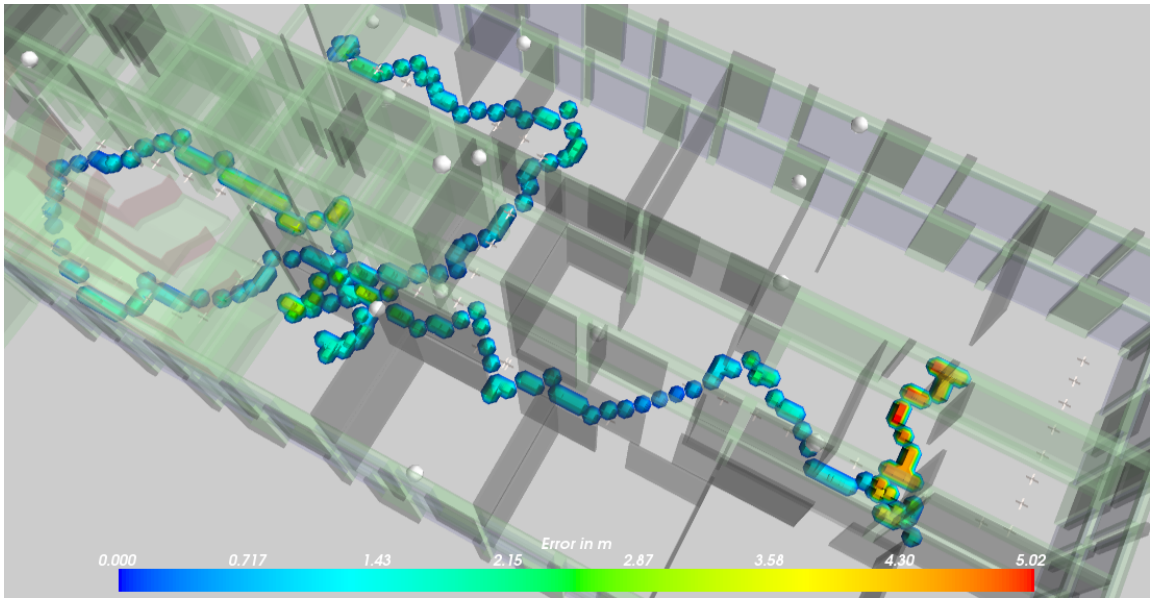

Figure 4.9 Localization result of a tracked path visualized with the VTK 3D-engine that is used in the Fat Client. The path start in the center of the first floor and ends somewhat right on the ground floor. At the end of the path the localization errors are up to $5m$.

inspecting the various properties of the models during runtime. Starting with the initialization phase, the Fat Client receives all needed configuration data for a scene from the server over HTTP. The configuration contains:

1. 3D-geometry of the building in form of a .obj-file.
2. Positions and associated information of all APs.
3. The location of the measurement points and the collected measurements from the different devices.
4. A corpus of location annotated measurements, also called tracked paths, for evaluation of the localizer performance.
5. A list of the optimization runs containing material parameters trained on the cluster nodes.

With an initialized system, the user gets the ability to select a raytracer configuration and the material parameters of an optimization run to request the simulation of radio propagation models from the Server. The Server distributes these requested jobs over the cluster nodes and receives all simulation results that are then further redistributed to the Fat Client. This makes the Fat Client independent of a local CUDA environment that is needed for running the GPU-driven raytracer.

With available radio propagation models for each AP, the localization engine is then used to evaluate and visualize localization result sequences s_1^T for position annotated measurements from the collected corpus of tracked paths. New paths

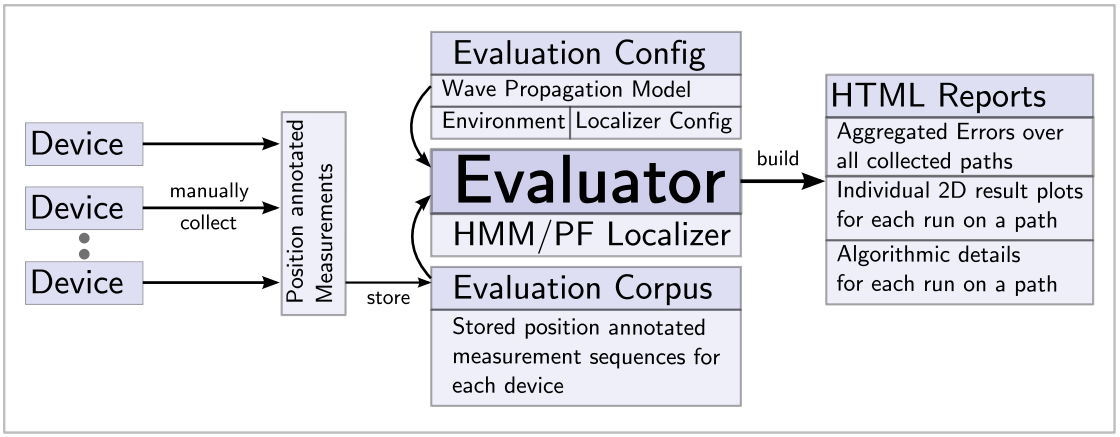

Figure 4.10 Schematics of the evaluation process for the overall localization framework performance.

can be designed and combined with synthetic measurements generated from radio propagation models.

As the running of the localization algorithms is only a CPU-bound task, in contrast to the radio propagation, this is done locally by using the same components that are also contained in the server deployment. Such a visualized localization result are shown in figure 4.9.

4.6 Evaluation

The evaluation of the different localization algorithms is performed by the Evaluator component that is shown in figure 4.1. The Evaluator has access to the corpus of position annotated measurements that were manually collected for different Wi-Fi capable devices.

The Evaluator is configured by different parameters that determine the target device, the used environment with radio propagation Model and the individual behavioural properties of the localization algorithms. The environment is constituted by a 3D-scene, with the relevant dimensionalities. The environment is enriched with SSMs of all configured APs for a selected material parameter optimization run. Furthermore, the resolution of the state space in ratios of the radio propagation resolution can be defined. A $1:1$ ratio leads to $2 \cdot 10^6$ states with $20cm$ edge size in the UMIC scene. At least, the localization algorithm is defined and the corresponding default configuration values can be overridden.

The evaluation is started by selecting a subset from the available evaluation corpus. These selected paths are then evaluated in parallel by running multiple Localizer instances. The different result sequences are compared with the available correct positions and different types of errors are computed. The HMM and the PF localizer return an offline, an online and an averaged online result sequence (see 4.3.1.5 and 4.3.2.4). The different types of errors contain the averaged error over all typed sequences in 2D and 3D form. Furthermore, the median error over all typed sequences for each dimensionality is reported. The median error is more robust to negative

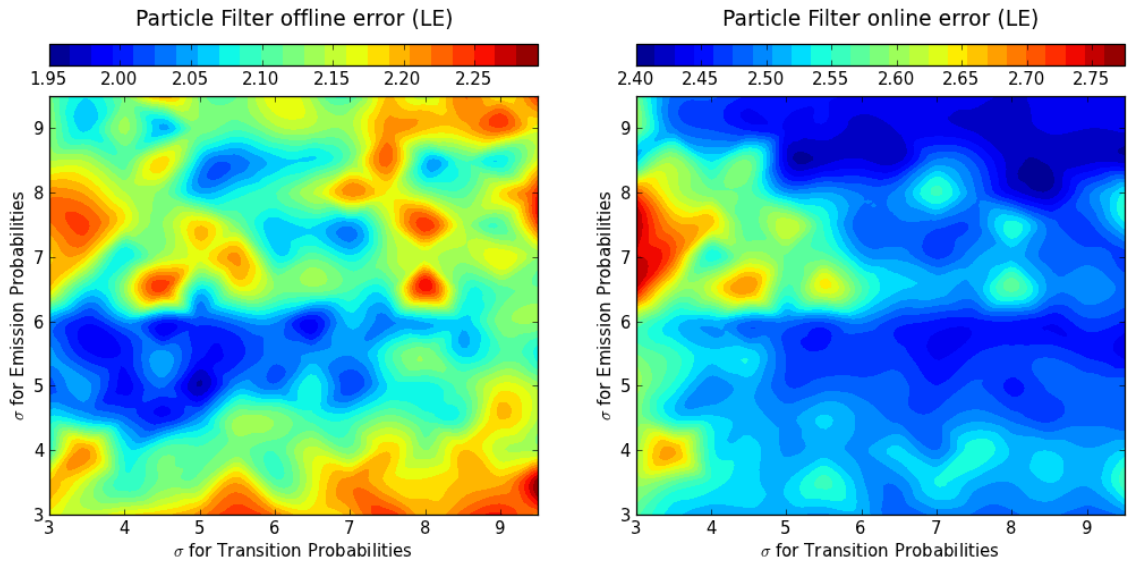

Figure 4.11 Example results of a brute force optimizing run with the Evaluator. Empirical values for the emission and transition are determined by looping over combinations of σ values for the transition and emission probabilities of the Particle Filter model. Different error types lead to different optima. The range of $\sigma = (5, 5)$ yields acceptable results for both error variants.

outliers as localization failures to recognize the transition from one building level to another or other pathological cases.

The results of an evaluation are given in plain HTML-reports for the different possible aggregations of the errors. Such aggregations contain sub-errors over the different configured paths. Results for individual tracked paths are drawn on a 2D-map and combined with a HTML-report of the algorithmic steps in resolution of a time frame. The results can be afterwards accessed via the HTTP-Server.

The Evaluator component was used to empirical fine-tune the different properties of localization algorithms. For example, the variance of the emissions and the transitions of the PF were determined by a brute force search over the parameter space as visualized in 4.11. A typical efficiently configured evaluation run for the 160 collected paths for a single device takes around 2 minutes on current Hardware. Thus, low dimensional brute force searches are feasible.

4.7 Summary

The presented architecture of the localization framework captures all necessary aspects that are needed for solving the positioning and tracking problem. A central HTTP-Server, build around loosely coupled components, provides the different communication channels between the consumers and producers of information. These information sources contain moving Wi-Fi capable devices, GPU-nodes and Fat Clients.

After allocating a 3D-model of the indoor localization environment with position configured APs, the PHOTON raytracer is employed to build a radio propagation

Model for the expected RSSI values. The free parameters of the model are then trained with a genetic algorithm that is distributed over an array of GPU-nodes.

The RSSI based localization algorithms, the Hidden Markov Model and the Particle Filter, are based on the same basic model assumption given by the state space model of Figure 2.11. The model parameters for the emission probabilities of the two algorithms are derived from the trained radio propagation models. Although the transition probabilities for both algorithmic approaches are modelled differently, both rely on the 3D-geometry for restricting invalid movements. Position annotated measurements, for training the model parameters with the classical parameter inference algorithms like the Baum-Welch algorithm in the HMM case, are not assumed to be available.

The Wi-Fi capable devices are only dumbed down information providers and have no higher functionality.

The support infrastructure is given by the Fat Client that is used for interactive debugging of the algorithms and associated data-streams, and finally by the Evaluator that is utilized to measure the overall performance in terms of error rates of the localization framework.

5

Implementation

In this chapter, the implementation of the localization framework will be examined under the different aspects of modern software engineering. The framework is responsible for the training and the simulation of the PHOTON generated propagation models. These models are the foundation for the implemented localization algorithms that process the RSSI data streams of mobile devices. A GUI can be used to analyse and debug the different data streams with an interactive 3D engine and an evaluation engine generates HTML reports over different propagation model/localization algorithm settings.

The framework has been written from the scratch during the six months of this thesis and although some design decisions have been made more in an "ad-hoc" way, it has proven to be a valuable tool for the posed problem. The Server, the Fat Client and the device software have been written in the Python programming language. Since Python is an interpreted languages, this makes the framework in general platform agnostic. Most of the development has been conducted under the Windows operating system whereas the primary deployment of the Server is running under Linux. The Fat Client was also successfully tested under OSX.

Python is an imperative, interpreted, dynamically typed, general purpose programming language with a very readable syntax. All the important compound datastructures like tuples, lists, dictionaries (hash maps) and sets are first class citizens in the language and can be manipulated with powerful techniques like List/Dict/Set Comprehension. Memory management is handled deterministically by a reference counting based garbage collector. Threading support is usable for tasks that should handle blocking IO, but Python byte-code will only be executed serially in the interpreter VM. The standard library, although not as excessive as the two behemoths JRE and CLR, covers all basic needs of modern applications. Additionally, a healthy environment of third party libraries, especially in the context of scientific information processing and web development, satisfy all the remaining needs.

The luxuries of the Python feature set come with a cost: Python is slow. But slowness is a relative measure. Interpreted Python is about 50 times slower than

compiled C and still around 20 times slower than JITed Java. For most of the individual use-cases found in the localization framework, this is fast enough. But for the number crunching cores of the localization algorithms, the use of pure Python can be excluded. Therefore, these parts of the frameworks that should be fast and memory efficient, are implemented in Cython, a Python-to-C compiler. Cython allows seamless integration of compiled machine code into the Python runtime and achieves significant faster execution times for computational and memory intensive operations.

The Python code is structured primarily with namespaces and classes. Instances of classes are mostly used as scoping containers, higher object oriented concepts such as subclassing is only used sporadically in the framework. Writing idiomatic Python code, adhering to the so called *Zen of Python*, was a relevant design principle. Sadly, the rules of good software practice have been violated by ignoring the task of writing unit tests for the framework. But up till now, the implementation is rather represented by laboratory than by an assembly line, so the need for them was not that pressing either.

The third programming language that was used, is JavaScript. The output of the HTTP-Server includes some dynamically generated HTML pages. There, the JavaScript code is responsible for adding some interactivity to the HTML frontend. The JavaScript code is strongly JQuery driven. The following table, the output of cloc-count[1], gives an overview over the usage of the three programming languages:

Language	Files	Blank	Comment	Code
Python	26	2530	1001	8627
Cython	1	472	216	1347
Javascript	2	86	13	328
Total	29	3088	1230	10302

5.1 Third Party Libraries

A number of open source libraries from the Python ecosystem were used in the framework. All, but Pygene, a set of genetic algorithms, are liberally licensed. Pygene uses the GPL v2, but the others are MIT like licensed. All, but Pygene, are actively maintained and backed by a strong community. But Pygene is a small library that can easily be replaced/reimplemented if that should become necessary.

The NumPy/SciPy/Matplotlib/Cython library ecosystem was the most important for the realization of the framework. There are no real alternatives of the same quality, so they cannot easily be replaced. The other presented libraries have either comparable alternatives or are only employed for less important use-cases as debugging[2] the data streams..

[1] http://cloc.sourceforge.net

[2] Less important with respect to the final results of the thesis, but surly important for reaching them.

NumPy/SciPy/Matplotlib

The NumPy library is a major building block for the data structures of the framework. Numpy provides fast and efficient multi-dimensional array structures (ndarrays) to the Python runtime. Additionally, a large number of linear algebra algorithms are available for the ndarrays due to linking against the BLAS/ATLAS runtimes. With NumPy, tight loops of custom algorithms have to be handled with blockwise elementary operations over the ndarrays. A multitude of elementary operations like `sum()`, `max()`, `abs()`, are predefined and the method of choice if speed is crucial. But designing algorithms under that "blockwise" philosophy leads to somewhat awkward formulated implementations. Therefore, only the ndarray data structures, the infrastructure for file-IO and some filters for data preprocessing/data visualization have been used. For example, the PHOTON output is loaded directly into the ndarrays with a single line of NumPy.
By this choices, the implementations of the localization algorithms use only the ndarray data structures and implement the tight loops in Cython, as this leads to more flexibility and readability.

The SciPy library supports NumPy by offering a large number of efficient numerical routines working on the ndarray data structures. Especially the muti-dimensional image manipulation routines have proven to be very convenient during the thesis. And finally Matplotlib, a very flexible plotting library that integrates well with the NumPy data structures and provides publication quality figures. The combination of these three libraries with Python scripting glue represents a more than adequate alternative to scientific computing environment like MATLAB.

Cython

Cython is a Python-to-C compiler that was used for implementation of the localization algorithms. If the compiler is executed on pure Python code, the gains in the form of execution speed are minimal. The real performance improvements come from hints that are given in the form of type annotations. Tight for-loops that use only such typed variables are directly translated into their C equivalents and executed independently of the Python Virtual Machine. This leads to speed ups of multiple orders of magnitude compared to code running in the Python VM. Furthermore, strong support for the ndarray data structures is given which results in a seamless integration into to NumPy/SciPy/Matplotlib ecosystem. Consequently, the SciPy library consists of a large number of Cython implemented routines.

