**University of Alberta**

# Bayesian Calibration for Monte Carlo Localization

by

**Armita Kaboli**

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**

**in Computer Engineering**

Department of Electrical and Computer Engineering

Edmonton, Alberta

Fall 2007

*Your file Votre référence*
*ISBN: 978-0-494-33272-6*
*Our file Notre référence*
*ISBN: 978-0-494-33272-6*

# Abstract

Robot localization is the problem of estimating the position of a robot given its sensory observations, control actions and a map. There exist efficient approaches to this problem, for example, Kalman and extended Kalman filters and Monte Carlo localization. These approaches address the problem by using probabilistic models for the motion and sensing of the robot. The effectiveness of these approaches is naturally dependent on the accuracy of these models.

Finding good models, or calibration, traditionally involves long and error-prone measurements and manual tuning. Therefore, an automatic calibration technique, which is the subject of this research, is highly desirable.

This thesis takes a Bayesian approach to calibration. Expert knowledge about the models is encoded as a prior distribution in parameter space.
A new belief about parameters' distribution is inferred from data collected onboard the robot. Since analytical inference of this distribution is not feasible, a Markov Chain Monte Carlo algorithm is used to draw samples from this distribution. The effectiveness of our technique is demonstrated both in simulation and on a real robot.

# Contents


**1 Introduction** **1**

1.1 Motivation . . . . . 2

1.2 Approach . . . . . 2

1.3 Contributions . . . . . 3

1.4 Guide to the Thesis . . . . . 3

**2 Background** **5**

2.1 Mobile Robot Localization . . . . . 5

2.1.1 Monte Carlo Localization . . . . . 7

2.1.2 Particle Smoothing . . . . . 9

2.1.3 Parameterized Models . . . . . 10

2.2 Bayesian Approach to Parameter Estimation . . . . . 11

2.3 Markov Chain Monte Carlo Methods . . . . . 12

2.3.1 Markov Chains . . . . . 12

**3 Bayesian Calibration** **17**

3.1 Mathematical Derivation . . . . . 17

3.2 Algorithm . . . . . 19

3.3 Using the Posterior Samples . . . . . 21

**4 Experiments** **23**

4.1 Experiments in Simulation . . . . . 23

4.1.1 Model 1: Simple-Amigobot . . . . . 24

4.1.2 Model 2: Biased-Amigobot . . . . . 25

4.1.3 Model 3: Biased-FNP-Amigobot . . . . . 28

4.1.4 Calibration with Wrong Models . . . . . 30
4.2 Real Robot Experiments . . . . . 33
4.2.1 Model 1: Simple-AIBO . . . . . 33
4.2.2 Model 2: Biased-FNP-AIBO . . . . . 35

**5 Discussion** **38**
5.1 Discussion of Paradoxical the Results . . . . . 38
5.2 Number of Particles . . . . . 39
5.3 Incremental Calibration . . . . . 40
5.4 Long Training Data Set . . . . . 41

**6 Related Work** **44**
6.1 Calibration in Robotics . . . . . 44
6.1.1 Stronger and Stone . . . . . 44
6.1.2 Roy and Thrun . . . . . 45
6.1.3 Eliazar and Parr . . . . . 46
6.1.4 Milstein and Wang . . . . . 46
6.2 State Estimation . . . . . 47
6.2.1 Andrews . . . . . 47
6.2.2 Storvik . . . . . 48

**7 Conclusion and Future Work** **50**
7.1 Contributions . . . . . 50
7.2 Future Work . . . . . 51

# List of Figures


4.1 Simple-Amigobot posterior distribution . . . . . 26
4.2 Simple-Amigobot localization accuracy graph 1 . . . . . 26
4.3 Simple-Amigobot localization accuracy graph 2 . . . . . 27
4.4 Simple-Amigobot localization accuracy graph 3 . . . . . 27
4.5 Biased-Amigobot posterior distribution . . . . . 28
4.6 Biased-Amigobot localization accuracy graph . . . . . 29
4.7 Biased-FNP-Amigobot posterior . . . . . 30
4.8 Biased-FNP-Amigobot localization accuracy graph . . . . . 30
4.9 Training with wrong model 1 . . . . . 31
4.10 Training with wrong model 2 . . . . . 31
4.11 Simple Amigobot trained with Amigobot-Biased-FNP data . . 32
4.12 3d representation of the parameter space . . . . . 33
4.13 AIBO setup . . . . . 34
4.14 AIBO posterior distribution . . . . . 34
4.15 AIBO localization accuracy graph . . . . . 35
4.16 Biased-FNP-AIBO posterior distribution . . . . . 37
4.17 Biased-FNP-AIBO localization accuracy graphs . . . . . 37

5.1 Connection between the training and testing particle numbers 40
5.2 Gradual calibration . . . . . 41
5.3 Gradually calibrated model localization accuracy graph . . . . 42
5.4 Calibration with a very long training data set . . . . . 43

# List of Algorithms


1 Particle Filtering. . . . . 9
2 Particle Smoothing. . . . . 11
3 Gibbs Sampling. . . . . 15
4 Metropolis sampling. . . . . 16
5 Bayesian calibration . . . . . 20
6 Storvik's Algorithm. . . . . 49

# Chapter 1

# Introduction

Most robotic tasks require the robot to have accurate information about its location. Therefore, robot localization is one of the most fundamental problems that needs to be addressed. More formally, mobile robot localization is the problem of estimating the position of the robot in a global coordinate system, given a map of the environment, and the history of sensor and odometry readings. Significant efforts have been put to solve this problem and a variety of effective techniques have been developed.

Kalman Filters (KF) [8], Extended Kalman filters (EKF) [5], Unscented Kalman Filters (UKF) [6] and Monte Carlo Localization (MCL) [4] are all well known approaches to this problem. In situations that the uncertainty about the position of the robot can be modeled with a unimodal distribution, Kalman and Extended Kalman filter can be used effectively for localization. However, in some problems such as localization in an office environment, where many corners and hallways are similar, using a unimodal distribution can fail. Monte Carlo localization has proven to be effective in dealing with problems where multimodal distributions are required to model the position of the robot.

The mentioned methods usually use parametric probabilistic models in order to model the uncertainties involved in the problem. Dynamic environments, noisy sensors and also inaccurate actuators of robots are examples of sources of uncertainties. In the mentioned localization methods uncertainties are described through the motion and sensor models of the robot. These models usually contain some parameters that must be tuned to mimic the real

world as closely as possible in order to localize the robot accurately. Tuning these parameters is called calibration. Traditionally calibration is done manually. This dissertation aims to develop a method to automate the calibration process.

## 1.1 Motivation

Models used in localization are traditionally hand-tuned. Such manual tuning involves tedious and error prone measurements. It should also be redone often, since the physical properties of the robots can change with time, e.g. the inflation of the tires changes with the temperature and gradually decreases over, time as well as motor wear and increased sensor noise. The environmental conditions, e.g. surface friction, lighting, etc., are also always changing. Additionally, in order to facilitate the presence of the robots in our day to day life, robots need to be able to coexist with people in un-engineered public enivornments. This means that there will be many unexpected or unknown factors that make it hard to manually calibrate for. Moreover, manual calibration can make it difficult to field new sensors or locomotion mechanisms. The aforementioned problems inspired the development of a method to calibrate the models involved in robot localization automatically, described in this thesis.

## 1.2 Approach

The calibration method described in this thesis is based on Bayesian approach. First, the knowledge about the parameters involved in models is encapsulated in a prior distribution over parameter space. Then the robot is allowed to move in the environment in which it should be localized, collecting data. This data is used with a Markov Chain Monte Carlo (MCMC) sampling algorithm to draw samples from the posterior distribution of the parameters given the collected data. Finally, different techniques are proposed for incorporating the posterior parameter samples into Monte Carlo Localization.

## 1.3 Contributions

The contributions of this dissertation are two fold.

**Effective Bayesian calibration for Monte Carlo localization.** An efficient Markov Chain Monte Carlo sampling method is proposed to draw samples from the posterior distribution of model parameters given the data collected by the robot. The proposed method is completely general, unlike similar methods in literature, which are usually limited to certain types of models. The technique described in this thesis can be used to calibrate general robot models, such as commonly used multimodal laser range-finder models.

**A novel extension to Monte Carlo Localization.** A new extension to Monte Carlo localization is also proposed in order to utilize the samples of the posterior distribution of model parameters.

The basic algorithm and some of the results presented in this thesis appeared in the Proceedings of the National Conference on Artificial Intelligence (AAAI) [7].

## 1.4 Guide to the Thesis

This thesis is organized as follows.

**Chapter 2: Background.** In this chapter, the framework that the proposed technique is based on is introduced. After explaining Monte Carlo Localization, its connection to particle filtering and particle smoothing is demonstrated. The attention then moves to the parametric models that are used in MCL and parameter estimation techniques. Finally, Bayesian techniques are described, including two examples of sampling algorithms - Gibbs and Metropolis sampling.

**Chapter 3: The Algorithm.** The proposed calibration technique is presented in detail in this chapter. First the Bayesian formulation of the problem

is described, followed by detailed presentation of the proposed algorithm. Afterwards three methods are introduced in order to exploit the output samples of the calibration procedure.

**Chapter 4: Results.** The results section is divided into two main parts: experiments in simulation and experiments on real a robot. In each section the data collected from the robot is used to calibrate a number of models with increasing complexity. Graphs of both samples of the posterior distribution after calibration are presented along with the resulting localization accuracy. The results verify the effectiveness of the proposed calibration technique. This section also highlights the fact that the calibrated parameters outperform the true parameters, which is counter-intuitive and paradoxical. The analysis of these paradoxical results is deferred to the next chapter.

**Chapter 5: Discussion.** This chapter starts with a discussion of paradoxical observations from the results chapter, demonstrating that they are, in fact, sensible. We also investigate whether the calibration procedure might find parameters suited to a particular number of particles used in MCL. The major finding is that although the experiments do not completely confirm to intuition, they do show that calibration with a number of particles smaller than the number used in MCL results in more robust localization. Finally the chapter presents results from a number of other empirical investigations, all exploring the robustness of the proposed technique.

**Chapter 6: Related Literature.** In this chapter relation of the proposed to other work on automatic calibration is explained pointing out the superiority of the new method to similar work.

**Chapter 7: Conclusion.** This final chapter brings major conclusion of the thesis and presents a sketch of possible future work.

# Chapter 2

# Background

This chapter provides an introduction to the subjects that the proposed calibration technique builds upon. It starts with an overview of mobile robot localization [14] which is the framework that the calibration technique operates within. Monte Carlo Localization (MCL), the application of particle filtering to the localization problem, is presented in detail along with particle smoothing which is an extension of particle filtering for identifying a complete trajectory. This is followed by discussion of Bayesian parameter estimation, and Markov chain sampling methods. The proposed Gibbs and Metropolis sampling methods are specifically emphasized, since they constitute the core of the proposed algorithm.

## 2.1 Mobile Robot Localization

The position of a mobile robot in an environment is not usually measurable by the robot itself. Even a robot equipped with a GPS device does not know its exact position accurately. Therefore, robots have to infer their positions using other measurable quantities, which are the observations of their sensors and also knowledge about their control actions. The robots then keep an internal *belief* about their positions. In probabilistic robotics the robot's belief about its state at time $t$ is represented as a conditional probability density function over the position space given the observation and action history. More formally, one can write:

$$bel(x_t) = P(x_t|z_{1:t}, u_{1:t}) \tag{2.1}$$

where $x_t$ is the state vector. Often $x_t$ is a three dimensional vector, where the first two dimensions are the location coordinates, and the third one is the orientation of the robot. $u_t$ represents the control action, while $z_t$ represents the observation vector. The dimensionality of $z$ depends on the sensors of the robot. When 1:t appears as a subscript, it refers to the sequence of variables beginning at time 1 through time t, e.g., $z_{1:t} = z_1, z_2, \ldots, z_t$.

The sequence of observations and control actions, i.e., $z_{1:t}$ and $u_{1:t}$, is called data and written as simply $\mathcal{D}$. In order to be able to update a robot's belief as it moves, the robot needs to have a generation rule as well as a measurement or observation rule. These rules are also called motion and sensor models. Since the actuators and sensors of robots are noisy, the rule cannot be represented as a deterministic function. Therefore, the motion and sensor models are represented as conditional probability density functions.
The motion model :

$$P(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1}) = P(x_t|x_{t-1}, u_t) \tag{2.2}$$

The sensor model :

$$P(z_t|x_{1:t}, u_{1:t}, z_{1:t}) = P(z_t|x_t) \tag{2.3}$$

In the motion model, $x_{t-1}$ and $u_t$ are assumed sufficient statistics of all previous control actions and observations, allowing to write the model in the simple form on right-hand side of Equation 2.2. In the sensor model the same assumption holds about $x_t$. This assumption is known as the Markov property. One approach to robot localization is recursive Bayes filtering, which computes the posterior distribution given the robot's observations and control actions. It does the posterior computation recursively, i.e. it calculates $bel(x_t)$ from $bel(x_{t-1})$ and the most recent observation and control action. In general, the posterior distributions do not have a closed form making recursive computation infeasible. Approximations to the posterior distribution are typically employed. For example, Kalman filters approximate the posterior as normal distribution while particle filters approximate it with a set of samples. In the next section, the application of particle filters to robot localization, typically called Monte Carlo localization, is described.

### 2.1.1 Monte Carlo Localization

Monte Carlo Localization is the application of particle filtering in robot localization.

#### Particle Filtering

Particle filtering approximates the current belief with a finite set of samples, or particles. The recursive update results in a new set of particles using importance sampling followed by resampling.

