LEHRSTUHL FÜR REALZEIT-COMPUTERSYSTEME
TECHNISCHE UNIVERSITÄT MÜNCHEN
UNIV.-PROF. DR. SC. S. CHAKRABORTY

Pedestrian Indoor Localization and Tracking using a Particle Filter combined with a learning Accessibility Map

Julian Straub

Bachelor Thesis

Pedestrian Indoor Localization and Tracking using a Particle Filter combined with a learning Accessibility Map

Bachelor Thesis

Executed at the Institute for Real-Time Computer Systems
Technische Universität München
Prof. Dr. sc. Samarjit Chakraborty

Advisor: Dipl.–Ing. Martin Schäfer

Author: Julian Straub
Otte-von-Stetzlingen Str. 24
86316 Friedberg

Submitted in August 2010

Acknowledgements

At first I want to thank Professor Chakraborty who enabled this thesis at the Institute for Real-Time Computer Systems (RCS) and all members of the RCS staff, who always were very supportive especially during the evaluation part. In particular my thank goes to my advisor Martin Schäfer. His good advice and the interesting discussions helped me very much to develop the Particle Filter and my ideas concerning the Accessibility Map. His fast responses to my questions and the fast proofreading of the thesis showed his understanding of my difficult time-constrained situation, which I am very grateful for.

Further more I want to thank my friends who always had a open ear when I needed to talk to someone during the stressful last period of the work. Especially I want to mention Stephan Herrmann, whom I could always discuss with about my ideas and who helped me to correct the English of this thesis. Last but not least I am very grateful for the help of Konrad Leibrandt with the design of the presentation.

Munich in July 2010

Table of Contents

List of Figures

List of Algorithms

List of Symbols

RCS	Institute for Real-Time Computer Systems
KLD	Kullback-Leibler Divergence
DOF	Degree Of Freedom
PDF	Probability Density Function
SIR	Sampling Importance Resampling
AM	Accessibility Map
SLAM	Self Localization and Map Building
ZUPT	Zero velocity Update

Abstract

As mobile phones are starting to get equipped with inertial sensors, indoor navigation for pedestrians becomes an increasingly interesting topic in research. This work aims to develop and evaluate the use of a Particle Filter to deal with noisy sensor measurements of an Inertial Measurement Unit (IMU) providing localization and tracking of a pedestrian in indoor environments. Designed at the Institute for Real-Time Computer Systems (RCS), the so called PiNav-System was used, which can extract the motion of a person from inertial sensor measurements. On this basis a Particle Filter was implemented, which uses Dead Reckoning in combination with a geometric floor plan to localize and track a person wearing the PiNav-System in a building. In addition the concept of the Accessibility Map (AM) is proposed which reflects human walking preferences in the degree of accessibility of space in a floor and which makes it possible to exploit this information in the Particle Filter. Reinterpreting the AM as a Radial Basis Function Network, a special type of Neural Network, a method for learning accessibility of space in a floor is derived. Measurements show that the additional use of the AM in the Particle Filter yields an improvement in the localization accuracy of up to 32%, resulting in an average accuracy of $1.1m$. Deploying the AM and the learning AM, also a more robust tracking is observed. Hence, besides the ability to monitor the walking patterns of a pedestrian in a building with a Particle Filter, the localization accuracy and the tracing robustness could be enhanced by the proposed AM.

Nachdem inertiale Sensoren zunehmend in Handys eingebaut werden, wird Navigation in Gebäuden zu einem immer interessanteren Forschungsgebiet. Diese Arbeit befasst sich mit der Entwicklung eines Partikel Filters zur Fußgänger-Lokalisierung in Gebäuden. Das am RCS entwickelte PiNav-System extrahiert die Bewegungen seines Trägers aus den Messdaten der eingebauten inertialen Sensoren. Diese Daten über Schrittlänge und Schrittrichtung verwendet der Partikel Filter in Verbindung mit einer Stockwerkskarte um die Position der Person zu schätzen. Außerdem wird die so genannte Accessibility Map (AM) vorgestellt, welche das durchschnittliche Fußgängerverhalten durch den Grad der Zugänglichkeit in einem Raum darstellt und für den Partikel Filter nutzbar macht. Eine Lernregel für die Zugänglichkeit von Raumbereichen wird durch die Uminterpretation der AM als ein Radial Basis Function Network (RBFN) hergeleitet. In der Lokalisierungsgenauigkeit war eine Verbesserung um bis zu 32% auf $1.1m$ messbar, wenn die AM zusätzlich zu dem Stockwerksgrundriss verwendet wurde. Es zeigt sich außerdem, dass der Einsatz der AM oder der lernenden AM eine robustere Positionsschätzung zur Folge hat. Insgesamt konnte, neben der Möglichkeit die Laufmuster von Personen aufzuzeichnen, mit dem entwickelten System aus Partikel Filter und lernender AM die Lokalisierungsgenauigkeit verbessert und die Pfadschätzung robuster gemacht werden.

1 Indoor Navigation

Today outdoor navigation can be thought of as solved by the General Positioning System(GPS). A GPS receiver can determine its position evaluating the timestamps and the position send by at least four satellites [26]. Most of us make use of this technology in their everyday lives. For example GPS receivers installed in nearly every car provide easy navigation. Even in sports GPS is used to track position and speed of athletes providing feedback on the sportive performance. Public transportation via airplanes or ships almost exclusively relies on the Global Positioning System to find safe trajectories to their destination. The main downside of this system is that localization fails in buildings, since it is impossible to receive the satellites' signals due to severe attenuation by walls. This is especially a problem when thinking of large indoor areas like airports or art museums, where similar to outdoor environments, navigation might be necessary to find a target position.

Due to the lack of inexpensive and reliable sensors, indoor localization remains unsolved and interesting for research. Basically there are two different approaches which are able to provide indoor localization and are in the focus of the developers: Infrastructure dependent positioning systems and systems which do not depend on an infrastructure.

Examples for infrastructure dependent positioning systems are systems that can find their posture using WLAN trilateration or WLAN strength measurements [15]. The infrastructure used here are WLAN spots, which have to be installed at known positions. In principle a portable sensing unit estimates its position by evaluating the distances to different WLAN stations which serve as beacons. The main downside of localization using a stationary system is that this imposes the need of actually deploying the infrastructure, which can get expensive for larger buildings and restricts the localization to that limited area.

To minimize cost for indoor positioning, systems which do not depend on a preinstalled system in the building are examined. Thinking of robotics, where localization is solved using sensors like laser range-finders, cameras and motor encoders, it is obvious that for pedestrian tracking smaller and lighter sensors are necessary, like inertial measurement units (IMUs). Recently the rapid development of micro-electro-mechanical systems (MEMS) provided smaller and cheaper IMUs, which on the downside are more susceptible to noise. Installed in portable devices such IMUs can be used to track the position of pedestrians relative to a starting position. As pointed out in [7] there are two ways of localizing a person: Inertial Navigation and Dead Reckoning.

The first approach uses acceleration sensors and gyrometers in combination with the fundamental differential equation $\ddot{\vec{r}} = \vec{a}$, where $\vec{r} = \begin{pmatrix} x & y & z \end{pmatrix}$ is the vector traveled and $\vec{a} = \begin{pmatrix} a_x & a_y & a_z \end{pmatrix}$ is the acceleration applied to the sensor. Since a double integration of the acceleration is necessary to achieve relative localization, the noise induced by the sensors generates a large drift in the position. The main problem are the perturbated sensor readings from the gyrometers, which are used to zero out the gravity from the accelerometer measurements. Small errors in the orientation of the gravity result in unwanted acceleration which is integrated twice producing a large position drift. This problem can be tackled using moments of non-movement to perform zero velocity updates (ZUPT) of the gyrometer rotation velocities, which have to be zero, and of the orientation of gravity [16]. ZUPT is usually used in systems with foot-mounted IMUs, where zero velocities occur every time the foot of the IMU stands on the ground. By this approach the errors of standard inertial navigation, which increase proportional to the square of the running time of the system, can be confined to an error linear in the number of steps.

Dead reckoning localization uses information about the walking behavior to keep track of the position of the pedestrian. Features like heading, stride length and step frequency describe the walking behavior of a person. Such information can be extracted from combination of different sensor types, like accelerometers, gyrometers and compass sensors [4] [16] using techniques like ZUPT. The big advantage is that such systems can be used universally in different buildings without further expenses. The map of the buildings can be obtained by digitalizing available building plans. Recently there are also attempts to generate the map of the building while localizing the person [20]. This so called Self Localization and Map building (SLAM) problem is regarded as solved for robotic applications [5] where precise sensors like laser range finders are available to extract features from the surrounding environment. In the pedestrian case no such additional measurements are available and the systems rely on techniques like using similarities between trajectories to extract features of the building outline. Since this is computationally demanding the solution proposed in [20] can only be processed off-line. On the downside a system which does not need an infrastructure, can only provide relative positioning based on noisy sensor measurements. Without advanced algorithms, small errors introduced by the sensors increase the localization error with every measurement as shown in [25]. Methods to deal with these errors are researched by various teams all over the world. The key idea behind those techniques is to estimate the true position of a person from noisy measurements obtained from a IMU using Bayesian filters.

This thesis describes a robust system for pedestrian indoor localization and tracking using dead reckoning in combination with a Bayesian filter namely a particle filter to compensate the noisy sensor measurements. In the first chapter an overview over the PiNav-System is given. The PiNav-System is an inertial measurement system developed to provide the stride and orientation data of a person wearing it. In contrast to [28] , [20] or [25] a hip-mounted IMU is used. Next a general introduction to predictive or Bayesian filters, which are used to estimate the state of a dynamic system, is given. Subsequent sections (3.2) and (3.3) elude the concept of particle filters as a special predictive filter. How online indoor pedestrian localization can be tackled with a particle filter in practice is detailed out in chapter (4). Special emphasis is put on the map as an observation device, as this leads to an improved map representation describing the accessibility of space called the Accessibility Map (AM), which enables it to incorporate human behavior in indoor environments. Finally a different way of interpreting the AM as a Radial Basis Function Network (RBFN) is proposed which leads to a novel concept for learning accessibility of space from the walking patterns of a pedestrian.

2 The PiNav-System

The PiNav-System is an inertial measurement system, which was developed at the Institute for Real-Time Computer Systems (RCS) at the Technical University of Munich (TUM). PiNav stands for pedestrian inertial navigation which clearly states the goal of the system: It aims to provide inertial navigation for pedestrians implying that the working area of the system is not constraint. Although navigation in indoor and outdoor environments is the goal, the focus of this thesis lies on the indoor localization of a person wearing the PiNav-System.

In the following a brief overview of the system will be given. An in depth description of the system would go beyond the scope of this thesis and is not required to fully understand the techniques for pedestrian localization, which will be presented.

2.1 Hardware

The PiNav-System consists of four main hardware parts: The pocket sensing unit, the foot-mounted sensor, a GPS receiver and the handheld computing device. The central device is the computing device which is connected with the three sensor units via Bluetooth, as depicted in Figure 2.1. On the computing device a linux system is installed, which enables the deployment of graphical user interfaces using the QT framework.

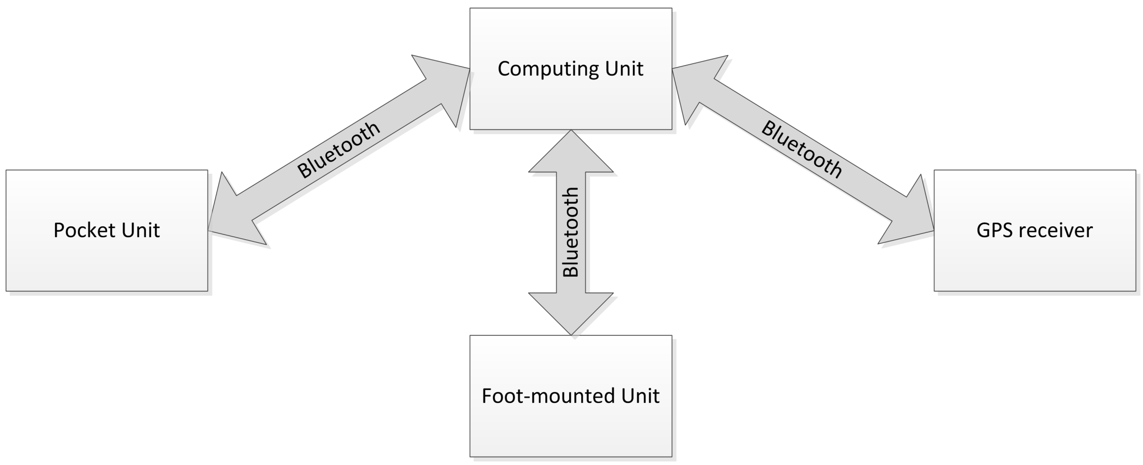


Figure 2.1: The PiNav-System consisting of pocket unit, foot-mounted sensor and computing device.

The pocket unit and the foot-mounted sensor were developed from scratch at the RCS. They are equipped with 3+3 degree of freedom(DOF) inertial sensing units. This means there is a 3-DOF accelerometer and a 3-DOF gyrometer on each of them. In addition the pocket unit has a 3-DOF magnetic field sensor and a pressure sensor for height estimation. GPS positioning in outdoor areas is provided by an additional standard GPS receiver.

At the time when this thesis was written the foot-mounted sensor has not yet been integrated into the system. Therefore this work will be constrained to the input data gathered from the pocket sensing device. Since the pocket device can provide all relevant data like stride length and orientation, this poses no restrictions to the overall functionality of the localization system proposed in this thesis. Adding the foot-mounted unit would only enhance the measurement results.

2.1.1 The Pocket Inertial Sensing Device

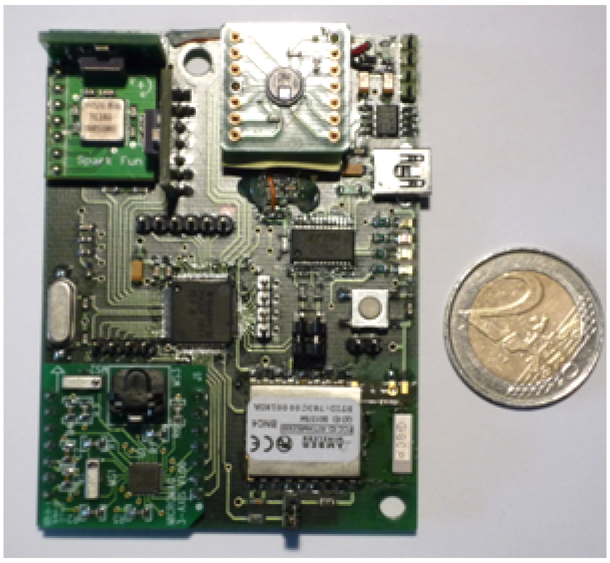


Figure 2.2: Picture of the pocket IMU without housing.

The electronics of the pocket inertial sensing unit are protected by a small black box surrounding them. The device can be put into ones pocket while walking around. The power is supplied by a single Li-Ion-accumulator cell which can be charged through a USB connection. A BlueNiceCom4 Bluetooth class 2 module from Amber-Wireless is used to provide wireless connection to any computing device. The whole circuit is built up around a MSP430 microprocessor as depicted in Figure 2.3. The software on that device handles both the readout of the sensors and the transmission of the data to the computing unit via Bluetooth.

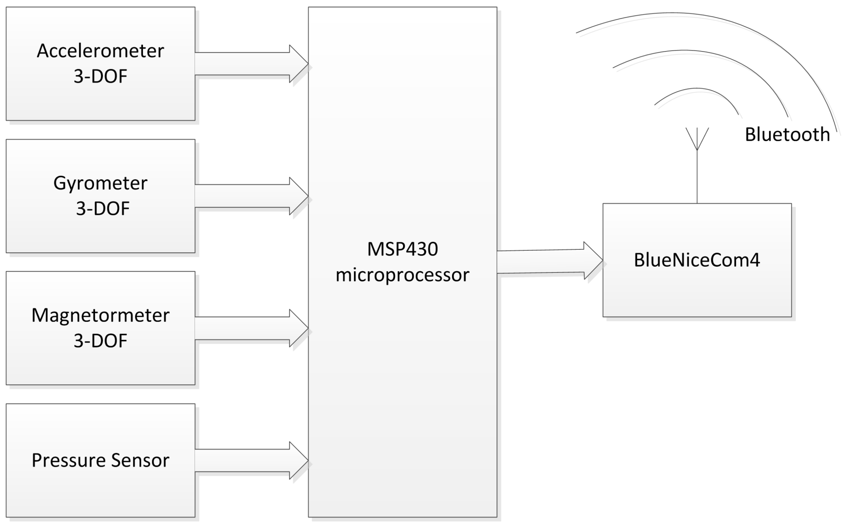


Figure 2.3: Schematic build up of hardware of the pocket IMU confined to the microprocessor, the sensors and the bluetooth module.

The main purpose of sensing the motions of a person is achieved by a three-degree-of-freedom(3-DOF) accelerometer, a 3-DOF gyrometer and a 3-DOF magnetic field sensor, which is used to measure the magnetic field of the earth. Besides these components there is a pressure sensor for height estimation, which can be used to detect the elevation during a transition between two floors.

For an in-depth description of the hardware unit and the involved sensors the reader is referred to [7].

2.2 Software

The whole PiNav-software is written in C++ using the QT-framework to display measurement-results in an easy way. The QT/C++ project can be cross compiled for the linux-system on the computing device.

The software consists of three threads: the first thread collects the data from the sensor-box and writes it into several ring buffers. The second thread filters the data in the ring buffers and evaluates the Particle Filter. Then the results are also stored in ring buffers. The third thread displays the data from the sensor and the information which is extracted by the second task, in a QT-window.

The start of a measurement sequence can be issued in the graphical user interface(GUI). After the synchronization of the clocks, measurements are received via bluetooth and stored in the input buffers from which they are processed by the filters. The results of the filter algorithms are then stored in the output buffers. Both the content of the input and the output buffers can be displayed in the GUI as depicted in Figure 2.4.

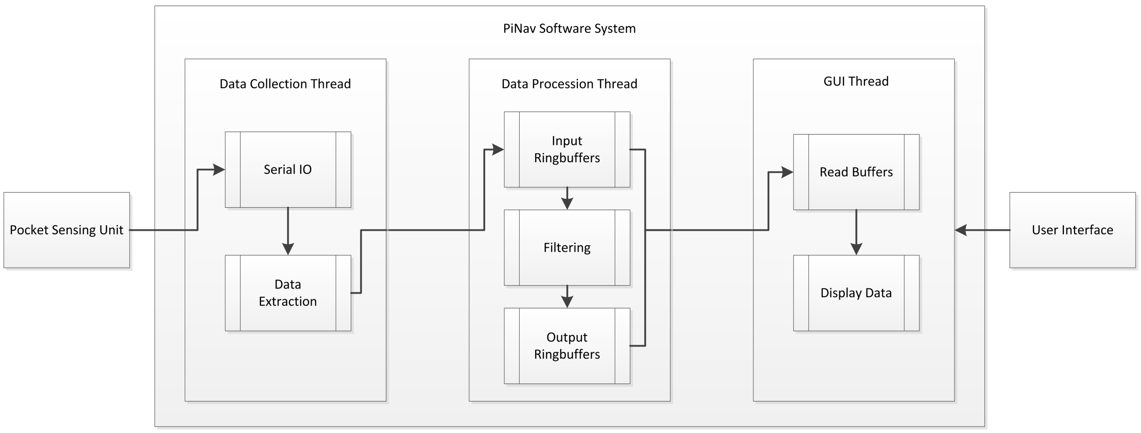


Figure 2.4: The software architecture of the PiNav-System.

2.2.1 The Data-Collection Thread

The whole communication between the pocket-sensor-box and the computing unit is secured with timestamps and check-sums. This makes the communication robust against transmission errors and provides a shared time-basis for the filtering algorithms. To establish this timebase the clocks of the microcontroller and the laptop, are synchronized at system start-up.

2.2.2 The Filtering Thread

In the second thread the filtering algorithms are applied to the data in the input ring buffers. The relevant filters for the Particle Filter are the orientation filter and the step detection algorithm. As depicted in Figure 2.5, those provide the input for the Particle Filter described in section (4.5), which estimates the position of the person carrying the pocket sensing device.

The orientation of the person is estimated using a Kalman filter, which fuses the orientation data of the gyrometers and the preprocessed data of the compass sensor. The raw output of the compass sensor needs to be corrected according to the local inclination and deinclination of the magnetic field of the earth [7].

The step-detection algorithm extracts the moments of foot contact with the ground. The frequency of these step-events is used to calculate the length of a step via the v-f-relation [1] as described in [7].

The pressure sensor is intended for the purpose of elevation detection, which indicates the transition between two floors. Although this functionality is not yet implemented in the PiNav-software, the localization system was designed to be easily upgradeable to manage transitions between floors. A floor selector chooses the correct floor with regard to the detected elevation and forwards it to the Particle Filter. Lacking the altitude estimation, the floor plan is at this stage of the software hardcoded in the PiNav-system.

A complete description of the orientation filter and the step detection algorithm will not be

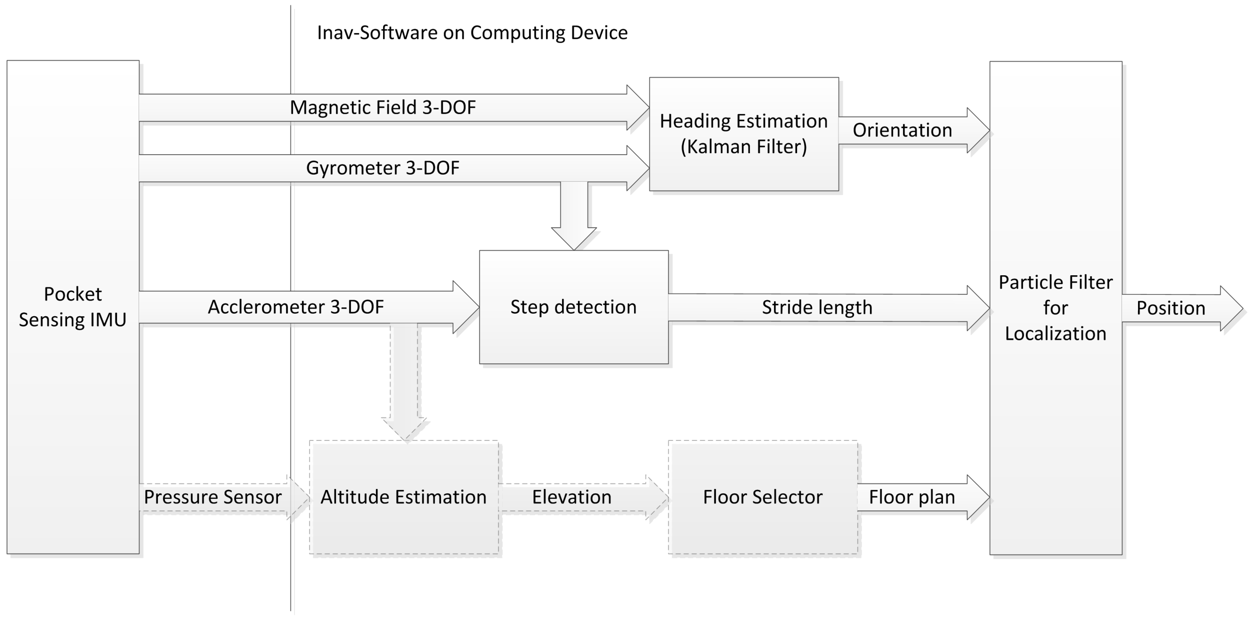


Figure 2.5: The main filters of the PiNav-System are the Kalman filter for heading estimation and the step detection algorithm. The altitude estimation was not yet implemented, but could be used to detect transitions between floors.

given here since this would go beyond the scope of this thesis. It is sufficient to have a general understanding of these methods to gather full insight into the Particle Filter, which is the topic of this work.

2.2.3 The Graphical User Interface Thread

The third thread runs the qt-window, which is used to control the whole system and to display the measured data as well as the output of the filters in real-time. Using specific keys to switch through the different curves, the user can track the content of the ringbuffers in which the sensor measurements and filter outputs are stored.

