TECHNISCHE UNIVERSITÄT BERLIN
FAKULTÄT VI – PLANEN BAUEN BAUEN UMWELT
INSTITUT FÜR GEODÄSIE UND GEOINFORMATIONSTECHNIK

MASTER-THESIS

Monte Carlo Localization for Pedestrian Indoor Navigation Using a Map Aided Movement Model

---

Author: Babak Naderi
Academic advisor: Prof. Dr. rer. nat. Thomas H. Kolbe
Date of submission: July 2012

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Ort, Datum	Unterschrift

Acknowledgements

I would like to express my gratitude towards Prof. Thomas H. Kolbe for the topic of this thesis, his time, supervision and suggestions. I would like to thank Mehmet Yildirim for his time and invaluable participation in the experiments. I would like to thank all of my colleagues who have participated in experiments. My deepest thanks go to Hywa and my family for their love, unconditional support and invaluable encouragement they offered me.

Abstract

In this thesis, a localization system for pedestrian indoor navigation is designed and implemented. This system combines the data of both motion and orientation sensors with environmental constraints pertaining to both the movement model of a pedestrian indoors and the measurement model with a localization filter to reduce uncertainty of a target's position. A target person takes a walk and carries a smartphone that is secured properly on his/her body. From the accelerometer and magnetic field sensor data of the smartphone, steps are detected and step events are created. A method for recognizing human gaits from acceleration data is introduced. The step event contains step length, change in direction and time elapsed, and is delivered to the localization model. The localization model is based on the particle filter and is known in robotics as Monte Carlo Localization. The Monte Carlo Localization represents the estimated position of target by weighted particles. The localization model is enhanced by three different movement models. Two of them employ a map of environment and modify step events for each particle to increase the particle's survival chance. In addition, environmental constraints are employed as a virtual sensor in the measurement model of the localization model to evaluate current state of each particle by computing their importance factors. Moreover, Kullback-Leibler Distance Sampling is used for adapting the number of particles during the localization process. A simulator is also implemented for testing the localization model on synthetic environments.

The localization model, enhanced by three movement models, is tested on a real 170 m walk taken in the 6^{th} floor of main building at the Technical University of Berlin. In all three cases, accuracy of the localization model was significantly higher than the normal pedestrian dead reckoning system. However, the movement model that does not employ the environmental constraints gained the highest estimation accuracy. As a result, using the environmental constraints in measurement model increased the accuracy of estimation, but additionally modifying the step events for increasing survival chance of a particle decreased chances of the other particles and reduced the accuracy of estimation.

The localization model enhanced by naive movement model was able to estimate a target's position within 2.67 m, 50% of the time and 4.4 m, 75% of time.

Contents

1 Introduction

Outdoor localization has reached some maturity with Global Navigation Satellite Systems (GNSS). Due to a weak and partially absent satellite signal indoors, other methods have been developed for indoor localization. The goal of this work is to develop a pedestrian indoor localization method by using a low-cost sensor configuration carried by a target person and supported with minimal infrastructure.

Previously, the author developed a Local Pedestrian Indoor Dead Reckoning Localization System. *Dead Reckoning* (DR) is a passive positioning system that determines a target's position by calculating displacement from a well-known starting point. Inertial sensors were used for estimating steps and displacement. For this work, an HTC Desire smart phone with the Android 2.2 operating system was used as a common, low-cost host device that contains accelerometers, magnitude field, and orientation sensors.

A pedestrian's steps were detected, and displacement was computed by estimating heading and a constant value as a step length. In experiments, the user should select a starting point. As the experiment proceeds, the positioning error grows. Drift in DR systems may reach up to 100 meters after a minute of usage [Woodman, 2010]. The current work builds upon this type of experiment.

The main result of this work is the design of a sensor fusion engine that combines step events and environmental constraints with respect to the movement model of a pedestrian inside of buildings. Consequently, uncertainty regarding a target's position is decreased. In addition, the integration of other localization methods, like cell-based localization, is taken into account during the design process.

The goal of this chapter is to provide a review of related works.

1.1 Pedestrian DR

DR computes the current position of an object using its previously determined or known position and then summing estimated displacements. A displacement between two points

is derived from an estimation of the direction (heading) of movement and traveled distance (or speed). Usually, 3-axis accelerometers are used for measuring the distance and 3-axis gyroscopes and/or a compass are used for measuring the direction; all of these are available as *Micro Electro-Mechanical System (*MEMS) inertial sensors. DR systems are also known as *Relative Positioning* systems when they track a target relative to its initial position and heading.

Pedestrian DR (PDR) is divided into two main groups. The first group is based on step detection and second group is based on the *Inertial Navigation System (INS)*. In first group, a human step is considered a displacement. Sensors, both motion and rotation, are fixed on the target's body to compute the course and distance of each step. The second group computes distance using the physical rule: *the second integral of acceleration is position.* The INS suffers from different kinds of errors, such as *Sensor-Level Errors* (systematic error). To avoid these errors, one must calibrate the INS. Figure 1 illustrates a histogram of the acceleration magnitude observed by a stationary smartphone. This result is due to bias introduced by sensors over gravitational force.

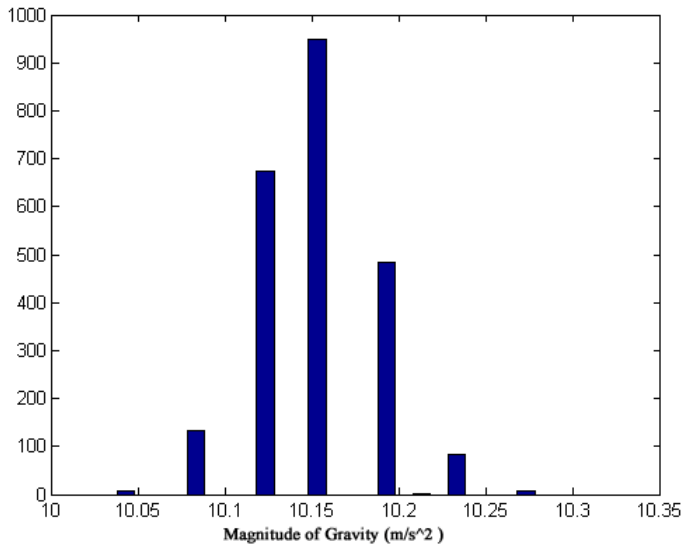


Figure 1: Histogram of acceleration magnitude recorded by a stationary smartphone.

1.2 Related Works

1.2.1 Probabilistic Robotic

In [Thrun et al., 2005], the authors used a *Monte Carlo Localization (MCL)* approach to solve global localization problems with robots. MCL is a non-parametric *Markov Localization* method that uses a *particle filter* to integrate a robot's *motion model* and a sensor's *measurement model*. The particle filter approximates the posterior probability, *i.e.* the target's position, by a finite number of values, each roughly corresponding to a region in *state space*. The state vector contains all relevant variables, particularly the target's two-dimensional position and heading. For the motion model, the authors proposed velocity based (translational and rotational velocity) and odometry based (changes in heading and distance) models. For the measurement model, they consider the environmental constraints. When a line between a particle's previous position and its new one intersects with any constraint in the map, the weight of the particle is set to zero.

Moreover, they proposed *Kullback-Leibler Distance Sampling* (*KLD-Sampling)* to adapt the size of sample sets on the fly, and, by adding new randomly distributed particles in every iteration, solve the *kidnapped robot problem*. This problem occurs when a robot believes it knows its correct position, but is, in fact, incorrect. In particular, this problem tests the ability of a localization system to recover from global localization failures.

1.2.2 Pedestrian Localization for Indoor Environments

[Woodman, 2010] tried to answer general questions regarding indoor pedestrian localization, like whether environmental constraints can be used to prevent the accumulation of drift in PDR over the time. This work is similar to robot localization methods, e.g. [Thrun et al., 2005], except that a large three-dimensional environment was considered when designing and prototyping. The state-of-the-art *inertial measurement unit* (IMU) with 100Hz sampling rate is attached to a target person's foot. The *PDR Filter* fires step events based on the IMU row data. Then, a *Localization Particle Filter* module determines the target's position by using the step events and environmental constraints. In addition, WiFi localization is used to restrict initial environment. Therefore, the initial number of particles is significantly decreased. An accuracy of 73 cm, 95% of the time, is reported.

Woodman used a velocity based motion model in the PDR filter. To reduce tilt error that accumulated during the previous steps, he used *Zero Velocity Updates* (ZVU) pseudo-measurements when the user's foot was known to be stationary. Woodman's method realizes a global localization, but the system had some problems with symmetries and the scale of the environment.

1.2.3 Map-Enhanced Movement Models for Pedestrian Localization

[Wendlandt et al., 2006] and [Khider et al., 2009] discussed global pedestrian localization with a human movement model. The main advantage of using a movement model instead of pure map aiding methods is that the movement model makes the method applicable to both indoor and outdoor environments. They used two different movement models; a Markov process chooses the active model.

The first model is a *3D Stochastic Behavioral Movement Model* (3D-SBMM). They take different variables (position dependent, system defined, and individual user dependent) into account to estimate the mean and covariance of a Gaussian probability distribution for each of speed and direction change [Wendlandt et al., 2006]. The new position is computed by sampling the Gaussians and applying laws of physics. However, as a consequence of using the 3D-SBMM, the estimated position can get stuck in a room or have problems getting through narrow openings and sharp turns [Khider et al., 2009].

The second model is a 3D Diffusion Movement Model (3D-DMM). It is a goal-oriented model based on the principle of gas diffusion in space [Khider et al., 2009]. This model works well if the target moves only to reach a known destination. If the destination is unknown, then randomly distributed points are used. Pre-computed diffusion matrices are used in real-time. This model assumes that the shortest path is used to reach the destination, which is not always representative of human movement.

The author's report their localization system had an average error of 2.1 meters during 10 minutes usage.

2 Method

2.1 Introduction

This work is intended to design a localization model for pedestrian indoor navigation that combines the data of both motion and orientation sensors with environmental constraints pertaining to the movement model of pedestrian inside of buildings to reduce uncertainty of a target's position. The goal of this chapter is to describe basic concepts, vocabulary, algorithms, and, in particular, the design of the localization system.

In this context, a *target* is a person whose *state* is being estimated. A target's location is at the core of the state. The state may contain a simple two-dimensional position or a complex vector with a three-dimensional position and additional information, such as heading or velocity.

The target's location is not directly observable by sensors and is instead estimated by measuring other information. Generally, two types of measurements can be used to observe the target's location. The first type is collecting information regarding the current state of the environment using environmental measurement sensors like cameras or laser range scanners. As such, *measurement data* contains information about the momentary state of the environment. The second type involves estimating the motion of the target. When the target takes a step, the state is changed. The *motion data* can be estimated by both motion and orientation sensors. In this thesis, *measurements* will refer to both types.

Measurements are always corrupted by noise. Sensors are never perfect and they all have a different error model. Using many sensors in a system increases the uncertainty of any conclusions derived with the data. In such a system, a probabilistic approach is key [Fox et al. 2003]. Most state-of-the-art localization systems use probabilistic state estimation methods for computing the probability distributions over possible states. These methods are different implementations of a Bayesian filter, which is described in the next section.

2.1.1 Bayesian Filters

The Bayesian filter probabilistically estimates the state of a dynamic system from noisy observations [Fox et al. 2003]. The state is not directly measurable and is instead estimated using sensor measurements.

Bayesian filters represent the state at time t by random variables x_t. At each point in time, a probability distribution over the state x_t is called a *belief*, denoted $Bel(x_t)$. A belief represents the probability that state x_t is equal to the true state. The belief distribution $Bel(x_t)$ is a posterior density over state x_t conditioned on all past available measurements $z_1, z_2, ..., z_t$:

$$Bel(x_t) = p(x_t \mid z_1, z_2, ..., z_t) \qquad \mathbf{1}$$

Roughly speaking, the belief shows the probability that the target is at location x given a history of sensor measurements. Generally, by increasing the number of measurements over the time, the complexity of computing such probabilities grows exponentially [Fox et al. 2003]. To avoid that, the Bayesian filter assumes that the dynamic system is a Markov process. A process is Markov when prediction of the next state (x_{t+1}), given the past history ($x_1, x_2, ..., x_t$), depends only on the last state (x_t). In other words, state x is assumed to be complete, which means x contains a sufficient summary of information to predict the next state.

Figure 2 illustrates the relationship between state sequence and measurement data. The measurement data at time t depends stochastically on the pedestrian's current physical location at time t (state x_t). The location at time $t+1$ depends only on the previous state at time t, and previous states do not provide more information [Fox et al. 2003].

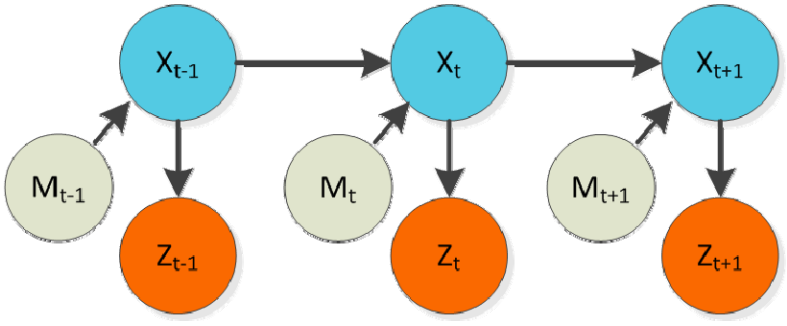


Figure 2 Graphical representation of the hidden Markov model for a localization problem. The relationships between states (X), motion data (M) and measurement data (Z) are described. States are not directly measurable (source [Thrun et al. 2005]).

Equation 1 can reformulated as Equation 2 below. Here, z_t is the measurement data at time t and m_t is the motion data at time t.

$$Bel(x_t) = p(x_t \mid z_{1:t}, m_{1:t}) \quad \textbf{2}$$

The state changes from x_{t-1} to x_t by applying the motion data. After that, the belief can be calculated before incorporating the current measurements. This probability distribution is called *prediction* and is denoted $\overline{Bel}(x_t)$. Calculating $Bel(x_t)$ from $\overline{Bel}(x_t)$ using the measurement data at time t is called *correction* or *measurements update*. These are the two essential steps of Bayesian filters.

$$\overline{Bel}(x_t) = p(x_t \mid z_{1:t-1}, m_{1:t}) \quad \textbf{3}$$

The Bayesian filter is the principle algorithm for computing $Bel(x_t)$ given $Bel(x_{t-1})$, current motion and measurement data. This filter is derived using Bayes' theorem, Markov assumption and law of total probability. Table 1 shows general Bayesian filter algorithm. Line 2 is the prediction step, where prediction of the current belief over x_t is computed based on current motion data and prior belief. The probability $p(x_t \mid m_t, x_{t-1})$ is the *state transition probability* and it describes how the state changes over time as a function of motion data. In line 3, the belief over x_t is calculated by applying the current measurement data and predicted belief. The probability $p(z_t \mid x_t)$ is called *measurement probability*. For more information about the Bayesian filter, its mathematical derivation and application in localization please refer to [Thrun et al. 2005].

Algorithm BayesFilter($Bel(x_{t-1})$, m_t, z_t)

1 for all x_t do
2 $\quad \overline{Bel}(x_t) = \int p(x_t \mid m_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$
3 $\quad Bel(x_t) = \eta p(z_t \mid x_t) \overline{bel}(x_t)$
4 end for
5 return $Bel(x_t)$

Table 1 Bayesian filter algorithm (source [Thrun et al. 2005]).

The Bayesian filter algorithm is intractable. Therefore, it must be realized using approximation methods. Generally, there are two popular families of recursive state estimation techniques for realizing the Bayesian filter. The two families differ in how they represent the probability density over state x_t.

The most popular family of Bayesian filter is *Gaussian filters*. They assume that the functional form of probability density is fixed and known, but its parameters are unknown. They represent belief by multivariate normal distributions expressed by two sets of parameters, the mean and the covariance [Thrun et al. 2005]. The Kalman filter is the most widely used member of this family. The Kalman filter approximates the probability density over state x_t by its first and second moment, which are virtually identical to the mean and covariance of a unimodal Gaussian, respectively [Fox et al. 2003]. To calculate the correct posterior by using the Kalman filter, the system must be a linear Gaussian system [Thrun et al. 2005].

The second popular family of Bayesian filter is *nonparametric filters*. They do not depend on a fixed functional form of posterior. Instead, they approximate belief by a finite number of values, each approximately corresponding to a region in state space. As the number of values approaches infinity, the approximation error converges uniformly to zero [Thrun et al. 2005]. Histogram filters and particle filters are two popular members of this group. The next section describes the particle filter, which is the basis of the localization model.

2.1.2 The Particle Filter

The particle filter is a nonparametric implementation of the Bayesian filter. It represents the belief by a set of state samples, called *particles,* drawn from the belief.

$$Bel(x_t) \approx X_t = \left\{ x_t^{(i)} \mid i = 1,..., M \right\} \quad \textbf{4}$$

The particle $x_t^{[m]}$ is a possible state at time t and M is the number of particles, which is usually a large number. The key point is the distribution of particles. A denser concentration of particles is related to a higher posterior probability. Figure 3 shows samples drawn from a non-Gaussian distribution.

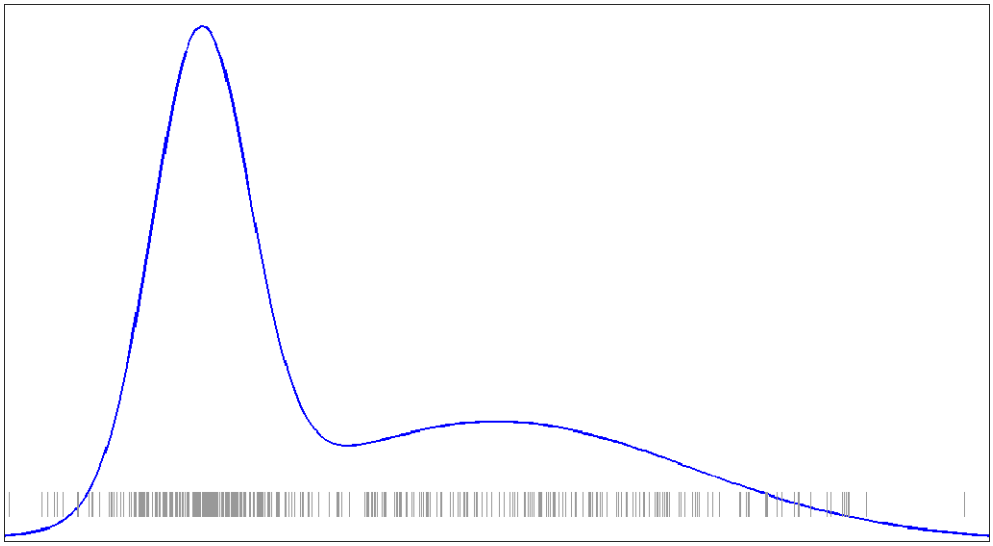


Figure 3 Sampling of a non-Gaussian probability density function.

As a consequence of this property, particle filters are optimal recursive estimators even for non-linear, non-Gaussian systems [Kwok et al. 2003]. In addition, the definition of the particle filter permits non-linear transformation functions for state variables. A generic algorithm of the particle filter is summarized in Table 2.

Algorithm ParticleFilter(X_{t-1}, m_t, z_t)

1 $\overline{X}_t = \varnothing$
2 $X_t = \varnothing$
3 *for j=1 to M do*
4 Sample $x_t^{[j]} \sim p(x_t \mid m_t, x_{t-1}^{[j]})$
5 $w_t^{[j]} = p(z_t \mid x_t^{[j]})$
6 $\overline{X}_t = \overline{X}_t + \left\langle x_t^{[j]}, w_t^{[j]} \right\rangle$
7 *endfor*
8 *for j=1 to M do*
9 *draw i with replacement from probability* $\propto w_t^{[i]}$
10 $X_t = X_t + x_t^{[i]}$
11 *end for*
12 *return* X_t

Table 2 Particle Filter algorithm (source [Thrun et al. 2005]).

The particle filter algorithm constructs a particle set X_t, representing belief at time *t,* from the particle set X_{t-1}, representing belief at time $t-1$. The first loop constructs a temporary set $\overline{X}_t$, which represents $\overline{Bel}(x_t)$. A second loop transforms particles from $\overline{X}_t$ to X_t, which approximates the posterior probability $Bel(x_t)$.

Line 4 generates a new particle, $x_t^{[j]}$, by sampling the state transition probability $p(x_t \mid m_t, x_{t-1}^{[j]})$. In other words, particle number j of prediction set is calculated using the j-th particle of the previous posterior distribution at time $t-1$ ($x_{t-1}^{[j]}$) and current motion data m_t. In line 5, a non-negative weight of the new particle is computed based on the observation likelihood. This is called the *importance factor*, denoted $w_t^{[j]}$. The importance factors for the entire particle set sum up to one. The importance factor of a particle is the probability that a measurement will produce data z_t if the target's state is $x_t^{[j]}$.

The second loop performs the *resampling* or *importance sampling* step. The algorithm fills the posterior set X_t with M particles drawn with replacement from the temporary set $\overline{X}_t$ based on their importance factor.

In some cases, the size of the belief particle sets at time $t-1$and time t can be different. The accuracy of the estimation increases with the number of particles. Although the estimation technique is sequential, the number of particles in belief set depends on both the computational resources and timing of new data. The primary goal of importance sampling is to approximate the most significant features so that the localization problem can be solved with the lowest computational cost [Kwok et al. 2003].

The particle filter is widely used in localization systems. This technique is accurate, robust and easy to implement, which allows integrating a variety of sensors for estimating non-linear, non-Gaussian systems. In this work, the localization model designed is based on a particle filter. The next section introduces MCL and basic concepts used in the localization technique needed for further discussions. For more details about the particle filter please refer to [Kwok et al. 2003], [Fox et al. 2003] and [Thrun et al. 2005].