**Importance sampling.** Importance sampling is a method for computing expectations of a distribution using Monte Carlo sampling of a different distribution. This method is usually used when there is no easy way to sample from the target distribution. Assume that $f(x)$ is the distribution for which expectations are to computed. Importance sampling instead draws samples from an auxiliary probability density function, or candidate distribution, $g(x)$ and associates each of the samples with a weight equal to the value of $f(x)/g(x)$. The only condition that $g(x)$ must fulfill is that it must not be zero for x's where f(x) is non-zero. The better $g(x)$ approximates $f(x)$, the closer are the samples to the true distribution, and the more accurate the computed expectations. It can easily be shown that the expectation of the generated samples using this method will converge to the true expectation of the distribution as the number of samples increases. Suppose $a(X)$ is a function of the random variable $X$ and wants to compute $E[a(X)]$.

$$E_f[a(X)] = \int_X a(x)f(x) = \int_X a(x)\frac{f(x)}{g(x)}g(x) = \tag{2.4}$$

$$E_g[a(X)\frac{f(X)}{g(X)}] \approx \sum_{x^{(i)}\sim g} a(x^{(i)})\frac{f(x^{(i)})}{g(x^{(i)})} \tag{2.5}$$

**Derivation.** Mathematical derivation for particle filtering using importance sampling follows. Samples of $bel(x_t)$ can be obtained by generating trajectory samples from $bel(x_{1:t})$ and discarding the earlier samples. Therefore, replacing

$bel(x_{1:t})$ with $bel(x_t)$ does not change the problem.

$$
\begin{aligned}
bel(x_{1:t}) &= P(x_{1:t}|z_{1:t}, u_{1:t}) && (2.6)\\
&= ZP(z_t|x_{1:t}z_{1:t-1}, u_{1:t})P(x_{1:t}|z_{1:t-1}, u_{1:t}) && (2.7)\\
&= ZP(z_t|x_t)P(x_{1:t}|z_{1:t-1}u_{1:t}) && (2.8)\\
&= ZP(z_t|x_t)\hat{bel}(x_{1:t}) && (2.9)
\end{aligned}
$$

where $\hat{bel}(x_{1:t})$ is the the belief after taking the control action at time $t$ but before receiving the observation.

$$
\begin{aligned}
\hat{bel}(x_{1:t}) &= ZP(z_t|x_t)P(x_t|x_{1:t-1}u_{1:t})P(x_{1:t-1}|z_{1:t-1}, u_{1:t-1}) && (2.10)\\
&= ZP(z_t|x_t)P(x_t|x_{t-1}, u_t)P(x_{1:t-1}|z_{1:t-1}, u_{1:t-1}) && (2.11)\\
&= ZP(z_t|x_t)P(x_t|x_{t-1}, u_t)bel(x_{1:t-1}) && (2.12)
\end{aligned}
$$

where $Z$ is a normalization constant.

Equation 2.7 follows 2.6 because of the Bayes' rule, and 2.8 is a result of Markov property. Equation 2.10 follows 2.8 applying chain rule and 2.11 is a result of Markov property. The inductive Equation 2.12 shapes the recursive basis of the particle filtering algorithm. Suppose that there is a set of particles sampled from $bel(x_{1:t-1})$, and one wants to draw samples from $bel(x_{1:t})$ distribution. However, it is not generally easy to draw samples from this distribution. Therefore, using the importance sampling technique, samples from $\hat{bel}(x_{1:t})$ can be drawn instead. The importance sampling correction, as can be seen from Equation 2.9 is then just $P(z_t|x_t)$, i.e., the sensor model. As can be seen in Equation 2.12, sampling from $\hat{bel}(x_{1:t})$ involves using the recursive samples from the previous time-step and then sampling from the motion model, all of which are tractable. Now there is a set of particles each associated with a weight, i.e., $(x_{1:t}^{(i)}, w_t^{(i)})$, which is a discrete representation of $bel(x_{1:t})$ distribution. The particles are then resampled with probability proportional to the weights, and as a result the new set of particles will actually be drawn from the target distribution. The initial set of particles is assumed as sampled from the prior $P(x_0)$.

**Algorithm.** Algorithm 1 summarizes the particle filtering algorithm. The derivation involves sampling a complete trajectory, but usually all one desires to know is the posterior distribution of the robot's current location. The presented algorithm drops the history of states and the particles only store the current time-step. If samples from the complete trajectory are to be drawn, the samples of the previous time steps can be kept. Connecting each particle to its parent particle will give a set of samples from the trajectory. However, since the resampling step in this algorithm causes high levels of degeneracy in the trajectory over time, generated trajectory samples are not favorable. Particle smoothing is the usual method that is used to generate samples of the trajectory.

**Algorithm 1** Particle Filtering.

1. Given parameters $X_{t-1}$, $z_t$, $u_t$
2. for i = 1 to N
   (a) Draw particle $i$ from the motion model:
   $$x_t^{(i)} \sim P(x_t|x_{t-1}, u_t)$$
   (b) Give the $i$th particle a weight proportional to the sensor model value:
   $$w^{(i)} = P(z_t|x_t)$$
3. for k = 1 to N
   (a) Resample: draw $i$ with probability $\propto w^{(i)}$
   (b) add $< x_t^{(i)}, w^{(i)} >$ to $X_t$
4. return $X_t$

### 2.1.2 Particle Smoothing

Particle smoothing [14] is also a recursive procedure that is applied backward to the output particles of the particle filtering algorithm. The idea behind particle smoothing is to extract the most probable trajectory through the available particles, instead of just connecting the parent points to their children.

Mathematical derivation of the particle smoothing algorithm follows:

$$
\begin{aligned}
x_{1:T} &\sim P(x_{1:T}|u_{1:T}, z_{1:T}) && (2.13)\\
P(x_{1:T}|u_{1:T}, z_{1:T}) &= P(x_T|u_{1:T}, z_{1:T}) \prod_{t=1}^{T-1} P(x_t|x_{t+1:T}, u_{1:t}, z_{1:t}) && (2.14)\\
x_t &\sim P(x_t|x_{t+1:T}, u_{1:t}, z_{1:t})\\
&= P(x_t|x_{t+1}, u_{1:t}, z_{1:t}) && (2.15)\\
&= ZP(x_{t+1}|x_t, u_t)P(x_t|u_{1:t}, z_{1:t}) && (2.16)\\
&= ZP(x_{t+1}|x_t, u_t)bel(x_t) && (2.17)
\end{aligned}
$$

Rewriting the posterior as in Equation 2.14, one can realize that all that is needed to be done is to recursively draw samples from the distribution $P(x_t|x_{t+1:T}, z_{1:t}, u_{1:t})$. Following the Bayes' rule, the distribution can be replaced with the expression in Equation 2.17. Applying importance sampling to Equation 2.17, one can draw samples from $bel(x_t)$ with probability proportional to importance sampling's correction term. Since $bel(x_t)$ is already sampled by particle filtering step, only resampling from the particle set with probability proportional to $P(x_{t+1}|x_t, u_t)$ is necessary. Algorithm 2 presents the particle smoothing algorithm.

### 2.1.3 Parameterized Models

Up to this point, the motion and sensor models have not been made explicit. Although the exact form of the models will depend upon the particular robot and application, it is common to consider a family of models parameterized by some vector $\theta$. Assume that the form of the model is known, but the model parameters are not and these need to be calibrated. Thus, from now on, the motion and sensor model will be written as follows:

- Motion model : $P(x_t|x_{t-1}, u_t, \theta)$
- Sensor model : $P(z_t|x_t, \theta)$

In all of the above algorithms, there is an implicit conditioning on the vector $\theta$.

**Algorithm 2** Particle Smoothing.

1. Given $X_{1:T}$, $W_{1:T}$, $u_{1:T}$
2. For $j = 1$ to $N$ (N samples from the trajectory):
   (a) Draw $i$ with probability $\propto w_T^{(i)}$

$$\bar{x}_T \leftarrow x_T^{(i)}$$

   (b) for $t = T - 1$ to 1
      i. Recalculate the weights of previous step particles:

$$w_t^{(i)} = P(\bar{x}_{t+1}|x_t^{(i)}, u)w_t^{(i)}, i = 1 \ldots N$$

      ii. Resample: draw $i$ with probability $\propto w_t^{(i)}$
      iii. $\bar{x}_t \leftarrow x_t^{(i)}$
   (c) $X_{1:T}^{(j)} \leftarrow (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_T)$
3. Return $X_{1:T}$

## 2.2 Bayesian Approach to Parameter Estimation

The goal of this thesis is to calibrate the motion and sensor models used in Monte Carlo localization. This problem is generally called *parameter estimation.* A Bayesian approach is adapted to solve the problem of parameter estimation. Bayesian parameter estimation, or Bayesian inference, usually consists of the following steps:

1. Considering $\theta$, the unknown parameters, as a random variable
2. Defining a probability density function over the parameter space, i.e. a *prior distribution* $P(\theta)$, which expresses the belief about the parameters without seeing any data.
3. Obtaining a new belief about $\theta$ after observing a sequence of data, i.e., $\mathcal{D}$. This is basically calculating the *posterior distribution* $P(\theta|\mathcal{D})$

However, in many problems, analytical calculation of the posterior distribution is impossible. In these cases different methods can be used in order

to approximate the posterior distribution. The approach selected here is to draw samples from the posterior distribution using Markov Chain Monte Carlo (MCMC) methods that will be explained in the following sections [15].

## 2.3 Markov Chain Monte Carlo Methods

The best way to get accurate estimates to the expectations of interest, is to continuously search for the regions of high probability. Monte Carlo methods based on Markov Chain theory are methods that have this property. In the following section the theory of Markov chains, in extent necessary to develop MCMC sampling, is presented.

### 2.3.1 Markov Chains

A Markov chain is a representation of a stochastic process, $X^{(t)}$, where $t$ is often interpreted as time. In a Markov chain the distribution of $X$ in time $t+1$ is entirely specified by the distribution of $X$ in time $t$. More specifically :

$$P(X^{(t+1)} = x^{t+1} | X^{(t)} = x^{(t)}, ..., X^{(1)} = x^{(1)}) = P(X^{(t+1)} = x^{t+1} | X^{(t)} = x^{(t)}) \tag{2.18}$$

This is called Markov property. A Markov chain is specified by the initial distribution of $X^{(0)}$, $p_0(x)$, and the transition probabilities of one state to another state, which is the conditional distribution of $X^{(t+1)}$ given the possible values for $X^{(t)}$. From now on, the probability of a transition from state $x$ to $x'$ at time $n$ will be designated as $T_n(x, x')$. If transition probabilities do not depend on time, the chain is called homogeneous or stationary and the transition probabilities will be represented as $T(x, x')$. The most important characteristic of Markov chains is that certain Markov chains converge to a unique invariant distribution. The invariant distribution of a Markov chain is defined as a distribution over the states of the chain that persists forever, when it is reached. To sample from a complex distribution, a Markov chain is built by specifying the transition probabilities, whose invariant distribution is the distribution of interest. The transition probabilities are usually defined by

a set of *base* transition probabilities represented by $B_k$ , each of which leaves the desired distribution unchanged.

More formally the invariant distribution, $\pi(x)$ satisfies:

$$\pi(x) = \sum_{x'} \pi(x')T(x', x), \tag{2.19}$$

or, in matrix form:

$$\pi = \pi T \tag{2.20}$$

$$T = B_1 \dots B_n \tag{2.21}$$

The invariant distribution is the eigenvector of Matrix $T$ with eigenvalue of 1. For most stochastic matrices a complete set of linearly independent eigen vectors can be found. The eigenvectors corresponding to eigenvalue of 1 can be chosen in a way so that all the elements of the vector are real and sum to one. Of special interest are in the transition matrices that have just one eigenvector corresponding to eigenvalue 1.

A Markov chain satisfies the detailed balance condition, if the probability of being in state $x$ and moving to state $x'$ is equal to the probability of being in state $x'$ and transition to state $x$. It is easy to show that the invariant distribution of such a chain exists.

A Markov chain is said to be ergodic if the probability of transitioning from any state to another possible state is greater than zero. It can be proven that if a Markov chain is ergodic, it will always converge to an invariant distribution regardless of the initial choice of the probability distribution $p_0(x)$. These two properties will be used while constructing Markov chains based on Gibbs and Metropolis methods, in order to prove that the chain's invariant distribution is our desired distribution [10].

### Gibbs Sampling Method

The Gibbs algorithm is conceptually the simplest of Markov Chain Monte Carlo sampling methods and is appropriate for problems where the variables have a small finite range or have conditional distributions of a form that are easy to sample from. Suppose that a set of random variables $X =$

$(X_1, \ldots, X_n)$ is given and samples are to be drawn from their joint distribution $P(X_1, \ldots, X_n)$. In order to simulate this distribution, Gibbs sampling method repeatedly selects one of the variables either randomly or in a special order. Then, keeping other variables unchanged, it replaces the selected variable with a value drawn from its conditional distribution given the values of other variables. This process is identical to building a Markov chain with a set of base transition probabilities $B_k$ for $k = 1, \ldots, n$ with the following definition:

$$B_k(x, x') = P(x'_k | \{x_i : i \neq k\}) \prod_{i \neq k} \delta(x_i, x'_i) \tag{2.22}$$

where $\delta(x, x')$ is a function taking on 1 if and only if $x = x'$ (otherwise 0). The base transition are usually assumed to be applied in sequence, however they can be picked randomly. It can be easily shown that each of the base transition probabilities leaves the invariant distribution unchanged. Therefore, any combination of the base transition probabilities will not change the joint distribution. It can also be shown that the chain is ergodic, regardless of whether transitions are chosen sequentially or randomly, as long as the conditional distributions are non-zero everywhere. Thus, the chain converges to the invariant distribution, which is the distribution of interest. Gibbs sampling algorithm relies on the assumption that sampling from the conditional distributions is feasible. However, in many problems this assumption is not valid. Metropolis algorithm is an alternative which builds upon looser assumptions. Algorithm 3 summarizes the Gibbs algorithm [10].

### Metropolis Sampling Method

The Metropolis algorithm is similar to the Gibbs sampling method, however it is more general and it does not require sampling from conditional distributions, which may be difficult. The only assumption that is made in this method is that the joint density function should be calculable. Suppose that samples are to be drawn from the joint distribution of $P(X_1, \ldots, X_n)$ where $X = X_1, \ldots, X_n$ are all random variables, either discrete or continuous. The Metropolis algorithm repeatedly leaves all the variables but one unchanged on

**Algorithm 3** Gibbs Sampling.

1. Initialize:

$$\begin{aligned} x &= X^{(0)} = (x_1, \ldots, x_n) \\ k &= 1 \\ S &= \oslash \end{aligned}$$

2. while (1)

   (a) Sample from conditional distribution:

$$x'_k \sim P(x_k | x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n)$$

   (b) $x = (x_1, \ldots, x'_k, \ldots, x_n)$

   (c) add $x$ to $S$

   (d) $k = (k+1) \bmod n$

3. return $S$

each iteration. If the $k$th variable is to be changed, a new candidate value is sampled from a *proposal distribution* $S_k(x, x'_k)$. The new value is kept with the acceptance probability $A(x, x')$, otherwise it is left unchanged as well. The acceptance function can have many forms, such as the one that used in the proposed algorithm :

$$A(x, x') = \min\left(1, \frac{P(x')}{P(x)}\right) \tag{2.23}$$

The mentioned process can be translated to a Markov chain with base transitions defined as follows:

$$\begin{aligned} B_k(x, x') &= S_k(x, x'_k) A(x, x') \prod_{i \neq k} \delta(x_i, x'_i) \\ &+ \delta(x, x')[1 - \sum_{\tilde{x}} S_k(x, \tilde{x}_k) A(x, \tilde{x}) \prod_{i \neq k} \delta(x_i, \tilde{x}_i)] \end{aligned} \tag{2.24}$$

where:

$$S_k(x, x'_k) = S_k(x', x_k) \text{ when } x_i = x'_i \text{ for all } i \neq k \tag{2.25}$$

It can easily be shown that the detailed balance property holds for all the chains that fulfill the above condition. Also the chain will be ergodic as long as $S_k(x, x'_k)$ is non-zero for all $x'_k$. These two conditions guarantee the convergence

of the chain to the joint distribution. Algorithm 4 summarizes the Metropolis algorithm [10].