Besides starting a normal measurement cycle, a replay of an recorded sequence of measurements can be issued. Using this simulation mode it is possible to evaluate different filter configurations on the same data basis, which can be compared against each other afterwards.

3 Theoretical Fundamentals of the Particle Filter

3.1 Bayesian Inference with Predictive Filters

Bayesian or predictive filters infer the development of the probability density function (pdf) of a dynamic system's true state from noisy observations over time. Based on [8] and [19] a general description of the Bayesian inference process will be given. The description of the systems evolution will be confined to discrete times $t_k = t_{k-1} + T_{k-1}$, where T_k denotes the interval between two observations of the systems true state. Note that these events do not have to be periodic, as T_k may vary over time.

At first let $\vec{x}_k \in \mathbb{R}^n$ be the n-dimensional random variable describing the system's state at the discrete time t_k. Given the process noise $\vec{v}_k$ and the transition model $\vec{f_k}(\vec{x}_k, \vec{v}_k)$, the state of the system at time $k + 1$ can be expressed as

$$\vec{x}_{k+1} = \vec{f_k}(\vec{x}_k, \vec{v}_k). \tag{3.1}$$

The random sequence $\vec{v}_k$ models unexpected disturbances of the system's transition model $\vec{f_k}$.

Now every discrete time t_k observations $\vec{y}_k \in \mathbb{R}^m$ of the state of the underlying system are obtained. These measurements relate to the state of the system according to the observation equation

$$\vec{y}_k = \vec{h}_k(\vec{x}_k, \vec{w}_k) \tag{3.2}$$

where $\vec{h}$ is the known observation model of the system and $\vec{w}_k$ is a multivariate random variable describing the noise of the measurement at time t_k.

The state of the system $\vec{x}_k$ depends on all previous observations $Y^k \triangleq \{\vec{y}_1 \dots \vec{y}_k\}$, which leads to the description of the belief of the system's state by the pdf $p(\vec{x}_k|Y^k)$. Knowing the initial distribution $p(\vec{x}_1|Y^0) \triangleq p(\vec{x}_1)$ a predictive filter can approximate $p(\vec{x}_k|Y^k)$ from $p(\vec{x}_{k-1}|Y^{k-1})$ and $\vec{y}_k$. This means an estimate of the system's state at time t_k based on all preceding measurements can be obtained recursively.

This recursion can be divided into two steps: prediction and update. As depicted in Figure 3.1, at first the next system state $p(\vec{x}_k|Y^{k-1})$ is predicted using the prior distribution $p(\vec{x}_{k-1}|Y^{k-1})$ and the transition model of the system $\vec{f}_{k-1}$. Second, the proposal distribution $p(\vec{x}_k|Y^{k-1})$ is updated using all new measurements $\vec{y}_k$ yielding $p(\vec{x}_k|Y^k)$.

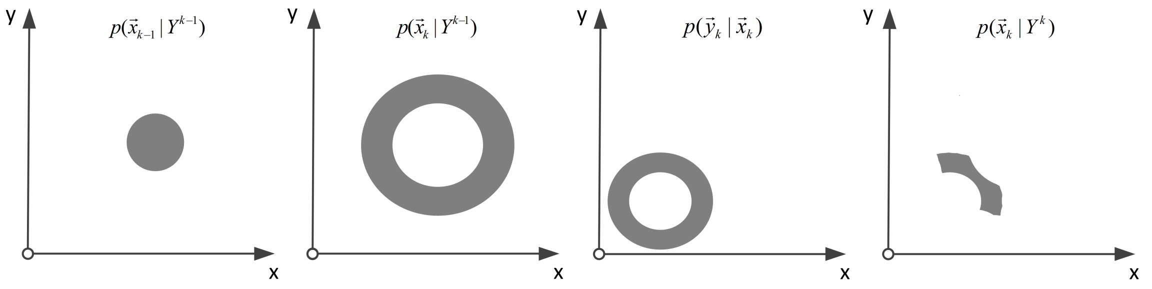


Figure 3.1: One iteration of a predictive filter. The grey areas depict equal probabilities greater than zero. From left to right, at first the transition model of the system is applied retrieving the proposal pdf $p(\vec{x}_k|Y^{k-1})$. Second the observation model $p(\vec{y}_k|\vec{x}_k)$ is incorporated yielding the updated belief of the system's state $p(\vec{x}_k|Y^k)$.

Assuming the prior distribution of the system's state $p(\vec{x}_{k-1}|Y^{k-1})$ at time t_{k-1} to be known, the predicted pdf $p(\vec{x}_k|Y^{k-1})$ can be obtained by the Chapman-Kolmogorov equation [17]:

$$p(\vec{x}_k|Y^{k-1}) = \int p(\vec{x}_k|\vec{x}_{k-1})p(\vec{x}_{k-1}|Y^{k-1})d\vec{x}_{k-1}. \tag{3.3}$$

The probabilistic evolution of the system's state $p(\vec{x}_k|\vec{x}_{k-1})$ is defined by the transition model of the system described in Equation (3.1) and the known process noise $\vec{v}_k$.

Incorporating the observation $\vec{y}_k$ the belief of the system's state $p(\vec{x}_k|Y^k)$ can be expressed as

$$p(\vec{x}_k|Y^k) = p(\vec{x}_k|\vec{y}_k, Y^{k-1}) = \frac{p(\vec{y}_k|\vec{x}_k)p(\vec{x}_k|Y^{k-1})}{p(\vec{y}_k|Y^{k-1})}, \tag{3.4}$$

using the Bayes' rule [2].

As in (3.3) applying the Chapman-Kolmogorov rule, the denominator $p(\vec{y}_k|Y^{k-1})$ can be deduced as

$$p(\vec{y}_k|Y^{k-1}) = \int p(\vec{y}_k|\vec{x}_k)p(\vec{x}_k|Y^{k-1})d\vec{x}_k \tag{3.5}$$

where the observation model $p(\vec{y}_k|\vec{x}_k)$ is defined by Equation (3.2) and the known statistics of $\vec{w}_k$.

Combining Equations (3.4) and (3.5), the desired formula for the posterior state pdf $p(\vec{x}_k|Y^k)$ of the system at time t_k can be given as

$$p(\vec{x}_k|Y^k) = \frac{p(\vec{y}_k|\vec{x}_k)p(\vec{x}_k|Y^{k-1})}{\int p(\vec{y}_k|\vec{x}_k)p(\vec{x}_k|Y^{k-1})d\vec{x}_k}. \tag{3.6}$$

The recursive Equations (3.3) and (3.6) form the basis of any predictive filter. In general these two equations can not be handled, as it is necessary to have a complete description of the involved pdfs, which would require vectors of infinite dimension. Therefore only restricted cases, where the densities are characterized by a finite number of properties, can be tackled in an optimal way. Which means it is possible to get the best estimation of the system's true state.

This class of optimal predictive filters consists of the so called Kalman filter [12], which relies on a linear system and Gaussian densities, and grid based filters [19], where the state space consists of discrete values and is finite.

In other cases no optimal predictive filter can be derived. Depending on the system different approaches lead to suboptimal predictive filters approximating the evolution of the system. In the non-linear Gaussian case analytical approximations such as the extended Kalman filter (EKF) [8] are deployed. The most general cases, where no assumptions on the shape of the underlying pdfs or the linearity of the system are made, are tackled using grid-based numerical approximations of the Equations (3.3) and (3.6) [19] or filters like the unscented Kalman filter (UKF) [23], Gaussian sum filters [21] or Particle Filters [9]. Since a complete description of all filters mentioned above would go beyond the scope of this thesis, the reader is referred to [19] for a in depth description.

This work elaborates on the Particle Filter, which has some special properties making him the most suitable choice for pedestrian dead reckoning localization. Those characteristics will be pointed out in chapter (4), where, basing on the theory of this chapter, a Particle Filter for pedestrian localization is derived.

3.2 The Sampling Importance Resampling Particle Filter

Since the seminal paper [9] on Particle Filters by Gordon, Salmond and Smith in 1993 on a new way of estimating the state of a nonlinear and non-Gaussian system, there is a lot of interest in this filters for various tracking applications including localization of pedestrians.

Particle filters perform sequential Monte Carlo estimation [19] of a system's state $\vec{x}_k$ representing the pdf $p(\vec{x}_k|Y^k)$ by a set S_k of so called particles s_k^i, where

$$S_k = \left\{ \left\langle s_k^i; w_k^i \right\rangle ; i = 1 \ldots N_k \right\} \triangleq \{s_k^i; w_k^i\}. \tag{3.7}$$

Drawn from the system's pdf $p(\vec{x}_k|Y^k)$, a particle s_k^i describes a possible state $\vec{x}_k$ of the system. It is weighted with the mass w_k^i which is proportional to the degree of the particle's contribution to the pdf $p(\vec{x}_k|Y^k)$.

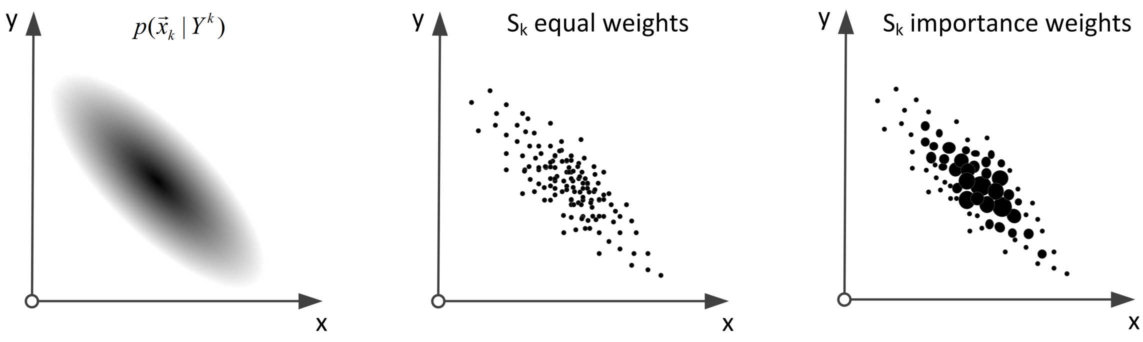


Figure 3.2: Three ways of representing a probability density function - The left pdf is represented by a closed function $p(\vec{x}_k|Y^k)$. To the right two variants of a particle representation of the pdf of the left are given. In the middle the particles have equal weights and the pdf is described by the density of the particles. The rightmost figure represents the pdf by particles, which are weighted according to their importance for the representation of $p(\vec{x}_k|Y^k)$. Notice that the weights of the particles are depicted by their radius. A larger radius describes a heavier weight.

Let S_1 be the initial set of particles describing the known prior probability density of the system's state $p(\vec{x}_1|Y^0) = p(\vec{x}_1)$. Starting from this, after every new observation y_k, the recursive algorithm of the Particle Filter is evaluated. Proposed by Gordon, Salmond and Smith [9], the Sampling Importance Resampling (SIR) filter, also called Bayesian bootstrap filter, is considered the basic model of a Particle Filter. As introduced in section (3.1) for predictive filters, the SIR filter is also divided into the two steps prediction and update, realizing the central Equations (3.3) and (3.6).

Prediction (Particle Propagation) By applying the system transition model f_{k-1} to each individual equally weighted particle of S_{k-1}, an intermediate set $\tilde{S}_k$ of particles is generated.

$$S_{k-1} = \left\{ s_{k-1}^i ; \frac{1}{N_{k-1}} \right\} \xrightarrow{f_{k-1}} \tilde{S}_k = \left\{ \tilde{s}_k^i ; \frac{1}{N_{k-1}} \right\} \tag{3.8}$$

Consistent with the theory of predictive filters, $\tilde{S}_k$ represents the pdf $p(\vec{x}_k|Y^{k-1})$, since Equation (3.8) realizes Equation (3.3).

Update (Particle Weighting) According to the observation density $p(\vec{y}_k|\vec{x}_k)$ every particle $\tilde{s}_k^i \in \tilde{S}_k$ is assigned a weight

$$\tilde{w}_k^i \sim p(\vec{y}_k|\tilde{s}_k^i). \tag{3.9}$$

The updated scattering of weighted particles $\tilde{S}_k$ describes the estimation of the state of the system $p(\vec{x}_k|Y^k)$ at time t_k under the new observation $\vec{y}_k$. The weight $\tilde{w}_k^i$ displays the importance of the particle $\tilde{s}_k^i$ for the distribution $\tilde{S}_k$ describing $p(x_k|Y^k)$. Since the SIR algorithm takes a set of particles with equal mass as input, it is necessary to sample the final distribution S_k from $\tilde{S}_k$ such that $w_k^i = \frac{1}{N_k}$.

$$\tilde{S}_k = \left\{\tilde{s}_k^i; \tilde{w}_k^i\right\} \xrightarrow{\text{Resample using } \tilde{w}_k^i} S_k = \left\{s_k^i; \frac{1}{N_k}\right\} \tag{3.10}$$

Essentially the resampling generates an amount of particles $s_k^i \in S_k$ from every particle $\tilde{s}_k^i \in \tilde{S}_k$ proportional to the specific weight $\tilde{w}_k^i$ of this particle. This means heavy weighted particles $\tilde{s}_k^i$ will generate several particles s_k^i whereas a lightweight particle might be deleted in this step.

To enable better understanding of the updating process of the particle distribution given an observation y_k Figure 3.3 is given. The example in the illustration is confined to one dimension x to ensure a straightforward overview. The general principle does not change in a higher dimensional state space. As can be seen at first the particles of $\{s_{k-1}^i; \frac{1}{N_{k-1}}\}$ are propagated shifting

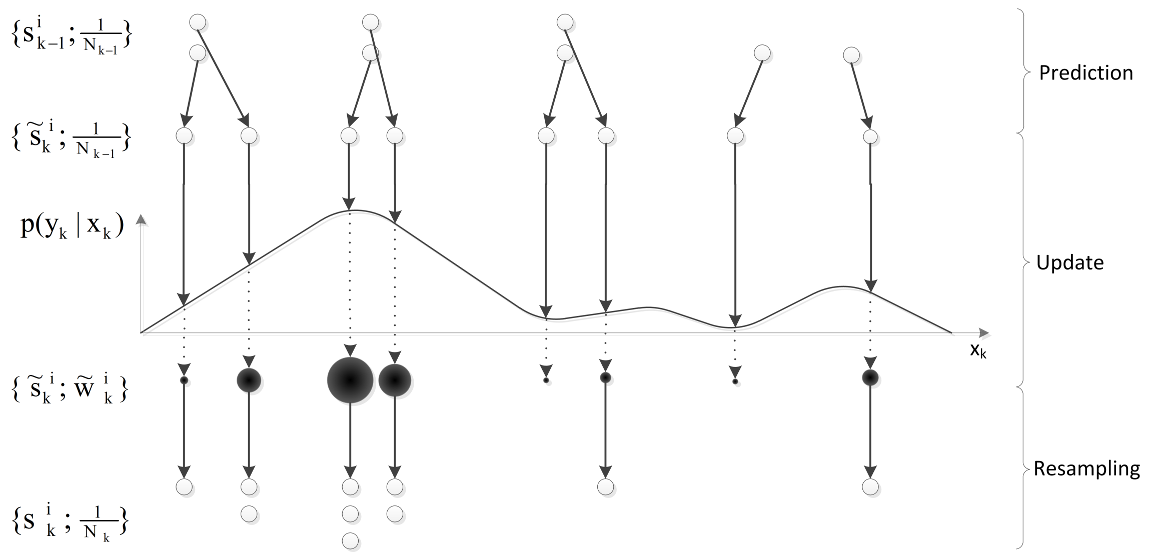

$\{s_{k-1}^i; \frac{1}{N_{k-1}}\}$
$\{\tilde{s}_k^i; \frac{1}{N_{k-1}}\}$
$p(y_k|x_k)$
$\{\tilde{s}_k^i; \tilde{w}_k^i\}$
$\{s_k^i; \frac{1}{N_k}\}$
x_k
Prediction
Update
Resampling

Figure 3.3: Illustration of one complete cycle of the Particle Filter.

them to slightly different x values according to the transition model $p(x_k|x_{k-1})$ providing the set $\{s_k^i; \frac{1}{N_{k-1}}\}$. Now the observation y_k is used to change the weights from the former $\frac{1}{N_{k-1}}$ to a $w_k^i \sim p(y_k|x_k)$, which is depicted using circles with radius proportional to $p(y_k|x_k)$. In the final resampling step equally weighted particles are generated from the weighted ones. The number of particles generated from a weighted particle $\tilde{s}_k^i$ is proportional to its weight $\tilde{w}_k^i$.

3.2.1 Efficient Resampling

The purpose of the resampling step is to randomly select particles of the intermediate distribution $\tilde{S}_k$ and to add them to the posterior distribution S_k. In order to preserve the distribution $\tilde{S}_k$ which represents $p(\vec{x}_k|Y^k)$, the chance of picking a particle $\tilde{s}_k^i \in \tilde{S}_k$ has to be proportional to the weight $\tilde{w}_k^i$ of this particle.

This can be achieved by drawing N_k standard uniform values $u_j \in [0, 1]$ and counting the number n_l of those u_j, which fall into the range $Q_{l-1} < u_j < Q_l$, where $Q_l = \sum_{i=1}^{l} \tilde{w}_k^i$; $Q_0 = 0$ is the cumulative sum of the weights $\tilde{w}_k^i$ of the particles $\tilde{s}_k^i$. The number n_l gives the amount of particles s_k^i which have to be generated from the particle $\tilde{s}_k^l$. This basic method of resampling is performed by Algorithm 3.1 which was proposed for example in [18].

Algorithm 3.1 $O(N \log(N))$ basic resampling algorithm for the SIR Particle Filter

```
method resampleONlogN$\left(\tilde{S}_k = \left\{\tilde{s}_k^i; \tilde{w}_k^i\right\}\right)$
  $Q_0 = 0$
  for $i = 1$ to $N_k$ do
      $Q_i = Q_{i-1} + w_k^i$
      Sample $u_i \in [0, 1]$ from uniform distribution
  end for
  Sort$\left(u_j\right)$ // sort in ascending order using any fast sorting algorithm
  $j = 1$
  $l = 1$
  while $j < (N_k + 1)$ do
      if $Q_{l-1} < u_j < Q_l$ then
          $j++$
          $s_k^l = \tilde{s}_k^l$
          $S_k = S_k \cup \left\langle s_k^l; \frac{1}{N_k} \right\rangle$
      else
          $l++$
      end if
  end while
return $S_k = \{s_k^i; \frac{1}{N_{k-1}}\}$
```

Since the implementation of this intuitive approach involves the use of a sorting algorithm the best achievable time complexity is $O(N \log(N))$ using for example a merge-sort or a heap-sort sorting-algorithm.

In [3] Carpenter, Clifford and Fearnhead utilized a more efficient algorithm, which resamples S_k from $\tilde{S}_k$ in $O(N)$ time. This method generates exponentially distributed random values $t_0 \dots t_{N_k}$ by $t_j = -\log_{10}(u_j)$, calculates the running total $T_j = \sum_{i=0}^{j} t_i$ and merges T_j and the Q_l as follows:

Algorithm 3.2 $O(N)$ resampling algorithm for the SIR Particle Filter

```
method resampleON$\left(\tilde{S}_k = \left\{\tilde{s}_k^i; \tilde{w}_k^i\right\}\right)$
    Sample $u_0 \in [0, 1]$ from uniform distribution
    $T_0 = -log_{10}(u_0)$
    $Q_0 = 0$
    for $i = 1$ to $N_k$ do
        Sample $u_i \in [0, 1]$ from uniform distribution
        $T_i = T_{i-1} - log_{10}(u_i)$
        $Q_i = Q_{i-1} + w_k^i$
    end for
    $j = 0$
    $l = 1$
    while $j < N_k$ do
        if $Q_l \cdot T_{N_k} > T_j$ then
            $j++$
            $S_k = S_k \cup \left\langle \tilde{s}_k^l; \frac{1}{N_k} \right\rangle$
        else
            $l++$
        end if
    end while
return $S_k = \left\{s_k^i; \frac{1}{N_{k-1}}\right\}$
```

Algorithm 3.2 uses the special features [14] of the running total of exponentially distributed random values making explicit sorting superficial.

3.2.2 The SIR Algorithm

Using the theory on Sampling Importance Resampling Particle Filters derived at the beginning of this section (3.2) and the $O(N)$ resampling Algorithm 3.2, the SIR filter is given. It has to be noted that Algorithm 3.3 is a recursive formulation, which refreshes the belief of the system's state under a new observation $\vec{y}_k$. Given the initial density $p(\vec{x}_1|Y^0)$ described by S^1 and a series of observations Y^k the state of the system at time t_k can be estimated.

Algorithm 3.3 The SIR Particle Filter algorithm performs an update of the particle distribution of time t_{k-1} to time t_k.

```
method updateBeliefSIR$\left(S_{k-1} = \left\{s^i_{k-1}; w^i_{k-1}\right\}, \vec{y}_k\right)$
  $w_{SUM} = 0$
  for $i = 1$ to $N_{k-1}$ do
      Propagate particle $\tilde{s}^i_k = f_{k-1}(s^i_{k-1})$
      Evaluate particle weight $\tilde{w}^i_k \sim p(y_k|\tilde{s}^i_k)$
      $\tilde{S}_k = \tilde{S}_k \cup \left\langle \tilde{s}^i_k; \tilde{w}^i_k \right\rangle$
      $w_{SUM} = w_{SUM} + \tilde{w}^i_k$
  end for
  for $i = 1$ to $N_k$ do
      Normalize weights $w^i_k = \frac{\tilde{w}^i_k}{w_{SUM}}$
  end for
  $S_k$ = resampleON$\left(\tilde{S}_k\right)$ // using Algorithm 3.2
return $S_k = \left\{s^i_k; w^i_k\right\}$
```

Note that all w^i_{k-1} have to be equal. This is secured by the resampling step at the end of the algorithm.

3.3 Kullback-Leibler Divergence Sampling

The standard SIR filter propagates a constant amount of particles to estimate the state of the system. Usually to get a good estimate of the system's state a large number of particles has to be used as depicted in [6]. In addition there is no straight-forward way of determining the necessary quantity of particles sufficient to accurately describe the true pdf of the state of the system.

A solution to that problem is to adapt the number of particles dynamically using the Kullback-Leibler divergence (KLD) as proposed in [6] and adopted to pedestrian localization in [28]. The Kullback-Leibler divergence quantifies the difference between two probability densities, one of which may remain unknown. Thus the difference between the estimated pdf, expressed by the particle distribution S_k, and the unknown true pdf of the system's state $p(\vec{x}_k|Y^k)$ can be measured, providing a criterion for the accuracy of S_k describing $p(\vec{x}_k|Y^k)$.

The idea of the KLD-sampling algorithm is to increase the number of particles until the Kullback-Leibler distance between the particle distribution and the true pdf can be guarantied with probability $(1-\delta)$ to be less than a predefined threshold ϵ. As derived in [6] the number N of samples needed for an accurate description of the underlying pdf is then given by the inequality

$$N \geq \frac{1}{2\epsilon}\chi^2_{q-1,1-\delta}, \tag{3.11}$$

where $\chi^2_{q-1,1-\delta}$ denotes a chi-square-distribution of $q-1$ degrees of freedom. In order to be able to estimate the KL distance it is necessary to discretize the state space, maintaining a list of subspaces, which will be referred to as bins. As depicted in Figure 3.4, the particles are then projected onto the grid of bins. A particle is assigned to the specific bin which describes the subspace in which the state of the particle falls.

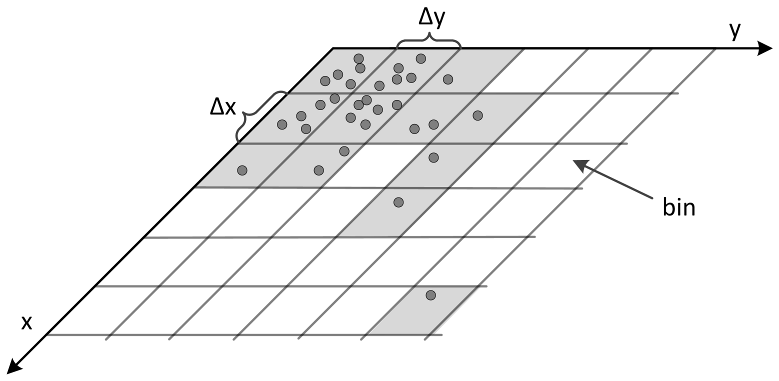


Figure 3.4: Two dimensional lattice of bins projected onto the state space. A set of particles is shown which is scattered over the state space. The relation of a particle and its bin becomes obvious. Supported bins, which are colored gray in the figure, can be counted, yielding a measure for the spread of the particle distribution.

