UC Riverside

UC Riverside Electronic Theses and Dissertations

Title

Mobile Robot Navigation With Low-Cost Sensors

Permalink

https://escholarship.org/uc/item/1g77q1fx

Author

Yap, Teddy Ng

Publication Date

2009

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Mobile Robot Navigation With Low-Cost Sensors

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Teddy Yap, Jr.

December 2009

Dissertation Committee:
Dr. Christian R. Shelton, Chairperson
Dr. Jay A. Farrell
Dr. Anastasios I. Mourikis

Copyright by
Teddy Yap, Jr.
2009

The Dissertation of Teddy Yap, Jr. is approved:

_____

_____

_____

Committee Chairperson

University of California, Riverside

Acknowledgments

I thank the following people for making this dissertation come to fruition.

My esteemed advisor, Dr. Christian Shelton, for his utmost guidance, support, encouragement, and patience during my long and arduous journey towards a Ph.D. Christian helped me tremendously from the first day I joined his research group. I will be forever grateful, thankful, and honored for all the valuable learning and advice he has given me. Christian will always be my role model of an excellent teacher and mentor. Thank you Christian.

My dissertation committee members, Dr. Jay Farrell and Dr. Anastasios Mourikis, for their helpful suggestions and comments for making this dissertation even better. To Dr. Anastasios Mourikis for the valuable help and encouragement he extended me while I was working on the monocular vision-based localization problem of this dissertation.

The former and current student members of the Riverside Lab for Artificial Intelligence Research (R-LAIR), namely Juan Casse, Busra Celikkaya, Alex Eisner, Dr. Yu Fan, Kevin Horan, Dr. Kin Fai Kan, Antony Lam, William Lam, Joon Lee, Dr. Guobiao Mei, and Jing Xu, for making EBU II 368 a wonderful and memorable place to learn and work.

The many friends I have met at UCR including Bilson Campana, Dr. Jason and Ruth Cantera, Anning Chen, Dr. Paul and Irma De Lay, Dr. Ece Gelal, Dr. Ann Gordon-Ross, Dr. WeeSan Lee, Dr. Mae Grace Nillos, Terri Phonharath, Arvind Ramanandan, Xiaoyue (Elaine) Wang, Dr. Yonghui Wu, Dr. Dragomir Yankov, and Lexiang Ye.

Finally, my family, especially my mom, for the love, faith, and support that they have always given me. I dedicate this dissertation to them.

ABSTRACT OF THE DISSERTATION

Mobile Robot Navigation With Low-Cost Sensors

by

Teddy Yap, Jr.

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, December 2009
Dr. Christian R. Shelton, Chairperson

Mobile robots are becoming ubiquitous and an essential part of our everyday lives. They are increasingly taking their place in service-oriented applications including domestic and entertainment roles. They open up many potential opportunities, but they also come with challenges in terms of their limited sensing capability and accuracy and minimal on-board computing resources. In this dissertation, we address three fundamental problems in mobile robotics and demonstrate our approach to each of the problems with a mobile robot equipped with low-cost and low-end sensors. The problems we consider are those of mobile robot calibration, mobile robot localization, and simultaneous localization and mapping.

Motion and sensor models are crucial components in current algorithms for mobile robot localization and mapping. We demonstrate how the parameters of both the motion and sensor models can be automatically estimated during normal robot operations via machine learning

methods thereby eliminating the necessity of manually tuning these models through a laborious calibration process. Mobile robot calibration is important especially for robots relying on cheap and less-accurate sensors. Results from real-world robotic experiments with a robot equipped with wheel encoders and sonar sensors are presented that show the effectiveness of the estimation approach.