2.1.3 MCL

Pedestrian indoor localization is the problem of determining the target's position in an indoor environment relative to a given map. In robot localization, using the Bayesian

filter algorithm in a localization problem is called *Markov Localization*. In this case, a map of the environment is a known argument. A realization of Markov localization based on the particle filter is called *MCL* and is one the most popular robot localization algorithms. The basic MCL algorithm is shown in Table 3. In contrast to pedestrian indoor localization, robot localization has reached some maturity due to widespread work in this area. To be consistent with robot localization techniques similar terms are used in this context.

Algorithm MCL(X_{t-1}, m_t, z_t, *map*)

1 $\overline{X}_t = \varnothing$
2 $X_t = \varnothing$
3 *for j=1 to M do*
4 $x_t^{[j]} = Sample\_Motion\_Model(m_t, x_{t-1}^{[j]})$
5 $w_t^{[j]} = Measurement\_Model(z_t, x_t^{[j]}, map)$
6 $\overline{X}_t = \overline{X}_t + \left\langle x_t^{[j]}, w_t^{[j]} \right\rangle$
7 *endfor*
8 *for j=1 to M do*
9 *draw i with replacement from probability* $\propto w_t^{[i]}$
10 $X_t = X_t + x_t^{[i]}$
11 *end for*
12 *return* X_t

Table 3 MCL (source [Thrun et al. 2005]).

This algorithm is similar to the particle filter algorithm, despite the map as a known argument. The Sample_Motion_Model algorithm calculates a new state of a particle. The motion model describes how a new state can be computed based on a previous state and the given motion data. The importance factor of the particle is computed by the *Measurement_Model* algorithm given current measurement data, the particle and map of the environment.

2.1.4 Indoor Maps – Virtual sensor

A map of the environment is needed for localization. The map is not only used for demonstrating the target's position, but also can be used for improving the localization

result. In a *map aiding* localization technique, a digital map is used as a virtual sensor and provides pseudo observations based on its geometry data to complement the information provided by other resources [Hofmann-Wellenhof et al. 2003]. The accuracy of a map enhanced localization technique depends on the accuracy and level of detail of the map. The map can be modeled and encoded in different formats.

Floor Plan - Raster Map

A two-dimensional floor plan can be represented in a raster graphic image. The raster map is mostly used for visualization. It is static and not scalable. Geometries inside the map may be detected based on the map's color schema. This encoding is normally not suitable for analyzing geometries.

IFC

The building sector's *Industry Foundation Classes (IFC)* is an open specification for *Building Information Modeling* [IAI 2009]. It is an object-oriented data model for describing and sharing information representing building elements, spaces, shapes and other properties of buildings [IAI 2009]. It is used regularly in the facility management industry [IAI 2009].

CityGML

City Geography Markup Language (CityGML) is an open data model and XML-based format standardized by the *Open Geospatial Consortium* (OGC) for the storage and exchange of virtual 3D city models [Gröger et al., 2012]. CityGML is a common semantic information model for the representation of 3D urban objects and an application schema for the Geography Markup Language version 3.1.1 (GML3). In CityGML, the building model is one of the most detailed thematic concepts represented in five levels of details. Briefly, in highest level of detail, thematic and spatial aspects of rooms, interior installations (like radiators and lamps), room furniture (like tables and chairs), floor, ceiling, interior walls, and so on can be modeled. A virtual 3D city that modeled by CityGML standard can store, represent and maintain by a *3D City Database* (3DCityDB) [3DCityDB 2012]. In addition, an implementation of a *Web Feature Service* (WFS) can be used with the 3DCityDB and then localization systems can use web services to obtain

details of a specific city object, providing thematic and spatial data required in a map aided localization system.

2.2 Localization System

This work aims to reduce uncertainty of a target's position in an indoor localization problem. The target is assumed to be carrying a host device that provides acceleration and orientation data. The proposed localization system consists of two major modules, illustrated in Figure 4. The first module is the *Movement Detector* that detects each step taken by the target using raw sensor data and consequently fires a corresponding *step event*. The step event records *step length (L)*, *changes in direction (δθ)* and *duration (t)*. The second module is the *Localization Model,* which receives step events to estimate a target's position by using a MCL reinforced by a movement model. Both modules are described in detail during the next sections.

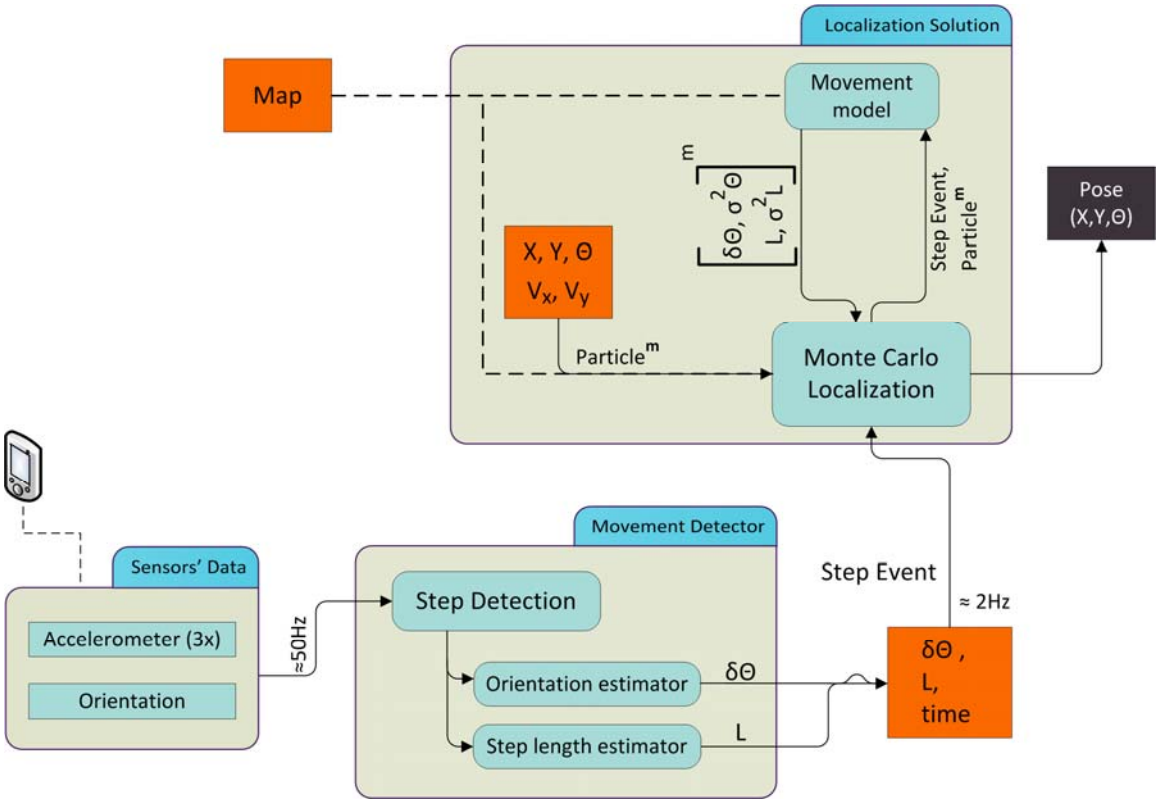


Figure 4 The main modules of the proposed localization solution are illustrated.

2.3 Movement Detector

Inputs of the Movement Detector module are the raw data from the sensors. Steps are detected based on acceleration data. The *Step Length Estimator* computes the *step length (L)*. The *Orientation Estimator* estimates direction of current step and calculates the *change in direction (δθ)*. In the following, each sub-module is described.

2.3.1 Step Detector

Steps are detected based on acceleration data acquired from a 3-axis accelerometer available on the host device. The acceleration data contains three major components: device translational acceleration, gravitational acceleration and external noise [Luinge 2002]. In order to obtain the translational acceleration, the contribution of gravitational acceleration must be eliminated. This can be accomplished by applying a high-pass filter to the magnitude of the acceleration data.

Usually, frequency domain analysis is used for detecting step occurrences in gait analysis using wearable accelerometer sensors [Takeda et al., 2009]. However, a frequency domain approach is inappropriate for this work, since the time intervals of acceleration data recorded by the host device are not equal and a real-time detection method is needed. Authors in [Chen et al., 2009] proposed an approach based on a sliding window, peak detection and zero-crossing in time domain with a satisfactory outcome. Nevertheless, their algorithm used a preset threshold for detecting step occurrences and experiments show that it should be adjusted for different people. The algorithm proposed in this thesis assigns a score to a sequence of acceleration data by examining different aspects of a step to achieve robustness.

Acceleration data must be smoothed to reduce the influence of noise. After smoothing, the data is checked for extreme values. A sequence of detected maxima and minima are assumed to express a step occurrence if they gain a proper score from Equation 5:

$$h(step_i) = \frac{score(step_i) + relative\_score(step_i)}{\sum_{j=1}^{5} W_j} \qquad \textbf{5}$$

Equation 5 compares quantities of detected maxima and minima and their time interval with predefined thresholds (i.e. $score(step_i)$) and the previous detected step (i.e. $relative\_score(step_i)$). Here, W_j is the weight corresponding to components.

$$score(step_i) = W_1 \times H1(step_i) + W_2 \times H2(step_i) + W_3 \times H3(step_i) \quad \textbf{6}$$

$$relative\_score(step_i) = W_4 \times H4(step_i) + W_4 \times H5(step_i) + W_5 \times H6(step_i) \quad \textbf{7}$$

The first component of Equation 6 is calculated by Equation 8. It scores the quantity of maximum acceleration using a preset threshold. The mathematical *min* function sets the upper limit of score given by this equation.

$$H1(step_i) = \min\left(1.5, \frac{A(\max)_i}{NormalQuantityThreshold}\right) \quad \textbf{8}$$

Equations 9 and 10 evaluate the quantity of minima with respect to the quantity of maxima and the time interval between them, respectively.

$$H2(step_i) = \min\left(1.5, \left|\frac{A(\min)}{A(\max)\times 0.1}\right|\right) \quad \textbf{9}$$

$$H3(step_i) = 1 - \frac{T(\min)_i - T(\max)_i}{PeakTimeThreshold} \quad \textbf{10}$$

Components of Equation 7 are calculated by Equations 11, 12 and 13. They score quantities of current maxima and minima as well as the time interval between them with respect to the previous step. Furthermore, the weight of the quantity scores (*H4* and *H5*) is calculated by Equation 14. In other words, quantities of extrema detected in previous step affect the calculated score if the time interval between two steps is small.

$$H4(step_i) = \min\left(1, \frac{A(\max)_i}{A(\max)_{i-1}}\right) \quad \textbf{11}$$

$$H5(step_i) = \min\left(1, \left|\frac{A(\min)_i}{A(\min)_{i-1}}\right|\right) \quad \textbf{12}$$

$$H6(step_i) = 1 - \frac{TwoStepTimeThreshold}{T(\max)_i - T(\max)_{i-1}} \quad \textbf{13}$$

$$W_4 = \min\left(W_6, \frac{TwoStepTimeThreshold}{T(\max)_i - T(\max)_{i-1}}\right) \quad \textbf{14}$$

2.3.2 Step Length Estimator

In [Chen et al. 2010], the authors compared linear, non-linear and artificial neural network models for step length estimation. All models contain user dependent parameters that should be determined during the training phase. Since the non-linear model consists of only one coefficient, it is suitable for implementing as a real-time estimator and requires less complicated training phase. The model is based on acceleration data and assumes the target's step length depends on the maximum and minimum acceleration in the step and a *user's individual factor* (Equation 15). The individual factor of the target person (*K*) differs from one another and should be estimated.

$$S = K\sqrt[4]{A_{\max} - A_{\min}} \quad \textbf{15}$$

The authors reported a 0.45% error in traveled distance for this model. [Alvarez et al. 2006] compared four different step length estimators, including the non-linear model, for wearable accelerometer devices. The estimator model shows a tendency to over-estimate the step length. The median value varies between 100% and 105% of the real distance. The other methods show a close result, but they depend on more than one coefficient.

The non-linear model is used in this thesis, and environmental constraints are assumed to influence the maximum and minimum acceleration of each step. The user's individual factor is estimated in pretests for each participant and is used during localization.

2.3.3 Orientation Estimator

The orientation estimator compares the heading of the user in two consecutive steps to calculate change in direction ($\delta\theta$) for the current step. The orientation estimator uses measurements of the magnetic field and accelerometer sensors to calculate the angle between an axis of the smartphone and magnetic north in each step, which indicates user's heading on that step.

Moreover, the step's heading is measured when the minimum acceleration of step is detected. This is roughly the time that the heel of swinging leg strikes on floor. In addition, sign of the $\delta\theta$ shows if changes are clockwise (+) or counterclockwise (-).

2.4 Localization Model

The localization model used is based on MCL. This is an implementation of a particle filter, which is often used for robot localization (see section 2.1.3 for details). In this section the localization model, its main components and an improved variant are explained.

As mentioned in section 2.1.2, the particle filter represents the belief by a set of state samples called particles. Each particle has a state vector

$$S = (x, y, \theta, v_x, v_y)$$

where (x,y) is position, θ is direction and (v_x,v_y) is velocity in each direction. Two-dimensional position and direction are the unknown parameters that must be estimated. As a result, each particle is a candidate for the target's current position.

The localization model is illustrated in Figure 4 and basic algorithm is shown in Table 4.

Algorithm MCL_with_Movement_Model(X_{i-1}, $stepEvent_i$, map)

1 $\overline{X}_i = \emptyset$
2 $X_i = \emptyset$
3 *for j=1 to M do*
4 $m_i^{[j]} = Movement\_Model(p_{i-1}^{[j]}, stepEvent_i, map)$
5 $p_i^{[j]} = Sample\_Motion\_Data(p_{i-1}^{[j]}, m_i^{[j]})$
6 $w_i^{[j]} = Measurement\_Model(p_i^{[j]}, p_{i-1}^{[j]}, map)$

7 $\overline{X}_i = \overline{X}i + \left\langle p_i^{[j]}, w_i^{[j]} \right\rangle$
8 *endfor*
9 *for j=1 to M do*
10 *draw k with replacement from probability* $\propto w_i^{[k]}$
11 $X_i = X_i + p_i^{[k]}$
12 *end for*
13 *return* X_i

Table 4 MCL algorithm improved by a movement model.

An iteration of the localization algorithm begins with receiving a new step event:

$$stepEvent = (L, \delta\theta, time)$$

where L is step length, $\delta\theta$ is change in direction and *time* is duration of the step. Particles inside the set X_{i-1} represent previous belief over the state vector. In the first loop, all particles will be moved according to a customized motion data, and their weight will be updated using their new state.

In line 4, the *Movement_Model* calculates an appropriate motion data ($m_i^{[j]}$) for each particle, given the particle ($p_{i-1}^{[j]}$), the current step event ($stepEvent_i$) and the map of environment (*map*). The motion data

$$m = (L', \sigma^2 L', \delta\theta', \sigma^2 \delta\theta')$$

contains parameters for specifying two Gaussian distributions, one over the step length and the other over the change in direction. The movement model takes natural human habits, such as avoiding an obstacle, in addition to environmental constraints, like a wall, into account by using a scoring system to determine the Gaussian's variances. In other words, the movement model utilizes a map to check environmental constraints. In line 5, the *Sample_Motion_Data* computes the new state of the particle ($p_i^{[j]}$) by sampling both Gaussians represented in its customized motion data ($m_i^{[j]}$). The particle's weight ($w_i^{[j]}$) is then calculated by *Measurement_Model* using a map of the environment. The weight or importance factor is computed based on the probability of transitioning to the new state given the previous state and feasibility of new particle state, both with respect to environmental constraints. In line 7, the new particle and its weight are added to a temporary set $\overline{X}_i$.

In the second loop, the temporary set of particles $\overline{X}_i$ is re-sampled to create a set of particles representing the current belief state, X_i. The posterior set X_i will contain *M* particles drawn with replacement from $\overline{X}_i$ based on their importance factor obtained from the measurement model. As a result, particles with higher weight may appear in final set X_i frequently while those with lower weight may disappear.

In the particle filter, the size of sample sets (*M*) plays a crucial role in the accuracy of estimation and the amount of computational resources required. The accuracy of the estimation increases with the number of particles, but so does the computational resource requirement. In the basic form, a constant number of particles can be employed. However, this is normally inefficient as the number of particles needed depends on the current state of system. In this work, an adaptive particle filter approach, called KLD-sampling, is used. KLD-sampling is explained in Section 2.4.4.

Prior knowledge about the target's starting point affects the particle distribution during initialization. In the initialization step, particles should normally be distributed over all dimensions of the state vector, *i.e*. two-dimensional position and direction. However, prior knowledge will restrict the particle's distribution.

In an extension of the basic filter, a limited number of randomly distributed particles can be added at the end of an iteration. This allows recovery from global localization failure, and is used in robot localization to overcome the kidnapped robot problem.

In next sections, main components of localization model are described in detail.

2.4.1 Movement Models

The movement model takes both behavioral and environmental constraints into account to evaluate the step event for each particle. These models compute appropriate motion data for each particle, given the current step event and the map. The motion data specifies the step length and the direction of current movement with two Gaussian distributions, one for each component. Generally, the movement model intends to determine the customized uncertainty of the incoming step event for each particle and express it in width of the Gaussian distributions. An illustration of the combination of the two Gaussians would result in a "banana-shaped" distribution. The Gaussians are sampled in

Sample_Motion_Data to compute next state of the particle (see section 2.4.2). In the following, three different movement models designed for this work are described.

Naive Movement Model

The naive movement model relies on the detected step event. The mean values of distributions come directly from the step event and two small constant values specifying the variance of distributions. Adding a small amount of uncertainty is crucial for the particle filter as duplicated particles are distributed in the re-sampling phase around states carrying huge weights. Therefore, they represent slightly different states.

Detective Movement Model

In the detective movement model, the mean values of the Gaussian distributions are equal to the mean values of the step event. However, the variances are partly estimated. The detective movement model temporarily calculates next state of the given particle using the exact values of the incoming step event. Afterwards, the model evaluates feasibility of the resultant state by considering environmental constraints. The same set of criteria as the measurement model is used (see Section 2.4.3 for more details). The variances of the distributions are determined by the evaluation result. The more feasible the calculated state, the smaller the variance will be. In other words, the detective movement model returns distributions with large variances when adding up the incoming step event to the current state of particle results an unfeasible state. Wide Gaussian distributions increase particle's chance of survival.

Modifier Movement Model

The modifier movement model evaluates the feasibility of particle's next state like detective movement model. However, it subsequently modifies both the mean and variance of the Gaussian distributions to increase the particle's chance of survival.

This model is allowed to modify the step event within a bounded range. The algorithm evaluates all acceptable states, given the current state and the step event. Thereafter, it partitions accepted states using connectivity based clustering. This model assumes the largest cluster is most proper area and that the particle may survive there. The algorithm sets the mean and variance of the Gaussian distributions by approximating the center of the largest cluster.

Figure 5 illustrates five steps of a particle in straight line. Dark blue points are positions of the particle at each step, light blue points are points suggested by this movement model, green lines show the direction of the particle, the brown areas are potentially accessible to the target and the red areas are not reachable due to the wall blocking the path. The modifier movement model customizes the straight-line step events for this particle. As illustrated, the direction and step length are changed after the third step. The expected trajectory of particle without using the modifier movement model is illustrated by dashed gray line.

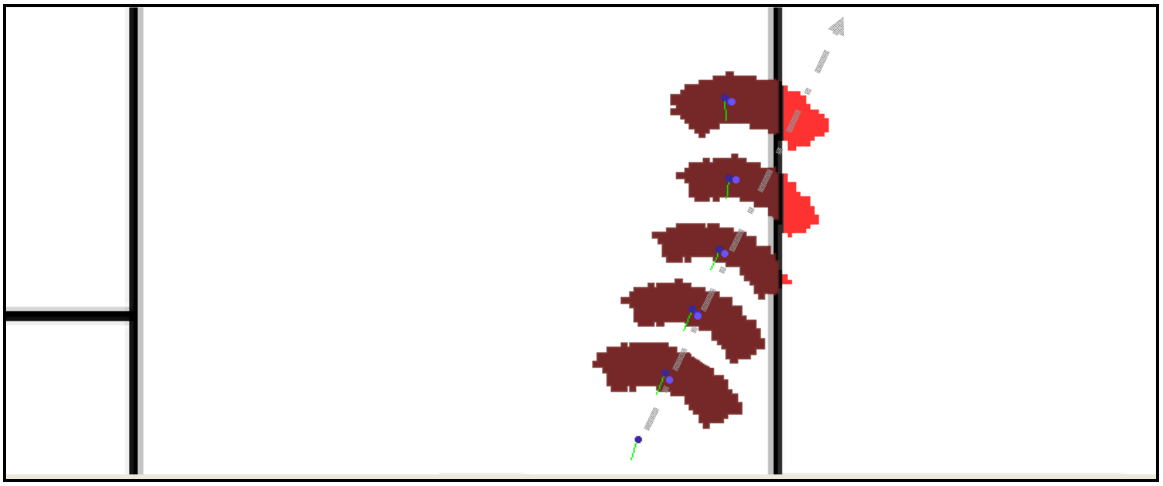


Figure 5 The effect of using the modifier movement model. The step events indicate a straight walk, but the modifier movement model has corrected the events to increase the particle's chance of survival.