Counting the number of bins occupied by particles the spread of the particle distribution can be measured. Using this information, the $\chi^2_{q-1,1-\delta}$ distribution can be approximated by the Wilson-Hilferty transformation [27] as proposed in [6]:

$$N \geq \frac{1}{2\epsilon}\chi^2_{q-1,1-\delta} = \frac{q-1}{2\epsilon}\left[1-\frac{2}{9(q-1)}+\sqrt{\frac{2}{9(q-1)}}z_{1-\delta}\right]^3, \tag{3.12}$$

where $z_{1-\delta}$ is the upper $1-\delta$ quantile of the standard normal distribution $\mathcal{N}(0,1)$ and q is the number of bins which contain particles. From Equation (3.12) can be concluded, that it is sufficient to count the number q of bins with support to get an estimate for the necessary amount N of particles to represent the underlying probability density in enough detail.

3.3.1 KLD-Sampling Particle Filter Algorithm

Since it is necessary to evaluate Inequation (3.12) after propagation and weighting of each particle, a sequential procession of the particles is necessary. After the intermediate distribution $\tilde{S}_k$ has been created, the weights $\tilde{w}_k^i$ are normalized and S_k is resampled from $\tilde{S}_k$ in order to generate equally weighted particles for the next cycle.

Algorithm 3.4 The KLD-sampling Particle Filter algorithm performs an update of the particle distribution of time t_{k-1} to time t_k using the KLD-sampling technique to adjust the number of particles describing the pdf.

```
method updateBeliefKLD$\left(S_{k-1} = \{s_{k-1}^i; w_{k-1}^i\}, \vec{y}_k\right)$
  Empty all bins $b$ // set all bins of state space to empty
  $q = 0$; $j = 0$; $w_{SUM} = 0$
  while $j \leq \frac{1}{2\epsilon}\chi^2_{q-1,1-\delta}$ do
      Randomly take $s_{k-1}^i$ from $S_{k-1}$
      Propagate particle $\tilde{s}_k^j = \vec{f}_{k-1}(s_{k-1}^i, \vec{v_{k-1}})$
      Evaluate particle weight $\tilde{w}_k^j \sim p(y_k|\tilde{s}_k^j)$
      $w_{SUM} = w_{SUM} + \tilde{w}_k^j$
      $\tilde{S}_k = \tilde{S}_k \cup \left\langle \tilde{s}_k^j; \tilde{w}_k^j \right\rangle$
      $j++$
      if bin $b_l$ containing $\tilde{s}_k^i$ is empty then
          $q++$
          $b_l$ = not empty
      end if
  end while
  for $i = 1$ to $N_k$ do
      Normalize weights: $\tilde{w}_k^i = \frac{\tilde{w}_k^i}{w_{SUM}}$
  end for
  $S_k$ = resampleON$\left(\tilde{S}_k\right)$ // using Algorithm 3.2
return $S_k = \left\{s_k^i; w_k^i\right\}$
```

Note that the formulation of Algorithm 3.4 is recursive like the SIR Particle Filter (Algorithm 3.3).

4 Pedestrian Localization with a Particle Filter

4.1 The System Model of a Pedestrian

From the theory of the predictive filters it is clear that the first thing necessary for the implementation of a Particle Filter is a mathematical description of the system, which is a moving pedestrian in an indoor environment.

A walking human's state at time t_k is defined by its position (x_k, y_k, z_k) in the world and its orientation θ_k with respect to the global coordinate system. As will be pointed out in section (4.2.2) it suffices to evaluated the Particle Filter in 2D space making the z_k coordinate superfluous for it. Consequently the state vector $\vec{x}_k$ at time t_k of a pedestrian on a particular floor can be given as

$$\vec{x}_k = \begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix} \iff s_k^i = \begin{pmatrix} x_k^i \\ y_k^i \\ \theta_k^i \end{pmatrix}. \tag{4.1}$$

This means also that each particle $s_k^i \in S_k$ stores the two coordinates x_k^i and y_k^i as well as its orientation θ_k^i. As pointed out in section (3.2) each particle is associated with a weight w_k^i which specifies the extent of the particle contribution to the underlying probability density of the pedestrian's state $p(\vec{x}_k|Y^k)$.

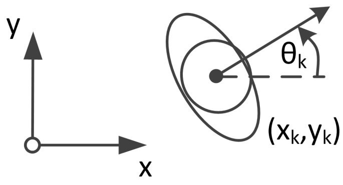


Figure 4.1: The state of a pedestrian in 2D space is described by his position (x_k, y_k) and his orientation θ_k. The person is depicted as two ellipses.

Gathered from the behavior of the pedestrian, the PiNav-System monitors every step and the heading of the person in the middle of a step. The measured orientation and the heading of the person do only correspond if the hip is parallel to the body of the person, which is the case directly in the middle of a step when both legs are parallel. The calibrated step detection algorithm computes the distance traveled using the v-f-relation as described in [7].

Since the pedestrian's orientation is filtered by a Kalman filter [12] the orientation change $\delta\theta_k$ can be assumed to be normal distributed according to $\delta\theta_k \sim \mathcal{N}(\mu_{\delta\theta_k}, \sigma^2_{\delta\theta_k})$, where $\mu_{\delta\theta_k}$ and $\sigma^2_{\delta\theta_k}$ can be obtained directly from the Kalman filter. The stride length l_k is thought of as Gaussian random variable distributed according to $l_k \sim \mathcal{N}(\mu_{l_k}, \sigma^2_{l_k})$, where μ_{l_k} and $\sigma^2_{l_k}$ are given by the step detection algorithm.

In summary the motion of a human in 2D-space is described by a series of turns and consecutive steps, as shown in Figure 4.2.

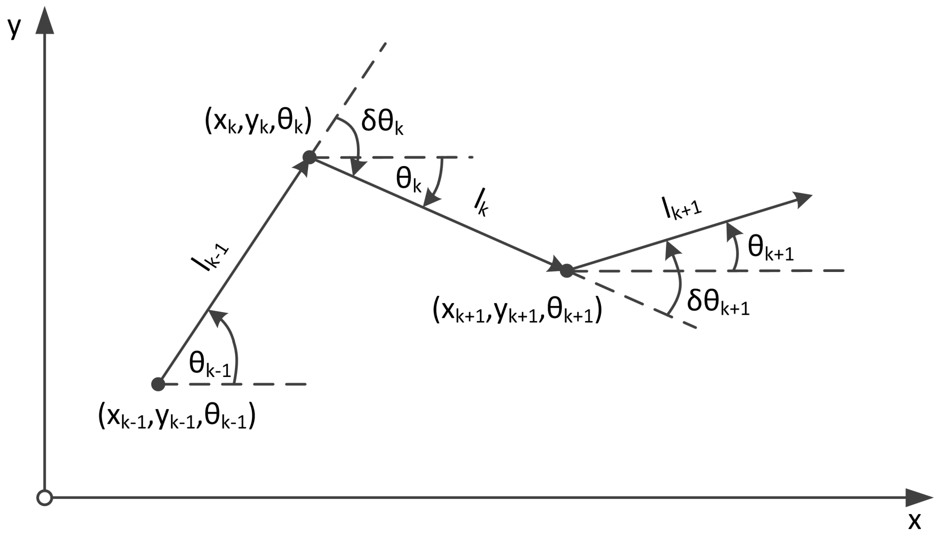


Figure 4.2: The system model of a pedestrian walking in 2D space. The arrows depict a series of three consecutive steps, which are abstracted to a vector with length l_k and orientation θ_k.

This implies that the system transition model $p(\vec{x}_k|\vec{x}_{k-1})$ can be described by the function $\vec{f}$ according to which the particles are propagated:

$$\vec{x}_k = \vec{f}(\vec{x}_{k-1}, \delta\theta_k, l_k) = \begin{pmatrix} x_{k-1} + l_k \cdot \cos(\theta_{k-1} + \delta\theta_k) \\ y_{k-1} + l_k \cdot \sin(\theta_{k-1} + \delta\theta_k) \\ \theta_{k-1} + \delta\theta_k \end{pmatrix} \tag{4.2}$$

where $\delta\theta_k$ needs to be sampled from $\mathcal{N}(\mu_{\delta\theta_k}, \sigma^2_{\delta\theta_k})$ and l_k from $\mathcal{N}(\mu_{l_k}, \sigma^2_{l_k})$. Also notice that the motion model itself is constant in time, which is expressed through the absence of an index t_k compared to Equation (3.1).

4.2 The Observation Model

In the following it will be pointed out how the map can be interpreted as an geometric observation model for the system's state. On a global scale a map can be divided into outdoor and indoor space. In this work the focus lies on the localization in indoor areas like buildings. Section (4.2.2) describes how the map of a building is represented within the program in a hierarchic structure. Finally the implementation of two methods is given which realize the observation of the system.

4.2.1 A Map as a Geometric Observation Model

One of the central parts of the Particle Filter for the localization of a person is the map of the building. It constrains the viable space of the pedestrian and thus the state space of the system. Exploiting this feature the localization of the pedestrian can be improved, since particles with impossible states can be sorted out by the filter.

In a first step it can be perceived that there are two classes of states $\vec{x}_k$: Those describing positions within the boundaries of the floors of the building and those which do not. Thus the map can be interpreted as an observation device for the state of the system as

$$y'_k = h(\vec{x}_k) = \begin{cases} \circ & \text{if } \vec{x}_k \text{ is within the map} \\ \bullet & \text{if } \vec{x}_k \text{ is out of the map} \end{cases},$$

where $\circ$ and $\bullet$ denote the two possible observations given a state $\vec{x}_k$ without any weighting. In reverse the map can be interpreted as the set $\mathbb{M} = \{\vec{x}_k | h(\vec{x}_k) = \circ\}$.

As it is impossible for a person to be out of the map, since this would require to break through the walls of the building, the observation $y'_k = \bullet$ can not be true for a possible state of the pedestrian. For all valid states of the person y'_k has to be $\circ$. In other words the pedestrian can only be inside a room of the building. In terms of probabilities this first part of the observation model $p(y'_k|\vec{x}_k)$ can therefore be expressed as

$$p(y'_k|\vec{x}_k) = \begin{cases} 1 & \text{if } y'_k = \circ \\ 0 & \text{otherwise} \end{cases}. \tag{4.3}$$

Furthermore a door has to be used if a person changes from one room to another. Modeling this observation, which is referred to as y''_k, in a mathematical way the inclusion of the state $\vec{x}_{k-1}$ is required. Using the state of time $k-1$ and k the trajectory of the person during the transition can be determined. In case the room of the person changed from time $k-1$ to time k, the state $\vec{x}_k$ can only be valid if the transition lead trough a door. The fact, that performing this observation makes only sense if the person is in the map, is expressed by the conditional dependence of the two observations y''_k and y'_k. All in all, the idea is expressed by the following distinction of cases:

$$p(y''_k|y'_k, \vec{x}_{k-1}, \vec{x}_k) = \begin{cases} 1 & \text{if passage from } \vec{x}_{k-1} \text{ to } \vec{x}_k \text{ is valid} \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

It is clear that $p(y'_k|\vec{x}_k)$ in Equation (4.3) is equal to $p(y'_k|\vec{x}_k, \vec{x}_{k-1})$, since a person can only be in a room at time t_k if he has been in a room in the previous timestep. Therefore Equation (4.3) and (4.4) can now be combined using the standard rule of conditional probability:

$$p(y'_k \cap y''_k|\vec{x}_{k-1}, \vec{x}_k) = p(y'_k|\vec{x}_k, \vec{x}_{k-1}) \cdot p(y''_k|y'_k, \vec{x}_{k-1}, \vec{x}_k) := p(y_k|\vec{x}_{k-1}, \vec{x}_k). \tag{4.5}$$

Equation (4.5) describes the probability of both, the observation that the person is in the map and the observation that the trajectory from $\vec{x}_{k-1}$ to $\vec{x}_k$ is valid. This yields the geometric observation model for a pedestrian walking around in a building.

4.2.2 Representation of the Map

Like the rest of the software the map and its functionality has been implemented in C++ exploiting the object orientation of this programming language.

The map is organized in a hierarchic structure as can be seen in Figure 4.3 detailing out how the map is composed of different C++ classes. Note that in the following *Building*, *Floor*, *Room*, *Stairs* and *Door* are used to refer to these classes in the algorithms.

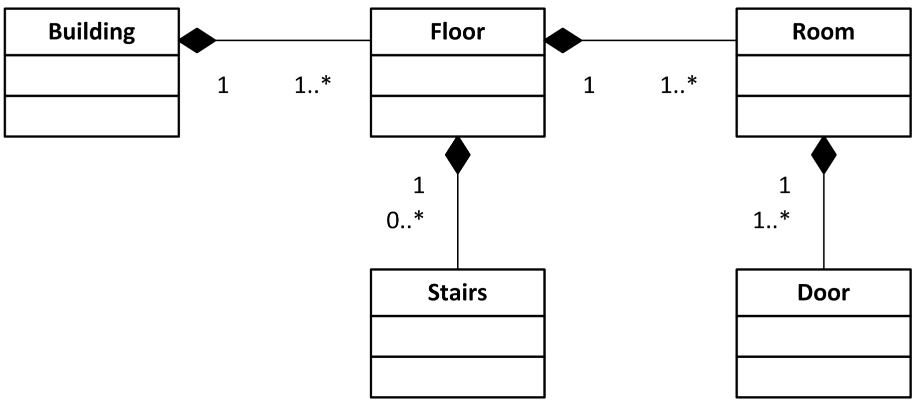


Figure 4.3: Composition diagram of the map.

The topmost class is the building class, which consists of a list of floor objects. The objective of this class is to give access to the different floors of a building. It is not practicable to have all floors of a building in the memory all the time, since only one floor is used in the Particle Filter at a time. The building class handles changes between floors by loading the consecutive floor from a known position on the flash or hard drive. The previous floor is deleted freeing its memory.

As already mentioned the Particle Filter operates on a single floor, which is represented by the floor class. Composing stairs and room objects, the floor class describes the outline of a floor and regions which enable the transition to other floors, like stairs or elevators.

Finally the room class represents a room by storing the contour of its walls as a polygon, which is described by a list of edge positions. Access to a room is only possible via one of its doors. These are represented by the door class describing a door by the coordinates of its two posts.

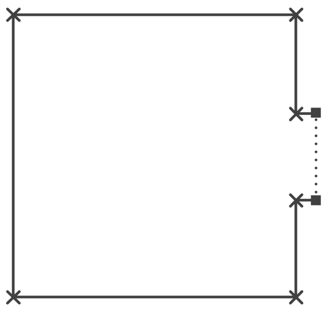


Figure 4.4: A room is represented by a polygon which stores the outline of the walls. In the picture the crosses and the squares denote the positions which are stored in the polygon. The door is depicted as a dotted line with the squares as the positions of its posts.

Representing the outline of a room as a polygon was chosen for two reasons. First, with a polygon any shape of room can be realized and therefore any floor outline is possible. Second, storing a list of positions consumes only few storage in comparison to for example a grid based approach.

All coordinates are stored in 3D-space to enable 2.5D localization as proposed in [28]. This approach propagates particles in 2D-space only, while the transitions in height are done in discrete steps, namely the different floors. Thus although the particles are associated with x_k and y_k coordinates only, the height z_k is implicitly stored in the information about the floor in which the particles are propagated at time k. As mentioned in section (2.2.2), height changes are monitored separately from the Particle Filter using the pressure sensor. If a significant elevation is detected in a stair-region all particles in such areas are propagated to the determined consecutive floor. Thus there is no full 3D operation but transitions to upper or lower floors are possible, which is described by the additional half dimension.

In Germany for most of the public buildings floor plans for each storey exist either on paper or even in digital form as Computer Aided Design (CAD) drawings. For the evaluation the storey of the RCS institute at the TUM was chosen. A CAD drawing of the floor outline could be obtained and is depicted in Figure 4.5.

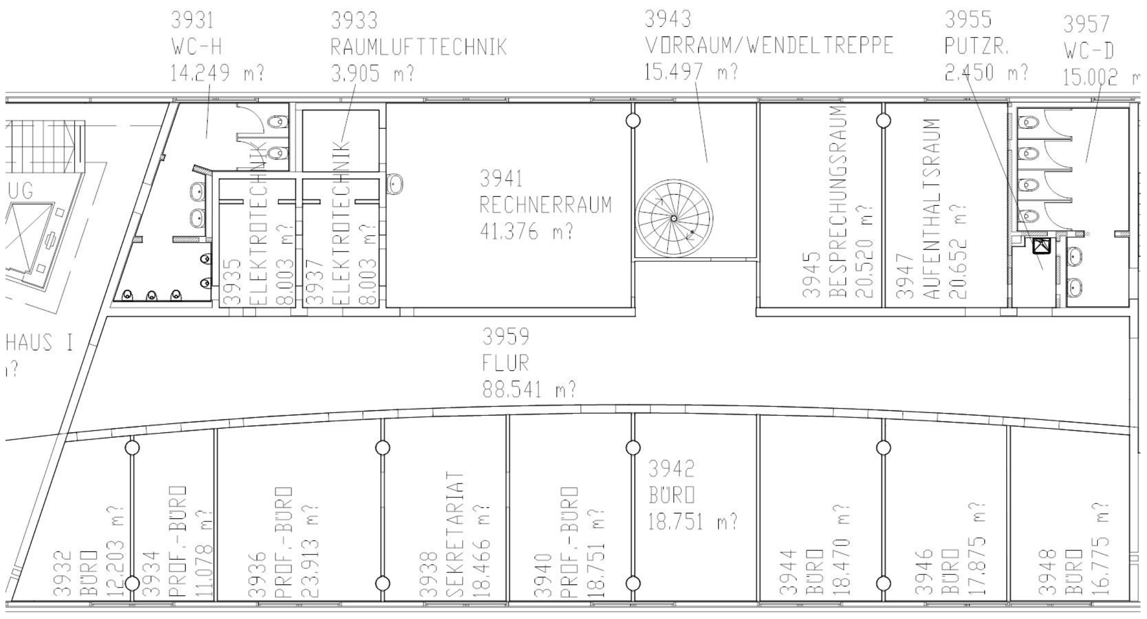


Figure 4.5: This is the CAD floor plan of the RCS institute as it could be obtained from TU Munich administration.

From this original the representation of the floor for the Particle Filter can be built up sequentially as depicted in Figure 4.6.

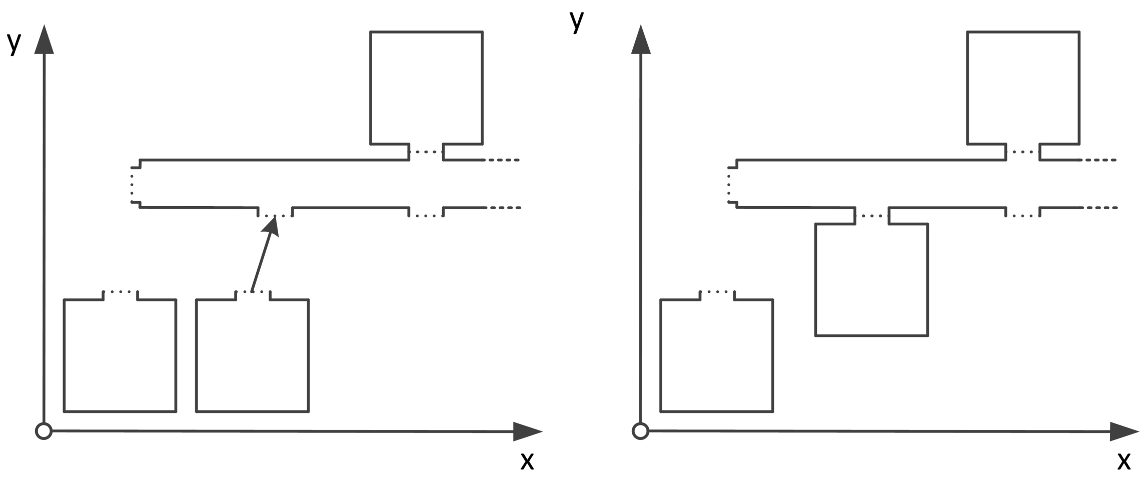


Figure 4.6: During build up of a floor, the different rooms are at first created and then shifted in place to generate the desired floor outline in the world coordinate system.

At first all room and door objects of a floor are created. This involves measuring the dimensions of every room and retrieving the coordinates of the doors of it. This can be done either manually from a floor plan as depicted in Figure 4.5, as it was done for the RCS map or computer aided, which was not in the scope of this work. In the second step the floor object itself is instanced and initialized with a starting room of known world coordinates. Now gradually the other rooms are connected to the first room by specifying connecting doors between the rooms. The knowledge that a particular door in room A (given in world coordinates) matches a door in room B suffices to compute a shifting vector for room B which translates room B into the world coordinate system.

Following this approach the map of the RCS was converted to a floor plan for the Particle Filter with a resolution of $0.01m$, as depicted in Figure 4.7.

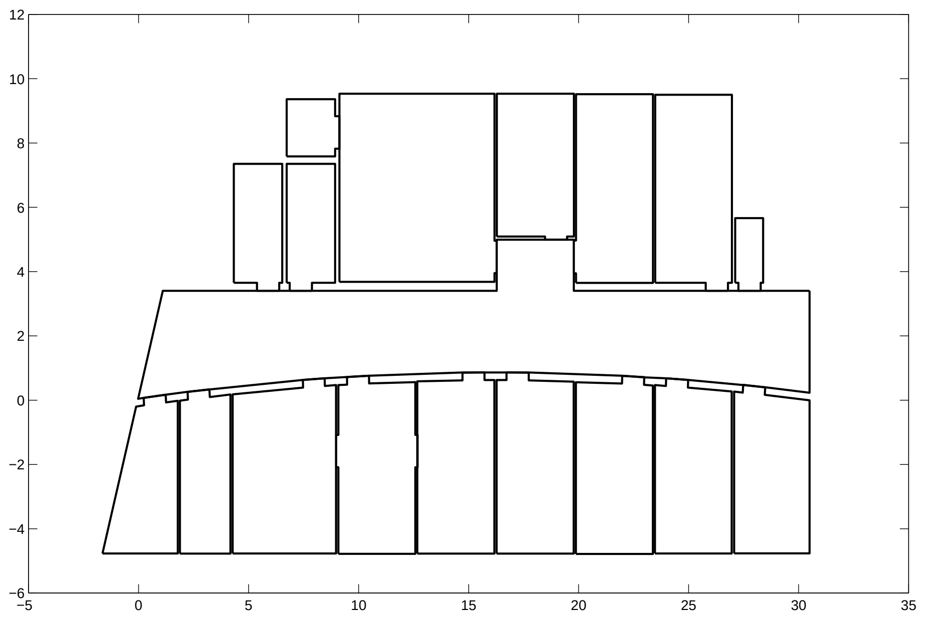


Figure 4.7: This is the floor plan of the RCS institute after the conversion to the format, which can be used for the Particle Filter. Note that doors are not drawn into the map. Therefore lines are not discontinued in the drawing, although this information is available for the algorithms.

4.3 Implementation of the Observation Model

As can be seen from the observation model Equation 4.5, it is necessary to generate two probabilities, which are combined with a logical AND. At first the Particle Filter needs a way to determine whether a specific position lies in the map or not. The getRoom algorithm described in the next section addresses that issue in an efficient way realizing $p(y'_k|\vec{x}_k)$. The second problem which has to be solved is to decide whether the transition from one room to an adjacent room is possible. Such a passage can only be valid if it leads through a door connecting both rooms, as expressed by $p(y''_k|y'_k, \vec{x}_{k-1}, \vec{x}_k)$, which is implemented in the usedDoor algorithm. Since the rooms are described by polygons the two methods rely on general geometry to compute the desired probabilities.