Monocular vision has long been regarded as an attractive sensor for the localization of a mobile robot. In this dissertation, we present a particle filtering approach to real-time pose estimation for a small-scale indoor mobile robot equipped with wheel encoders for its odometry and aided by a standard perspective camera. Vision is used for detecting naturally occurring static three-dimensional point features or landmarks from the environment and utilizing the information for correcting the pose as suggested by the odometry. We validate the effectiveness of the particle filter approach extensively with both simulations as well as real-world data and compare its performance against that of the extended Kalman filter.

Simultaneous localization and mapping (SLAM) is a well-studied problem in mobile robotics and the majority of the existing techniques rely on the use of accurate and dense measurements provided by laser rangefinders to correctly localize the robot and produce accurate and detailed maps of complex environments. In this dissertation, we present our approach to SLAM with low-cost but noisy and sparse sonar sensors in large indoor environments involving large loops. Results from robotic experiments demonstrate that it is possible to produce good maps of large indoor environments with large loops despite the inherent limitations of sonar sensors.

Contents

List of Tables **xi**

List of Figures **xii**

1 Introduction **1**

1.1 Motivation 1

1.2 Mobile Robot Calibration 3

1.3 Mobile Robot Localization 4

1.4 Simultaneous Localization and Mapping 5

1.5 Contributions 7

2 Background **9**

2.1 Bayes' Filter 10

2.2 Kalman Filter 14

2.3 Extended Kalman Filter 18

2.4 Particle Filter 23

2.4.1 Importance Sampling 28
2.5 Other Filters 32
2.6 Discussion 32

3 Mobile Robot Calibration **35**
3.1 Introduction 35
3.2 Related Work 37
3.3 Probabilistic Motion and Sensor Models 39
3.3.1 Motion Model 39
3.3.2 Sensor Model 43
3.4 Particle Filtering and Smoothing 46
3.4.1 Particle Filtering 47
3.4.2 Particle Smoothing 47
3.5 Expectation Maximization and Parameter Estimation 51
3.5.1 Expectation Maximization 52
3.5.2 Parameter Estimation Framework 57
3.6 Experimental Results 65
3.7 Summary 72

4 Mobile Robot Localization **74**
4.1 Introduction 74
4.2 Related Work 77

4.3 Particle Filtering Approach 81
4.3.1 Motion Model 82
4.3.2 Sensor Model 83
4.3.3 Feature Parametrization 86
4.3.4 Feature Estimation Using Gauss-Newton Minimization 90
4.3.5 Summary of Particle Weighting 95
4.3.6 Unscented Transform 96
4.4 Experimental Results 98
4.4.1 Simulation Results 98
4.4.2 Real-World Results 100
4.5 Discussion and Summary 104

5 Simultaneous Localization and Mapping **107**

5.1 Introduction 107
5.2 Related Work 110
5.3 Localization Approach 113
5.3.1 Motion Model 114
5.3.2 Sensor Model 115
5.3.3 Particle Filtering 118
5.4 Map Building Approach 118
5.4.1 Line Segment Extraction 118