2.4.2 Sample Motion Data

Sample motion data computes the new state of the particle given its previous state $p_{i-1}^{[j]}$ and customized motion data

$$m = (L', \sigma^2 L', \delta\theta', \sigma^2 \delta\theta')$$

where $(L', \sigma^2 L')$ is mean and variance of the Gaussian distribution over the step length and $(\delta\theta', \sigma^2 \delta\theta')$ is mean and variance of Gaussian distribution over the changes in direction. Current step data, length L'' and changes in direction $\delta\theta''$, are drawn from the normal distributions $N(L', \sigma^2 L')$ and $N(\delta\theta', \sigma^2 \delta\theta')$, respectively. The new state is then calculated from previous state and the current step data:

$$\theta_i = \theta_{i-1} + \delta\theta''$$
$$x_i = x_{i-1} + L'' \times \cos(\theta_i)$$
$$y_i = y_{i-1} + L'' \times \sin(\theta_i)$$
$$v_x = \frac{|x_i - x_{i-1}|}{time}$$
$$v_y = \frac{|y_i - y_{i-1}|}{time}$$

2.4.3 Measurement Model

As described before, the measurement model computes the importance factor of a particle and the probability of a state transition based on measurement data. In this work, a map of the environment provides measurement data as a virtual sensor (map aiding). Three different criteria score the current situation separately. The result of measurement is equal to the lowest score from all criteria.

State Transition Feasibility

State transition feasibility is evaluated with respect to environmental constraints using a simple criterion: if moving from the previous position of a particle to the new one would intersect with any obstacle, the weight of the particle should be set to zero. In other words, any movement that intersects an environmental constraint is unacceptable. In Figure 6(a), an unacceptable state transition of a particle is illustrated. Hence, the weight of the particle is set to zero.

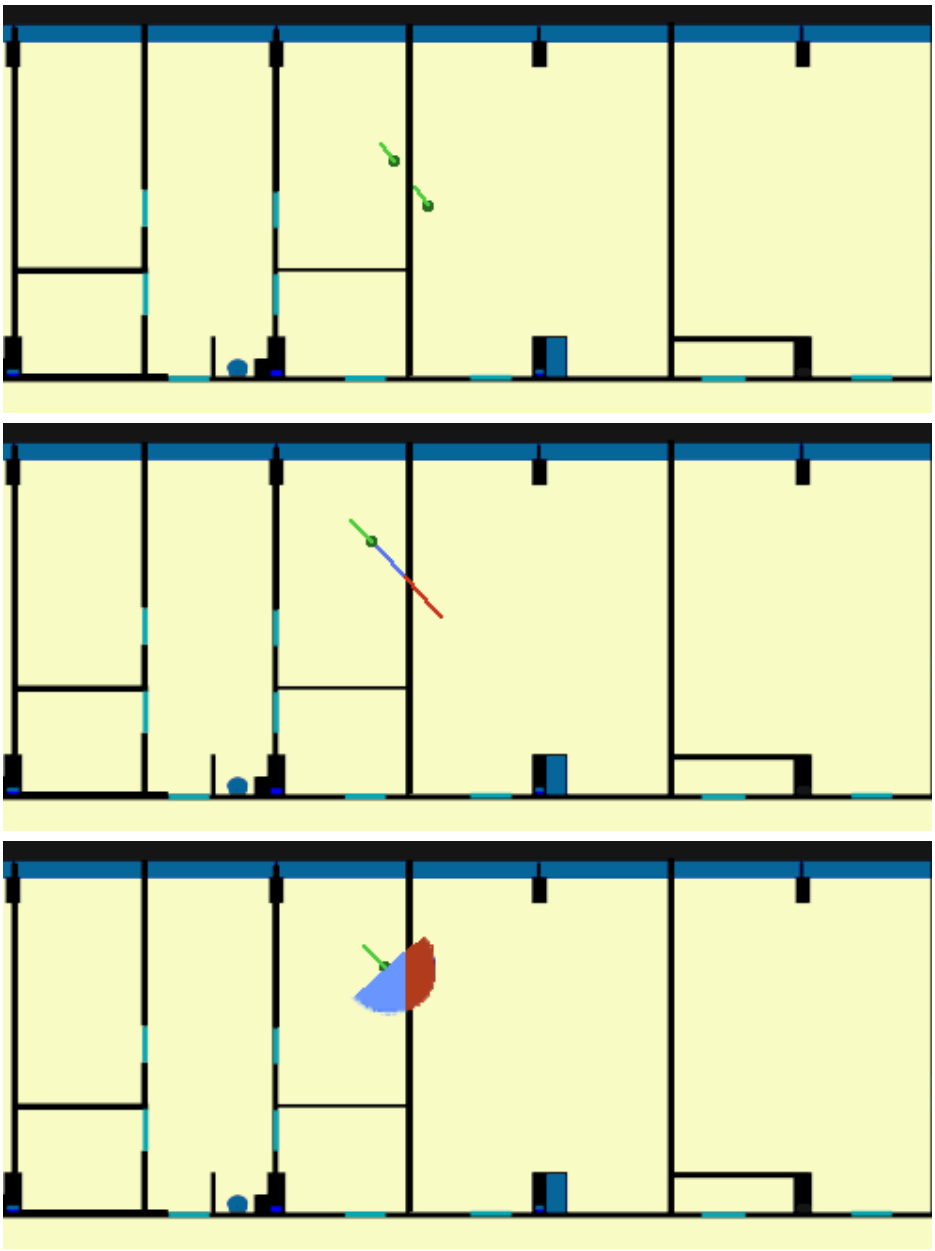
Feasibility of New State – Line of Sight

Naturally, human beings are not walking into an obstructed area. Thus, the feasibility is assumed to be directly related to the line-of-sight of the target. Consequently, the ratio of line of sight to a specific range is measured. Figure 6(b) illustrates line of sight, position and direction of a particle. The line of sight is marked by blue and red colors. The blue side shows amount of available line of sight. Roughly speaking, about 50% of the expected line of sight is acceptable.

Feasibility of New State – Obstacles Surrounding Position

It is assumed that human beings avoid walking into an area occupied by obstacles. Therefore, this criterion measures the ratio of accessible surrounding area of target

position to a circulation space. The circulation space is assumed to be a semicircle area with specific radius, in a direction that covers the user's viewpoint. In Figure 6(c), the circulation area around a particle is illustrated, and reachable area is marked by blue color.



(a)

(b)

(c)

Figure 6 The measurement model uses three criteria for evaluating the particle's state transition and its new state with respect to the map of the environment.

2.4.4 KLD-Sampling

In the particle filter, the size of the sample sets plays a crucial role in the efficiency of the filter. By increasing the size of the sample sets, the accuracy of estimation may be increased but more computational resources are needed, especially in global localization. KLD-sampling is a statistical approach that adapts size of sample sets during the estimation process [Fox 2003]. A particle set is a sample-based approximation of true posterior probability, containing approximation error. KLD-sampling measures the difference between the sample-based maximum likelihood estimate and the current approximation of the true posterior using the *Kullback-Leibler Distance*, and bounds this error by adjusting the quantity of particles over time.

Fox in [Fox, 2003] states the key idea of KLD-sampling as follows:

"At each iteration of the particle filter, determine the number of samples such that, with probability 1-δ, the error between the true posterior and the sample-based approximation is less than ε."

KLD-sampling assumes that the true posterior is given by a discrete, multimodal distribution. Suppose this distribution has b bins. If the number of samples, M_{req}, is selected with

$$M_{req} \approx \frac{b-1}{2\varepsilon}\left\{1-\frac{2}{9(b-1)}+\sqrt{\frac{2}{9(b-1)}}z_{1-\delta}\right\}^3, \qquad \textbf{16}$$

where $z_{1-\delta}$ is the upper $1-\delta$ quantile of the standard normal distribution, then the probability that the approximation error is less than ε is $1-\delta$ [Fox, 2003]. The values of $z_{1-\delta}$ are available in standard statistical tables. Several assumptions stated in [Fox, 2003] make it possible to derive this formula and an efficient implementation of it.

As it is seen from Equation 16, the required number of samples M_{req} is proportional to the inverse of the error bound ε, and is first order linear in the number of the bins with support b (*i.e.* bins containing at least one particle) [Fox, 2003]. The algorithm for KLD-sampling is shown in Table 5.

Algorithm KLD_Sampling(X_{i-1}, $stepEvent_i$, map)

// $X_{i-1} = \left\{ \left\langle p_{i-1}^{[j]}, w_{i-1}^{[j]} \right\rangle \mid j = 1,...,n \right\}$ represents the previous belief

// Global constants: bounds ε, δ, bin size Δ, and minimum number of particles

// M_{min}

// Initialization

1 $X_i = \varnothing$, $n=0$, $M_{req}=0$, $b=0$, $\alpha=0$

2 *create bins regarding to the size and set all of them empty*

3 do

4 *draw j with replacement from probability* $\propto w_{i-1}^{[j]}$

5 $m_i^{[n]}$ =*Movement_Model*($p_{i-1}^{[j]}$,$stepEvent_i$, map)

6 $p_i^{[n]}$ = *Sample_Motion_Data*($p_{i-1}^{[j]}$, $m_i^{[n]}$)

7 $w_i^{[n]}$ =*Measurement_Model*($p_i^{[n]}$, $p_{i-1}^{[j]}$,map)

8 $\alpha = \alpha + w_i^{[n]}$

// Insert the new particle

9 $X_i = X_i + \left\langle p_i^{[n]}, w_i^{[n]} \right\rangle$

10 if ($p_i^{[n]}$ *falls into empty bin B*) then

11 b=b+1

12 B=*non-empty*

13 if $b>1$ then

// Update the number of needed particles

14 $$M_{req} = \frac{b-1}{2\varepsilon}\left\{1 - \frac{2}{9(b-1)} + \sqrt{\frac{2}{9(b-1)}}z_{1-\delta}\right\}^3$$

15 n=n+1

16 while ($n<M_{req}$ or $n<M_{min}$)

// Normalize the weights

17 for $j=1$ to n do

18 $$w_i^{[j]} = \frac{w_i^{[j]}}{\alpha}$$

19 *end for*

20 *return* X_i

Table 5 KLD-Sampling algorithm improved by a movement model (source [Thrun et al. 2005] and [Fox 2003]).

In the algorithm, it is assumed that the bounding parameters ε and δ, the size of bins Δ and necessary minimum number of particles, M_{min}, are constant and defined globally, although they can be set as input arguments of the function. First, a fixed grid covering all dimensions of a particle's state is initialized for approximating the multinomial distribution. In each update of filter, all the grid's bins are set to empty (line 2). At each iteration through the loop, a particle ($p_{i-1}^{[j]}$) will be drawn with replacement from the set of previous particles based on their importance factor. A new particle ($p_i^{[n]}$) will be generated and added to the current set of particles (X_i) representing the current belief (line 5 to 9). Thereafter, the algorithm determines if the new particle falls in an empty bin (line 10). If so, the number of non-empty bins (b) is increased and the current bin (B) is marked as non-empty. Then, the requested number of particles (M_{req}) will be recalculated using Equation 16 (line 14). In line 16, the number of created particles in current particles set is increased and checked against both the minimum required particles (M_{min}) and the requested number of particles (M_{req}). In the last phase, the weight of particles will be normalized (line 17 to 19).

During the initial iterations of filter, the requested number of particles (M_{req}) will be greatly increased because newly generated particles usually fall into empty bins. However, after a while, the frequency of particles falling into an empty bin will strongly decrease and, consequently, the requested number of particles (M_{req}) will stabilize. KLD-sampling chooses a small number of samples when focused on a small part of the state space and chooses a large number if the state uncertainty is high [Fox 2003]. See [Fox 2003] for more details about KLD-sampling.

2.4.5 Detecting Convergence

Localization systems usually provide the current coordinate of a target for an end user or *Location Based Services* instead of representing the current belief by set of particles. Therefore, a method for detecting convergence to the most likely position of the target from particles' set is needed. The general idea is to partition particles into different

clusters based on the position they are representing. Afterwards, the center of mass of the cluster, which contains the largest proportion of the particles, should be calculated. When all of the particles are placed in a single cluster, the particle cloud has *converged* [Woodman 2010].

For clustering, a primary centroid based model is designed which assigns all particles to unspecified number of clusters. A particle will be drawn from the list of unassigned particles and added to a cluster if it is sufficiently close to the cluster's center point. Consequently, the center point of that cluster should be updated to show the weighted average of the position of all particles located inside of the cluster.

$$X_c = \frac{\sum_{i=1}^{N} w^i \times x^i}{\sum_{i=1}^{N} w^i}$$

$$Y_c = \frac{\sum_{i=1}^{N} w^i \times y^i}{\sum_{i=1}^{N} w^i}$$

$$\theta_c = a\tan 2(\sum_{i=1}^{N} w^i \times \sin\theta^i, \sum_{i=1}^{N} w^i \times \cos\theta^i)$$

In the case that the particle is not assigned to any available cluster, a new cluster will be created which contains that particle. The procedure will continue until all particles are assigned to clusters. Thereafter, the center point of the cluster with maximum number of members is determined to be the most likely position of the target.

Although the clustering model is elementary and incomplete, the algorithm is computationally efficient and produces acceptable results. Indeed, any other clustering algorithm that does not require a predefined number of clusters could be used. Moreover, the convergence detection algorithm is only used if the number of remaining particles is below a specific range to reduce computational complexity of localization system.

3 Implementation

3.1 Introduction

The localization system described in Section 2.2 was implemented in this work. Figure 7 illustrates the major components and data flow of the resulting system. The *localization model* realizes the model explained in Section 2.4, which estimates the target's current position by catching step events and satisfying environmental constraints. Two different components may provide step events, depending on user preferences. In simulation mode, the *simulator* component provides step events according to a predefined trajectory specified by the user. Using this component, the localization model can be examined in various synthetic environments and trajectories. The second component is the *real data initiator,* which provides step events based on an actual walk in a real-world environment. The *real data collector* is an Android application developed for this purpose. The *real data collector* stores required data in a file that will be post-processed by the *real data initiator* component. These two components allow the designed localization model to be examined in both synthetic and real conditions.

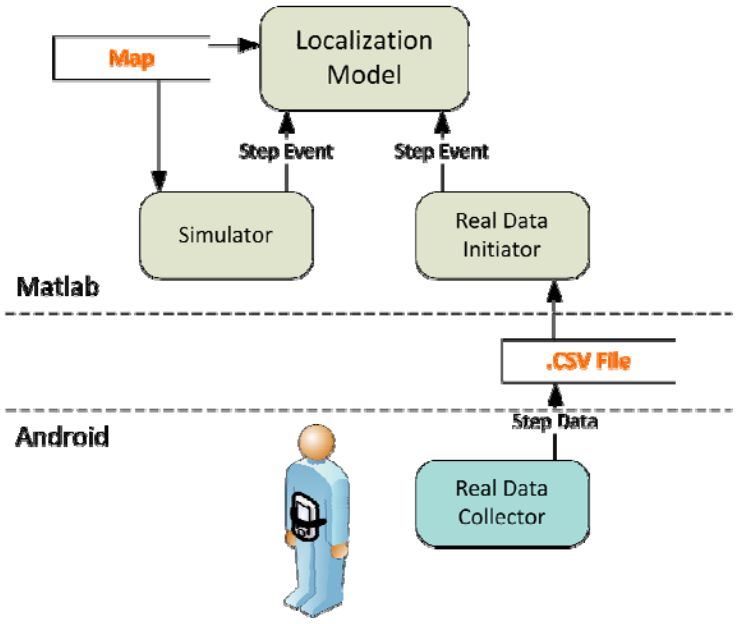


Figure 7 Main modules of the implemented system.

The localization model, simulator and real data initiator components are implemented in MATLAB. Figure 8 shows the graphical user interface (GUI) of the program during simulation mode. At the top is a map of the environment, and at the bottom are control and preference panels. The *running mode* radiobutton group switches between the step events providers. The right-hand side panel contains the preferences of the selected provider. Figure 8 illustrates the simulation mode. The *control* panel contains all required buttons for changing status of the step event providers and, consequently, the localization process. Moreover, it is possible to capture the floor plan while the system is running by checking the *Record Movie* checkbox. The *Global Localization* checkbox specifies the type of localization problem (global or local). As mentioned before, the target's starting point is approximately known in local problems, but in global localization problems the system does not have any prior knowledge about the starting point.

The user specifies the type of movement model. The model can be selected from the corresponding drop-down list. While the localization model is proceeding, relevant information will be shown in the message box, *e.g.* the number of particles currently approximating the target's position. The panels for the two step event providers are described in their own sections.

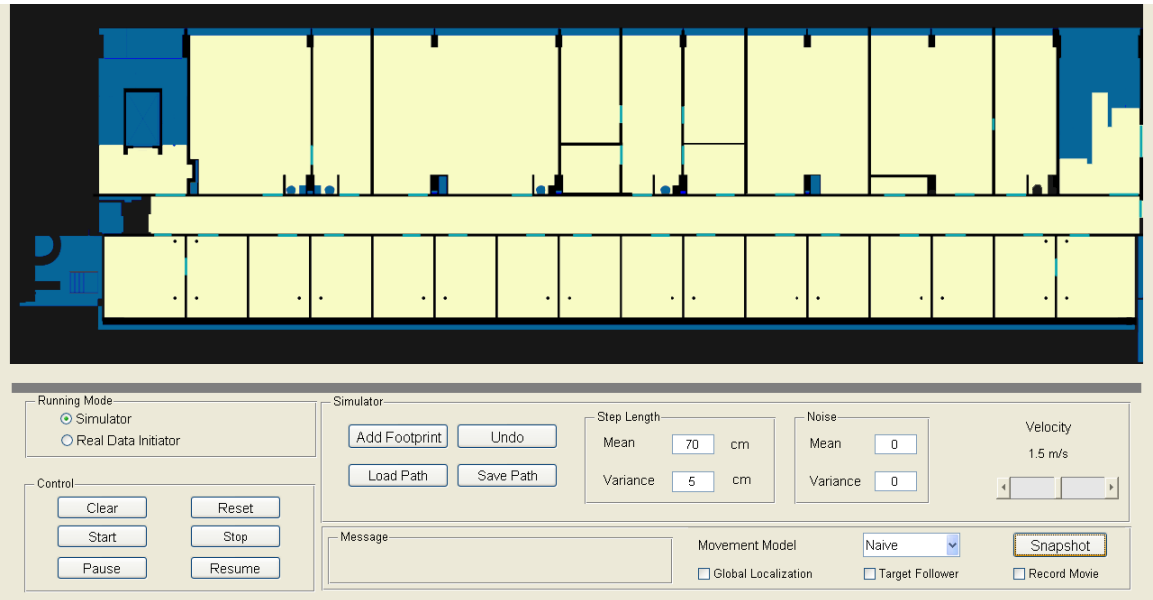


Figure 8 GUI of the program implemented in MATLAB.

The four major components are described in following sections from an implementation point of view.

3.2 Localization Model

The localization model explained in section 2.4 includes the MCL, KLD-Sampling, different movement models, the measurement model and the convergence detector has been implemented. The localization system is configured by a set of variables defined in the *globalfacts.m* MATLAB source file. Modifying corresponding configuration variables in this file can change the employed localization and movement models.

Initial distribution of particles depends on the type of localization problem. In the local case, particles will be uniformly distributed around a known starting point. Dimensions of the covered area and initial number of particles are specified by global variables. In the global case, particles will be uniformly distributed among the area the target is able to occupy, i.e. all regions of the environment that are not blocked by obstacles. Assuming a particle has three dimensions, the initial number of particles is equal to the minimum number of particles is required to occupy entire state space. Resolutions of initial particles' distribution in each dimension are specified by global variables. Therefore, the resolution of particle distribution is adjustable using the configuration file.

The basic MCL is an implementation of the Monte Carlo Localization algorithm described in 2.4. It is improved by adding randomly distributed particles among the state space after the resampling phase, which leads to recovery from global localization failures. The amount of random particles is specified by a global variable and is a percentage of the total number of particles. The KLD-sampling is implemented as described in 2.4.4. Default values of configuration variables, such as the minimum number of required particles, size of bins and bounding parameters, are taken from [Woodman 2010].

3.2.1 Environmental Constraints

In this work, raster graphic images representing floor plans are used. Generally, a pair of raster images must be specified by global variables. Both images are representing the

floor plan of same environment, one for rendering and the other for determining the importance factor.

The first image is used for visualizing the outcome of the localization system to the user. In this image, features can be represented by arbitrary colors. The image may also contain annotations and symbols for displaying features in a human readable format. Figure 9 illustrates an example image showing the floor plan of a synthetic office. Workstations are represented with light blue, and a cafeteria is indicated by a representative symbol. The current status of a room is shown as either light green or red, meaning free or in use, respectively. Although the applied color schema and annotations are human readable, it is an inappropriate representation for recognizing obstacles in a measurement model. Therefore, a basic machine-readable color scheme is defined for representing obstacles in the floor plan.

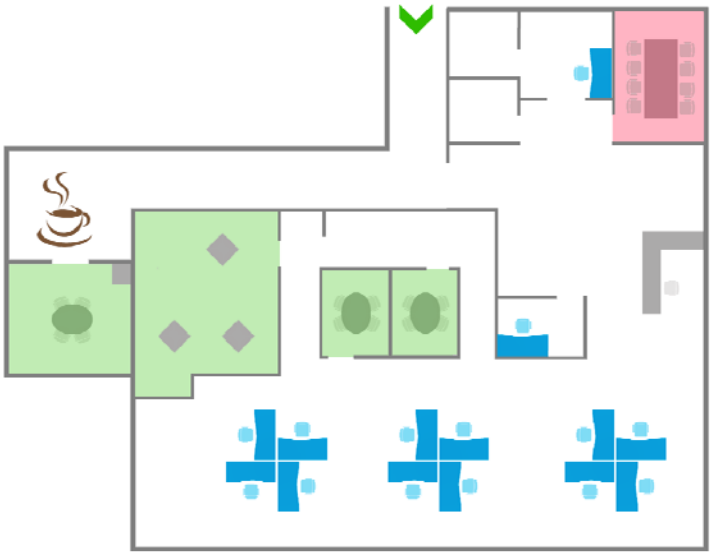


Figure 9 Floor plan of a synthetic office proper for visualization.

