Prof. Dr.-Ing. B. Wagner

Institute of Systems Engineering

Real-Time Systems Group

University of Hannover

A Real-Time Implementation of a Probabilistic Localization Method for Mobile Robots

Master Thesis

Mohamed Khalaf-Allah

August 2004

First Examiner	:	Prof. Dr.-Ing. B. Wagner
Second Examiner	:	Prof. Dr.-Ing. K. Jobmann
Supervisor	:	Dipl.-Ing. Oliver Wulf

Prof. Dr.-Ing. B. Wagner

Institute of Systems Engineering

Real-Time Systems Group

University of Hannover

A Real-Time Implementation of a Probabilistic Localization Method for Mobile Robots

Master Thesis

Mohamed Khalaf-Allah

August 2004

First Examiner	:	Prof. Dr.-Ing. B. Wagner
Second Examiner	:	Prof. Dr.-Ing. K. Jobmann
Supervisor	:	Dipl.-Ing. Oliver Wulf

I hereby declare that I have achieved this work independently and that I have not used any other sources except the ones indicated in this work as references.

Hannover, August 17th 2004.

Mohamed Khalaf-Allah

To my beloved Egypt.

Abstract

Localization is a key problem for mobile robot autonomy. It is the problem of determining a robot's pose from noisy sensor data. MCL algorithms represent a robot's belief about its location by a set of weighted hypotheses (samples), which approximate the posterior distribution under a common Bayesian formulation of the localization problem. This work presents a self-localization system for mobile robots based on odometry data, 3D laser perception and a line-feature map of the environment. MCL is utilized to fuse data from the different sensors and the given map in order to generate an accurate location estimate. This localization system has been implemented on an experimental robot platform and tested in a real world indoor environment. Furthermore, intensive simulation studies have been conducted. Real world tests and simulation results confirm robustness and reliability of the developed localization system.

Keywords: mobile robot localization, Monte Carlo Localization, MCL, particle filter, position tracking, global localization, 3D laser scanner, odometry, line-feature map, sensor fusion, data fusion.

Contents

List of Figures

List of Tables

List of Symbols

$Bel\ (s_t)$	The robot's belief state at time t, also called posterior distribution
s_t	State at time t, i.e. (x, y, φ)
$d_{0...t}$	Sensor data delivered from time 0 up to time t
o_t	Observation data (e.g. laser scan) at time t
a_t	Action data, i.e. odometry at time t
m	World model, i.e. map
$p(x\| y)$	Probability of x given y
η	Normalization constant
n	Number of samples (particles)
w	Weight or importance factor of a sample
σ_{trans}	Standard deviation of translation
σ_{rot}	Standard deviation of rotation
$\sigma_{drift,\ trans}$	Standard deviation of translational drift
$\sigma_{drift,\ rot}$	Standard deviation of rotational drift
σ_{laser}	Standard deviation of laser range measurement

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
EKF	Extended Kalman Filter
GPS	Global Positioning System
KF	Kalman Filter
MC	Monte Carlo Method
MCL	Monte Carlo Localization
MHT	Multi-Hypothesis Tracking
ML	Markov Localization
PF	Particle Filter
SIR	Sampling/Importance Resampling
TOF	Time-Of-Flight

Chapter 1

Introduction

Mobile robot localization is defined as the estimation of the robot's position and orientation in its environment using a map of this environment, and a history of performed actions and perception data. It is a key problem in mobile robotics and has been referred to as "the most fundamental problem to providing a mobile robot with autonomous capabilities" [Cox1991].

1.1 Types of Mobile Robot Localization Problems

The mobile robot localization problem has three main types [Bor1996]. The first and the simplest is *position tracking* in which the initial pose of the robot is known, and the problem is to compensate incremental errors in the robot's odometry. Algorithms for position tracking often make restrictive assumptions on the size of the error and the shape of the robot's uncertainty, required by a range of existing localization algorithms. The second and more challenging is the *global localization problem*, where the initial pose of the robot is unknown. Consequently the robot has to determine its pose from scratch. This problem is more difficult because the robot has to handle multiple, distinct hypotheses, since the error in its estimate cannot be assumed to be small. Finally, the third and most difficult is the *kidnapped robot problem*. Here a well-localized robot is teleported to some other position without registering any odometry data or control commands. This problem is often used to test a robot's ability to

recover from severe localization failures. All these problems become harder in dynamic environments, where robots operate in the proximity of, e.g., people who corrupt the robot's perception measurements [Fox1999].

1.2 Goal of the thesis

The goal of this thesis was to implement the Monte Carlo Localization (MCL), which is a probabilistic localization algorithm for mobile robots. This implementation was to achieve real-time position tracking and global localization within indoor environments for the experimental mobile robot platform built at the Institute of Systems Engineering, Real-Time Systems Group, see figure 1.1. The data inputs of the localization system are a line feature map of the environment, odometry readings and 2D laser range scans. These 2D laser range scans are provided using a 3D laser scanner in the form of virtual 2D scans in order to avoid the effects of objects that are normally not included in the map, e.g. furniture. Furthermore, with the help of 3D scans, the corrupted information of range data in dynamic environments will be reduced to a minimum. The software realization followed in two phases, off-line implementation and simulation using MATLAB, and implementation in the C language into the existing real-time robot software running under the real-time operating system Linux RTAI (Real-Time Application Interface).

1.3 A Brief History of Monte Carlo Methods

The Monte Carlo method (MC) is a powerful modeling tool for the analysis of complex systems. It may be generally defined as a methodology for obtaining estimates of the solution of mathematical problems by means of random numbers. These random numbers are generated using a roulette-like machine of the kind utilized in the gambling casinos of the Monte Carlo Principality: hence, the name of the method.

An early example that conceives the basic idea of what we now call a MC method is known as *Buffon's needle* [Dör1965], first stated by the French naturalist Georges Louis Leclerc Comte de Buffon (1707-88) who considered a flat surface with a grid of parallel lines spaced by a distance D and computed the probability P that a needle of length L (where $L < D$)

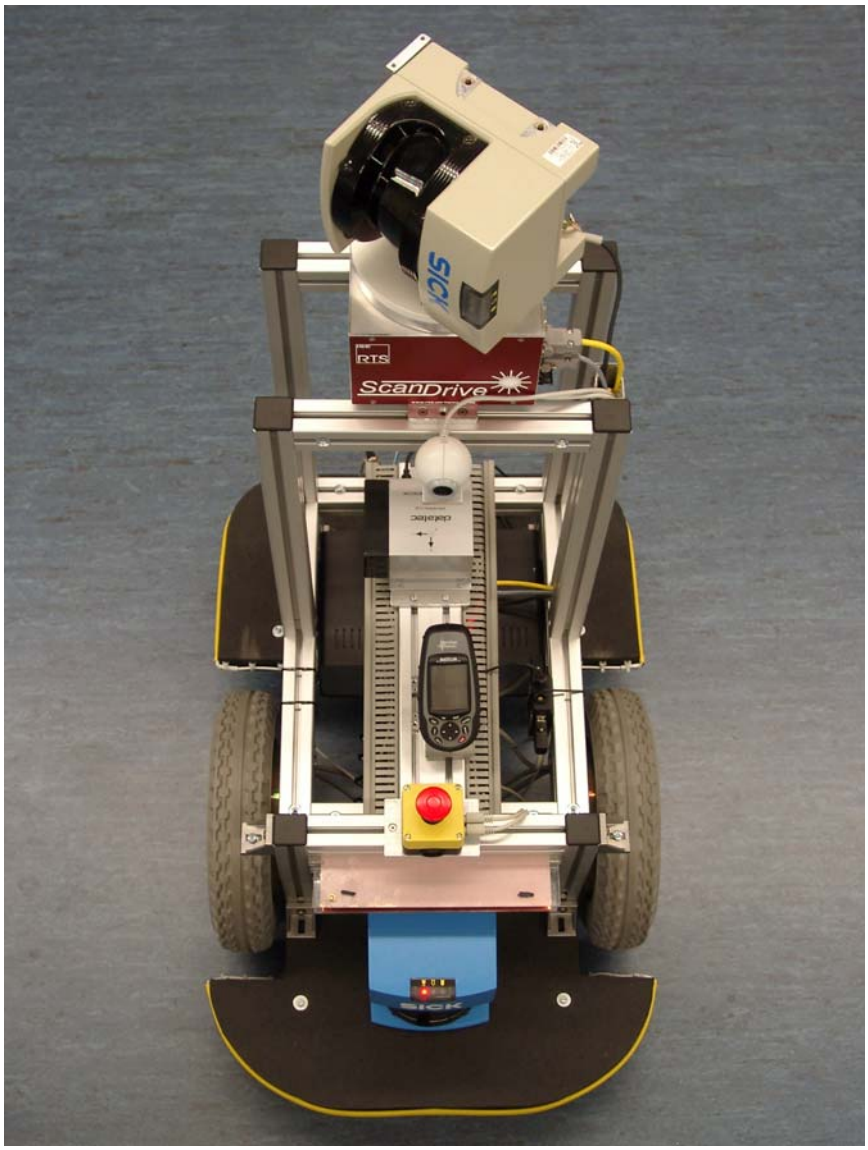


Figure 1.1: The experimental mobile robot platform ERIKA.

randomly thrown onto the grid would intersect one of these lines [Com1777]. Under ideal conditions, he obtained $P = \frac{2L}{\pi D}$. Later, Laplace observed that Buffon's experiment represents a tool for computing π by just throwing a needle on a floor [Lap1886]. Thus, if we let p_N be the proportion of intersects in N throws, an estimate of π can be obtained as

$$\hat{\pi} = \lim_{N \to \infty} \frac{2L}{p_N D} \tag{1.1}$$

which will converge to π as N increases to infinity.

In later years, Kelvin used MC techniques to solve integrals within the kinetic gas theory [Kel1901]; he drew numbered cards from a drawer and wondered about possible correlations between the drawn numbers due to an imperfect mixing of the cards.
Gosset (Student) was among many scientists who tackled probability problems with methods of random sampling. He used a MC method to estimate the correlation coefficient of his famous *t*-distribution [Stu1908].
A revival of the systematic use of MC method appeared in the early days of electronic computing and accompanied the development of the world's first programmable super computer, MANIAC (Mathematical Analyzer, Numerical Integrator and Computer), at Los Alamos in the course of the Manhattan Project during WWII [Liu2001]. Ulam, von Neumann, Metropolis, Fermi, and others invented a statistical sampling-based method for solving numerical problems concerning random neutron diffusion in fissile materials in atomic bomb designs and for estimating eigenvalues of the Schrödinger equation. It was Metropolis who coined the name "Monte Carlo", which played an essential role in popularizing the method.
In the early 1950s, statistical physicists introduced a Markov-chain-based dynamic MC method for the simulation of simple fluids. In the 1980s, statisticians and computer scientists developed Monte-Carlo-based methods for a wide variety of tasks such as combinatorial optimizations, nonparametric statistical inference, likelihood computations with missing observations, statistical genetics analysis, and Bayesian modeling and computations. In the 1990s, MC method began to play an important role in computational biology. Now, the list of application areas of MC methods includes biology, chemistry, computer science, economics and finance engineering, material science, physics, statistics, and many others.
The aforementioned concise chronology of MC methods reveals that simulating random processes to help estimate certain quantities of interest is now an essential part of scientific computing.

1.4 Thesis Outline

This thesis is organized as follows. Chapter 2 provides a summary of Bayes filtering based techniques used for mobile robot localization. An introduction to the world of Bayes filtering that founds the mathematical basis for MCL is presented in chapter 3. Chapter 4 discusses issues of data inputs to the developed localization system. The software system description is given in chapter 5. Chapter 6 provides statistical results of the conducted experiments that

studied the performance of the developed 3D laser sensor based MCL. The work is concluded in chapter 7 accompanied with recommendations for future developments.

Chapter 2

State of the Art

Bayes filter [Dou2001] is an abstract concept that only provides a probabilistic framework for recursive state estimation. There are two different implementations of Bayes filter in the context of mobile robot localization. They differ mainly in the way they represent belief distributions over the state space, see figure 2.1. The first implementation represents *continuous* belief distributions, where the second represents *discrete* distributions.

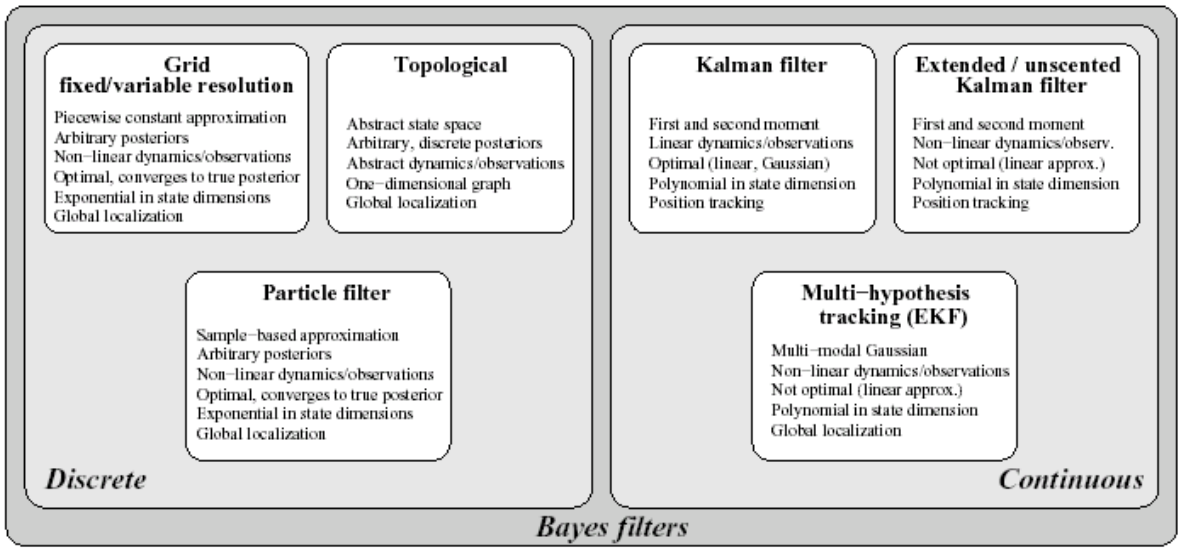


Figure 2.1: An overview of different implementations of Bayes filter for mobile robot localization [Fox2003].

2.1 Continuous Bayes filters

2.1.1 Kalman filters

Kalman filters [Kal1960] approximate beliefs by their first and second moments, i.e. mean and covariance. They are very efficient when the initial state uncertainty is a unimodal Gaussian and the perception model and system dynamics are linear in the state with Gaussian noise. The main drawback of Kalman filters is their incapability to represent multiple hypotheses due to the unimodal assumption. Thus, where Kalman filter solves the position tracking problem efficiently, it is unable to solve the global localization problem. Another disadvantage of Kalman filters is their restriction to linear and Gaussian models.

2.1.2 Extended and Unscented Kalman filter

If the system dynamics are non-linear in the state, non-linearities can be approximated by linearization at the current state, resulting in the so called extended Kalman filter. An enhancement to this approach is the unscented Kalman filter, in which samples taken from the Gaussian state are generated deterministically. A Gaussian approximation of the predicted samples coming out of the non-linear system dynamics follows immediately. The linear approximations allow tracking the position, but non-optimally. The global localization problem is still unsolved.

2.1.3 Multi-hypothesis tracking (MHT)

The belief is represented by multiple Gaussians. Each Gaussian is tracked by an extended Kalman filter. Therefore, MHT is able to represent multi-modal beliefs, and thus solves the global localization problem. Apart from uni-modality, MHT still shares the same assumptions of Kalman filters. Moreover, multi-hypothesis tracking requires sophisticated heuristics to solve the data association problem (i.e. which sensor data corresponds to which belief) and to determine when to add or delete hypotheses and which hypothesis is the most likely one.

2.2 Discrete Bayes filters

2.2.1 Topological Markov Localization

The environment is represented by symbolic and graphic Structures. The state space is a set of discrete and distinctive locations such as corners and crossings of corridors. Topological ML can represent multi-modal distributions over the discrete state space. Therefore it can solve the global localization problem. Moreover, topological ML may scale well towards high dimensional state spaces because the complexity of the topological structure does not directly depend on the dimension of the underlying state space. The main drawback, however, lies in the coarseness of the representation. Thus, only rough information about the location is provided by position estimation. Furthermore, adequate features might not be available in arbitrary environments, and only perception readings related to the environment model can be used.

2.2.2 Grid-based Markov Localization

Grid-based ML [Fox1999] depends on discrete, piecewise constant representations of the belief. As the topological ML, it can represent arbitrary distributions over the state space and is able to solve the global localization problem. The metric approximations with high robustness to sensor noise can provide highly accurate position estimates, which is a clear advantage over topological ML. The drawback of grid-based ML lies in its high computational burden, which is caused by keeping a three-dimensional position probability grid in memory that has to be updated every time new sensor information is received. Such a disadvantage, which affects the applicability for real-time localization, can be overcome by using efficient sensor models, selective update schemes, and adaptive, tree-based representations. However, since the complexity of these alternatives is directly proportional to the number of dimensions, there is no guarantee whether they can be successfully applied to higher-dimensional state spaces.

2.2.3 Particle Filters / Monte Carlo Localization (MCL)

In particle filters [Dou2001, Fox2001], or Monte Carlo Localization (MCL) [Fox1999-2, Thr2001, Fox2001], the belief is approximated by a set of samples (or particles) drawn from this belief. Each sample is a vector contains an x-y coordinate, an orientation and an associated weight called *importance factor*. This importance factor represents the probability of being at that location. A key advantage of MCL is its ability to represent arbitrary probability distributions, hence its ability to solve the global localization problem. Furthermore, MCL converges to the true posterior in non-Gaussian, non-linear dynamic systems. Compared to MHT, updating the state vector of one sample is computationally more efficient and simpler than updating a Gaussian function, allowing a large number of samples to be used to approximate the probability distribution. The main advantage of MCL over grid-based ML lies in its efficiency, since it concentrates resources in areas of state space with high likelihood. Therefore, MCL can be applied to high-dimensional state spaces if it focuses the posterior on small, lower-dimensional regions of the state space. The decision to use MCL in this project was made based on results of experimental comparisons testing a number of localization algorithms published in [Gut1998, Gut2002, Kri2003]. These comparisons showed the superiority of MCL over other localization techniques in efficiency, precision, and real-time requirements.

2.3 Mixed Techniques

The aforementioned implementations of Bayes filter differ in the way they represent and update the distribution of the belief. In dynamic Bayesian networks representations are made using independences in the structure of the state space to break the state into lower-dimensional sub-spaces. Each sub-space can be represented using the most suitable representation, e.g. continuous densities, or samples. Combinations of particle filters with Kalman filters provide efficient and robust techniques to higher dimensional state estimation. Such combinations are known as Rao-Blackwellised particle filters.
In the next chapter the derivation of the Bayes filter and more details on MCL will be provided.

Chapter 3

Theoretical Foundations

MCL is a recursive Bayes Filter that estimates the posterior belief distribution of a robot's pose (Cartesian position and orientation or heading direction) given a map of the environment and a history of sensor data. Bayes filters estimate the state of a dynamical system, i.e. partially observable Markov chain, using data delivered from sensor measurements. In the context of mobile robot localization, the dynamical system is the mobile robot and its environment, the state is the robot's pose relative to that environment, and sensor measurements, which may include laser and/or sonar range measurements, camera images, and odometry readings.