4.3.1 getRoom Algorithm

The observation model, proposed in section (4.2.1), requires to check whether a specific particle position lies within the map or not. Additionally transitions between rooms have to be recognized, to realize $p(y''_k|\vec{x}_{k-1}, \vec{x}_k)$. Since the Particle Filter is always executed on a single floor (see section (4.2.2)) it is sufficient to check for the validity of the position in the current floor instead of the whole map. As a floor is an aggregation of rooms the getRoom algorithm needs to find out in which room of the floor the particle is located or if it lies out of the map. Therefore at first the inRoom method is proposed which is able to find out whether a particle lies within a room or not.

As the walls of a room are modeled by a polygon, it was necessary to find an algorithm to test whether a position $\vec{r} = (x, y)^T$ lies within a polygon described by $\{\left(Poly_x[i], Poly_y[i]\right)^T ; i = 0 \dots n\}$ or not. A efficient method for this task was suggested in [22]. As shown in Figure 4.8, in principle the inRoom Algorithm 4.1 draws a line starting at the probing position and extending to infinity. The number of intersections of this line with the polygon is then counted. If the intersection-count is an odd number the point must lie inside the polygon. An even number of intersections will be observed if the point is out of the polygon.

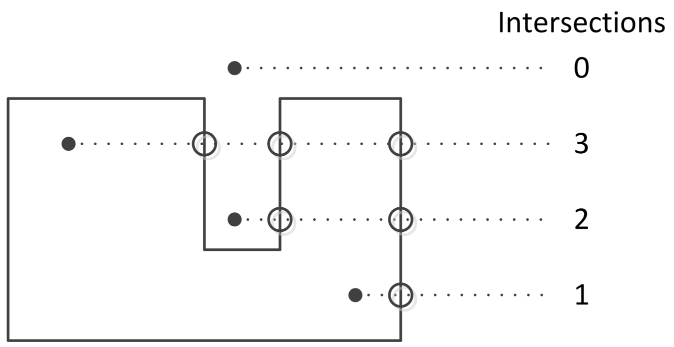


Figure 4.8: Principle of the inRoom algorithm.

Following this principle the inRoom algorithm is given as:

Algorithm 4.1 The inRoom algorithm tests whether a position $\vec{r}$ lies within a given room *Room* or not.

```
method inRoom$\left(\vec{r} = \begin{pmatrix} x & y \end{pmatrix}^T, Room\right)$
  $Polygon = Room.getPolygon = \{\begin{pmatrix} Poly_x[i] & Poly_y[i] \end{pmatrix}^T; i = 0 \ldots n\}$
  $inRoom$ =true
  $j = n$
  for $i = 0$ to $n$ do
    if $(Poly_y[i] \le y$ and $Poly_y[j] \ge y)$ or $(Poly_y[j] \le y$ and $Poly_y[i] \ge y)$ then
      if $Poly_x[i] + (y - Poly_y[i])/(Poly_y[j] - Poly_y[i]) * (Poly_x[j] - Poly_x[i]) \le x$ then
        $inRoom$ = not $inRoom$
      end if
    end if
    $j = i$
  end for
return $inRoom$
```

As pointed out before the getRoom Algorithm 4.2 needs to find the room in which the particle position is located. If no such room exists the particle is assumed to lie outside the floor.

Algorithm 4.2 The getRoom algorithm finds the room of a position $\vec{r}$ in case the position is within the outline of the given floor *Floor*. If not *null* is returned indicating that the position is outside *Floor*.

```
method getRoom$(\vec{r}, Floor)$
  for each $Room$ in $Floor$ do
    if $inRoom(\vec{r}, Room)$ = true then
      return $Room$
    end if
  end for
return $null$
```

4.3.2 usedDoor Algorithm

There are cases when the states of the system at time t_{k-1} and time t_k indicate that a particle changed the room. This can be observed by monitoring the room of the particles using the getRoom Algorithm 4.2 described beforehand. As depicted in section (4.2.1), this leads to the question of the validity of this transition, which is described by Equation (4.4) of the observation model.

This problem can be transformed into the calculation of the intersection point of two lines in 2D space. One line connects both posts of the door and the other line passes through the position of a particle in this time-step and in the one before. If both lines intersect between the posts of the given door it can be assumed that the particle moved through the door.

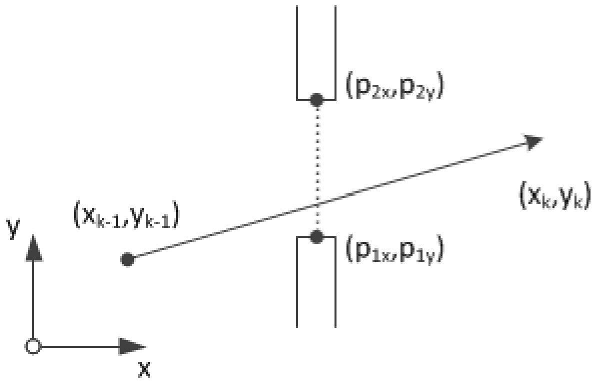


Figure 4.9: The geometric problem of the usedDoor algorithm.

Let $\vec{p}_{1/2} = \begin{pmatrix} p_{1/2x} & p_{1/2y} \end{pmatrix}^T$ be the position of the two post and let $\vec{r}_{k-1} = \begin{pmatrix} x_{k-1} & y_{k-1} \end{pmatrix}^T$ and $\vec{r}_k = \begin{pmatrix} x_k & y_k \end{pmatrix}^T$ be the position of a particle in two consecutive time-steps. This provides two linear equations which describe the two lines q and p, where q is the trajectory of the person from time t_{k-1} to t_k and p is the line connecting the posts of the door.

$$q : \vec{r} = \vec{r}_{k-1} + (\vec{r}_k - \vec{r}_{k-1}) \cdot \lambda_1 = \vec{r}_{k-1} + \Delta\vec{r} \cdot \lambda_1 \tag{4.6}$$

$$p : \vec{p} = \vec{p}_1 + (\vec{p}_2 - \vec{p}_1) \cdot \lambda_2 = \vec{p}_1 + \Delta\vec{p} \cdot \lambda_2 \tag{4.7}$$

The intersection point can now be derived by setting $\vec{r} = \vec{p}$. This yields the following linear system of equations:

$$\vec{p}_1 - \vec{r}_{k-1} = \begin{pmatrix} \Delta\vec{r} & -\Delta\vec{p} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} \Rightarrow \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} \Delta\vec{r} & -\Delta\vec{p} \end{pmatrix}^{-1} (\vec{p}_1 - \vec{r}_{k-1}) \tag{4.8}$$

From Equation (4.7) it is clear that if $0 < \lambda_2 < 1$ holds, the intersection point is between the posts of the door. In this case the trajectory of the particle is seen as valid.

The usedDoor Algorithm 4.3 given below uses Equation (4.8) to determine whether the passage from $\vec{x}_{k-1}$ in Room A to $\vec{x}_k$ in Room B leads through a door connecting these Rooms or not. Depending on the validity of the trajectory TRUE or FALSE is returned.

Algorithm 4.3 The usedDoor algorithm tests whether a trajectory defined by the two positions $\vec{r}_{k-1}$ and $\vec{r}_k$ leads through a door or not.

```
method usedDoor$\left(\vec{r}_{k-1} = \begin{pmatrix} x_{k-1} & y_{k-1} \end{pmatrix}^T, \vec{r}_k = \begin{pmatrix} x_k & y_k \end{pmatrix}^T, Floor\right)$
  $RoomA = getRoom(\vec{x}_{k-1}, Floor)$
  $RoomB = getRoom(\vec{x}_k, Floor)$
  $\Delta\vec{r} = \vec{r}_k - \vec{r}_{k-1}$
  for each Door connecting RoomA and RoomB do
      Get posts $\vec{p}_{1/2} = \begin{pmatrix} \vec{p}_{1/2x} & \vec{p}_{1/2y} \end{pmatrix}^T$ of Door
      $\Delta\vec{p} = \vec{p}_2 - \vec{p}_1$
      $\lambda_2 = \frac{\left((x_k - x_{k-1})\left(p_{1y} - y_{k-1}\right) - (y_k - y_{k-1})(p_{1x} - x_{k-1})\right)}{\det\begin{pmatrix} \Delta\vec{r} & -\Delta\vec{p} \end{pmatrix}}$
      if $0 \leq \lambda_2 \leq 1$ then
          return true
      end if
  end for
return false
```

Notice that this computationally expensive intersection calculation is only done for those doors connecting the two rooms. This functionality is easily realized using the list of doors stored within a room object.

4.4 Preparations for KLD-Sampling

As described in section (3.3), the KLD-sampling algorithm needs an estimate of the particle's spread in the state-space. This estimate can be obtained by dividing the state space into bins with a discrete extent in all three dimensions x, y and θ as depicted in Figure 4.10 and then counting the bins which contain particles. A bin occupies a space $\Delta x \times \Delta y \times \Delta\theta$, where Δx, Δy and $\Delta\theta$ are the discretization intervals, chosen small enough to describe the state space in an accurate but not too detailed way. Too much precision would not provide better results while boosting the storage usage.

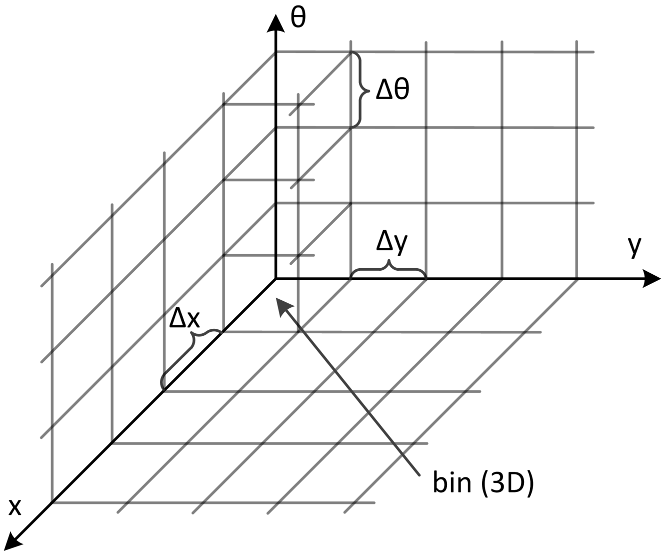


Figure 4.10: The state space is discretized in all three dimensions using bins of the size $\Delta x \times \Delta y \times \Delta\theta$.

Since the KLD algorithm only needs the number of bins with support of at least one particle, it is sufficient to store a 0 or a 1 for every bin, where 0 means no particle fell into that bin and 1 designates supported bins. This way of storing the bins requires only a 32bit 2D array b with the dimensions $[\frac{x_{max}-x_{min}}{\Delta x}] \times [\frac{y_{max}-y_{min}}{\Delta y}]$ increasing the memory efficiency. The θ-dimension is stored in the single bits of the 32bit integer value. This imposes the restriction $\Delta\theta \geq \frac{360°}{32} = 11.25°$ for the resolution of the heading of a pedestrian, which proved to be sufficient. The area occupied by a person can approximately be described by an ellipse with the major diameter being the shoulder length and the minor diameter being the depth of the human body. In average the shoulder length is given as $0.45m$ according to [24]. Since the orientation is not taken into consideration the person has to be modeled as the circumscribed circle of the ellipse. The major diameter of the ellipse gives the diameter of the circle: $0.45m$. Therefore it is convenient to chose Δx and Δy equal to $0.45m$.

The probability $1 - \delta$ describing the likelihood, that the KL divergence between the true and the approximated pdf of the system's state will stay below the threshold ϵ, was chosen to be

99%. Therefore $z_{0.99}$ in Equation (3.12) equals 2.3263 [10]. The threshold $\epsilon = 0.17$ was chosen by trial and error starting from 0.25 as proposed in [6].

A method for handling the state of a bin which is associated with the state $\vec{x}$ is given in Algorithm 4.4. If the bin did not have support before the update, the algorithm marks the bin as occupied by setting the bit to 1 and returns TRUE to indicate that a new bin has support now. This can be used to increase the number of supported bins q in accordance with the KLD-sampling method as given in section (3.3).

Algorithm 4.4 The updateBin algorithm returns whether the bin to which $\vec{x}$ belongs is occupied or not. In case the bin has been empty it is set to occupied.

```
method updateBin $\left(\vec{x} = (x, y, \theta)^T\right)$ // bin array $b[\frac{x_{max}-x_{min}}{\Delta x}] \times [\frac{y_{max}-y_{min}}{\Delta y}]$ is kept at global scale
    $i_x = floor(\frac{x-x_{min}}{\Delta x})$
    $i_y = floor(\frac{y-y_{min}}{\Delta y})$
    $i_\theta = floor(\frac{\theta-\theta_{min}}{\Delta \theta})$
    if bit $i_\theta$ of $b[i_x, i_y] == 0$ then
        set bit $i_\theta$ of $b[i_x, i_y] = 1$
        return true
    end if
return false
```

Notice that the bin matrix b is assumed to be stored at a global level, which eliminates the need to pass it every time the function is executed, for clarity reasons. The $floor$ function performs a conversion of a floating point to an integer value by rounding it off.

4.5 A KLD-Sampling Particle Filter Algorithm

The transfer function (4.2) of a pedestrian's system model has been given, the observation function and its implementation has been derived and and a helping method for the measurement of the KL distance has been shown. Now the KLD-sampling Particle Filter for pedestrian indoor localization can be given in Algorithm 4.5, which performs one update cycle of the estimate of the system's state under a new observation of the stride length and the orientation of the pedestrian obtained from the PiNav-software.

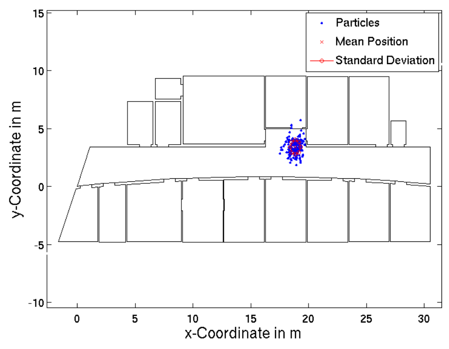
Algorithm 4.5 This KLD-sampling Particle Filter updates the particle distribution, describing the position and orientation of a pedestrian, from time t_{k-1} to time t_k.

```
method updateBeliefKLD$\left(S_{k-1} = \{s^i_{k-1};\ i = 0 \dots N_{k-1}\}, Floor, \mu_{\delta\theta_k}, \sigma^2_{\delta\theta_k}, \mu_{l_k}, \sigma^2_{l_k}\right)$
  $q = 0, j = 0$
  Set all elements of bin matrix $b$ to 0
  while $j \le \frac{1}{2\epsilon}\chi^2_{q-1,1-\delta}$ do
      Randomly take $s^i_{k-1}$ from $S_{k-1}$
      Sample $\delta\theta_k \sim \mathcal{N}(\mu_{\delta\theta_k}, \sigma^2_{\delta\theta_k})$
      Sample $l_k \sim \mathcal{N}(\mu_{l_k}, \sigma^2_{l_k})$
      Propagate particle $s^j_k = \vec{f}(s^i_{k-1}, \delta\theta_k, l_k)$ using Equation (4.2)
      $Room_{k-1} = getRoom(s^i_{k-1}, Floor)$
      $Room_k = getRoom(s^j_k, Floor)$
      if $Room_k \neq null$ and
      ($Room_{k-1} == Room_k$ or ($Room_{k-1} \neq Room_k$ and $usedDoor(s^i_{k-1}, s^j_k, Floor)$)) then
          $S_k = S_k \cup \left\langle s^j_k; 1\right\rangle$ weight is set to 1 since it is not used anyway
          if $updateBin(s^j_k)$ then
              $q++$
          end if
          $j++$
      else
          delete $s^j_k$
      end if
  end while
return $S_k = \left\{s^i_k; w^i_k\right\}$
```

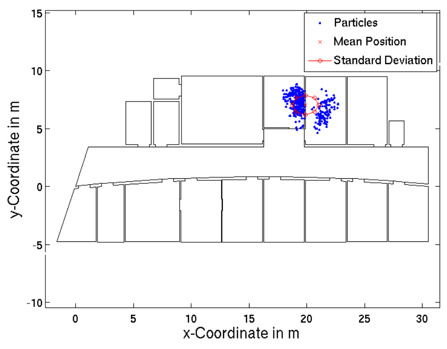
Notice, that the if-clause at the end of the while-loop realizes the observation model as derived in section (4.2.1), keeping the particles, if they are within the map and if the trajectory is valid, or deleting them. This is equal to assigning weights w^j_k of 1 in the former case and 0 in the latter. Since the weights are assigned indirectly and the surviving particles have all the same weights no resampling stage is needed to eliminate lightweight particles. The absence of a resampling stage also makes the intermediate particle distribution $\tilde{S}_k$ obsolete.

4.6 Position Estimation

A particle distribution is assumed to be localized if the particle distribution consists only of a single cluster of particles describing a unimodal state pdf in the dimensions x and y. Such a distribution exhibits no ambiguity in contrast to a multimodal pdf as depicted in Figures 4.11a and 4.11b.



(a) Unimodal particle distribution



(b) Multimodal particle distribution

Figure 4.11: The left figure depicts a unimodal belief of the position of the tracked person, whereas on the right side a multimodal distribution of particles is shown. In case of a distribution like the right one it can not clearly be decided in which room the person really is at that time.

It would be possible to determine the number of clusters in the particle distribution using sophisticated algorithms. But since this would be computationally demanding, the spread of the particle distribution is taken as a measure for the number of clusters. Clearly if only a single cluster exists the spread will be lower than in the case of multiple clusters. The standard deviation can be used as a measure for the spread of a random sample. On the one hand it is not prone to few outliers and on the other hand in the case of a multimodal distribution like in Figure 4.11b it will grow significantly in contrast to a unimodal distribution. The standard deviation of the particle distribution in the dimensions x and y is given as

$$\sigma_x = \sqrt{E\left[x^2\right] - E\left[x\right]^2} = \sqrt{\frac{\sum_{i=1}^{N_k} \left(x_k^i\right)^2 \cdot w_k^i}{\sum_{i=1}^{N_k} w_k^i} - \left[\frac{\sum_{i=1}^{N_k} x_k^i \cdot w_k^i}{\sum_{i=1}^{N_k} w_k^i}\right]^2} \tag{4.9}$$

$$\sigma_y = \sqrt{E\left[y^2\right] - E\left[y\right]^2} = \sqrt{\frac{\sum_{i=1}^{N_k} \left(y_k^i\right)^2 \cdot w_k^i}{\sum_{i=1}^{N_k} w_k^i} - \left[\frac{\sum_{i=1}^{N_k} y_k^i \cdot w_k^i}{\sum_{i=1}^{N_k} w_k^i}\right]^2}, \tag{4.10}$$

where $\left(x_k^i, y_k^i\right)$ is the position of the particle s_k^i.

If both the standard deviation in the x dimension σ_x and in the y dimension σ_y fall below predefined thresholds σ_x^{loc} and σ_y^{loc} the distribution is assumed to be unimodal and therefore localized. The thresholds were calibrated manually by evaluating several recorded paths. It was observed that with the threshold $\sigma_x^{loc} = \sigma_y^{loc} = 0.85m$ localized particle sets could robustly be distinguished from ambiguous particle distributions.

An estimator for the position $(\bar{X}_k, \bar{Y}_k)$ of the person at time t_k is the mean of the positions of the particles in the set S^k. These can be computed via the weighted sum as follows

$$\bar{X}_k = E\left[x_k^i\right] = \frac{\sum_{i=1}^{N_k} x_k^i \cdot w_k^i}{\sum_{i=1}^{N_k} w_k^i} \tag{4.11}$$

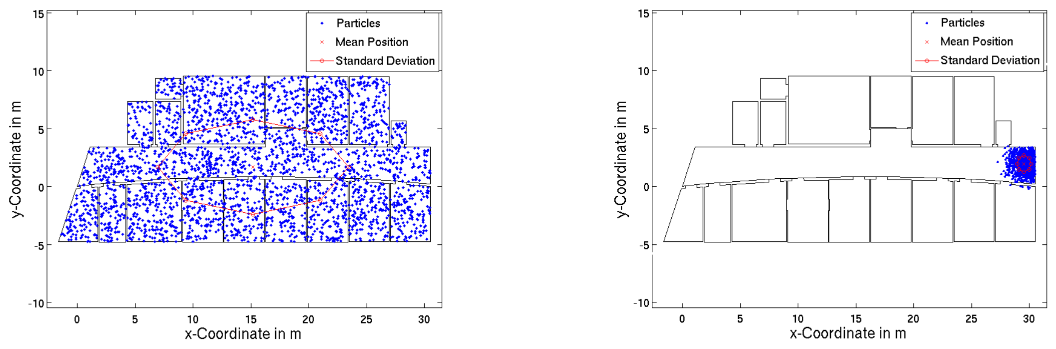
$$\bar{Y}_k = E\left[y_k^i\right] = \frac{\sum_{i=1}^{N_k} y_k^i \cdot w_k^i}{\sum_{i=1}^{N_k} w_k^i} \tag{4.12}$$

Note that the estimation can only be trusted in case of a localized particle distribution.

4.7 Practical Aspects

4.7.1 Prior Distribution

Algorithm 4.5 performs one cycle of the Particle Filter. Recursing from the prior particle distribution S_1 this method keeps track of the estimated evolution of a walking pedestrian's position on a specific storey. As S_1 describes $p(\vec{x}_1|Y^0)$, it represents the knowledge about the belief of the position of the pedestrian at the start-up of the Particle Filter. Without any further knowledge about the starting situation, S_1 would be an equal distribution of particles spread out over all floors of all buildings facing all possible directions. Since this is computationally intractable, a closer look has to be taken at how the prior distribution can be restricted.



(a) Equally distributed particle set (b) Around a position normally distributed particle set

Figure 4.12: Two examples for possible prior distribution are shown. On the left side no further information on the position of the person is available. Therefore the particles are equally distributed over the whole floor. On the right side the particle set is distributed around a known starting position at the entrance of the floor.

At first the prior S_1 is confined to the storey behind the door through which the person entered the building. Since the speed of the person is limited and the time of the entry can be estimated using the outdoor-path and the time of GPS reception loss occurring immediately after the entry, only a restricted area within the floor is part of the prior distribution S_1.

Second the orientation on entry into the building is limited to at least a range of 180°. Using the compass sensor to get an absolute measurement of the heading of the pedestrian, the orientation could be narrowed down even more.

4.7.2 Integration in the PiNav-Software

Now that the algorithm for one iteration of the Particle Filter has been derived, it has to be integrated into the PiNav-software, which delivers the necessary input parameters like stride length and orientation. As can be seen in Figure 4.13 the PiNav-Software does the step detection and the orientation filtering. As soon as a step is detected the inputs are passed to the Particle Filter, which performs one update of the believe of the system's state. After that the system starts over again.

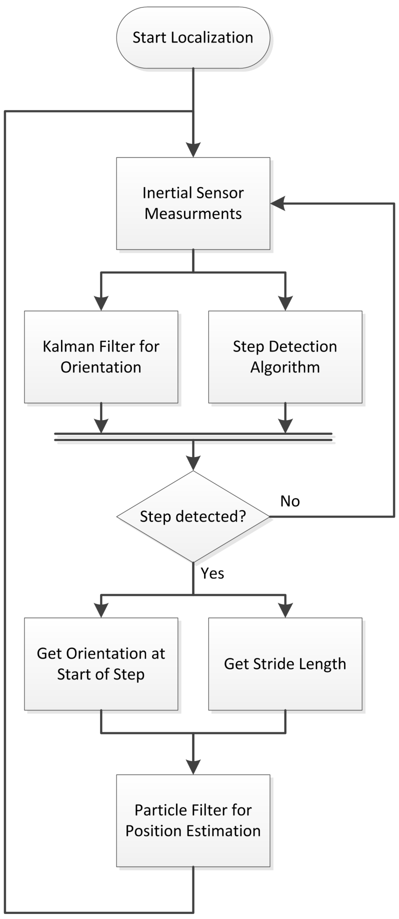

Start Localization
Inertial Sensor Measurments
Kalman Filter for Orientation
Step Detection Algorithm
Step detected?
No
Yes
Get Orientation at Start of Step
Get Stride Length
Particle Filter for Position Estimation

Figure 4.13: Flowchart of the integration of the Particle Filter into the system of the PiNav-software.