The second image is a grayscale image of the floor plan representing obstacles and spaces using a basic, machine-readable color scheme. Obstacles are marked with dark pixels (intensity below 50) and spaces are shown by bright pixels (intensity above 200). Features that the target may pass through, like doors, are represented by gray pixels (intensity between 50 and 200). Figure 10 illustrates a grayscale image of the same floor

plan. Features that a target cannot pass through are represented with black pixels. A door of the room in use is represented by darker color than a door of room that is free.

Using two different images allows a variety of annotations, colors, and line styles to be used when designing a customized floor plan while all measurements are taken in the background using the second image.

A scale factor for converting image's pixel size to a length should be specified using the global variable *pix2cm.* This must be set for each pair of images. As a result, the length in pixel units can be converted into centimeter units via multiplying by *pix2cm*.

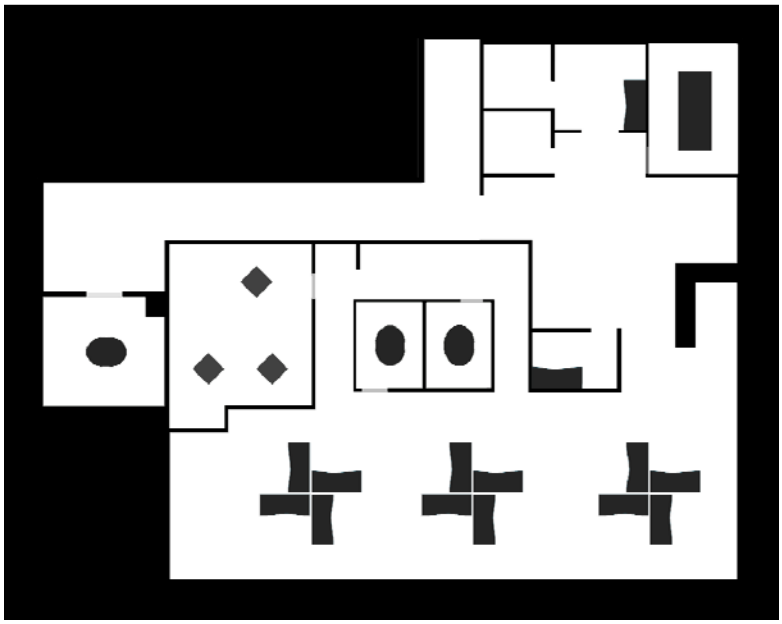


Figure 10 Floor plan of the synthetic office proper for using in map aiding section.

3.3 Simulator

The system architecture permits the designed localization model to be examined in a variety of different cases. In simulation mode, synthetic environments and trajectories can be employed to study the reaction of different movement models to arbitrary scenarios. To do that, global variables are used to specify the environment's map and a desired trajectory is drawn or loaded into the GUI. Thereafter, the simulator creates proper step events and passes them to the localization model component. The simulator also visualizes the result of localization model.

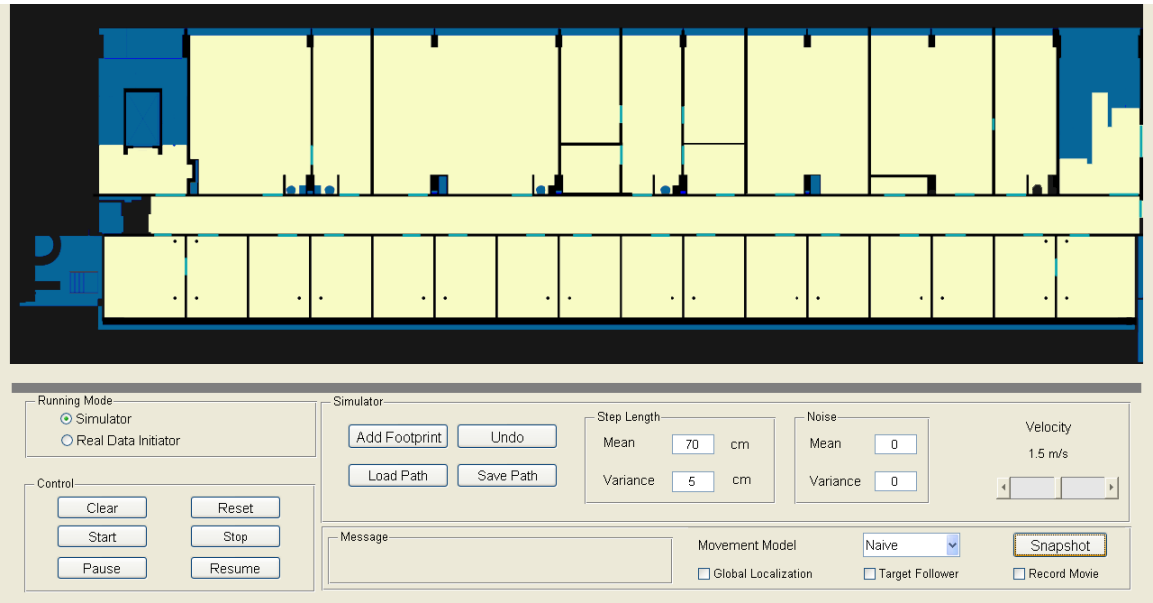


Figure 11 GUI of program running in *Simulator* mode.

For creating a trajectory, a set of tools, marked by 1 in Figure 11, are available. A trajectory is a polygonal chain that is specified by a sequence of points. Click on *Add Footprint* to start drawing a new trajectory. Upon clicking, the pointer changes to crosshairs. Then, clicking on the floor plan will add a new vertex to the polygonal chain. To remove the most recent point, click *Undo*. In addition, a created trajectory can be stored in a file for later use by clicking on *Save Path*. Similarly, a stored trajectory can be loaded by clicking on *Load Path*. Figure 12 illustrates an arbitrary designed trajectory.

By clicking on the *Start* button, the simulator begins to create step events and deliver them to the localization model component. The simulator creates an artificial target and tries to move as naturally as possible based on the trajectory drawn by user and environmental constraints. The artificial target moves step-by-step. The length of each step is basically drawn from a Gaussian distribution and updated based on environmental constraints. In the *step length* panel, the user should specify the mean and variance of the Gaussian distribution. Obstacles surrounding the new target position affect the step length based on two criteria. First of all, if an obstacle blocks the short-range line of sight of the target, the step length may be reduced up to fifty percent. Second, the smaller the circulation space around the new target's position, the smaller the step length and velocity.

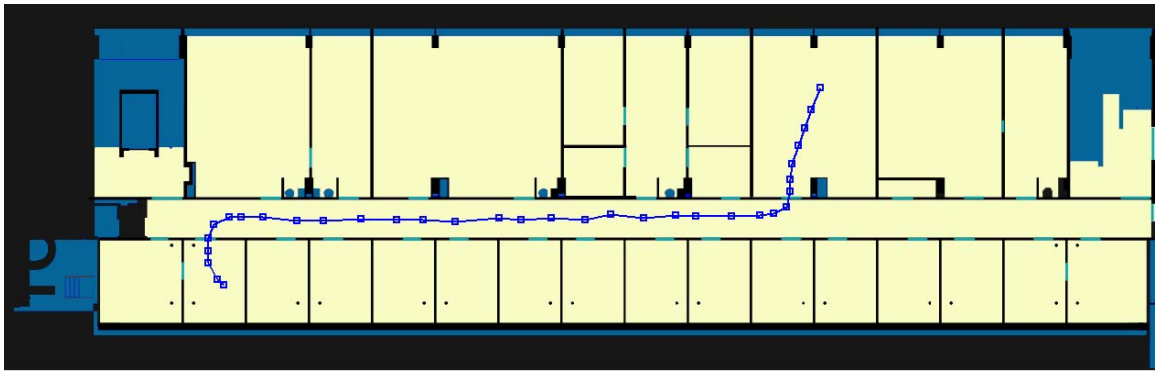


Figure 12 An arbitrary trajectory drawn by user.

Step heading is calculated so that the artificial target follows the trajectory as much as possible, and avoids unnecessary or sharp changes in heading. Small changes will be neglected until the sum of a set of small changes is larger than a threshold. Sharp turns will be divided into two or more medium turns. Figure 13 illustrates a sample trajectory drawn by user (blue polyline) and the path that an artificial target walked (red polyline).

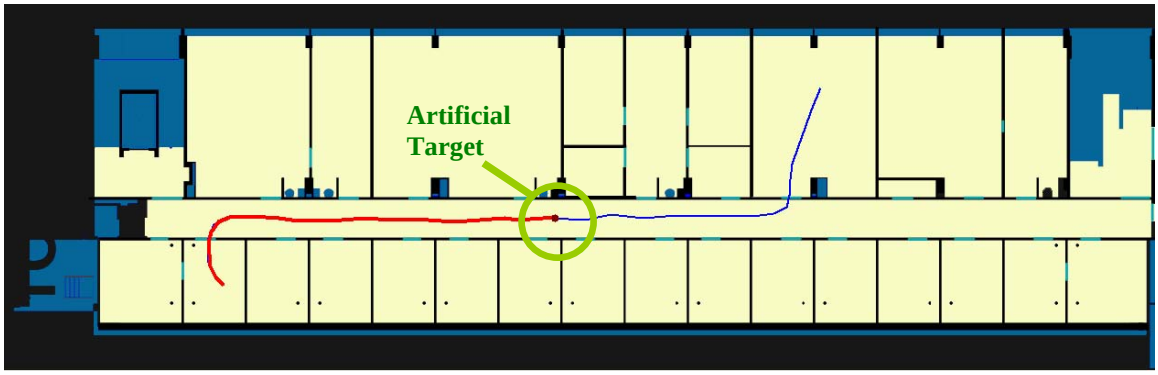


Figure 13 Simulation of a target person without localization. The user has supplied the trajectory (blue line). The red line shows the route traveled by the target.

The user also configures the walking velocity of target. The slider is marked by 2 in Figure 11 shows the velocity of the user in normal situation. However, the velocity of each step will be updated based on environmental constraints.

When the artificial target takes a step, a step event will be created. The step event contains length, changes in heading and duration of the step. However, the simulator adds noise to the step event data before delivering it to the localization model component. The

noise is added to all three components of step event to simulate sensor measurement error. This noise is drawn from a Gaussian distribution, which has parameters specified by the *Noise* panel in GUI (marked by 3 in Figure 11). If clean step events are desired, then the mean and variance of the Gaussian distribution should be zero. *Target Follower* visualizes the created step events after adding noise. During the simulation, a polyline is drawn showing the path based on the noisy step events. Figure 14 illustrates the trajectory drawn by user (blue polyline), the path taken based on correct step events (red polyline) and the path taken using noisy step events (green polyline). The errors are generated by a normal distribution, $N(10,400)$.

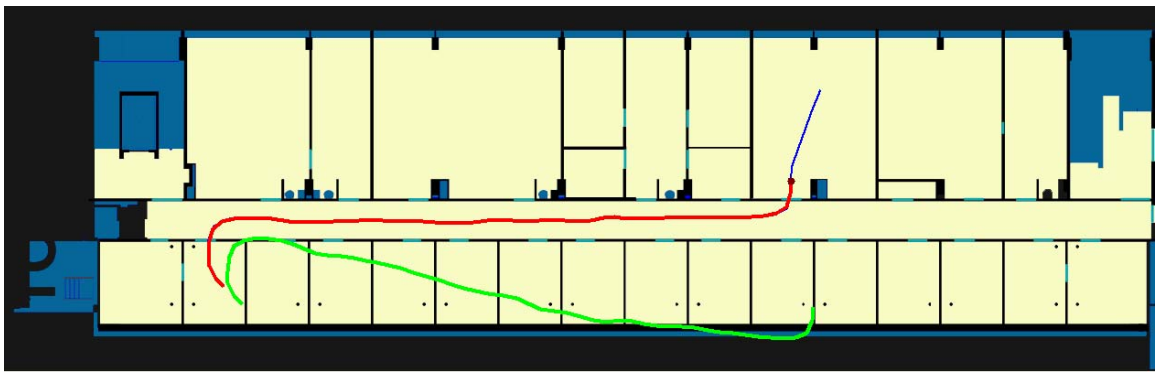


Figure 14 Simulation of a walk without localization. The green line illustrates the trajectory using noisy step events.

Results of simulation are presented in section 4.4.

3.4 Real Data Initiator

The real data initiator component generates step events based on a real walk taken in a real environment. As in the simulation mode, the map of the environment must be specified using global variables (see Section 3.2.1). As illustrated in Figure 7, this component depends heavily on step data. The step data must be collected from a real walk by *real data collector* and recorded in a comma-separated values (CSV) file. For each detected step, the step number, maximum and minimum acceleration, duration, orientation, start time and score are stored in the file. The real data initiator utilizes the step data file in addition to other input arguments to create step events and deliver them to the localization model component.

Figure 15 illustrates the program's GUI while running in the real data initiator mode. This component calculates the step length based on the method explained in Section 2.3.2, given the target's individual factor and step's data. The target's individual factor should be calculated one time for each target person during pretests. The complete procedure is explained in next section. The target's individual factor should be entered in the appropriate field in real data panel.

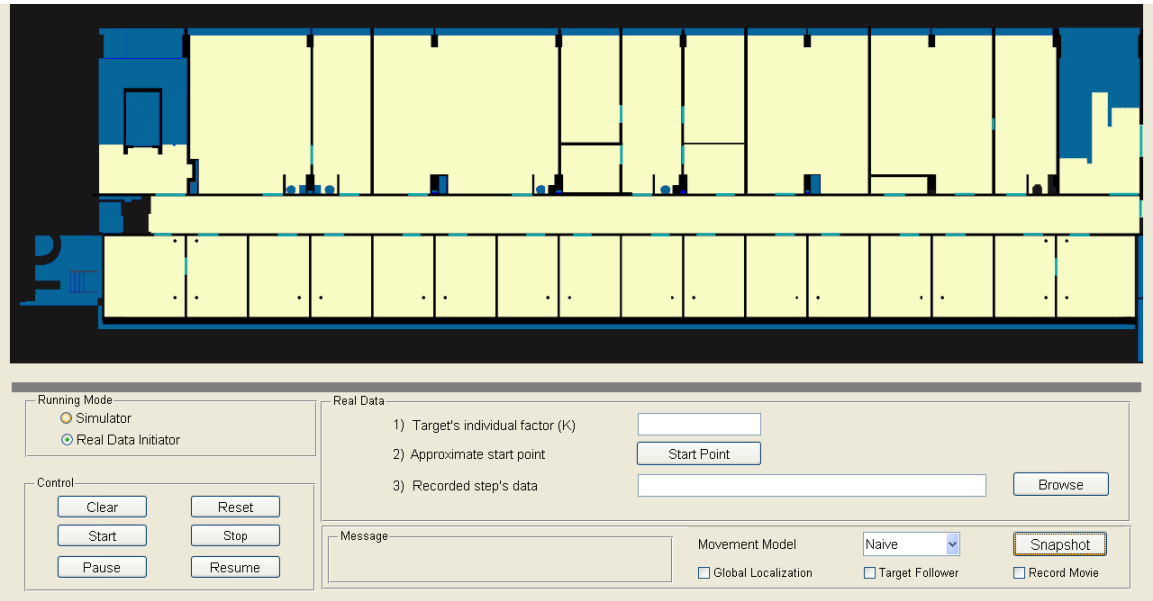


Figure 15 GUI of program running in *Real Data Initiator* mode.

The real data initiator supports both local and global localization problems. However, approximate coordinates of the starting point must be specified in both cases. By clicking on the *Start Point* button, the pointer changes to crosshairs and then the start point can be selected by clicking on the floor plan. In the local case, particles are initially distributed around the approximate starting point. In both cases, the initial point is used as a known start point in a PDR localization system running simultaneously with the designed localization model. The PDR system positions an artificial target at the starting point, which is then moved with respect to each step event. As a result, the difference between the PDR localization system and the designed system are instantly visualized.

Step events are created based on the step's data and the individual target factor, and are delivered to the localization model component. The outcome of the model and

approximate trajectory calculated by the PDR system are simultaneously visualized. Localization results using real data are presented in Section 4.5.

3.4.1 Estimating a Target's Individual Factor

For each user, an individual factor must be estimated in pretests. The individual factor is used for calculating the step length using Equation 15 during localization procedure. In the pretest, the user walks a predefined trajectory with a known distance. The walk is monitored by *real data collector,* like a normal localization procedure, which extracts the required features from detected steps and stores them in a CSV file. Thereafter, a MATLAB function (*computeK.m*) developed for estimating the factor should be called using the CSV file and the exact distance taken by the user. The function estimates the user's individual factor *k*

$$k = \frac{Dis}{\sum_{i=1}^{n} \sqrt[4]{A^i{}_{\max} - A^i{}_{\min}}} \qquad \textbf{17}$$

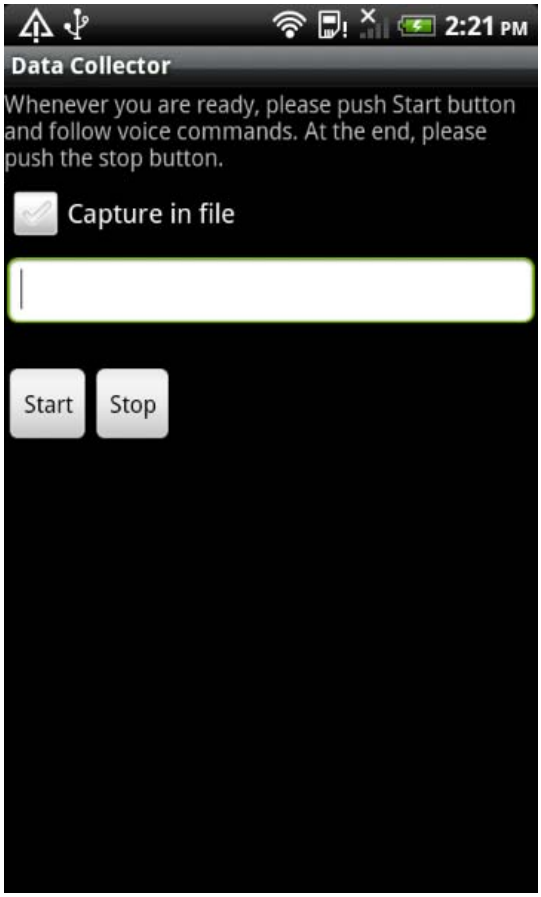
where A^i_{max} and A^i_{min} are the maximum and minimum accelerations detected from i-th step, *Dis* is a total distance taken by user and *n* is number of steps.

Other approaches for estimating the user's individual factor have been examined, such as using the user's fixed position during each step given by a GPS device. However the accuracy of the other methods was unacceptable.

3.5 Real Data Collector

The real data initiator and real data collector are implemented to assess the effectiveness of the localization model for a real walk taken in a real environment. The movement detector component of designed localization system (explained in Section 2.3) is realized by combination of real data collector and real data initiator. The data collector is part of an Android application developed throughout this thesis. The data collector was developed for recognizing a human gait and collects data while the phone is carried by the target. The application is compatible with Android version 2.2 and higher. It has been tested on HTC Desire, HTC Desire S and SAMSUNG GALAXY S II. Figure 16(a) illustrates the GUI of the Android application developed throughout this thesis.

(a)



(b)

Figure 16 *Data Collector* application developed for Android. (a) Application contains four different sub applications. (b) GUI of Data Collector.

The data collector uses 3-axis accelerometer and magnetic field sensors available in all device implementations of Android 2.2. Sensors data are available base on Event-driven architecture (EDA) in Android. Therefore, it is not possible to poll sensors. According to the Android 2.2 Compatibility Definition, device implementations are required to deliver accelerometers' events at a frequency of at least 50 Hz. Events, however, are not delivered in exact time intervals.

Figure 16(b) shows the GUI of data collector. Clicking on the Start button starts the data collection procedure. The user (target) must then follow application's voice commands, beginning with sensor calibration. The phone should be rotated around its three axes until the data recorded by the compass is of sufficient accuracy. The next message asks user to set up the phone on his/her abdomen with portrait orientation. After securing the phone

properly, the user should stand still for five seconds. When directed, the user can begin walking. The application recognizes steps and extracts required features while the target is walking. At the end of their walk, the user should push the stop button. The extracted features are written to a CSV file including step number, maximum and minimum acceleration, duration, orientation, start time and score of each detected step. The data collector uses the step detection model described in 2.3.1 for detecting and extracting features.

It is assumed that target's heading in a step is equal to phone's orientation at the time of detecting minimum acceleration in that step. The device's orientation is calculated in world coordinate system based on acceleration and geomagnetic data by following the procedure described in the documentation of SensorManager class of Android in [Android 2012].

4 Results

4.1 Introduction

Results of the developed and implemented localization system are described in following two sections. In Section 4.2, results of the movement detection module are explained and the results of the localization module are presented in Section 4.3. Note that the capability of this software exceeds the results presented in this thesis. A variety of test procedures across a wide range of scenarios would be planed based on implemented software.

In all test procedures where human gaits are observed, participants must set up the device following the instructions explained in Section 3.5. As the step detection algorithm is adjusted based on the vibration of the upper half of body, the device should be fixed to the abdomen (or lower lumbar spine) with an adjustable corset to avoid movement artifacts (see Figure 17).

Figure 17 Target person walking through corridor carrying the smartphone for collecting data.