3.1 Bayes Filtering

Bayes Filters assume that the environment is *Markovian*, i.e. past and future data are conditionally independent if the current state is known. The key idea of Bayes filtering is to estimate a posterior probability density over the state space conditioned on the sensor data. This posterior distribution is called the *belief* and is denoted

$$Bel\ (s_t) = p\ (s_t \mid d_{0\ldots t},\ m) \tag{3.1}$$

where $Bel(s_t)$ is the robot's belief state at time t, s_t is the state at time t, $d_{0...t}$ denotes the data delivered from time 0 up to time t, and m is the model of the environment, i.e. a map. Two types of sensor data can be distinguished: *perceptual data*, i.e. data that characterizes the momentary situation such as laser range scans or camera images, and *odometry data*, i.e. data that delivers information about robot motion. The former is referred to as *observations* and the later as *actions*. Assuming that *observations* and *actions* data arrive in an alternating sequence expression (3.1) can be rewritten as

$$Bel(s_t) = p(s_t \mid o_t, a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0, m) \tag{3.2}$$

The desired posterior is estimated *recursively* by applying Bayes rule[1] and the theorem of total probability[2] [Bar2001] to expression (3.2), and exploiting the Markov assumption twice. Note that the *initial* belief characterizes the *initial* knowledge about the system state. If such knowledge is unavailable, it is initialised by a *uniform distribution* over the state space. This *uniform distribution* corresponds to the global localization problem, where the initial pose of the robot is unknown. To begin with applying Bayes rule, expression (3.2) can be transformed to

$$Bel(s_t) = \frac{p(o_t \mid s_t, a_{t-1}, \ldots, o_0, m)\, p(s_t \mid a_{t-1}, \ldots, o_0, m)}{p(o_t \mid a_{t-1}, \ldots, o_0, m)} \tag{3.3}$$

The denominator represents the probability of getting the observation o_t when the robot reached a location due to the action a_{t-1}. This probability is constant relative to s_t and is denoted as η, where $\eta = p(o_t \mid a_{t-1}, \ldots, o_0, m)^{-1}$ is the normalization constant. Therefore expression (3.3) is usually written as

$$Bel(s_t) = \eta\, p(o_t \mid s_t, a_{t-1}, \ldots, o_0, m)\, p(s_t \mid a_{t-1}, \ldots, o_0, m) \tag{3.4}$$

Applying *Markov assumption,* we find that

---

[1] $p(a \mid b, c) = \dfrac{p(c \mid a, b)\, p(a \mid b)}{p(c \mid b)}$

[2] $p(A) = \int p(A \mid B_i)\, p(B_i)\, dB_i$

$$p(o_t \mid s_t, a_{t-1}, \ldots, o_0, m) = p(o_t \mid s_t, m),$$

and hence expression (3.4) is simplified to

$$Bel(s_t) = \eta \; p(o_t \mid s_t, m)\, p(s_t \mid a_{t-1}, \ldots, o_0, m) \tag{3.5}$$

Employing the theorem of total probability, the right most term in expression (3.5) is expanded by integrating over the state at time *t-1* as follows

$$Bel(s_t) = \eta \; p(o_t \mid s_t, m) \int p(s_t \mid s_{t-1}, a_{t-1}, ..., o_0, m)\, p(s_{t-1} \mid a_{t-1}, ..., o_0, m) ds_{t-1} \tag{3.6}$$

Exploiting *Markov assumption* for the second time we get

$$p(s_t \mid s_{t-1}, a_{t-1}, ..., o_0, m) = p(s_t \mid s_{t-1}, a_{t-1}, m)$$

Thus, expression (3.6) can be simplified to

$$Bel(s_t) = \eta \; p(o_t \mid s_t, m) \int p(s_t \mid s_{t-1}, a_{t-1}, m)\, p(s_{t-1} \mid a_{t-1}, ..., o_0, m) ds_{t-1} \tag{3.7}$$

Noting that the second term in the integration is simply $Bel(s_{t-1})$, substituting it into expression (3.7) we obtain the following desired recursive equation

$$Bel(s_t) = \eta \; p(o_t \mid s_t, m) \int p(s_t \mid s_{t-1}, a_{t-1}, m)\, Bel(s_{t-1}) ds_{t-1} \tag{3.8}$$

In the context of mobile robot localization this equation is often referred to as *Markov localization,* which is the recursive update equation in Bayes filter. Combined with the initial belief, it defines a recursive estimator for the state of a partially observable system and acts as the basis for MCL algorithms. To implement equation (3.8), we need to specify two conditional, usually time-invariant densities. The first is $p(s_t \mid s_{t-1}, a_{t-1}, m)$, which is called *motion model*, and characterizes the effect of actions *a* on the robot's pose. The second is $p(o_t \mid s_t, m)$, which is called *perceptual model* or *sensor model*.

3.2 Probabilistic Models of Robot Motion and Perception

3.2.1 The Motion Model

To update the belief when the robot moves, the motion model $p(s_t \mid s_{t-1}, a_{t-1}, m)$ have to be specified. This model can be considered as a probabilistic generalization of the mobile robot's kinematics [Cox1990]. The conventional kinematic equations describe only the expected pose s_t that a noise-free robot would reach, if it starts at pose s_{t-1} to perform the action a_{t-1}. In practice, the robot motion is erroneous. To account for the uncertainty in the expected pose s_t, the motion model $p(s_t \mid s_{t-1}, a_{t-1}, m)$ describes a posterior density distribution over possible expected poses. Errors in translation and rotation are assumed to be normally distributed. They are modelled as zero-centered Gaussian distributions that are added to the translation and rotation components of the odometry readings. The variances of these are proportional to the length of the measured motion. Figure 3.1 shows two examples of the motion model. The greyly shaded areas describe the uncertainty of the odometry measurements: the darker a pose, the more probable it is. For the MCL algorithm, a closed-form motion model is not needed. Instead, a sampling model of $p(s_t \mid s_{t-1}, a_{t-1}, m)$ will do the work. The routine of this sampling model accepts s_{t-1} and a_{t-1} as inputs and generates random poses s_t distributed according to $p(s_t \mid s_{t-1}, a_{t-1}, m)$. Figure 3.2 demonstrates a sample motion model applied to a sequence of odometry measurements. It is clear that the sequence of particle sets approximate the distribution densities of a robot's poses that only measure odometry.

3.2.2 The Perception Model

The perception model $p(o_t \mid s_t, m)$ depends on the kind of sensor employed [Fox2001]. If o_t is raw camera images, the computation of $p(o_t \mid s_t, m)$ is related to the computer graphics problem in that the appearance of an image o_t at pose s_t has to be predicted. However, mobile robots are commonly equipped with range finders, such as sonar sensors or laser range finders. Using range finders makes computing $p(o_t \mid s_t, m)$ considerably simple. Such sensors measure the distance to nearby obstacles by using sound or structured laser light. Figure 3.3 demonstrates the perception model of a mobile robot for a planar 2D laser range finder.

Figure 3.3a shows an example of a laser range scan and a map of the environment. The perception model $p(o_t \mid s_t, m)$ is computed in two steps. First, computing the correct measurement to get in an ideal, noise-free environment. This is easily done for laser range finders using ray-tracing in a geometric map of the environment as shown in Figure 3.3a. Second, the desired density $p(o_t \mid s_t, m)$ is obtained from a mixture of three densities: a Gaussian that models the event of measuring the correct distance with a small added Gaussian noise, an exponential density that models random readings as often caused by people moving in the environment (left side of figure 3.3b), and a discrete large probability that models max-range measurements, which frequently occur when the detection of an object fails (right side of figure 3.3b).

3.3 Bayes Filtering at Work

To illustrate Bayes filtering and show how does equation (3.8) work in reality, let us take a look at figure 3.4. In this figure a one-dimensional scenario is demonstrated, in which a person carrying a camera (the sensor in use) is walking down a hallway. The camera cannot distinguish different doors. In figure 3.4a, the person's position at the beginning is unknown, hence, the uniform distribution over all possible locations. At the current location, the camera detects a door and sends a "door detected" signal. As shown in figure 3.4b, high probabilities are placed at positions next to doors in the hallway. Notice that positions away from doors still have non-zero probabilities due to the uncertainty inherent in sensing. The resulting belief distribution shows the ability of Bayes filter to handle multi-hypotheses in ambiguous situations. After some more steps to the right, figure 3.4c shows the effect of this motion on the belief distribution. The resulting belief is shifted to the right, following the motion direction, and smoothed due to the inherent uncertainty in the motion estimation. Figure 3.4d depicts the belief distribution after detecting another door. It can be noticed that ambiguity has been resolved and the location of the person estimated to be somewhere within a small proportion of the hallway. Figure 3.4e depicts the belief after the next motion of the person. Bayes filter has now a high confidence in the estimation of the person's location.

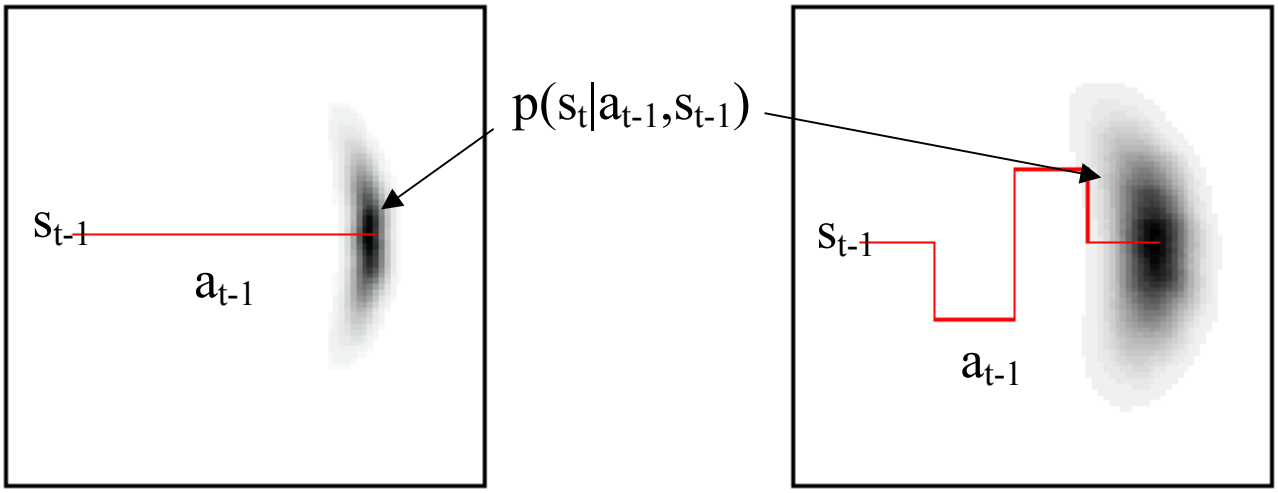


Figure 3.1: The density $p(s_t \mid s_{t-1}, a_{t-1}, m)$ after moving 40 meter (left) and 80 meter (right). The darker an area, the more likely it is [Thr2000].

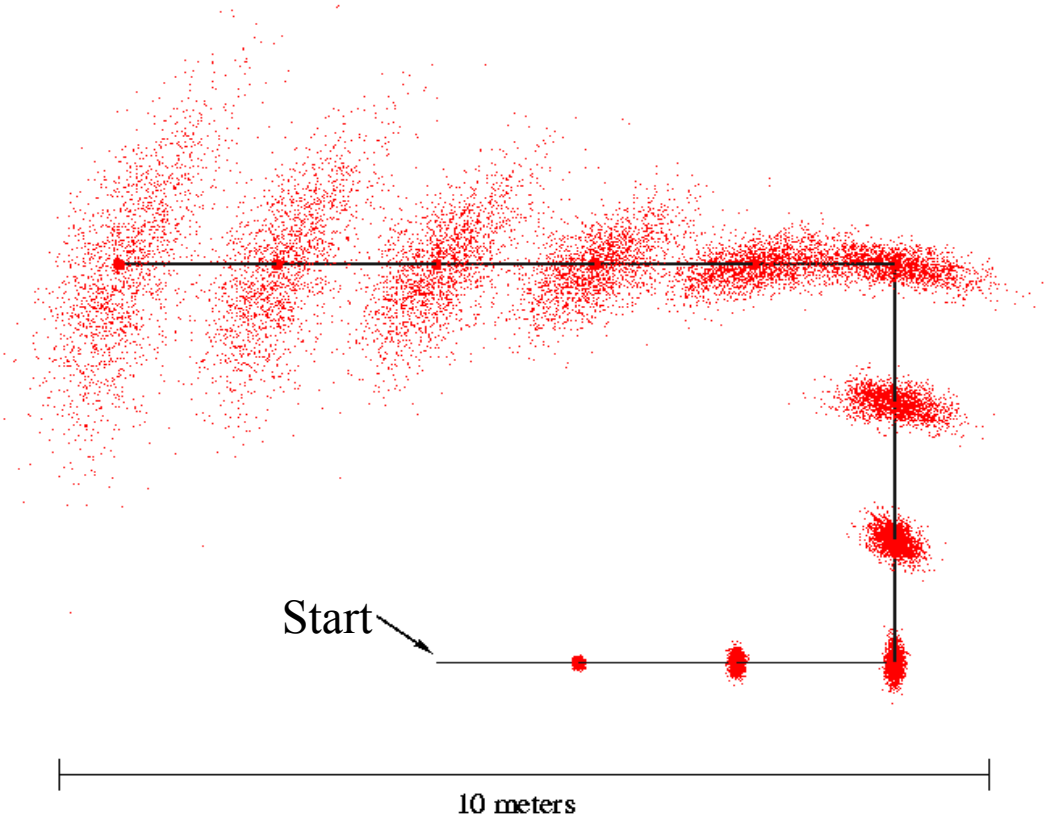
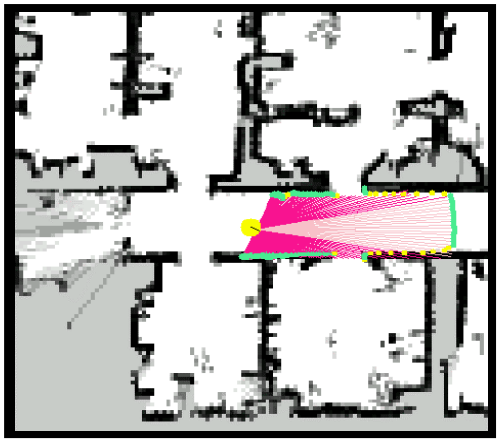
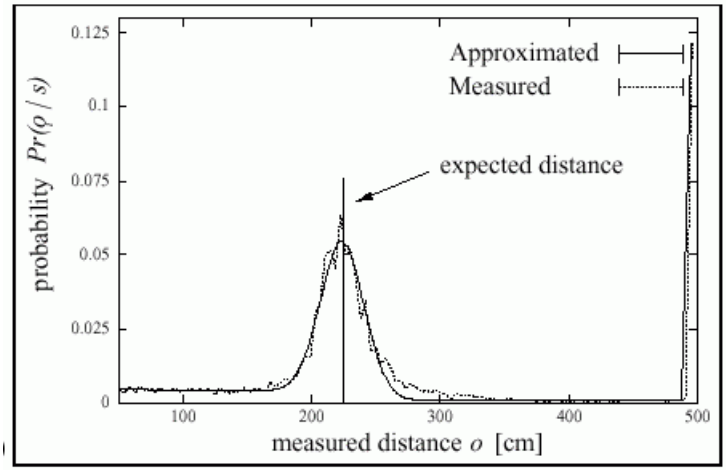


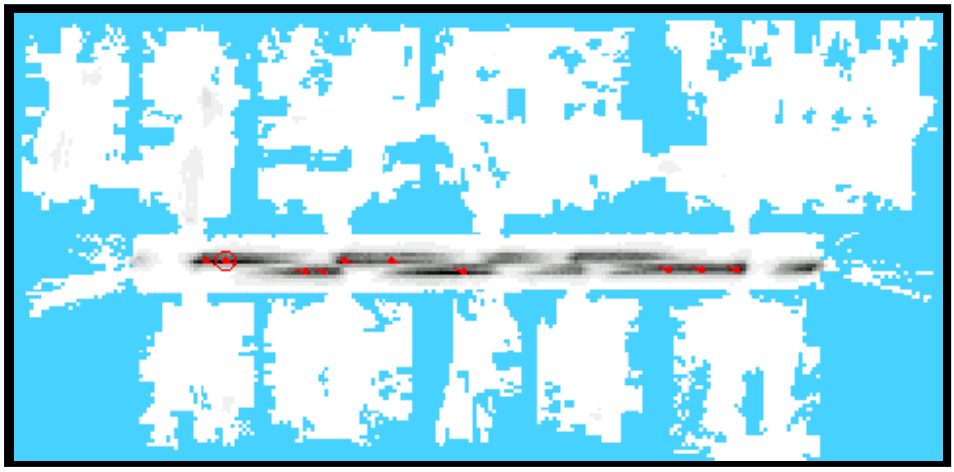
Figure 3.2: Sampling-based approximation of the position belief for a non-sensing robot. The solid line displays the actions, and the samples represent the robot's belief at different points in time [Fox1999-2].



(a) Laser scan and map



(b) Sensor model



(c) Probability distribution for different poses

Figure 3.3: (a) Laser scan range, projected into a map. (b) The density $p(o_t \mid s_t, m)$. (c) $p(o_t \mid s_t, m)$ for the scan shown in (a). Based on a single sensor scan, the robot assigns high likelihood for being somewhere in the main corridor [Fox2001].

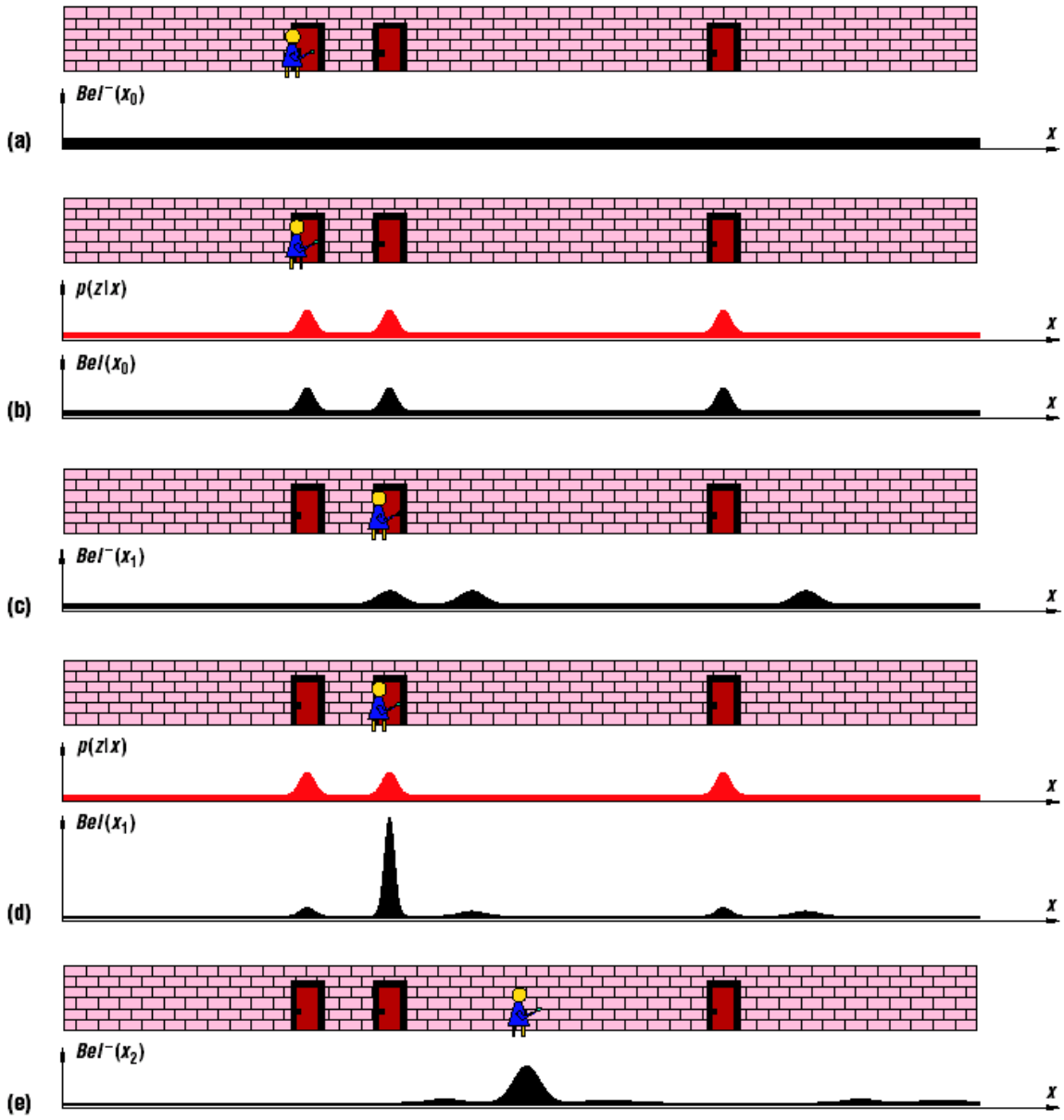


Figure 3.4: A one-dimensional illustration of Bayes filters. A person carries a door-sensing camera that cannot distinguish different doors. Each frame depicts the person's position in the hallway and the current belief *Bel* (x_t): (a) The person's location is unknown. (b) The sensor sends a "door found" signal. (c) The person moves. (d) The sensor observes another door. (e) The person moves again. Additionally, (b) and (d) depict the observation model $p(z|x)$, the probability of observing a door at different locations in the hallway [Fox2003-2].