5 Improving the Observation Model

As already mentioned in section (4.2.1) it is important for the Particle Filter to have a detailed knowledge of the probability of a observation in a certain state, denoted by $p(y_k|\vec{x}_k)$, to be able to weight the particles according to their importance in representing the pdf of the system's state $p(\vec{x}_k|Y^k)$. So far the observation model described by Equation (4.5) delivered only the importance of 0% or 100%: Either the observation that a particle is in viable space was possible or it was impossible - and nothing in between. Naturally the question arises: Can the observation model get more accurate and if so how can this be done in the case of pedestrian localization?

5.1 The Accessibility Function

Accessibility will be referred to as a property of a certain state of a system which describes the likelihood of this state to be adopted by the system. In the case of a pedestrian, accessibility is a local quality of a space combined with the person's orientation in that area. It describes the possibility that a specific space is entered by a pedestrian with a specific heading during his walk.

The idea of the accessibility function $f_{Access}(\vec{x})$ is to store the accessibility of every state $\vec{x}$ in a floor. This is of interest because the accessibility of a person's state $\vec{x}$ can be written as $p(y|\vec{x})$, where y is the observation that a state is accessible. Thus $p(y|\vec{x})$ describes the degree of accessibility of a certain state $\vec{x}$. For example the space near a corner of a room has a low accessibility since a person would rarely walk into that area. Provided the whole accessibility function can be obtained this yields an observation model for the state of the system, which has the potential to describe $p(y|\vec{x})$ in much more detail than the model obtained by the geometric floor plan as introduced in section (4.2.1) resulting in a more precise way to weight the particles.

5.1.1 Pedestrian Behavior in Indoor Environments

An approximation of the accessibility function can be obtained by modeling the average human walking behavior in indoor environments. In literature [24] no general statement on the heading of a pedestrian in indoor environments is made, therefore it is assumed that the orientation of a walking person follows no preferences in specific situations and may be ignored when modeling pedestrian behavior. According to [24] the main factor that has been proven to affect human walking behavior is the proximity to walls or other obstacles.

In other words, the possibility that a person will walk into a specific area of a room is only determined by the position of this area in the room respectively its proximity to the walls of the room or to other obstacles. This means the accessibility of a state $\vec{x}$ is only dependent on the position of the area, described by the two coordinates x and y.

The values Weidmann gives in [24] are: In average a person walking in a corridor keeps a minimal distance of $0.25m$ to a wall made of concrete and $0.20m$ to a wall made of metal. Obstacles in a general environment are avoided with a gap of at least $0.10m$ between the pedestrian and the obstruction.

5.1.2 The Accessibility Map

Since the behavior of a pedestrian in indoor environments depends mainly on the position, as described in the previous section, the accessibility function can be confined to the two dimensions x and y yielding

$$f_{Access}(x, y) = \Lambda \in [0, 1] \tag{5.1}$$

where Λ is the degree of accessibility, from 0 denoting occupied positions to 1 meaning the space is fully accessible.

Since f_{Access} can not be obtained in an analytical form in general, the function has to be approximated. This is done by discretizing the x-y-plane in squares of $\Delta X \times \Delta Y$. For every square one value is stored, which describes $f_{Access}(X_i, Y_i)$, where (X_i, Y_i) denotes the position of the center of the square as depicted in Figure 5.1.

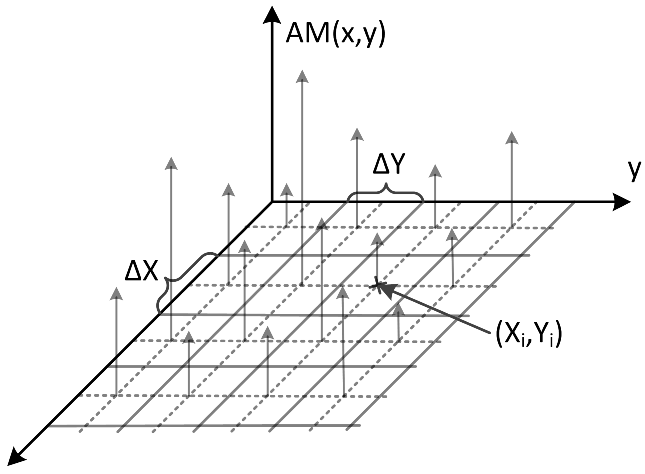


Figure 5.1: The position of the elements of the AM is located in the middle of a grid-cell. In the figure these are the intersection points of the dotted lines.

This grid based representation of f_{Access} will be referred to as accessibility map (AM). The value $f_{Access}(X_i, Y_i)$ which is stored in the AM is denoted $AM(X_i, Y_i) = \Lambda_i$.

All positions in the i^{th} square are now associated with the stored value $f_{Access}(X_i, Y_i) = AM(X_i, Y_i)$ as illustrated in Figure 5.1. This means the position (X_i, Y_i), where the accessibility Λ_i is known, corresponding to a position (x, y) can be found by

$$X_i = \frac{\Delta X}{2} + \left\lfloor \frac{x}{\Delta X} \right\rfloor \tag{5.2}$$

$$Y_i = \frac{\Delta Y}{2} + \left\lfloor \frac{y}{\Delta Y} \right\rfloor \tag{5.3}$$

where the expression $\lfloor x \rfloor$ rounds the value x to the lower integer value. Note that the index i of the grid elements is used to refer to a special element and does not imply any relations between different elements. In the following the formulation $AM(x, y)$ implies the use of Equation (5.2) to find (X_i, Y_i), which is then used to retrieve $AM(X_i, Y_i) = f_{Access}(X_i, Y_i)$. In short this means $AM(x, y)$ provides an approximation of the accessibility function $f_{Access}(x, y)$.

In the implementation a two-dimensional matrix of floating point values suffices to describe the AM. The elements of the matrix store the accessibility Λ_i of the positions (X_i, Y_i). The number of rows and columns of the matrix can be calculated as

$$columns = \left\lceil \frac{X_{max} - X_{min}}{\Delta X} \right\rceil \tag{5.4}$$

$$rows = \left\lceil \frac{Y_{max} - Y_{min}}{\Delta Y} \right\rceil, \tag{5.5}$$

where $\lceil x \rceil$ rounds the floating point value x up. The positions (X_{min}, Y_{min}) and (X_{max}, Y_{max}) denote the minimum and maximum x and y values of all positions describing the outline of the floor associated with the AM. Since pedestrians avoid proximity of less than $0.2m$ to walls, as described in section (5.1.1), it is practical to discretize the floor plan in squares of $\Delta X \times \Delta Y = 0.2m \times 0.2m$. For example a floor extending over $10m \times 100m = 1000m^2$ would result in a memory footprint of $1000 \times 4byte = 4kB$.

The Array element $[i_x, i_y]$ which corresponds to a position (x, y) can be found using

$$i_x = \left\lfloor \frac{x - X_{min}}{\Delta X} \right\rfloor \tag{5.6}$$

$$i_y = \left\lfloor \frac{y - Y_{min}}{\Delta Y} \right\rfloor. \tag{5.7}$$

The expression $AM[i_x, i_y]$ implies the use of the AM in its matrix representation as stored in the computer.

5.1.3 Incorporating Human Walking Behavior

The AM can be initialized from a given floor-plan according to the rules of human walking behavior as described in section (5.1.1). On the outside of the map and in the area occupied by walls the accessibility has to be zero whereas in distances larger than $0.4m$ to walls the accessibility function is set to 0.5. Only 50% accessibility is assumed and not 100%, since these areas could also be occupied by furniture or other obstacles. The 50% accessibility account for this uncertainty. The transition from 50% accessibility in the inner areas of the floor to zero accessible space outside the floor plan should be smooth reflecting the decreasing accessibility

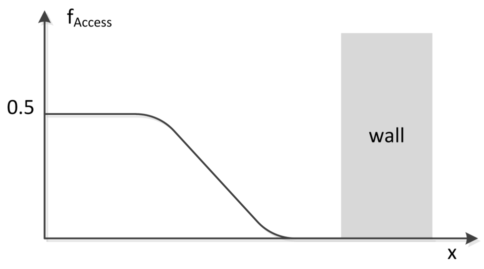


Figure 5.2: The accessibility decreases in the proximity of a wall and drops to zero within 0.2m distance to the wall.

of the areas in proximity to walls. A one dimensional example of how such an accessibility function could look like, is shown in Figure 5.2.

Creating an AM, which reflects pedestrian behavior, is done using the floor plan. At first the whole AM is set to an accessibility of 0.5. Second every grid element which is partly out of the floor outline is set to zero. This generates a coarse AM as depicted in Figure 5.3, which simply reflects the shape of the floor.

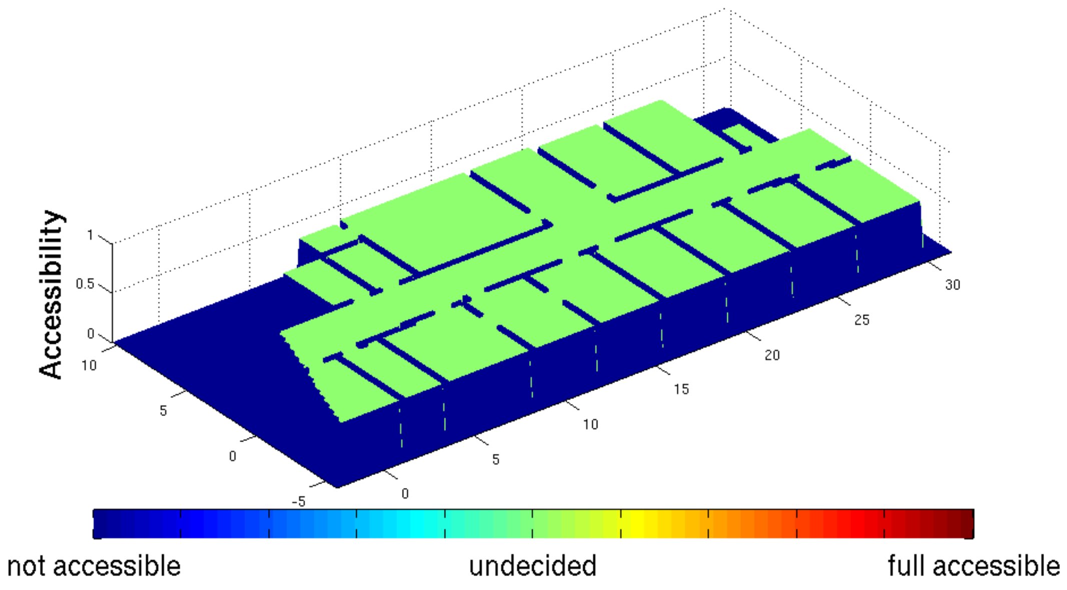


Figure 5.3: This coarse accessibility map reflects the underlying floor plan but does not add information yet. The grid consists of squares with an area of $\Delta x \times \Delta y = 0.2m \times 0.2m$.

In the next step the fully accessible parts of the AM are eroded setting the outermost accessible grid elements to zero. This is necessary for convolution of the accessibility grid with a Gaussian kernel matrix in two dimensions, which yields the smooth transitions between accessible and occupied space.

Let $h[i_k, i_l]$ be the approximation of a 2D Gaussian kernel as depicted in Figure 5.4 with the dimensions 9×9, a mean of 5 and a variance of 2.25:

$$h[i_k, i_l] = 10^{-3} \cdot \begin{pmatrix} 0.058 & 0.275 & 0.834 & 1.625 & 2.028 & 1.625 & 0.834 & 0.275 & 0.058 \\ 0.275 & 1.301 & 3.953 & 7.700 & 9.616 & 7.700 & 3.953 & 1.301 & 0.275 \\ 0.834 & 3.953 & 12.009 & 23.391 & 29.211 & 23.391 & 12.009 & 3.953 & 0.834 \\ 1.625 & 7.700 & 23.391 & 45.559 & 56.896 & 45.559 & 23.391 & 7.700 & 1.625 \\ 2.030 & 9.616 & 29.211 & 56.896 & 71.054 & 56.896 & 29.211 & 9.616 & 2.030 \\ 1.625 & 7.700 & 23.391 & 45.559 & 56.896 & 45.559 & 23.391 & 7.700 & 1.625 \\ 0.834 & 3.953 & 12.009 & 23.391 & 29.211 & 23.391 & 12.009 & 3.953 & 0.834 \\ 0.275 & 1.301 & 3.953 & 7.700 & 9.616 & 7.700 & 3.953 & 1.301 & 0.275 \\ 0.058 & 0.275 & 0.834 & 1.625 & 2.030 & 1.625 & 0.834 & 0.275 & 0.058 \end{pmatrix}$$

Note that their elements are chosen to add up to 1, which leaves the height of the AM in inner areas of the rooms unchanged at 0.5.

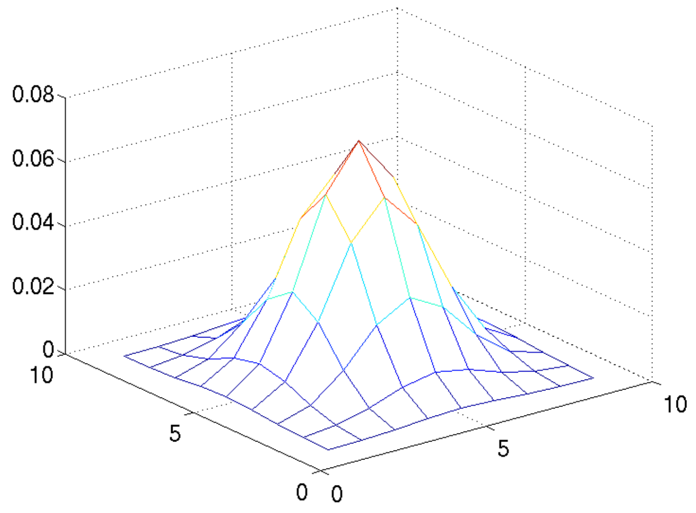


Figure 5.4: Plot of the Gaussian kernel for the convolution.

The discrete two-dimensional convolution of the AM matrix with the Gaussian kernel is expressed by

$$AM[i_x, i_y] = \sum_{l=i_y-4}^{i_y+4} \sum_{k=i_x-4}^{i_x+4} AM[k, l] \cdot h[i_x + 4 - k, i_y + 4 - l]. \tag{5.8}$$

Applying the convolution yields an AM with smooth transitions from zero accessibility behind walls to 0.5 accessibility in the inner areas, as depicted in Figure 5.5. Without the erosion step the convolution would generate grid elements with an accessibility unequal to zero outside the outline of the floor, which is not wanted, since these areas are not accessible at all.

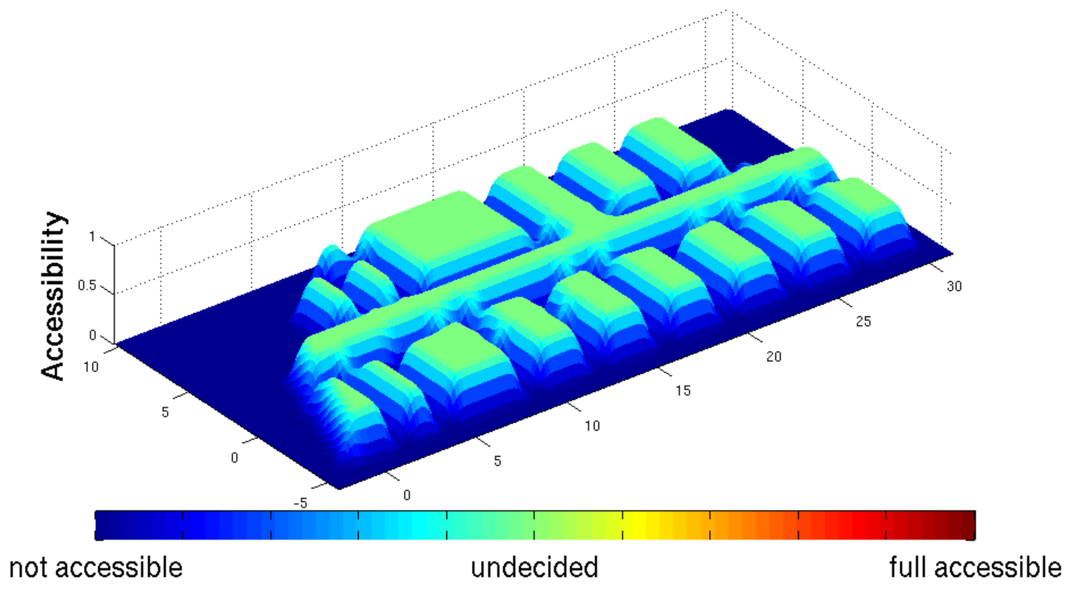


Figure 5.5: The smoothed accessibility map now reflects the human walking behavior near walls, in that the accessibility drops in proximity of walls.

In practice every time a floor is changed in the Particle Filter, the corresponding AM has to be loaded as well. Since the AM of a specific floor does not change, the AM can be created in advance. The generation of the AMs for the floors is only necessary once and can be processed offline on a high speed PC. Therefore standard algorithms for the erosion and the convolution have been implemented.

5.1.4 A Particle Filter using the Accessibility Map

Since it is possible to enrich the geometric observation model with the accessibility map, a modified KLD-sampling SIR Particle Filter is given with Algorithm 5.1, which weights the particles according to the accessibility of their positions. The accessibility map is created in advance to reflect pedestrian walking behavior as described in section (5.1.3) To provide equally weighted particles for the next cycle of the algorithm, resampling of the weighted particles becomes inevitable in contrast to Algorithm 4.5, which uses only the geometric observation model.

Algorithm 5.1 This KLD-sampling Particle Filter uses an Accessibility Map reflecting human walking behavior to weight the particles which are propagated from time t_{k-1} to t_k.

```
method updateBeliefKLDAM$\left(S_{k-1} = \{s^i_{k-1};\ i = 0 \dots N_{k-1}\}, Floor, \mu_{\delta\theta_k}, \sigma^2_{\delta\theta_k}, \mu_{l_k}, \sigma^2_{l_k}\right)$
  $q = 0,\ j = 0$
  Empty all bins $b$
  $w_{SUM} = 0$
  while $j \leq \frac{1}{2\epsilon}\chi^2_{q-1,1-\delta}$ do
      Randomly take $s^i_{k-1}$ from $S_{k-1}$
      Sample $\delta\theta_k \sim \mathcal{N}(\mu_{\delta\theta_k}, \sigma^2_{\delta\theta_k})$
      Sample $l_k \sim \mathcal{N}(\mu_{l_k}, \sigma^2_{l_k})$
      Propagate particle $\tilde{s}^j_k = \vec{f}(s^i_{k-1}, \delta\theta_k, l_k)$ // using Equation (4.2)
      $Room_{k-1} = getRoom(s^i_{k-1}, Floor)$
      $Room_k = getRoom(\tilde{s}^j_k, Floor)$
      if $Room_k \neq NULL$ and
      ($Room_{k-1} == Room_k$ or ($Room_{k-1} \neq Room_k$ and $usedDoor(s^i_{k-1}, \tilde{s}^j_k, Floor)$)) then
          Evaluate particle weight $\tilde{w}^j_k \sim AM_k(\tilde{s}^j_k)$
          $\tilde{S}_k = \tilde{S}_k \cup \left\langle \tilde{s}^j_k; \tilde{w}^j_k \right\rangle$
          if $updateBin(s^j_k)$ then
              $q++$
          end if
          $j++$
          $w_{SUM} = w_{SUM} + \tilde{w}^j_k$
      else
          delete $s^j_k$
      end if
  end while
  for $i = 1\ :\ N_k$ do
      Normalize weights: $\tilde{w}^i_k = \frac{\tilde{w}^i_k}{w_{SUM}}$
  end for
  $S_k = $ resampleON$\left(\tilde{S}_k\right)$ // using Algorithm 3.2
return $S_k = \left\{s^i_k; w^i_k\right\}$
```

5.2 Learning Accessibility

When looking at the way the map is represented it is obvious, that much detail is left out. Only walls are taken into consideration as obstacles ignoring any pieces of furniture like desks, lockers and bookcases. This general representation is sufficient for coarse localization in buildings. But adding more details to the depiction of a floor would enable higher accuracy in the localization. Besides the fact that it would need significant effort to add such details to the map in advance, these obstacles are more or less movable, which makes it impractical to fix them to an area in the environment. Therefore the question arises how more details can be added to the observation model from the data obtained by the Particle Filter.

The key idea behind the solution proposed in the following is that human moving patterns in a room will reflect the accessibility of the different areas, since a person would for example not climb across a desk if it were possible to walk around it. A possible walking pattern of a pedestrian in a room is depicted in Figure 5.6.

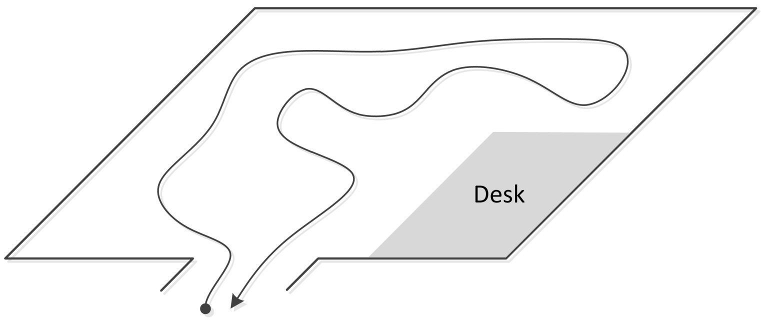


Figure 5.6: The walking pattern of the person reflects the accessibility of space in the room, since areas under and next to the desk are avoided.

Given several such patterns for the same room it would become obvious that the area of the desk is not accessed by the pedestrians. Using this observation the AM could be updated to have a higher accessibility in the free area than in the region of the desk, which would increase the weights added to particles in the free areas.

This problem is similar to the question of SLAM, since in parallel to the localization of the pedestrian it should be recorded, which places were accessible and which were not. The difference to the SLAM approaches known from robotics is, that unlike in robotics no additional sensors like laser-range finders can be carried around by a pedestrian. So the features which have to be extracted from the environment for SLAM [5] can only be obtained from the moving patterns [20] and the behavior of the person. The difference to the hard SLAM problem examined in [20] is that using the coarse floor outline in combination with the Particle Filter, a vague localization is possible.

5.2.1 The Accessibility Map as a Radial Basis Function Network

The aim is to update the AM according to the walking patterns of a person through the corresponding floor. In other words the AM should be enabled to learn the accessibility of its grid elements from the estimated position of the person wearing the localization system.

To derive the learning rule for accessibility, at first a reinterpretation of the AM as a radial basis function network (RBFN) [11] is necessary. This kind of three layer neural network can approximate arbitrary functions. The RBFN concept originates from the theory of Neuronal Networks with the difference that the activation functions of the neurons in the second layer produce significant outputs only locally around a predefined center. Therefore the output of the RBFN only depends on few inputs from the second layer. This property makes them suitable for online learning, since in the learning phase only the few weights of the neurons in the second layer, which contributed to the output have to be adapted. All the other weights are left unchanged. Therefore the function described by the RBFN will change in the area where new knowledge was acquired and it will remain unchanged in areas where no new information has been given, thus keeping the memory of the previously learned course of the function. This kind of learning behavior suits well for the objective to learn the accessibility of space, since this has to be learned locally.

The AM can be interpreted as a RBFN of the structure depicted in Figure 5.7.