5.4.2 Line Merging and Map Management 122

5.5 Rao-Blackwellized Particle Filter 124

5.6 Experimental Results 126

5.7 Summary 132

6 Conclusions **134**

Bibliography **137**

List of Tables

3.1 Parameter estimation experiments summary 68

3.2 The initial and estimated values of the parameters 69

3.3 The navigation times of the robot 70

4.1 The mean RMSE of the MSCKF and particle filter 100

4.2 Parameter settings for the real-world experiments 101

5.1 SLAM experiments summary 128

List of Figures

3.1 The block diagram for the parameter estimation framework 66

3.2 The first test environment and its associated map with waypoints 67

3.3 The second test environment and its associated map with waypoints 67

3.4 The generated sonar maps 71

3.5 Sonar endpoint accuracy 72

4.1 The simulation environment and the true trajectory of the robot (circle) 99

4.2 Some sample images taken from the first real-world experiment 102

4.3 The estimated trajectory in the first experiment 102

4.4 Some sample images taken from the second real-world experiment 104

4.5 The estimated trajectory in the second experiment 105

5.1 Computation of the match value 117

5.2 Distances for overlapping and non-overlapping lines 123

5.3 The three test environments and their associated maps 130

5.4 The estimated maps and robot trajectories 131

Chapter 1

Introduction

1.1 Motivation

As we progress further into the 21st century, mobile robots are becoming increasingly important, useful, and commonplace in our society. Mobile robots are being employed in growing numbers not only in industrial but also in service-oriented applications (which include domestic and entertainment applications) as well. According to a recent study by the International Federation of Robotics Statistical Department [2009], a total of about 7.2 million service robots for personal and domestic use were sold through the end of 2008, of which about 4.4 million units are for domestic use and about 2.8 million units are for entertainment and leisure (compare that to 63,000 service robots for professional use and 113,300 industrial robots sold through the end of the same year). Service robots for personal and domestic use are largely in the areas of domestic (household) robots, which include vacuum-cleaning and

lawn-mowing robots, and entertainment and leisure robots, which include toy robots, hobby systems, and education and training robots. In 2008, about 940,000 vacuum-cleaning robots were sold (almost 50% more than in 2007) while more than 21,000 lawn-mowing robots were sold. Other service robots for personal and private use with fast-emerging markets include robots for handicap assistance, robots for personal transportation, and home security and surveillance robots. The IFR projects that approximately 11.6 million service robots will be sold during the period 2009–2012. This figure includes some 4.8 million domestic robots and about 6.8 million entertainment and leisure robots. As such, service robots are now considered the most common form of robot, in sharp contrast to industrial robots being the most common back in 1960s.

Mobile robots open up many potential opportunities; at the same time, they introduce challenges in terms of their limited sensing capability and accuracy and minimal on-board computing resources. These limitations are a result of two factors: 1) these robots are typically mass-produced and they are often equipped with cheap and low-end sensors and computing power so as to attain an affordable market price for the general and price-sensitive consumers and 2) they are usually small, lightweight, and low-powered (e.g. battery-operated) machines thus significantly limiting the sensors and computing resources that can be installed and used. Nevertheless, mobile service robots are still expected to accomplish their goals or tasks with whatever modest computing resources and sensors that they may have.

In this dissertation, we consider three important problems in mobile robotics and demonstrate our approach to each of the problems with a mobile robot equipped with low-cost

sensors. The problems we consider are those of mobile robot calibration, mobile robot localization, and simultaneous localization and mapping by a mobile robot. In the next three sections, we separately introduce the problems we address and briefly discuss the approaches we take for each problem.

1.2 Mobile Robot Calibration

Mobile robot calibration is the problem of identifying the parameters that describe the kinematic and perceptual processes of a mobile robot. Robot calibration is an important step in robotics since a calibrated robot performs better than an uncalibrated one. However, the parameters are typically provided and hand tuned by a human operator and those are often derived from intensive and careful calibration experiments as well as the operator's knowledge and experience with the robot and its operating environment. Since the parameters are influenced by the robot's characteristics and environmental properties, there is a need for manual recalibration whenever there is a significant change in the robot or the environment.

In Chapter 3, we demonstrate how the parameters of both the kinematic and perceptual models of a mobile robot can be automatically estimated during its normal operation via machine learning methods thereby eliminating the necessity of manually identifying the parameters through a laborious calibration process. We assume that the robot has access to the map of the environment as well as the historical noisy account of its motion (i.e. odometry information) and perception (i.e. sonar readings). We use the expectation maximization

[Dempster et al., 1977] approach to estimate the parameters of the robot. To obtain the likely trajectory of the robot through the environment, we perform particle filtering [Arulampalam et al., 2002] and smoothing [Doucet et al., 2000b, Godsill et al., 2004] using the sensor data obtained by the robot during its normal operation. We then compute the maximum likelihood estimates of the parameters given the estimated robot trajectory and actual data. Results from real-world robotic experiments are presented that show the effectiveness of the estimation approach.