Although the phone's orientation is not essential for step detection and length estimation, it is a crucial parameter for estimating the heading. Generally, the sensor's data are delivered based on the device's coordinate system, and the data collector then converts them into a global coordinate system. In order to do that, the data collector needs to know the current orientation of device. Therefore, it is assumed that the device is fixed in the upright position during tests.

4.2 Movement Detection

4.2.1 Step Detection

The designed and implemented step detection model (Section 2.3.1) is examined by measurements that were taken from a group of five men, ages 27 to 32. Height of the subjects varied from 174 cm to 192 cm. The subjects participated in three different tests. They all set up the phone according to the instructions and walked following different paths. Actual steps were counted and compared with the output of data collector. During first test procedure, they were asked to walk along a 52 m corridor following a straight path. In second test they entered and left a room, in addition to walking through the corridor. As a result, this trajectory contains at least four turns. During the last walk, participants were asked to take an arbitrary walk. Consequently, trajectories involve several turns with a variety of different velocities.

Participants repeated each experience twice. The program's configuration was equal for all participants during all test procedures.

Style	Average number of actual steps	Average Percentage Error
Straight walk	71.8	0.424%
Straight walk with some turns	76.5	0.922%
Arbitrary walk	97	2.754%

Table 6 Results of step detection module.

Table 6 shows the average percent error among different tests. Steps were successfully detected more than 97.2% of the time.

4.2.2 Step Length Estimation

The step length estimator was evaluated by a two-phase experiment. Participants were asked to walk following two specific paths, three times. The distance of each path was measured by hand. The first route was shorter (6.49 m) and straight. The second route was longer (11.74 m) and contained a turn.

Using Equation 17 (see section 3.4.1) the user's individual factor k was computed for each participant based on a set of observations recorded as users walked the first route. Thereafter, the traveled distance of the second route was estimated based on the computed user's factor and collected data from second route. Comparing the estimated and measured distances, a 4.304% error (arithmetic mean) was determined.

Additionally, a third test procedure was carried out three days later in a 52 m corridor with the same subjects. They were asked to walk along the corridor following a straight path three times. The actual traveled distances were measured by hand. The distances were also estimated using the previously calculated individual factors and collected step data. As a result, a 5.315% approximation error was determined. As a result by increasing the traveled distance, the error of the total step length estimation has been increased.

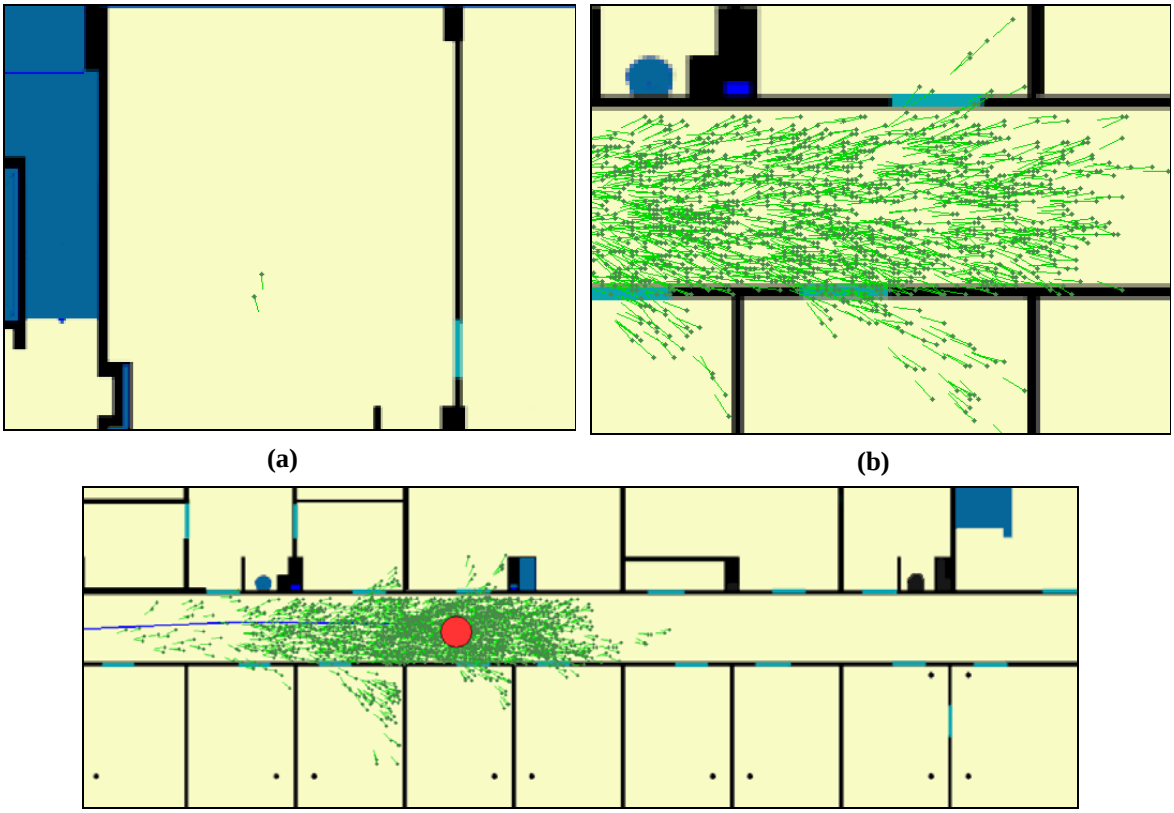
4.3 Localization

The implemented localization model offers a wide variety of configuration parameters. It can tackle a local or global localization problem by employing three different types of movement models for processing both synthetic and real data. This section focuses on the results of the KLD-sampling algorithm for each movement model. Each model was tested on both synthetic and real walking data. During the localization process, the program captured a movie, took snapshots for each step and generated a report. The generated data amounted to more than 21 GB. Therefore, a small number of snapshots are presented in the thesis and some others are available on the DVD attached to the thesis.

Configuration parameters of the KLD-sampling and the initial distribution of particles in the local problem were constant in all following tests. For KLD-sampling, the minimum number of required particles was set to be 300. Three-dimensional bins were created over state-space with size $200cm \times 200cm \times 30^{\circ}$. Each bin has three dimensions: two lengths

and an angle. In addition, the bounding parameters were chose to be ε = 0.015 and δ = 0.01. When initializing the filter, particles were normally distributed in an $8m \times 8m$ area surrounding the position that user had marked as the start point.

A particle is represented by a green dot with a tail that indicates the current heading of the particle. Figure 18(a) illustrates two particles and Figure 18(b) shows a particle cloud. When the number of particles falls below 5000, the convergence detection algorithm is used for estimating the most likely position of the target given the current particle distribution. A red or green filled circle represents the target's estimated position. Certainty and uncertainty of estimation is shown by the red and green colors, respectively. Figure 18(c) illustrates target's estimated position computed from the particle distribution during the localization process.



(a) **(b)**

(c)

Figure 18 Aspects of the localization process.

4.4 Synthetic Environments

The localization model was tested on a synthetic data by using the simulator. First, a floor plan of a synthetic office was imported into the program using global variables (see Section 3.2.1 for more details). Then, a sample trajectory was drawn using the simulator's GUI (see Section 3.3). In the sample scenario, an employee (target) walked from his workspace down the corridor to the cafeteria, and then walked to a room for a meeting. Figure 19 illustrates the trajectory drawn by the user with a blue line. As mentioned in Section 3.3, the simulator provides step event by moving an artificial target from the starting point to the end point following the specified trajectory.

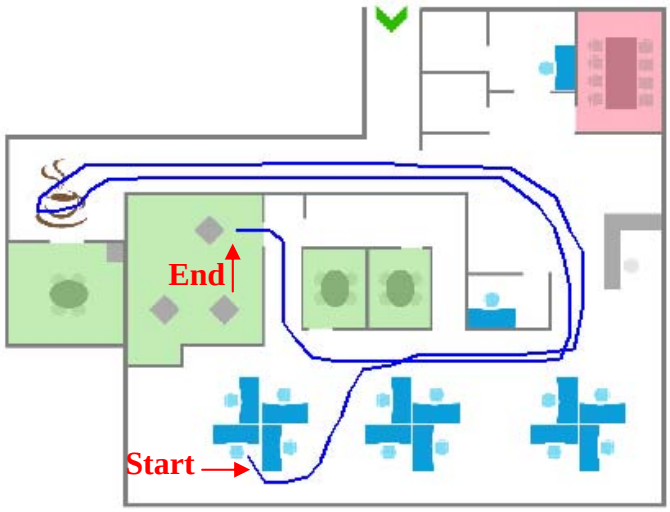


Figure 19 A synthetic trajectory in a synthetic office environment.

The simulator has been configured to draw step lengths from a Gaussian distribution with $\mu=70$ and $\sigma=5$. Each generated step event moves the artificial target one step forward. However, the step events were manipulated by introducing errors before delivering them to the localization model. The errors were drawn from a Gaussian distribution with $\mu=0$ and $\sigma=8$ and added to both the step length and changes in heading of each step event. Moreover, the target follower mode of simulator was active. As a result, the manipulated step events were not only delivered to the localization model, but also to a PDR system.

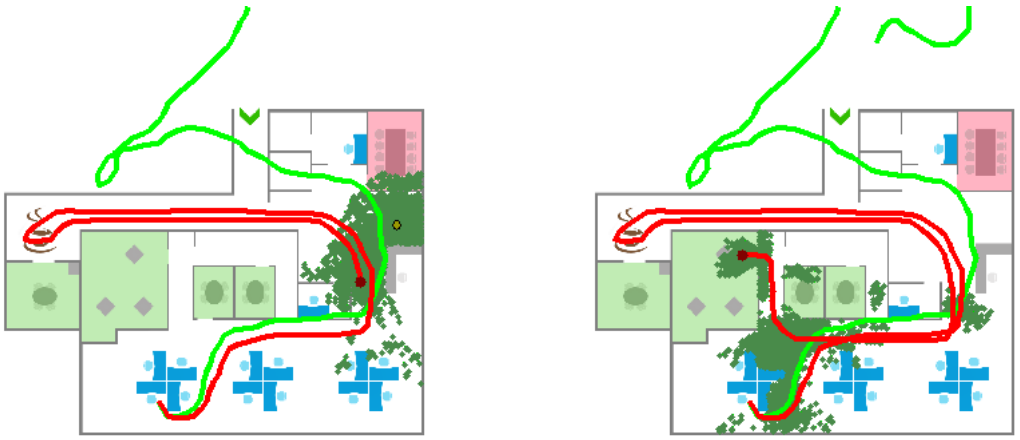
This system estimated the target's position and its results are drawn with a green line during localization. In addition, the simulator was configured to generate same input data for all three movement models, i.e. the same particle distribution in initialization part and same step events.

In the following, results of localization processes for each movement model are presented separately.

4.4.1 Naive Movement Model

The naive movement model was used during a localization process for the example problem explained above. As described in Section 2.4.1, the naive movement model draws two random numbers from two Gaussian distributions to slightly change the values (length and changes in heading) of the step event for each particle. For each particle in this test, a normal distribution with μ=0 and σ=25 was used for step length manipulation, and a normal distribution with μ=0 and σ=8 was used for customizing the changes in heading. The sample test was 150 steps long. Figure 20 illustrates the simulation after 33, 80, 115 and 150 steps. The particle distribution, current position of target and position estimated by both the localization model and the PDR system are illustrated in each image.

(a) **(b)**



(c) **(d)**

Figure 20 Snapshots from outcome of the localization system enhanced by the naive movement model. Particle distribution, correct traveled trajectory (red line), estimated trajectory by PDR (green line), current correct position of user (brown circle connected to the red line) and current estimated position of the target by localization system (red or green circle) are illustrated. State of environment in step 33 (a), 80 (b), 115 (c) and 150 (d) are shown.

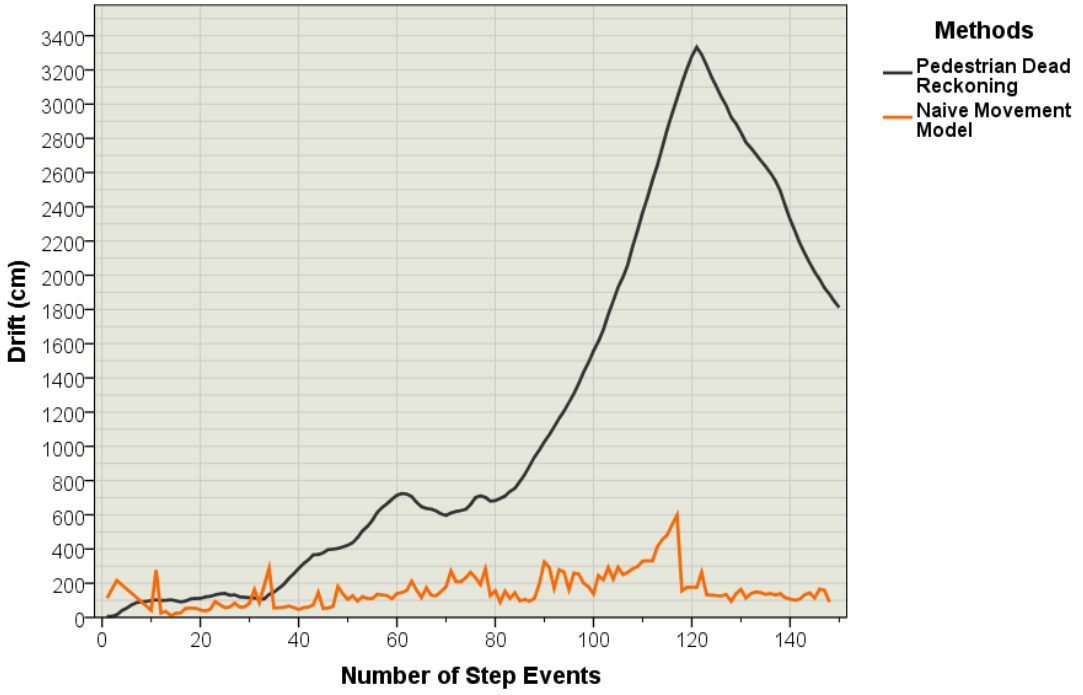


Figure 21 Drift of PDR and localization system enhanced by the naive movement model for processing the simulation data.

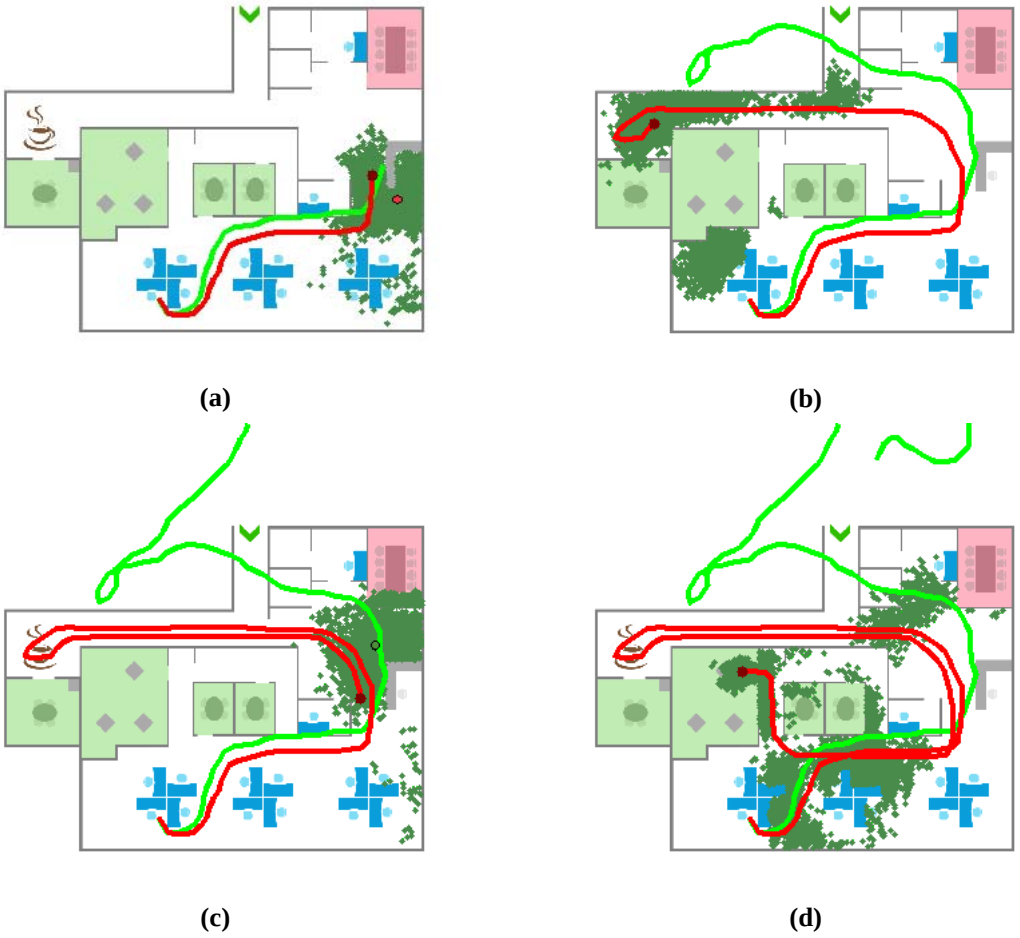
Figure 21 plots the two-dimensional drift in the estimated positions by the localization model with the naive movement model and the PDR system. The drift is assumed to be the Euclidean distance between the estimated and true positions. Note that the exact starting point was given to the dead reckoning system, but in the localization system all surrounding points (in an 8 m range) have a same probability to be the correct start point. The localization model estimates targets position by detecting particles convergence in 138 points out of 150 steps.

As it can be seen in Figure 21, the drift in the naive movement model was higher between steps 60 and 80 when compared to most other positions. That period is when the target was in the cafeteria, which involved moving from a narrow environment to a broad area.

4.4.2 Detective Movement Model

The detective movement model was tested on the explained localization problem. As described in Section 2.4.1, the detective movement model also changes the step event's data for each particle. For each particle, it adds two random numbers drawn from two Gaussian distributions to the values of each step event (length and changes in heading). However, the parameters of normal distributions are variable for all particles. Parameters depend on how satisfactory the next state of the particle would be if it were directly calculated using the pure step event. In this test, mean values of both distributions were chosen to be zero. The standard deviations' range of normal distribution used in combination with the step length was from 20 cm to 30 cm, and the standard deviation of distribution used for customizing the changes in heading was from 5 to 11 degrees. As a result, the more satisfactory the pure step event for current particle, the narrower the normal distributions.

Figure 22 illustrates the simulation after 33, 80, 115 and 150 steps. The sample test was 150 steps long. The particle distribution, current position of target and position estimated by both the localization model and the PDR system are illustrated in each image.



(a) **(b)**

(c) **(d)**

Figure 22 Snapshots from outcome of the localization system enhanced by the detective movement model. Particle distribution, correct traveled trajectory (red line), estimated trajectory by PDR (green line), current correct position of user (brown circle connected to the red line) and current estimated position of target by the localization system (red or green circle) are illustrated. State of environment in step 33 (a), 80 (b), 115 (c) and 150 (d) are shown.

Figure 23 illustrates the Euclidean distance between the true position and the position calculated by the localization model with the detective movement model (orange line) and the PDR system (black line). The localization model estimates targets position by detecting particles convergence in 92 points out of 150 steps.