**Algorithm 4** Metropolis sampling.

1. Initialize:

$$\begin{aligned} x &= X^{(0)} = (x_1, \ldots, x_n) \\ k &= 1 \\ S &= \oslash \end{aligned}$$

2. while (1)
   (a) Draw sample from the *proposal distribution:*

$$x_k^* \sim S_k(x, x_k)$$

   (b) $x = x^*$ with the probability of $A(x, x^*)$
   (c) add $x$ to $S$
   (d) $k = (k + 1) \bmod n$
3. return $S$

# Chapter 3

# Bayesian Calibration

In this chapter a new approach to calibration problem for Monte Carlo Localization is presented. It starts with a formal mathematical derivation of the approach, followed by presentation of the algorithm itself. Finally, different methods that can be used to exploit the output of our algorithm in MCL are discussed.

## 3.1 Mathematical Derivation

As explained in the previous chapter, the motion and the observation models are the key elements in Monte Carlo localization and can be represented as follows:

$$\text{Motion model}: \quad P(x_t|x_{t-1}, u_t, \theta) \tag{3.1}$$

$$\text{Sensor model}: \quad P(z_t|x_t, \theta) \tag{3.2}$$

As mentioned before, $\theta$ is the parameter vector to be calibrated, which is considered a random variable in Bayesian approach. The goal is to calculate the posterior distribution of $\theta$ given the data collected from the robot, which consists of a history of the observation and control action vectors. More formally, the following distribution should be characterized:

$$P(\theta|\mathcal{D}) = P(\theta|z_{1:T}, u_{1:T}) \tag{3.3}$$

Since this distribution does not have a simple closed form, it can be approximated by generating a set of samples from it. More formally :

$$\theta^{(i)} \sim P(\theta|z_{1:T}, u_{1:T}) \tag{3.4}$$

Marginalizing over the unknown trajectories one can instead draw samples from the joint distribution in Equation 3.6:

$$P(\theta|z_{1:T}, u_{1:T}) = \int_{\tilde{X}_{1:T}} P(\theta, \tilde{X}_{1:T}|z_{1:T}, u_{1:T}) \mathrm{d}\tilde{X}_{1:T} \tag{3.5}$$

$$(\theta^{(i)}, X_{1:T}^{(i)}) \sim P(\theta, \tilde{X}_{1:T}|z_{1:T}, u_{1:T}) \tag{3.6}$$

As explained in the previous chapter, the Gibbs sampling algorithm guarantees that iterating between the following two steps, will eventually result in drawing samples from the joint distribution of $P(\theta, \tilde{X}_{1:T}|z_{1:T}, u_{1:T})$.

$$x_{1:T}^{(i)} \sim P(\tilde{X}_{1:T}|z_{1:T}, u_{1:T}, \theta) \tag{3.7}$$

$$\theta^{(i)} \sim P(\theta|\tilde{X}_{1:T}, z_{1:T}, u_{1:T}) \tag{3.8}$$

Equation 3.7 requires to draw samples from the posterior distribution of complete trajectories given the observations, control actions and a specific parameter value. Recalling from the background chapter, one can realize that this is the operation that particle smoothing algorithm performs. Unfortunately, the sampling operation of Equation 3.8 is not feasible. From the Markov chains' perspective, by performing particle smoothing a Markov chain is built with a set of base transitions, that leaves the distribution unchanged. However, the chain is not definitely ergodic without the second step of the Gibbs algorithm. Thus, more transition rules must be added to the chain.

Going back to the joint distribution, $P(\theta, x_{1:T}|\mathcal{D})$, an alternative approach

is to break down the distribution applying chain and Bayes rules:

$$
\begin{aligned}
& P(\theta, x_{1:T}|\mathcal{D}) \\
&= P(x_{1:T}, z_{1:T}|\theta, u_{1:T})P(\theta)/Z && (3.9)\\
&= \frac{P(\theta)}{Z}\prod_{t=1}^{T} P(x_t, z_t|x_{1:t-1}, z_{1:t-1}, u_{1:T}, \theta) && (3.10)\\
&= \frac{P(\theta)}{Z}\prod_{t=1}^{T} \begin{pmatrix} P(x_t|x_{1:t-1}, z_{1:t-1}, u_{1:T}, \theta) \\ P(z_t|x_{1:t}, z_{1:t-1}, u_{1:T}, \theta) \end{pmatrix} && (3.11)\\
&= \frac{P(\theta)}{Z}\prod_{t=1}^{T} P(x_t|x_{t-1}, u_t, \theta)P(z_t|x_t, \theta), && (3.12)
\end{aligned}
$$

As can be observed in Equation 3.12, it is possible to write the distribution as a product of prior, motion and sensor model density functions and a normalizing factor $Z$. As explained in the previous chapter, the only necessary condition for employing the Metropolis algorithm for sampling from a distribution, is the ablility to calculate the distribution function up to an unknown factor. Consequently, the Metropolis algorithm is applicable to the problem.

Returning to the Markov chain's perspective, more transition rules must be added to the chain. These transitions will make the chain ergodic as well as keep its distribution unchanged.

## 3.2 Algorithm

As explained in the previous section, samples can be drawn from the conditional distribution of $P(\tilde{X}_{1:T}|z_{1:T}, u_{1:T}, \theta)$ and the joint density function $P(\theta, \tilde{X}_{1:T}|z_{1:T}, u_{1:T})$ can efficiently be evaluated. Therefore, a hybrid MCMC technique can be employed to draw samples from the target distribution. In one step, i.e. inference, a sample of a complete trajectory is drawn using particle smoothing, keeping the parameters unchanged. In the next step, i.e. estimation, the Metropolis algorithm is applied, using a proposal distribution and formulating an acceptance function based on the joint distribution. More specifically, small Gaussian perturbations are applied in turn to one of the dimensions of the parameter vector, and the new state is then accepted with the probability of

the acceptance function, defined as follows:

$$A(\theta, \hat{\theta}) = \min\left(1, \frac{P(\hat{\theta}, x_{1:T}|\mathcal{D})}{P(\theta, x_{1:T}|\mathcal{D})}\right) = \min\left(1, \frac{P(\hat{\theta})}{P(\theta)} \prod_{t=1}^{T} \frac{P(x_t, |x_{t-1}, u_t, \hat{\theta}) P(z_t|x_t, \hat{\theta})}{P(x_t, |x_{t-1}, u_t, \theta) P(z_t|x_t, \theta)}\right). \tag{3.13}$$

Algorithm 5 summarizes this algorithm.

**Algorithm 5** Bayesian calibration

The parameter $N$ is the number of steps of change after burn-in and $m$ is the number of Metropolis iterations per Gibbs step.

1. Given training data $\mathcal{D}$ and parameters $N$, $m$.
2. Initialize Markov chain.

$$\theta^{(0)} \sim P(\theta)$$

3. For $i \leftarrow 1$ up to $(N+T)$:
   (a) Sample a trajectory.

$$x^{(i)} \leftarrow \text{PARTICLESMOOTHING}(z_{1:T}, u_{1:T}, \theta^{(i-1)})$$

   (b) Sample model parameters.

$$\theta^{(i)} \leftarrow \theta^{(i-1)}$$

   Repeat $m$ times: For $j \leftarrow 1$ up to $d$:
      i. Generate candidate.

$$\begin{aligned} \delta_k &\leftarrow \text{SAMPLENORMAL}(0, \sigma_k^2) \\ \hat{\theta}_k &\leftarrow \theta_k^{(i)} + \begin{cases} \delta_k & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

      ii. Accept candidate.

$$\begin{aligned} p &\leftarrow \text{SAMPLEUNIFORM}(0, 1) \\ \theta^{(i)} &\leftarrow \begin{cases} \hat{\theta} & \text{if } p < \min\left(1, \frac{P(\hat{\theta}, x_{1:T}|\mathcal{D}_{1:T})}{P(\theta, x_{1:T}|\mathcal{D}_{1:T})}\right) \\ \theta^{(i)} & \text{otherwise} \end{cases} \end{aligned}$$

4. Return samples $(\theta^{(i)}, x^{(i)})$ for $i = (T+1) \ldots (T+N)$.

Step 3(a) of Algorithm 5, the inference step, is computationally much more demanding than the estimation step. It requires $2Tn$ evaluations of the motion and sensor model density functions (where $n$ is the number of particles and $T$ is the length of the calibration trajectory), while the estimation step needs $dT$ evaluations (where $d$ is the dimensionality of the parameter vector). For each inference step the estimation step is repeated, which takes a shorter time, $m$ times in order to increase the simulation rate and speed up the convergence. Indeed, the value of $m$ is much smaller than $n$.

## 3.3 Using the Posterior Samples

In the previous section, a method was presented to generate a set of samples from the posterior distribution of the parameter vector, given the collected data. These samples should now be used in Monte Carlo localization. The simplest approach is to extract a single suitable parameter vector from the set of samples. The most straightforward choice is to simply calculate the mean of the samples:

$$\theta_{\text{mean}} = \sum_{i=1}^{N} \theta^{(i)}/N. \tag{3.14}$$

Another simple method is to choose the maximum a posteriori (MAP) sample from the Markov chain,

$$\theta_{\text{MAP}} = \underset{i=\{1,\ldots,n\}}{\text{argmax}}\, P(\theta^{(i)}, x_{1:T}^{(i)}|\mathcal{D}), \tag{3.15}$$

using Equation 3.12. The accuracy results using both methods are presented in the next chapter. However, neither of the two mentioned methods are properly Bayesian, making use of the uncertainty inherent in the posterior samples. A more Bayesian approach is to construct posterior motion and sensor models. A fully Bayesian approach, though, would result in non-Markov models since the next state and observation would no longer be independent of history. As the calibration method used a batch approach, the posterior models are based only on the training data, ignoring new observations. Essentially, the model parameters are assumed as conditionally independent of new data given the

training data (this is called *batch assumption* in the derivation below). As the size of the training data set increases, this assumption becomes more and more justified.

**Posterior Motion Model.** The posterior motion model is derived by integrating over the unknown model parameters,

$$\begin{aligned} & P(x_t|x_{t-1}, u_t, \mathcal{D}) \\ &= \int_\theta P(x_t|x_{t-1}, u_t, \mathcal{D}, \theta) P(\theta|x_{t-1}, u_t, \mathcal{D}) && (3.16) \\ &= \int_\theta P(x_t|x_{t-1}, u_t, \theta) P(\theta|\mathcal{D}), && (3.17) \end{aligned}$$

where Equation 3.17 follows from conditional independence of the training data given the model parameters and the batch assumption. Samples can be drawm from this distribution by simply sampling $\theta \sim P(\theta|\mathcal{D})$ and then sampling $x_t \sim P(x_t|x_{t-1}, u_t, \theta)$. Since $\theta^{(i)}$ are samples from the model parameter posterior, approximate sampling can be achieved by choosing $i \in \{1, \ldots N\}$ with uniform probability and then drawing the sample of $x_t$ using the parameters $\theta^{(i)}$.

**Posterior Sensor Model.** For the posterior sensor model, a posterior can be derived by integrating over the unknown model parameters,

$$\begin{aligned} P(z_t|x_t, \mathcal{D}) &= \int_\theta P(z_t|x_t, \mathcal{D}, \theta) P(\theta|x_t, \mathcal{D}) && (3.18) \\ &= \int_\theta P(z_t|x_t, \theta) P(\theta|\mathcal{D}), && (3.19) \end{aligned}$$

using conditional independence and the batch assumption, as was done with the posterior motion model. This can be approximated using a Monte Carlo estimate with samples of $\theta$ drawn from the posterior distribution. Since $\theta^{(i)}$ are samples from the posterior,

$$P(z_t|x_t, \mathcal{D}) \approx \sum_{i=1}^{N} \frac{1}{N} P(z_t|x_t, \theta^{(i)}). \quad (3.20)$$

An additional practical assumption is also made. Rather than summing over all of the posterior model parameter samples, a subsample $k < N$ is considered, and the mean observation likelihood is used as the approximate posterior likelihood.

# Chapter 4

# Experiments

The proposed Bayesian calibration technique was examined both in simulation and on a real robot. Three different models were used to simulate a wheeled robot with sonar sensors. The data generated from each of these models was used for calibrating the models with the same form, as well as models with simpler forms. The technique was also used to calibrate two different models with the data collected from a real robot. The accuracy of localization using calibrated models was evaluated as well.

## 4.1 Experiments in Simulation

Amigobot is a simulation of the real Amigobot robot. It is a wheeled robot with 8 sonars around its perimeter. There are six sonar sensors on the front and two on the back. Three models with increasing complexity were used to simulate this robot. The calibration procedure was applied to all three of these models, using data generated from identical models or more complicated ones. In the initial experiments, the calibration procedure was applied to two types of data sets: shorter training data sets covering about 10 minutes, and longer ones about 30 minutes. The results were statistically convincing that training with longer data sets are much more effective. Therefore, the longer data sets were used for later calibrations. In the first three experiments, calibration was done assuming the form of models was known. This almost never happens in real-world problems. In the second set of experiments, we examined calibration on wrong models. In these experiments we examined to what extent calibration

could help when the model is wrong. In addition to requiring the form of the model, the calibration procedure requires a prior over the model parameters. The parameters of the models were all assumed to be independent of each other. The prior density functions over the parameters are preset, and for each calibration procedure a set of parameters was drawn from the prior density functions. Then a trajectory was simulated with those randomly generated parameters, and this data set was used for the calibration procedure. The calibration procedure was repeated for a number of i.i.d. samples from the parameters' prior density functions.