3.4 The Monte Carlo Localization

In the case of mobile robot localization, the state space representation is *continuous*. Therefore, implementing equation (3.8) is indeed not a trivial problem, especially if efficiency is taken into account.

The basic idea of MCL is to represent the belief *Bel(s)* by a set of *n* weighted samples (or particles) distributed according to *Bel(s)*:

$$Bel(s) \approx \{s^{(i)}, w^{(i)}\}_{i=1,\ldots,n} \tag{3.9}$$

where each $s^{(i)}$ is a sample of the random variable *s* (the hypothesized pose of the robot). The parameters $w^{(i)}$ are non-negative numeric values and called *importance factors*, which sum up to 1, but it is worthy to mention that this is not strictly required in particle filtering. These importance factors determine the weight or the importance of each sample. The continuous belief *Bel(s)*, thus, is approximated by a discrete probability function defined by the sample set. The initial sample set represents the initial knowledge or belief *Bel* (s_0) about the state of the dynamical system. In position tracking, the initial belief is a set of samples drawn from a narrow Gaussian distribution centered on the start position of the robot, where each sample has the same uniform initial weight n^{-1}. In the global localization problem, the initial belief is represented by a set of samples distributed uniformly over the whole possible poses in the robot's working environment. This initial belief also is annotated by the uniform importance factor n^{-1}. The recursive update of the basic MCL is realized in three steps, computing equation (3.8) from right to left:

(1) Draw a random sample s_{t-1} from the current belief *Bel* (s_{t-1}), with a likelihood given by the importance factors of the belief *Bel* (s_{t-1}).
(2) For this sample s_{t-1}, predict a successor pose s_t, according to the motion model $p(s_t \mid s_{t-1}, a_{t-1}, m)$.
(3) Assign a preliminary (non-normalized) importance factor $p(o_t \mid s_t, m)$ to this sample and add it to the new sample set representing *Bel* (s_t).

Repeat step 1 through 3 *n* times. Finally, normalize the importance factors in the new sample set *Bel* (s_t) so that they sum up to 1. As already mentioned, normalization is not strictly

required. The sample set converges to the true posterior *Bel* (s_t) as *n* reaches infinity; with a convergence speed *O* ($1/n^{-2}$) [Thr2001]. Table 3.1 summarizes the basic MCL algorithm. There are various versions of the MCL algorithm; the version described in this thesis is referred to as Sampling/Importance Resampling (SIR) [Dou2001].

Inputs: $Bel(s_{t-1}) = \{s_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1,\ldots,n}$ representing belief *Bel* (s_{t-1}), odometry data a_{t-1}, observation o_t.

// Initialize desired belief and normalization constant
Bel (s_t) := 0, $\alpha = 0$

// Generate n samples representing *Bel* (s_t)
for *i*:=1,….,n **do**

// Resampling: Draw a sample from current belief according to its weight
Draw $s_{t-1}^{(i)}$

// Sampling: Predict next state using odometry data
Sample $s_t^{(i)}$ from $p(s_t \mid s_{t-1}, a_{t-1}, m)$

// Importance sampling: Compute importance factor
$w_t^{(i)} := p(o_t \mid s_t^{(i)}, m)$
$\alpha := \alpha + w_t^{(i)}$

// Insert sample into sample set
Bel (s_t) := *Bel* (s_t) $\cup \{ s_t^{(i)}, w_t^{(i)} \}$

// Normalize importance factors
for *i*:=1,….,n **do**
$w_t^{(i)} := w_t^{(i)} / \alpha$

return *Bel* (s_t)

Table 3.1: The basic MCL algorithm.

3.5 MCL at Work

Recalling the one-dimensional hallway example described in section 3.3 and illustrated in figure 3.4, let us apply MCL to this example. In figure 3.5a we see a uniformly distributed sample set representing the initial unknown position of the walking person. The equal heights of all bars indicate that each sample is assigned the same importance factor *w(x)*. The detection of a door by the camera is incorporated by adjusting and normalizing the importance factor of each sample. The resulting sample set is shown in figure 3.5b. The importance factors of the samples are now proportional to the observation probability distribution. When the person moves, MCL randomly draws samples from the current sample set with probability given by the importance factors. Then MCL uses the motion model to predict a successor new sample for each sample in the current set, leading to the new sample set shown in figure 3.5c. Most samples are now concentrated on three locations. This concentration is achieved through sampling proportional to the weights. In figure 3.5d, the camera detects a second door. The sample set shown is obtained by computing the importance factors according to the sensor model. The next prediction step causes most of the samples to concentrate at the true position of the person (see figure 3.5e), yielding a consistent result with that obtained in figure 3.4 using Bayes filtering.

3.6 The Course of a Mobile Robot Global Localization

In this section, a description of a real-world example of the global localization problem for a mobile robot is provided and illustrated. The working space of the robot is a typical indoor office environment. This robot is equipped with a laser range finder and a map of the environment. At start, the robot is totally uncertain about its location; hence the samples are spread uniformly over the whole state space as illustrated (after projection into 2D) in figure 3.6 upper. After approximately 1 meter of motion, most of the samples are grouped at two probable locations (figure 3.6 middle). After another 2 meters of motion, the robot has resolved the ambiguity and localized itself with high confidence, as indicated by the concentration of samples on the true location (see figure 3.6 lower).

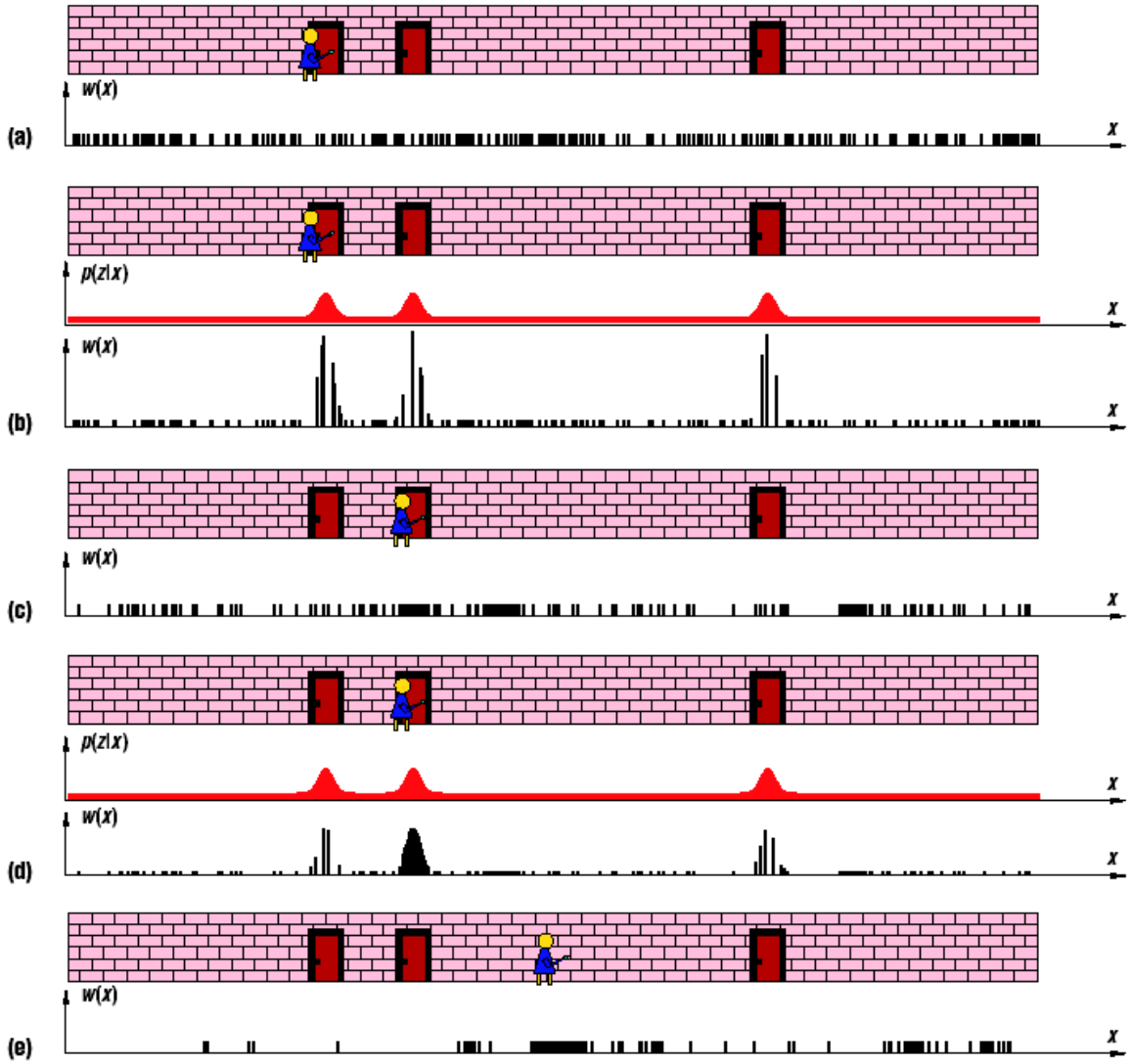


Figure 3.5: Applying particle filters to location estimation. The black bars depict the particles representing the belief *Bel* (x_t). (a) A uniformly distributed sample set presents the person's initially unknown position. (b) A sensor detecting the left door. The sample set is obtained from weighing the importance factors in proportion to the likelihood of the measurement. (c) An implementation of the prediction step. The samples were drawn from the previous set with probability proportional to the importance factors. (d) A sensor detecting a second door. (e) The sample set obtained after another prediction step [Fox2003-2].

It is worthy noting that the models of MCL, i.e., motion model, sensor model and the map, are extremely crude and simplistic, since probability models carry their own notion of uncertainty. This makes probabilistic algorithms relatively easy to code, unlike traditional algorithms that require deterministic models.

In this chapter, the necessary background for MCL has been thoroughly discussed. Chapter 4 will provide details of the MCL models for mobile robot localization.

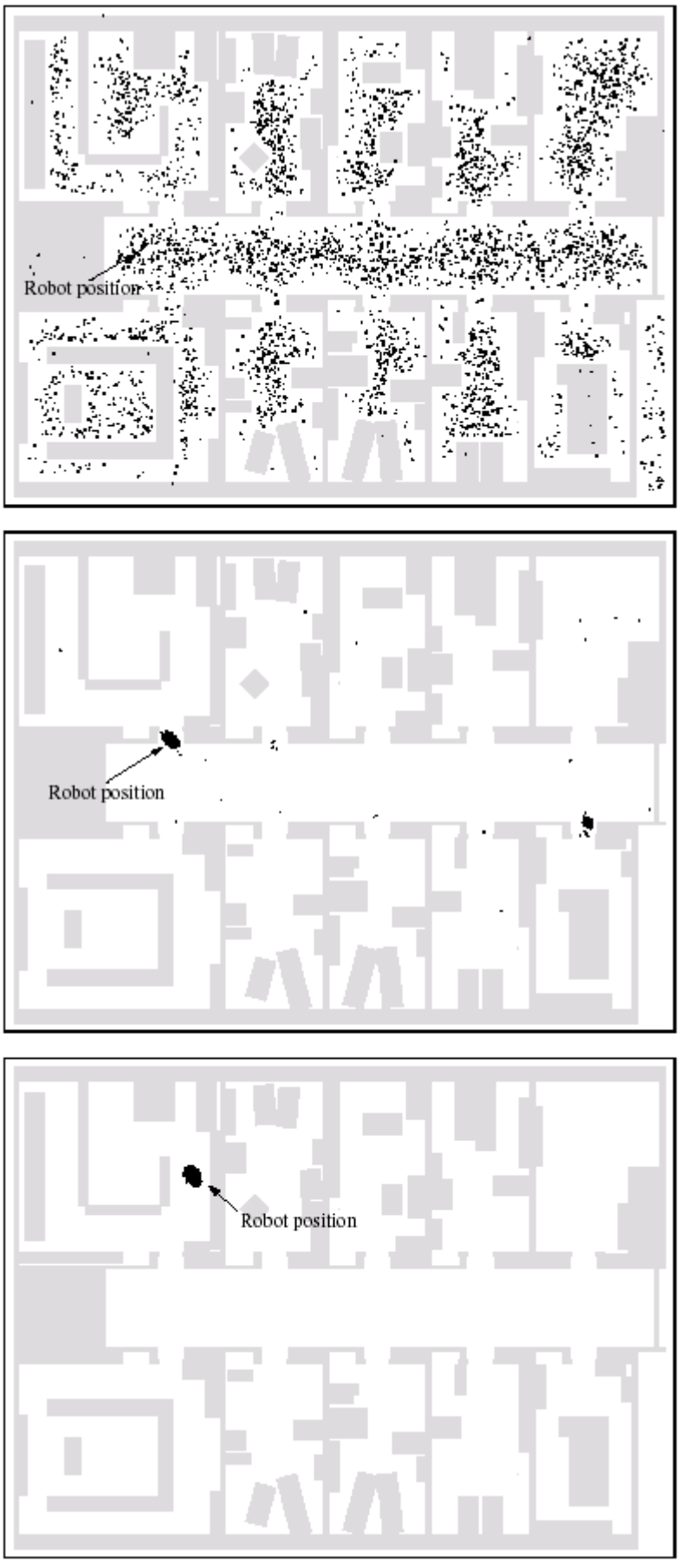


Figure 3.6: Global localization of a mobile robot using MCL with 10,000 samples [Fox2001].

Chapter 4

Sensors and World Representation

This chapter is about data input to the localization system of the experimental robot "ERIKA". The pose estimation problem can be divided into two phases, *prediction* and *update* as shown in figure 4.1. If the robot can determine its pose with enough accuracy directly and unambiguously at any time from odometry data, the update phase can be omitted. In real world situations this is usually not the case.

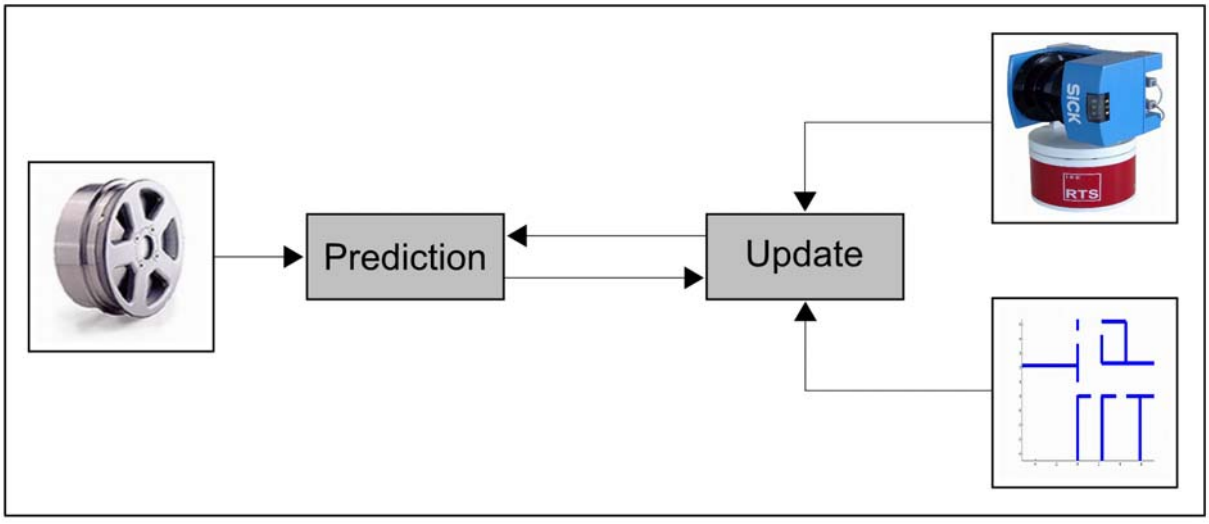


Figure 4.1: The pose estimation problem is usually divided into prediction and update phases. Here prediction utilizes odometry data, while update uses information from a 3D laser scanner accompanied with a map of the environment.

4.1 Sensors

Mobile robots are provided with different sensors for world perception in order to navigate autonomously. The most frequently used sensors include odometry, sonar, laser range finders and video cameras. In this work, odometry and a 3D laser scanner are used to deliver information about the environment. Examples of such sensor data are shown in figure 4.2.

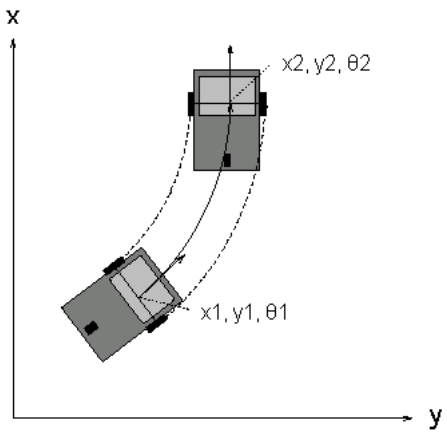
4.1.1 Odometry

The robotic platform used in this work is wheeled. Encoders are used to measure the rotation of the wheel axes. These data measurements are commonly known as *odometry*. Odometry provide information about the change in pose of the robot, and is extracted using sensors, which count the number of rotations for the wheel and the steer axes.

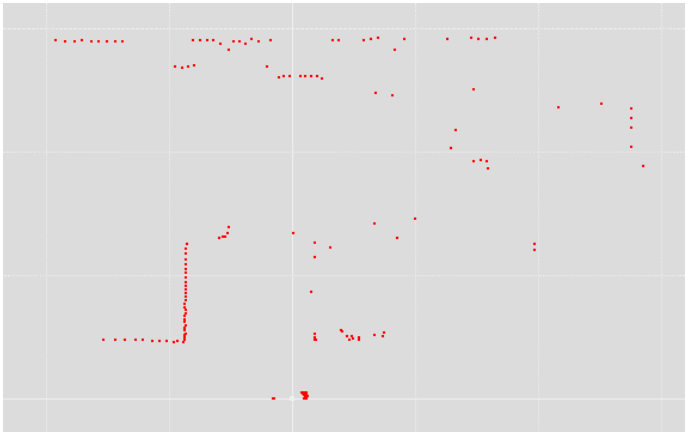
An odometry model is nothing but an approximation of the true kinematics of the robot system. Indoor robots usually have better odometric accuracy than outdoor ones, because of the non-planar surfaces, which face outdoor robots.

Sources of error that contribute to the accumulation of uncertainty during motion include wheel slippage, difference in the diameters of the wheels and anomalies of the floor. For modeling purposes, the uncertainty in odometry could be categorized into translational error and rotational error. Modeling these errors can be statistically achieved by drawing random variables from four Gaussians with σ_{trans}, σ_{rot}, $\sigma_{drift,\ trans}$ and $\sigma_{drift,\ rot}$ standard deviations. The first and second Gaussians model the error accumulated during pure translations and rotations of the robot respectively. The third Gaussian models the uncertainty occurs when estimating the x or y coordinate of the robot pose due to a translation in y or x direction respectively. The last Gaussian models the error in the estimation of rotation due to pure translations.