Another benefit of Cython comes by the flexible control over the threading behaviour. As mentioned in the introduction of this chapter, Python has no strong threading support. Native OS threads are supported but the execution of the thread-specific Python byte-code is conducted serially due to a globally enforced interpreter Lock (GIL). The Cython library enables to control this locking behaviour. It allows to release the Lock temporarily, and therefore let the Python VM continue on other threads. But this is only allowed if the implementation ensures not to access any objects from the Python runtime, during the lockless execution.
This is exactly the case during the tight loops of the localization algorithms where

accessing Python primitives like Lists or Dicts is out of the question anyway. Releasing the global interpreter lock makes Python based threading efficient. For example, 4 instances of an algorithm can be started simultaneously from the Python VM and execute their binary code in parallel. Even more fine grained threading control can be reached by using the support for OpenMP instructions that are available in pure C space. By only using a slightly modified syntax for loop control, the loop becomes instrumented by the OpenMP backend compiler support (GCC or MSVC) and will be executed on multiple automatically spawned threads.

The localization algorithms have therefore been implemented by applying the same procedure: prototype the algorithm in pure Python on smallish datasets with ndarrays. Move the Python code to Cython and make the algorithm fast by applying type annotations. If the algorithm is then only composed as a sequence of GIL-free C loops over ndarrays, it can already be executed in parallel. Tuning the multi-core capabilities even more with OpenMP has only been done for the HMM implementation as the "magic" of OpenMP imposes some constraints on the algorithm. These constraints relate to sharing of variables over different scopes which would invalidate some assumptions of the Particle Filter implementation. Although this is not a surprise, as efficient parallel algorithms are often difficult to implement.

CherryPy

CherryPy is a light-weight HTTP-Server implemented in pure Python. The Server component of the framework is designed as a CherryPy application. A central concept of a HTTP-Server, the URL-routing is defined by exposing functions of the application to act as URL-handlers returning some mime-type encoded string. Providing a new service behind an url is therefore cheap and has made the prototyping of the different communication channels between the information producers/consumers an easy task. Although CherryPy acts as a standalone HTTP-Server it would make sense to run the service behind an Apache proxy server in the future.

Pygene

The Pygene library was used for the implementation of the optimization process for the radio propagation models. It provides a set genetic algorithms that are modelled around the concepts of population, organism and gene. Organisms are a compound data structure for genes and populations hold a number of organisms. Genes represent parameter values of an optimization and have a type corresponding to the primitive Python types like floats. Furthermore, genes can define upper and lower boundaries for their values. An organism holds a collection of genes with values and represents a solution in the search space of the optimizer. The search space is explored by crossing the genes of selected organisms from a population. This crossing leads to a new population of organisms and represents a new set of solutions.

It is an exotic library with respect to the GPL license and the small algorithmic core. It would not be that hard to implement a comparable set of features, but the inner workings of genetic algorithms were of minor concern in this thesis.

SL4A

The Scripting Layer For Android (SL4A) framework was used for programming the device software. The SL4A enables the use of Python on Android devices and provides access to the larger part of the Java-based APIs. Only the Wi-Fi APIs have been accessed which has resulted in around 50 lines Android specific code. It can be expected that the distribution of SL4A based Python scripts will have deployment issues in contrast to native Java based applications. But it would be easy to get rid of the SL4A dependency if this becomes be necessary.

Traits-UI and MayaVi

The Traits-UI and MayaVi libraries are used to drive the core of the Fat Client. Traits-UI is a Python based GUI framework with a nice balance between the need for configurability and the "beauty" of the results. The MayaVi library is a wrapper around the VTK 3D visualization toolkit and provides the integration, of this excellent exploration tool for 3D data of different shapes, into the Traits-UI environment. Both have been shown to support a rapid prototyping style that was needed to adapt to the different ideas that have come up during the time of the thesis. The API of the 3D MayaVi plotting library is conveniently designed after the 2D plotting library Matplotlib. This makes the mental overhead even smaller. The central Server component has no dependency on the rather fat libraries.

ConfigObj

The configuation of the localization server is driven by a .ini style config format that is processed by ConfigObj library. Its is a pure Python library with the capability to map the string represented config parameters to primitive Python types and simple compound structures like *int_lists*. For this mapping a specification file, also in .ini style format, has to be defined which represents also a good place to document the behaviour of these values. This is comparable to define a XML-Schema for some custom XML configuration dialect but with readability included.

JQuery

The JavaScript that is executed in the browsers of the mobile devices and on some debugging terminals is driven by the JQuery framework. Adding dynamic elements to the HTML pages, for example the interactive recording of location annotated RSSI readings, has not required much efforts due to convenient API of the library. JQuery has a memory footprint of 25kb which should not pose any problems for consumer connection conditions.

5.2 Modules

In this section, the different major components of the framework are described. The used data structure are focused and interesting algorithmic aspects highlighted.

5.2.1 Server

The Server component implements the core of the localization framework and is implemented as a CherryPy application. The Server can be started from the Python package `lws` by pointing to a configuration file:

```
>> python -m lws.cli path_to_config.ini
```

After loading the config and initializing the HTTP-subsystem, the Server listens for commands from the clients. If such commands arrive over the HTTP channel they will be dispatched by the Server to the responsible subsystem. For debugging purposes most subsystems will be code-reloaded before processing the command. This has made developing the system very efficient.

Simulator

The Simulator is a subsystem of the Server and is responsible for coordinating the construction of radio propagation models. It is a singleton and drives a priority queue from the Python standard library. The queue contains `Job()` instances that represent a set of material parameters and a set of .obj (Wavefront Format) files. The consumers of the jobs are the GPU-nodes whereas the producers of the Jobs are either the Optimizer or the Fat Client. The Optimizer pushes jobs into the queue during the training process, and the Fat Client uses the Simulator for retrieving full propagation models that should analysed. The raytracer voxel arrays are compressed transmitted from the GPU-nodes over the Simulator to the consumers as they contain about $10mb$ raw data that can be reduced by a factor of 3.

The priority aspect of the queue is useful for serving "high priority" requests from the Fat Client before processing the next thousands of Optimizer started training jobs. Furthermore, it is more efficient for the Optimizer to combine all jobs for a single genetic organism, N jobs for the evaluation of the material parameters over N APs, in one priority slot since this leads to more efficient cluster usage.

Optimizer

The Optimizer is responsible for the training of the radio propagation models with an optimization algorithm and it employs the PyGene library for this task. It is initialized via type checked HTTP-GET parameters over the Server component. Such a parameter set is defined in table 6.3. Additionally to the optimizer parameters a restart of older optimization runs can be enforced. This restart is implemented by deserializing the last population of the old optimization run and using it in the current run. Additionally some new randomized organisms can be injected.

After the initialization phase, the core loop of the optimization run is started threaded. In the core loop, the evolutionary process is simulated by executing successive generations of populations with the PyGene API. The complex part of the implementation, the custom logic for the raytracer training is implemented in the `fitness()` function of a `RadioPropConverger()` class that is derived from a `pygene.Organism()`. The `fitness` function is called from the PyGene framework for each organism instance of

a population. In this function, a `Job()` for each AP is queued in the Simulator and configured with the free parameter values obtained from gene values of the organism instance.

The different free parameters of the model have to be declared in the `genome` class variable of the `RadioPropConverger()`. Each element of the `genome` is a named subclass of `pygene.Gene()`, for example in the Optimizer implementation the `MaterialGene(pygene.FloatGene)` gene class is used.

If all submitted jobs are returned, the fitness of organism can be computed which leads to progress in the population evaluation. If all organisms are fitness-evaluated, the Optimizer can decide to stop the process. This is either decided by a hard limit onto the number of allowed generations of if the difference between the quality of two populations becomes to small.

The individual `fitness()` results with the corresponding free parameters are stored and the full convergence process can be visualized with the Plotter. The output of such a visualization can be seen in figure A.10.

Plotter/Evaluator

A smaller component that is only implemented at the module level, is the Plotter. Its main responsibility, which can be derived from the name, is the output of different graphics that are used to understand the localization results of the framework. It is accessed by the Evaluator and the Server component. Whereas the Evaluator uses the Plotter to produce plots of the evaluated localization results (see figure A.9) or other special details of the algorithms, the Server only retrieves these plots and delivers them to the clients. The plotter uses primarily the presented Matplotlib library and a generic 2D drawing engine, the Python Image Library (PIL). Since the different produced plots share only minor properties, the Plotter consists only of a list of functions instead of being implemented at the class level.

The Evaluator is implemented as a Python class that it responsible to evaluate the error rates and other properties of the localization results. The instantiation of an Evaluator class is configured by a set of initialization parameters that describe the setup of the environment and the configuration of the chosen localization algorithm. The instance can then be instrumented to analyze a part of the evaluation corpus by calling the `evalAll(pathid2runids)` method. The pathid2runids represents the chosen subset of the corpus and is a mapping from pathids to lists of sample runs. The chosen samples of the corpus are evaluated in parallel by using the convenient `concurrent` package from the Python standard library. Since all algorithms are executed in GIL-free C-space, the Evaluator is able to exploit multiple cores efficiently.

After computing the localization results for the chosen samples, the different variants of the error rates are determined and stored in path-separated XML files for later inspection.

Renderer

The last subsystem of the Server is the Renderer that is used for generating HTML pages for the various web-browser based user interfaces. It consists only of a col-

lection of functions that are used to generate the HTML code with Python string processing capabilities. Further development on the framework will probably induce the need for a higher level templating engine.

5.2.2 Localization Algorithms

Each localization algorithms in the presented framework is implemented as a Python class derived from the `Localizer()` class. The `Localizer()` class is responsible for handling the general interactions with the framework, whereas the subclasses implement algorithmic specific code that has a low computational footprint. The implementation of the computational expensive parts is relayed to the Cython language for each of the three algorithms.

Environment

The information about the scene specific details needed for the application of the localization algorithms is stored in the `Environment()` class which is therefore available as a private member to the `Localizer()` derived implementations. The `Environment()` class is initialized by a list of AP definitions which will lead to loading the corresponding stored PHOTON output files. The 3D voxel data of PHOTON is stored in .raw files which compose of a simple concatenation of `float32` values, representing RSSI values, that are loaded into a 3D ndarray with NumPy-native functions.

Furthermore, the scene geometry, represented by a .obj file, will also be loaded by the Evaluator. An initialized Environment hold thus the two primary datastructures that are used for deriving the location conditional probabilities. The scene geometry triangle mesh is used for generating the ndarrays for the transition probabilities. And the PHOTON output is used for computing the ndarrays for the emission probabilities.

HMM

The HMM based algorithm is implemented in the `HMMLocalizer()` Python class in conjunction with the `ViterbiDecoder()` Cython class. On initialization, the `HMMLocalizer()` loads the cached or refreshes the needed emission probabilities by delegating the PHOTON data stored at the `Environment()` to the Cython function `buildStatesWithEmissionProbs()`. The returned ndarray `emission_probs` is stored at classlevel.
The ndarray `transition_probs` for the transition probabilities is also generated in this step. At first, the triangle mesh of the scene is voxelized by Cython accelerated functions. On the voxelized representation of the geometry, the transition probabilities can be efficiently computed by the Cython function `buildTransitionProbs`.

The `ViterbiDecoder()` class is initialized with both ndarrays for the probabilities and can be used from the `HMMLocalizer()` for the decoding of a sequence of RSSI measurements. The basic flow of the input in form of RSSI reading is given by the

call chain:
`HMMLocalizer().evaluateMeasurements(ms)` → `self.decoder.decode(ms)`. During the first call, the high level representation of the measurements that are instances of the `Measurements()` class is interpolated to adjust the number of RSSI readings to the constraints of the employed $(5, 5, 3)$ transition model. This has to be done since the coverable distance between two succesive RSSI vectors would be limited to jump over two voxels. Afterwards, the measurements are additionally converted into an efficient indexable ndarray.

The decoding process in `decode()`, implements the Viterbi Algorithm described in section 2.5.2 with the designed pruning technique described in section 4.3.1.4. To keep track of the best hypotheses during the decoding of the RSSI values for a timeframe, the SkipList [19] datastructure is employed. This datastructure represents a permanently sorted list with efficient insertions of new members.

After decoding the unpruned state space, which is essentially composed of multiple tight loops over the measurements ndarray and the ndarray of the probabilities, three Python lists representing the localization results for the online, averaged online and offline variant are returned.

For the offline variant of the resulting sequence, the backtracking ndarray `back_track` is employed which *links* each location hypothesis to its predecessor hypothesis. This ndarray dominates the memory consumption of the algorithm due to its $T \times S$ complexity, if T is the number of timeframes and S the number of locations. Therefore, the ndarray stores only `int8` values, which represent the *link* by its transition index which is of the range 0..75 for the $(5, 5, 3)$ transition model. If the transition model would be of higher complexity, and exceed 256 jump possibilities, the memory consumption would be doubled as an `int16` address space becomes necessary.

Particle Filter

The Python side of the PF algorithm, in the form of the `PFLocalizer()` class, is very similar to the described `HMMLocalizer()`. Contrary to the HMM case, no transition probabilities are generated from the scene geometry. Only the voxelized representation of the geometry is constructed that is later used for rejecting particles that interfere with that structure. The other difference is given by the missing interpolation step of the RSSI readings, as constraints like the $(5, 5, 3)$ HMM transition model are not given.

The `PFLocalizer()` prepares a pool of values for a three dimension Gaussian that are used in a round-robin scheme during the particle spawning or sampling process. This spawning is a part of the simulation of the modelled stochastic process which is implemented as the Cython sibling class `ParticleFilter()`. This class is initialized with and ndarray containg emission probabilities, the sampling pool and the voxelized representation of blocked zones. All three ndarrays are used during decoding the RSSI sequence.

To efficiently keep track of the top N hypothesized locations for a single timeframe, the SkipList datastructure is used similar as in the HMM case. The top N hypothesized locations of a time frame are used for calculating the averaged online prediction of a location.

Like the `HMMLocalizer()`, the `PFLocalizer()` decoder returns three lists of positions for the offline, online and averaged online result variant.

LMSE

The most simple algorithm, is the LMSE based approach which is implemented in the `LMSELocalizer()` class and the corresponding `LMSE()` Cython decoder class. The Python side computes only the ndarray for the emission probabilities for the voxelized RSSI values for each AP. This ndarray initializes the `LMSE()` class that is afterwards instrumented via its `decode(ms)` function to evaluate the measurement sequence of the input signals.

To efficiently keep track of the top N solutions for the nearest neighbour distance computations that are executed over the total localization space, the already in the HMM and the PF employed SkipList structure is used. The top N solutions can be understood as a simple clustering of the best locations and has for example in the ARIADNE [13] system been enhanced to a k-means clustering model.

5.2.3 Fat Client

The Fat client is the primary debugging tool for the behaviour of the radio propagation models and the algorithmic progress of the localization algorithms. It is build around the Traits-UI GUI toolkit that conveniently embeds the MayaVi 3D engine for scalar data visualization of the localization space. The location related scalar data has to be given in the form of ndarrays that are already used all over the implementation of the framework. Most transformations of the algorithm specific ndarrays to a MayaVi compatible structure is done by the native functionality of the NumPy/SciPy libraries. If the solution could only be awkwardly implemented with NumPy block operations, Cython functions have been implemented for better readability and flexibility.

The GUI needs access to the described Server infrastructure that is defined by its HTTP address. It uses the Server for initialization to a specific 3D geometry and the corresponding AP placement. Furthermore, it can investigate the training corpus for the radio propagation models and the evalution corpus for the localization algorithms. The PHOTON generated RSSI predictions can be visualized with the MayaVi engine after beeing processed by the GPU-nodes and subsequently delivered by the Server. Due to the GPU-driven nature of the PHOTON raytracer, this design choice makes sense as no special hardware and CUDA driver setup is needed on the client side.

As mentioned, the evaluation corpus for the localization algorithms are also fetched from the Server over HTTP and can be used to investigate the results of the algorithms. The algorithms are executed locally as the Fat Client has access to the code of the corresponding modules from the Server infrastructure. The different `Localizer()` classes can be instantiated with `Environment()` conditions that can be manipulated from the GUI. The algorithmic progress from timeframe to timeframe of the top N location hypotheses can be visualized for the HMM, PF and LMSE implementations. This is very helpful to find weaknesses in algorithmic properties.