1.3 Mobile Robot Localization

In order for a mobile robot to be able to navigate safely and successfully in a given environment and accomplish its goals, it is necessary that it knows its position and orientation within the environment. For example, a mail delivery service robot for an office environment cannot properly pick up and drop off mail and avoid bumping into walls if it does not know where it is and when it has arrived at its intended destination. Determining the position and orientation of the robot within its environment is known as the *mobile robot localization problem*. Robot localization is considered to be one of the most fundamental problems in mobile robotics [Cox and Wilfong, 1990, Cox, 1991, Borenstein et al., 1996] and it can be described by the question *"Where am I?"* Correct localization of the robot is crucial since wrong estimates for the position and orientation of the robot can lead to erroneous behavior (e.g. entering the wrong room or going to the wrong floor in the building) and even hazardous outcomes

(e.g. colliding with objects or obstacles in the environment) thereby eventually preventing the robot from achieving its goals (e.g. not being able to deliver mail, which can cause further problems, and so on). For the robot to localize itself within its operating environment, it is usually provided with a map of its environment beforehand and it is equipped with sensors to perceive itself and its environment. Using the map of the environment and observations (measurements) from the sensors, the robot determines its position and orientation.

In Chapter 4, we present an approach to real-time pose estimation for a small-scale indoor mobile robot equipped with wheel encoders for its odometry and aided by a standard perspective camera without an *a priori* map of the environment. We apply particle filtering for processing measurements provided by the robot's odometry and observations of static three-dimensional point features or landmarks from the environment by the camera. We propose an appropriate sensor model for computing the weights of the particles in the filter. We validate the effectiveness of our approach extensively with both simulation as well as real-world data and compare its performance against that of the extended Kalman filter (EKF). Results from the tests show that the particle filter is better than the EKF in terms of the root mean squared error for the simulation data and it is capable of achieving good localization accuracy in unmodified indoor environments in real time.

1.4 Simultaneous Localization and Mapping

In many application areas, a mobile robot needs to be able to navigate safely and successfully around an unknown environment of which it has no *a priori* map nor does it know its initial position and orientation in the environment. For instance, a mass-produced mobile robot for housekeeping should be able to operate successfully in virtually any household environment. In this case, the robot needs to generate a map (representation) of its environment from scratch while simultaneously localizing itself relative to the map from its sensor data. The preceding problem is known in the mobile robotics community as the *simultaneous localization and mapping problem* [Leonard and Durrant-Whyte, 1991b, Thrun et al., 2005, Durrant-Whyte and Bailey, 2006, Bailey and Durrant-Whyte, 2006], commonly abbreviated as *SLAM*, and it was introduced by Smith and Cheeseman [1986] and Smith et al. [1990]. SLAM is also known as *concurrent mapping and localization* or *CML* [Thrun et al., 2000, Leonard et al., 2002, Tardós et al., 2002, Lorenzo et al., 2004, Thrun et al., 2005, Durrant-Whyte and Bailey, 2006]. SLAM is a complex problem as it intertwines the problems of localization and mapping. For the robot to correctly estimate its position and orientation (i.e. localization), it needs an accurate map of its surroundings, but to generate an accurate map (i.e. mapping), it requires correct estimates for its position and orientation. A solution to the SLAM problem is regarded as a "holy grail" [Dissanayake et al., 2001, Durrant-Whyte and Bailey, 2006, Abrate et al., 2007] for the mobile robotics community as it would pave the way for building truly autonomous mobile robots.

In order to accomplish SLAM, the robot is usually equipped with sensors (e.g. wheel encoders, range sensors, cameras) that allow it to observe (though only partially and inaccurately) the state of the world including itself. While the SLAM problem is inherently difficult and complex, the specific sensor used also contributes to the hardness of SLAM. SLAM is a well-studied problem and has become one of the mainstream research areas in mobile robotics. The majority of the proposed techniques for SLAM rely on the use of accurate and dense measurements provided by laser rangefinders to correctly localize the robot and produce accurate and detailed maps of complex environments. Little work has been done on the use of low-cost but noisy and sparse sonar sensors for SLAM in large indoor environments involving large loops.