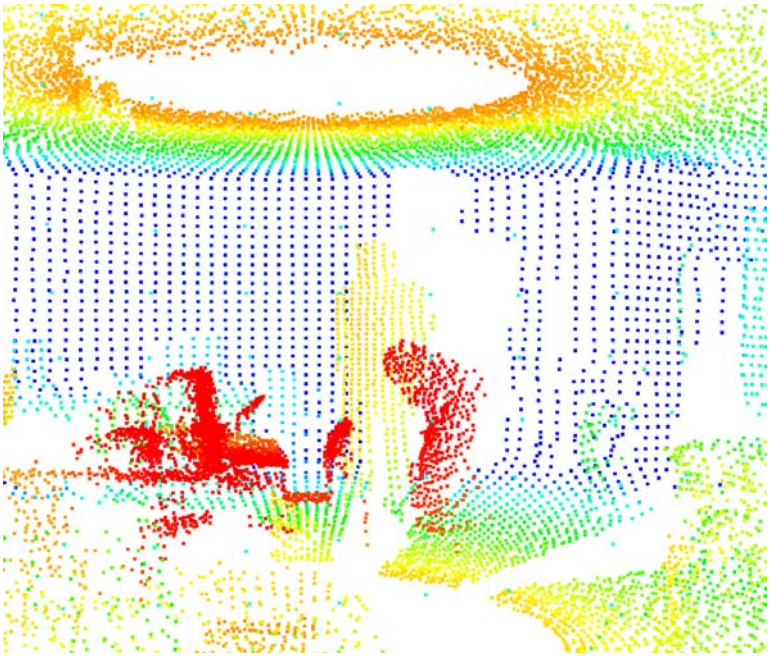
Information delivered by odometry is cheap and easy to use. Therefore, odometry has become a central part of most localization systems. It is highly reliable over short distances and low speeds, but as the traveled distance or the speed increases, the reliability of odometry degrades, due to the unbounded positioning error.



(a)



(b)



(c)

Figure 4.2: (a) An example of odometry data used by mobile robots. (b) A 2D laser scan. (c) A 3D laser scan.

To use the odometry information, four coordinate systems are considered (figures 4.3 and 4.4). The first is the odometry coordinate system, which is marked with "odometry". It is defined as the robot coordinate system when the robot was turned on, or the robot coordinate system when the encoders were zeroed. The other two are the robot coordinate systems at times *t* and *t+1* respectively. After every motion step, the rotation and translation of the robot are calculated as well as the *x* and *y* components (x_{robot}, y_{robot}) of the robot's position at *t+1* in the robot's coordinate system at *t* as follows:

$$rot = phi_{t+1} - phi_t \tag{4.1}$$

$$dx_{odometry} = x_{t+1} - x_t \tag{4.2}$$

$$dy_{odometry} = y_{t+1} - y_t \tag{4.3}$$

$$trans = (dx^2_{odometry} + dy^2_{odometry})^{1/2} \tag{4.4}$$

$$\begin{bmatrix} x_{robot} \\ y_{robot} \end{bmatrix} = \begin{bmatrix} dx_{odometry} & dy_{odometry} \end{bmatrix} \cdot \begin{bmatrix} \cos(phi_t) & -\sin(phi_t) \\ \sin(phi_t) & \cos(phi_t) \end{bmatrix} \tag{4.5}$$

To represent errors of the odometry model, random noise must be added to x_{robot}, y_{robot} and *rot*, yielding $x_{robot\ noise}$, $y_{robot\ noise}$ and rot_{noise} respectively.

$$\begin{bmatrix} x_{robot\ noise} \\ y_{robot\ noise} \\ rot_{noise} \end{bmatrix} = \begin{bmatrix} x_{robot} \\ y_{robot} \\ rot \end{bmatrix} + \begin{bmatrix} x_{robot} & y_{robot} & 0 & 0 \\ y_{robot} & x_{robot} & 0 & 0 \\ 0 & 0 & rot & trans \end{bmatrix} \cdot \begin{bmatrix} \sigma_{trans} \\ \sigma_{drift,trans} \\ \sigma_{rot} \\ \sigma_{drift,rot} \end{bmatrix} * randn \tag{4.6}$$

where *randn* is a normally distributed random number.

The fourth coordinate system is the world or map coordinate system (figure 4.4). When the robot moves the odometry coordinate system will remain fixed with respect to the world coordinate system under ideal conditions. However, due to wheel slippage and an imperfect odometric model, the odometry coordinate system will drift.

Every sample (*x, y, phi*) that holds an estimation of the robot pose will produce a new sample (x_{new}, y_{new}, phi_{new}) using the odometry information as follows:

$$\begin{bmatrix} x_{new} \\ y_{new} \\ phi_{new} \end{bmatrix} = \begin{bmatrix} x \\ y \\ phi \end{bmatrix} + \begin{bmatrix} x_{robot\ noise} & y_{robot\ noise} & rot_{noise} \end{bmatrix} \cdot \begin{bmatrix} \cos(phi) & \sin(phi) & 0 \\ -\sin(phi) & \cos(phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

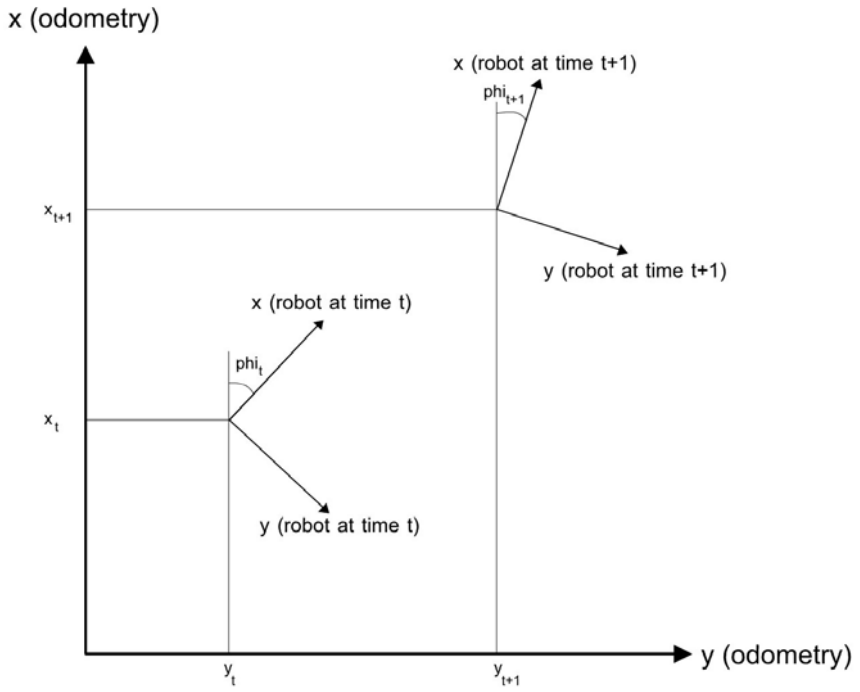


Figure 4.3: The odometry information is given in the odometry coordinate system.

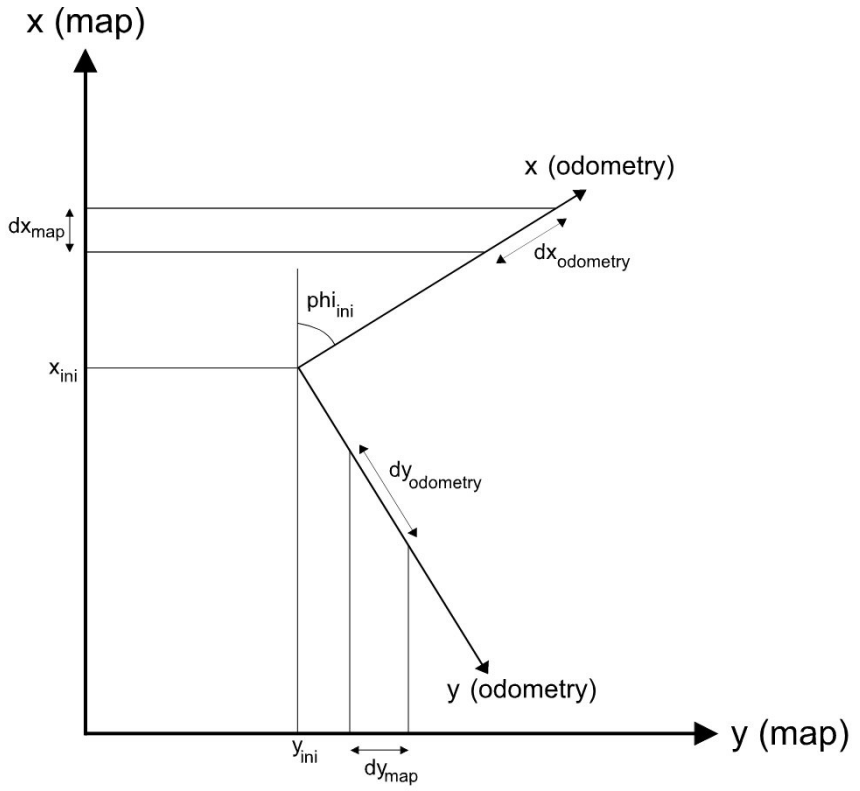


Figure 4.4: Odometry coordinate system relative to the map coordinate system.

4.1.2 Laser Range Finders

Laser sensing has applications in so many areas. In the field of mobile robots, estimating the distance to objects in the environment is a key objective. One of the dominating technologies for doing this is the laser range finder shown in figure 4.5. Every range finder scan is a finite sequence of numbers, which represent the x and y components (with respect to the robot's coordinate system) of the distance to the nearest obstacle along the direction of the laser beam. From these x and y components, the distance to the nearest obstacle and direction of the laser beam can be easily calculated.

4.1.2.1 Classification

Laser range measurements are usually based on one of the following techniques [Dud2000, Jen2001]:

1. Triangulation

In triangulation-based laser range finders the geometric relationships between the outgoing light beam, the incoming ray, and its position on the film plane is used, which is essentially the same principle used with range finders based on regular light sources. Laser sources, however, have the advantage of being better collimated.

2. Time-Of-Flight (TOF)

In TOF-based systems, the time delay for an outgoing short laser pulse to hit an object and return is measured. This is the same measurement principle as for standard sonar sensors. TOF-based laser sensors are often referred to as *laser radar* or *lidar*. A high precision means for measuring time is needed to realize such a system. In terms of mobile robot localization, a resolution in the order of centimeters is desirable. The speed of light is approximately $3 * 10^8$ m/s. Thus, the precision in time has to be in the order of 100 ps, i.e. corresponding to a frequency of 10 GHz. This puts high demand on the laser sensor. It is worth mentioning that one advantage of the short pulses is that higher levels of power can be used, giving better range coverage, keeping a high safety level, and consuming low power. Available systems on the market already have below centimeter accuracy.

3. Phase-Shift

In phase-shift-based laser sensors, a continuous wave is transmitted. The key idea is to compare the phase of the returned signal with a reference signal generated by the same source. Using the Doppler shift, the velocity of the detected object as well as the distance to it can be measured. The main disadvantage of such systems is that it can not distinguish between phase-shifts greater than one wavelength from a phase-shift smaller than one wavelength, i.e. ranges above the wavelength of the modulator can not be distinguished from ranges below it.

(a) The SICK scanner. (b) The IBEO scanner.

Figure 4.5: Examples of commercial laser range finders used for mobile robots.

Briefly, unlike triangulation-based systems, TOF and phase-shift systems exploit the time delay for signal propagation rather than geometric effects. The use of a narrow collimated beam can permit high resolution and good power efficiency. This is possible with lasers, because they are coherent sources that can produce short pulses that remain collimated over long distances.

Low-power lasers have an operating range of few meters, whereas higher power systems may operate over distances of a kilometer or more.

Most of the commercial laser systems do not measure distances in a single direction. To achieve a scanning effect, they are typically mounted on either a rotating unit (e.g. pan-tilt) or as a mirror assembly that covers a wide view of the environment. Laser sensors are an effective alternative to other sensing technologies (e.g. sonar sensors) for robot localization given their accuracy and repeatability, although they are still expensive compared to sonar sensors.

It is possible to add one more degree of freedom and let the laser sensor scan, e.g. up-down as well; yielding 3D scans [Wul2003], which is the case in this work (see figure 4.6). 3D scans, thus, overcomes the perception quality of 2D scans, which provide range information limited to a plane, i.e. only a 2D intersection of the 3D world can be sensed. However, 3D laser measurements are not instantaneous like 2D ones, because 3D scans require more time for its mechanics to rotate. Therefore, it must be thought about compensating for the motion of the robotic platform [Ise2003].

(a) A SICK 3D scanner.

(b) An IBEO 3D scanner.

Figure 4.6: 3D scanners developed at the Institute of Systems Engineering, Real-Time Systems group.

In this work a laser range finder of the company SICK is employed in the 3D scanner system as shown in figure 4.7. The model used is LMS 291-S14, which is an example of a TOF laser scanner.

Figure 4.7: The 3D scanner system consists of a SICK LMS 291 2D laser scanner and a rotating scan drive.

4.1.2.2 Material Dependence

Objects detected by laser range finders are usually divided into 6 categories depending on their geometric and surface characteristics [Jen2001].

(1) Flat and smooth surfaces (e.g. buildings, walls, wood)
(2) Small isolated objects (e.g. tree trunks, poles, wires)
(3) Depth texture (e.g. bushes, grass, textile, wool, foam)
(4) Transparent and semi-transparent surfaces (e.g. windows, plastics, glass)
(5) Reflecting surfaces (e.g. mirrors, wet smooth surfaces, polished steel)
(6) Absorbing objects (mate dark surfaces, smoke, foam rubber)

It can be noticed that objects of type (1), (2) and (4) are dominant in indoor environments (e.g. offices, laboratories). To increase the accuracy of data delivered, a filter can be designed to handle data from a particular object type, or the laser sensor may be combined with some other source of information in order to achieve a robust system, e.g. in environments full of glass material which appear transparent for laser.

4.1.2.3 Ray-tracing

The model of the laser range finder is already described in section 3.2.2. Following the description, the distribution $p(o_t \mid s_t, m)$ is a mixture of three densities:

(1) A Gaussian *P* that models the event of measuring the correct distance with a small added Gaussian noise. This density is calculated as

$$P = \frac{1}{\sigma_{laser}\sqrt{2\pi}} e^{-\frac{(d-ol)^2}{2\sigma_{laser}^2}} \tag{4.8}$$

where σ_{laser} is the standard deviation, *d* is the measured distance by the laser range finder at a given location, and *ol* is the expected measurement at the same given location calculated from the map by ray-tracing. *P*, thus, denotes the probability of measuring *d* at the given location. If this given location is correct, *P* will yield a high value, and vice versa.

(2) A constant small value to model objects that cannot be represented in the map, e.g. furniture, or dynamic obstacles that appear randomly, e.g. walking people (see left side of figure 3.3b).

(3) A high value that models max-range measurements (see right side of figure 3.3b).

The goal is to calculate $p(o_t \mid s_t, m)$ for every scan point in the sequence of one complete scan and sum up all the results to represent the weight of one sample. This procedure is repeated for every sample in the sample set, which is the third step of the MCL algorithm as described in 3.4.

The rest of this section is dedicated to explain how to compute *ol* by ray-tracing. Given a pose, i.e. a sample, and a map of the environment, the distances to nearest obstacles in that environment must be determined. These distances represent the expected measurements that would be obtained if the robot were at the given pose taking a scan. Thus, *ol* represents a single expected distance measurement in a complete scan. It is determined by calculating the length *b* of the straight line L_2 starting at the given pose and ending at the intersection point with the nearest obstacle (line) in the environment. The known parameters are: the coordinates of the start and end point of a line L_1, i.e. (x_1, y_1) and (x_2, y_2) respectively, the coordinate of the start point of L_2 (x_3, y_3), and its orientation Θ. The value of *b* is determined by solving the following equation system:

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + a\begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} = \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} + b\begin{pmatrix} \cos\Theta \\ \sin\Theta \end{pmatrix} \tag{4.9}$$

If $a < 0$ or $a > 1$, then the two Lines L_1 and L_2 intersect outside the specified length of L_1, and *b* must be discarded. This procedure is repeated with every line in the environment. The smallest value of *b* is the desired value. Intuitively, the value of *b* must not exceed the maximum working range of the laser range finder.

4.2 World Representation

There are many ways to represent the knowledge about indoor semi-structured environments. A typical example of such environments is an office setting, which is the case in this thesis. The major paradigms for mapping indoor environments include: *topological, grid-based* and *feature-based* representations. In the following, a short description of theses representation paradigms will be given.

Topological representations:

Here, the world is represented as a connected graph [Cha1985], where the nodes in the graph correspond to *places* of importance and the edges are the *connections*. A Place is defined as an area that is a functional or topological unit. A topological unit might be a room, a corridor,

etc., whereas a printer and a TV set are functional units. Examples of connections are doors, stairways and elevators, which connect the places, hence the name.

Grid-based representations:
The world is divided into a grid. Each cell represents a small area (or volume) of the world. The size specifies the resolution and accuracy of the map. A likelihood value expressing the probability of being occupied is annotated to each cell.

Feature-based representations:
Geometric features, such as points and lines, are used to model the world.

The main disadvantage of topological approaches lies in its coarse resolution, leading to a low positioning accuracy. Precise positioning using grid-based methods cannot be achieved without high computational costs. Therefore, the decision was made to employ the feature-based approach in this thesis to model the environment. A feature-based map can in general be expressed as

$$m = \{f_i \mid i = 1,\ldots,M\} \tag{4.9}$$

where f_i is a feature and M is the number of features in the map. In the context of this work, a concise map is generated using lines in the environment (figure 4.8).
A good modeling practice is to develop a simple model that captures the large scale structure of the environment and to assure robustness over time. Considering typical indoor environments, it is found that the most dominant large scale features are the walls that define the border of the room. These walls are not likely to move with time, at least in the near future. This confirms the feasibility of using a line feature map in the environment model of this work. The lines in the map are measured by hand using a tape measure. The total number of lines is 44, representing an L shaped hallway and 14 rooms.
Description of the different data inputs for the localization system of the robotic platform in use has been discussed. In the next chapter, software implementation details will be given.

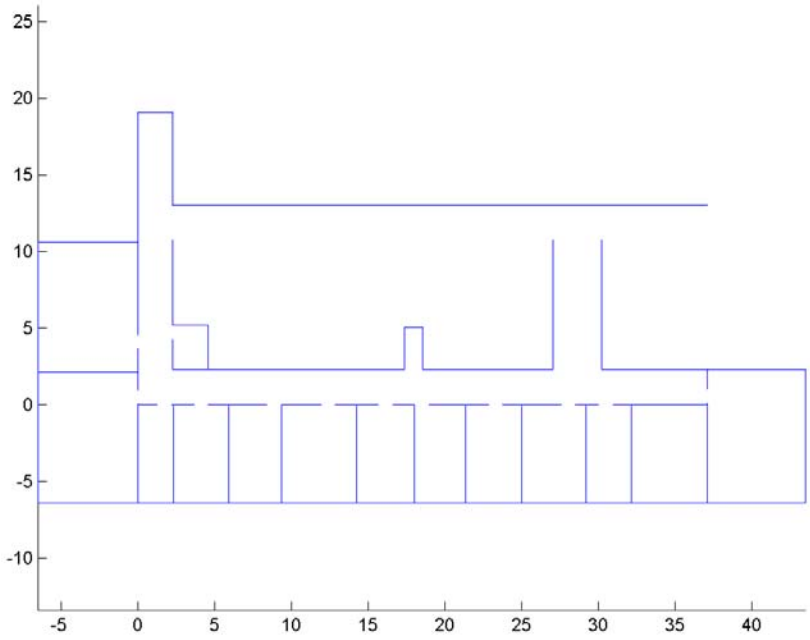


Figure 4.8: A line-based map of the Institute of Systems Engineering, Real-Time Systems group.

Chapter 5

Software

This chapter describes the software system developed for the MCL version, which utilizes a 3D laser scanner for the world perception. Figure 5.1 shows the course of the continuous update of the robot's belief. This process combines data from odometry and 3D laser scans with a given line-feature map of the environment to generate the belief about the true pose of the robot. The MCL is an anytime algorithm, i.e. it will always provide a pose estimation whenever needed using the most up-to-date information at hand.