Another convenient implementation detail of the Fat Client is given by the embedded Python interpreter which can be used to analyse most datastructures of the framework interactively. Formulating questions over this scripting interface is very efficient from a developer point of view.

5.3 Summary

In this chapter, a cursory overview of the chosen programming paradigms and some of the implementation details have been presented. As can be observed, the developer of the system, the author of the thesis was satisfied with the implementation results that have been reached under the constraints of a six-month thesis. The choice of Python and Cython as development platform has lead to an efficiently prototyped localization framework that employs fast algorithms with a low memory footprint. The chosen open source libraries, especially the NumPy/SciPy/Matplotlib ecosystem, have shown to be of very high quality and can therefore be recommended as well.

6

Evaluation

This chapter is dedicated to the evaluation of the presented localization framework. Special attention is given to the description of the setting and the environment conditions of the experiments as this was often only briefly reported in the investigated literature[1].

After describing the experimental setup, the results for the training process of the radio propagation models are analysed. A focus is placed on the behaviour of the training process under different levels of 3D geometry granularity. An additional experiment for the radio propagation context is conducted by evaluating the ability of the models to generalize over multiple devices. The generated models and the corresponding observations are subsequently used for the evaluation of the localization algorithms that were designed and implemented.

It will be reported how the three algorithms, the LMSE, the HMM and the PF, perform on a selected fully trained propagation model. In the first step, the algorithms are compared on synthetic data obtained from the propagation model under different noise levels. The differences between the results of the individual algorithms are explained by their underlying algorithmic properties. The use of synthetic data makes this task easier.

Using the results of the radio propagation evaluation and the experiences from the analysis on synthetic data enables a thorough investigation of the framework performance on real world conditions. Location annotated RSSI readings for a distance of about $8000m$ were manually collected for eight different track configurations of various complexity. This evaluation corpus is used to estimate the localization errors of the three algorithms over differently trained radio propagation models. Furthermore, the effect of the devised Device Adaptation scheme described in 4.2.3, is analysed and interpreted.

[1] Although it should be emphasized, that the investigated literature was mostly available in the form of journal papers which have surely made compromises due to page count constraints.

6.1 Radio Propagation Model

The evaluation of the framework starts with a detailed explanation of the environment that is used for the experiments. The introduction of the setup starts by describing the different properties of the manually created 3D geometry that is utilized to drive the PHOTON raytracer. Afterwards, the placement of the different AP groups and the resulting coverage conditions is analysed. In the next step, the composition of the training corpus that drives the optimization process is presented. A relevant number of measurements over 100 locations in the UMIC scene were taken with four different Android devices. These measurements will represent the largest part of the training corpus that is employed in the evaluation.

After the introductory part, it will be seen how plausible the trained propagation models adapt to the true nature of the environment by using the optimization criterion, the RPE, as a measure[2]. An experiment that investigates the training process over measurements of different devices and another one that tries to understand how important the granularity of the 3D geometry is, will finalize this part of the evaluation.

6.1.1 Scene and Setup

The evaluated localization scenario is located in the UMIC Building[3] of the RWTH Aachen. It is a four level building with a rectangular floor-area of around $900m^2$. The first three floors were modelled in Blender leading to a volume of around $8100m^3$, as the height of a level is around $3m$. It is composed of "reduced to functionality" office architecture and was erected in the year 2009.

6.1.1.1 3D-Model and Materials

The 3D-model consists of 11 different object classes given in the form of triangular meshes. The class of an object is defined by the assigned material. The choice of the materials was made by reasoning about their importance of their impact on the radio propagation. Probably the most important ones are the *Concrete* and the *LightWalls* class of objects. 2D-Maps of the building at the resolution of these two materials were available and are used as a base for these two classes of objects in the 3D-model. Further modelled detail is given by doors with the three material classes *IronDoor*, normal *Doors* and *GlassDoor*. The limits of the building are modelled with the two materials *Fascade* and *GlassWindow*. Additional furniture details are captured by the materials *Cupboard* and *Table*. Also included for completeness is the metal based railing of the stairs and hardware components that are located next to some of the APs or components that exhibit a massive structure, like the hardware array of the server room.

[2]This can lead to a serious overfitting of the model, but since the training corpus had only a size of 100 locations it has been concluded that they are to valuable to sacrifice some of them to allow the better cross-validation technique.

[3]Google Maps reference: http://g.co/maps/q3mp6

Material	N_{vertex}	$N_{triangle}$	$\approx V\ in\ m^3$	avg. Thickness in m
Concrete	3344	5849	1523	0.3
LightWalls	1822	2942	174	0.15
Doors	528	792	2	0.03
GlassDoor	480	684	1	0.02
IronDoor	120	180	2	0.05
Fascade	1432	2144	127	0.4
GlassWindow	1372	2046	28	0.05
Cupboard	368	540	12	n/a
Table	328	492	13	n/a
Hardware	48	72	3	n/a
Railing	240	408	3	n/a

Table 6.1 Properties of the UMIC scene geometry with respect to the different material classes.

All these materials are listed in table 6.1 with corresponding vertex and triangle counts for the UMIC scene. The enclosed volume V for each material has only been approximated from a voxelized representation. Since the voxel-representation tends to overestimate the volume at the boundaries of wall-like objects, the volume was adapted with respect to the empirically determined average thickness of the scene objects for each material.

6.1.1.2 Accesspoints

As the primary information source for the localization system, 22 APs were employed in the evaluation. 9 of these APs are part of the EDUROAM infrastructure, that is provided as a service of the university and are therefore stationary in the experimental setup. 7 APs are ASUS models and were placed to cover the floors evenly. 4 APs of a different ASUS model were part of an experiment of another research group and are also assumed as stationary. The last two APs are Linksys WRT54G models, that were salvaged from the harware pool of the author. They were placed near the stairways since the other APs had somewhat neglected this area.

As can be seen in figure 6.1, the coverage density of the APs is pretty high. Most space is covered by at least 6 APs. As is concluded later in the evaluation, lowering AP density below that threshold leads to much higher errors in the localization algorithms. Further steps to optimizing the setup can be conducted by relocating the APs with the target to maximize the average AP density. Other optimization targets for a guided relocalization could try to maximize the variety of the signals, as this is the primary information source for the localizer. These approaches can probably be automated with the help of the raytracer.

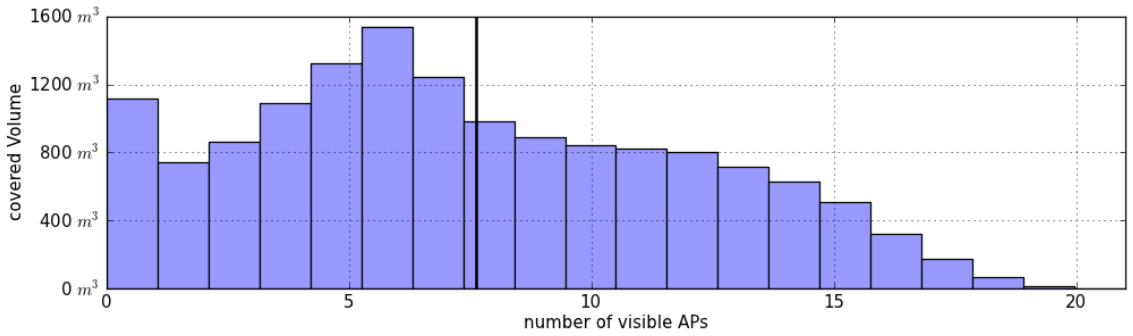

Figure 6.1 Access Point coverage of the building after placing the moveable APs for the training procedures. The histogram is extracted from raytracer simulations. The total volume of the modelled UMIC building is given by $8100m^3$. The average AP density at a location in space, from at most 22 APs, is given by 7.6 APs per voxel.

6.1.1.3 Devices

Multiple devices were evaluated by taking measurements at locations with different ratios of completeness with respect to 100 defined locations. Investigating the performance of the framework over more than one device leads to a better understanding about the generalization capabilities of the system. This means, it can be seen if the trained propagation model adapts only to device-specific and not to the environment-specific properties, which are the more interesting here.

The primary two Wi-Fi capable devices were an Acer Iconia Android tablet and a Nexus 4 smartphone. Some experiments were also conducted with the WISPY spectrum analyzer. The remaining two devices are again Android smartphones. One is a Galaxy P and one a Nexus 1. And the last device was an external TP-Link USB-Adapter connected to a linux driven netbook. As can be seen in the next table, the first two android devices have been analyzed more thoroughly.

Device	**Type**	**API**	N_{locs}	N_{aps}	N_{all}	N_{rssis}
Iconia	Tablet	Android 3.2	86	22	924	25715
Nexus	Phone	Android 4.0	94	22	919	20428
Galaxyp	Phone	Android 2.3	68	22	496	7644
Nexus1	Phone	Android 2.3	73	22	515	8055
Messbook	USB-Adapter	Libpcap	44	20	492	4478
Wispy	Spectrum Analyzer	Custom	19	3	41	41

The different numbers represent the following values: N_{locs} is the number of visited locations. N_{aps} the number of unique APs for which measurements have been collected. N_{all} is the number of (location, AP) tuples with measured RSSI values. Some combinations have no values due to their distance. And finally N_{rssi} is the total number of RSSI readings that were taken at for the N_{all} combinations.

The measurements for the Android devices were taken by interfacing with the OS-level APIs. For the linux netbook a Python wrapper over libpcap was used.

41 special measurements over three APs of the asus AP class, have been taken with the WISPY spectrum analyzer. The spectrum analyzer measures signals from the APs at a lower layer that the packet-aware Wi-Fi-chipsets. It reports the energy level of the electromagnetic field for a frequency vector with a fine grained time-sampling. But this means also, that the AP source of the signal is not known. The primary use-case of the device consists of finding the Wi-Fi channels that are less used by the different consumers of the electromagnetic spectrum. Switching APs to unused channels can increase the service quality of the Wi-Fi infrastructure.

The problem of AP identification with the WISPY was solved by customizing the firmware of three asus-APs. The modification allows the AP to send a burst signal sequentially over all 14 Wi-Fi channels. The underlying burst signal can be identified easily on the Wi-Fi data stream. The main drawback of this technique has been materialized in the efforts that are needed to measure a single AP RSSI value. The values of the different APs need to be recorded serially instead of synchronously as in the case of the Android devices. For a single measurement, around 3 minutes were needed with all preparations. To reach the 924 AP readings of the Iconia device, 50 hours of measurements would have been required. Therefore, the simpler strategy, to rely only on the reported readings from the Wi-Fi-APIs of the different operating systems, was chosen.

6.1.1.4 Training Corpus

The training corpus for the raytracer simulated radio propagation models was build by measuring the RSSI readings for the 22 APs at 100 different locations. This relates to one location for $81m^3$ or a cube with edge size of around $4m$. The locations are evenly distributed over the three modelled floors and the measurements were taken about $1m$ above each floor level.

The averaged standard deviation σ_{all} for measured RSSI values over N_{all} different location/AP tuples is in the range of $2 - 4dBm$. The wispy device was excluded as there were not enough measurements to lead to significant estimates. The class of the 9 EDUROAM APs has a slightly better variance with $\sigma_{edu} = 2.5dBm$ over their $N_{edu} = 1067$ readings, than the other three AP classes. These APs consist of CISCO models with a larger antenna array for MIMO support, and they are not targeted at the consumer market. Another possible explanation for the lower variance of these models can originate in a slightly adapted measurement technique. The CISCO APs emit RSSI values for multiple software simulated SSIDs. By introducing SSID aliases for these virtual APs all of them have been grouped together. This grouping can probably also lead to the recognized lower variance. The details for the different device/AP-class combinations are listed in table 6.2.

For each location, around 670 readings were taken for each AP over the Android devices. This relates to a measurement duration of 17 minutes with a resolution of 1.5 seconds per RSSI reading. The effect of the body shadow was minimized by choosing a neutral position with respect to the placements of the APs. The measurements were mostly taken during the office hours, so there was also some sporadic traffic of moving people in the surroundings. It can be concluded, that under the trade-off: effort-versus-precision, the effort was reduced. So it is sensible

Device	N_{all}	σ_{all}	N_{asus}	σ_{asus}	N_{edu}	σ_{edu}	N_{wrt}	σ_{wrt}	N_{pisa}	σ_{pisa}
Iconia	924	3.3	367	3.5	271	2.6	72	4.3	214	3.2
Nexus	919	3.8	327	4.3	282	3.2	90	4.0	220	3.7
Galaxyp	496	2.1	166	2.2	152	2.1	58	1.8	120	2.0
Nexus1	515	2.0	177	2.1	156	1.7	62	2.2	120	2.0
Messbook	492	2.6	247	2.6	206	2.1	12	7.7	27	4.6
Total/Avg	3346	3.0	1284	3.3	1067	2.5	294	3.2	701	2.9

Table 6.2 The training corpus for the radio propagation model is composed of measurements from different devices.

to assume that more elaborate measurement techniques lead to more precise readings with less variance.

Most readings were taken by using the OS-level Wi-Fi-APIs of the Android and Linux devices. These readings were measured in RSSI with minimal readings of around $-10dbm$ and a maximum of $-100dbm$

6.1.2 Training of free Parameters

The training of the free parameters for the PHOTON propagation model was conducted by using the process described in 4.2.2. It is reasonable to assume, that an optimal trained radio propagation model, a high quality representation of the real world, leads to lower error rates for localization algorithms relying on this information source. Therefore, the first part of the evaluation investigates how many prerequisites in form of a 3D-geometry and how much training data in form of manual measurements is needed to derive a good propagation result. The optimization target of the Genetic Algorithm is given by the RPE. The quality of the propagation model is assumed to be proportional to the reached RPE in the optimization process which is therefore used for evaluation as well.

For the following evaluations, a default optimization parameter set is used that is defined in the following table:

PHOTON Parameters		**Genetic Parameters**	
Name	**Value**	**Name**	**Value**
Numphotons	500000	Startpop	80
Density in m	0.3	Childcount	80
AP Power in W	10^{-5}	Childcull	30
Resolution	(320, 99, 58)	Mutprob	0.15
BBox in m	(-1.0, 60.0), (-1.0, 18.0),(-4.0, 7.1)	Mutamt	0.15

Table 6.3 The default optimization parameter set.

This parameter set contains the PHOTON configuration and the initialization values of the genetic algorithm. Their effects are briefly explained starting with the *PHOTON Parameters*:

A total of *Numphotons*=$5 \cdot 10^5$ photons are used during raytracing, which has been shown to give a nice balance between accuracy and computing time. The *Density* parameter represents the variance of the Gaussian, used for the final smoothing step. The *AP Power* value of 10^{-5} indicates, that the genetic algorithm searches over the range $[10^{-6}..10^{-4}]$ for the antenna gain PHOTON parameters of the 4 different AP power groups. The *Resolution* in combination with the *Bounding Box* determines the size and the voxelization ratio of the scene. With the given numbers, a voxel has a homogeneous edge length of $20cm$.

The *Genetic Parameters* define the initial size of the population with the *Startpop* parameter. 80 organisms lead to 1760 GPU-node driven jobs for the 22 APs of the UMIC scene. In each generation *Childcount*=80 new organisms are created by randomly chosen parents. The fittest *Childcull*=30 children determine the population of the next generation. The *Mutprob* and the *Mutamt* parameters control, how often mutations happen (*Mutprob*) and how strong the induced variability is (*Mutamt*). These chosen values have been found to lead to good convergence for the UMIC scenario under the constraints of 50 GPU nodes and around 10 hours cluster usage for each experiment.

6.1.2.1 Granularity of the 3D geometry

An interesting evaluation can be conducted by choosing different levels of granularity for the 3D-geometry that is used in the raytracer. Lesser needed detail for the 3D-geometry is translated in lesser model acquisition costs, which is therefore worth investigating. For this evaluation, only the measurements of the Iconia device are used to minimize eventual device specific distortions. The default parameter set for the optimizer was used and combined with different 3D-geometries. The *Basic* geometry consists of the mesh triangles for the *Concrete* and *LightWalls* materials that are listed in table 6.1. The 3D geometry for these 2 material classes is probably the cheapest to obtain, as the basic information is available in 2D form for most buildings. From this 2D map, the composition of a 3D scene can be approximated, which has for example been done in the work of El-Kafrawy [8]. Under the assumption, that the doors are also available on an eventual 2D map, the *Basic+Doors* setup was chosen. The *Full* setup represents the geometry with all materials of table 6.1. In the following table, the optimization results in form of the RPE are given:

Mesh	**Basic1**	**Full1**	**Basic2**	**Full2**	**Basic+Doors5**	**Full11**
Materials	1	1	2	2	5	11
RPE in *dBm*	5.7	5.8	4.4	4.1	4.2	4.3

It can bee seen in the first and the second column, that the reduction to one free material parameter leads to worse convergence of the training process.
After distinguishing between the two materials, *Concrete* and *LightWalls* in the Basic2 scenario and *Concrete* with the combined material *Other* in Full2, the RPE drops significantly. The 5-material setup remains in the same RPE range as does the full-detail model of the last column. In the corresponding section 6.2.3.3 for the localization evaluation, it is investigated whether these results have an impact on the performance of the localization algorithms.