Then, the posterior samples from the calibration procedure were used in the three proposed localizing techniques, i.e. *Mean, MAP and Posterior.* The results are presented for each of the three forms of calibrated models as well as the calibrated wrong models. The localization accuracies of the three mentioned methods, are compared to those of the mean of the prior distribution and the true parameter values.

The training and testing data was generated in an asymmetrical L-shaped room, 3×3 meters in size. The readings from all eight sonar sensors and also odometry information were used to localize the robot.

### 4.1.1 Model 1: Simple-Amigobot

The first model that the calibration procedure was applied to simply added independent zero-mean Gaussian noise to the robot: motion and sensors. In the motion model noises were added to forward, sideways and rotational movements. The sensor model also consisted of independent zero-mean Gaussian noise added to each of the sensor's exact reading. Therefore, there were four parameters to be calibrated, which were the variances of the Gaussian noises. The prior density functions were chosen to be gamma distributions. The posterior distribution from the calibration procedure applied on a shorter training data set is shown in the upper part of Figure 4.1 while the lower plots illustrate the posterior distribution from training with a longer training data set. The dark crosses represent the samples from the posterior distribution using the Bayesian calibration method, while the long lines are the true parameter

values that the training data was generated with. Comparing the two graphs, one can see that training with longer training data sets moves the posterior distribution farther from the true values than training with the shorter data set. Although this might seem counterintuitive, one should delay making conclusions to after evaluating the localization accuracy results.

As explained in the previous section, the accuracy of localization using the Mean, MAP and Posterior methods were compared to the mean of the prior and the true values the data was generated with. Figures 4.2 and 4.3 show this comparison for four different parameters, learnt from long and short training datasets. As can be observed in the figure, the parameter values extracted from the learnt posterior distribution localize the robot more robustly than the mean of the prior in both training cases. More precisely, the Mean and Map methods outperform the Posterior method. However, the results are not statistically significant to compare the Mean and Map methods. Compared to them, the Posterior method performs poorly in localization. Another counterintuitive observation in both figures is that Mean and MAP values are localizing the robot even better than the true values. This issue will be analyzed later in section 5.1. To compare the effectiveness of training with the short and long data sets, one needs to consider Figures 4.3 and 4.2. In order to do so, the mean error of each method was averaged over all four parameters. This comparison is shown in Figure 4.4. It can be observed that for all methods, the parameters calibrated with the longer data sets outperform the parameters calibrated with the shorter data sets.

Overall, although calibration does not necessarily result in a correct inference of the true posterior, it is effective for improving localization accuracy.

### 4.1.2 Model 2: Biased-Amigobot

In the following experiments the previous motion model is augmented with biases in forward and sideways movements. Consequently, the number of parameters of the model increases to 6. To model the bias in motion, identical independent distributed Gaussian noises with bias values as the mean of the

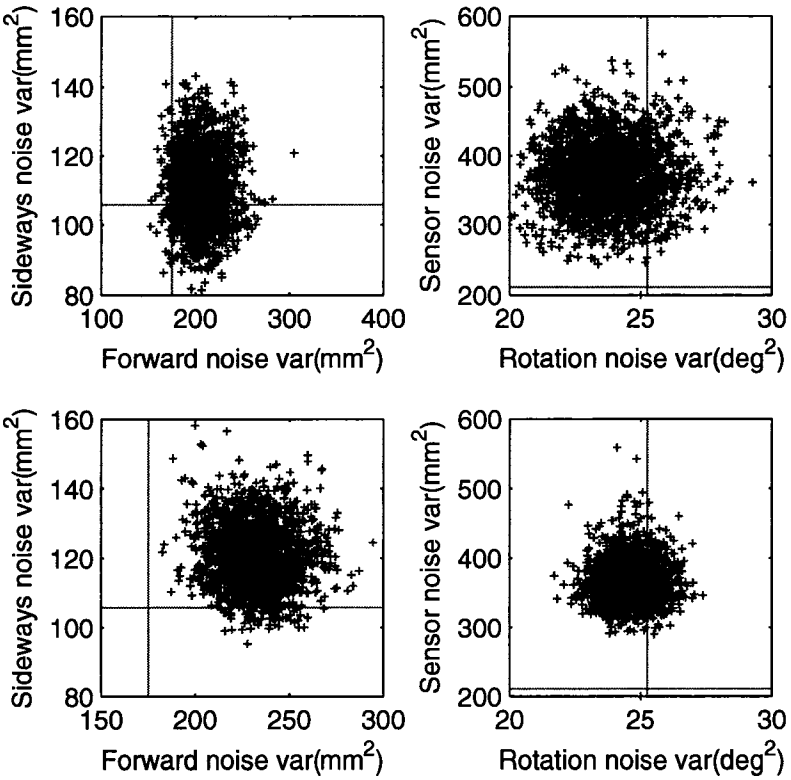


Figure 4.1: Amigobot posterior parameter samples with small (top) and large (bottom) amount of training.

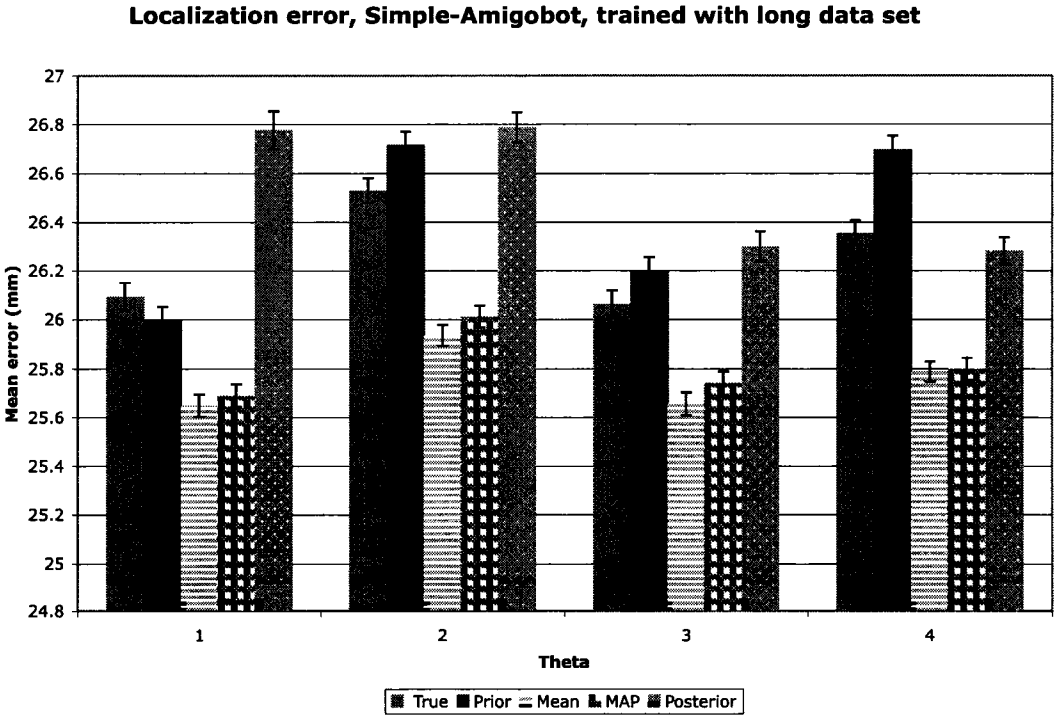


Figure 4.2: Amigobot localization results with 95% confidence intervals. (Trained with long data set)

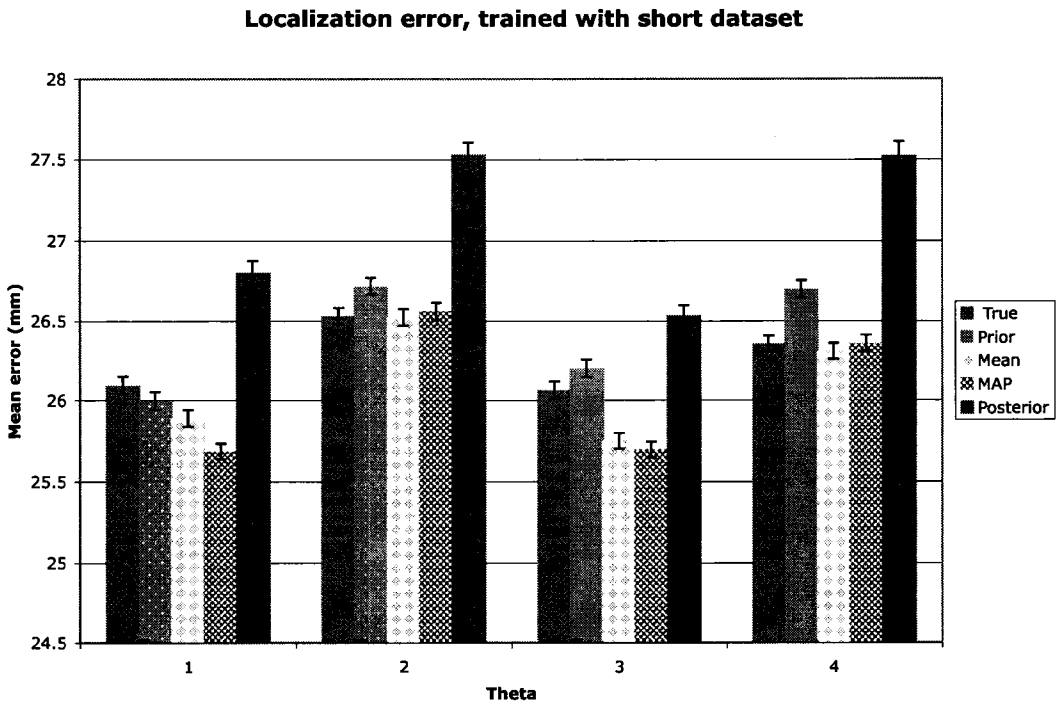


Figure 4.3: Amigobot localization results with 95% confidence intervals. (Trained with short data set)

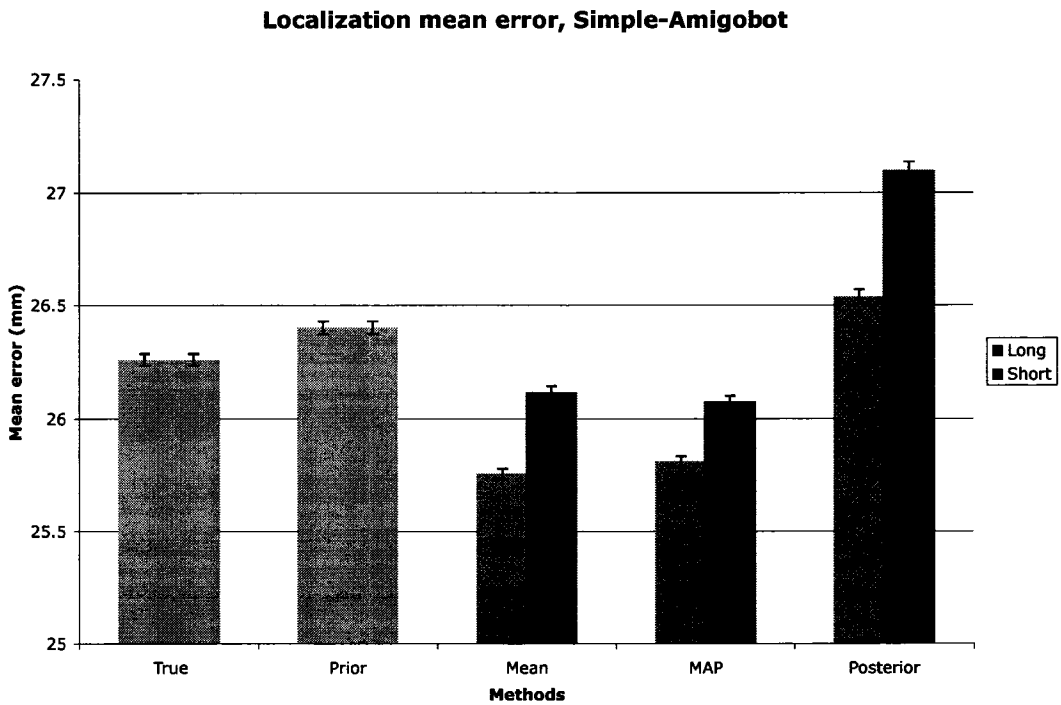


Figure 4.4: Amigobot localization results with 95% confidence intervals (short vs. long).

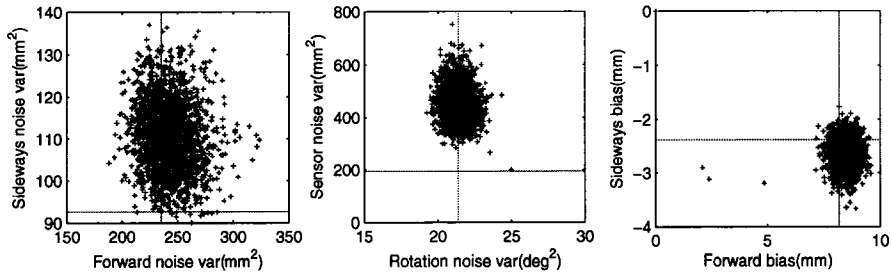


Figure 4.5: Amigobot with biases in motion model, posterior parameter samples after training

distributions were added to the forward and sideways movements. The prior distribution over the biases' space is considered to be Gaussian, since negative numbers are valid values for bias in motion. The sensor model is identical to the sensor model of the previous experiments. The posterior distribution after calibration and the localization accuracy graphs are shown in Figures 4.5 and 4.6. Each column in the graph is the representative of a different set of parameters.

Localization accuracy graphs for the biased model, are compatible with the previous observations of the simpler model. The parameter values from the posterior distribution can localize the robot considerably better than the mean of the prior. They also usually perform better or equally well in comparison to the true parameter values. Although calibration does not infer the true parameters, it is still effective in improving localization accuracy.