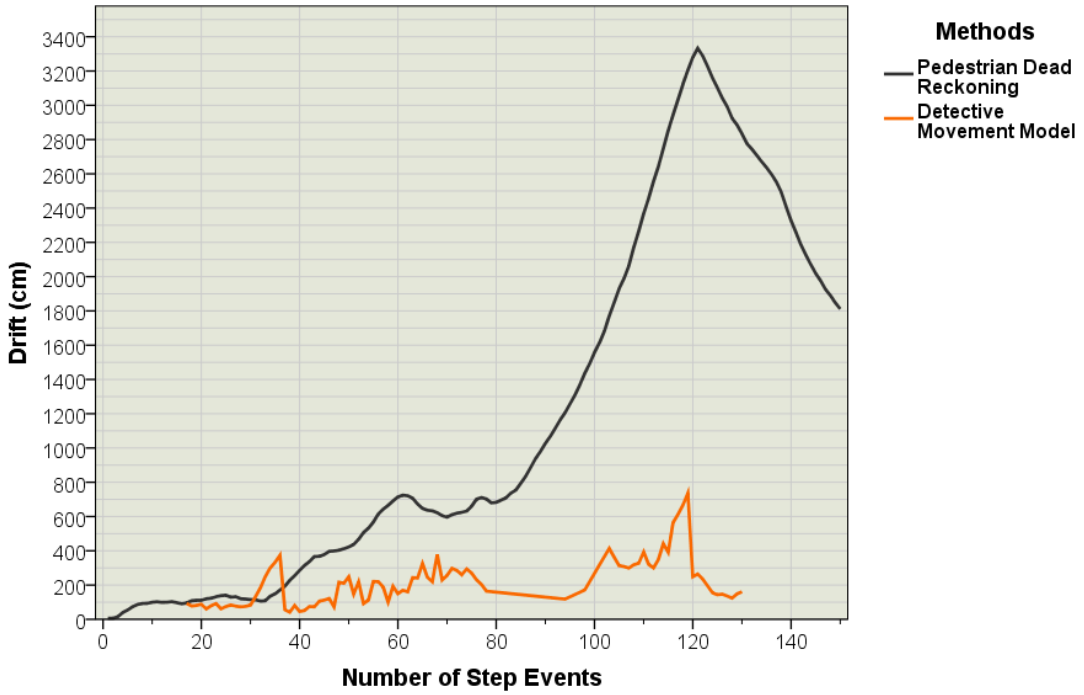


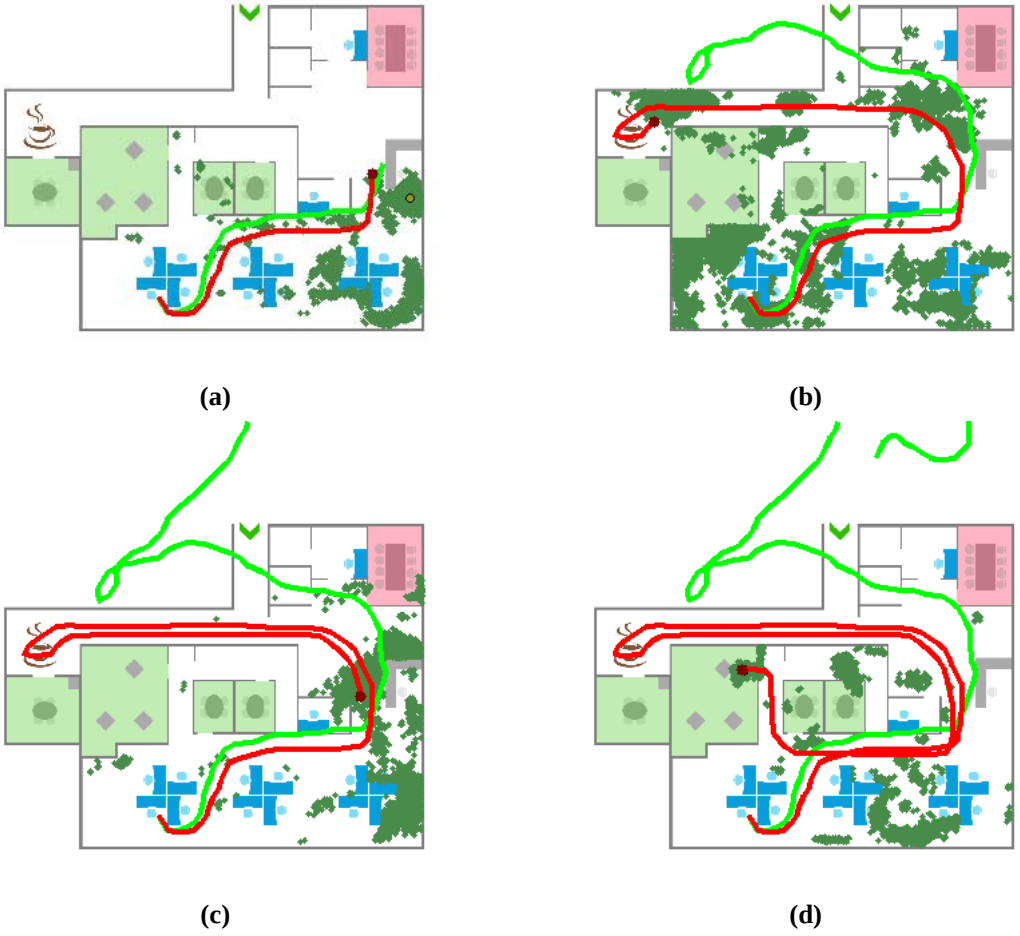
Figure 23 Drift of PDR and localization system enhanced by the detective movement model for processing the simulation data.

4.4.3 Modifier Movement Model

As described in Section 2.4.1, the modifier movement model attempts to update the state of each particle by using the current step event such that the particle's chance of survival is increased. Therefore, the model inspects a predefined range around the values of the step event to find the best state for each particle after updating. In this test, a range of 50 cm for step length and 50 degrees for changes in direction were checked. Moreover, it was assumed that the step length of human being is between 50 cm and 110 cm.

After finding the proper step length and change in direction for each particle, a small portion of randomness was added to the step event's values to avoid duplicate particles in posterior. The random values were drawn from a normal distribution with $\mu=0$ and $\sigma=12$ for step length, and from a normal distribution with $\mu=0$ and $\sigma=3$ for changes in direction.

Figure 24 illustrates the outcomes of simulation after 33, 80, 115 and 150 steps. The sample test was 150 steps long.



(a) **(b)**

(c) **(d)**

Figure 24 Snapshots from outcome of the localization system enhanced by the modifier movement model. Particle distribution, correct traveled trajectory (red line), estimated trajectory by PDR (green line), current correct position of user (brown circle connected to the red line) and current estimated position of target by localization system (red or green circle) are illustrated. State of environment in step 33 (a), 80 (b), 115 (c) and 150 (d) are shown.

Figure 25 illustrates the Euclidean distance between the true position and the position calculated by the localization model with the modifier movement model (orange line) and the PDR system (black line). The localization model estimates targets position by detecting particles convergence in 58 points out of 150 steps.

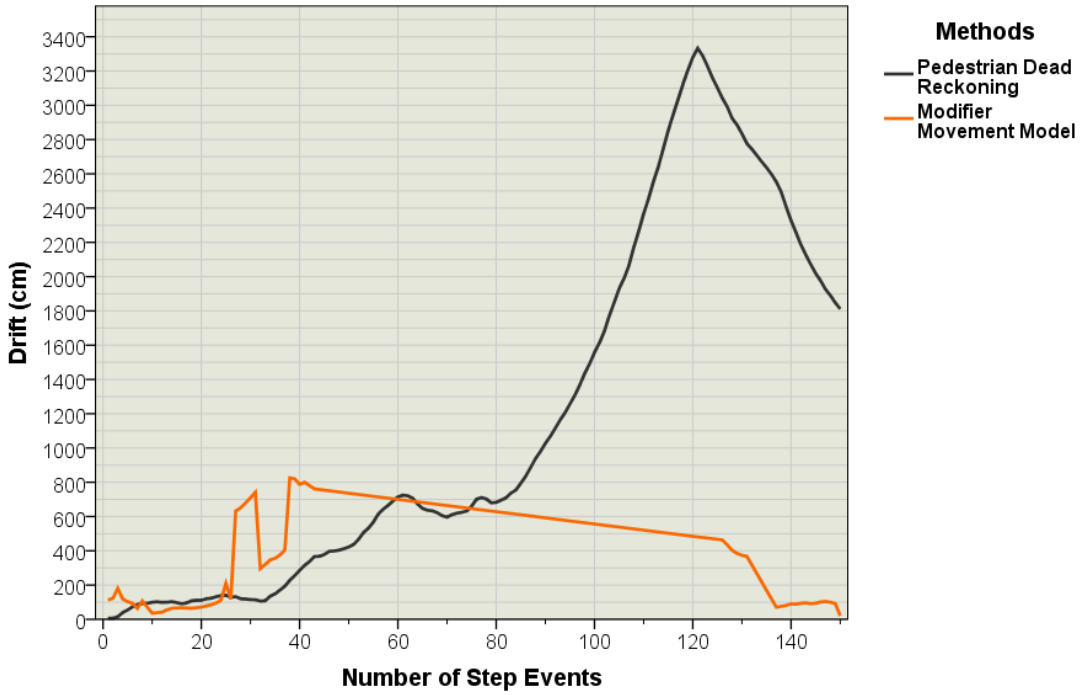


Figure 25 Drift of PDR and the localization system enhanced by the modifier movement model for processing the simulation data.

4.5 Real Observations

The localization system was also tested on a set of observations recorded as a user walked along a path through the 6th floor of main building of Technical University of Berlin. The approximate trajectory is illustrated in Figure 26. During the test procedure, the subject started from a point inside the GIS laboratory (marked by a green point) and visited seven other rooms. The room selection was subject to availability at the test time. The trajectory was 242 steps and about 170 m long. The user set up the smartphone following the instructions and walked freely between rooms.

Nine benchmark points were considered in the path. The user was required to stop when he reached the nearest place to a benchmark point in his path. The user's position at that point was measured by hand and recorded with its corresponding step number. Thus, the real position of the user at nine points and their corresponding step numbers are known. These nine benchmark points are marked in Figure 26 by filled red circles. These points are used for evaluating the accuracy of localization system.

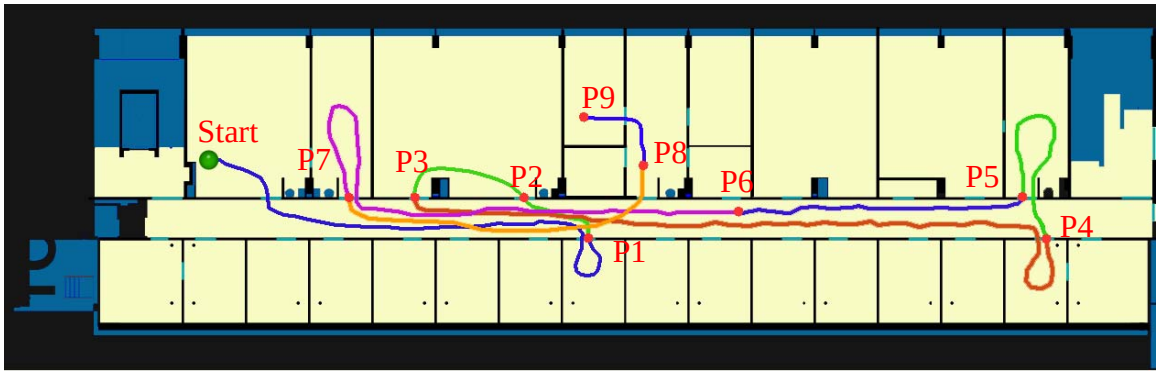


Figure 26 Trajectory walked by the target person, starting from green point and passing through all middle benchmarks to reach last point P9.

These measured benchmark points are compared with the estimated positions to determine the accuracy of the localization system. The step number acts as a logical key that matches a measured position with an estimated position.

The developed localization model is a randomized algorithm, and both the number of particles and results are random variables. Therefore, the observation log was processed 30 times by each movement model to assess their accuracy. The PDR system is based on a deterministic algorithm, so the log was processed one time with the dead reckoning system.

It was assumed that this problem was a local localization problem and the same configurations as simulation tests were applied on each movement model.

4.5.1 Naive Movement Model

The naive movement model was employed to estimate the target's position given the collected data from the real walk. Two normal distributions, one with μ=0 and σ=25 and the other with μ=0 and σ=8 were used for customizing the step events for each particle. The Euclidean distances between the nine measured positions and the corresponding estimated positions were taken as the drift. Figure 27 illustrates the estimated position at step 39 that is corresponding to the first benchmark (P1 in Figure 26). The red line indicates the calculated trajectory by the PDR system.

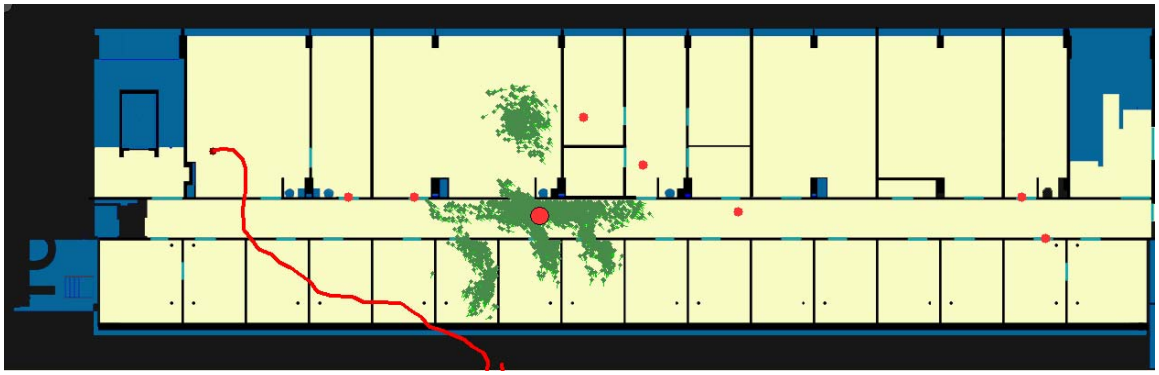


Figure 27 Snapshot of result of the localization system enhanced by the naive movement model when processing data collected from a real walk. The red line shows the trajectory determined by PDR and the big red circle shows the position estimated by the localization system. Small red circles are benchmarks.

The observation log was processed 30 times. Figure 28 plots the empirical cumulative distribution of the Euclidean distances between corresponding measured benchmark positions and estimated positions by the localization model enhanced by the naive movement model.

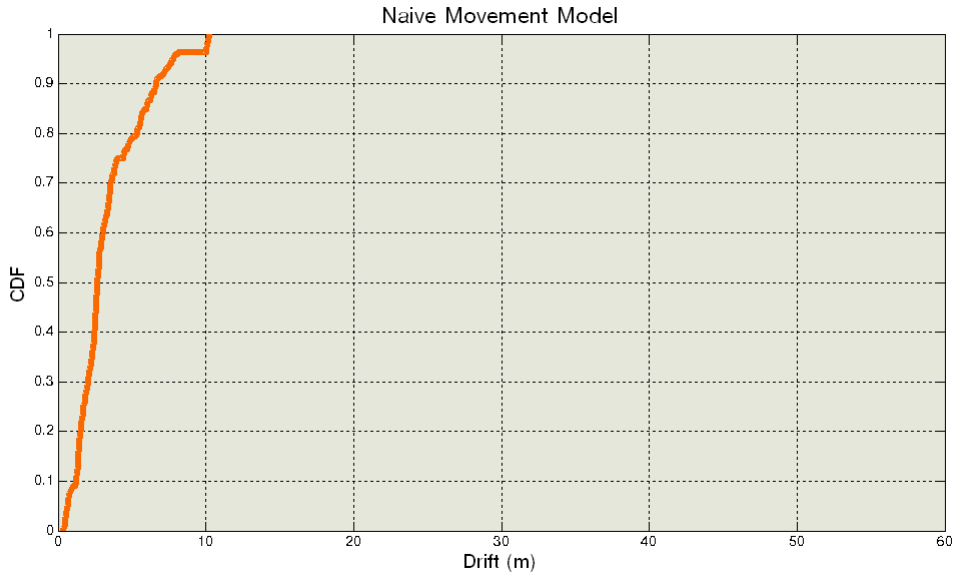


Figure 28 Euclidean distance (drift) between benchmark points and estimated positions by the localization model (enhanced by the naive movement model).

The localization model enhanced by the naive movement model was able to estimate the position of the target within 2.67 m 50% of the time, within 4.4 m 75% of the time and within 7.82 m 95% of time. Drift of estimation at each benchmark point is illustrated in Figure 29 by a box plot. The highest amount of drift was at points 8 and 9.

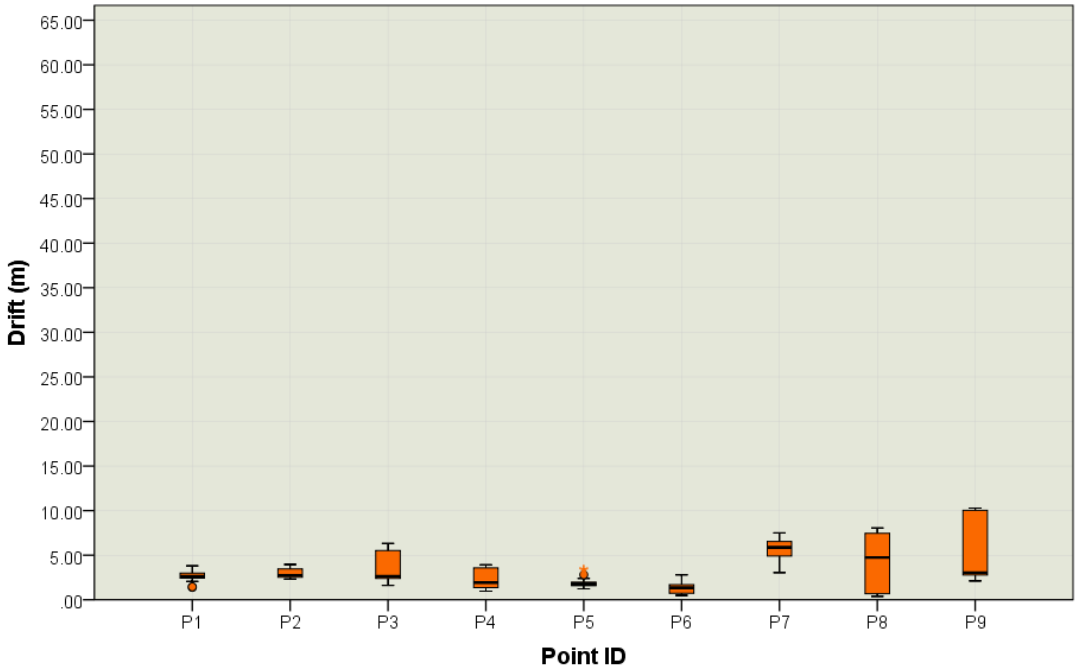


Figure 29 Boxplot shows Euclidean distance (drift) between benchmark points and estimated positions by localization model (enhanced by the naive movement model).

4.5.2 Detective Movement Model

The detective movement model was employed to estimate the target's position in nine benchmarks given the data collected from the real walk. The detective movement model examines the potential state space of each particle given the pure step event data and subsequently adds two random numbers drawn from two Gaussian distributions to the values of the step event. The parameters of normal distributions are variable and depend on how satisfactory the particle's state would be after applying the pure step event data. Hence, a more satisfactory pure step event for the current particle leads to narrower normal distributions. Similar to the simulation part, in this test normal distributions with μ=0 and σ=[*20,30*] and with μ=0 and σ=[*5,11*] were used for adding randomness to the step length and changes in heading, respectively. For more details, see Section 2.4.1.

As described before, the error in the localization system was measured by calculating Euclidean distances between the measured and estimated positions. Figure 30 illustrates the estimated position at step 119 that corresponds to the fourth benchmark (P4 in Figure 26). The red line indicates the estimated trajectory by the PDR system, which went beyond the bounds of the figure.

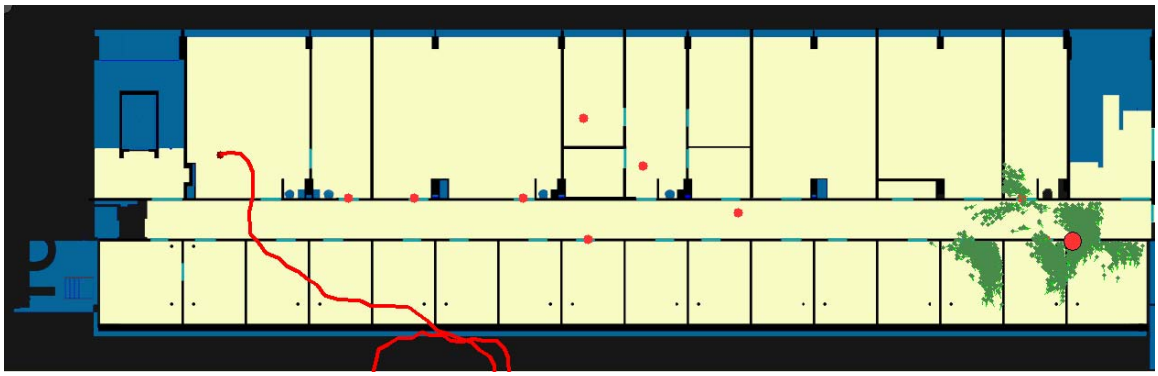


Figure 30 Snapshot of the result of the localization system enhanced by the detective movement model during processing data collected from a real walk. The red line shows the trajectory determined by PDR and the big red circle shows the position estimated by the localization system. Small red circles are benchmarks.

The collected data from the real walk was processed 30 times with the localization model using the detective movement model. Figure 31 plots the empirical cumulative distribution of the Euclidean distances between corresponding measured benchmark positions and estimated positions by the localization model enhanced by the detective movement model.

The localization model with the detective movement model was able to estimate the position of the target within 2.84 m 50% of the time, within 4.89 m 75% of the time and within 10.04 m 95% of time. The box plot illustrated in Figure 32 indicates the drift of estimation at each benchmark point. The highest amount of drift was at points 8 and 9.

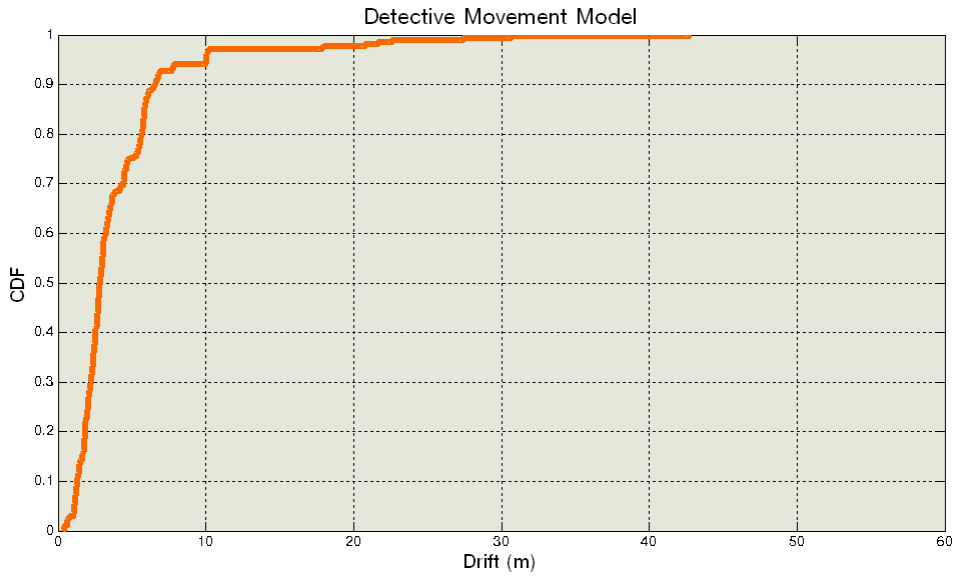


Figure 31 Euclidean distance (drift) between benchmark points and estimated positions by the localization model (enhanced by the detective movement model).