6.1.2.2 Multiple Devices

In this experimental setup, three propagation models were trained by using measurements from the training corpus for different device combinations. The first propagation model was only trained on the Iconia tablet, the second model includes the measurements of the Nexus smartphone and the last model uses measurements from all four Android based devices. The reached RPE under the parameter optimization are reported in the following table:

Device	**Iconia**	**Iconia/Nexus**	**Iconia/Nexus/Galaxyp/Nexus1**
RPE in dBm	4.3	3.6	5.0

Although the combination of the four devices in the last column leads to a high RPE of around $5dBm$, the theory, that more employed devices lead to worse propagation models is contradicted by the first two results. The combination of the Iconia/Nexus measurements exhibits an even better converging optimization process than the Iconia-only case. It also shows satisfying results with respect to the evaluation of the localization algorithms. Such an evaluation is conducted for the Iconia/Nexus model in the following section 6.2.3.2.

6.2 Localization

This section presents the performance evaluation of the three presented localization algorithms. The HMM and the PF are compared with respect to the baseline given through the LMSE approach. After describing the properties of the evaluation site and the different evaluated localization paths, the algorithms are initially tested on synthetic data. This will allow to compare them under idealized conditions with respect to errors originating in the radio propagation model. The required artificial sequences of RSSI vectors are therefore obtained from the SSMs generated by the PHOTON raytracer.

After the synthetic evaluation, the performance of the localization framework on real world data will be analysed. Experiments with two different devices classes, a tablet and a smartphone, were conducted. The real world experiments are composed of multiple localizations paths, that were manually tracked and annotated to establish the evaluation corpus for the localization problem.

6.2.1 Scene and Setup

The evaluation was conducted in the UMIC building, described in 6.1.1, over two android devices, the Iconia tablet and a Galaxy Nexus smartphone. Eight paths that lead through the building with various degrees of complexities were defined and subsequently used for taking the measurement streams. Each path has a forward and a backward variant which effectively doubles the number of tracked paths to sixteen. The tables 6.4 and 6.5 show the different properties of the defined paths in detail.

Path	Length	Turns	avg. Signals	avg. APs	Figure
eg-room-change	40.0m	12	30.5	8.9	A.1
og1-classic	36.3m	8	40.6	12.1	A.2
og1-eg	59.8m	11	65.4	6.6	A.3
og1-eg-right	69.2m	13	62.3	10.4	A.4
og1-long-rooms	81.2m	16	93.3	7.3	A.5
og1-long-straight	41.9m	7	40.6	8.4	A.6
og1-room-change	14.3m	9	43.2	9.9	A.7
stairs-upward	47.6m	11	66.7	4.4	A.8

Table 6.4 Physical properties of the defined paths in the first two columns and AP coverage extracted from the evaluation corpus.

The *avg. Signals* column of the table 6.4 represents the number of these readings. The *avg. APs* column indicates how many APs are received in average for a single reading. From the *avg. Time* column of table 6.5 can be concluded, that successive RSSI reading are received in approximately $1Hz$ which was empirically determined to represent an appropriate pulse frequency.

Path	avg. Time	avg. Speed	Iconia(F/B)	Nexus(F/B)
og1-long-straight	38s	1.1m/s	11/11	12/13
og1-eg-right	69s	1.0m/s	13/13	11/11
stairs-upward	51s	0.9m/s	17/14	10/10
og1-classic	39s	0.9m/s	13/10	16/15
eg-room-change	52s	0.8m/s	10/10	10/10
og1-eg	60s	1.0m/s	11/11	10/10
og1-room-change	32s	0.4m/s	12/12	10/10
og1-long-rooms	86s	0.9m/s	11/10	12/12

Table 6.5 Time and speed related properties of the defined paths of the evaluation corpus. And the size of the evaluation corpus with respect to the employed devices.

As can be seen in the columns *Iconia(F/B)* and *Nexus(F/B)* of table 6.5, at least 10 evaluation samples were collected for the forward (F) and the backward (B) variant of each defined path. Such an evaluation sample is a sequence of location annotated RSSI readings. The measurements were recorded at different sessions mostly during office hours and no special care was taken to ensure perfectly constant environment conditions. This means, that sometimes doors, which probably effect the radio propagation, were either open or closed. In some path samples, people are either crossing the way or occupying radio propagation relevant places. Therefore, the evaluation corpus can be assumed to represent an appropriate representation of live conditions which lead to random noise artefacts in the RSSI readings.

The synthetic and the real world evaluation use radio propagation models that are trained and optimized according to 6.1.2. For the evaluation of synthetic data, a model that was only trained on Iconia measurements is used. The same model is also employed in the real world evaluation. Additionally, the real world evaluation

uses an Iconia/Nexus trained model to investigate whether the localization system generalizes over multiple devices. The optimization result of these two models is reported in 6.1.2.2. Furthermore, the distinction between the different devices is only made during the real world evaluation.

Offline/Online Results

For both evaluation setups, the three algorithms are compared. The PF and the HMM approach return *Offline* and *Online* results. Both adapt to RSSI information contained in the history due to their algorithmic nature. The offline result is the predicted sequence of locations under the knowledge of the *full* history. The online result is the predicted sequence of locations under the knowledge of the limited history up to a time frame. The latter represents the result that is available for a user of a localization service who expects to be informed about his current locations, and not about the way he came.

Therefore, it is reasonable to distinguish between the online and the offline result and it can be expected, that the offline variant outperforms the online result due to its larger knowledge pool. The LMSE does not use the history of the RSSI values, therefore it reports only an online result.

The online error of the PF and the HMM is given in an additional averaged form. This form is defined by using the mean location over the top 50 candidate hypothesis of a time frame t instead of choosing the best hypothesized location.

6.2.2 Synthetic Measurements

In this section, the performance of the three localization algorithms is evaluated on synthetically generated RSSI sequences used to simulate readings of a mobile device with different noise conditions. For each of the 16 path variants 20 samples were generated therefore leading to 320 evaluated path samples. From the lengths of the paths given in table 6.4 can be derived that around $8000m$ of covered distance are evaluated. The average movement speed is assumed to be $1m/s$ reflecting the properties of the real world measurements that are concluded from table 6.5. For every $0.75s$, an RSSI vector is generated that represents the target frequency of the RSSI pushing android devices[4].

The synthetic measurements were generated by using the RSSI values of a trained radio propagation model as the means of Gaussians with a variable noise σ. From these Gaussians the measurements are sampled for $0dBm \leq \sigma \leq 18dBm$ with a step size of $0.5dBm$. The results for the algorithmic variants under online and offline conditions, for a localization space with $40cm$ voxel size, can be obtained from figure 6.2 and table 6.6.

It can be seen, that the LMSE based algorithm leads to incompetitive error rates over all noise conditions with respect to the HMM and the PF approach. This was expected since it does not use the valuable information contained in the measurement

[4] Although the target frequency was not completely reached in the real world measurements due to network latency over the HTTP communication channel.

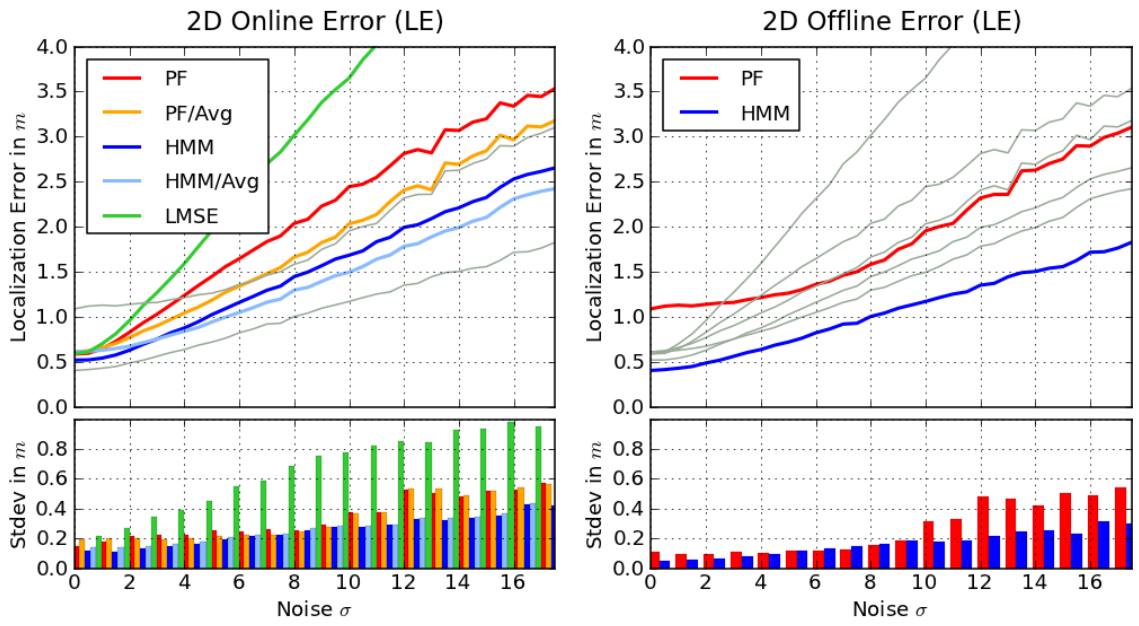

Figure 6.2 Online and offline 2D localization errors over 36 noise steps. The HMM outperforms the PF during the online and offline mode. The *averaged* online result leads to lower errors for both algorithms.

history. Interestingly, it is even slower than the PF, since *all* states of the location space have to be visited at each time frame. And this is costly for a high resolution state space.

The HMM algorithm leads to the lowest error rates over all noise conditions. Even both online errors are lower than the PF offline error. This comes by the costs of increased computational complexity by the factor 10 and an increase in memory consumption of around factor 100. It can be concluded that the HMM approach reaches lower error rates: By conducting an exhaustive search over the hypotheses the HMM approach is guaranteed to find the optimal solution, whereas the PF approach is not to able to generate this solution by sampling.

Furthermore, both algorithms show the behaviour that the averaged variant of the online error leads to a better prediction of the hidden location sequence but are outperformed by the offline error in both cases. An interpretation of this result is, that valuable information is distributed over either the particles or the unpruned states of the HMM. The backtracked solutions in the offline mode can make sense of this information whereas during the online mode it can only partially be accessed by the averaging process.

Additionally, it can be observed from the variance plot in figure 6.2, that the PF, compared to the HMM, shows an elevated decrease in accuracy under increasing noise at the interval $8dBm < \sigma < 14dBm$. By investigating the individual result sequences it was observed that at these noise levels the effect of sample impoverishment (see 4.3.2.3) becomes significant and seems not to be handled optimal. It is only countered with random re-sampling which induces a new noise component and therefore leads to a degraded result. A better way to handle the decay of the set of particles in the PF algorithm is proposed in the work of Widyawan [29]. The basic idea is to conserve the particle history, manipulate it by using information from the

Online and Offline LEs in m for Noise: $\sigma = 14dBm$ (forward+backward)							
Path	HMM/off	HMM	HMM/avg	LMSE	PF/off	PF	PF/avg
eg-room-change	1.76	2.34	2.08	4.97	2.51	3.00	2.51
og1-classic	1.09	1.66	1.41	3.60	2.11	2.46	2.04
og1-eg	1.60	2.44	2.23	5.00	3.04	3.54	3.22
og1-eg-right	1.51	2.27	2.04	5.38	3.03	3.46	3.15
og1-long-rooms	1.83	2.53	2.33	6.05	3.07	3.67	3.28
og1-long-straight	1.38	2.07	1.87	4.20	2.14	2.61	2.28
og1-room-change	1.63	2.05	1.84	3.61	2.45	2.74	2.34
stairs-upward	1.20	2.31	2.11	5.98	2.69	3.09	2.76
Mean in m	1.50	2.21	1.99	4.85	2.63	3.07	2.70
Stdev in m	0.25	0.32	0.34	0.91	0.39	0.44	0.47

Table 6.6 Online and offline localization errors under $\sigma = 14dBm$. The differently defined paths show significant differences in their error rates.

degraded set and restart the sampling at some point in the modified history. But the analysis of this approach is reserved for future experiments and has not been covered in this thesis.

Another interesting result is derived from table 6.6. The defined paths, that represents different localization scenarios, show significantlys different error rates. These difference can have the following causes: Either the path is covered with more APs which leads to more valuable emission probabilities. Or the path is complex with respect to the number of turns or direction changes which is an unmeasurable event for the RSSI only sensors. The same argument holds for an eventual stopping of the movement which was the case in the *eg-room-change*, *og1-long-rooms*, and the *og1-room-change* scenario. It can be assumed, that a more elaborate transition model, which would probably depend on additional sensors, leads to a better adaptation to variable movement conditions.

A more detailed overview of the results on synthetic data is given in the appendix A.3. There, the results for 5 different noise levels are listed with additional forward/backward path separation. From comparing the forward and the backward case, it can be seen, that they show a significant difference in some cases. A probable explanation for this observation can be derived from the sensibility to the starting conditions of the HMM and PF algorithm. If the starting point of the path lies in an area with less AP coverage, the initial location hypotheses are distributed widespread, which can, for example lead to early sample impoverishment in the PF case. Another reason originates in the handling of the sequential nature of the tracking problem. The HMM/PF algorithms are able to retain accuracy in zones with less AP coverage by depending on the history of the signals. At an uninformative starting point such valuable history is not available.

It will be interesting to see if these results are also applicable to the case of real world measurements which are inspected in the next section.

6.2.3 Real World Measurements

For the evaluation of the algorithms under reals world conditions, the evaluation corpus, described in table 6.5, that is composed of the defined paths in table 6.4 are used. The scene and the methods for obtaining the corpus were carefully described in section 6.2.1.

The first experiment is conducted on the evaluation samples from the Iconia device. 160 samples were taken from the evaluation corpus. This was done by selecting 10 samples for each defined path. This relates to an evaluated distance of around $4000m$. Furthermore, an optimized radio propagation model is used, trained only on the measurements for the Iconia device.

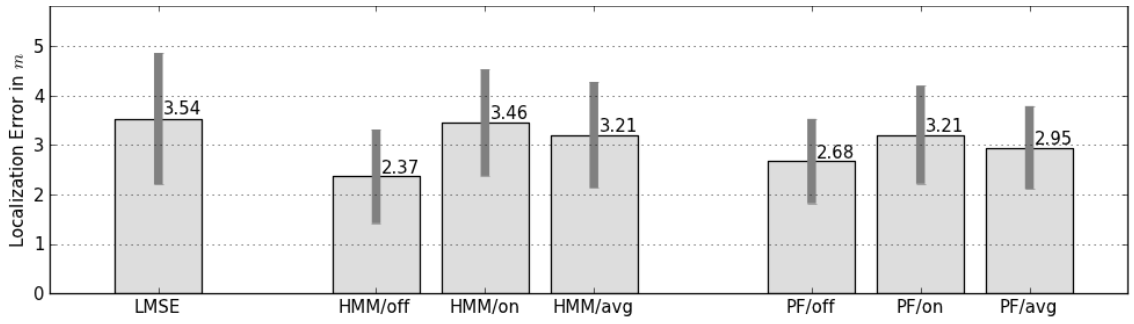

Figure 6.3 Localization errors for the different algorithms on the real world measurements from the Iconia device.

As can be seen in figure 6.3, the ranking of the errors for the different approaches are comparable to the results of the synthetics evaluation (figure 6.2) although the online variants of the HMM show higher errors than their PF counterparts. The LMSE approach is outperformed by the PF, which is less accurate in the offline variant of the error than the HMM. The reduction of the error rates of the PF and HMM with respect to the LMSE approach are less satisfying than under the synthetic conditions. Under synthetic conditions with a noise of $14dBm$ as in table 6.6, the LMSE is outperformed by a factor nearly of 2 by the HMM and PF algorithm. Under the real world conditions of this setting, the margin is reduced to around 20%. From the details of the results for the different paths separated in table 6.7 can be seen, that some of the defined paths are predicted quite inaccurately. This observation can also be derived from the elevated standard deviation of $45cm$ and upwards, whereas the compared synthetic result show an upper limit of $45cm$ (without the LMSE).