### 4.1.3 Model 3: Biased-FNP-Amigobot

In the following experiments, the previous model is made more complicated by adding false positive and false negative to the sensor model and also adding bias to rotational movements. Each sensor reading is still assumed to be independent. Since each sensor reading is either a true positive, false positive or a false negative, the three ratios should add up to one. Therefore a Dirichlet distribution is considered as the prior distribution over these values. This sensor model simulates a robot's sensor that occasionally fails to detect an obstacle, and sometimes detects a nonexistent obstacle, either due to the presence of moving objects or other factors. In this model the sensor model is

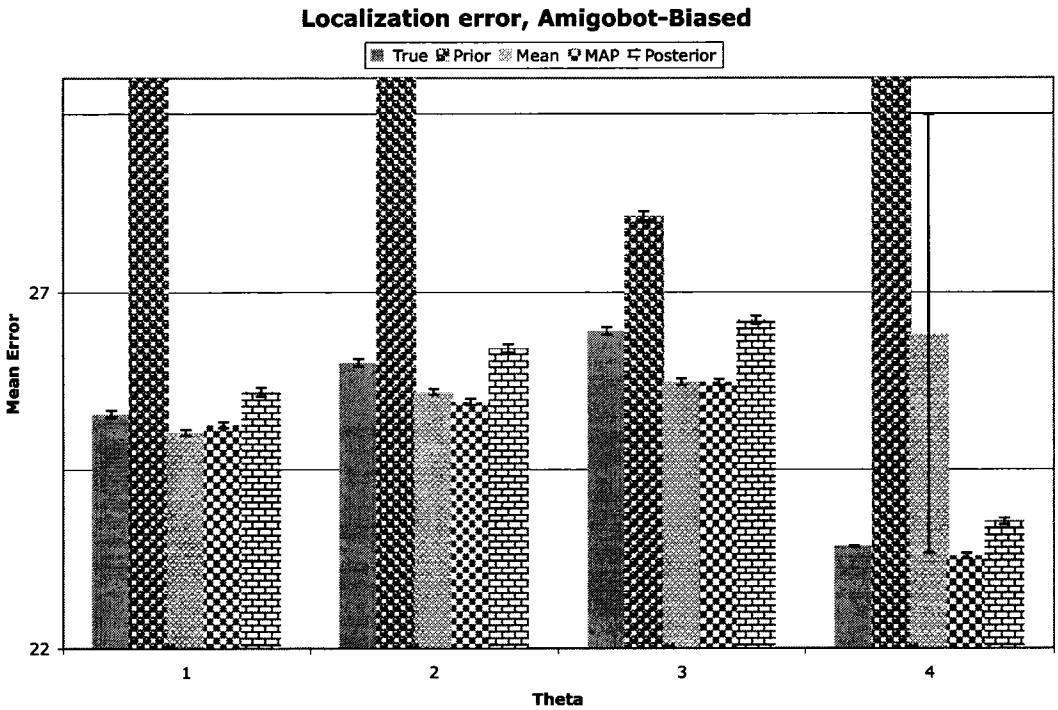


Figure 4.6: Amigobot with biases localization results with 95% confidence intervals.

no longer Gaussian, but a combination of three probability density functions. More formally the sensor model is represented as follows:

$$
\begin{aligned}
P(z_{t,i}|x_t) &= \sum_{\mathcal{T}_{t,i}=\{\mathrm{TP,FN,FP}\}} P(z_{t,i}, \mathcal{T}_{t,i}|x_t) && (4.1)\\
&= \sum_{\mathcal{T}_{t,i}=\{\mathrm{TP,FN,FP}\}} P(z_{t,i}|x_t, \mathcal{T}_{t,i}) P(\mathcal{T}_{t,i}|x_t) \\
&= P(z_{t,i}|x_t, \mathcal{T}_{t,i} = \mathrm{TP}) P(\mathcal{T}_{t,i} = \mathrm{TP}|x_t) \\
&\quad + P(z_{t,i}|x_t, \mathcal{T}_{t,i} = \mathrm{FP}) P(\mathcal{T}_{t,i} = \mathrm{FP}|x_t) \\
&\quad + P(z_{t,i}|x_t, \mathcal{T}_{t,i} = \mathrm{FN}) P(\mathcal{T}_{t,i} = \mathrm{FN}|x_t), && (4.2)
\end{aligned}
$$

where $\mathcal{T}_{t,i}$ represents the set of states of observation of the $i$th sensor({true positive, false positive, false negative }). The states of the observation are assumed independent of the position of the robot. Therefore one can write:

$$
\begin{aligned}
P(z_{t,i}|x_t) &= P(z_{t,i}|x_t, \mathcal{T}_{t,i} = \mathrm{TP}) P(\mathcal{T}_{t,i} = \mathrm{TP}) \\
&\quad + P(z_{t,i}|x_t, \mathcal{T}_{t,i} = \mathrm{FP}) P(\mathcal{T}_{t,i} = \mathrm{FP}) \\
&\quad + P(z_{t,i}|x_t, \mathcal{T}_{t,i} = \mathrm{FN}) P(\mathcal{T}_{t,i} = \mathrm{FN}), && (4.3)
\end{aligned}
$$

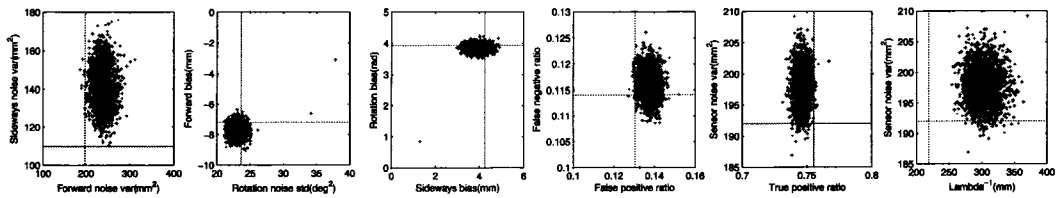

Figure 4.7: Amigobot with biases in motion model and false positive and negatives in sensor model, posterior parameter samples after training

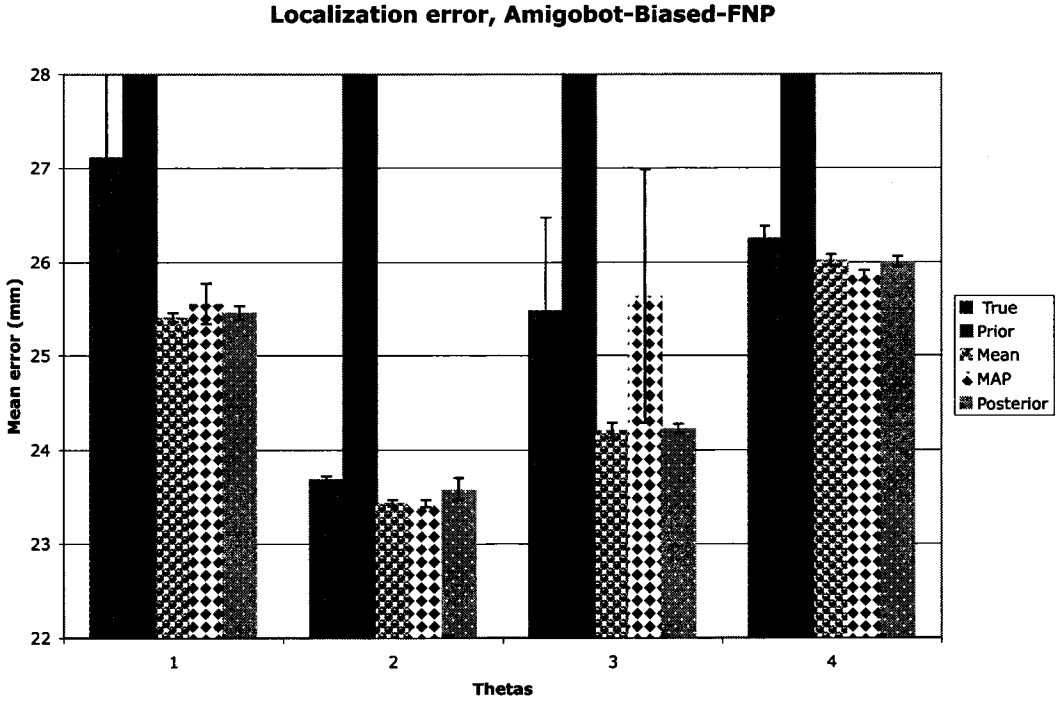


Figure 4.8: Amigobot with biases in motion model and false positive and negatives in sensor model localization results with 95% confidence intervals.

where $P(\mathcal{T}_{t,i} = \{\text{TP}, \text{FN}, \text{FP}\})$ is the probability distribution function over the states of the observation of the $i$th sensor and adds up to one.

Figure 4.7 shows the samples from the posterior distribution after calibration and Figure 4.8 shows the localization accuracy graphs for this model. These results also confirm the observations in the previous experiments.

### 4.1.4 Calibration with Wrong Models

Knowing the form of the models beforehand is an unrealistic assumption. Therefore, another case was examined, where the model that the calibration procedure was applied to was of different form than the model used to generate the training data.

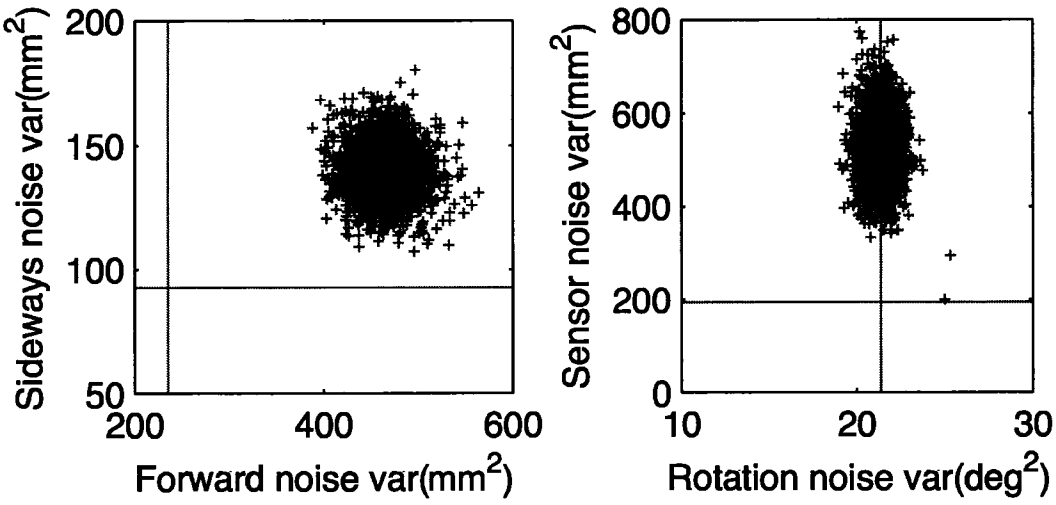


Figure 4.9: Simple Amigobot trained with Amigobot-Biased data

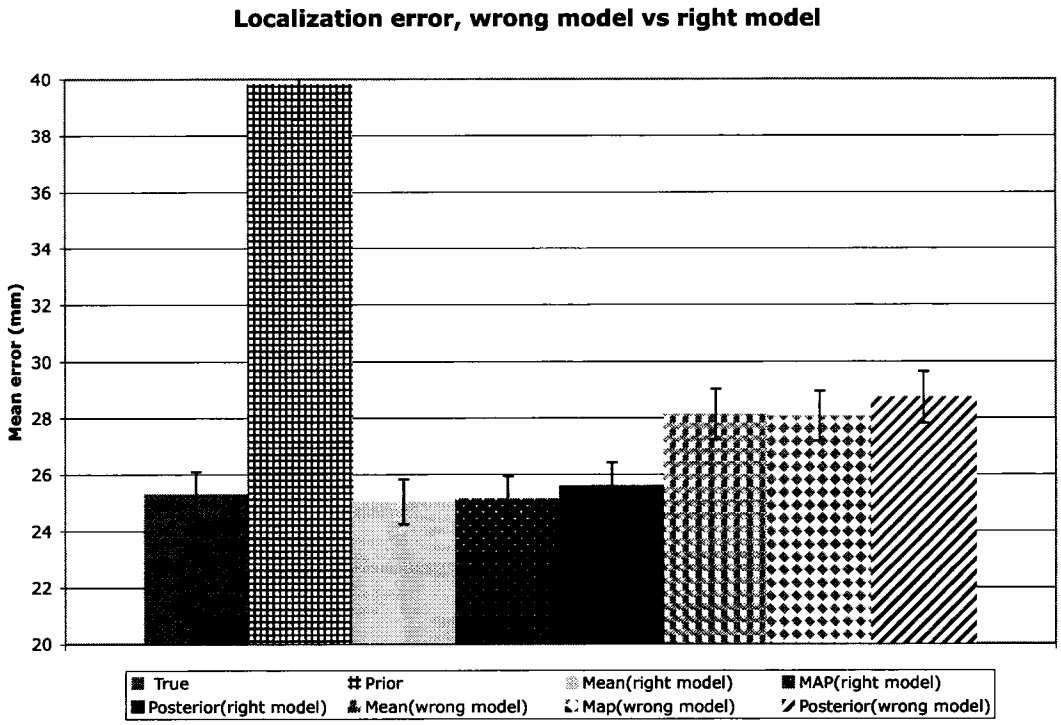


Figure 4.10: Simple Amigobot trained with Amigobot-Biased data

**Experiment One.** In the first experiment the training data was generated with the biased Amigobot, and the calibration procedure was applied to the simple Amigobot model. Figure 4.9 shows the samples of the posterior distribution of the simple model's parameters, trained with data collected from the biased Amigobot. Comparing Figure 4.9 and Figure 4.5, one can see that the posterior distribution over the forward and sideways variances in the simpler model is much farther from the true values. This observation is intuitive. In

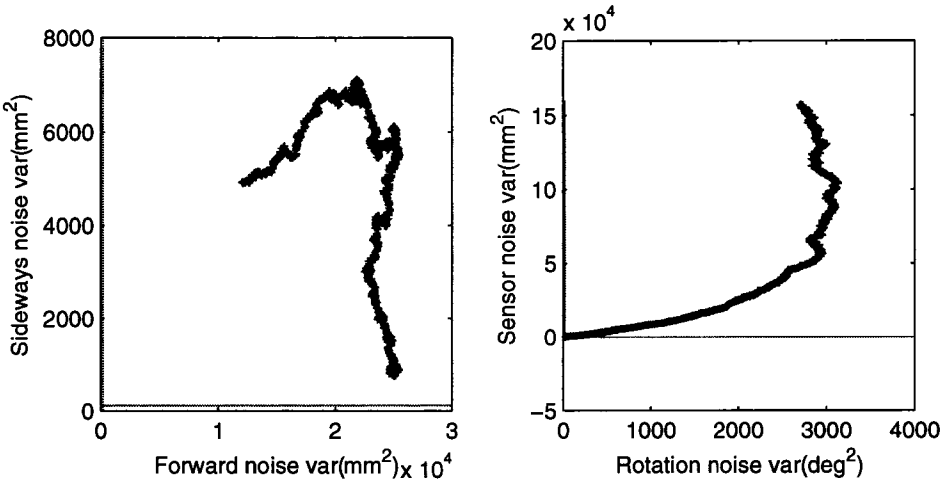


Figure 4.11: Simple Amigobot trained with Amigobot-Biased-FNP data

order for the simpler model to be able to explain the data from the model with biases, the calibration procedure sets the variances of the motion model to higher values. Figure 4.10 compares the localization capability of the wrong calibrated model and the right calibrated model. Note that the prior for both the correct and wrong model perform identically since the mean of the bias parameters are zero. Although the calibrated wrong model does not localize the robot as well as the calibrated right model, calibration considerably improves its performance.