The MCL is a part of the map module developed earlier for the robotic platform. It uses standardized data packages coming from different sensors. The whole software system is coded in the C Language. The MCL implementation utilizes 12 functions to achieve its goal (see figure 5.2). The map module has many functions, four of them are: The *moduleInit*, which call necessary functions at the initialisation phase and loads the map of the environment. This function runs only once at startup. The *moduleOn* is turned on to assure continuous odometry and laser data flow. It initialises variables that will hold odometry data and the estimated pose. Furthermore, the initial belief of the robot is initialised according to the available information at the moment, which determines whether it is about position tracking or global localization. The *moduleOn* generates error messages if either the continuous odometry or laser data flow is interrupted. The *moduleOff* turns the map module off. In the *moduleLoop* function, the MCL expresses its potentials in providing precise pose estimates as long as the robot moves and senses its world.

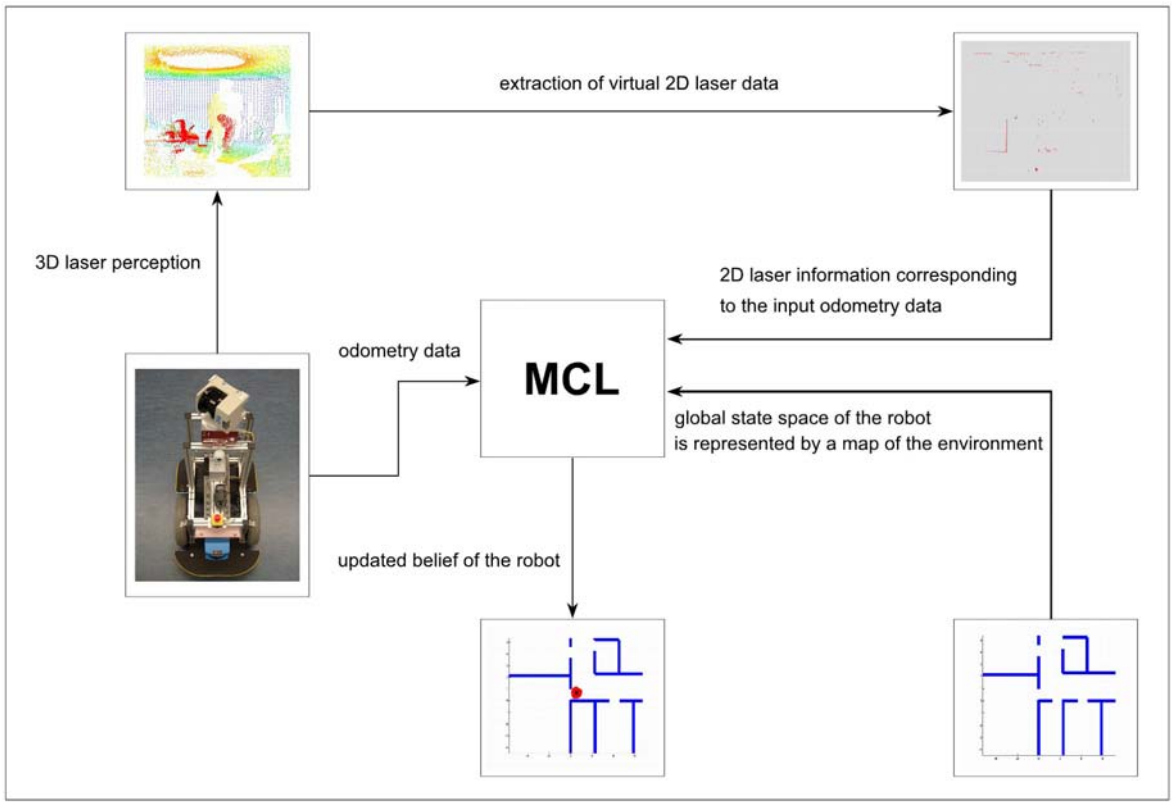


Figure 5.1: System overview of the developed 3D laser scanner based MCL.

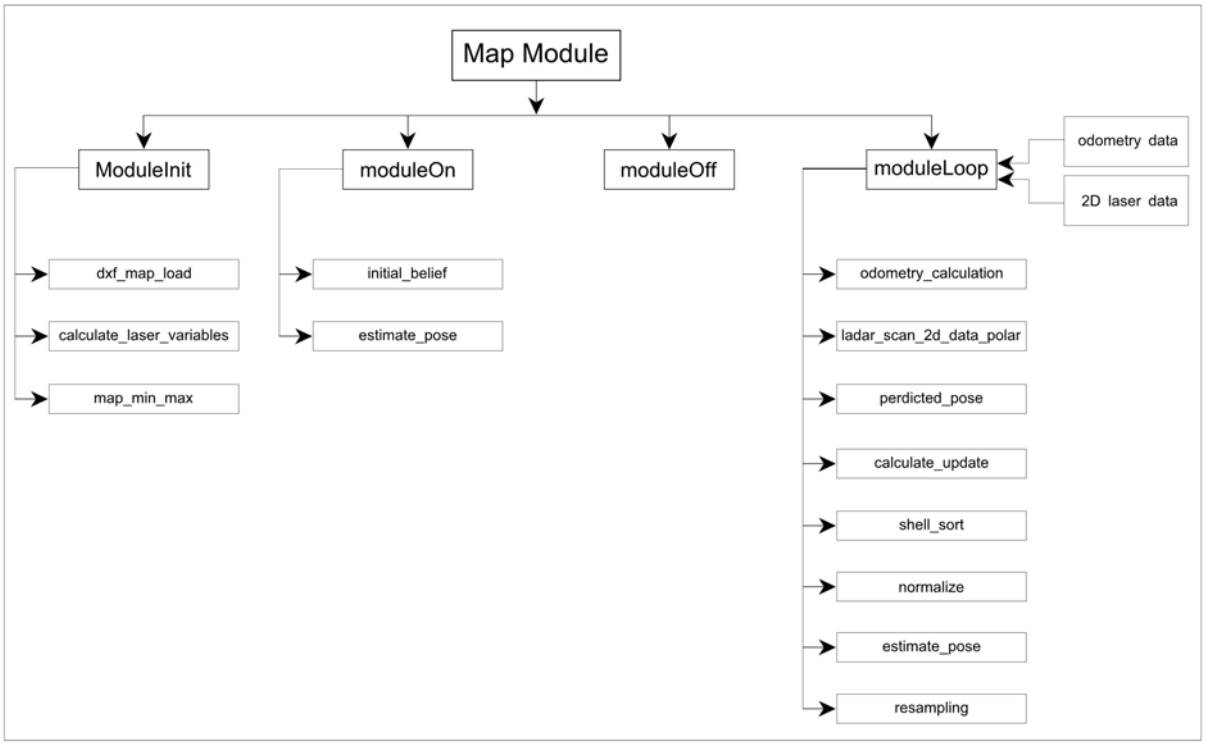


Figure 5.2: MCL software architecture.

5.1 Functions

In this section a short description of the functions developed for the MCL will be given.

`dxf_map`

This function loads and saves line-feature maps in AutoCAD dxf file format.

Input: An AutoCAD dxf file that contains line features.

Output: A map array that holds the x and y coordinates of the start and end point for every line and an integer that represent the total number of lines in the map.

`calculate_laser_variables`

Calculates terms of the Gaussian function and the laser sensor model.

Input: Standard deviation of the laser sensor model, which is constant.

Output: Global variables that represent the Gaussian function and the sensor model.

`map_min_max`

Determines the minimum and maximum x and y coordinates of the world model, which will be used to assure that samples are never placed outside the assigned environment.

Input: An array with x and y coordinates of the lines in the given map and an integer that holds the total number of lines.

Output: Four variables that hold the maximum and minimum values of the x and y coordinates in the map.

`initial_belief`

The developed localization system proved its capability to solve position tracking and global localization problems as will be shown in chapter 6. In practical applications of the robotic platform neither the precise starting pose will be given nor a pure global localization will be

run. The initial samples will always be randomly generated in a circle around an approximate starting pose. The diameter of the samples circle can vary according to the structure of the environment. Standard deviation of the initial orientation lies between 30° and 45°.
This function generates a circle cloud of random samples around an approximate starting pose.

Input: An array for the pose belief of the robot, an approximate starting pose and the number of samples.
Output: The initial belief array of samples with equal importance factors.

`estimate_pose`

Generates an estimate for the current pose of the robot. It represents the average of the best 10% samples in the current belief.

Input: An array of the current belief of the robot and the number of samples.
Output: A vector that holds an estimation of the current xy position and orientation of the robot.

`odometry_calculation`

Calculates the translation and rotation performed during the last motion step. Furthermore, it determines the x and y components of the robot's position in the previous robot coordinate system. These variables are to be used by `predicted_pose`.

Input: The current and previous odometry data.
Output: The translation and rotation performed in the last motion step and the robot's position in the previous robot coordinate system.

`ladar_scan_2d_data_2_polar`

Transforms laser scan data from xy coordinates into polar coordinates that represent distances to landmarks and orientation of laser beams.

Input: x and y coordinates of the virtual 2D laser scans.

Output: Distances to and orientations of the detected landmarks.

predicted_pose

Generates new samples according to the motion model and odometry variables already calculated by `odometry_calculation`.

Input: The last translation and rotation performed, maximum and minimum coordinates of the map, the robot's position in the previous robot coordinate system, current belief of the robot, the number of samples and the parameters of the motion model.

Output: A new belief according to the last action performed and the motion model with equal importance factors.

calculate_update

It is the most CPU time consuming function of the developed MCL. It determines the weight of every sample using laser scanner data and a map of the environment as described in 4.1.2.3.

Input: The current belief, number of samples, the map array, total number of lines in the map, standard deviation of the sensor model, terms of the Gaussian function and the laser sensor model, distances to and orientations of the detected landmarks and the maximum range of the employed laser system.

Output: The belief array where each sample is assigned with an importance factor representing its weight.

shell_sort

Sorts the samples in the belief array according to their weights. The sorting algorithm used is "shell sort".

Input: The current belief array with unsorted samples, and the total number of

samples.

Output: The belief array with the samples sorted according to their weights.

`normalize`

Normalizes the weights in the belief array so that they add up to the number of samples used.

Input: The belief array and the number of samples.

Output: The belief array with modified weights so that they add up to the number of samples.

`resampling`

Duplicates samples of higher weights and discards weak samples from the belief.

Input: The current belief and the number of samples.

Output: A new belief produced from the strongest samples in the previous belief.

5.2 Programming Issues

The generation of normally distributed random numbers is an important issue in any probabilistic algorithm such as MCL. Furthermore, MCL must perform a sorting procedure in every run of its loop. These two issues will be handled in the next subsections.

5.2.1 Generating Gaussian Random Numbers

MCL is a probabilistic algorithm that utilizes the normal (Gaussian) distribution many times in the motion and perception models. Therefore, providing Gaussian random numbers is of great importance for the MCL implementation.

In most cases, software- and hardware-based random number generators are actually considered *pseudo-random number generators* because they have a finite number of values to work from. They usually extract these values from their surroundings, which are predictable in nature—the values can come from the system's time or from CPU cycles. If the starting values are predictable, the numbers they generate cannot be truly random. An example of a true random number generator would be a system that collects radiation from a radioactive item. The elements that escape from the radioactive item do so in an unpredictable manner, and the results are used as seed values for key generation.

Many quantities of the motion and perception model in the MCL algorithm are distributed according to a normal distribution

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}} \tag{5.1}$$

where $\bar{x}$ is the average value for x and σ is the standard deviation.

The problem now is to generate Gaussian pseudo-random numbers given a source of uniform pseudo-random numbers. Actually, there are many ways to solve this problem, but only one important method will be briefly discussed here. If we have an equation that describes our desired distribution function, then it is possible to use some mathematical trickery based upon the *fundamental transformation law of probabilities* to obtain a transformation function for the distributions. This transformation takes random variables from one distribution as inputs

and outputs random variables in a new distribution function. Probably the most important of these transformation functions is known as the Box-Muller formula [Box1958], which is a polar method. Roughly speaking, this method involves picking a point uniformly distributed within the unit circle and then projecting it. The routine assumes the existence of a uniform random number generator. It transforms two uniformly distributed deviates, z_1 and z_2, into two independent normally distributed deviates, y_1 and y_2. The expressions for the transformation are

$$y_1 = \sqrt{-2\ln z_1}\ \sigma \sin(2\pi z_1) \tag{5.2}$$

$$y_2 = \sqrt{-2\ln z_1}\ \sigma \cos(2\pi z_2) \tag{5.3}$$

5.2.2 Sorting Algorithms

Sorting is an algorithm that puts elements of a list into a certain order. It is one of the fundamental problems of computer science, and there is a plethora of solutions to this problem. Some sorting algorithms are simple and intuitive. Others, are extremely complicated, but produce lightening-fast results.

In the resampling step of the MCL algorithm, poses of higher importance factors produce more new samples than poses of lower importance factors. Furthermore, pose that have zero or very low importance factors will die out. Therefore, the resampling step is preceded by a sorting step to order the poses of the robot's belief according to their weights.

Common sorting algorithms can be divided into two classes by the complexity of their algorithms. Algorithmic complexity is generally written in a form known as Big-O notation, where *O* represents the complexity of the algorithm and a value *n* that represents the size of the set the algorithm is run against.

The two classes of sorting algorithms are $O(n^2)$, which includes the bubble, insertion, selection and shell sorts; and $O(n \log n)$, which includes the heap, merge and quick sorts.

The important thing is to pick the sorting algorithm that is appropriate for the task at hand. The $O(n \log n)$ algorithms are blazingly fast, but that speed comes at the cost of complexity. These algorithms make extensive use of recursion (except the heap sort), which should be better avoided when programming for embedded systems in order to avoid unexpected memory use-up.

In this work, the scanning time of the 3D laser system lies between 1.5 and 3 seconds (depending on the rotation speed of the 3D laser system); and the expected maximum number of samples used will never exceed 10000 samples. Therefore, the decision was made to employ an $O(n^2)$ algorithm to exploit their simplicity (with the possible exception of the shell sort). Figure (5.3) shows the performance comparison of the bubble and shell sorts on a Pentium III 500 MHz processor. However, the robotic platform employs a Pentium III 700 MHz processor, which have a better performance by a factor of 1.4.

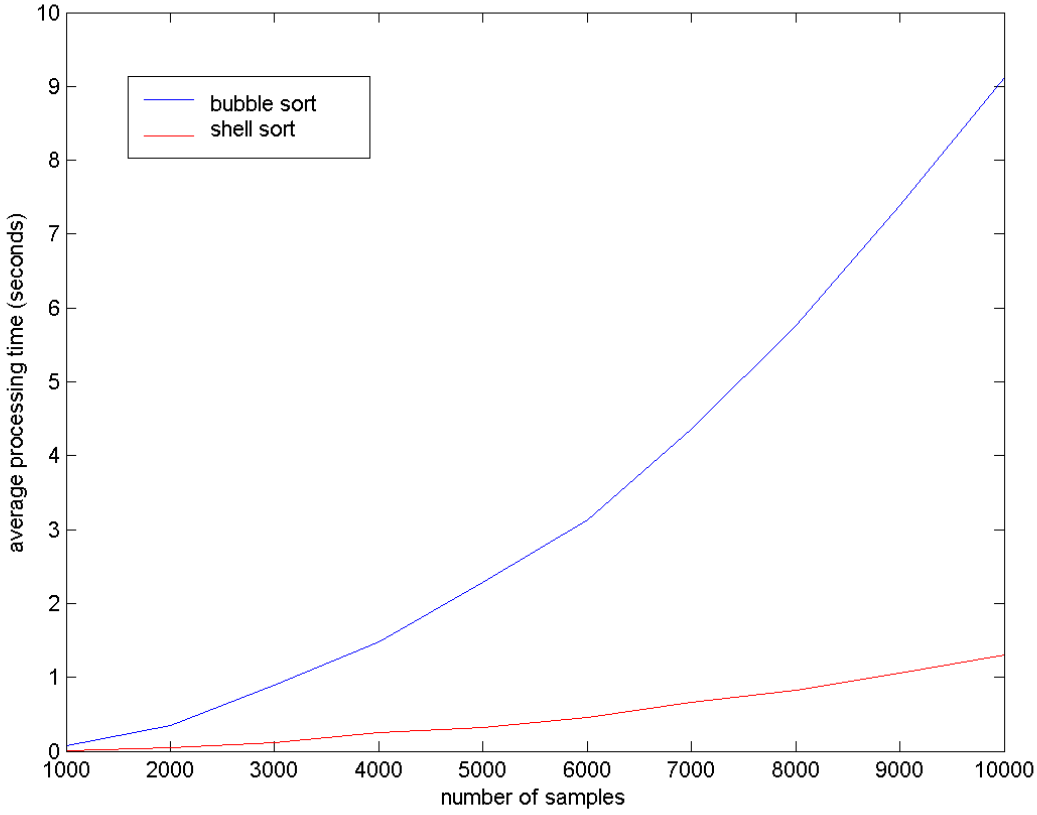


Figure 5.3: Performance comparison of the bubble and shell sorts with different number of samples.

It can easily be noticed that shell sort clearly overcomes bubble sort, hence the decision in the favor of shell sort to use in the MCL implementation.

Chapter 6

Experimental Results

Statistical experiments are setup to evaluate the performance of the developed 3D laser scanner based MCL. Experiments are divided into three sets; position tracking experiments, general experiments and global localization experiments.
In the position tracking experiments, the effects of the following parameters on the localization error (translational and rotational) have been analysed:

1. Number of samples
2. Standard deviation of the laser sensor model
3. Standard deviation of translation
4. Standard deviation of translational drift
5. Standard deviation of rotation
6. Standard deviation of rotational drift

Each parameter setting is run 10 times to get reasonable results for the statistical analysis.
The general experiments have been run before the global localization tests to study the effects of the number of samples and the number of scan points to consider on the performance of one MCL iteration. The results of these tests can help reduce simulation time for global localization experiments. The experiments were commenced by studying the influence of taking different scan points from each scan data on the localization error.

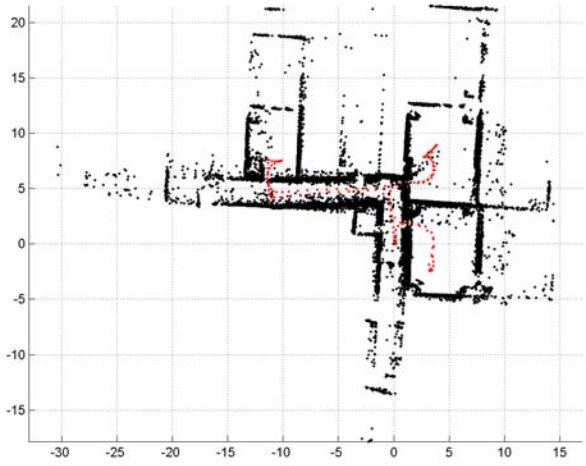
Global localization experiments are conducted to study the chances for correct localization when varying the number of samples and the standard deviation of the sensor model σ_{laser}. Each combination is run 5 times, which are quite enough to indicate the success chances of each combination. Furthermore, The average number of motion steps and average time required for correct localization using different number of samples have been investigated.