Especially the *stairs-upward* path shows an unsatisfying offline error of around $4m$ for the HMM/PF algorithm. Whereas under synthetic condition with $14dBm$ noise, it is only around $2.5m$. This can probably be explained by the relatively low AP coverage of around 4 APs as reported in table 6.4. Another outlier is given by the *og-eg* path. Removing both paths from the corpus reduces the average error rates over all algorithms by nearly $40cm$.

Therefore, it can be concluded, that the absolute error rates reported in this evaluation are very scene specific. As such, they cannot be used for a meaningful comparison with the systems found in literature.

Path	HMM/off	HMM	HMM/avg	LMSE	PF/off	PF	PF/avg
eg-room-change	1.71	2.91	2.66	2.31	1.56	2.25	2.16
og1-classic	1.46	2.85	2.60	2.65	2.13	2.59	2.46
og1-eg	3.04	4.35	4.13	4.21	3.18	3.89	3.59
og1-eg-right	1.85	2.83	2.52	3.19	2.62	2.90	2.56
og1-long-rooms	2.02	3.54	3.40	3.86	2.82	3.08	2.93
og1-long-straight	2.94	4.00	3.84	3.32	2.62	3.28	3.32
og1-room-change	2.58	2.66	2.36	2.59	2.54	2.71	2.54
stairs-upward	3.17	3.49	3.22	5.82	3.50	4.35	3.58
stairs-upward-r	3.55	5.53	5.08	6.48	4.42	5.59	4.43
Mean in m	2.37	3.46	3.21	3.54	2.68	3.21	2.95
Stdev in m	0.68	0.87	0.87	1.17	0.69	0.85	0.66

Table 6.7 Detailed errors for the different paths from the evaluation corpus. The aggregation of the results can be seen in figure 6.3. The *stairs-upward-r* path, that is addionally shown in the forward/backward variant, is a drastic negative outlier, probably due to complex nature of the stairway scene and the suboptimal placements of APs.

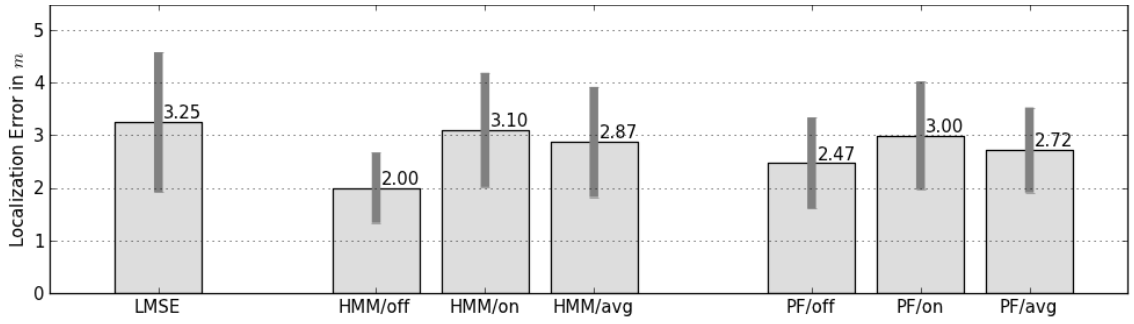

Figure 6.4 Localization error over all algorithms for the Iconia device after applying the Device Adaptation technique. The error rates for the PF and the LMSE algorithm drop by around $20cm$.

6.2.3.1 Device Adaptation

The evaluation is continued by investigating the Device Adaptation method described in the design chapter 4.2.3. This method assumes, that the reported RSSI values from the device are distorted with respect to some unknown function. The configuration of this function can be learned from the radio propagation models if location annotated measurements are available for the individual model of the device or for the more general device class.

Such a function was trained for the previously described and evaluated Iconia setup. As the mentioned location annotated measurements, the training corpus for the radio propagation model was used. The comparison of the results in figure 6.4 with the previous results in figure 6.3 reveals that the procedure has a measurable effect on the error rates. For the PF and the LMSE algorithm, the reduction is about $20cm$. The error rates of the HMM approach are reduced by about $40cm$. Not surprisingly,

Path	HMM/off	HMM	HMM/avg	LMSE	PF/off	PF	PF/avg
eg-room-change	1.61	2.50	2.27	2.24	1.41	2.04	1.99
og1-classic	1.44	2.30	2.11	2.31	1.86	2.25	2.10
og1-eg	2.23	3.87	3.70	3.77	3.03	3.62	3.33
og1-eg-right	2.06	2.73	2.48	3.05	2.88	3.02	2.74
og1-long-rooms	1.80	3.03	2.92	3.45	2.46	2.86	2.64
og1-long-straight	1.94	3.48	3.37	2.77	2.21	2.89	2.90
og1-room-change	1.87	2.13	1.81	2.35	2.13	2.37	2.14
stairs-upward	3.07	4.75	4.31	6.02	3.78	4.92	3.90
Mean in m	2.00	3.10	2.87	3.24	2.47	3.00	2.72
Stdev in m	0.48	0.94	0.91	1.18	0.73	0.90	0.66

Table 6.8 Localization error at the individual path level. The forward and backward variants of the paths have been combined.

the relative differences between the defined path scenarios remain the same as can be seen by comparing table 6.8 and the results without the adaptation step in table 6.7.

The observed positive result, the effect of the adaptation function can either be interpreted as a better understanding of the true nature of the propagation model or it can originate from a device specific property. In the former case, it can be understood as an additional training step which follows the basis training described in 6.1.2. But the latter case, the interpretation of the effect as an unknown device specific property, was the initial assumption that had motivated the approach during this thesis.

The main problem of this technique is the need for device and location annotated RSSI measurements. This problem can be approached by investigating whether the needed information can be obtained by using localization results as a ”trustworthy” source. If this ”trustworthyness” is measurable, only results with a high accuracy have to be fed back into the system which would probably lead to a better performance.

6.2.3.2 Multiple Devices

In this part of the evaluation, the performance of the system over two devices is investigated. In the first setting, a radio propagation model is used that was trained with measurements of both devices. For each device, 160 paths from the evaluation corpus are evaluted. The results for the HMM and the PF algorithm are shown in figure 6.5.

The average error of the predictions for the Nexus device is comparable to the Iconia device although the latter performs slightly better. The variability of the results for the Nexus is visibly higher than that of the Iconia device. An evidence for this result can be obtained by inspecting the radio propagation training corpus. As can be seen in table 6.2, the standard deviation of the measurements for the Nexus is significantly larger than the standard deviation of the Iconia device. A cause for

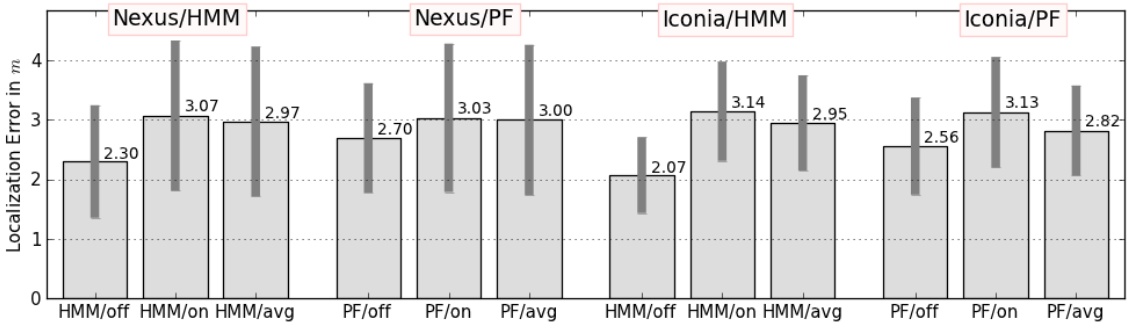

Figure 6.5 The two devices, the Iconia tablet and the Nexus smartphone, show comparable error rates on a radio propagation model that is trained with measurements from both.

this behaviour can be found by remembering the device classes of both models. The Iconia is a tablet device with a ten inch display, whereas the Nexus is of the smaller smartphone device class. Due to the size of its body, a possible interpretation is, that the Iconia tablet is equipped with a larger antenna which leads to smother RSSI readings.

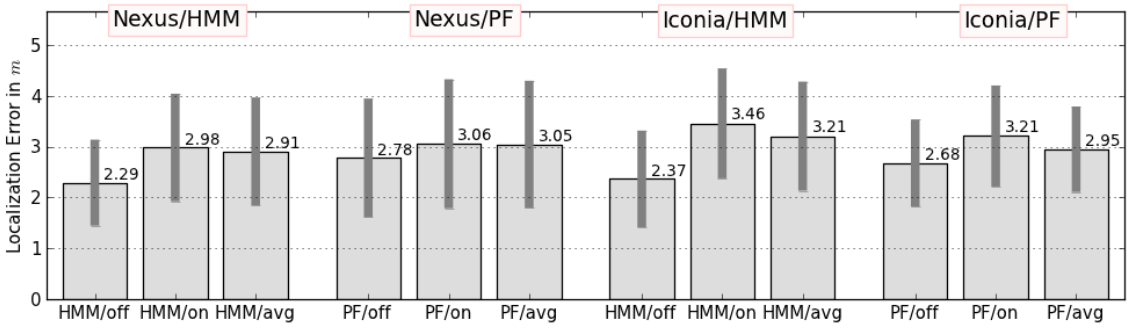

Figure 6.6 Interestingly, the Nexus smartphone outperforms the Iconia tablet, with respect to the localization accuracy, on a radio propagation model, that is only trained on the latter.

In the last part of the evaluation it is investigated whether the assumption holds, that the radio propagation model generalizes over the two used devices. Therefore, an experiment with both devices is conducted on a radio propagation model that is only trained with measurements of the Iconia device. The results of the employed model for the Iconia were already shown in figure 6.3. Using the model also for the Nexus device leads to the results of figure 6.6.

Comparing the results of both devices leads to the conclusion, that the Iconia trained radio propagation model generalizes well for the Nexus specific localization problem. The reported error rates for the Nexus are even lower for most algorithmic combinations which has not been expected.

The next step of the evaluation is conducted by adding an additional Device Adaptation step as described in the previous section 6.2.3.1. Figure 6.7 reports the results of this procedure.

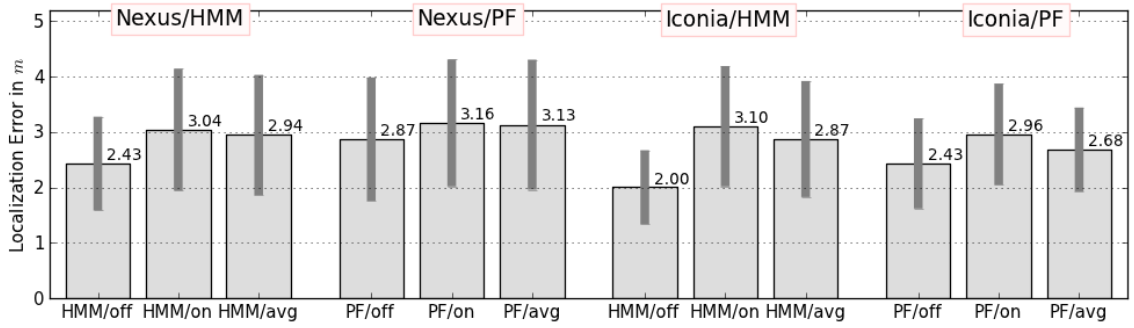

Figure 6.7 The Device Adaptation shows no effect on the error rates of the Nexus.

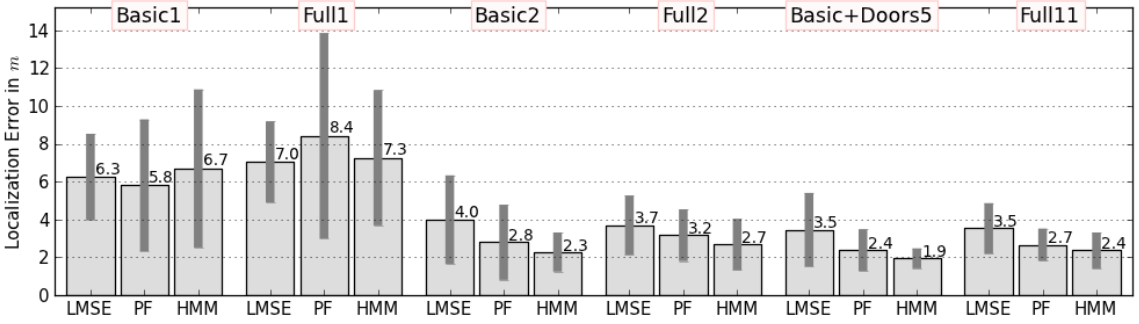

Figure 6.8 Localization errors for radio propagation models trained on material annotated 3D geometry of different complexities. From left to right, the geometry becomes more complex. It can be observed that using the most complex geometry of *Full11* leads unexpectedly to worse results than using the less complex *Basic+Doors5*.

Under the Iconia trained propagation model, the Device Adaptation for the Nexus shows no significant enhancement. The results become even marginally worse. Under the assumption, that the Device Adaptation is a form of final training for radio propagation model, this result makes no sense. The measurements of the Nexus device should have been a valuable information to adapt to the less ”familiar” propagation model as it has only be trained on the Iconia device.

6.2.3.3 Granularity of the 3D geometry

In the following experiment, the impact of the accuracy of the 3D geometry on the localization error is analysed. Six radio propagation models were trained on 3D geometry that represents the UMIC environment in different levels of complexity. The details of the six models are described in the former evaluation of these models in section 6.1.2.1.

160 path samples originating from the Iconia device were selected from the evaluation corpus, as including the samples from the Nexus can lead to undesired device specific distortions in the outcome of the experiment. The samples are evaluated with all three localization algorithms which leads to results that are listed in figure 6.8.

Unexpectedly, the single material variants *Basic1* and *Full1* perform worse than the other variants. This underperformance was also visible in the reported RPE

of section 6.1.2.1. It can therefore be concluded, that modelling the environment with one material leads to an insufficient propagation model with respect to the localization problem. Even increasing the complexity of the scene geometry does not leads to improvements. On the contrary, the *Full1* propagation model performs less accurate than the simpler *Basic1* model.

The *Basic2* propagation model performs very good with regard to its simple mesh structure. Only the concrete and light walls are employed in the 3D geometry which is a relatively easy to obtain information about a building. The model leads to better error rates over all algorithms compared to the *Full2* model. The *Full2* can be understood as an upgraded variant of the geometry from *Basic2*. They share the scene objects composed of concrete material, but the *Full2* combines *all* remaining available triangles to the secondary material, whereas the *Basic2* only uses the light walls for that. Concluding from the error rates, this seems to be a bad approach.

It is an interesting result, that the *Basic+Doors5* geometry outperforms the *Full11* model on the HMM/PF algorithms quite significantly and leads to the overall best results. The additional knowledge that is included in the most complex model *Full11* seems not to be an exploitable information source for the raytracer under the setup of this thesis.

6.3 Summary

From the overall results presented in this chapter, it can be summarized that the designed framework is able to solve the localization problem with acceptable accuracy for PHOTON driven radio propagation models. The proposed method of training the free model parameters with genetic algorithms leads to material and AP specific coefficients that are usable to generate RSSI estimates that diverge only about $4dBm$ from the true reality represented by the training corpus. By further device specific fine tuning with the presented Device Adaptation scheme, the divergence can be reduced to $3.5dBm$.

The analysis of the performance of the three different localization algorithms has brought up the conclusion that the HMM approach outperforms the PF approach over most scenarios. On synthetic data, the gap between the error rates of both algorithms is more significant than under real world conditions. On an assumed noise level of $14dBm$, the offline error of the HMM approach is given by a promising $1.5m$ whereas the Particle Filter reaches only an offline accuracy of $2.6m$. As was expected from its algorithmic properties, the simple LMSE approach leads not to competitive results on synthetic data, as it reports a localization error of around $5m$.

The relations between the error rates of the algorithms shift quite significantly if the evaluation is conducted on real world measurements. The HMM and the Particle Filter algorithm perform now very similar although the HMM shows a visible advantage under offline conditions. The average error rates over all defined paths are in the range of $2-3m$ for most evaluated scenarios. It was observable, that the error rates between paths of different complexity show a high variability. Especially the stairway path can only be inferred with an accuracy of less than $4m$. Removing this path and another outlier shifts the offline/online error rates down to $1.5m/2.0m$ over the remaining six easier scenarios.