**Experiment Two.** In the second experiment, the simple Amigobot model was trained with the data collected from the Amigobot's model with biases in motion and false negative and positives in the sensor model. Figure 4.11 shows the posterior samples after applying the calibration procedure. As can be seen in the figure the calibration procedure explores the parameter space. This exploration starts from spaces with large variances in the motion model and small variances in the sensor model and stops in the space with large variances in the motion model and very large variances for the sensor model. Ignoring one of the least important dimensions, the parameters' space can be illustrated in a three-dimensional space, as in Figure 4.12. The three methods of localization were evaluated with the calibrated wrong model. All methods failed in localizing the robot. The reason is that the model used for calibration is just so far removed from the true model that it cannot even approximate

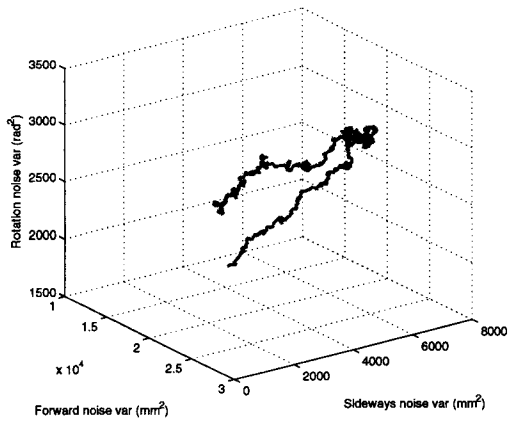


Figure 4.12: 3d representation of the parameter space

the complicated sensor model to make localization possible.

From the last two experiments one can conclude that the closer the selected models are to the true model used to generate the training dataset, the more effective is calibration.

## 4.2 Real Robot Experiments

A Sony AIBO ERS-7 robotic dog was used as the second platform for the experiments. The AIBO is a four-legged robot with a CMOS color camera as its primary sensing device. The AIBO had been used in the Robocup Legged League,in an environment similar to the one used in the experiments. The calibration procedure was applied to data collected from the robot moving in a 2.7×1.8 meter field with bi-colored markers in each corner. The calibration procedure was applied to two different models of the robot, with increasing complexity. An overhead camera was mounted in order to extract the true position of the robot for use as ground truth. As can be seen in Figure 4.13, the robot was marked with a green circle, in order to simplify the extraction of its true position.

### 4.2.1 Model 1: Simple-AIBO

The first motion model used in the experiments was identical to the motion model in Simple-Amigobot: It involved three variances related to three inde-

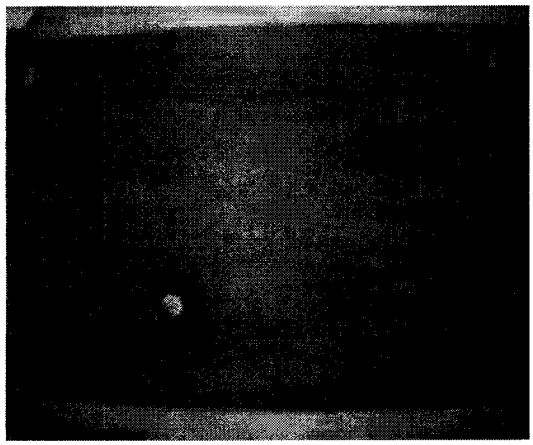

Figure 4.13: AIBO setup

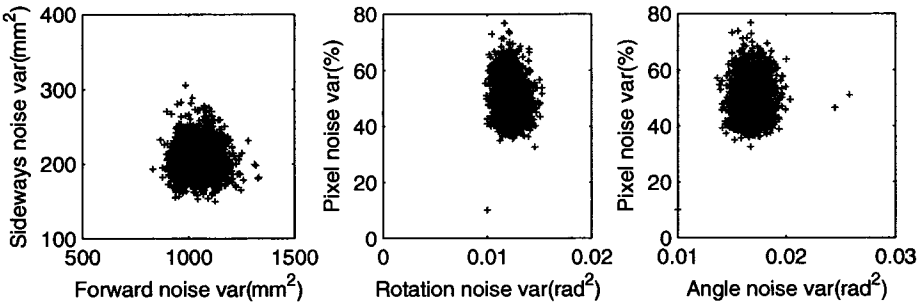


Figure 4.14: AIBO posterior parameter samples

pendent zero-mean Gaussian noise variables added to movements in the three dimensions. For the sensor model two features were extracted from each visible marker: the number of visible pixels and the relative angle towards the marker. The sensor model adds zero-mean Gaussian noise to the relative angle towards each visible marker and zero-mean Gaussian noise to the number of visible pixels from each marker. The variance of the latter noise is assumed to be proportional to the number of visible pixels from each marker. Therefore, there are five parameters that need calibration. Figure 4.14 shows the samples of the posterior distribution after calibration. Localization accuracies of these parameters were then measured by averaging the localization error over hundreds of localization trials on a single ten minute trajectory. Figure 4.15 shows AIBO's localization error bars with 95% confidence intervals. Like the results in simulation, the calibrated parameters outperform the mean of the prior.

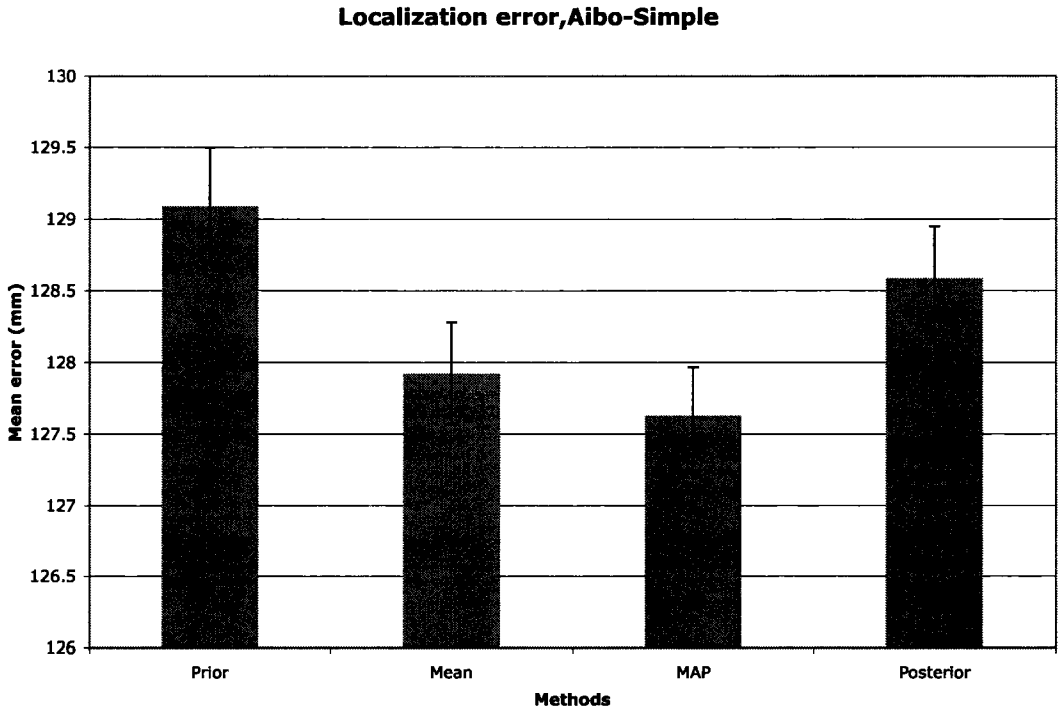


Figure 4.15: AIBO localization results with 95% confidence intervals.

### 4.2.2 Model 2: Biased-FNP-AIBO

The second model used to localize AIBO was a model that supported false positives and false negatives in the sensor model as well as biases in the motion of the robot. The motion model was identical to the motion model of Biased-Amigobot, and the sensor model simulated a noisy camera, which sometimes lead to a misplacement of the marker (false positive) and sometimes lead to missing a marker (false negative). The formal representation of the sensor model follows:

$$
\begin{aligned}
P(z_{t,i}|x_t) &= \sum_{\mathcal{T}_{t,i}=\{\text{TP,FN,FP}\}} P(z_t, \mathcal{T}_{t,i}|x_t) \qquad (4.4)\\
&= \sum_{\mathcal{T}_{t,i}=\{\text{TP,FN,FP}\}} P(z_t|x_t, \mathcal{T}_{t,i}) P(\mathcal{T}_{t,i}|x_t)\\
&= P(z_{t,i}|x_t, \mathcal{T}_{t,i} = \text{TP}) P(\mathcal{T}_{t,i} = \text{TP}|x_t) +\\
&\quad P(z_{t,i}|x_t, \mathcal{T}_t = \text{FP}) P(\mathcal{T}_t = \text{FP}|x_t) +\\
&\quad P(z_{t,i}|x_t, \mathcal{T}_t = \text{FN}) P(T_t = \text{FN}|x_t)\\
&= P(\text{pix}_{t,i}, \text{ang}_{t,i}|x_t, \mathcal{T}_{t,i} = \text{TP}) P(\mathcal{T}_{t,i} = \text{TP}|x_t) +\\
&\quad P(\text{pix}_{t,i}, \text{ang}_{t,i}|x_t, \mathcal{T}_{t,i} = \text{FP}) P(\mathcal{T}_{t,i} = \text{FP}|x_t) +\\
&\quad P(\text{pix}_{t,i}, \text{ang}_{t,i}|x_t, \mathcal{T}_t = \text{FN}) P(\mathcal{T}_t = \text{FN}|x_t)\\
&= P(\text{ang}_{t,i}|x_t, \text{pix}_{t,i}\mathcal{T}_{t,i} = \text{TP}) P(\text{pix}_{t,i}|x_t, \mathcal{T}_{t,i} = \text{TP}) P(\mathcal{T}_{t,i} = \text{TP}|x_t) +\\
&\quad P(\text{ang}_{t,i}|x_t, \text{pix}_{t,i}, \mathcal{T}_{t,i} = \text{FP}) P(\text{ang}_{t,i}|x_t, \mathcal{T}_{t,i} = \text{FP}) P(\mathcal{T}_{t,i} = \text{FP}|x_t) +\\
&\quad P(\text{ang}_{t,i}|x_t, \text{pix}_{t,i}, \mathcal{T}_{t,i} = \text{FN}) P(\text{pix}_{t,i}|x_t, \mathcal{T}_{t,i} = \text{FN}) P(\mathcal{T}_{t,i} = \text{FN}|x_t) \qquad (4.5)
\end{aligned}
$$

where

$$
\sum_{\mathcal{T}_{t,i}=\{\text{TP,FN,FP}\}} P(\mathcal{T}_{t,i}|x_t) = \sum_{\mathcal{T}_{t,i}=\{\text{TP,FN,FP}\}} P(\mathcal{T}_{t,i}) = 1 \qquad (4.6)
$$

and

$$
\begin{aligned}
P(\text{pix}_{t,i}|x_t, \mathcal{T}_{t,i} = \text{TP}) &= N(pix_{t,i}, pix_i^\star, \sigma^2)\\
P(\text{pix}_{t,i}|x_t, \mathcal{T}_{t,i} = \text{FP}) &= exppdf(\text{screensize}, \mu)\\
P(\text{pix}_{t,i}|x_t, \mathcal{T}_{t,i} = \text{FN}) &= \begin{cases} 0 & \text{if } pix_{t,i} > 0 \\ 1 & \text{if } pix_{t,i} == 0, \end{cases}
\end{aligned}
$$

where *exppdf* is the exponential density function. Similar density functions are selected for $\text{ang}_{t,i}$. Figure 4.16 shows samples of the posterior distribution after calibrating the second model. As one can see, the posterior distribution of $P(\text{FP})$ and $P(\text{FN})$ clusters around zero, although the means of the prior distributions are equal to 0.1. The vision system being used already disregarded blobs of a small number of pixels. Therefore, false negatives and false positives were rarely happening.

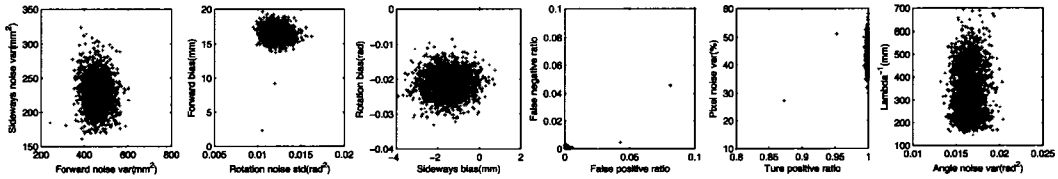

Figure 4.16: Biased-FNP-AIBO samples from the posterior distribution

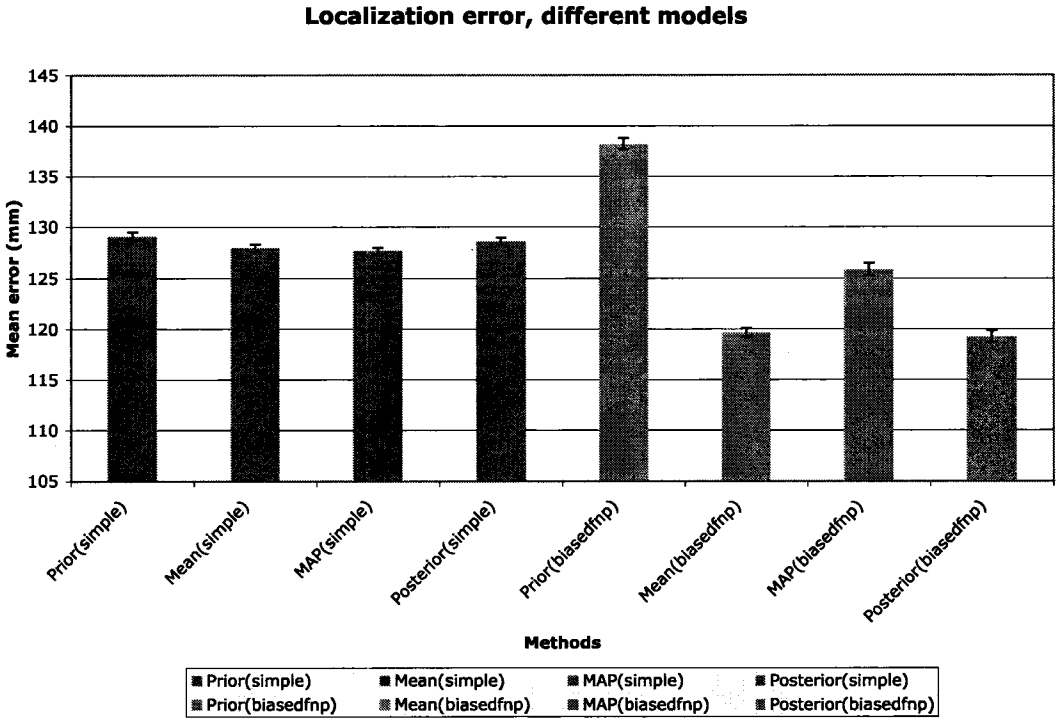


Figure 4.17: Biased-FNP-AIBO localization results with 95% confidence intervals.