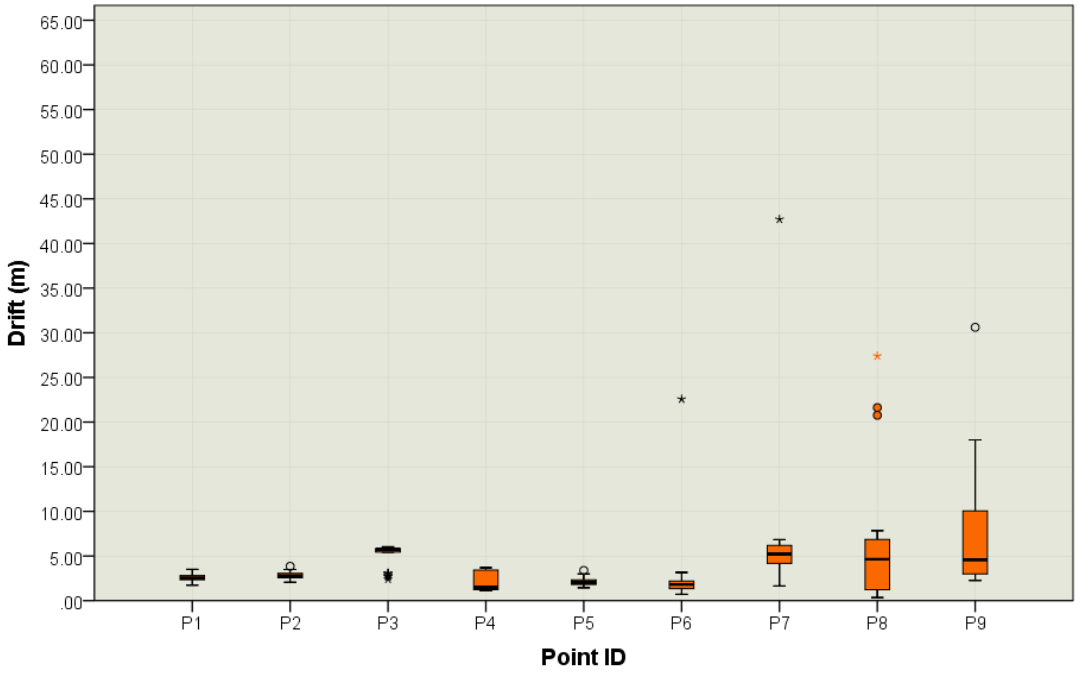


Figure 32 Boxplot shows Euclidean distance (drift) between benchmark points and estimated positions by the localization model (enhanced by the detective movement model).

4.5.3 Modifier Movement Model

The modifier movement model attempts to update the state of a particle given the current step event such that the particle's chance of survival is increased. As described in Section 2.4.1, the modifier movement model inspects a predefined range around the values of the step event for finding the best proper state of each particle after updating. In this test, a range of 50 cm for step length and 50 degrees for changes in direction were checked. In addition, it was assumed that the step length of human being must be in a range between 50 cm and 110 cm.

A small portion of randomness was added to the step event's values to avoid duplicate particles in posterior during resampling. The random values were drawn from a normal distribution with $\mu=0$ and $\sigma=12$ for step length, and from another normal distribution with $\mu=0$ and $\sigma=3$ for changes in direction.

Error in the localization system was measured at nine sample points by calculating the Euclidean distances between the measured and estimated positions. Figure 33 shows the estimated target position at step 242 that corresponds to the end of the walk (P9 in Figure 26). The red line indicates the estimated trajectory by the PDR system, which went beyond the bounds of the figure after step 55.

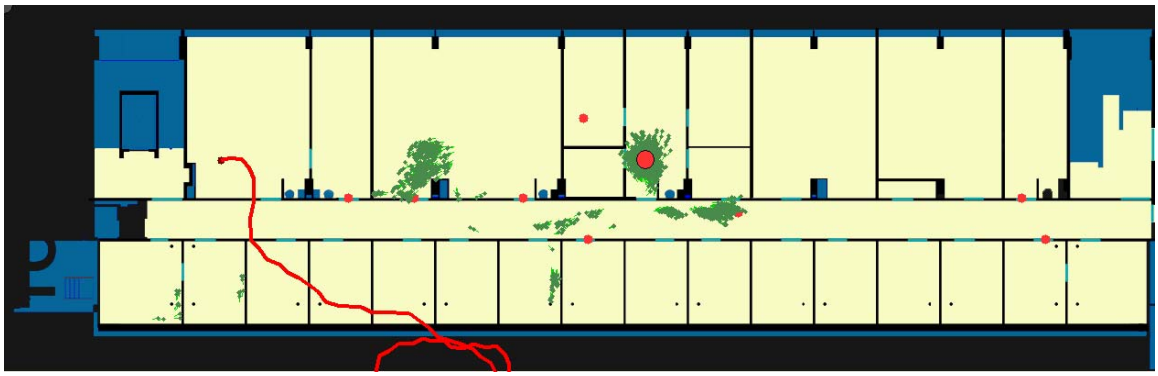


Figure 33 Snapshot of the result of the localization system enhanced by the modifier movement model during processing data collected from real walk. The red line shows the trajectory determined by PDR and the big red circle shows the position estimated by the localization system. Small red circles are benchmarks.

The localization model with the modifier movement model processed the collected data from the real walk 30 times. Figure 34 plots the empirical cumulative distribution of the Euclidean distances between corresponding measured benchmark positions and estimated positions by the localization model enhanced by the modifier movement model.

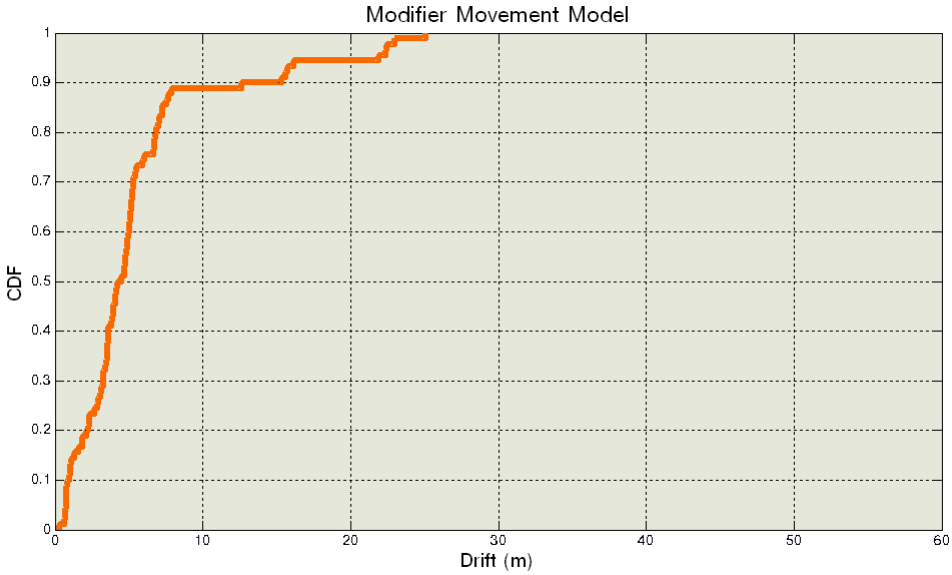


Figure 34 Euclidean distance (drift) between benchmark points and estimated positions by the localization model (enhanced by the modifier movement model).

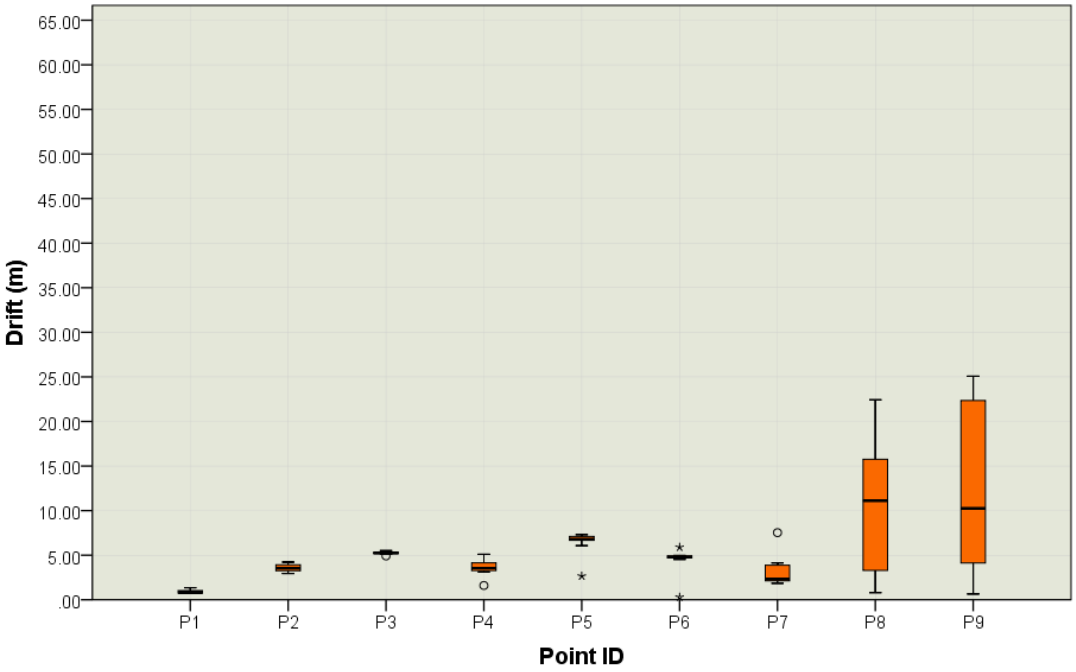


Figure 35 Boxplot shows Euclidean distance (drift) between benchmark points and estimated positions by the localization model (enhanced by the modifier movement model).

The localization model with the modifier movement model was able to estimate the position of a target within 4.37 m 50% of the time, within 6.22 m 75% of the time and within 22.1 m 95% of the time. The box plot illustrated in Figure 35 indicates the drift of estimation at each benchmark point. The highest amount of drift was at points 8 and 9.

4.5.4 Pedestrian Dead Reckoning System

The Pedestrian Dead Reckoning (PDR) system processed step events in parallel to all movement models in order to assess their accuracy. The PDR system is a deterministic algorithm that uses pure values of step events for updating the current state of target, given a known starting point. The red line in Figure 36 illustrates estimated trajectory using PDR at step 32 during using naive movement model. The error of PDR increases over time, as the estimation error accumulates with each step. Hence, the estimated trajectory shown in Figure 33 went out of image's bounds after step 55.

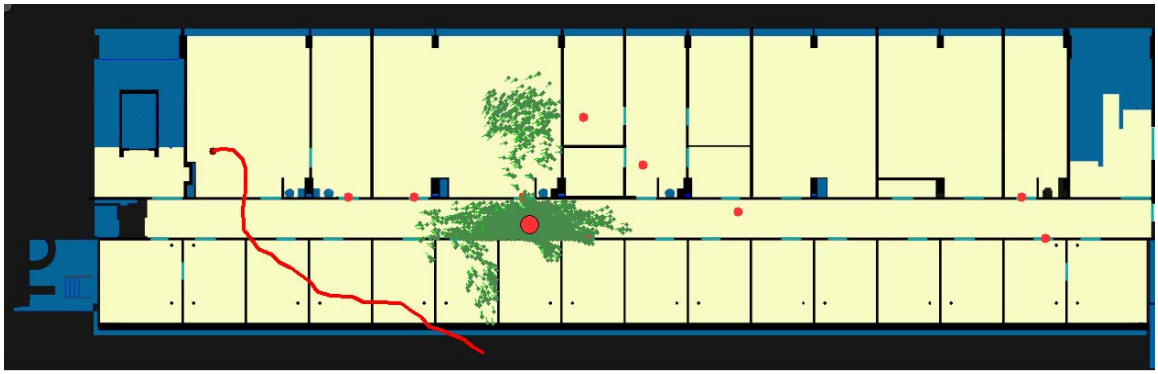


Figure 36 Snapshot the results of the localization system enhanced by one of the movement models during processing collected data from the real walk. The red line shows the trajectory determined by PDR and the big red circle shows the position of target estimated by the localization system.

Figure 37 shows the empirical cumulative distribution of the Euclidean distances between corresponding measured benchmark positions and positions estimated by PDR.

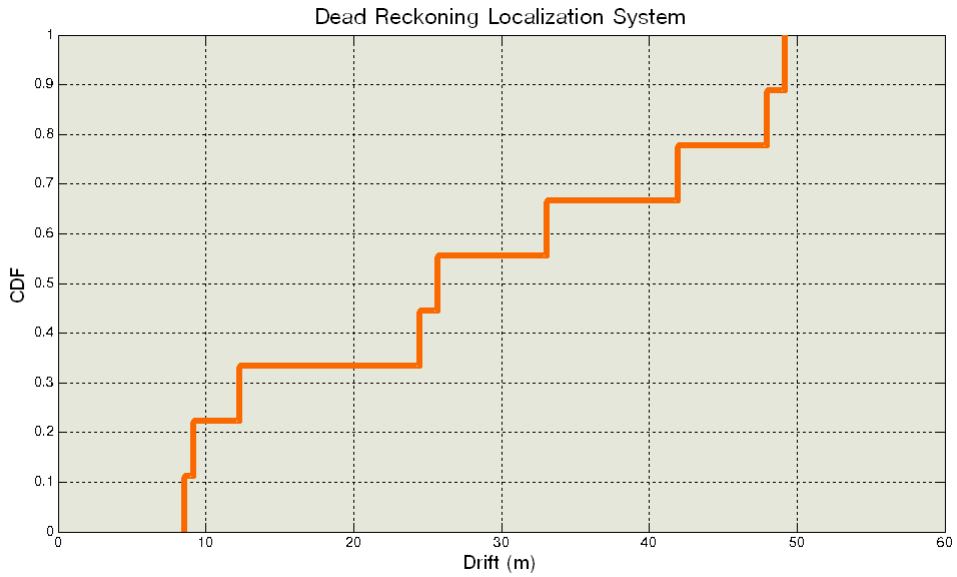


Figure 37 Euclidean distance (drift) between benchmark points and determined positions by PDR.

Drift of PDR 50% of time was within 25.67 m and 75% of time was within 44.97 m. Figure 38 illustrates the estimation error in PDR at each benchmark point. As expected, the error is increasing over time.

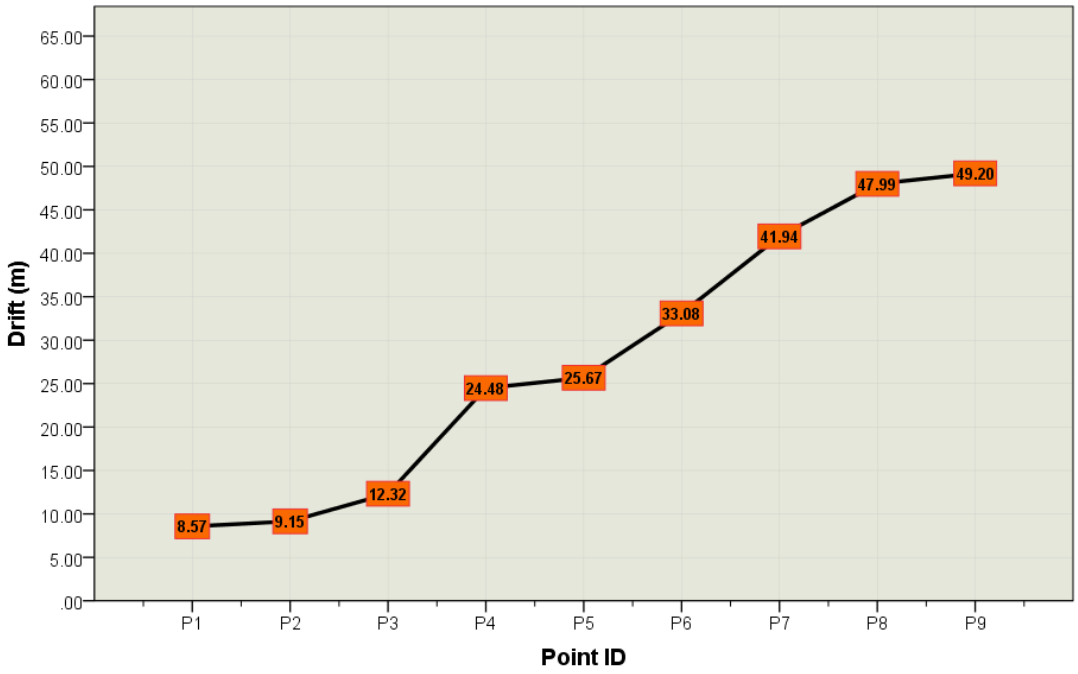


Figure 38 Euclidean distance (drift) between the benchmark points and positions estimated by PDR.

5 Discussion and Conclusion

5.1 Discussion

This work aimed to design a sensor fusion engine combining accelerometer and compass sensor data with environmental constraints with respect to a movement model of a pedestrian indoors to decrease uncertainty of a target's position during indoor localization. A step event is created from pure sensor data that represents two-dimensional movement of target by step length and change in heading. The MCL based model is used for estimating a target's position given the step events and a floor plan of the environment. The following discussion interprets the results and discusses problems that were faced.

5.1.1 Movement Detection

5.1.1.1 Step Detection

The proposed step detection formula (2.3.1) performed well independent of the height of users. Steps were successfully detected 97.2% of the time. However, the phone must be securely fixed on the user's upper body because it may detect any small motion. Although this may affect the usability of system, all dead reckoning and inertial navigation systems suffer from the same problem. In all cases, the sensor's structure should be fixed on the moving body.

When compared to other available step detection approaches, the proposed formula works better across a range of situations and, in particular, a variety of users. However, several thresholds are used in the formula, which make it complicated. In tests, most failures in detecting steps were in turns and walking backwards. For example, if the target takes a walk in a circular route with small radius, then some of steps may not be detected.

5.1.1.2 Step Length Estimation

For estimating the step's length (Section 2.3.2), a formula proposed in [Chen et al., 2009] was used. The main advantage of this formula is that it depends on only one unknown

variable (the user's individual factor) as well as the maximum and minimum acceleration of the step. Estimating a user's individual factor is easy during a learning phase. In addition, the natural human tendency to reduce velocity when reaching an object is already accounted for by the inclusion of the maximum and minimum acceleration. In pretests, (Section 4.2.2) more than 94% of the traveled distance was estimated by the formula. However, the total traveled distance was compared to the sum of the estimated step lengths, which does not measure the accuracy of a particular step length. Figure 39 plots a histogram of all estimated steps during the real walk discussed in Section 4.5. This histogram demonstrates that the estimated step lengths are quite similar. As a result it may be possible to avoid user's individual factor estimation in training phase.

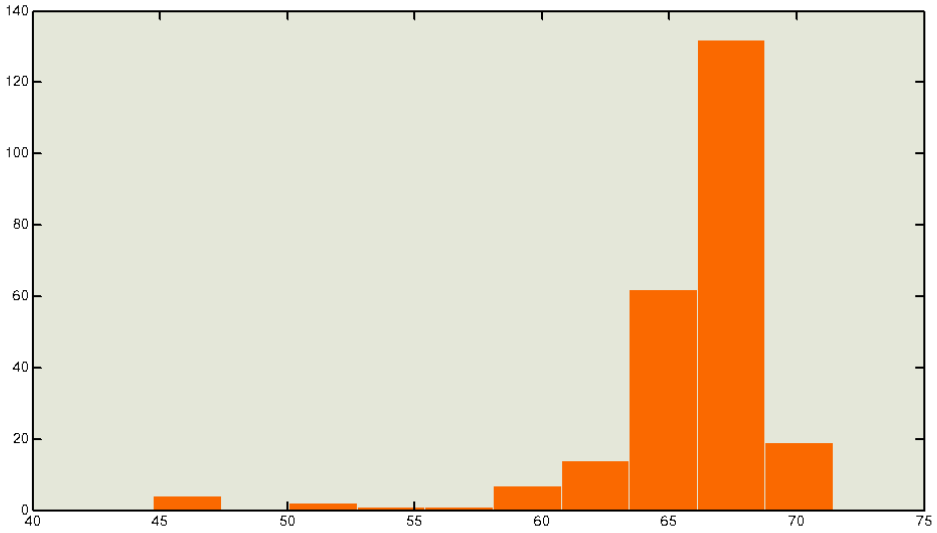


Figure 39 Histogram of estimated step lengths during the real walk sample.

5.1.1.3 Direction Estimation

Correct heading estimation is the most crucial part of movement detection. In PDR systems, most of the error comes from wrong heading estimation [Woodman, 2010]. This measurement, like all others, suffers from bias error. Therefore, using *changes in heading* instead of *current heading* was essential in neglecting the bias error.

As described in Section 2.3.3, observations of the magnetic field sensor are used in orientation estimation. However, magnetized structural steel inside of buildings may

cause interference in orientation estimation. Therefore, there is a strong possibility that the proposed system does not work in some buildings.

Fixing the smartphone on the abdomen can be a source of small tilts (~10 degrees) in each step event. In each gait, one leg is in the stance phase and the other is in swing phase. When the phone is secured on the abdomen and the orientation value of the step is recorded during minimum acceleration, a small tilt in the direction of the stance leg always occurs. For example, if the user walks in a straight line and, in the current step, the right leg is swinging, a small tilt to the left hand side is detected as a change in heading even though the user is walking straight.

5.1.2 Localization model

Results for all movement models were significantly more accurate than PDR. As expected, the error of PDR increased with time. Drift of the implemented localization system has dependencies other than time. All criteria used in the measurement model evaluate the state of particles with respect to environmental constraints. Hence, the accuracy depends on the traveled trajectory and the quantity of environmental constraints around the target. In the following, each part of the localization model is assessed separately.