The proposed Device Adaptation technique shows a promising increase of accuracy for the Iconia device but only inconclusive results for the Nexus. The localization experiments, that were conducted over both devices, lead to the conclusion that the trained propagation models are able to generalize over multiple devices. The system reports similar error rates if the propagation model is trained with either only measurements from the Iconia or from the combined measurements of both devices.

The final evaluation step, the investigation of how much granularity of the 3D geometry is needed for the PHOTON generated propagation models that are subsequently used for estimating the localization errors, lead to an interesting result. The best performing radio propagation variant is driven by a 3D geometry only consisting of concrete material, light walls and three different types of doors (iron, glass, normal). The full detail model that contains additional elements like windows and furniture is somewhat overspecified for the PHOTON $\rightarrow$ localization algorithm pipeline. But this leads also to the more positive result, that sufficient detailed geometry can probably be generated automatically from rasterized 2D floor maps, since both types of walls and the location of doors can be obtained from them in most cases.

7

Conclusion

The motivation of this thesis, the analysis of state-of-the-art localization algorithms based on the prediction of Wi-Fi signal distributions has lead to the design of a localization framework providing a combined controll of the raytracing system, the localization system and the evaluation tools. The implemented system was employed to build a representative evaluation corpus for the localization problem and a training corpus for the Wi-Fi prediction problem. The evaluation of the prediction capabilities of the used PHOTON raytracer and the accuracy of the localization algorithm over both data sets has brought up the following conclusions:

The achievable prediction accuracy of the radio propagation models is about $4dBm$ compared to real world measurements by investigating only the basic components, i.e. reflection and attenuation, of the PHOTON raytracer and using 3D scene geometry that distinguishes at least between two material types. This result has been observed over 22 deployed APs on the investigated $8000m^3$ UMIC scene. Furthermore, the propagation models were able to generalize over two device classes. This means, the localization accuracy was not affected by using two different types of client devices and without a specific device adaptation process although they provided significant different RSSI measurements.

The primary goal of this work, the evaluation of different localization algorithms on a complex scene and a large evaluation corpus under the constraints of the accuracy induced by the raytracer model, has been reached. The evaluation of the HMM, PF and LMSE localization algorithms has shown, that the PHOTON driven models can be used for predictions with an accuracy in the range of $2-3m$ under the constraints of the described environment settings.

The evaluation of the location accuracy has shown that the RSSI based system has a large variance for the localization error. The primary source for the variability in the observed accuracy is the constitution of the collected evaluation corpus. For example, some paths have complex sections that lead along stairways with only a sparse AP coverage. On the simpler pathway zones of the building, that are well covered by about 6-8 APs, the system can reach localization error rates of between

$1 - 2m$, whereas a localization in the difficult zones lead to significant errors of $4m$ for the PF and of $6m$ for the HMM algorithm.

Du to the history-awareness of the PF and the HMM algorithms, the reported error rates are distinguished in the online and the offline variant. The online error relates to the localization result that is available under knowledge about the preceding measurements, and the offline error relates to the prediction of the location under the knowledge about all following measurements as well. This distinction has not been made in the reported literature, although this is a relevant information as both error variants differ by about $0.4m$ accuracy. Whereas the use-case of the online variant represents a singular localization request under live conditions, the offline variant is, for example, more valuable for analysing the movement patterns for a group of people. The offline prediction is therefore useful to *learn* from the environment, whereas the online variant is for use-cases like user navigation.

With respect to the real world evaluation results, the HMM algorithm outperforms the PF based approach on the same available knowledge about the environment under offline conditions, whereas under online conditions, the PF leads to slightly better estimates. Furthermore, the PF approach is less computational demanding than the HMM approach which, for example, makes it more portable to smaller and less powerfull devices.

The presented localization framework has shown to be a viable platform to efficiently evaluate the localization algorithms on the information derived from the raytracing generated radio propagation models. The training procedure, devised for the unknown raytracer material parameters in form of a genetic algorithm leads to well behaving RSSI predictions that are usable as a foundation for the localization algorithms.

7.1 Future Work

This section presents a brief collection of future research topics that were identified during this thesis with respect to the two main topics radio propagation models and Wi-Fi localization algorithms.

In order to achieve more accurate propagation models using raytracing generated RSSI predictions the following open topics remain:

- The variance of the RSSI predictions, if modelled by Gaussian as in this thesis, originates partly in the multipath effects of the radio propagation behaviour. The PHOTON raytracing algorithms can be configured to report the signal distribution for different recursion levels of its tracing algorithms. It would be interesting if this can be interpretable as the described multipath effect and therefore can be used as a further source of information for the localization algorithms in terms of signal variance for the individual locations.

- Another feature of the raytracer are antenna patterns that are able provide a more realistic signal distribution in contrast to the simple isotropic model that was used for this thesis. The additional free parameters of the antenna

model can also be trained with the proposed optimization procedure of this framework.

- Furthermore, it would be interesting to analyse the effect of the AP placement on the error rates. More APs lead to better performance of the system, but is there a reachable optimum if the number of APs is limited? This optimal placement of the APs can probably also be found automatically.

The next topics relate to the localization problem and can help to further increase the accuracy of the presented algorithms:

- Lifting the constraints on the knowledge pool and fusing additional sensor input into the RSSI measurement stream is probably the most promising way to increase the location accuracy. And, as modern smartphones are more and more equipped with sensors like gyroscopes, accelerometers and compasses there exists a rich set of options. It should pose not much effort to adapt the HMM and PF algorithms to these information sources.

- The presented HMM model can be extended to process more of the measurement history by introducing higher order models. On current hardware, at least second order models are computable, and will probable lead to a more adaptable recognition systems. For example, this will enable the possibility to model direction changes using the transition probabilities.

- The PF algorithm can be extended to handle the degradation of the particle set under bad prediction conditions with a more elaborate technique than the currently implemented random resampling. A backtracking based proposal found in the related literature to the topic seems to be suited for this problem.

Bibliography

[1] *Deterministic Propagation Model for the Planning of Hybrid Urban and Indoor Scenarios* (2005), vol. 1. http://dx.doi.org/10.1109/PIMRC.2005.1651518.

[2] Bahl, P., and Padmanabhan, V. N. Radar: An in-building RF-based user location and tracking system. In *INFOCOM (2)* (2000), pp. 775–784.

[3] Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D., and Smith, K. Cython: The best of both worlds. *Computing in Science Engineering 13*, 2 (2011), 31 –39.

[4] Bishop, C. M. *Pattern recognition and machine learning*, 1st ed. 2006. corr. 2nd printing ed. Springer, Oct. 2006.

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

[6] Chen, Y. C., Chiang, J. R., Chu, H. H., Huang, P., and Tsui, A. W. Sensor-assisted wi-fi indoor location system for adapting to environmental dynamics. In *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems* (New York, NY, USA, 2005), MSWiM '05, ACM, pp. 118–125. http://dx.doi.org/10.1145/1089444.1089466.

[7] Doucet, A., and Johansen, A. M. A tutorial on particle filtering and smoothing: fifteen years later. *The Oxford Handbook of Nonlinear Filtering* (Dec. 2009), 4–6.

[8] El-Kafrawy, K., Youssef, M., El-Keyi, A., and Naguib, A. Propagation modeling for accurate indoor WLAN RSS-based localization. In *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd* (2010), IEEE, pp. 1–5. http://dx.doi.org/10.1109/VETECF.2010.5594108.

[9] Eleryan, A., Elsabagh, M., and Youssef, M. Aroma: Automatic generation of radio maps for localization systems. *CoRR*.

[10] Figueiras, J., and Frattasi, S. *Mobile Positioning and Tracking*. John Wiley & Sons, 2010.

[11] Hassan-Ali, M., and Pahlavan, K. A new statistical model for site-specific indoor radio propagation prediction based on geometric optics and geometric probability. *Wireless Communications, IEEE Transactions on 1*, 1 (2002), 112–124.

[12] Hatami, A., and Pahlavan, K. 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* (Jan. 2006), vol. 2, IEEE, pp. 1018–1022. http://dx.doi.org/10.1109/CCNC.2006.1593192.

[13] Ji, Y., Biaz, S., Pandey, S., and Agrawal, P. Ariadne: a dynamic indoor signal map construction and localization system. In *Proceedings of the 4th international conference on Mobile systems, applications and services* (New York, NY, USA, 2006), MobiSys '06, ACM, pp. 151–164. http://doi.acm.org/10.1145/1134680.1134697.

[14] Julier, S. J., and Uhlmann, J. K. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE 92*, 3 (Mar. 2004), 401–422. http://dx.doi.org/10.1109/JPROC.2003.823141.

[15] Klepal, M. *Novel Approach to Indoor Electromagnetic Wave Propagation Modelling*. PhD thesis, Czech Technical University, 2003.

[16] Nicoli, M. B. Hmm-based tracking of moving terminals in dense multipath indoorenvironments. *Eusipco 2005* (2010).

[17] Pandey, S., Kim, B., Anjum, F., and Agrawal, F. Client assisted location data acquisition scheme for secure enterprise wireless networks. vol. 2, pp. 1174–1179 Vol. 2.

[18] Prasithsangaree, P., Krishnamurthy, P., and Chrysanthis, P. On indoor position location with wireless LANs. vol. 2, pp. 720–724 vol.2.

[19] Pugh, W. Skip lists: a probabilistic alternative to balanced trees. *Commun. ACM 33*, 6 (June 1990), 668–676. http://dx.doi.org/10.1145/78973.78977.

[20] Richalot, E., Bonilla, M., Wong, M.-F., Fouad-Hanna, V., Baudrand, H., and Wiart, J. Electromagnetic propagation into reinforced-concrete walls. *Microwave Theory and Techniques, IEEE Transactions on 48*, 3 (Mar. 2000), 357–366. http://dx.doi.org/10.1109/22.826834.

[21] Roweis, S., and Ghahramani, Z. A unifying review of linear gaussian models, 1999.

[22] Schmitz, A., Karolski, T., and Kobbelt, L. Using spherical harmonics for modeling antenna patterns. IEEE Radio and Wireless Symposium.

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

[24] Schmitz, A., and Wenig, M. The effect of the radio wave propagation model in mobile ad hoc networks. In *MSWiM '06: Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems* (New York, NY, USA, 2006), ACM Press, pp. 61–67. http://dx.doi.org/10.1145/1164717.1164730.

[25] Seitz, J. V. A hidden markov model for pedestrian navigation. *Positioning Navigation and Communication*, 7th Workshop (Mar. 2010), 120–127.

[26] Valenzuela, R. A ray tracing approach to predicting indoor wireless transmission. In *Vehicular Technology Conference, 1993 IEEE 43rd* (May 1993), IEEE, pp. 214–218. http://dx.doi.org/10.1109/VETEC.1993.507047.

[27] Wallbaum, M., and Wasch, T. Markov localization of wireless local area network clients. In *WONS* (2004), vol. 2928 of *Lecture Notes in Computer Science*, Springer, pp. 1–15.

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

[29] Widyawan, Klepal, M., and Beauregard, S. A backtracking particle filter for fusing building plans with PDR displacement estimates. 207–212. http://dx.doi.org/10.1109/WPNC.2008.4510376.

[30] Wilson, R. Propagation losses through common building materials 2.4 GHz vs 5 GHz., Aug. 2002.

[31] Woodman, O., and Harle, R. Pedestrian localisation for indoor environments. In *Proceedings of the 10th international conference on Ubiquitous computing* (New York, NY, USA, 2008), UbiComp '08, ACM, pp. 114–123. http://dx.doi.org/10.1145/1409635.1409651.

[32] Youssef, M., Mah, M., and Agrawala, A. Challenges: device-free passive localization for wireless environments. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking* (New York, NY, USA, 2007), ACM, pp. 222–229. http://dx.doi.org/10.1145/1287853.1287880.

A

Appendix

A.1 List of Abbreviations

AOA	Angle Of Arrival
AP	Access Point
API	Application Programming Interface
BRDF	Bidirectional Reflectance Distribution Function
CPU	Central Processing Unit
GPS	Global Positioning System
GPU	Graphics Processing Unit
GSM	Global System for Mobile Communications
HMM	Hidden Markov Model
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
LMSE	Least Mean Squared Error
LOS	Line Of Sight
LDS	Linear Dynamic System
LE	Localization Error
NLOS	None Line Of Sight
OpenGL	Open Graphics Library
OpenMP	Open Multi-Processing
PF	Particle Filter
PDF	Probability Density Function
RPE	Radio Propagation Error
RSSI	Received Signal Strength Indication
SSID	Service Set IDentification
SSM	Signal Strength Map
TDOA	Time Difference Of Arrival
TOA	Time Of Arrival
UTD	Uniform Theory of Diffraction
WAF	Wall Attenuation Factor

A.2 Localization Paths

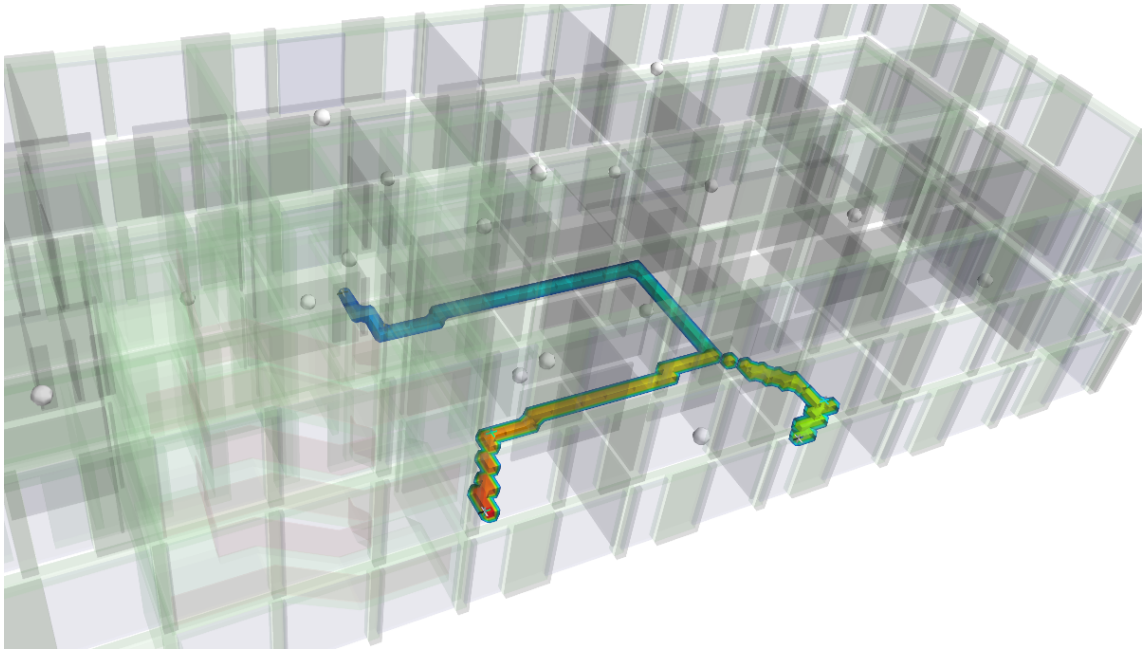

Figure A.1 The path *eg-room-change* starts in a room in the lower floor and visits 2 other rooms on the same level. In the middle room (greenish) the localization target holds its position for around 5 seconds. The path has a length of $40m$ and is covered by an average of 9 APs.