Figure 4.17 compares the localization accuracies of the calibrated and uncalibrated complicated models. The parameters extracted from the posterior distribution localize the robot more accurately. One can also observe that localization with the more complicated model is more robust than the simpler model.

# Chapter 5

# Discussion

The experiment results were presented in the previous section. Although most results matched the expectations, some of them were not so intuitive. In this chapter, we explore some of these results are further explored. In the first section the phenomenon of true parameter values being outperformed by calibrated parameters is examined, relying on more experiments as well as observations from the robot localization literature. Subsequently, other questions are examined, all related to the convergence of the calibration and the effects of the number of particles used with calibration and localization.

## 5.1 Discussion of Paradoxical the Results

In Section 4.1.1, it was shown that in simulated experiments the parameters extracted from the posterior distribution outperformed the true parameter values when used in localization. The calibration procedure usually finds parameter values larger than the true parameter values. In the literature on mobile robot localization researchers usually recommend setting the parameter values higher than the values obtained from manual tuning, since it results in more robust localization. This appears to be what the calibration procedure does naturally.

As mentioned before, MCL is an effort to estimate $P(x_T|z_{1:T}u_{1:T})$ posterior distribution given the robot's motion model, $P(x_t|x_{t-1},\theta)$ and sensor model, $P(z_t|x_t,\theta)$. At first glance, one might think that having the true values for $\theta$, would give the true posterior distribution, and therefore, the least error. How-

ever, recall that MCL, which is based on importance sampling, guarantees that the weighted mean of samples converges to the true mean of the target probability density function with an infinite number of particles. Therefore, the samples that MCL provides, are just an estimate of the posterior distribution. Consequently, there is room for improvement in this estimation. Therefore, samples of a posterior with different parameter values, might generate a closer mean to the true trajectory and also a higher probability.

As MCL is a step of the calibration procedure, the algorithm uses the same approximation used when localizing the robot. This suggests that the algorithm might be able to compensate for this approximation error. In other words, our calibration technique might be able to find parameters that are suitable to the number of particles used in localization. This possibility is explored experimentally in the next section.

## 5.2 Number of Particles

This section examines the effect of the number of particles used during calibration to localization accuracy. Both the number of particles used in the particle smoothing step of calibration as well as the number of particles used in localization were varied when evaluating the resulting calibrated models. All experiments were performed on Simple-Amigobot. The environmental setup was the same as in Section 4.1. The calibration procedure was run with 50, 100, 200 and 400 particles for 10 randomly chosen sets of parameters. Then the learnt parameters were tested and compared with the true and prior parameter values in localization using 50, 100, 200 and 400 particles, by averaging the mean error over thousands of trajectories. The graph in Figure 5.1 shows four lines, each color representing parameters calibrated with a different number of particles in the calibration procedure. The figure shows that calibration with a fewer number of particles will result in more robust parameters for localization. In other words, the parameters should be calibrated using a particle set smaller than the particles set used in localization.

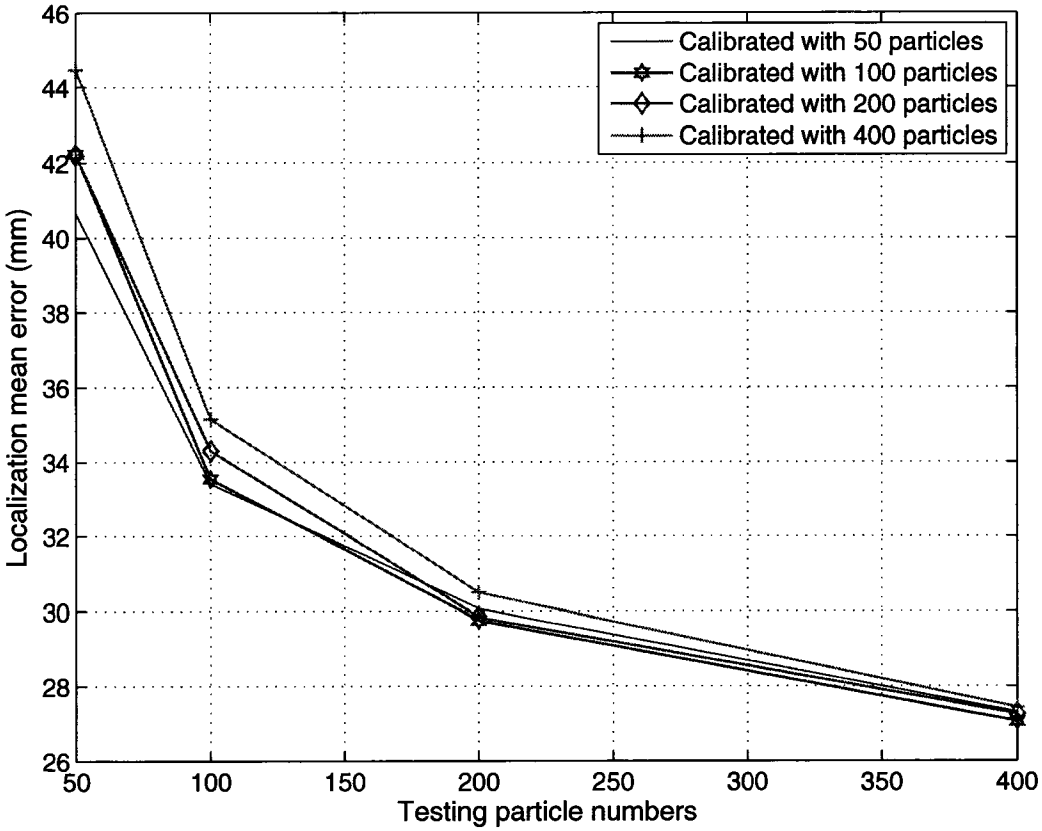


Figure 5.1: Connection between the training and testing particle numbers

## 5.3 Incremental Calibration

The likelihood function over the parameter space with the entire data sequence may involve very sharp peaks of likelihood making it difficult for MCMC to find and sample from these regions. Although it was not clear whether this was a problem, data was slowly being added during calibration to compensate for this potential problem. The motivation was that the posterior distribution does not change significantly as small amounts of data are added. So a sample drawn from the posterior given only a portion of the data would be a good starting point for the posterior given slightly more data. Thus, each iteration of MCMC can include more and more data until the entire trajectory is being considered. Figure 5.2 compares the output posterior distributions after applying the calibration procedure using the whole data set and the distribution when the training data was added gradually. Figure 5.3 shows the resulting differences in localization accuracy. There is no appreciable improve-

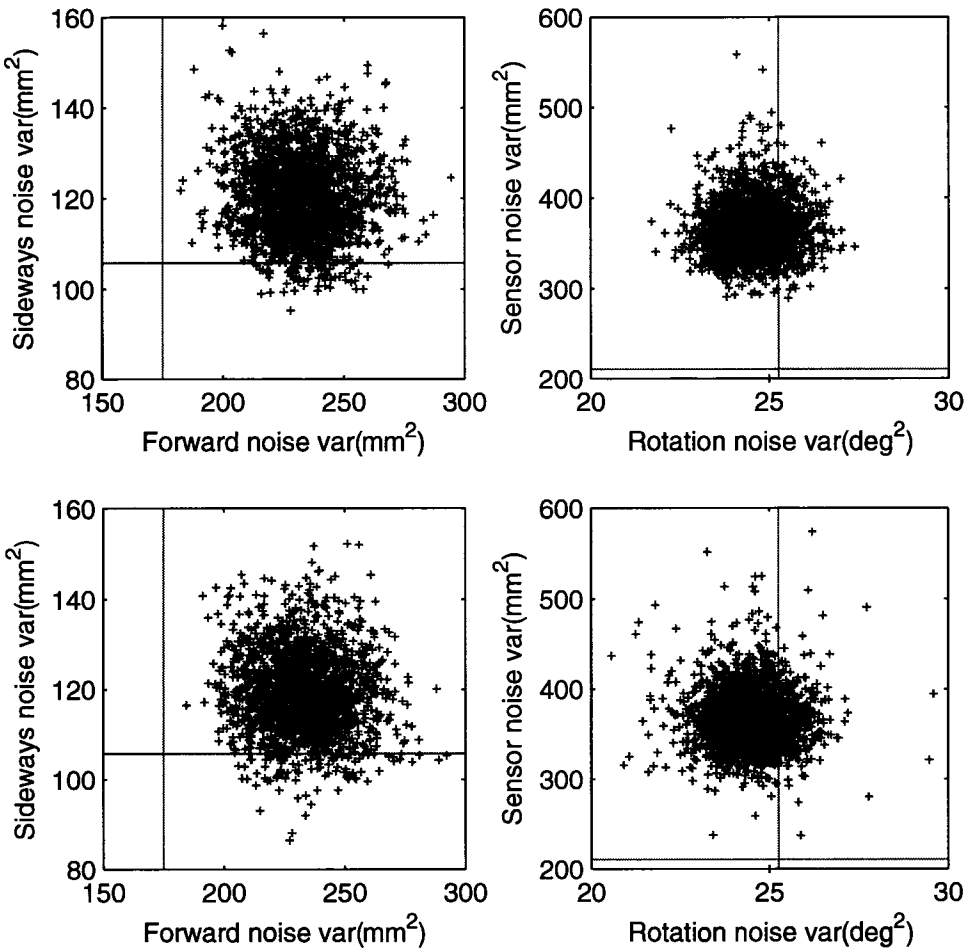


Figure 5.2: Posterior distributions after calibration with a complete data set (top) versus gradually added data (bottom)

ment by incrementally including data, as neither does the posterior nor the resulting localization improve in a significant way. One might as well start drawing samples from the target distribution instead of wasting effort shifting the distribution from the prior towards the posterior distribution.

## 5.4 Long Training Data Set

Lastly, the effect of a very long training trajectory was examined to confirm whether this might affect the rate of convergence of the chain. Simple-Amigobot model was trained using a trajectory with 20000 steps (i.e., 5.5 hours). Figure 5.4 shows the posterior distribution after the calibration procedure. As one can observe, the distribution is rather concentrated and the chain

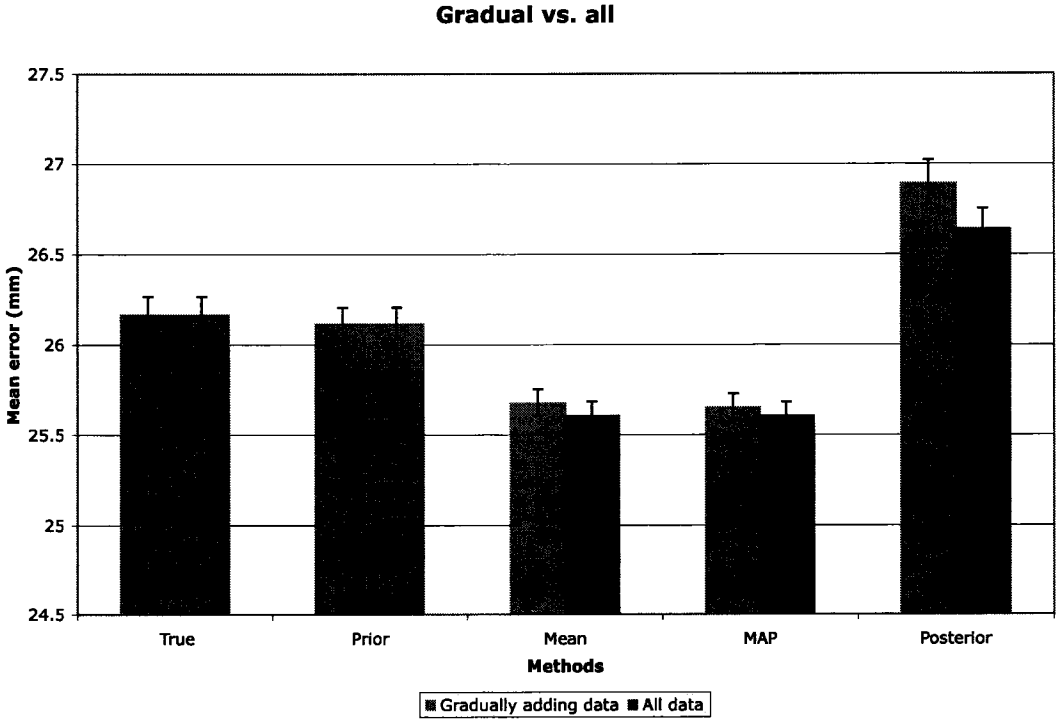


Figure 5.3: Localization accuracy graphs after calibrating with a complete data set versus gradually added data

does not diverge due to the prolonged data. Although one might imagine that a very long data set would produce small values for the acceptance function and therefore, a low acceptance rate.