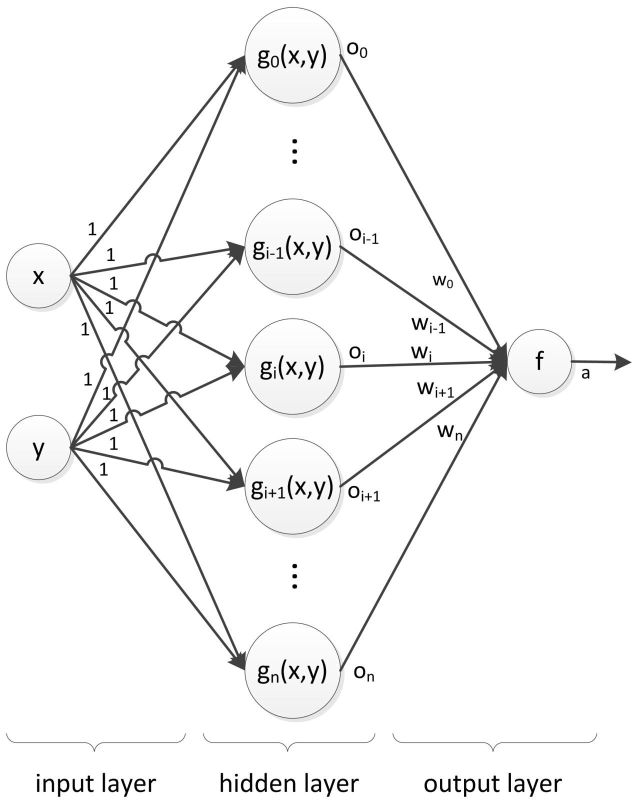


Figure 5.7: The AM interpreted as a RBFN.

The RBFN has two input neurons for the x and y coordinate and one output neuron with a linear activation function

$$f(o_1, \ldots, o_n) = f\left(\sum_{i=1}^{n} w_i \cdot o_i\right) = \sum_{i=1}^{n} w_i \cdot o_i. \tag{5.9}$$

Every grid element i of the AM is now interpreted as a neuron i in the hidden layer, with the activation function $g_i(x, y)$ as depicted in Figure 5.8 and mathematically defined by

$$o_i = g_i(x, y) = \begin{cases} 1 & \text{if } (X_i - \frac{\Delta X}{2} < x < X_i + \frac{\Delta X}{2}) \text{ and } (Y_i - \frac{\Delta Y}{2} < y < Y_i + \frac{\Delta Y}{2}) \text{ holds} \\ 0 & \text{otherwise} \end{cases}, \tag{5.10}$$

where (X_i, Y_i) denotes the position of the center of the i^{th} grid element in the AM. The Gaussian activation function which is normally used in RBFNs is here approximated by $g_i(x, y)$, to decrease computational efforts. Note that the weights of the connections from the input neurons to the hidden layer neurons are set to 1 handing over the input (x, y) to the hidden layer neurons without further weighting.

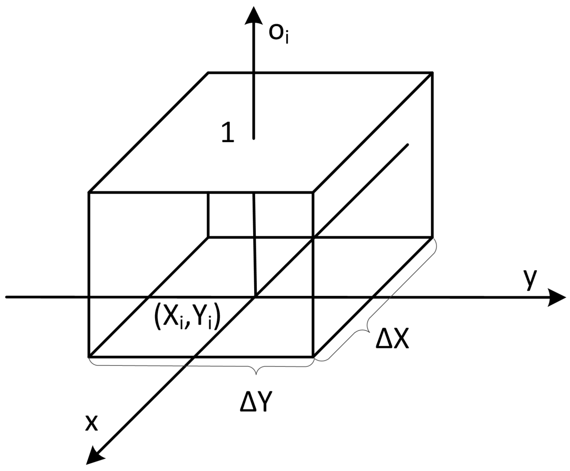


Figure 5.8: The activation function of the i^{th} neuron of the hidden layer of the RBFN. Only in a rectangle of the dimension $\Delta X \times \Delta Y$ around the position (X_i, Y_i) of the neuron an output $o_i = 1$ is produced. Inputs outside this rectangle do not activate the neuron hence producing an output of zero.

Since the grid elements are equally distributed over the floor with a distance of ΔX respectively ΔY to the neighbor elements, the activation functions do not overlap. Therefore a given input (x, y) always activates only a single neuron in the hidden layer.

The accessibility Λ_i in the AM is a property of the grid element, which can now be interpreted as the weight w_i of the connection from the i^{th} hidden layer neuron to the output neuron, since o_i is either one or zero.

5.2.2 The Learning Accessibility Map

Given this new interpretation of the AM as a RBFN, training techniques for neuronal networks can be applied to learn the accessibility function from walking patterns of pedestrians. Since the connections between the input and the hidden layer have fixed weights, only the weights between the hidden layer neurons and the output neuron, which represent the accessibility of the area associated with a hidden neuron, need to be trained. A single layer of neurons can be trained using the delta rule [13], which modifies the weights w_i in order to minimize the difference between the output of the neuronal network and the desired output. The delta rule utilizes a gradient descend method to find parameters such that the error between the training values and the output of the neuronal network is minimized.

Let $\alpha \in [0, 1]$ be the learning rate of the neuronal network and Λ^* the desired output for a given position (x, y), then the delta rule for the RBFN is

$$w_i = w_i + \alpha \cdot o_i \cdot (\Lambda^* - f(o_1, \dots, o_n)) = w_i - \alpha \cdot o_i \cdot \left(\Lambda^* - \sum_{i=1}^{n} w_i \cdot o_i \right), \tag{5.11}$$

where the o_i depend on the input (x, y) according to Equation (5.10). This usually means that for every new Λ^* which has to be trained, all weights w_i have to be updated. As mentioned beforehand the special form of the neurons' activation function in the hidden layer and their alignment ensures that only a single neuron I is activated for a given input, resulting in an $o_I = 1$ and $o_i = 0 \;\forall\, i \neq I$. This yields the learning rule

$$w_i = \begin{cases} w_i + \alpha \cdot (\Lambda^* - w_i) & \text{for } i = I \\ w_i & \text{for } i \neq I \end{cases} \tag{5.12}$$

which implies that only the weight w_I associated with the activated neuron, has to be trained ensuring high performance for the learning process.

The last unknown variable is the desired accessibility Λ^* of a position (x, y). Clearly only positions on the trajectory of the pedestrian, can be seen as accessible. Since these positions can only be estimated, as described in section (4.6), not the full accessibility can be assigned to them. Therefore the degree of accessibility of a visited position is chosen to be dependent on the accuracy of the position estimate. As the standard deviation of the position estimate is calculated anyway to determine whether the distribution is localized, it can be exploited as a measure for the accuracy of the mean position without further computational efforts.

Assuming that the particle distribution is localized which means $\sigma < \sigma^{loc}$, the accessibility of the estimated position is set to

$$\Lambda^* = 1 - \frac{\sigma^{loc} - 0.5\sigma^{min} - 0.5\sigma}{\sigma^{loc} - \sigma^{min}} \in [0.5, 1] \tag{5.13}$$

where $\sigma^{loc} = \max\left(\sigma_x^{loc}, \sigma_y^{loc}\right)$, $\sigma^{min} = \min\left(\sigma_x^{min}, \sigma_y^{min}\right)$ and $\sigma = \max\left(\sigma_x, \sigma_y\right)$. Equation (5.13) will output accessibility estimates ranging from 0.5 to 1. The lower bound is 0.5 in case $\sigma = \sigma^{loc}$, which is the case when a particle distribution starts to be classified as localized, indicating that the position estimate is rather vague and cannot be trusted too much. The upper bound of an accessibility of 1 is achieved if the standard deviation of the position estimate equals σmin, which is the standard deviation at which the position estimate is assumed to be accurate enough to set the accessibility to 100%. In between those boundaries the accessibility is inversely proportional to the positioning accuracy measured by the standard deviation. Referring to the property of the normal distribution $\mathcal{N}(\mu, \sigma)$, that a sample drawn from it will be in the range of $\mu \pm 2\sigma$ with a likelihood of $2 \cdot 34.1\% + 2 \cdot 13.6\% = 95.4\%$, σ_{min} was set to $0.4m$. This means that at least 95.4% of all particles have to be located within a circle of $0.8m$ around the estimated position to assign full accessibility to that position.

Inserting Equation (5.13) into Equation (5.12) yields the accessibility training rule given a position estimate and its standard deviation:

$$w_i = \begin{cases} w_i + \alpha \cdot \left(\frac{\sigma^{loc} - 0.5\sigma^{min} - 0.5\sigma}{\sigma^{loc} - \sigma^{min}} - w_i \right) & \text{for } i = I \\ w_i & \text{for } i \neq I \end{cases}. \tag{5.14}$$

Transferring back to the grid based interpretation of the AM this learning rule can be formulated as

$$AM_{k+1}(x, y) = AM_k(x, y) + \alpha \cdot \left(\frac{\sigma^{loc} - 0.5\sigma^{min} - 0.5\sigma}{\sigma^{loc} - \sigma^{min}} - AM_k(x, y) \right) \tag{5.15}$$

where $AM_k(x, y)$ denotes the accessibility of the grid element associated with the position (x, y) at time t_k.

In case the particle distribution is localized in two subsequent timesteps, the whole trajectory from the position at time t_{k-1} to the position at time t_k is seen as valid and all grid elements which lie on that path are updated according to the learning rule (5.15). The standard deviation is interpolated linearly between σ_{k-1} and σ_k to generate the necessary input for learning the accessibility of the positions on the path yielding

$$\sigma(l) = \sigma_{k-1} + \frac{l \cdot (\sigma_k - \sigma_{k-1})}{L}, \tag{5.16}$$

where L is the distance between the two positions and $l \in [0, L]$ denotes the distance traveled on the path, as depicted in Figure 5.9.

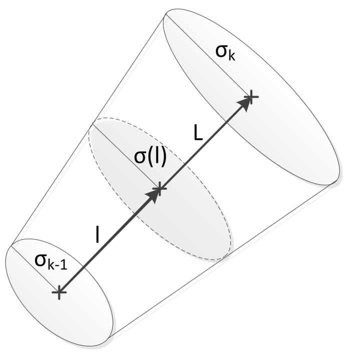


Figure 5.9: The standard deviation is linearly interpolated between two positions at time t_{k-1} and t_k.

Algorithm 5.2 performs learning of the accessibility of space visited by a pedestrian on his walk.

Algorithm 5.2 An update of the accessibility map is performed if the particle distribution is localized. In case the distribution in the timestep before was localized as well, the accessibility is updated on the path from the estimated position at time t_{k-1} to the position at time t_k.

```
method learnAccessibility($AM_k, S_k, S_{k-1}$)
  $AM_{k+1} = AM_k$
  if $S_k$ is localized then
      $pos_k = \left(mean_x(S_k), mean_y(S_k)\right)^T$
      $\sigma_k = \max(\sqrt{var_x(S_k)}, \sqrt{var_y(S_k)})$
      if $S_{k-1}$ was localized then
          $pos_{k-1} = \left(mean_x(S_{k-1}), mean_y(S_{k-1})\right)^T$
          $\sigma_{k-1} = \max(\sqrt{var_x(S_{k-1})}, \sqrt{var_y(S_{k-1})})$
          $L = \|pos_k - pos_{k-1}\|_2$
          for all positions $(x, y)$ on path from $pos_{k-1}$ to $pos_k$ do
              $l = \|(x, y)^T - pos_{k-1}\|_2$
              $\sigma = \sigma_{k-1} + \frac{l \cdot (\sigma_k - \sigma_{k-1})}{L}$,
              $AM_{k+1}(x, y) = AM_k(x, y) + \alpha \cdot \left(\frac{\sigma^{loc} - 0.5\sigma^{min} - 0.5\sigma}{\sigma^{loc} - \sigma^{min}} - AM_k(x, y)\right)$
          end for
      else
          $AM_{k+1}(x, y) = AM_k(x, y) + \alpha \cdot \left(\frac{\sigma^{loc} - 0.5\sigma^{min} - 0.5\sigma}{\sigma^{loc} - \sigma^{min}} - AM_k(x, y)\right)$
      end if
  end if
return $AM_{k+1}$
```

The expression $\|(x, y)\|_2$, used in the algorithm, denotes the euclidean norm of the vector (x, y). The mean positions pos_{k-1} and pos_k are calculated using Equation (4.11) whereas the standard deviations σ_{k-1} and σ_k are computed according to Equations (4.9).

An example for an AM trained with one walk and a learning rate of $\alpha = 0.4$ is shown in Figure 5.10. The AM learned from a trajectory of a pedestrian starting at the left. The path leads in and out of the middle room and then to the right end of the corridor. This can be seen from the increased accessibility in the areas where the particle distribution was localized and thus accessibility was learned. Interesting is the accessibility in the middle room, since it decreases in that area. This indicates that the distribution had a higher standard deviation there.

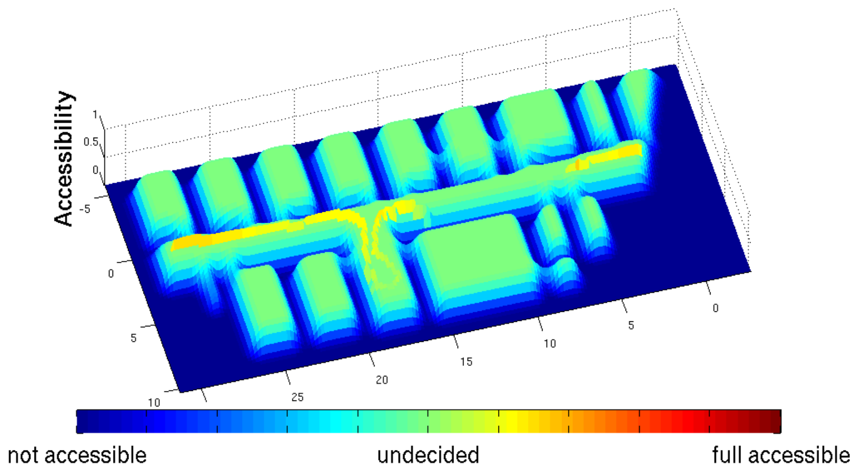


Figure 5.10: The AM depicted was trained by a single path which started at the left, lead in and out of the middle room and ended at the right side of the floor.

As stated at the beginning the obstacles in a room might change their positions or orientation from time to time. This means learning of accessibility is not enough. It is also necessary to forget about the accessibility of space in case the area has not been visited for a longer period of time. One way to implement this feature would be to periodically train all AM elements the pure accessibility map as obtained from the simulated walking behavior of pedestrians, yielding a process of forgetting learned accessibility. This would enable the accessibility map to represent also dynamically changing environments.

5.2.3 A Particle Filter using the learning Accessibility Map

The learning Algorithm 5.2 derived in the previous section is integrated into the Particle Filter Algorithm 5.1 yielding a learning accessibility map, which is used to weight the particles according to the degree of contribution to the state pdf $p(\vec{x}|Y^k)$.

Algorithm 5.3 This KLD-sampling Particle Filter uses a Accessibility Map, which is enabled to learn accessibility of visited space, to weight the particles which are propagated from time t_{k-1} to t_k.

```
method updateBeliefKLDLAM$\left(S_{k-1} = \{s^i_{k-1};\ i = 0 \dots N_{k-1}\}, Floor, \mu_{\delta\theta_k}, \sigma^2_{\delta\theta_k}, \mu_{l_k}, \sigma^2_{l_k}\right)$
  $q = 0,\ j = 0$
  Empty all bins $b$
  $w_{SUM} = 0$
  while $j \leq \frac{1}{2\epsilon}\chi^2_{q-1,1-\delta}$ do
      Randomly take $s^i_{k-1}$ from $S_{k-1}$
      Sample $\delta\theta_k \sim \mathcal{N}(\mu_{\delta\theta_k}, \sigma^2_{\delta\theta_k})$
      Sample $l_k \sim \mathcal{N}(\mu_{l_k}, \sigma^2_{l_k})$
      Propagate particle $\tilde{s}^j_k = \vec{f}(s^i_{k-1}, \delta\theta_k, l_k)$ // using Equation (4.2)
      $Room_{k-1} = getRoom(s^i_{k-1}, Floor)$
      $Room_k = getRoom(\tilde{s}^j_k, Floor)$
      if $Room_k \neq NULL$ and
      $(Room_{k-1} == Room_k$ or $(Room_{k-1} \neq Room_k$ and $usedDoor(s^i_{k-1}, \tilde{s}^j_k, Floor)))$ then
          Evaluate particle weight $\tilde{w}^j_k \sim AM_k(\tilde{s}^j_k)$
          $\tilde{S}_k = \tilde{S}_k \cup \left\langle \tilde{s}^j_k; \tilde{w}^j_k \right\rangle$
          if $updateBin(s^j_k)$ then
              $q{+}{+}$
          end if
          $j{+}{+}$
          $w_{SUM} = w_{SUM} + \tilde{w}^j_k$
      else
          delete $s^j_k$
      end if
  end while
  for $i = 1\ :\ N_k$ do
      Normalize weights: $\tilde{w}^i_k = \frac{\tilde{w}^i_k}{w_{SUM}}$
  end for
  $AM_{k+1}$ = learnAccessibility$\left(AM_k, \tilde{S}_k, S_{k-1}\right)$ // using Algorithm 5.2
  $S_k$ = resampleON$\left(\tilde{S}_k\right)$ // using Algorithm 3.2
return $S_k = \left\{s^i_k; w^i_k\right\}$
```

6 Evaluation

In the previous chapters (3) to (4), at first the theory for Bayesian filters was introduced, yielding the concept of a SIR Particle Filter. This Filter type was expanded to incorporate KLD-Sampling, which adapts the size of the particle set dynamically. In chapter (5), a Accessibility Map was proposed, which describes human walking behavior. In an attempt to learn accessibility from pedestrian walking patterns, a learning rule for the AM was derived. Exploiting these fundamentals three different Particle Filters for pedestrian tracking were proposed:

- The KLD-sampling Particle Filter using a geometric floor plan (Algorithm 4.5)
- The KLD-sampling Particle Filter using an accessibility map (Algorithm 5.1)
- The KLD-sampling Particle Filter using a learning accessibility map (Algorithm 5.3)

In general there are two different tasks which have to be performed by the Particle Filter: localization and tracking of a pedestrian. The first functionality denotes the process of finding the position of the person starting from vague previous knowledge about the location in a building. The second one aims to estimate the subsequent trajectory of a pedestrian starting from a unimodal position pdf. Obviously without further knowledge at first localization has to be performed yielding a belief of the persons position which is free from ambiguity. From this position estimate the trajectory can be tracked.

In the following at first the computational performance during a typical walk starting from an unknown position is evaluated. After that the performance of the three different Particle Filters is examined. Since the two tasks, localization and tracking, are separated and of fixed order in time, they were evaluated individually.

6.1 General System Setup

To enable faster testing of the algorithms and the software system an 1.6GHz laptop with 512MB RAM was used. Since there is a cross-compiler for the system on the computing device the software can be deployed on the mobile device as well.As mentioned in chapter (2), the foot-mounted sensor was not yet completely integrated into the software architecture and is therefore not used. This poses no critical restriction, since the pocket sensing device can deliver stride information and orientation values on its own. All in all the setup as depicted in Figure 6.1 was used for the development and evaluation of the algorithms.

The test walks were done on the third floor of building three on the TU Munich main campus, where the Institute for Real-Time Computer Systems is located. The corresponding CAD floor

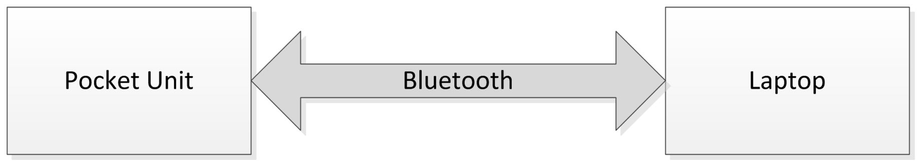


Figure 6.1: The PiNav-System as used for the evaluation consisting of pocket unit and a laptop replacing the computing device.

plan is depicted in Figure 4.5. Using a CAD viewer the rooms and doors were digitalized for the PiNav-System with a resolution of $0.01m$, as depicted in Figure 4.7. Since the transitions between floors cannot be tracked, due to the lack of a filter in the PiNav-software extracting the elevation of the person, the evaluation of the Particle Filters was confined to this single storey.

The step-detection algorithm in the PiNav-software could not yet provide stride length information. Therefore the step length has been fixed to a length of $60cm$ in the test walks. This could be achieved by affixing post-its to the ground at the positions of the single foot-steps using a yard stick. Therefore the mean value of the stride length l_k was set to $\mu_{l_k} = 0.6m$ with a standard deviation of $\sigma_{l_k} = 0.2m$, which accounts for the errors in the calibration of the trajectory and the errors induced by imperfect strides. The difference in the heading between two steps $\delta\theta_k$ could be retrieved from the Kalman Filter in the PiNav-System yielding $\mu_{\delta\theta_k}$. The standard deviation of $\delta\theta_k$ was fixed to $\sigma_{\delta\theta_k} = 3°$, which describes the perturbations of the orientation estimation.

The walks which have been evaluated were recorded using the pocket sensing device and the PiNav-software. After that the data was fed into the PiNav-software using the replay function to estimate the trajectory with the Particle Filter. All generated outputs like the particle sets at all timesteps, the estimated positions and their standard deviations were stored. These results were visualized and evaluated using Matlab.

6.2 Computational Performance

As new sensor information becomes available after every step, the Particle Filter has to be evaluated within one step for on-line localization. According to [24] the maximum walking speed is $1.69\frac{m}{s}$ at a stride length of $0.65m$ which results in a maximum walking frequency of a healthy human of $2.68Hz$. This imposes a deadline for the filter algorithm of a maximum execution time of $373ms$ per cycle.

For large numbers of particles this deadline becomes critical if the algorithms in the Particle Filter are of higher time-complexity. Therefore care has been taken to incorporate only algorithms with $O(N)$ time-complexity in the KLD-sampling Particle Filters: The main loop, in which prediction and update are performed, is of $O(N)$. Also the normalization of the particle weights and the resampling, which is only necessary for the Particle Filters using an AM, is $O(N)$. Calculating the mean of the particle positions in the current particle set, which yields a position estimate $(\bar{X}_k, \bar{Y}_k)$ as described in section (4.6), can be done in $O(N)$ time as well. The standard deviation of the estimated position has to computed as a measure for the distribution's spread. As can be seen from Equation (4.9), this can also be done with $O(N)$ time-complexity.

Figure 6.2 depicts the normalized curves of the evolution of the particle set size N_k and the time t_U taken for the update of the particle set by a KLD-Sampling Particle Filter. As can be seen from this plot, besides some jitter induced by the operating system, the relation between N_k and t_U is linear, which confirms that the time complexity of the Particle Filter is $O(N)$.

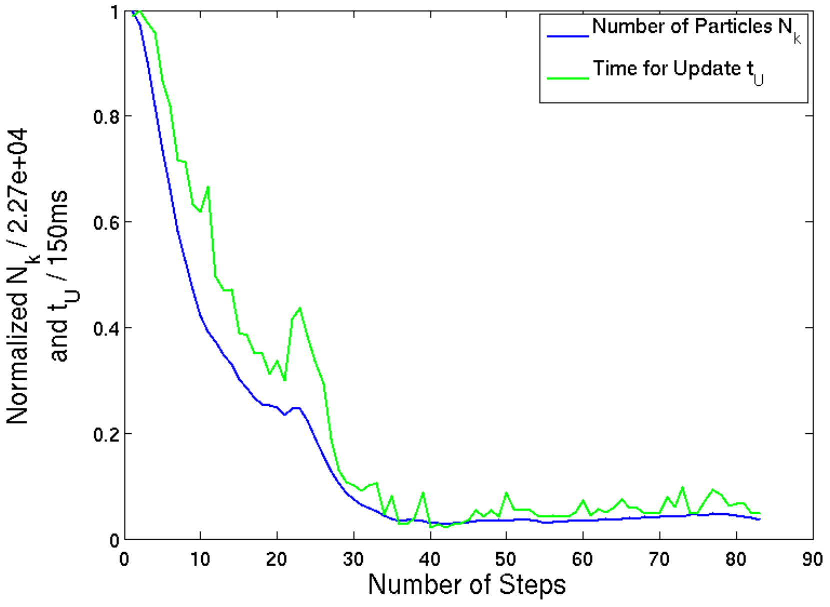


Figure 6.2: A typical plot of the normalized execution time t_U and number of particles N_k. This plot was created from an walk trough the RCS floor starting from an equally distributed particle set as depicted in Figure 4.12a on page 35. The PiNav-System was run on a 1.6GHz processor with 512MB RAM.