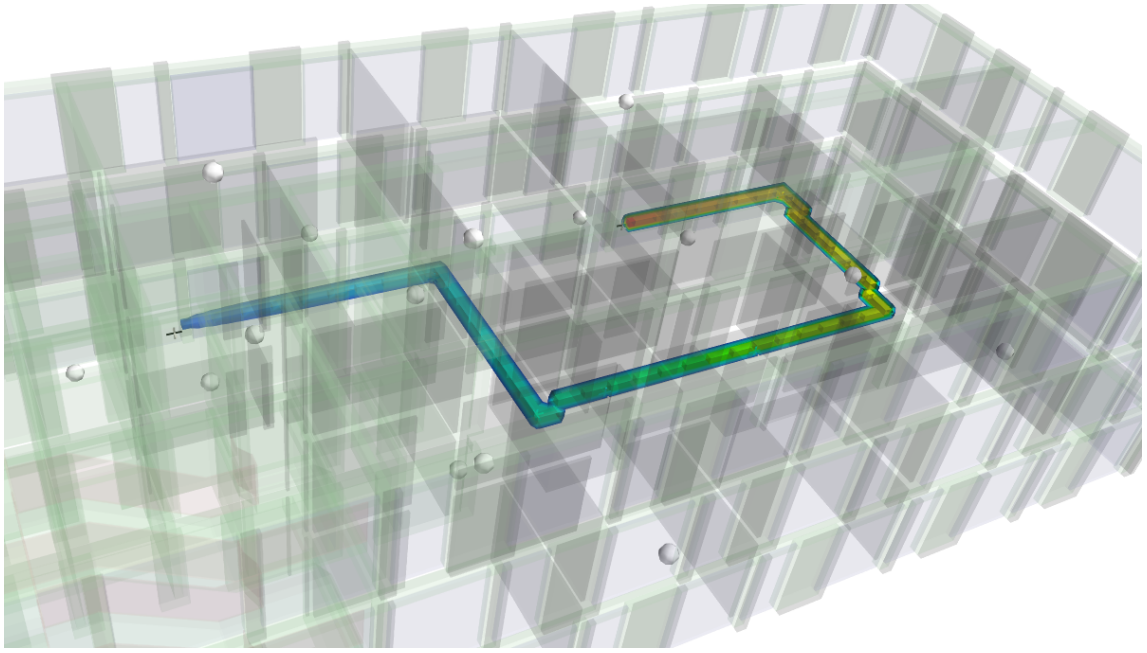

Figure A.2 The path *og1-classic* start in a room on the right side of the first level. It leaves the room and visits another room by circling around the core room (Hardware Pool) of the level. The path has an length of $36m$ and is covered by 12 APs.

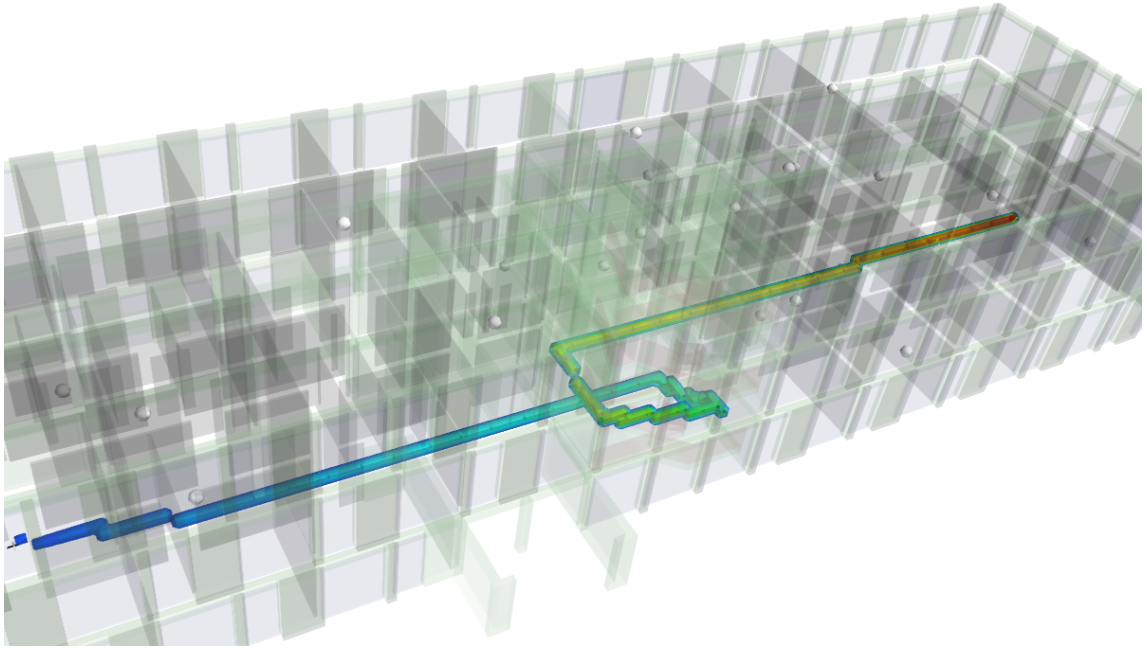

Figure A.3 The path *og1-eg* starts deep in the left side of the ground level and leads by the stairways to the right side of the first level of the building. The path has distance of $60m$ and is covered by an average of 7 APs over all positions.

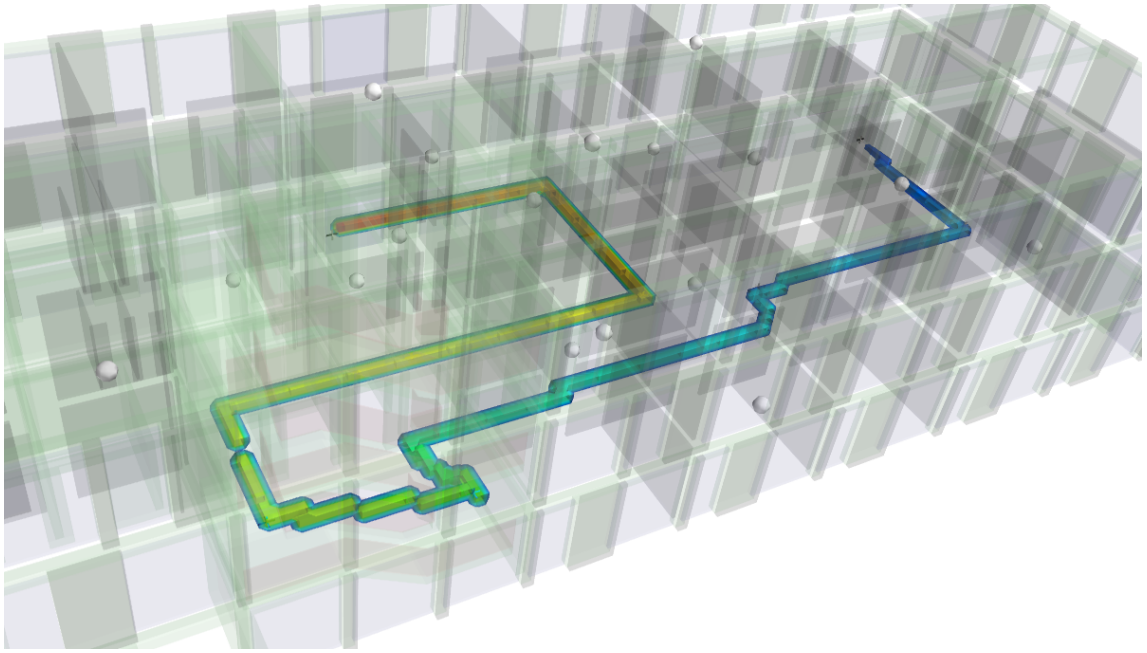

Figure A.4 The path *og1-eg-right* starts in the same room as *og1-classic*. It leads from the first level to the right side of the ground level by passig the stairways. The total covered distance is $70m$ and an average of 10 APs are seen during the transition.

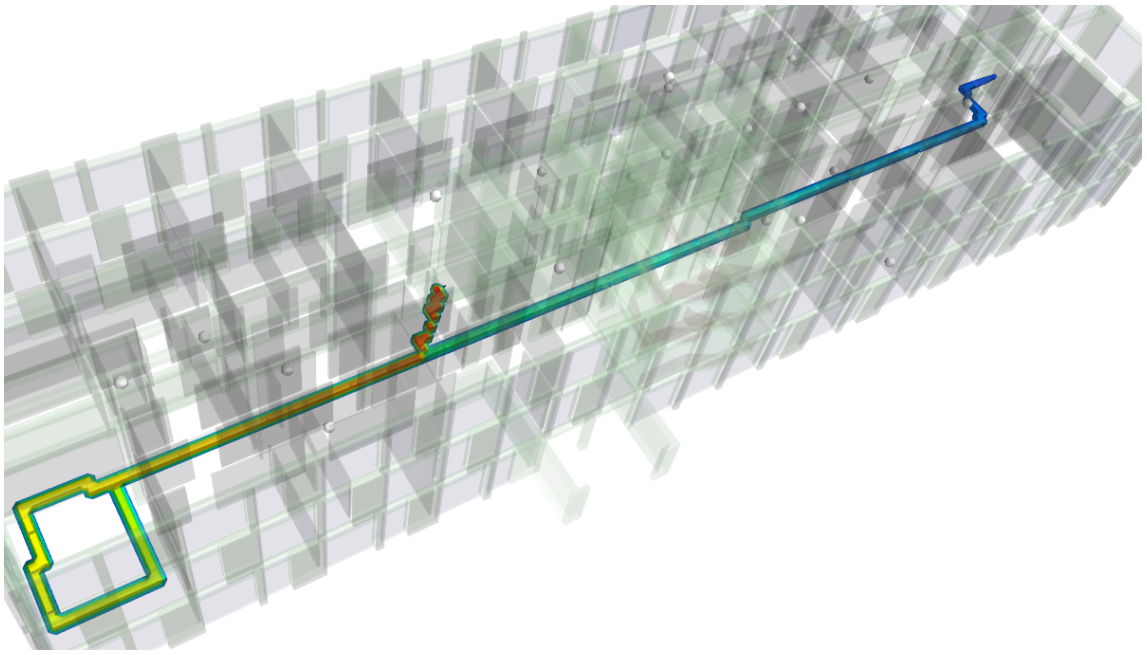

Figure A.5 the path *og1-long-rooms* starts at the right side of the first level and leads deep into the left side of the same level. There, a room is covered and the path leads back again into some sort of kitchen zone. In that zone, the mobile device rests for around 5 seconds. The path has a length of $81m$ and is covered by an average of 7 APs over all positions.

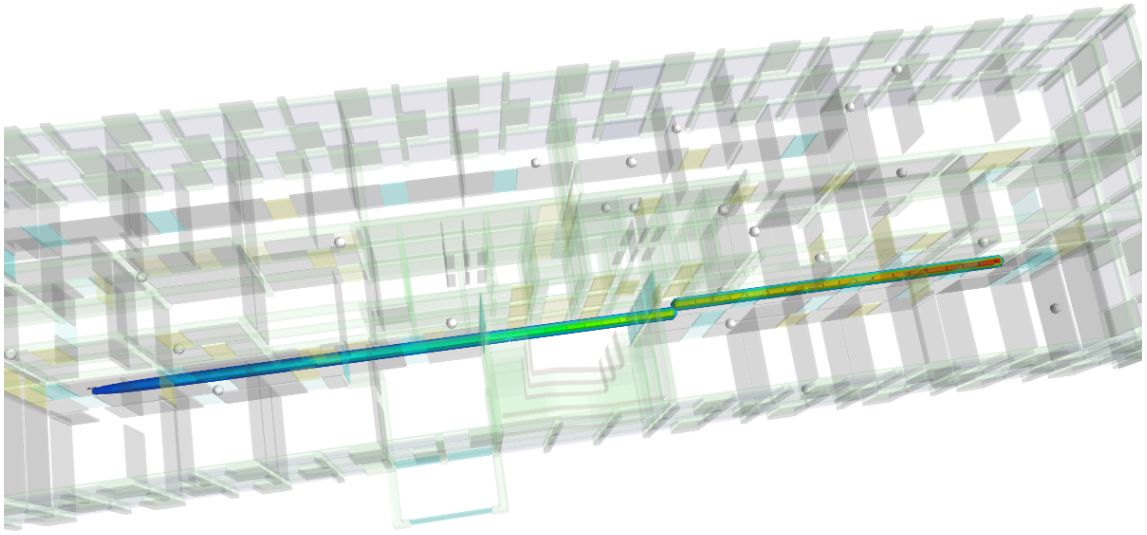

Figure A.6 The path *og1-long-straight* forms a straight line from one side to the other side of the building and remains on one level. The path is $42m$ long and is covered by an average of 8 APs.

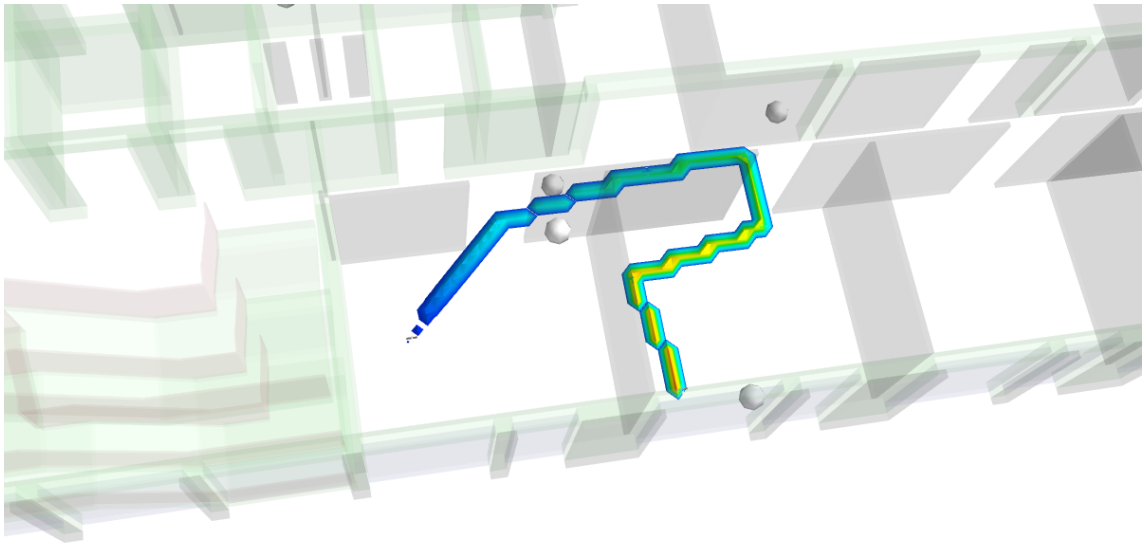

Figure A.7 The path *og1-room-changes* represents a short room transtions. At the start and the end of the path, the device rests around 5 seconds and does not move. The scene is covered by 10 APs and the path has a length of $14m$.

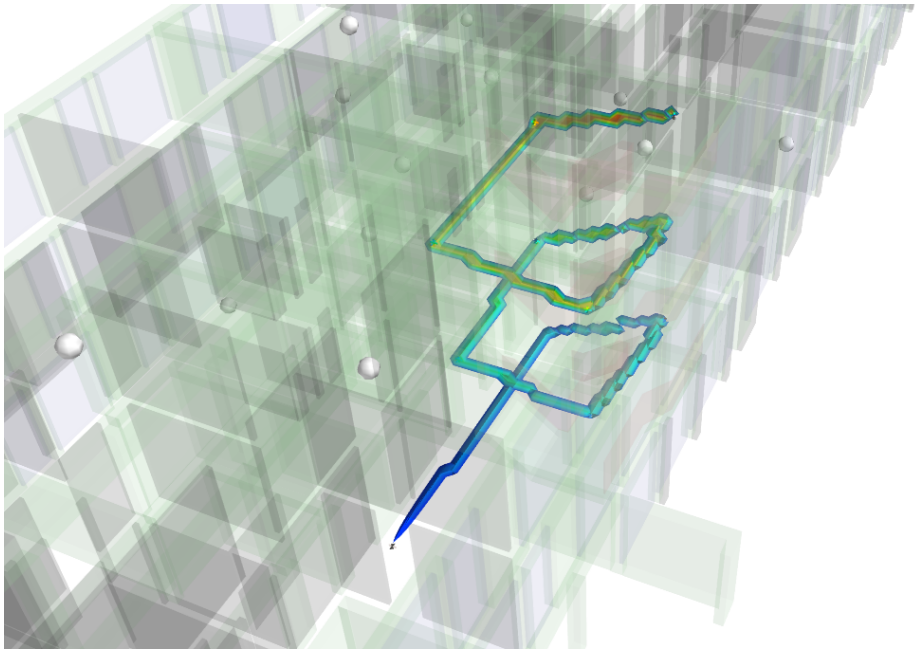

Figure A.8 The path *og1-stairs-upward* starts at the ground level and uses the stairways to relocate to the second level. It has a length of around $50m$ and is covered by an average of 4 APs over all positions.