In Chapter 5, we present our approach to SLAM with sonar sensors by applying particle filtering and a line-segment-based map representation with an orthogonality assumption [Nguyen et al., 2006] to map indoor environments much larger and more challenging than those previously considered with sonar sensors. Results from robotic experiments demonstrate that it is possible to produce good maps of large indoor environments with large loops despite the inherent limitations of sonar sensors.

1.5 Contributions

Our contributions can be summarized as follows:

- We propose an automated technique for the simultaneous calibration of both the motion and sensor models of a mobile robot from data without the need for specific calibration experiments and special calibration setup [Yap and Shelton, 2008]. This allows the robot to perform calibration with little or no human intervention and the calibration can be done during its normal operation.

- We present an approach to real-time pose estimation for a small-scale indoor mobile robot equipped with wheel encoders for its odometry and aided by a standard perspective camera without an *a priori* map of the environment.[1] This readily endows the robot with localization capability with the use of a cheap, lightweight, and common sensor.

- We show through extensive experiments that it is possible to produce good-quality maps of large indoor environments with large loops even with noisy and sparse sonar sensors [Yap and Shelton, 2009]. This serves as a step towards enabling small-scale robots in achieving full autonomy even with the use of cheap and less-powerful sensors such as sonars.

[1] This work is currently in preparation for submission.

Chapter 2

Background

The problems of mobile robot localization and mapping are often viewed as state estimation problems using sensor data. State estimation addresses the problem of estimating quantities from sensor data that are not directly observable, but that can be inferred [Thrun et al., 2005]. In mobile robot localization and mapping, the exact location of the robot and all the obstacles in the environment are quantities (or state variables) that are not directly measurable and they need to be estimated from sensor data. However, sensor data only provide partial information about those quantities and they are usually corrupted by noise. In this chapter, we review probabilistic state estimation algorithms which are at the heart of many current state-of-the-art robotic systems. These algorithms compute probability distributions (often called *belief distributions*) over possible values of the state variables using the available sensor data. In this chapter, our discussion focuses particularly on dynamic state estimation, i.e. estimating

the state of a system that changes over time, using a sequence of noisy sensor measurements made on the system.

We start by discussing the Bayes' filter, a recursive state estimation algorithm, that provides a framework for recursively computing probability distributions over the state variables as they evolve using sensor measurements that are received over time. We then discuss three concrete implementations of the Bayes' filter, namely the Kalman filter [Kalman, 1960], the extended Kalman filter (EKF), and the particle filter [Arulampalam et al., 2002] (which is the technique used in this dissertation). Welch and Bishop [2006] provides a good introduction to the Kalman filter and EKF.

2.1 Bayes' Filter

In order to discuss the Bayes' filter, we first define the dynamic state estimation problem (also called the *tracking problem*). Consider a system with state vector $\mathbf{x}$ that evolves over time according to

$$\mathbf{x}_t = \mathbf{f}_t \left(\mathbf{x}_{t-1}, \mathbf{c}_t, \mathbf{v}_t \right) \ , \tag{2.1}$$

where $\mathbf{f}_t$ is a (possibly) nonlinear function of the previous state $\mathbf{x}_{t-1}$ at time $t-1$, the control or system input $\mathbf{c}_t$ at time t, and the process noise vector $\mathbf{v}_t$. $\mathbf{v}_t$ (for $t = 1, 2, ...$) is a sequence of independent and identically distributed (i.i.d.) process disturbances. t is the discrete time index ($t = 1, 2, ...$). The goal of tracking is to recursively estimate the state $\mathbf{x}_t$ from sensor