6.1 Experimental Setup

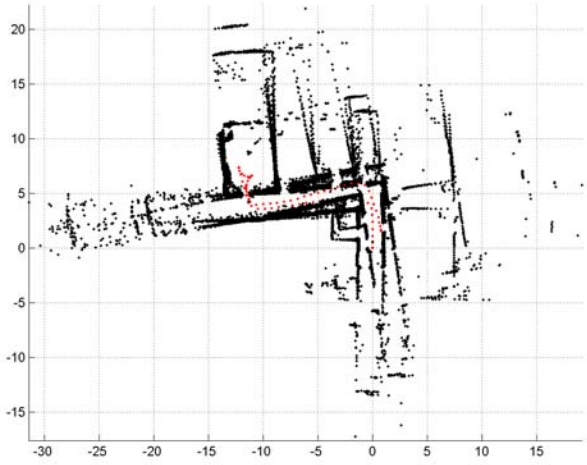
The experiments was conducted at the Institute of Systems Engineering, Real-Time Systems Group, in which the experimental robot platform was commanded wirelessly to start moving from a know pose and return back to the same starting point. Three test journeys were performed as shown in figure 6.1. The robot took perception data and registered odometry information during every journey. Perception information corresponding to each odometry data has been saved for the further simulations with MATLAB. Parameter settings that led the robot erroneously have been discarded. For the accepted settings, the localization error at the end of the journey was the deciding factor to determine the best setting. The decision to reject a parameter setting can be easily made by inspecting the simulated motion of the robot. The next sections will give the results of these experimental studies.

6.2 Position tracking Experiments

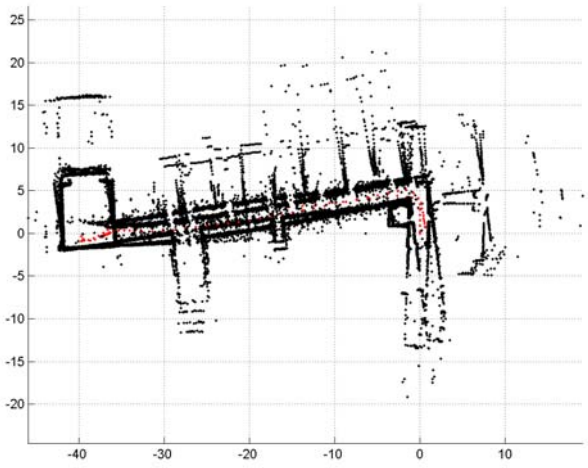
The conducted experiments had the task to uncover the best parameters' settings in order to enhance the pose estimation process. The total number of simulation runs performed for this set of experiments is 3190. If the results of the three tests are similar, results of only one test will be illustrated. In the next sub-sections, the experimental results of testing each parameter will be given and discussed.



(a)



(b)



(c)

Figure 6.1: Path of the test journeys. The robot started moving at (0,0) and returned back to the start point.

6.2.1 Number of Samples and Localization Error

To determine a suitable number of samples to achieve reliable localization, different values for the number of samples were tested against localization error. In figure 6.2, the relationship between the number of samples and localization error is shown. It can be noticed that all values yielded precise pose estimations with translation error under 20 cm and a rotation error not more than 1°. These results are obtained by simulating the test journey shown in figure 6.1a. For the test journeys shown in figures 6.1b and 6.1c, the translation error was also under 20 cm and the rotation error was under 2° and 3° respectively. According to these results, 400 samples have been chosen to achieve reliable position tracking.

6.2.2 Laser Sensor Model and Localization Error

The next series of experiments had the task to find out how does the standard deviation of the laser sensor model σ_{laser} (already discussed in 3.2.2) affect the precision of position estimation. Figure 6.3 illustrates the relationship between σ_{laser} and localization error for the three cases in figure 6.1. It is clear that optimistic values of σ_{laser} cannot put up with noise usually associated with the delivered laser data. Therefore, the precision of laser readings should be slightly underestimated. Adequate position estimation could be achieved when σ_{laser} has a value of 0.18.

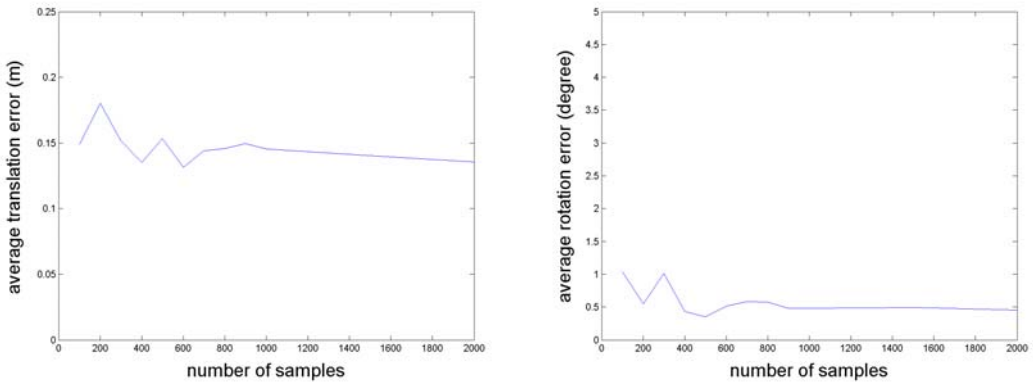
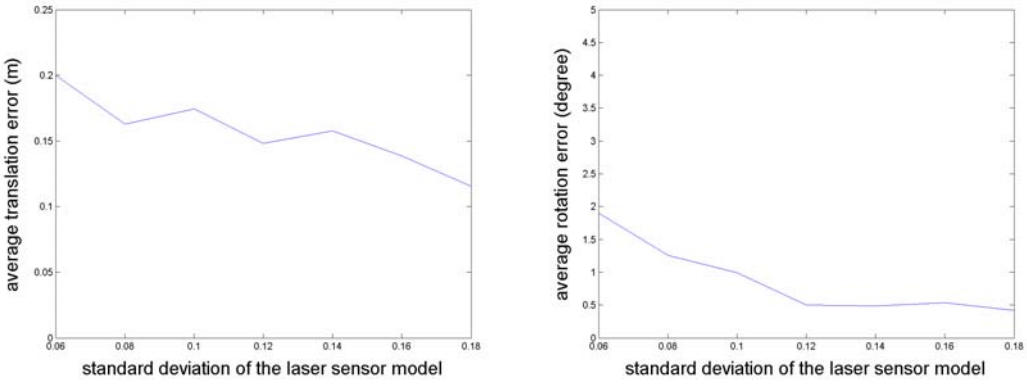
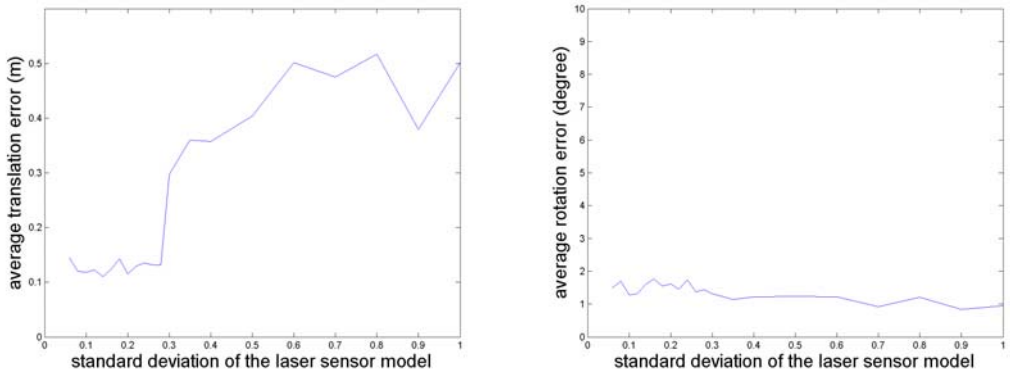


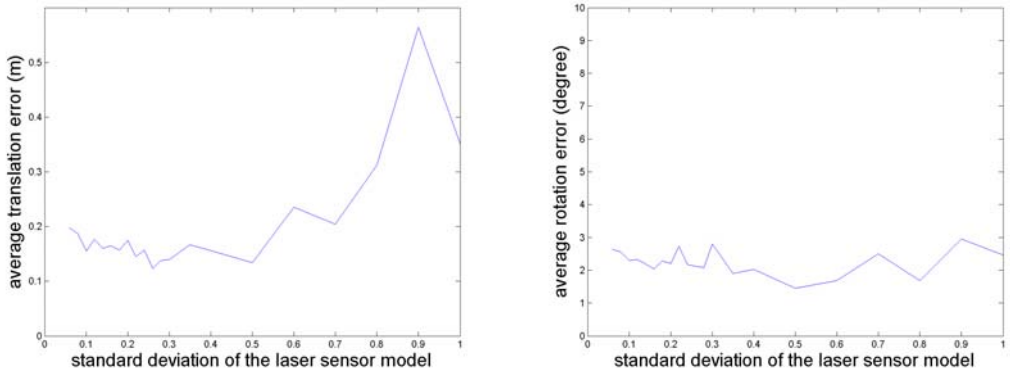
Figure 6.2: The relationship between the number of samples and localization error.



(a)



(b)



(c)

Figure 6.3: The relationship between the parameter of the laser sensor model and localization error for the three test journeys.

6.2.3 Motion Model and Localization Error

The influence of the motion model parameters on the localization quality has been studied. This part of experiments varies the values of four parameters to determine the best ranges for reliable position estimation. Values of only one parameter are varied at a time. The results of this set of experiments are discussed in the following subsections.

6.2.3.1 Standard Deviation of Translation

A large range of values for σ_{trans} has been studied. If only localization error at the end position is considered and the localization quality during the complete movement is neglected, all simulation values for σ_{trans} deliver relatively similar results. However, it is very difficult to measure the localization error at every motion step. Therefore, the matching of scan points with lines in the environment during the robot's motion has been inspected. When the matching is clearly weak, the σ_{trans} setting is discarded whatever is the value of localization error at the end position. In figure 6.4 the relationship between σ_{trans} and localization error is depicted. Only values between 0.12 and 0.22 yield quality position estimation.

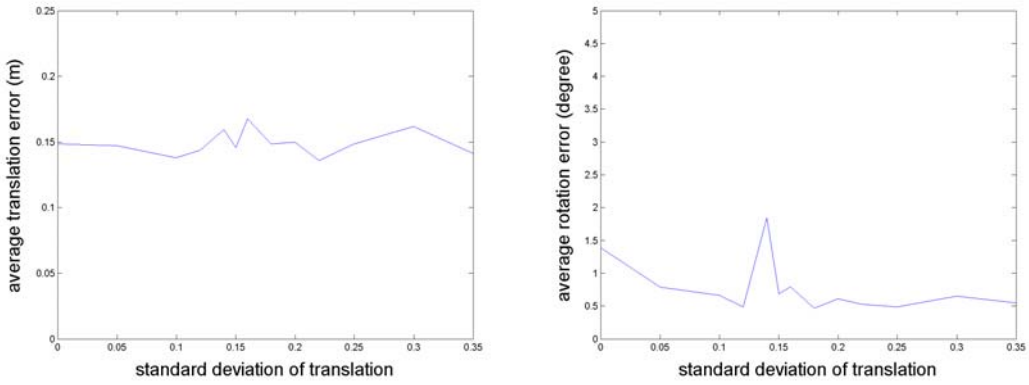
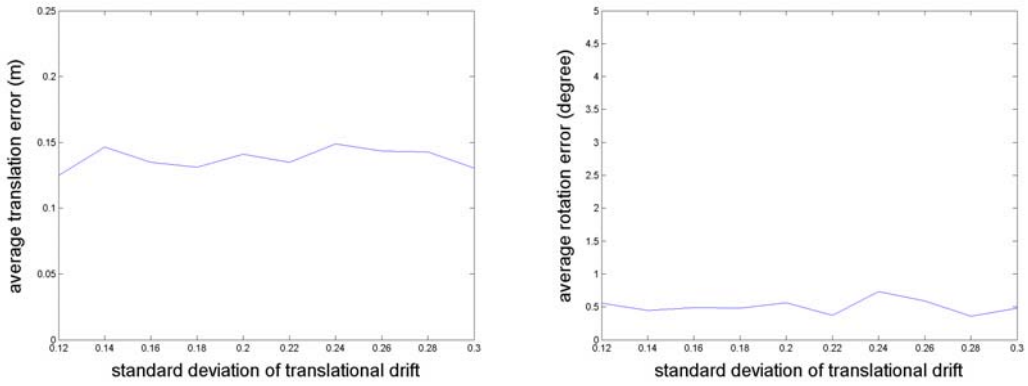


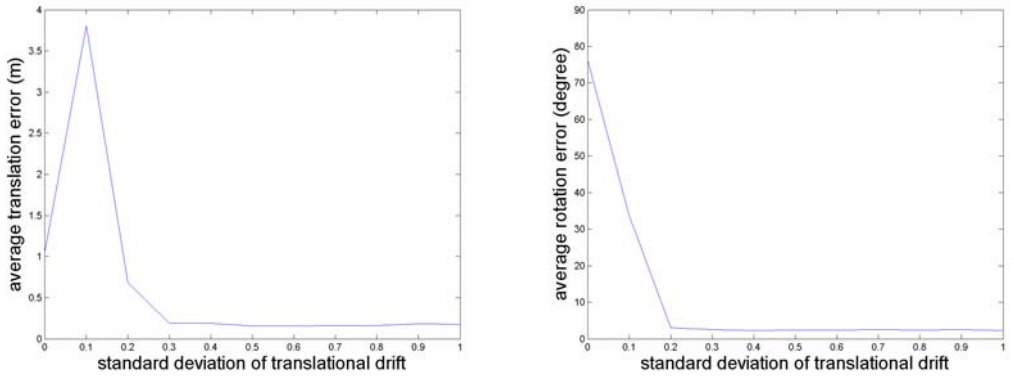
Figure 6.4: The relationship between σ_{trans} and localization error.

6.2.3.2 Standard deviation of Translational Drift

This parameter compensates translational errors occurring during motion in straight lines. Unequal wheel diameters and slippery or non-planner grounds are the main sources of such errors. These errors are known as drift errors. If the robot moves along the x-axis, it may drift left or right, where the y component of the pose delivered by odometry remains unchanged. Such scenarios will lead to inaccurate pose estimation. Therefore, $\sigma_{drift,\ trans}$ has been employed to minimize drift effects. Figure 6.5 illustrates the results of two test journeys. Admitted values for $\sigma_{drift,\ trans}$ lie between 0.12 and 0.3.



(a)



(b)

Figure 6.5: The relationship between $\sigma_{drift,\ trans}$ and localization error.

6.2.3.3 Standard deviation of Rotation

The value of σ_{rot} must be carefully chosen. If high values are allowed the position estimation will quickly fail due to false orientation and the semi-symmetric property of indoor environments. Therefore, it is recommended that σ_{rot} does not exceed 0.01. Figure 6.6 depicts the results of the σ_{rot} experiments. It has been found that a minimum localization error is achieved when σ_{rot} is set to 0.006.

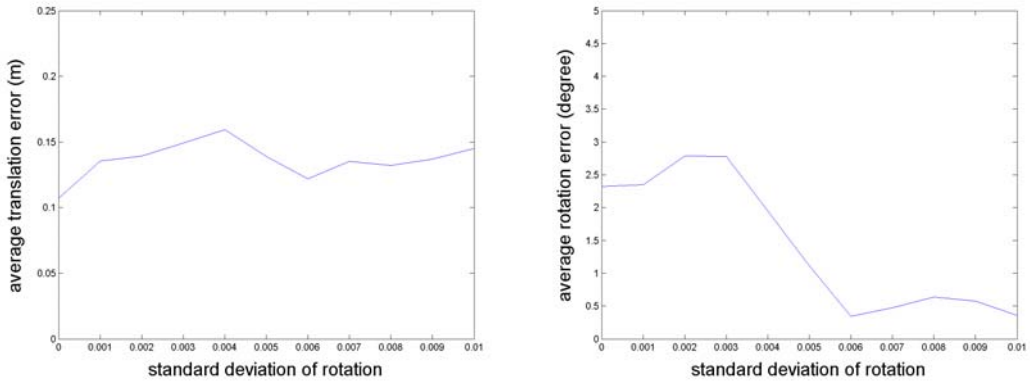
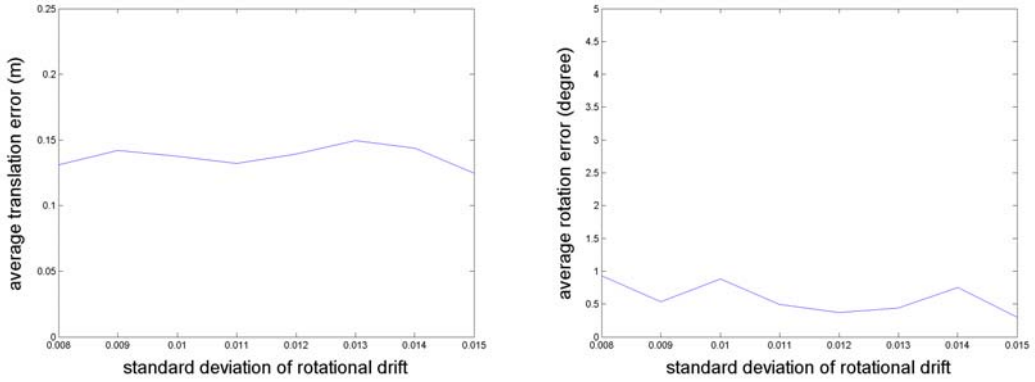


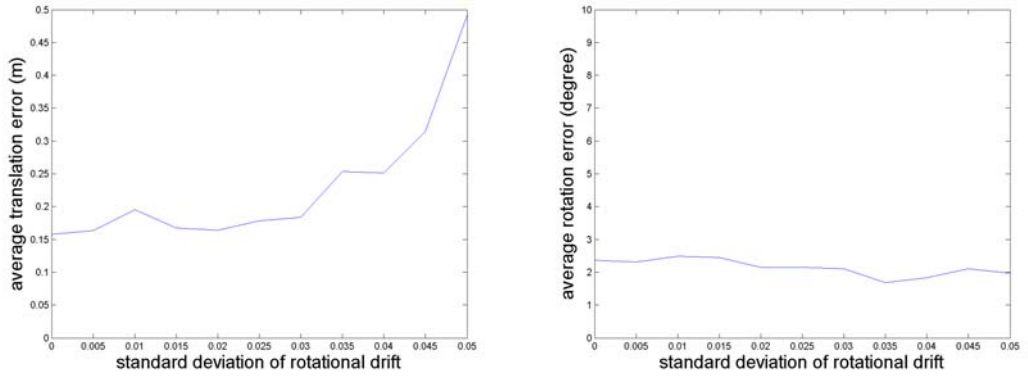
Figure 6.6: The relationship between σ_{rot} and localization error.

6.2.3.4 Standard Deviation of Rotational Drift

Rotational drift works just like translational drift (see 6.1.3.2), but rather it compensates rotational errors due to drift. Unequal wheel diameters and slippery grounds will cause the robot to slightly rotate during straight-line movements. Therefore, it is reasonable not to over estimate the value of $\sigma_{drift,\ rot}$. Figures 6.7a and 6.7b depict the results of the test journeys illustrated in figures 6.1a and 6.1c respectively. Values that yielded acceptable results for the first test journey lie between 0.008 and 0.012. Setting $\sigma_{drift,\ rot}$ to 0.012 enabled good pose estimation. Figure 6.7b illustrates the effect of higher values of $\sigma_{drift,\ rot}$ on position estimation. The translational error increases with increasing $\sigma_{drift,\ rot}$.



(a)



(b)

Figure 6.7: The relationship between $\sigma_{drift,\ rot}$ and localization error.

6.3 General Experiments

Before conducting global localization experiments, it is reasonable to test some parameters that affect the real-time capability of the developed localization system. Performing these experiments saves a huge time that can be spent to experiment global localization with parameter settings that are not feasible in real-time terms, despite their precise pose estimation. The results to show in this section discuss the influence of the number of samples and the number of scan points to consider on the computation time. The utilized 3D laser system delivers 240 measurements when its mechanics rotate at the minimum speed with an angle resolution of 1.5°. It may be redundant and time consuming to consider every scan point delivered by the 3D laser system. Therefore, it is rational to determine how many scan points can be extracted for processing without sacrificing the quality of pose estimation.