In the first phase after the initialization of the Particle Filter with an initial particle density, the goal is to localize the person on the given floor. Depending on the amount of information which can be exploited to constrain the prior belief of the system's state this phase is computationally demanding resulting in high execution times. The localization from an equally distributed initial particle set is the worst case for the execution time of the Particle Filter update. For approximately $22.7 \cdot 10^3$ particles on the RCS floor with an area of about $420m^2$ this takes $150ms$ according to Figure 6.2. Since the PiNav-System needs some computation time as well to filter the incoming data, this means that the deadline of $373ms$ is likely to be violated for the first steps.

After the person has been localized on a floor, the computational effort decreases, since the amount of particles representing the pdf of the state of the person gets smaller, as can be seen in Figure 6.2. As soon as the particle distribution is localized, which in this case is after about 35 steps, the the size of the cluster remains approximately constant at about $1.1 \cdot 10^3$ particles, which results in an execution time of less than $15ms$. This is more than one order of magnitude smaller than the deadline of $373ms$, which shows that the Particle Filter can provide on-line tracking of a pedestrian on the laptop used.

6.3 Localization Behavior

6.3.1 Experimental Setup

The main indicators for the performance in the localization phase is the distance which has to be walked until ambiguities are resolved and the accuracy of the obtained position estimate. This performance was evaluated starting from an equally distributed particle set as depicted in Figure 6.3. Since training the AM for such a large area imposes the need to walk through the free space of every room several times, the localization behavior is evaluated only for the KLD-Sampling Particle Filter using the geometric floor plan compared against the performance of the Particle Filter using an AM which describes pedestrian behavior as depicted in Figure 5.5.

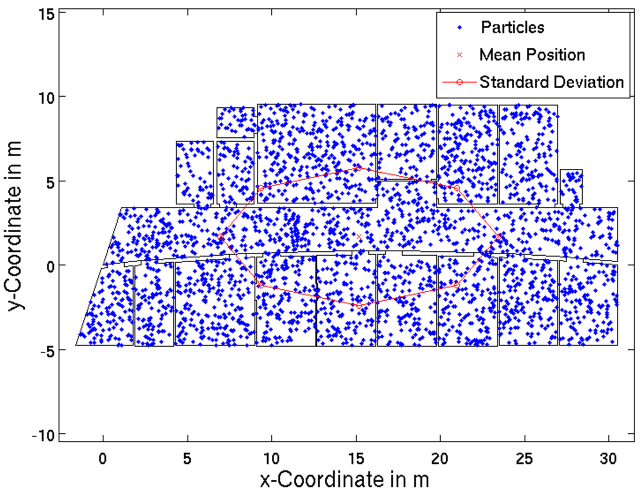


Figure 6.3: The Localization performance was evaluated starting from the depicted equally distributed particle set.

The test-trajectory to evaluate the localization performance of both algorithms is depicted in Figure 6.4. The starting point was at the left end of the track. The only room it leads trough is the copier room of the RCS. The adjacent room left of the copier room is the server room of the RCS. This path was chosen, since it reveals several interesting properties which show the strength and weaknesses of the Particle Filter in the localization phase. The first three to four meters could have been walked starting in every of the bigger rooms of the floor, which is ambiguous. Moving more than four meters to the left, the starting point of the trajectory can not have been in one of the rooms with a width smaller than four meters. This leaves only the corridor, the big server room or one of the three connected rooms below the server room as candidates for the position of the person. After walking in and out of the copier room, the only possible position is somewhere in front of this room. But as soon as the person walks back into the corridor, there are, due to uncertainty in the orientation of the person, two possible trajectories. Besides the true trajectory into the corridor, the person could also have walked into the server room. At the end of the corridor no ambiguity is left in the position of the person.

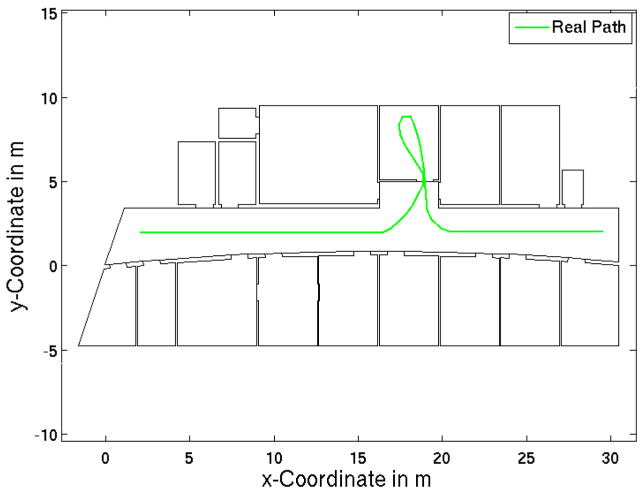


Figure 6.4: The trajectory which was chosen for the evaluation of the localization and tracking performance of the algorithm. The track was walked from left to right and leads through the copier room.

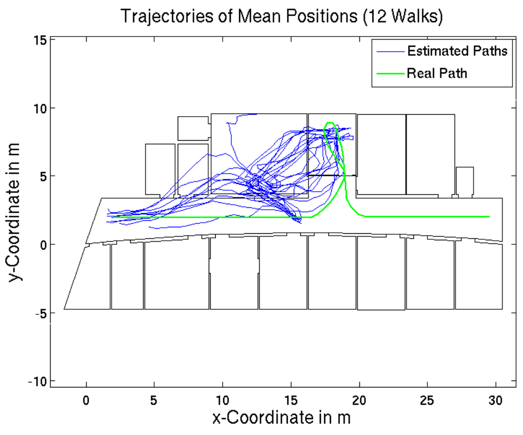
In order to be able to measure the localization accuracy, the position of every step of the path was marked with post-its, which were calibrated using a laser range finder. In the final analysis, this real path was compared against the position estimates during localization, which are obtained using the mean over all positions in the particle set as introduced in section (4.6). The distance of the estimated position $(\bar{X}_k, \bar{Y}_k)$ from the real position (X_k, Y_k) at time t_k was quantified using the euclidean distance

$$d_k = \sqrt{\left(X_k - \bar{X}_k\right)^2 + \left(Y_k - \bar{Y}_k\right)^2}. \tag{6.1}$$

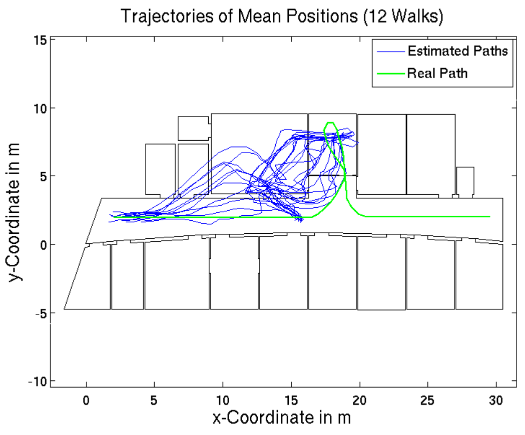
The spread of the particle distribution was measured using the standard deviation as introduced in Equation (4.9).

6.3.2 Results

Figure 6.5 shows the trajectories of the mean positions of the particle sets after each step. The estimated trajectories start from somewhere in the center of the floor, since an equally distributed prior pdf is assumed, which explains the mean position at the beginning. The trajectories end on the true path which indicates that the localization phase is finished. From here on the tracking phase would start. Interestingly the localization fails in one walk if only the floor plan is used as can be seen in Figure 6.5a.

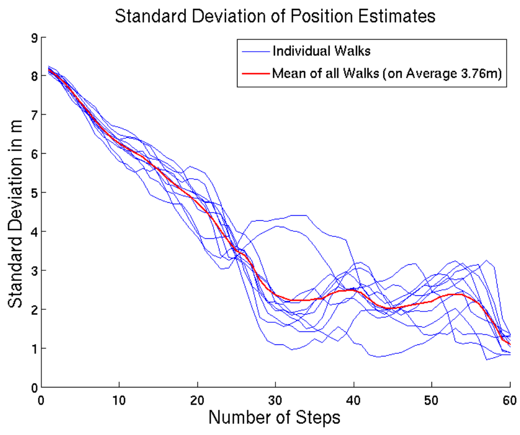


(a) Using the Geometric Floor Plan

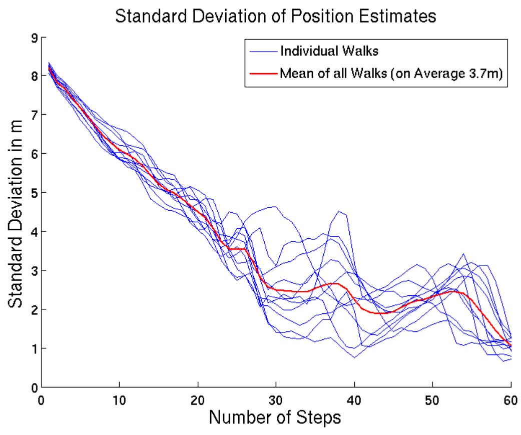


(b) Using the Accessibility Map

Figure 6.5: Localization Phase Trajectories.



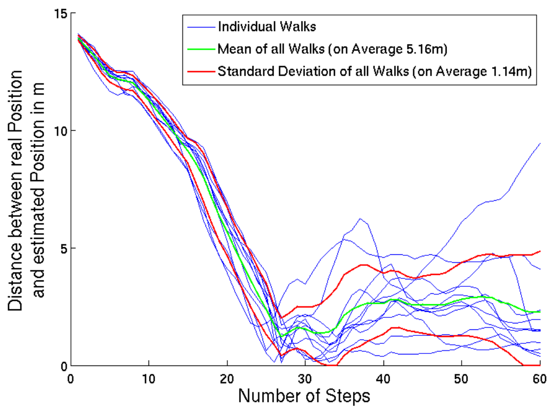
(a) Using the Geometric Floor Plan



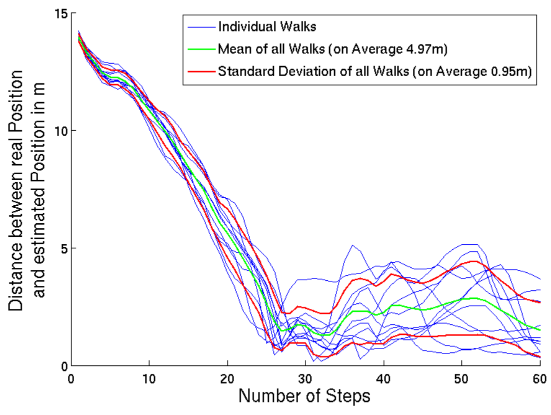
(b) Using the Accessibility Map

Figure 6.6: Localization Phase Standard Deviation of Position Estimates.

Figure 6.6 shows the standard deviation of the position estimates. In both cases, the standard deviation starts at a high level of about $8m$, indicating that the spread of the particle distribution is large at the beginning since the particles are equally distributed over the whole floor. The process of localization can be seen from the decreasing average standard deviation of the particle sets, which drops to about $2.5m$ after 30 steps. This is an indicator for remaining ambiguities. After 57 steps, the particle set is reduced to a single cluster with a mean standard deviation of about $1m$ in both cases.



(a) Using the Geometric Floor Plan



(b) Using the Accessibility Map

Figure 6.7: Localization Phase Distances between the Real Position and the Estimated ones.

The deviation of the estimated tracks from the true path is depicted in Figure 6.7 for both algorithms. Again at the beginning the distance to the real path is high, since the mean of initial particle set is somewhere in the center of the floor. This distance decreases until the position is approximately found after about 27 steps. From that step number on the deviation increases again, which is caused by the particles, that are propagated in the server room as depicted in Figure 6.8. This draws the mean position of the particle set in the direction of the server room. At the end the deviation decreases as soon as the particles in the server room are sorted out because they hit the wall. After 60 steps the mean distance of the position estimate to the real path in Figure 6.7a is $2.26m$ whereas in Figure 6.7b it is $1.53m$.

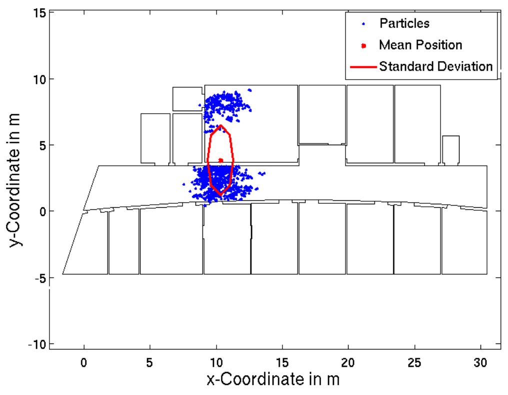


Figure 6.8: At the end of the path the particle distribution splits into two clusters.

6.3.3 Interpretation

In summary the main results in the localization phase are shown in the following table:

	Particle Filter using	
Performance Criterion	Geometric Floor Plan	Accessibility Map
Average standard deviation of the distances between the estimated and the true positions	1.14m	0.95m
Mean distance to the real path at the end of the walks	2.26m	1.53m
Steps till first minimum in the distance to the real path	27	27

Figure 6.9: Main results of the evaluation of the localization phase.

As can be seen from Table 6.9, the average standard deviation of the distances between the real path and the paths estimated with the Particle Filter using the AM is 13.2% lower than in the case of the Particle Filter with the geometric floor plan. This indicates that the Particle Filter using the AM produces similar trajectories in a more robust way than the Particle Filter without AM. This assumption is further confirmed by the outlier produced by the Particle Filter without AM which shows an increasing distance to the real path as visible in Figure 6.7a and 6.7a.

From Table 6.9 it becomes clear that in the examined scenario, the Particle Filter with the support of the AM can localize the person with a 32% higher accuracy in comparison to the Particle without the AM. The first minimum in the distance between estimated paths and the true path occurs after 27 steps with both Particle Filters indicating that the localization speed is the same.

In the examined scenario, the Particle Filter using the AM shows more robust and more precise localization characteristics than the Particle Filter without the AM. The localization speed is not influenced by the usage of the AM.

6.4 Tracking Performance

6.4.1 Experimental Setup

The localization phase is followed by the tracking phase which mainly aims to provide exact position estimates. Therefore the performance of the Particle Filters can be measured by the deviation of the position estimate from the true positions. Ground truth was established manually by walking the predefined trajectory depicted in Figure 6.4. As mentioned in the previous section, the true path was defined by the step positions marked with post-its.

Since in the tracking phase the particle distribution is assumed to be localized, the initial distribution for the Particle Filter was chosen to be a normal distribution of particles around the real starting position of the track as depicted in Figure 6.10. The standard deviation is chosen as $0.85m$, which is approximately equal to the standard deviation at the end of the localization phase.

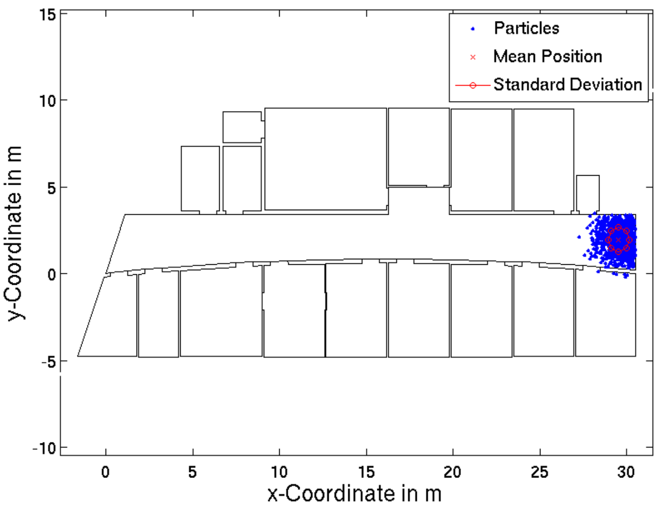
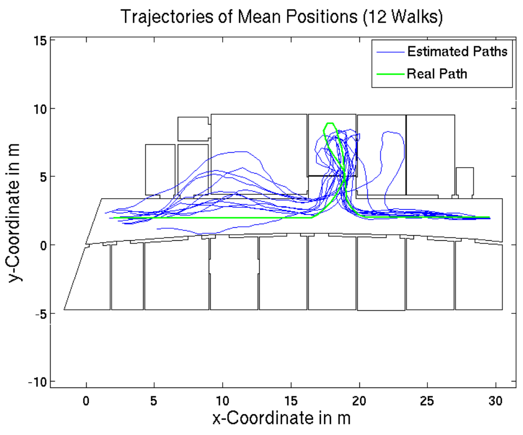


Figure 6.10: In the tracking phase the position is approximately known, which is expressed by the normally distributed particles around the starting position.

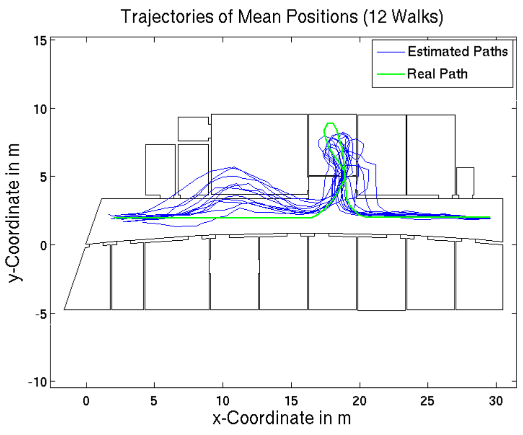
Starting from a localized particle set, learning of accessibility becomes possible. Therefore in the following the results of all three Particle Filter variants are compared. The learning Particle Filter started from the same AM as used by the non-learning Particle Filter with AM. Successively as the different paths were walked, the AM was trained. The learning factor was set to $\alpha = 0.4$, to enable fast learning of the patterns, since only 12 walks were available.

6.4.2 Results

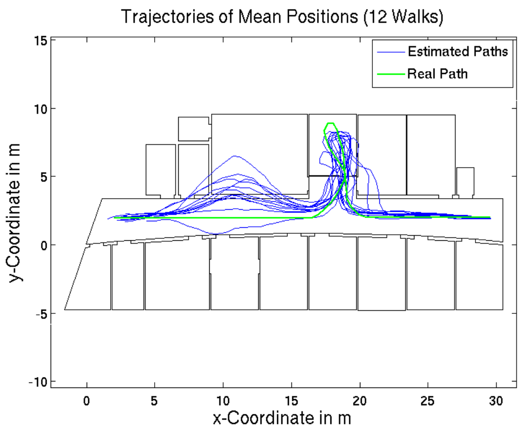
Starting from the same position at the right entrance of the corridor of the RCS floor, the real track and the blue estimated trajectories are almost equal as depicted in Figure 6.11. After turning right to enter the copier room, the trajectories start to diverge. In all three cases there are outliers, which lead through the room right of the copier room, as a result of errors in the orientation estimation. Its is visible that the path estimates of the Particle Filters with AM are located denser around the real track than those of the Particle Filter without AM. Also the outliers are drawn more to the real path if an AM is used. After leaving the copier room, an interesting situation occurs. The real trajectory leads back to the corridor, whereas the particle set has two possibilities: Due to the uncertainty of the sensor measurements, the person could also have entered the server room. In such a situation the particle set just splits and takes both possible paths. The mean of both clusters is somewhere in between them, which accounts for the trajectories which bend into the server room, before the ambiguity is resolved and the estimated paths end on the real track.



(a) Using the Geometric Floor Plan



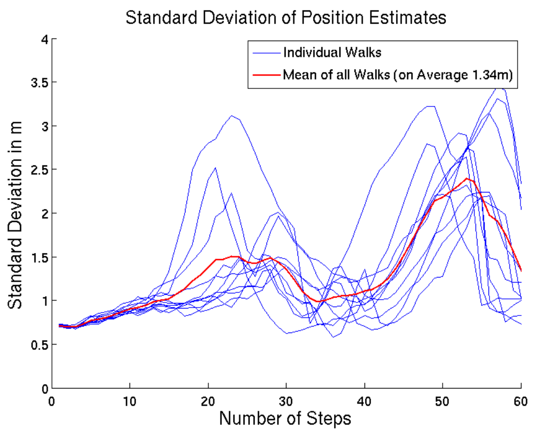
(b) Using the Accessibility Map



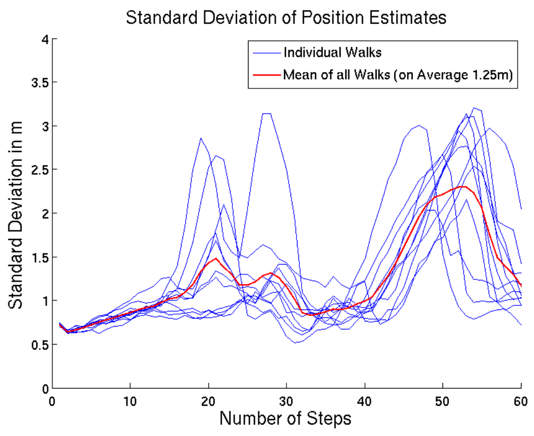
(c) Using the Learning Accessibility Map

Figure 6.11: Tracking Phase Trajectories with and without AM

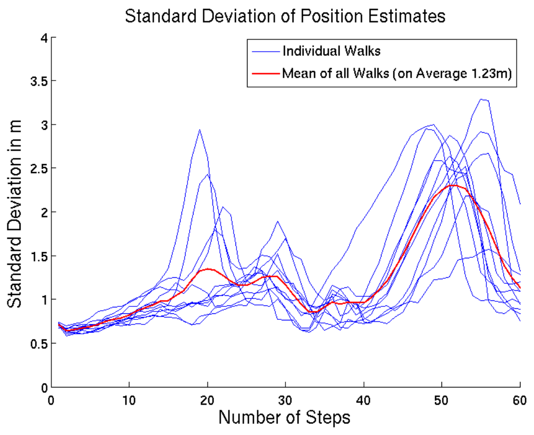
The standard deviation of the position estimates show mainly two peaks as depicted in Figure 6.12. The first smaller peak occurs, when the copier room is entered, since there is a small chance that the person entered one of the rooms beside this room. The result is that the standard deviation increases, since a few particles are propagated in the neighbor rooms. The second and higher peak, originates from the ambiguity at the end of the path, where two clusters of particles are propagated, one in the server room and one on the corridor.



(a) Using the Geometric Floor Plan



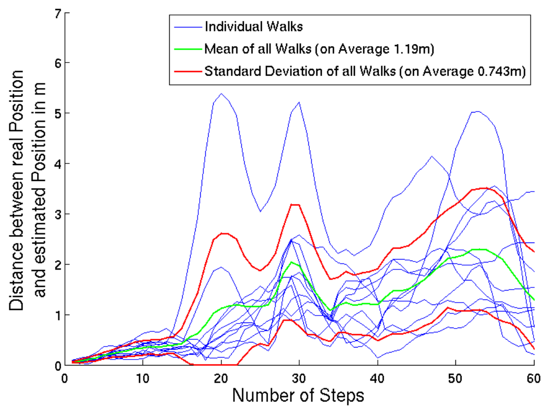
(b) Using the Accessibility Map



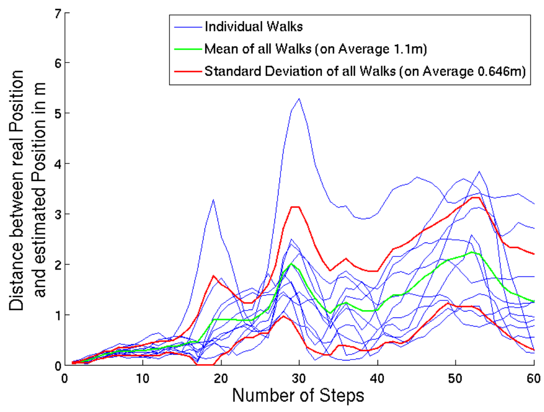
(c) Using the Learning Accessibility Map

Figure 6.12: Tracking Phase Standard Deviation of the Particle Sets with and without AM.