measurement

$$\mathbf{s}_t = \mathbf{h}_t\left(\mathbf{x}_t, \mathbf{n}_t\right) \ , \tag{2.2}$$

where $\mathbf{h}_t$ is a (possibly) nonlinear function of the current state $\mathbf{x}_t$ at time t and measurement noise vector $\mathbf{n}_t$. $\{\mathbf{n}_t\}$ is a sequence of i.i.d. measurement noises. In filtering, the goal is to estimate the state $\mathbf{x}_t$ based on the set of all available controls $\mathbf{c}_{1:t} \triangleq (\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_t)$ and measurements $\mathbf{s}_{1:t} \triangleq (\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_t)$ up to time t, assuming controls and measurements arrive starting at time 1.

The Bayesian approach to dynamic state estimation recursively computes the posterior probability density function (pdf) (or probability distribution for discrete random variables) $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ over the possible values of the state vector $\mathbf{x}_t$ given all the available control $\mathbf{c}_{1:t}$ and sensor data $\mathbf{s}_{1:t}$ up to time t. The pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ represents the belief distribution at time t. It is assumed that the initial pdf of the state vector (also called the *prior*) $p\left(\mathbf{x}_0\right)$ at time 0 is known or available.

Given the pdf $p\left(\mathbf{x}_{t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}\right)$ at time $t-1$, the Bayes' filter computes the pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ at time t in two steps. In the first step, called the *prediction step*, it computes the pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)$ using the system model defined in Equation 2.1 and the Chapman-Kolmogorov equation as follows.

$$p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right) = \int p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right) p\left(\mathbf{x}_{t-1}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right) d\mathbf{x}_{t-1} \tag{2.3}$$

$$= \int p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t\right) p\left(\mathbf{x}_{t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}\right) d\mathbf{x}_{t-1} \ , \tag{2.4}$$

where $p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right) = p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t\right)$ since the system is Markovian of order one according to Equation 2.1.[1] $p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t\right)$ is called the *probabilistic system model* as defined by Equation 2.1 and the known statistics of the process noise vector $\mathbf{v}_t$. The pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)$ is often called the *prior pdf* at time t to emphasize the fact that it is the pdf *before* the latest measurement $\mathbf{s}_t$ at time t is considered. In the second step, called the *update step*, upon receipt of measurement $\mathbf{s}_t$, the Bayes' filter computes the pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ using the prior pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)$ at time t and the measurement model defined in Equation 2.2 using Bayes' theorem[2]:

$$p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right) = \frac{p\left(\mathbf{s}_t|\mathbf{x}_t, \mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right) p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)}{p\left(\mathbf{s}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)} \tag{2.5}$$

$$= \frac{p\left(\mathbf{s}_t|\mathbf{x}_t\right) p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)}{p\left(\mathbf{s}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)} \ , \tag{2.6}$$

where $p\left(\mathbf{s}_t|\mathbf{x}_t\right)$ is called the *probabilistic observation model* as defined by Equation 2.2 and the known statistics of the measurement noise vector $\mathbf{n}_t$. The pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ is often called the *posterior pdf* at time t to signify that it is the pdf *after* the latest measurement $\mathbf{s}_t$ at time t has been incorporated. The denominator $p\left(\mathbf{s}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)$ in Equation 2.6 is a

[1] The Markov assumption is both important and convenient as it greatly reduces both time and space complexities and makes the filtering process computationally tractable.

[2] Bayes' theorem or Bayes' rule states that $p\left(x|y\right) = \frac{p\left(y|x\right) p\left(x\right)}{p\left(y\right)}$ when $p\left(y\right) \neq 0$.

normalizing constant and can be expanded as

$$p\left(\mathbf{s}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right) = \int p\left(\mathbf{s}_t|\mathbf{x}_t, \mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right) p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right) d\mathbf{x}_t \tag{2.7}$$

$$= \int p\left(\mathbf{s}_t|\mathbf{x}_t\right) p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right) d\mathbf{x}_t \ , \tag{2.8}$$

where $p\left(\mathbf{s}_t|\mathbf{x}_t, \mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right) = p\left(\mathbf{s}_t|\mathbf{x}_t\right)$ since the measurement model defined in Equation 2.2 assumes that the measurement $\mathbf{s}_t$ is conditionally independent of the controls $\mathbf{c}_{1:t}$ and previous measurements $\mathbf{s}_{1:t-1}$ given the state $\mathbf{x}_t$ at time t.