Figure 6.8 illustrates the relationship between the number of considered scan points and localization error. This figure indicates that a robust localization can be achieved by taking over 20 scan points. Figure 6.9 depicts the average computation time required to process different number of scan points using 1000 samples on a Pentium III 500 MHz processor. The average computation time required for processing 20 and 24 scan points are approximately 294 and 337 msec respectively. The Pentium III 700 MHz used with the experimental robotic platform "ERIKA" will yield an average computation time of approximately 210 and 241 msec for both cases respectively. The relationship between the number of samples and the average computation time required for processing on 500 and 700 MHz Pentium III using 24 scan points is illustrated in Figure 6.10. It shows that a global localization could be achieved using 10000 samples, since the time between two laser scans is about 3 sec. If more samples are needed for some specific situation, laser scan data can be processed, e.g. every two inputs, rather than taking every input into consideration. Thus, the time available for localization could be increased to approximately 6 sec.

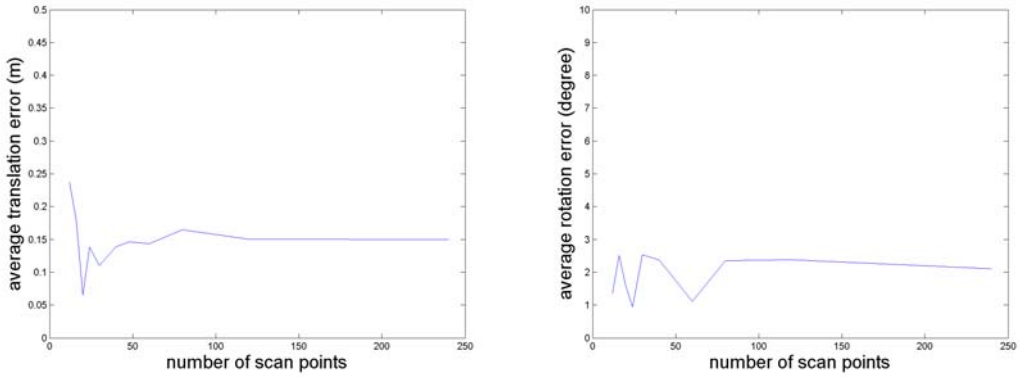


Figure 6.8: The relationship between the number of processed scan points and localization error.

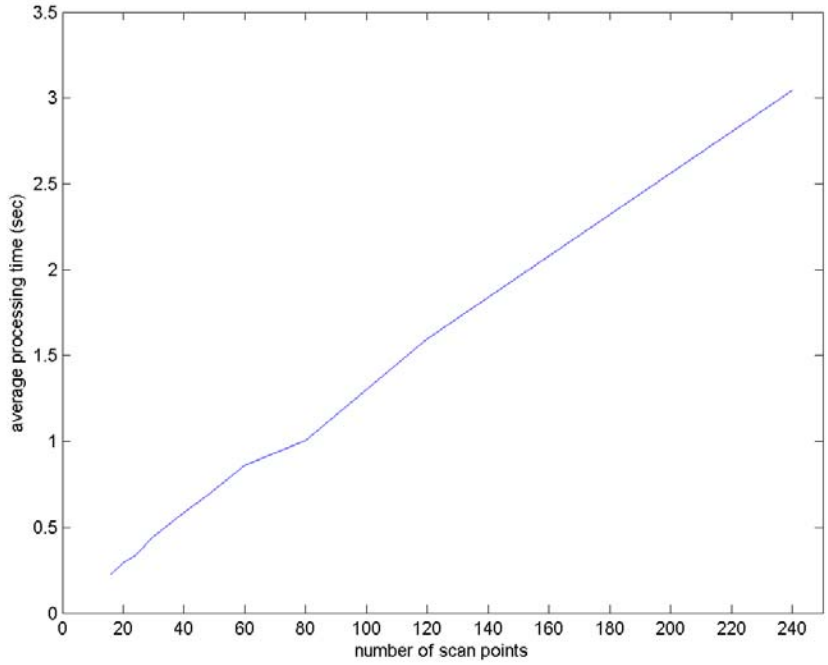


Figure 6.9: The average processing time for different number of scan points using 1000 samples on a Pentium III 500 MHz processor.

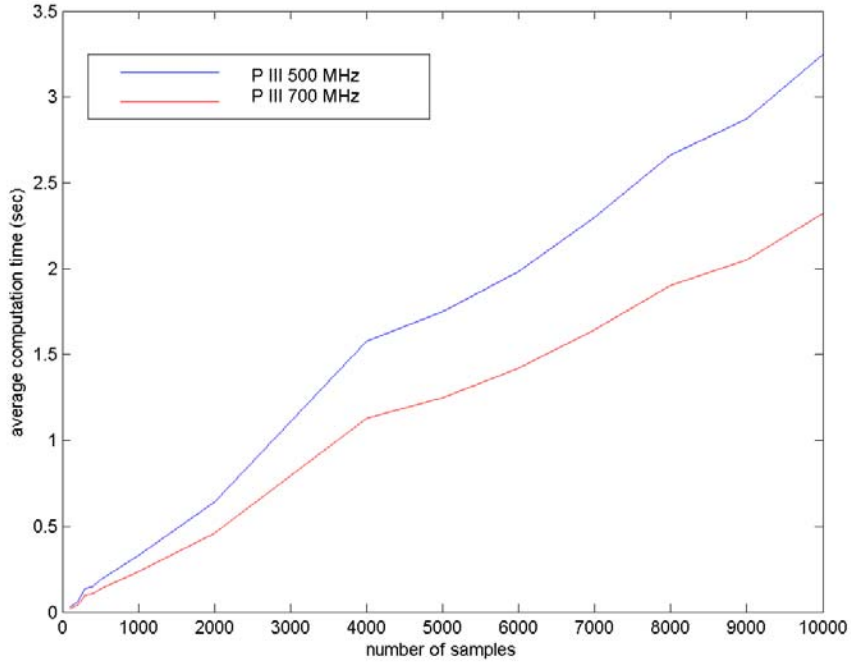


Figure 6.10: The relationship between the number of samples and the computation time required on Pentium III 500 and 700 MHz processors using 24 scan points. The results for P III 500 MHz are simulated and for P III 700 MHz are calculated.

6.4 Global Localization Experiments

6.4.1 Effect of the Laser Sensor Model and Number of Samples

Different settings for the laser sensor model parameter σ_{laser} (0.05 – 0.16) have been tested when varying the number of samples n between 2000 and 10000. The goal was to determine which combination of σ_{laser} and n has the best chances to solve the global localization problem. Each combination is repeated for 5 times, yielding 45 tests for every n setting and a total number of 540 tests. The results are shown in figure 6.11. These results assure that the smaller the value of σ_{laser}, the better the chances to solve the global localization problem. It is recommended to choose σ_{laser} as small as possible. The simulation tool did not allow values for σ_{laser} under 0.05. All settings below 0.05 crashed the system down. This problem is only operating system specific and would not face the software system running on the robotic platform. Tests were repeated for $\sigma_{laser} = 0.05$ to get more reliable results. Figure 6.12 depicts the successful localization chances for different number of samples when $\sigma_{laser} = 0.05$. As can be seen the correct localization chances are at least 70% except for $n = 9000$ samples where the correct localization probability is surprisingly not exceeding 60% even when repeating the tests. This could be due to the way the simulator generates random samples. On the contrary, the developed MCL achieved correct localization with probabilities 80%, 90% and 100% for $n = 3000$; 5000; 7000, 6000 and 8000; 10000 samples respectively.

The results in figures 6.11 and 6.12 are for the test journey shown in figure 6.1a. In this test four doors in the environment were closed during the experiment and consequently these doors also were closed in the world model. It was interesting to repeat the simulations when these doors are opened in the world model, which should be the default case. Figure 6.13 presents the comparison results. It is clear that the more accurate the world model is, the better the chances for correct global localization.

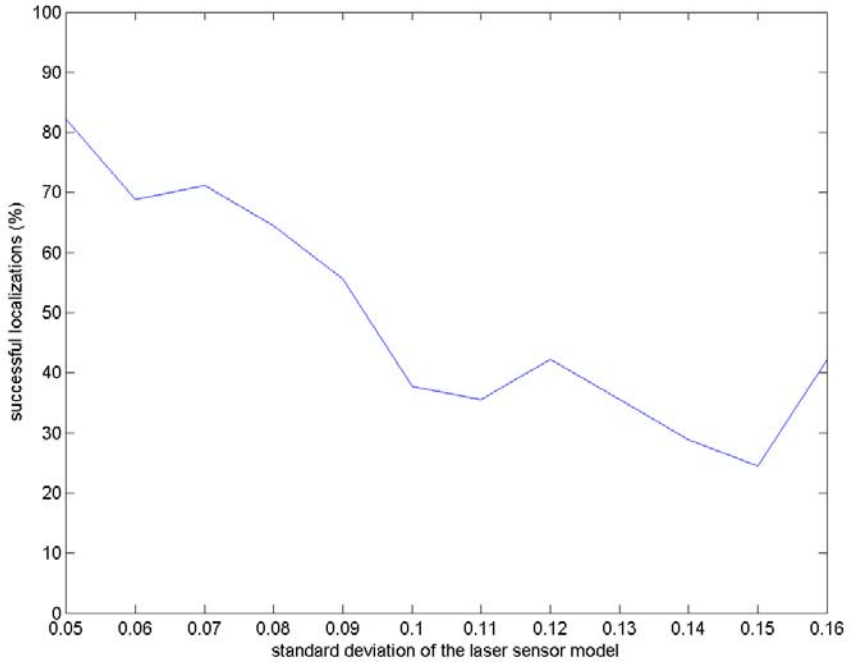


Figure 6.11: The effect of varying σ_{laser} on the capability of the developed MCL to solve the global localization problem.

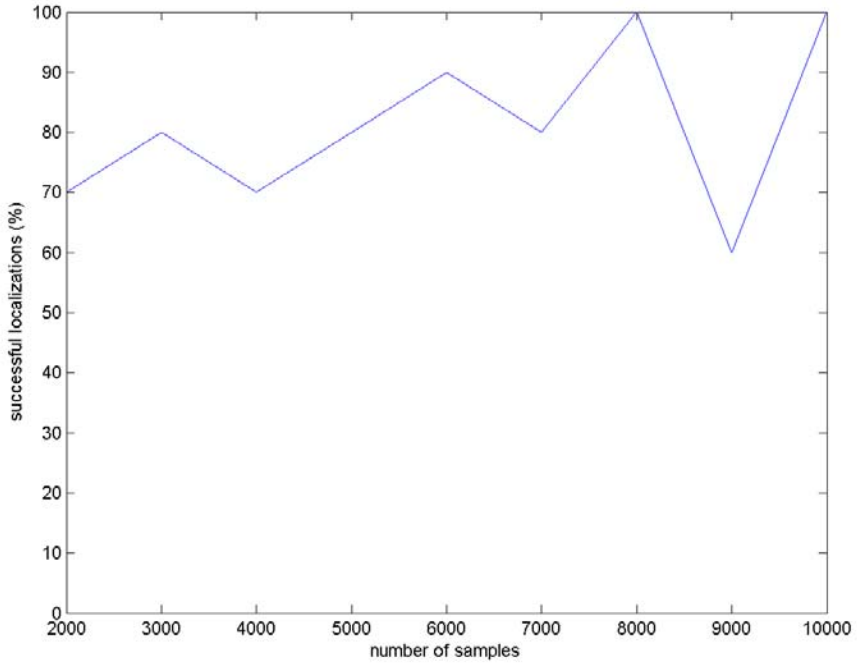


Figure 6.12: Successful localization chances for different number of samples when σ_{laser} = 0.05.

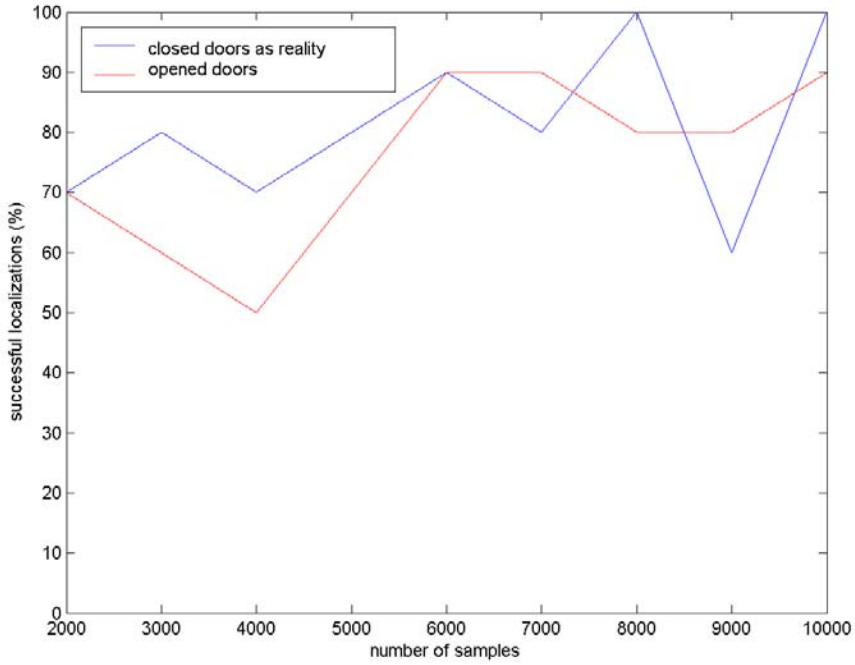


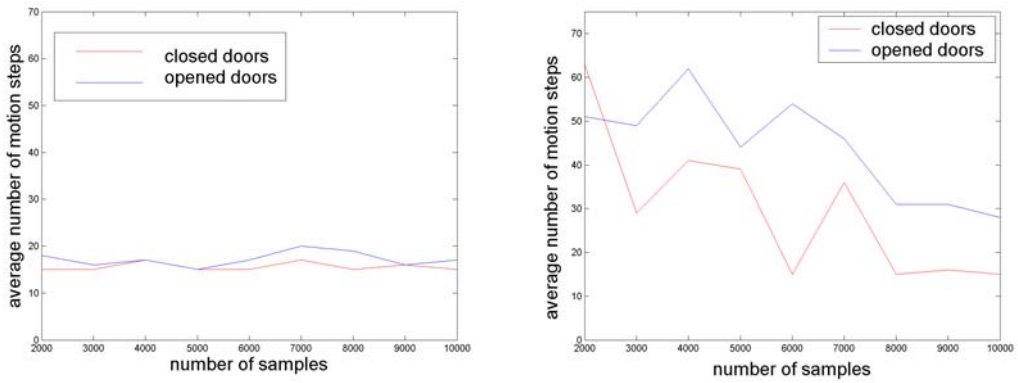
Figure 6.13: Comparison of successful localization chances when doors are closed (as reality) and opened using different number of samples. $\sigma_{laser} = 0.05$.

6.4.2 Average Number of Motion Steps and Average Time

To investigate the average time required for the robot to solve the global localization problem it was necessary to detect the average motion steps to do so. A motion step can be defined as the translation and rotation performed by the robot during one complete laser scan. Since the mechanics of the 3D laser system takes a minimum and maximum time of approximately 1.5 and 3 sec to perform a 360° rotation, the time for one motion step is consequently between 1.5 and 3 sec. Figure 6.14 depicts the average number of motion steps required for the robot to uniquely and correctly localize itself in normal and worst cases. The tests were run for the both cases when the doors were closed (accurate world model) an opened (approximate world model). Figure 6.14 shows again that the more realistic the world model is, the better the results we get. In normal cases the average number of motion steps required does not depend on the number of samples used (see Figure 6.14a). With an accurate world model the robot needs an average of 15 motion steps to solve the global localization problem. Using an approximate world model the robot required on average 17 motion steps. If the 3d laser

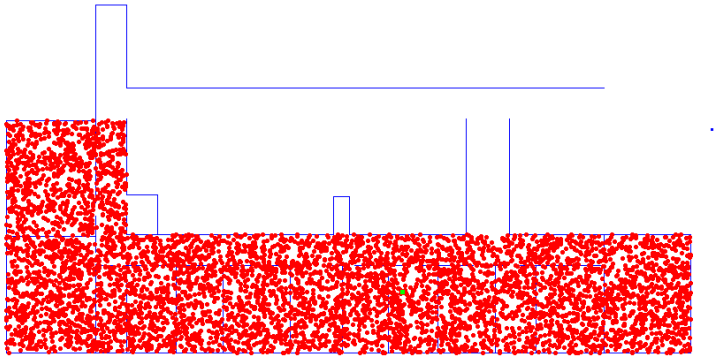
system is rotating at its maximum speed, the average time required for global localization would be 22.5 and 25.5 sec when using an accurate and approximate world models respectively. Worst case results (figure 6.14b) shows that more samples are generally needed to reduce the number of motion steps needed for global localization. With an accurate world model the robot needs on average 30 steps and with an approximate model an average of 44 motion steps. The average time required is 45 and 66 sec respectively if the laser system is rotating at its maximum speed.

Figure 6.15 illustrates the course of a global localization using 6000 samples. The black points represent a virtual 2D scan, the red spots are the samples and the green point is the pose estimation.

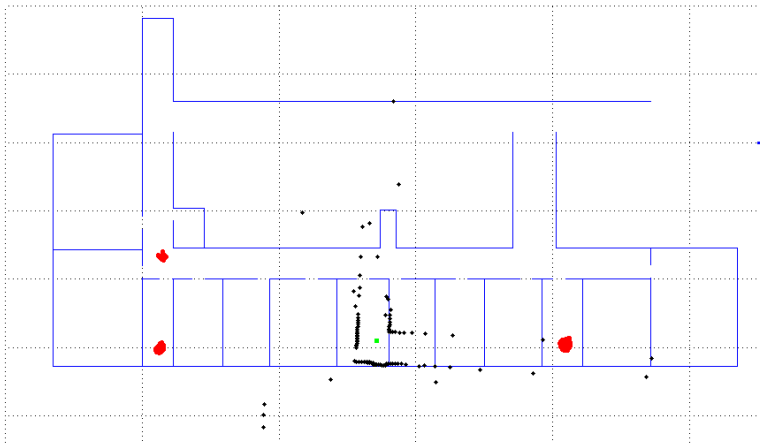


(a) (b)

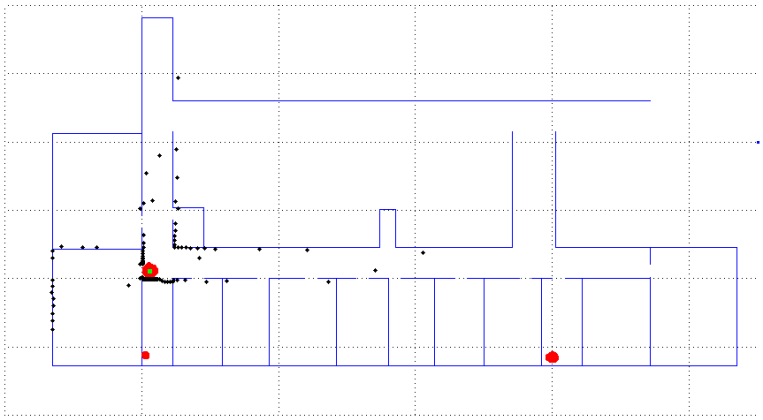
Figure 6.14: The average number of motion steps required for global localization using different number of samples in normal cases (a) and when worst cases are taken into consideration (b).



(a)



(b)



(c)

Figure 6.15: Global localization with 6000 samples. Red spots are samples, black are laser scan points and the green point is the estimated pose. At the start samples are distributed over the whole working area (a). The situation is still ambiguous after 15 steps (b). The robot has successfully localized itself after 19 motion steps (c).

6.5 Additional Experiments

The developed localization system has been tested in another indoor environment and outdoors as well. Furthermore, global localization has been run using samples initially generated at fixed reference locations in the environment rather than distributing them randomly over the whole working area of the robot. However, more experiments are necessary to confirm the results shown in the next subsections.

6.5.1 Position Tracking

The robot has been experimented in another indoor environment to track its position. The path of the test journey is shown in figure 6.16. The experiment took place at the ground floor of the building in which the Institute of Systems Engineering, Real-time Systems Group is located.