5.1.2.1 Particle Filter

Estimating the target's position by using the particle filter has several advantages. First, the target's position cannot be estimated by parametric estimators since the shape of the distribution strongly depends on obstacles surrounding the target at each point. The particle filter, however, is a nonparametric method and does not suffer this problem. Second, particle filtering requires fewer prerequisites than the PDR system. With the particle filer, knowledge of the exact starting point is not necessary. In addition, deviation between north according to the floor plan and compass north is not important since changes in heading apply to the absolute heading of particles and initial headings are assigned randomly. Third, integrating measurements of a new sensor is quite straightforward. Measurements that describe the current state of the user can be added to the measurement model and influence the particles' importance factor. For example, estimations of a cell-based localization system can be integrated into the measurement

model. Simply, the importance factor of a particle is set to zero if the particle is not inside of any of cells that the cell-based localization estimates the target may be in.

The amount of required computational resources of the particle filter depends to the dimension of state vector and the number of particles. By using KLD-Sampling, the number of required particles was varied in each iteration depending on the distribution of the new set of particles. Figure 40 plots the resources consumed in the real walk test for three different movement models, assuming each particle is a resource. It can be seen that the naive and the detective movement models have used nearly equal number of particles, however due to the algorithms the detective movement model needs more computation per particle. The modifier movement model not only used significantly more computational resources than the other two models, but also it requires more computation per particle.

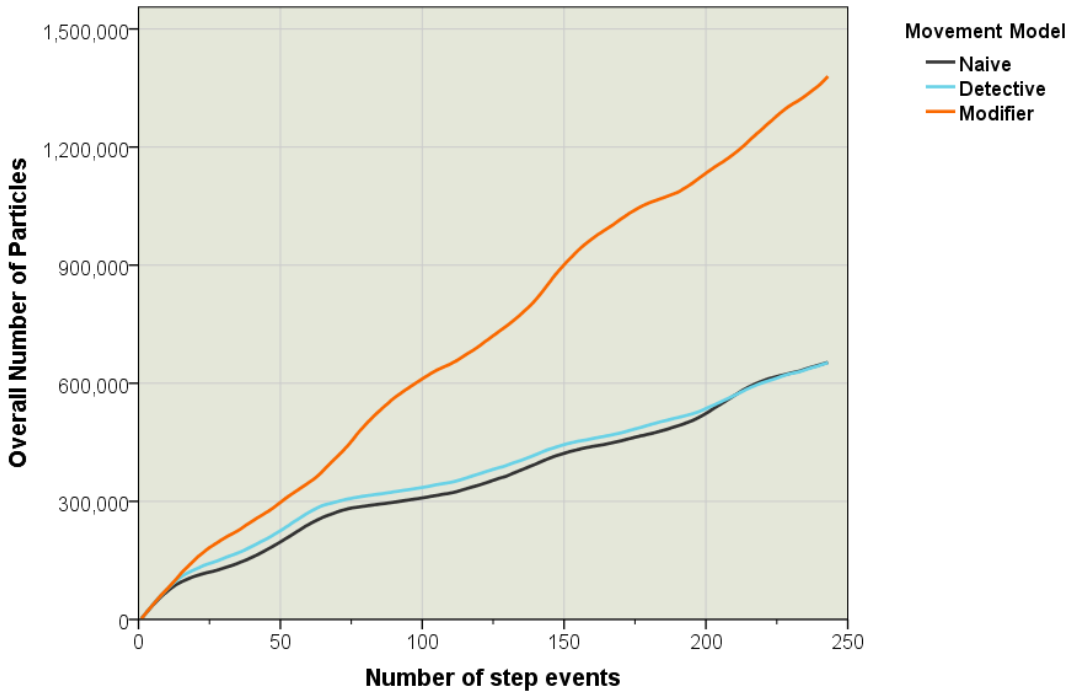


Figure 40 Overall number of particles used by each localization model enhanced with different movement model.

5.1.2.2 Measurement Model

Any further measurement that describes current state of target should be integrated with localization system in the measurement model. When the state of a particle is supported by current available measurements, a high importance factor is assigned to the particle. Accuracy of estimation is strongly depends on this module. In the implemented localization model, the map of environmental constraints acts as a virtual sensor and is used in the measurement model via three criteria (see Section 2.4.3). However, this was the only source of data. Therefore, the accuracy of the implemented localization model depends strongly on the uniqueness of the trajectory and the structure of the environment. If the trajectory is not unique, several particle clouds may be created and each one may represent the correct position of the target.

As described before, the cell-based localization system can be integrated with implemented localization system by slight changes in the measurement model.

5.1.2.3 Movement Models

Three different movement models were designed (see Section 2.4.1) and implemented. The results of these models were presented within this thesis. All three use different internal parameters that are assigned empirically using the sample walk presented in results section.

The naive movement model presents the best results for both real and synthetic data. The implementation of this model is easier and requires fewer computational resources than the other movement models. However, more sample data should be collected from different walks taken by different people in a variety of environments to fully examine the models. With such tests, the overall result of detective movement model may be better than the naive movement model since the internal parameters of naive movement model are static.

The detective movement model produces better results than the modifier model, but worse results than the naive movement model. The number of particles generated by localization system was increased in comparison to the naive movement model. This is because the particles are distributed widely during the resampling phase if their predicted weight is too small. In addition, the detective movement model needs more

computational resources than the naive movement model but less than the modifier movement model.

The modifier movement model produced a less accurate result than the other movement models. Modifying the step events to increase a particle's survival chance consequently decreases the chance of other particles during the resampling phase while modifying the step events not only changes the distribution of particles, but also changes their normalized weight. The weighted distribution of all particles always represents the best estimate of the position of target.

5.1.2.4 Convergence Detection

The convergence detection algorithm is simple and does not need a predefined number of clusters. It assigns particles to the most suitable cluster or, when needed, creates a new cluster for them. The center point of the cluster with maximum number of members (weighted) is returned as an estimated position of target. However, instead of detecting one point as a current estimated position of target, it is better to provide a circle (or ellipse) that shows the area where target is likely to be together with a reliability factor. Currently, the color of the estimated position in plots shows the confidence of the estimate.

Although the problem is localization, it is better to avoid jumping of estimated positions in each step.

5.1.2.5 Map

The map of environment was employed as a virtual sensor in the measurement and movement models. Due to the results and discussion in previous sections, the measurement model is the best place for applying environmental constraints in the localization system. The map is a suitable virtual sensor since it is permanent, easily accessible and low-cost. However a map is not sufficient since there are many more areas that a map confirms as a potential target position than the areas rejected by the map (ratio of free areas to blocked areas is high). For instance, consider when target enters a large room. Even if a dense particle cloud estimates the current position of a target, the particles will disperse after entering to a large free area (see Figure 41). Thereafter, the

particles may accumulate if the target passes through an area that is bounded by environmental constraints like doors or corridors.

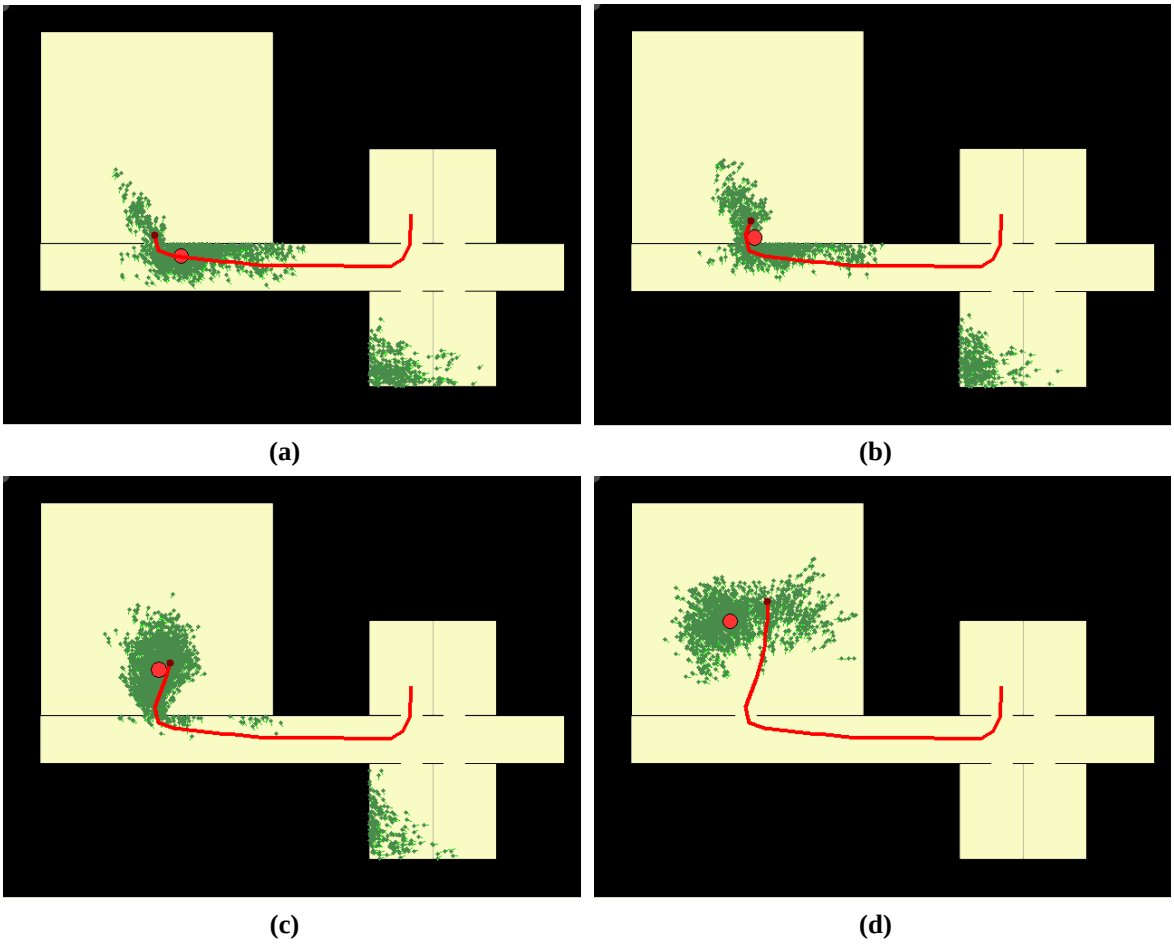


Figure 41 Particles will disperse after entering to a large free area. State of environment in step 20 (a), 21 (b), 23 (c) and 27 (d) are shown.

Having a map of the environment with a high level of detail increases the accuracy of estimation. In contrast, Symmetry of an environment and trajectory as well as absence of environmental constraints (e.g. a large room) may strongly reduce the accuracy of system. For example, the estimated positions of a target by all movement models at benchmark points 1, 4 and 5 were more accurate than the others since the trajectory before reaching them was relatively unique. In contrast, the estimates for points 8 and 9 were the worst. This is because the part of trajectory from point 7 to point 8 and then to 9 is neither long enough to become unique nor short enough to lead to small estimation errors.

5.1.2.6 System Evaluation

In the absence of a control system, the manually measured benchmarks were considered as ground truth. The idea of using benchmark points is quite practical, although much different than having a continuous control system. Based on the results of sample walk, the position of benchmark points may strongly influence calculated drift of system. Trajectory and benchmark points can be organized in a way that accuracy of system increases. However in the above sample, points were distributed both near to and far from constraints, and trajectory contains unique and common as well as short and long routes.

5.2 Conclusion

In this work, an indoor localization system was designed and implemented. The localization model is based on MCL used in robotic localization. Two modules of the localization model are enhanced with an indoor map. The first module was the measurement model, which employs the indoor map as a virtual sensor to evaluate feasibility of the updated state of a particle. The second module was the movement model. Three different movement models were developed: the naive, detective and modifier models. The detective and modifier movement models utilize the map of the environment for customizing the step event in a way that the particle's chance of survival increases after the updating process. In addition, a movement detection module was designed and developed for creating step events from pure accelerometer and magnetic field sensor data. This was divided in two parts. The first part was an Android application for detecting steps and recording data into a log file. The second was a module developed in MATLAB for providing step events from the log file. This architecture is needed for evaluating the localization model. The model is based on randomized algorithms and, therefore, each recorded data file should be processed several times. A simulator component was also developed, which makes examining the localization model with synthetic data possible. The localization model and simulator are also implemented in MATLAB.

Data recorded from a real walk of 170 m was processed by both the PDR system and the localization model enhanced by each movement model. Results show that the localization model was significantly more accurate than PDR. Drift of PDR 50% of time was within

25.67 m and 75% of time was within 44.97 m. The estimation error of target position by the localization model with the naive movement model was less than 2.67 m 50% of time and less than 7.82 m 95% of time.

Estimation accuracy of localization model depends on the employed movement model. The naive movement model, which does not employ map of environment, produced the most accurate results. The modifier movement model, which modifies the step event for each particle to increase its survival chance, produced the worst result. The reason for this is that manipulating the state of a particle that has less weight will increase the overall weights of particles and, consequently, reduces the chance that other particles with high importance factor will be resampled. Therefore, the environmental constraints should be employed in the measurement model as a virtual sensor to reach high estimation accuracy.

The estimation error of designed localization model does not rely the traveled distance, but rather depends on the uniqueness of trajectory and to what extent the position of the target is surrounded by obstacles on the map. The reason is that the only measurement that both describes the current state of target and is used for evaluating the state of particles is the map of the environment.

In addition, a scoring system for recognizing the pattern of human gait from acceleration data was introduced. In tests, the proposed system successfully detected more than 97.2% of steps. The step length estimator was tested on a straight 52 m walk and produced a result with 5.315% approximation error.

All in all, the estimation accuracy of the designed localization model is significantly higher than the PDR system due to using a map of the environment as a virtual sensor in the measurement model of the particle filter.

6 Outlook

6.1 Outlook

Pedestrian indoor localization contains many aspects. In this work, the essential components for a complete running system were designed and implemented. However, this thesis was limited by time. In the following, ideas for further work are overviewed.

Movement detection

Accuracy of the localization system can be improved by increasing the accuracy of computed step events. As mentioned before, the error of PDR system mostly comes from heading estimation. Nowadays, a gyroscope sensor is included in most of state-of-the-art smartphone configurations. Combining the gyroscope and magnetic field sensors data in a Kalman filter will increase the accuracy of heading estimation. This combination is a sample of complementary redundancy while the orientation computed from accelerometer and magnetic field sensors data has long term stability, although in the short term it may suffer from magnetic disturbances inside a building. On the other hand, the short-term accuracy of a gyroscope is high and its error increases with time (see [Ladetto and Merminod, 2004] or [Luinge et al., 1999] for more details).

Map

The map of environment is the only source of data that is used for evaluating particles states. Using other formats, like CityGML, provides three-dimensional maps, which can be useful for three-dimensional positioning. Changes in altitude (e.g. stairs) can be detected easily. This makes a trajectory unique since there are not normally many places inside a building where the target's altitude can change. In addition, the more details presented in the map, the more accurate the estimation of localization model. Information about constraints can be also useful. For example, how movable a constraint is. A chair is moveable, but tables and walls are less moveable. Knowledge regarding the strength of a wall may also be useful. In case of disaster (e.g. earthquake), the probability that the wall has fallen down can be computed.

A user's objectives and access privileges can also be modeled on the map. For example, if a student is walking along corridor it is less probable that he/she enters one of offices when a classroom is near; it is most probable he/she intends to go for a lecture. In another case, points of interest that have are visited more often are more likely to be visited by the current target. In a mall, for instance, the shops with more famous brands are a more likely destination of the target, or during lunch time cafeterias and restaurants are the most probable places which may be visited.

Other Source of Information

The designed localization system can be easily combined with a cell-based localization system. Assuming that the outcome of a cell-base localization system is a list of cells and there is a probability that target may be inside each of them, the outcome can be integrated in measurement model for evaluating current position of particles. Moreover, sensors like Bluetooth, which may provide data with arbitrary delay, can be also involved in a cell-based localization system. If the state vector of particles containing a short-term history about particle's previous positions, then incoming measurements of the cell-based localization system can be an integrated event with delay. Moreover, the cell-based localization can greatly increase the performance of the system in global localization while particles are only distributed in specified cells.

Localization Model

The localization model can be improved by adding a tracking mode. With this module, particles are spread across an open area when target passes through a door and enters to a large room, although they may have converged on entrance. This is because of the normal distributions that are used in movement model and the absence of environmental constraints. It may be possible to make the distributions narrower when particles are converged and the target's position is likely estimated.

Error of Localization Model

As described earlier, the error of in the localization model is not similar to the error of PDR. The error in the localization model mostly depends on the uniqueness of the trajectory and proximity of the target to obstacles on the map. More test scenarios with different participants are needed to learn more about this type of error. In addition, it may

be possible to avoid a training phase for the user's individual factor by using the detective movement model.

Simulator

Providing a more complicated error model for manipulating step events could be a first step for improving the simulator. Instead of just adding Gaussian noise, a combination of several normal distributions may be used for creating step events' noise. It could be interesting to see how the localization model reacts to more complicated measurement noise. In addition, creating a similar trajectory to the real observations and comparing the result of simulation with real data initiator could bring more information about efficiency of simulator.

Evaluation

Due to the fact that the reported accuracy of the indoor localization system strongly depends on the position of benchmarks, walked trajectory and the structure of the building, creating a testbed for assessing an indoor localization system is needed. With such a testbed, the accuracy of available systems can be compared easily.

7 Literature

3DCityDB 2012
Official homepage of 3DCityDB. http://www.3dcitydb.net.

Alvarez et al. 2006
D. Alvarez, R.C. Gonzalez, A. Lopez, J.C. Alvarez: "Comparison of Step Length Estimators from Weareable Accelerometer Devices", *28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 5964-5967, 2006.

Android 2012
SensorManager Class Documentation in Official Android Developer Website. http://developer.android.com/reference/android/hardware/SensorManager.html.

Chen et. al 2009
W. Chen, Z. Fu, R. Chen, Y. Chen, O. Andrei, T. Kroger, and J. Wang: "An integrated GPS and multi-sensor pedestrian positioning system for 3D urban navigation", *Proc. Urban Remote Sensing Event*, Shanghai, China, 2009.

Chen et al. 2010
W. Chen, R. Chen, Y. Kuusniemi, H. Wang: "An Effective Pedestrian Dead Reckoning Algorithm Using a Unified Heading Error Model", *Proceedings of IEEE/ION PLANS 2010*, Indian Wells, CA, 2010.

Fox 2003
D. Fox: "Adapting the sample size in particle filters through KLD-sampling", *International Journal of Robotics Research*, 22:985–1003, 2003.

Fox et al. 2003
D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello: "Bayesian filtering for location estimation", *IEEE Pervasive Computing*, 2(3):24–33, 2003.

Gröger et al., 2012
G. Gröger, T. H. Kolbe, C. Nagel, K. Häfele: "OGC City Geography Markup Language (CityGML) Encoding Standard", Version: 2.0.0, *OpenGIS® Encoding Standard*, OGC Doc. No. 05-099r2, 2012.

Hofmann-Wellenhof et al. 2003
B. Hofmann-Wellenhof, K. Legat, M. Wieser: "Navigation. Principles of positioning and guidance", Wien ; London: Springer, 2003.

IAI 2009
International Alliance for Interoperability: "Industry Foundation Classes Release 2x4", http://buildingsmart-tech.org/ifc/IFC2x4/rc3/html/index.htm, 2009.

Khider et al., 2009
M. Khider, S. Kaiser, P. Robertson, M. Angermann: "Maps and Floor Plans Enhanced 3D Movement Model for Pedestrian Navigation", *The Institute of Navigation, Inc. ION GNSS 2009*, Georgia, USA, 2009.

Ladetto and Merminod 2002
Q. Ladetto, B. Merminod: "Digital Magnetic Compass and Gyroscope Integration for Pedestrian Navigation", *9th Saint Petersburg International Conference on Integrated Navigation Systems*, Russia, 2002.

Luinge 2002
H. J. Luinge: "Inertial sensing of human movement", PhD. thesis, University of Twente, Netherlands, 2002.

Luinge et al. 1999
H. J. Luinge, P.H. Veltink, C. T. M. Baten: "Estimating orientation with gyroscopes and accelerometers", *BMES/EMBS Conference*, 2:844, 1999.

Kwok et al. 2003
C. Kwok, D. Fox, M. Meila: "Adaptive real-time particle filters for robot localization", *IEEE International Conference on Robotics and Automation*, 2: 2836- 2841, 2003.

Takeda et al. 2009
R. Takeda, S. Tadano, M Todoh, M Morikawa, M Nakayasu, S. Yoshinari: "Gait analysis using gravitational acceleration measured by wearable sensors", *Journal of biomechanics*. 42(3):223-33, 2009.

Thrun et al., 2005
S. Thrun, W. Burgard and D. Fox: "Probabilistic Robotics", MIT Press, 2005.

Wendlandt et al., 2006
K. Wendlandt, M. Khider, M. Angermann and P. Robertson: "Continuous Location and Direction Estimation with Multiple Sensors Using Particle Filtering", *Multisensor Fusion and Integration for Intelligent Systems. IEEE Verlag,* Germany, 2006.

Woodman, 2010
O. J. Woodman: "Pedestrian Localization for Indoor Environments", Ph.D. Thesis. University of Cambridge, Computer Laboratory. U.K., 2010.

8 List of Abbreviations

CDF	Cumulative Distribution Function
CityGML	City Geography Markup Language
DR	Dead Reckoning
EDA	Event-driven architecture
IFC	Industry Foundation Classes
KLD-Sampling	Kullback-Leibler Distance Sampling
MCL	Monte Carlo Localization
OGC	Open Geospatial Consortium
PDR	Pedestrian Dead Reckoning
WFS	Web Feature Service

9 List of Figures

10 List of Tables