A.3 Synthetic Localization Error Tables

Online and Offline LEs in m for Noise: $\sigma = 0dBm$ (forward+backward)							
Path	HMM/off	HMM	HMM/avg	LMSE	PF/off	PF	PF/avg
eg-room-change	1.78	2.30	2.04	4.88	2.43	2.93	2.40
eg-room-change-r	1.75	2.38	2.13	5.07	2.60	3.07	2.62
og1-classic	1.11	1.79	1.58	3.65	2.26	2.76	2.36
og1-classic-r	1.07	1.52	1.25	3.54	1.95	2.16	1.73
og1-eg	1.49	2.31	2.08	4.98	3.29	3.72	3.36
og1-eg-r	1.70	2.56	2.38	5.03	2.79	3.37	3.09
og1-eg-right	1.44	2.17	1.97	5.34	3.08	3.54	3.25
og1-eg-right-r	1.59	2.36	2.11	5.43	2.98	3.39	3.05
og1-long-rooms	1.86	2.50	2.31	6.19	3.12	3.68	3.25
og1-long-rooms-r	1.80	2.56	2.36	5.91	3.01	3.66	3.32
og1-long-straight	1.29	2.06	1.89	4.29	2.13	2.60	2.30
og1-long-straight-r	1.47	2.08	1.85	4.10	2.16	2.62	2.26
og1-room-change	1.60	2.06	1.82	3.63	2.51	2.73	2.35
og1-room-change-r	1.67	2.04	1.86	3.58	2.39	2.75	2.33
stairs-upward	1.17	1.80	1.58	5.92	2.62	2.96	2.56
stairs-upward-r	1.22	2.81	2.63	6.03	2.76	3.21	2.95
Mean in m	1.50	2.21	1.99	4.85	2.63	3.07	2.70
Stdev in m	0.25	0.32	0.34	0.91	0.39	0.44	0.47

Online and Offline LEs in m for Noise: $\sigma = 4dBm$ (forward+backward)							
Path	HMM/off	HMM	HMM/avg	LMSE	PF/off	PF	PF/avg
eg-room-change	1.78	2.30	2.04	4.88	2.43	2.93	2.40
eg-room-change-r	1.75	2.38	2.13	5.07	2.60	3.07	2.62
og1-classic	1.11	1.79	1.58	3.65	2.26	2.76	2.36
og1-classic-r	1.07	1.52	1.25	3.54	1.95	2.16	1.73
og1-eg	1.49	2.31	2.08	4.98	3.29	3.72	3.36
og1-eg-r	1.70	2.56	2.38	5.03	2.79	3.37	3.09
og1-eg-right	1.44	2.17	1.97	5.34	3.08	3.54	3.25
og1-eg-right-r	1.59	2.36	2.11	5.43	2.98	3.39	3.05
og1-long-rooms	1.86	2.50	2.31	6.19	3.12	3.68	3.25
og1-long-rooms-r	1.80	2.56	2.36	5.91	3.01	3.66	3.32
og1-long-straight	1.29	2.06	1.89	4.29	2.13	2.60	2.30
og1-long-straight-r	1.47	2.08	1.85	4.10	2.16	2.62	2.26
og1-room-change	1.60	2.06	1.82	3.63	2.51	2.73	2.35
og1-room-change-r	1.67	2.04	1.86	3.58	2.39	2.75	2.33
stairs-upward	1.17	1.80	1.58	5.92	2.62	2.96	2.56
stairs-upward-r	1.22	2.81	2.63	6.03	2.76	3.21	2.95
Mean in m	1.50	2.21	1.99	4.85	2.63	3.07	2.70
Stdev in m	0.25	0.32	0.34	0.91	0.39	0.44	0.47

Online and Offline LEs in m for Noise: $\sigma = 8dBm$ (forward+backward)							
Path	HMM/off	HMM	HMM/avg	LMSE	PF/off	PF	PF/avg
eg-room-change	1.78	2.30	2.04	4.88	2.43	2.93	2.40
eg-room-change-r	1.75	2.38	2.13	5.07	2.60	3.07	2.62
og1-classic	1.11	1.79	1.58	3.65	2.26	2.76	2.36
og1-classic-r	1.07	1.52	1.25	3.54	1.95	2.16	1.73
og1-eg	1.49	2.31	2.08	4.98	3.29	3.72	3.36
og1-eg-r	1.70	2.56	2.38	5.03	2.79	3.37	3.09
og1-eg-right	1.44	2.17	1.97	5.34	3.08	3.54	3.25
og1-eg-right-r	1.59	2.36	2.11	5.43	2.98	3.39	3.05
og1-long-rooms	1.86	2.50	2.31	6.19	3.12	3.68	3.25
og1-long-rooms-r	1.80	2.56	2.36	5.91	3.01	3.66	3.32
og1-long-straight	1.29	2.06	1.89	4.29	2.13	2.60	2.30
og1-long-straight-r	1.47	2.08	1.85	4.10	2.16	2.62	2.26
og1-room-change	1.60	2.06	1.82	3.63	2.51	2.73	2.35
og1-room-change-r	1.67	2.04	1.86	3.58	2.39	2.75	2.33
stairs-upward	1.17	1.80	1.58	5.92	2.62	2.96	2.56
stairs-upward-r	1.22	2.81	2.63	6.03	2.76	3.21	2.95
Mean in m	1.50	2.21	1.99	4.85	2.63	3.07	2.70
Stdev in m	0.25	0.32	0.34	0.91	0.39	0.44	0.47

Online and Offline LEs in m for Noise: $\sigma = 12dBm$ (forward+backward)							
Path	HMM/off	HMM	HMM/avg	LMSE	PF/off	PF	PF/avg
eg-room-change	1.78	2.30	2.04	4.88	2.43	2.93	2.40
eg-room-change-r	1.75	2.38	2.13	5.07	2.60	3.07	2.62
og1-classic	1.11	1.79	1.58	3.65	2.26	2.76	2.36
og1-classic-r	1.07	1.52	1.25	3.54	1.95	2.16	1.73
og1-eg	1.49	2.31	2.08	4.98	3.29	3.72	3.36
og1-eg-r	1.70	2.56	2.38	5.03	2.79	3.37	3.09
og1-eg-right	1.44	2.17	1.97	5.34	3.08	3.54	3.25
og1-eg-right-r	1.59	2.36	2.11	5.43	2.98	3.39	3.05
og1-long-rooms	1.86	2.50	2.31	6.19	3.12	3.68	3.25
og1-long-rooms-r	1.80	2.56	2.36	5.91	3.01	3.66	3.32
og1-long-straight	1.29	2.06	1.89	4.29	2.13	2.60	2.30
og1-long-straight-r	1.47	2.08	1.85	4.10	2.16	2.62	2.26
og1-room-change	1.60	2.06	1.82	3.63	2.51	2.73	2.35
og1-room-change-r	1.67	2.04	1.86	3.58	2.39	2.75	2.33
stairs-upward	1.17	1.80	1.58	5.92	2.62	2.96	2.56
stairs-upward-r	1.22	2.81	2.63	6.03	2.76	3.21	2.95
Mean in m	1.50	2.21	1.99	4.85	2.63	3.07	2.70
Stdev in m	0.25	0.32	0.34	0.91	0.39	0.44	0.47

Online and Offline LEs in m for Noise: $\sigma = 16dBm$ (forward+backward)							
Path	HMM/off	HMM	HMM/avg	LMSE	PF/off	PF	PF/avg
eg-room-change	1.78	2.30	2.04	4.88	2.43	2.93	2.40
eg-room-change-r	1.75	2.38	2.13	5.07	2.60	3.07	2.62
og1-classic	1.11	1.79	1.58	3.65	2.26	2.76	2.36
og1-classic-r	1.07	1.52	1.25	3.54	1.95	2.16	1.73
og1-eg	1.49	2.31	2.08	4.98	3.29	3.72	3.36
og1-eg-r	1.70	2.56	2.38	5.03	2.79	3.37	3.09
og1-eg-right	1.44	2.17	1.97	5.34	3.08	3.54	3.25
og1-eg-right-r	1.59	2.36	2.11	5.43	2.98	3.39	3.05
og1-long-rooms	1.86	2.50	2.31	6.19	3.12	3.68	3.25
og1-long-rooms-r	1.80	2.56	2.36	5.91	3.01	3.66	3.32
og1-long-straight	1.29	2.06	1.89	4.29	2.13	2.60	2.30
og1-long-straight-r	1.47	2.08	1.85	4.10	2.16	2.62	2.26
og1-room-change	1.60	2.06	1.82	3.63	2.51	2.73	2.35
og1-room-change-r	1.67	2.04	1.86	3.58	2.39	2.75	2.33
stairs-upward	1.17	1.80	1.58	5.92	2.62	2.96	2.56
stairs-upward-r	1.22	2.81	2.63	6.03	2.76	3.21	2.95
Mean in m	1.50	2.21	1.99	4.85	2.63	3.07	2.70
Stdev in m	0.25	0.32	0.34	0.91	0.39	0.44	0.47

A.4 2D Localization Result

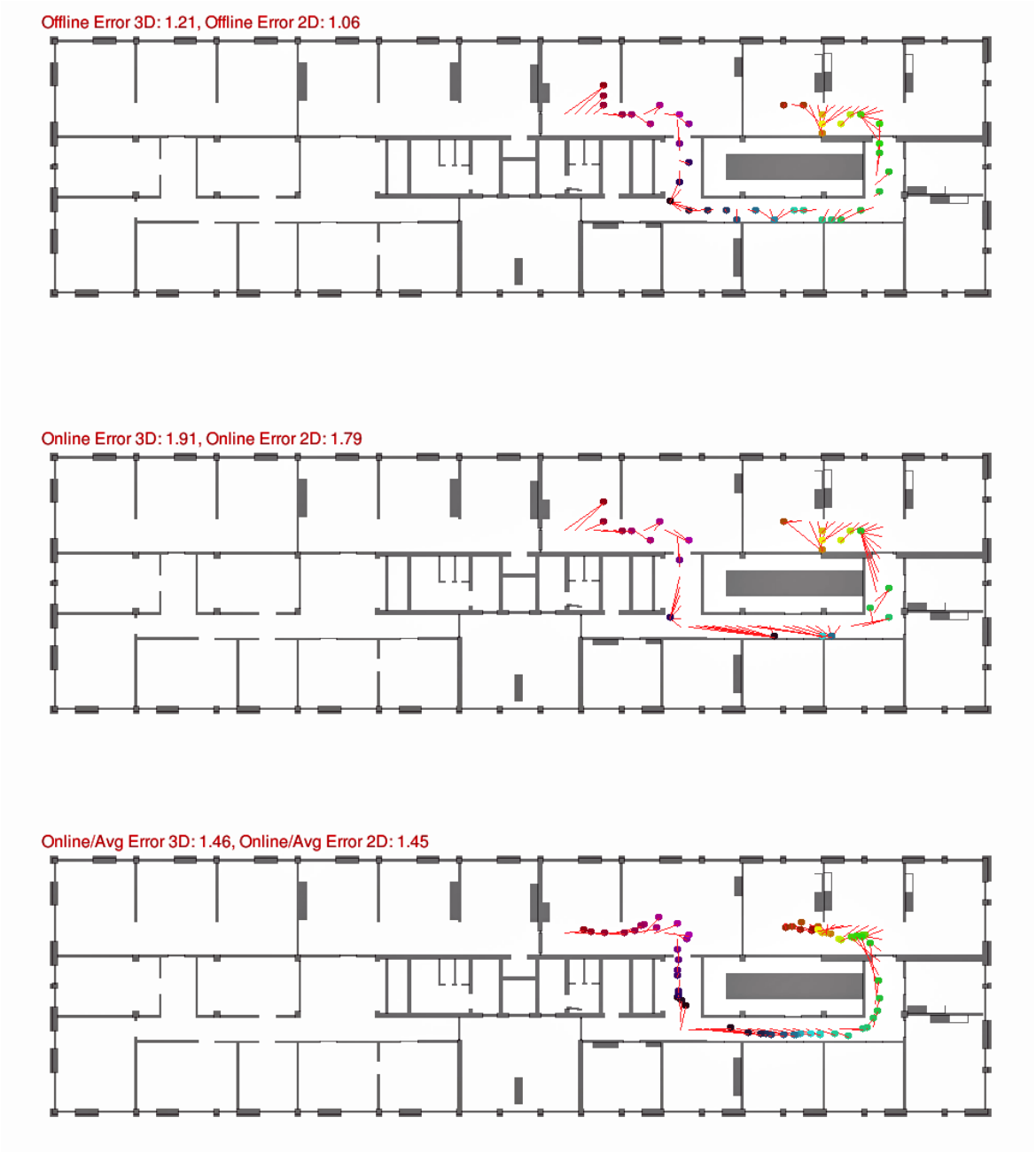

Figure A.9 Visualization of the HMM driven localization results for the path *og1-classic-r*. The first result is only available at the end of the path, whereas an online user would be more interested in last two results.

A.5 Optimization Process

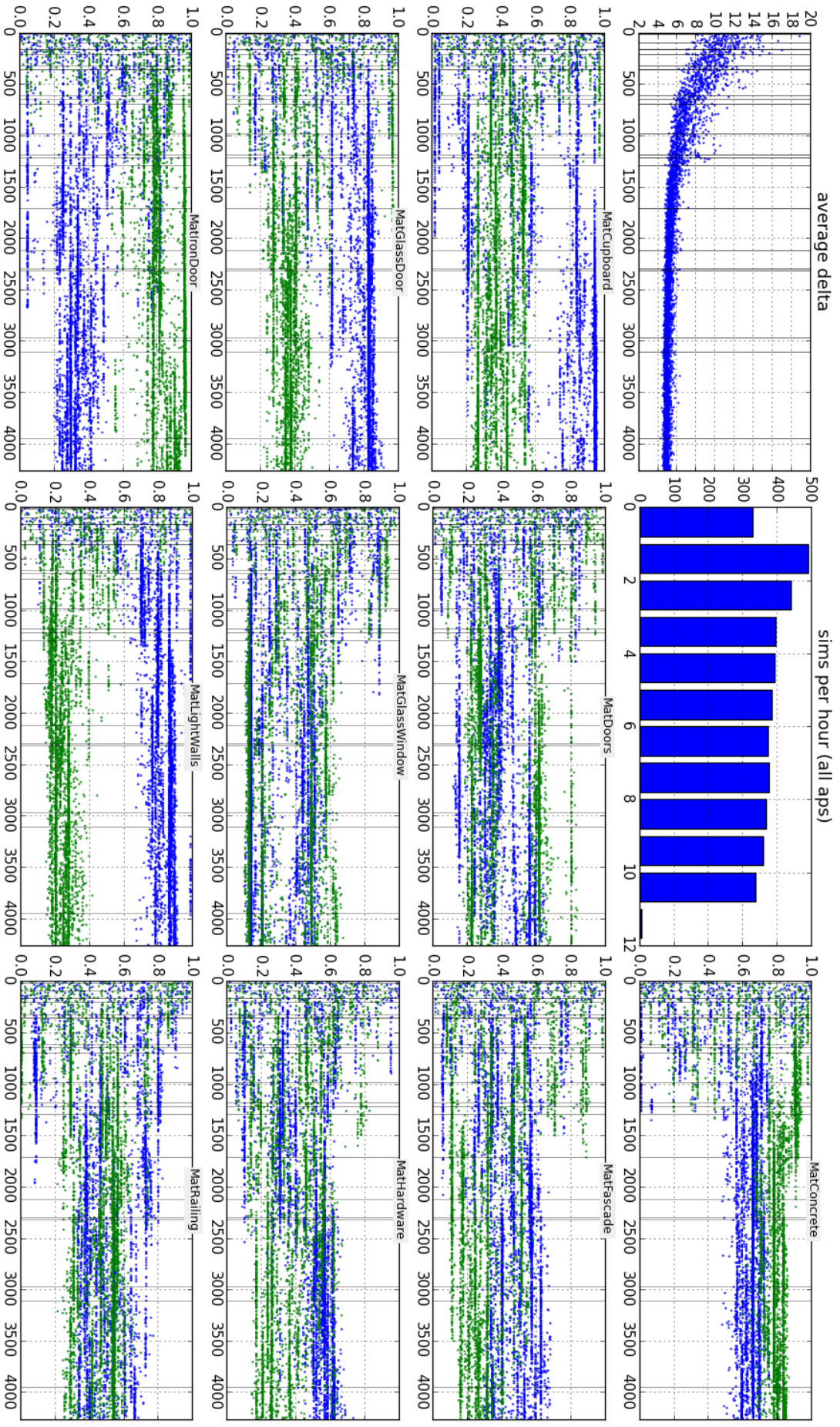

Figure A.10 Convergence of material parameters during an optimization run with genetic algorithms. The optimization target can be seen in the upper left corner, it is the average delta between the simulation results and the real world measurements.