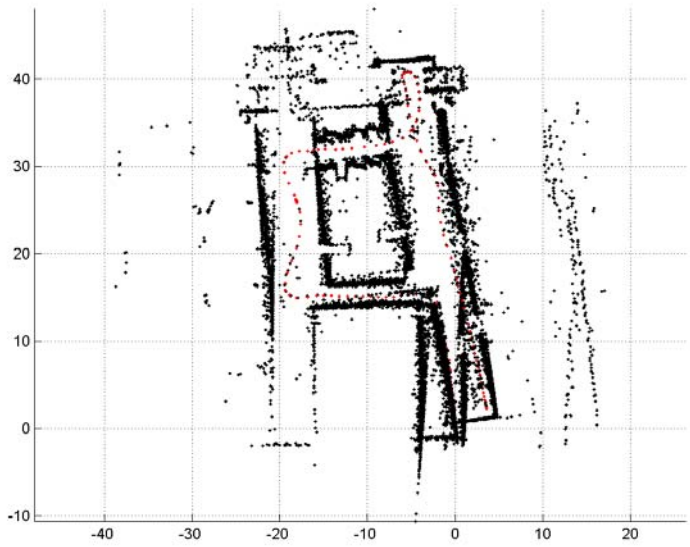


Figure 6.16: The test path is indicated by the red points.

The robot started moving at (0,0) and returned back to the same starting point. The accumulated translational and rotational errors at the end of the journey were 2.49 m and 12.36° respectively as reported by using only odometry information. When employing the developed MCL system, the translational and rotational errors were 0.42 m and 5.1° respectively. The reliability and robustness of the localization system can be traced in the snapshots shown in figure 6.17. Black points represent laser scans, red spots are the samples and the green square depicts the estimated position.

6.5.2 Outdoor Position Tracking

A line-feature map of the campus area at the University of Hannover has been obtained from local authorities. Fusing GPS data into the prediction phase of the MCL led to localization failures. This is due to the surrounding high buildings that reduced the quality of the received GPS information. However, localization was very successful if an approximate starting point is given and if depending only on odometry and laser data. Figure 6.18 depicts the localization process. A green square and a line out of it indicate the estimated position and orientation respectively. To have an idea of the GPS data quality, a black square and an outcoming line indicate GPS position and heading respectively. Laser scan data are marked with ‘x’ signs and samples are represented by red spots. Localization has been achieved using 400 samples. It is clear that the delivered GPS information was too inaccurate to depend on.

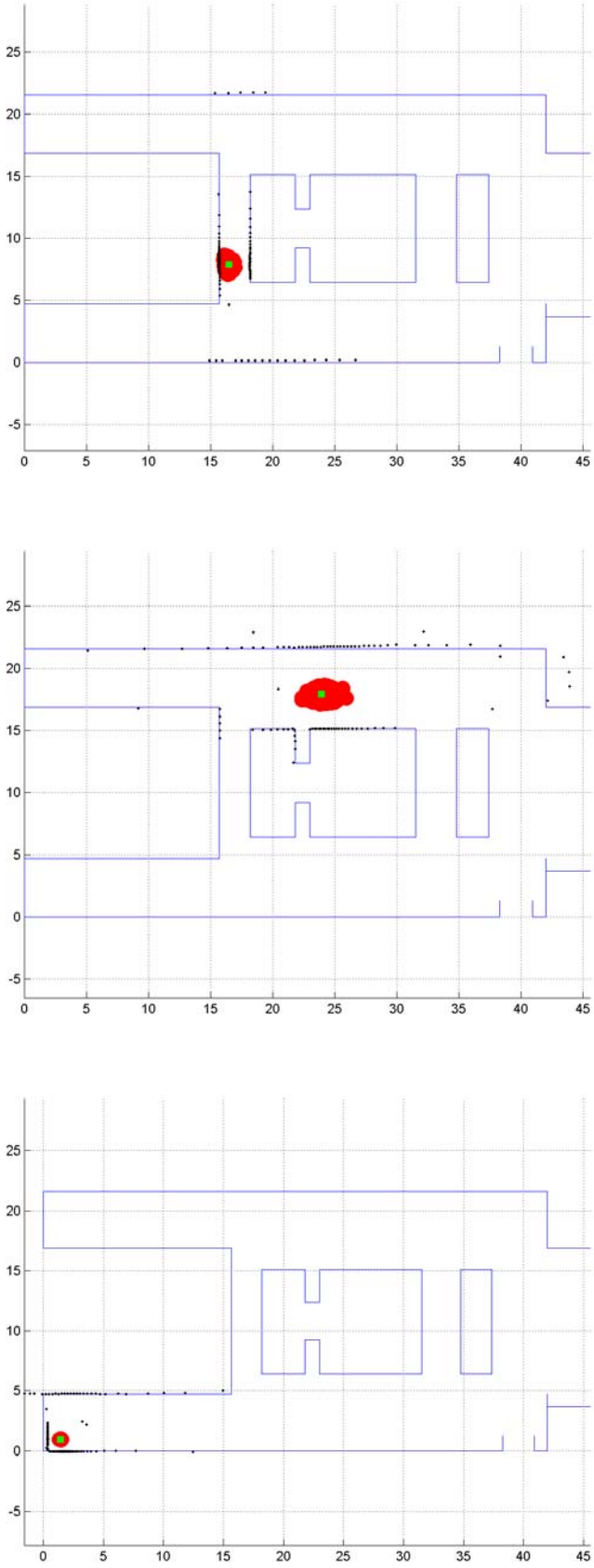


Figure 6.17: The developed localization system proved its robustness when employed in a new indoor environment.

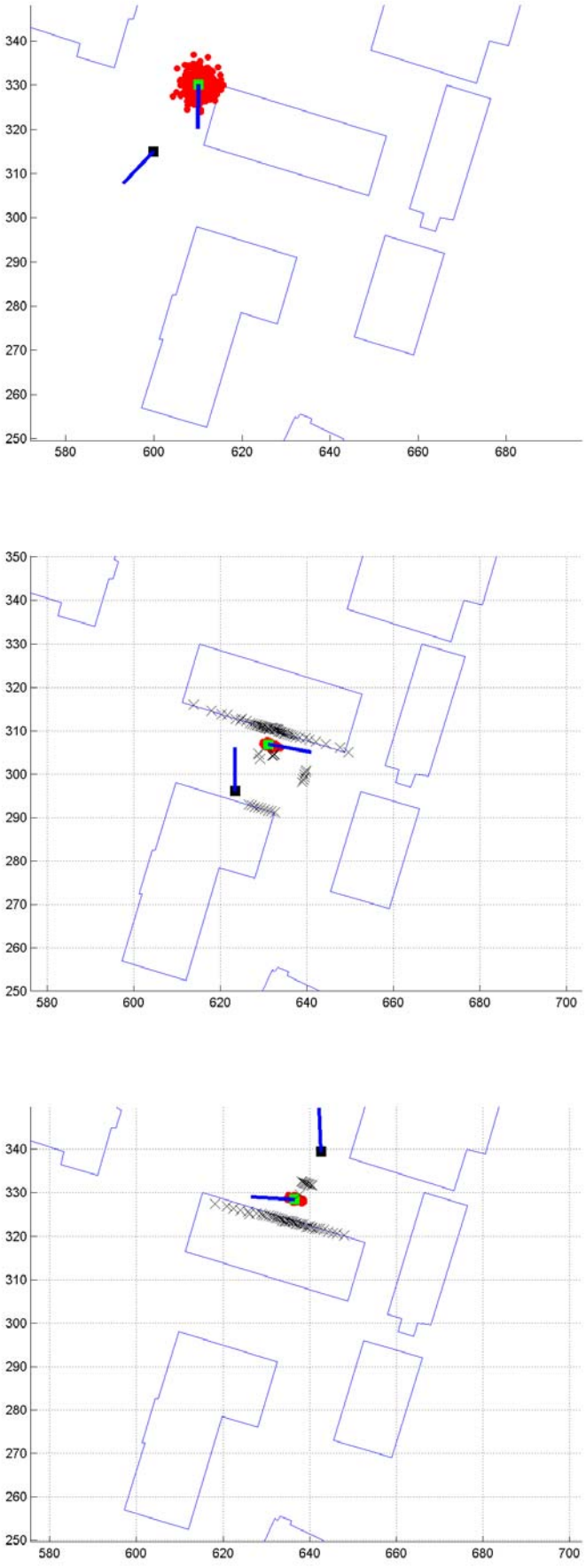


Figure 6.18: Position tracking at the campus of Hannover University.

6.5.3 Global Localization

To reduce the samples needed to achieve correct global localization, a different approach to generate the initial belief has been experimented. The initial belief of global uncertainty is usually represented by randomly distributed samples over the whole state space of the robot. In this approach, samples have been generated at some fixed reference locations in the state space. The number of reference locations used is 10. Simulations have been conducted using 400 and 200 samples, placing 40 or 20 samples at each location respectively. The orientation of one sample can take the value 0°, 90°, 180°or 270°. Experiments of both cases (using 400 and 200 samples) confirmed the capability of the developed MCL system to globally localize the robot using this approach. However, more tests should be run in order to judge this approach definitively. It should be experimented how this method would work with different starting positions and with different number of reference locations. Figure 6.19 illustrates the initial situation using the aforementioned approach with 400 samples.

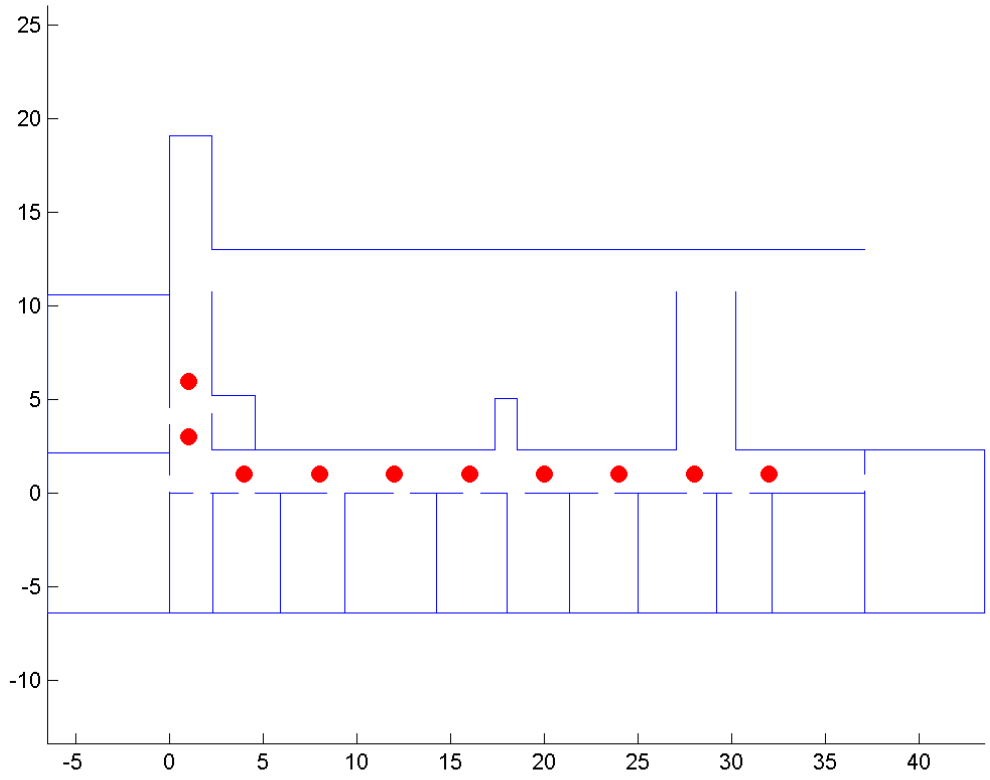


Figure 6.19: Initial situation of a global localization using 10 reference locations and 400 samples.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

A fundamental property for an autonomous robot is the capability to correctly localize itself in its working environment. This knowledge about location is the necessary background to fulfil the tasks assigned to the robot. This work has proved the feasibility of fusing data from a 3D laser scanner and odometry into the MCL for pose estimation in indoor environments.

Experiments have been conducted to confirm the successful employment of the developed 3D laser scanner based MCL. The main advantage of 3D laser scanning lies in the ability to extremely minimize the effects of dynamic obstacles. Furthermore, it concentrates perception on the invariant features of the environment, thus increasing the certainty in the belief generated by the MCL.

The MCL has many attractive properties. Firstly, it is very fast and simple. Secondly, it is to a large extent modular. That is, when changing the problem one need only change the expressions for the prediction and update phases in the code. This enables the efficient employment and integration of further sensor data. Finally it provides a complete description of the posterior distribution, not just a single point estimate.

To achieve good performance of MCL, a pessimistic noise model for 2D laser sensors is usually necessary. Tests have proven that this is not the case for 3D laser sensors. A pessimistic noise model performs well in position tracking to put up with very noisy

perception data, but for global localization MCL performs much better with certain sensor models. This is due to the high accuracy of the laser scanner used combined with a nearly complete model of the world's features.
The main drawback of MCL can be traced in global localization. If not enough samples are generated at the correct location, MCL fails. It should always be recalled that samples generation is completely random.

7.2 Future Work

It is only by building and testing real world systems that make new problems and issues arise. In this section some extensions to the developed localization system will be suggested.
The localization system developed in this work could be combined with occupancy grid maps for obstacle avoidance and path planning.
The possibility of outdoor deployment raises the interest in some open problems as already introduced in section 6.5.2. Integrating a GPS device into the MCL system is advisable due to its acceptable price. However, the quality of GPS data is questionable where high buildings may prevent seeing properly constellated satellites necessary for precise measurements. In such situations relaying only on GPS readings is very risky. Line-feature maps of the deployment area could be provided accompanied with an approximate start pose to assure robust localization. Self-localization and navigation in such scenarios still have to be thoroughly tested.
It is of uttermost importance to have mechanisms for fault detection. An autonomous system must be able to detect failures and preferably counteract them. The sooner the causing events are detected the easier it is to handle them. Examples are slippage when driving over a threshold or collisions in dense crowd. Adding some "salt and pepper" noise to the odometry model could represent these sources of error. The introduction of inertial sensor data to the developed localization system would greatly contribute to improving the performance in such situations especially when the robot is deployed in very large indoor or in outdoor environments.
Vision is the sensor that offers the largest potential source of information. Despite being influenced by, e.g. varying light intensities, valuable information is still to be extracted. In indoor global localization, a door offers so much more information than just the relative position to the robot. Typically there are signs and names in the vicinity of the door giving

strong evidence about the location. Buildings, trees and bushes could be classified into patterns that would be recognized during navigation outdoors.

The MCL could be modified to help the robot recover from sever localization failures. For example, random samples could be generated at some fixed reference locations in the environment every time the localization procedure is run. If the robot is lost, samples at some reference location will be assigned high importance factors causing MCL to take these samples into consideration, hence accelerating the recovery process. In section 6.5.3, this method is applied but only in the first step of a global localization.

A speech input/output system could be integrated to improve the interaction between the robot and humans. This speech system could decrease the time required to input instructions.

An important point that must be taken into consideration is the need for really long term experiments. The length of tests should be measured in days and weeks and not hours. This raises the problem of power supply. In [Gär2004] an automatic charging station that can be recognized by the robot has been developed to cope with this problem. Currently most robotic systems use batteries that limit their autonomy to between 2 and 5 hours.

References

[Bar2001] Bar-Shalom, Y.; Li, X.; Kirubarajan, T.: Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software, Wiley, New York, 2001.

[Bor1996] Borenstein, J.; Everett, H.R.; Feng, L.: Navigating Mobile Robots: Systems and Techniques, A.K. Peters, Wellesley, MA, 1996.

[Box1958] Box, G.E.P.; Muller, M.E.: A Note on the Generation of Random Normal Deviates, The Annals of Mathematical Statistics, 29 (1958).

[Cha1985] Chatila, R.; Laumond, J.-P.: Position Referencing and Consistent World Modeling for Mobile Robots, IEEE Intl. Conf. on Robotics and Automation (ICRA), 1985.

[Com1777] Comte de Buffon, G.: Essai d'Arithmetique Morale, Supplement a l'Histoire Naturelle, Vol. 4, 1777.

[Cox1990] Cox, I.J.; Wilfong, G.T. (Editors): Autonomous Robot Vehicles, Springer, Berlin, 1990.

[Cox1991] Cox, I.J.: Blanche: An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle, IEEE Transactions on Robotics and Automation, 7 (2), 1991.

[Dör1965] Dörrie, H.: 100 Great Problems of Elementary Mathematics: Their History and Solution, Dover Publications, New York, 1965.

[Dou2001] Doucet, A.; Freitas, N. de; Gordon, N. (Editors): Sequential Monte Carlo in Practice, Springer, New York, 2001.

[Dud2000] Dudek, G.; Jenkin, M.: Computational Principles of Mobile Robotics, Cambridge University Press, 2000.

[Fox1999] Fox, D.; Burgard, W.; Thrun, S.: Markov Localization for Mobile Robots in Dynamic Environments, J. Artificial Intelligence, Res. 11 (1999).

[Fox1999-2] Fox, D.; Burgard, W.; Dellaert, F.; Thrun, S.: Monte Carlo Localization: Efficient Position Estimation for Mobile Robots, American Association for Artificial Intelligence, 1999.

[Fox2001] Fox, D.; Thrun, S.; Dellaert, F.; Burgard, W.: Particle Filters for Mobile Robot Localization, in [Dou2001].

[Fox2003] Fox, D.: Adapting the Sample Size in Particle Filters Through KLD-Sampling, The International Journal of Robotics Research, 22(12), 2003.

[Fox2003-2] Fox, D.; Hightower, J.; Liao, L.; Schulz, D.; Borriello, G.: Bayesian Filtering for Location Estimation, IEEE Pervasive Computing, vol. 2, no. 3, 2003.

[Gär2004] Gärtner, Olaf: Entwicklung einer automatischen Ladestation für mobile Robotersysteme, Diploma Thesis, University of Hannover, 2004.

[Gut1998] Gutmann, J.-S.; Burgard, W.; Fox, D.; Konolige, K.: An Experimental Comparison of Localization Methods, Int. Conf. on Intelligent Robots and Systems (IROS), 1998.

[Gut2002] Gutmann, J.-S.; Fox, D.: An Experimental Comparison of Localization Methods Continued, Int. Conf. on Intelligent Robots and Systems (IROS), 2002.

[Ise2003] Isernhagen, H.: Entwicklung von Verfahren zur verzerrungsfreien Aufnahme von 3D-Laserscans mit fahrenden Robotern, Diploma Thesis, Unversity of Hannover, 2003.

[Jen2001] Jensfelt, P.: Approaches to Mobile Robot Localization in Indoor Environments, Ph.D. Dissertation, Royal Institute of Technology, Stockholm, 2001.

[Kal1960] Kalman, R.E.: A New Approach to Linear Filtering and Prediction Problems, Trans. ASME, J. Basic Engineering, 82 (1960).

[Kel1901] Lord Kelvin: Nineteeth Century Clouds Over the Dynamical Theory of Heat and Light, Phil. Matg., serie 6, 2, 1, 1901.

[Kri2003] Kristensen, S.; Jensfelt, P.: An Experimental Comparison of Localization Methods, the MHL Sessions, Int. Conf. on Intelligent Robots and Systems (IROS), 2003.

[Lap1886] Laplace, P.: Theorie Analytique des Probabilites, Oeuvres Completes de Laplace, Vol. 7, Livre 2 (365-366), Academie des Sciences, Paris, 1886.

[Liu2001] Liu, Jun S.: Monte Carlo Strategies in Scientific Computing, Springer, New York, 2001.

[Stu1908] Student: Probable Error of a Correlation Coefficient, Biometrika, 6, 302, 1908.

[Thr2000] Thrun, S.: Probabilistic Algorithms in Robotics, AI Magazine, 21(4), 2000.

[Thr2001] Thrun, S.; Fox, D.; Burgard, W.; Dellaert, F.: Robust Monte Carlo Localization for Mobile Robots, Artificial Intelligence, 128 (2001).

[Wul2003] Wulf, O.; Wagner, B.: Fast 3D Scanning Methods for Laser Measurement Systems, International Conference on Control Systems and Computer Science, vol. 1, Bucharest, Romania, July 2003.