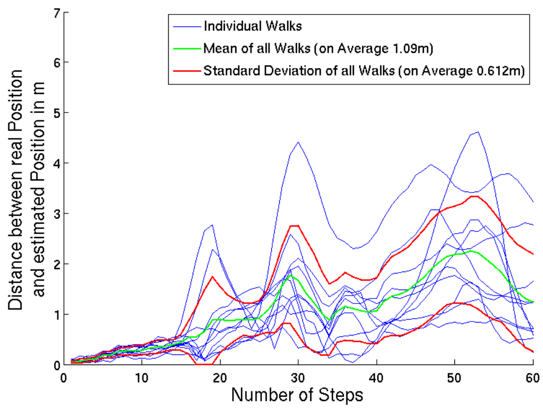
Figure 6.13 depicts the distances between the real positions and the estimated ones for all three Particle Filters. At first the deviation remains low, since the path only leads straight to the left. As soon as the track bends to the right where several possible paths open up, the deviation increases. The last peak in the distance to the real path originates again from the ambiguity between corridor and server room, since the mean positions are drawn away from the real position by the second cluster of particles in the server room. As the number of particles in the server room cluster shrinks, the position estimates are pulled back to the real positions.



(a) Using the Geometric Floor Plan



(b) Using the Accessibility Map



(c) Using the Learning Accessibility Map

Figure 6.13: Tracking Phase Distances between the Real Position and the Estimated ones.

The AM which was trained by the tracks depicted in Figure 6.11c, reflects the real path as can be seen in Figure 6.14. Only at the end of the track where the ambiguity between server room and corridor occurs, no accessibility could be learned since no particle set in all walks was localized in that section of the path.

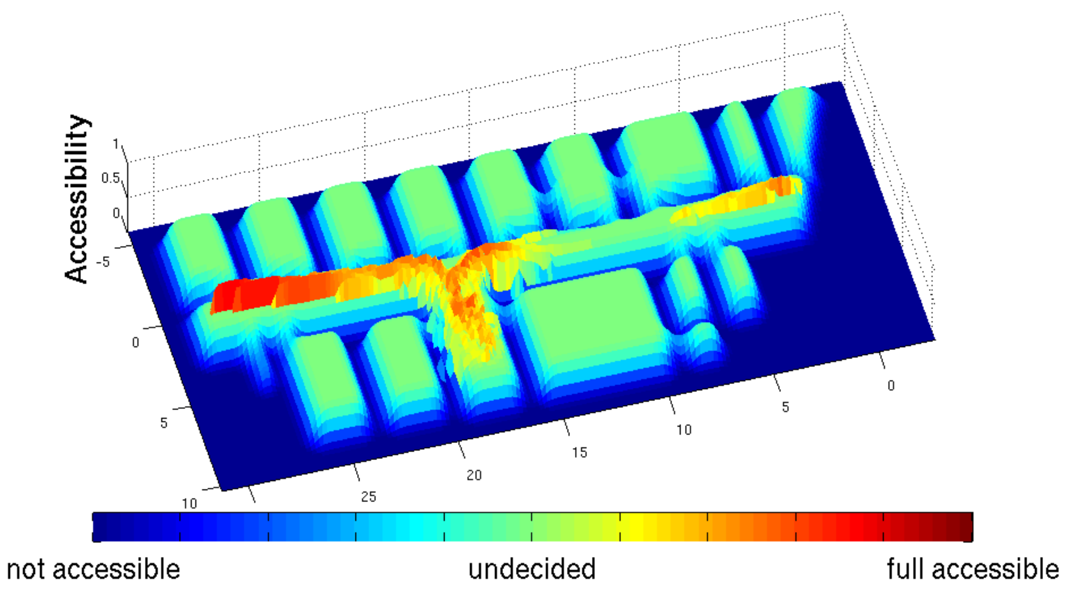


Figure 6.14: Using 12 walks the AM was trained the path of the tracking evaluation scenario.

6.4.3 Interpretation

The main results are summarized in the table below:

	Particle Filter using		
Performance Criterion	Geometric Floor Plan	Accessibility Map	Learning Accessibility Map
Average distance to the true positions	1.19m	1.10m	1.09m
Average standard deviation of the distances between the estimated and the true positions	0.74m	0.65m	0.61m
Average standard deviation of the position estimates	1.34m	1.25m	1.23m

Figure 6.15: Main results of the evaluation of the tracking phase.

As can be seen from Table 6.15, the Particle Filters with AM show slightly smaller average standard deviation of the position estimates compared to the Particle Filter without AM, implying that their particle distributions are denser. Since KLD-Sampling is used this means that less particles describe the pdf of the state of the system which results in a faster execution of the Particle Filters with AM.

The average mean distance of the estimated paths from the real path is 7.6% lower when using the AM than without it. Furthermore, the inclusion of the learning AM yields an enhancement of 8.4% in the tracking accuracy. The mean standard deviation of the distance between real and estimated track shows an improvement of 12.2% when using the AM in addition to the geometric floor plan and an improvement of 17.6% when the learning AM is adopted. This implies that supplementing the floor plan with the AM provides a more robust tracking of pedestrians, since the different path estimates are narrower confined. Deploying the learning AM the robustness of pedestrian tracking can also be improved while additionally recording the moving patterns of a person.

The average deviation from the real path is about 1.1m when using either the AM or the learning AM. The overall lower accuracy in comparison to the results of Woodman et al. in [28] originates from the deployed hardware. Woodman uses a foot-mounted IMU which can provide better measurements of stride length and stride direction. In contrast to this, the PiNav-System is worn in the user's pocket. This makes it more comfortable to use the navigation system but it also makes it harder to extract accurate stride information from the IMU. In addition, for the test runs fixed stride lengths were used as input for the Particle Filters. Under these two worsening prerequisites the achieved localization accuracy is quite promising after all.

6.5 Using a trained Accessibility Map

6.5.1 Experimental Setup

The capability of learning the accessibility in a room and the effects on the tracking accuracy is further examined in the biggest room of the RCS floor, the server room. The room is mainly occupied by several desks and storage racks leaving only sparse accessible areas. For this experiment a localized particle distribution was assumed as initial distribution, since accessibility can only be learned from a distribution free of ambiguity as described in section (5.2).

The starting point for all recorded tracks was three steps outside the server room. Around this starting position, the initial particle set was normally distributed with a standard deviation of 0.85m as depicted in Figure 6.16.

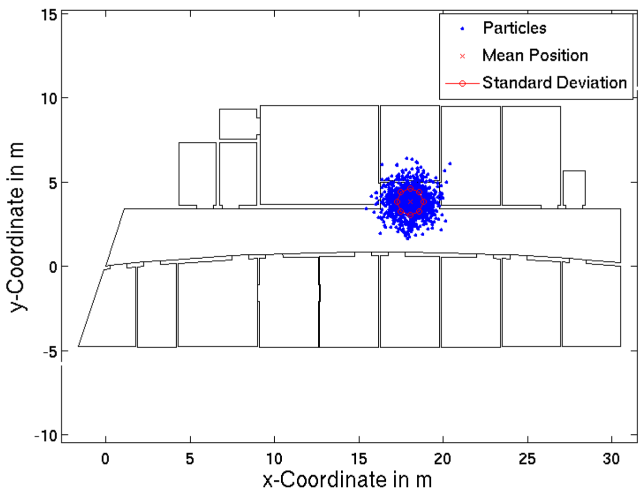


Figure 6.16: The particles are normally distributed around the starting position in front of the server room.

For the evaluation 25 walks on a predefined trajectory and 10 random paths were recorded, covering all accessible regions in the server room. Again the step length was fixed to 60cm using post-its for the step positions. For the training of the AM, 14 of the 25 walks on the predefined path were randomly selected and together with the 10 random walks used to train the AM. The learning factor α was set to 0.4 to allow fast training. After that the 11 walks which were not learned by the AM, could be used to evaluate the positioning performance of all three Particle Filters.

6.5.2 Results and Interpretation

The trained AM, as depicted in Figure 6.17, reflects the free space in the server room.

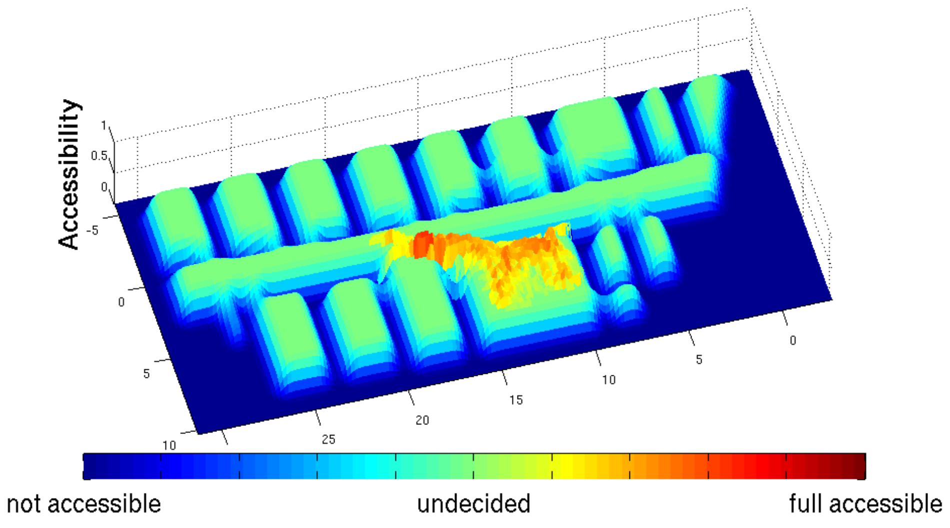
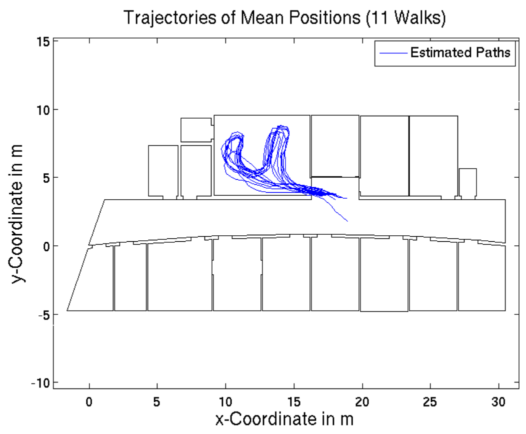


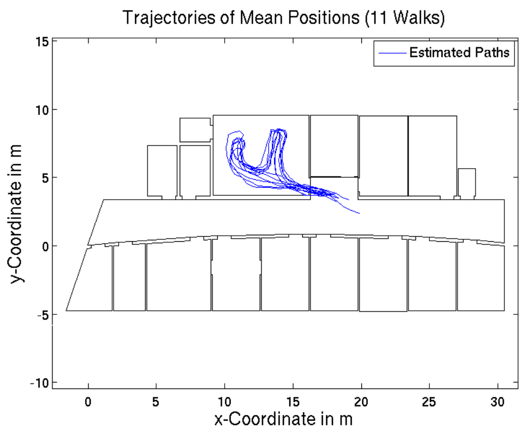
Figure 6.17: Using 24 walks the AM was trained the free space in the server room.

It can be seen that the area with higher accessibility has the shape of an “U”, since there are desks separating the server room into two free regions. By using the 10 random and the 14 defined trajectories to train the AM, a good coverage of these free areas could be achieved. This becomes evident from the almost equally high accessibility in these regions.

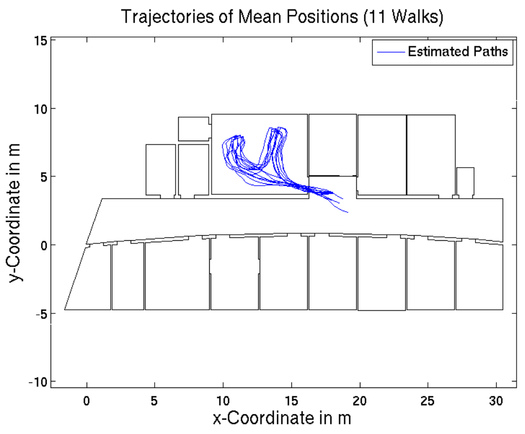
Figures 6.18a to 6.18c depict the estimated trajectories of the 11 test-walks obtained from the three different Particle Filters. Although they produce very similar trajectories, it can be seen that the path estimates of the Particle Filters using only the geometric floor plan, are more expanded than those of the other two Particle Filter variants. This is an effect of the AM, which decreases in proximity to walls, where a person is unlikely to walk.



(a) Using the Geometric Floor Plan



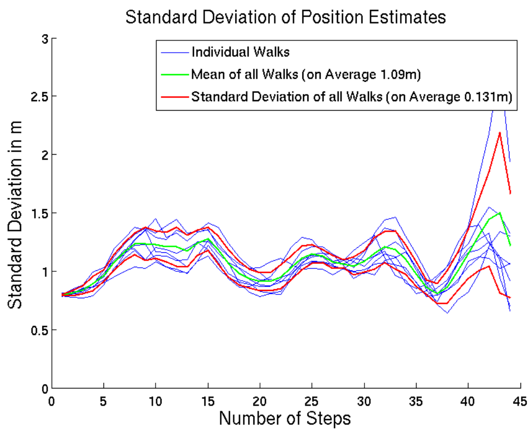
(b) Using the Accessibility Map



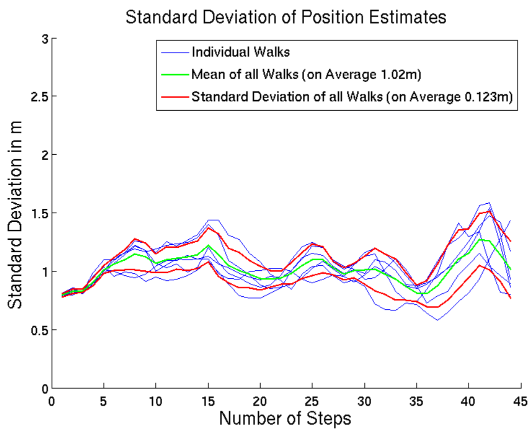
(c) Using the Learning Accessibility Map

Figure 6.18: Estimated Trajectories through the Server Room of the three different Particle Filters.

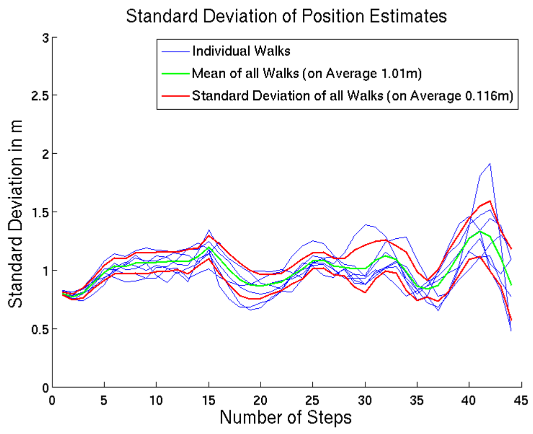
The standard deviation of the particle sets during these walks are shown in Figures 6.19a to 6.19c. With an average standard deviation of 1.09m the particle sets of the Particle Filter using a geometric floor plan have an 6.4% wider spread than if the static AM is used and a 7.3% wider spread if the trained AM is used. The standard deviation from this average decreases from 0.131m using only the geometric floor plan to 0.116m using the learning AM. Compared against the standard deviation of the average curve of the Particle Filter with the geometric floor plan, this deviation decreased by 6.1% when using the untrained AM and by 11.5% when the information of the trained AM were used.



(a) Using the Geometric Floor Plan



(b) Using the Accessibility Map



(c) Using the Learning Accessibility Map

Figure 6.19: Standard deviation of the particle sets during the walk through the server room.

This results imply that the spread of the particle sets decreases if either one of the AM types is used. The deployment of a trained AM yields particle set standard deviations which vary less in comparison with the other two Particle Filter types. Since all 11 test-walks were obtained from the same real path, this indicates that the estimation of the path is more robust with a trained AM than with the two other methods.

7 Conclusion

From the theory of Bayesian Filters the derivation of the concept of a Particle Filter was shown. Basing on these fundamentals, a KLD-Sampling Particle Filter for pedestrian localization was developed, implemented and integrated into the existing PiNav-System. The Particle Filter can provide online localization and tracking of a person in a building given its floor plans. KLD-Sampling adapts the number of particles to describe the probability density function of the state of the person dynamically which improves the computational efficiency of the Particle Filter. All algorithms involved have a time complexity of $O(N)$, where N is the number of particles in the current set, which makes the Particle Filter suitable for online tracking of a pedestrian. In a novel approach the accessibility of space in a floor was modeled according to human walking behavior, which is mainly the tendency to avoid direct proximity of walls. This information was encapsulated in the Accessibility Map, which is a grid based approximation of the accessibility function of the specific floor. This means that the floor is discretized into small squares for which the accessibility of this area is stored. According to the avoidance of small distances to walls, areas immediately next to walls are assigned a lower accessibility than those in the inner regions of a room of the floor. A Particle Filter can exploit the AM to weight particles proportional to the accessibility of the area they are in. Since particles with heavier weights have a higher chance to be propagated, the set of particles will be denser in areas with high accessibility, modeling human walking behavior. The observation, that obstacles in a room are reflected in the walked paths of a person through the room, led to the idea to learn accessibility from the estimated paths. Interpreting the AM as a Radial Basis Function Network, the learning techniques for Neural Networks could be transferred to the AM yielding a rule to learn accessibility of visited areas.

In the evaluation, the performance of the Particle Filter using only a floor plan was compared against the use of an AM and a learning AM for one trajectory through the RCS floor. The two phases Localization and Tracking were evaluated separately from each other. In this scenario, it turned out that, in comparison to a Particle Filter with a geometric floor plan, a pedestrian can be localized in a floor with an up to 32% higher accuracy, when using an untrained AM. During tracking of a person the AM yields a slight enhancement of 7.6% in the position estimation. Additionally, if the floor plan is supplemented with an AM, the robustness of pedestrian tracking can be improved as indicated by the standard deviation of the distance of the position estimates to the real position which decreases by 12.2%. The learning AM can enhance both the tracking accuracy and the tracking robustness while recording the walking patterns of a pedestrian. In the best case, this leads to an absolute deviation of about $1.1m$ form the real path during tracking using the AM, which is a promising result for the deployed hip-mounted IMU.

8 Outlook

The future application in the PiNav-System requires some additional implementations to enable tracking over multiple floors. These were not realized since altitude estimation was not yet implemented. Given an estimate of the current elevation, a method has to be implemented which determines, whether the person changed floors, and which loads the subsequent floor plan to hand it over to the Particle Filter. In the Particle Filter itself it would be desirable to have defined stairwell regions which allow changes between floors. Only Particles which are located in these areas would be copied to the next floor when a significant difference in elevation is detected. This way a transition between floors would serve as an observation which significantly constrains the possible location of the particles. For this thesis the layout of the RCS floor was digitalized manually from a given CAD drawing, which in a real application would not be feasible. Therefore an automated generation of the floor outline would be desirable. Since for most modern buildings of public interest in Germany CAD drawings should be obtainable, importing such a CAD floor outline into the Particle Filter format could be a possible solution.

Concerning the AM proposed in this thesis, some improvements could also be implemented regarding storage efficiency and performance. At the moment simply a rectangle around the whole floor plan is taken as the area which is described by the AM. Therefore also areas between rooms have accessibility values although particles cannot get their anyway. A more sophisticated implementation with for example a quadtree representation of the AM, could improve the storage efficiency in that areas outside rooms would not be described in such a high resolution. This is important when thinking of a real application of the positioning system, since a lot of AMs would have to be stored, if for example all public floor plans of a whole city like Munich should be available on a mobile device. As already mentioned the learning AM needs to be able to forget about previously learned accessibility in areas which are not visited for a longer period of time, to react to dynamic changes in the environment. The proposed solution to train the unlearned AM from time to time has to be evaluated in a long term test of the system. It is anticipated that the performance of the learning AM could be improved if the accessibility tracks learned would be smoothed from time to time, yielding gentle transitions between the single elements of the AM. Smoothing could be done using a convolution with a Gaussian kernel. The effect of the smoothed learned accessibility would be that gaps between close accessibility peaks would be lifted. Small gaps of less than $0.4m$ are unlikely to originate from an obstacle in a room, since obstacles like desks or chairs tend to have a larger extend. Smoothing the trained AM would realize this observation.

Bibliography

[1] John E. A. Bertram and Andy Ruina. Multiple walking speed-frequency relations are predicted by constrained optimization. *Journal of Theoretical Biology*, 209(4):445–453, 2001. 6

[2] William M. Bolstad. *Introduction to Bayesian Statistics*. Wiley-Interscience, 2 edition, April 2007. 10

[3] James Carpenter, Peter Clifford, and Paul Fearnhead. An improved particle filter for non-linear problems. pages 2–7, 2004. 15

[4] F. Cavallo, A. M. Sabatini, and V. Genovese. A step toward gps/ins personal navigation systems: real-time assessment of gait by foot inertial sensing. In *Intelligent Robots and Systems, 2005 (IROS 2005)*, pages 1187–1191, 2005. 2

[5] M. W. M. Gamini Dissanayake, Paul Newman, Stefen Clark, Hugh F. Durrant-whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17:229–241, 2001. 2, 44

[6] Dieter Fox. Kld-sampling: Adaptive particle filters. In *In Advances in Neural Information Processing Systems 14*, pages 713–720. MIT Press, 2001. 16, 17, 31

[7] Thomas Gebauer. Entwicklung eines dead reckoning prototypen zur persönlichen positionsbestimmung. Studienarbeit at the TU Munich, Mai 2008. 1, 5, 6, 19

[8] Siome Klein Goldenstein. A gentle introduction to predictive filters. *Revista de Informatica Teorica e Aplicada (RITA) XI*, 1:61–89, 2004. 9, 11

[9] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEEE Proceedings*, 140(2):107–113, 1993. 11, 12

[10] J. Hartung, B. Elpelt, and K.-H. Klösener. *Statistik: Lehr- und Handbuch der angewandten Statistik*. Oldenbourg, München, 1982. 31

[11] Braun Heinrich. *Neuronale Netze Optimierung durch Lernen und Evolution*. Springer-Verlag, 1997. 45

[12] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering*, (82 (Series D)):35–45, 1960. 11, 20

[13] George F. Luger. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Personal Education Limited, 4 edition, 2002. 47

[14] D. Lurie and H. O. Hartley. Machine-generation of order statistics for monte carlo computations. *The American Statistician*, 26(1):26–27, 1972. 15

[15] Martin Obermeir. Wlan basierte positionsbestimmung auf einem arm9 system. Diploma thesis at the TU Munich, April 2009. 1

[16] Lauro Ojeda and Johann Borenstein. Non-gps navigation with the personal dead-reckoning system. In *Unmanned Systems Technology IX*, volume 6561. SPIE Defense and Security Conference, 2007. 2

[17] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. Mc-Graw Hill, 1984. 10

[18] Ioannis M. Rekleitis. A particle filter tutorial for mobile robot localization. Technical Report TR-CIM-04-02, Centre for Intelligent Machines, McGill University, 2004. 14

[19] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, 2004. 9, 11, 12

[20] P. Robertson, B. Krach, and M. Khider. Slam dance inertial-based joint mapping and positioning for pedestrian navigation. *InsideGNSS*, May 2010. 2, 44

[21] H. W. Sorenson and D. L. Alspach. Recursive bayesian estimation using gaussian sums. *Automatica*, 7(4):465–479, 1971. 11

[22] Ivan E. Sutherland, Robert F. Sproull, and Robert A. Schumacker. A characterization of ten hidden-surface algorithms. *ACM Computing Surveys*, 6(1):1–55, 1974. 26

[23] E.A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158, 2000. 11

[24] Ulrich Weidmann. Transporttechnik der fußgänger - transporttechnische eigenschaften des fußgängerverkehrs (literaturauswertung). In *Schriftenreihe des IVT (90)*, Zürich, 1992. Institut fuer Verkehrsplanung und Transportsysteme. 30, 37, 38, 54

[25] Widyawan, Martin Klepal, and Stephane Beauregard. A backtracking particle filter for fusing building plans with pdr displacement estimates. In *WPNC '08: Proceedings of the 5th workshop on positioning, navigation and communication*, pages 207–212, March 2008. 2

[26] Wikipedia. Gps - position calculation introduction. `http://en.wikipedia.org/wiki/Global_Positioning_System#Position_calculation_introduction`. 1

[27] E. B. Wilson and M. M. Hilferty. The distribution of chi-square. In *Proceedings of the National Academy of Sciences of the United States of America, 17*, pages 684–688, 1931. 17

[28] Oliver Woodman and Robert Harle. Pedestrian localisation for indoor environments. In *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*, pages 114–123, New York, NY, USA, 2008. ACM. 2, 16, 23, 66