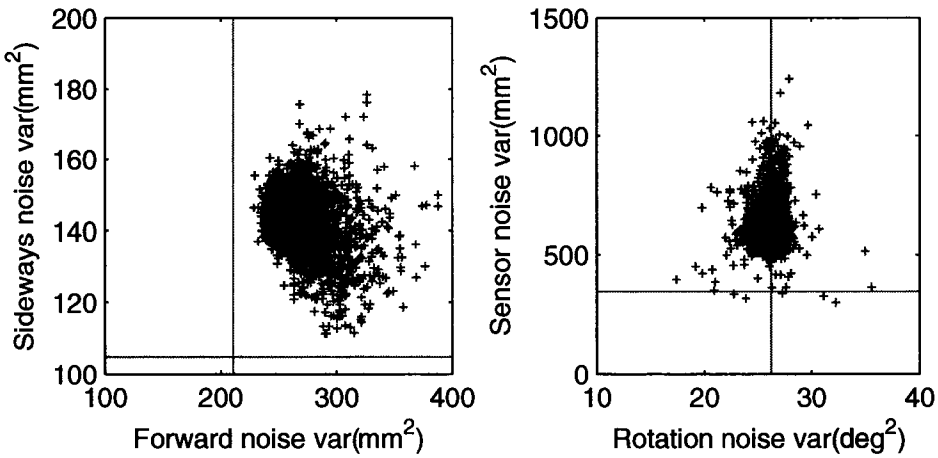


Figure 5.4: Calibrating the simple model of Amigobot with a very long training data set

# Chapter 6

# Related Work

This chapter examines previous work on the calibration problem. Although the need for accurate models in robot localization is crucial, the calibration problem has not received much attention in the robotics community and much of the work focuses primarily on motion model calibration.

This problem has received more attention as the general problem of parameter or model estimation within the state-estimation community [2]. Therefore, this chapter is presented in two main sections. The first section examines related work on automatic calibration within the robotics field. The second section focuses on previous work in the more general state-estimation community.

## 6.1 Calibration in Robotics

In this section, calibration related work within the robotics community is examined.

### 6.1.1 Stronger and Stone

Stronger and Stone [13] model the motion and sensor of a robot with deterministic models. The models that they use have the following form:

$$\text{Motion model:} \quad x_t = x_0 + \int_0^t A(u_s)ds \tag{6.1}$$

$$\text{Sensor model:} \quad x_t = S(z_t) \tag{6.2}$$

where $x_t$ represents the distance from the marker, therefore it has only one dimension. $u_t$ represents the action command that robot received at time $t$. $S$ is the function that relates the observation $z_t$ to a specific location and $A$ is a function that specify the relation between the actions and the location of the robot. The authors use the following function approximators to define the functions $A$ and $S$.

$$A(u_t) = \sum_{i=0}^{d} a_i u_t^i \tag{6.3}$$

$$S(z_t) = \sum_{i=0}^{d} s_i {x_t}^i, \tag{6.4}$$

where $d$ represents the degree of the polynomials. The algorithm attempts to learn coefficients $a_i$ and $s_i$. As one can see in Equations 6.1 and 6.2, there are two sources for calculating $x$, and this basically shapes the basis of the algorithm proposed in [13]. The technique, SCASM (Simultaneous Calibration of Action and Sensor Models), is un-supervised and learns one model from the other one. The authors present a weighted polynomial regression method to calibrate the models. However, since the models they use are not probabilistic, the models and consequently the calibration technique cannot be directly used in Monte Carlo localization.

### 6.1.2 Roy and Thrun

Roy and Thrun [11] use an incremental maximum likelihood estimator to calibrate the motion model online. However, the authors assume that the noise term in the model is known and just calibrate for the bias terms in translational and rotational movements. More specifically, the motion model has the following form:

$$X_t = AX_{t-1} + N(B, \Sigma_B), \tag{6.5}$$

where $\Sigma_B$ is assumed to be known and the goal is to calculate $B$ which represents the bias vector including bias factors in forward and rotational movements. The authors estimate the bias vector maximizing the likelihood of bias parameters given the difference of two sequential readings of the sensors and

the robot's measurement of the transition. More precisely they suggest an efficient method to solve:

$$\underset{B_{trans},B_{rot}}{\text{argmax}} \; P(B_{trans}, B_{rot}|\mathcal{D}) \tag{6.6}$$

Roy and Thrun calibrate the motion model completely relying on the sensor readings of the robot. Without using a probabilistic model for sensors. However, sensors have varying degrees of accuracy. The more noisy a sensor is, the more crucial it is to be modeled probabilistically. The proposed approach might be effective for the robot equipped with laser range-finders used by the authors, however it might deteriorate for a robot with noisier sensors.

### 6.1.3 Eliazar and Parr

Eliazar and Parr [3] go one step further than Roy and Thrun and calibrate not only the terms that account for systematic errors but also the noise terms related to the stochastic nature of the robot's motion. The framework that the calibration is applied to is that of Simultaneous Localization and Mapping (SLAM). The presented motion model involves 12 parameters and calibrated by Expectation Maximization (EM) technique. In the E-step, particle smoothing is applied to the history of the particles and the average of the likelihoods is calculated. Later, a weighted least squares method is used to maximize the likelihood function in the M-step. The advantage of Eliazar and Parr's method over Roy and Thrun's is that it does not rely on sensor readings in order to calibrate the motion model. However, both proposed techniques are incomplete in the sense that they do not attempt to calibrate the sensor model.

### 6.1.4 Milstein and Wang

Milstein and Wang [9] developed a dynamic calibration technique to calibrate the motion model for MCL. Like Roy and Thrun, they also use a robot equipped with a laser range-finder. Therefore, the technique is based on the assumption that the sensors and, therefore, the sensor model is precise and the data is reliable enough to calibrate the motion model. The proposed approach is basically to correct the motion model with the collected observation

up to that time step. The mean of the particles is frequently collected before and after the resampling step of MCL. Then the collected data is matched to two Gaussian distributions, whose parameters are linear functions of the noise parameters considered for forward, sideways and rotational movements. The problem is formulated as an optimization problem and solved using conventional optimization methods. Although satisfactory experimental results are reported, this method does not calibrate the sensor model and would not be as effective for robots with less precise sensors.

## 6.2 State Estimation

This section describes related research in the state-space community.

### 6.2.1 Andrews

Andrews' Bayesian learning algorithm [1], is the most closely related work to the technique proposed in this thesis. Andrews suggests a Bayesian approach to simultaneous state and parameter estimation in nonlinear state-space models (NSSMs) with more stress on parameter estimation. The state and observation generative models are of the following form:

$$x_t = A\phi(x_{t-1}) + \epsilon_x \tag{6.7}$$

$$y_t = B\psi(x_t) + \epsilon_y \quad \text{If y is continuous} \tag{6.8}$$

$$y_t = \frac{1}{Z_h} e^{B\psi(x_t)} \quad \text{If y is discrete,} \tag{6.9}$$

where

$$\phi(x_t) = e^{-(x_t-\mu_k)^T {\Sigma_{\phi_k}}^{-1}(x_t-\mu_k)} \tag{6.10}$$

$$\psi(x_t) = e^{-(x_t-v_k)^T {\Sigma_{\psi_k}}^{-1}(x_t-v_k)} \tag{6.11}$$

The author then presents an MCMC method to draw samples from the posterior distribution of $P(\theta, x_{1:T}|\mathcal{D})$ where $\theta = \{A, B\}$.
Similar to the method proposed in this thesis, Andrews' technique consists of iterating between two steps : parameter estimation and inference. Suggested inference step is very similar, as it employs particle smoothing in order to draw

samples from the state vectors. However, the approach to learning is different since the generation models are assumed to be of a specific form. Consequently the author is able to derive a method to sample from the posterior distribution of $P(\theta|\mathcal{D}, x_{1:T})$, assuming that the prior distribution over the parameters $A$ and $B$ are Gaussian:

$$\theta^{(i)} \sim P(\theta|\mathcal{D}, {x_{1:T}}^{(i)}) \tag{6.12}$$

$$\sim P(A|\mathcal{D}, {x_{1:T}}^{(i)})P(B|\mathcal{D}, {x_{1:T}}^{(i)}) \tag{6.13}$$

where:

$$P(A|\mathcal{D}, {x_{1:T}}^{(i)}) \propto P(A)P({x_{1:T}}^{(i)}|A) \tag{6.14}$$

$$P(B|\mathcal{D}, {x_{1:T}}^{(i)}) \propto P(B)P(\mathcal{D}^{(i)}|B) \tag{6.15}$$

Since the posterior distributions above are products of Gaussian distributions, they are Gaussian themselves. Therefore they can be simulated easily. Andrew's approach, however, is not applicable to calibration problems in robotics, since the models that are used in robotics, are more complicated (e.g., the mixtures of distributions used in the Biased-FNP-Amigobot model from Section 4.1.3) and cannot be represented by forms suggested in [1].

### 6.2.2 Storvik

Storvik [12] proposes an online Rao-Blackwellized particle filter technique to draw samples from the joint posterior of unknown dynamic states and static parameters. The algorithm benefits from the existence of *sufficient statistics* for the history of states and observations, in order to work in real-time. This technique is to marginalize the static parameters out of the joint distribution and simplify the distribution of the unknown parameters given the history of observation and unknown states by replacing the history with sufficient statistics, which are updated recursively. More specifically in order to draw samples from the joint distribution of $P(x_{1:t}, \theta|z_{1:t})$, applying Bayes and chain

rules, Storvik factorizes the distribution as follows:

$$
\begin{aligned}
P(x_{1:t}, \theta | z_{1:t}) &= CP(x_{1:t}, \theta, z_t | z_{1:t-1}) && (6.16)\\
&= CP(x_{1:t-1} | z_{1:t-1}) P(\theta | x_{1:t-1}, z_{1:t-1}) && (6.17)\\
&\times \; P(x_t | x_{1:t-1}, z_{1:t-1}, \theta) P(z_t | x_{1:t}, z_{1:t-1}, \theta)\\
&= CP(x_{1:t-1} | z_{1:t-1}) P(\theta | T_{t-1}) P(x_t | x_{t-1}, \theta) P(z_t | x_t, \theta) && (6.18)
\end{aligned}
$$

where $T_{t-1}$ is the sufficient statistics for the history of observations and states. Storvik's algorithm is summarized in Table 6.

**Algorithm 6** Storvik's Algorithm.

1. Importance sampling : for i = 1 to N
   (a) sample $theta^{(i)} \sim P(\theta | T_{t-1}^{(i)})$
   (b) sample $x_t^{(i)} \sim P(x_t | x_{t-1}, \theta^{(i)})$
   (c) $w^{(i)} = P(z_t | x_t^{(i)}, \theta^{(i)})$
2. Resampling: for i = 1 to N
   (a) sample $j$ from $1, \ldots, N$ with probability $\propto w^{(i)}, i = 1, \ldots, N$
   (b) sample $x_t^{(i)} = x_t^{(j)}$
   (c) $T_t^{(i)} = T(T_{t-1}^{(i)}, x_t^{(i)}, z_t)$
   (d) $w^{(i)} = 1$
3. return $X_t$

This approach suffers from the same limitation as Andrews' approach in that it can't represent common models used in robotics.

# Chapter 7

# Conclusion and Future Work

The goal of this thesis was to develop a method for automatic calibration of parametric probabilistic models used in Monte Carlo localization. This final chapter summarizes the contributions of this thesis and presents some directions for future work.

## 7.1 Contributions

There are two main contributions of this thesis:

**Efficient Bayesian calibration for Monte Carlo localization.** An efficient Markov Chain Monte Carlo (MCMC) sampling method was developed. The method approximates the posterior distribution of parameters of the models used in Monte Calro localization, depending on the data collected from the robot. Since the problem was approached in a general scope, the proposed technique, unlike the previously developed methods in the literature, is applicable to complicated models which are commonly used in robotics.

**A novel extension to Monte Carlo Localization.** An extension to Monte Carlo localization was proposed, which can exploit the generated samples from the posterior distribution of parameters given the collected data.

The effectiveness of the proposed techniques was evaluated both in simulation and on a real robot.

## 7.2 Future Work

As shown in this thesis, it is possible to automatically calibrate the motion and sensor models entirely from data gathered on the robot. Unfortunately, non-engineered environments tend to be irregular and no single model will work in all environments or even throughout a single environment. For example, motion on concrete or muddy ground differs significantly, while weather conditions can drastically affect the behavior of sensors. In order to adapt models to often rapidly changing conditions, the robot needs to frequently collect data and calibrate its models in real-time. However, the current MCMC technique, can not be used for online calibration, since the computation time increases with the length of the collected data. Keeping the collected data also becomes infeasible in long run.

If the posterior distribution of parameters given the collected data depends on low dimensional *sufficient statistics*, the computation time for sampling the posterior distribution becomes independent of the length of the data. As mentioned in the related work chapter, Storvik proposes a Rao-Blackwellized particle filtering technique for online state estimation with the presence of unknown static parameters. Rao-Blackwellization basically marginalizes the static parameters out of the posterior distribution. It then recursively calculates some low-dimensional sufficient statistics from the history of observations and hidden states, which is the key to the real-time feasibility of the approach. However, Storvik's approach is limited to certain types of models and also assumes that the parameters of the sensor model are either known, or that some special treatments should be applied to the algorithm.

Extending Storvik's approach to be applicable to online calibration of common robotic models is an interesting direction for future work. Comparing the offline method developed in this thesis, and such an online technique in various combinations would help establish the ideal applications of calibration for robot localization.

# Bibliography


[1] Mark W. Andrews. Bayesian learning in nonlinear state-space models, 2005. Unpublished manuscript.

[2] A. Doucet, J. F. G. De Freitas, and N. J. Gordon, editors. *Sequential Monte Carlo Methods in Practice. New York.* Springer-Verlag, New York, 2000.

[3] Austin I. Eliazar and Ronald Parr. Learning probabilistic motion models for mobile robots. In *Twenty-First International Conference on Machine learning*, 2004.

[4] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 343–349. 1999.

[5] A. H. Jazwinski. *Stochastic Processes and Filtering Theory.* Elsevier, 1970.

[6] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[7] Armita Kaboli, Michael Bowling, and Petr Musilek. Bayesian calibration for Monte Carlo localization. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*, pages 964–969, 2006.

[8] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basis Engineering*, 82:35–45, 1960.

[9] Adam Milstein and Tao Wang. Localization with dynamic motion models: Determining motion model parameters dynamically in monte carlo localization. In *Proceedings of the Third International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2006.

[10] Radford Neal. Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

[11] Nicholas Roy and Sebastian Thrun. Online self-calibration for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2292–2297, 1999.

[12] Geir Storvik. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions on Signal Processing*, 50:281–289, Feb 2002.

[13] Daniel Stronger and Peter Stone. Simultaneous calibration of action and sensor models on a mobile robot. In *IEEE International Conference on Robotics and Automation*, 2005.

[14] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[15] Larry Wasserman. *All of Statistics*. Springer, 2004.