Equations 2.4 and 2.6 are used in the Bayes' filter to recursively compute the posterior pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ at time t from the pdf $p\left(\mathbf{x}_{t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}\right)$ at time $t-1$. The Bayes' filter algorithm is shown in Algorithm 1 where $\eta = p\left(\mathbf{s}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)^{-1}$ is a normalizing constant to ensure that the pdf $p\left(\mathbf{x}_{1:t}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ is a valid one.

Algorithm 1 Bayes' Filter

Input:
$p\left(\mathbf{x}_{t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}\right)$: pdf at time $t-1$
$\mathbf{c}_t$: control at time t
$\mathbf{s}_t$: measurement at time t
Output:
$p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$: posterior pdf at time t
Process:
for all $\mathbf{x}_t$ do
$p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right) \leftarrow \int p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t\right) p\left(\mathbf{x}_{t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}\right) d\mathbf{x}_{t-1}$
$p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right) \leftarrow \eta p\left(\mathbf{s}_t|\mathbf{x}_t\right) p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)$
end

Note that in order to perform Bayes' filtering, three probabilistic models must be known: the initial pdf $p\left(\mathbf{x}_0\right)$ at time 0, the system model $p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t\right)$, and the observation model

$p(\mathbf{s}_t|\mathbf{x}_t)$. These models are highly application specific and in Chapter 3 we discuss how to learn the parameters of the system model $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)$ and the measurement model $p(\mathbf{s}_t|\mathbf{x}_t)$ from data of an actual robot system.

The Bayes' filter provides a conceptual framework for recursive state estimation and it is the basis for the Kalman filter, EKF, particle filter, and other filters. Note that, given the system and measurement models of a process, the Bayes' filter provides the optimal way of computing the pdf over the state of the process. However, to implement the Bayes' filter, we need to be able to perform the integration as well as the product of probability distributions in closed form, which can be intractable for large state spaces and arbitrary probability distributions. Existing methods implementing the Bayes' filter either restrict themselves to finite state spaces or they make certain assumptions about the models and the forms of the probability distributions.

2.2 Kalman Filter

The Kalman filter, introduced by Kalman [1960], is a concrete implementation of the Bayes' filter. It assumes that the pdf $p(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t})$ at every time step t is Gaussian parametrized by a mean vector $\mu_{t|t}$ and covariance matrix $\mathbf{P}_{t|t}$, i.e. $p(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}) = \mathcal{N}\left(\mathbf{x}_t; \mu_{t|t}, \mathbf{P}_{t|t}\right)$.[3] The mean $\mu_{t|t}$ represents the most likely (best) estimate of the state of the system given all control data $\mathbf{c}_{1:t}$ and sensor data $\mathbf{s}_{1:t}$ up to time t while the covariance matrix $\mathbf{P}_{t|t}$ represents the un-

[3]The notations $\mu_{t_1|t_2}$ and $\mathbf{P}_{t_1|t_2}$ are used to denote the mean and covariance of $p(\mathbf{x}_{t_1}|\mathbf{c}_{1:t_1}, \mathbf{s}_{1:t_2})$, respectively. $\mathcal{N}(\mathbf{x}; \mu, \mathbf{P})$ denotes a multivariate Gaussian distribution over the variable $\mathbf{x}$ with mean vector μ and covariance matrix $\mathbf{P}$.

certainty in the estimate of the filter. If $p\left(\mathbf{x}_{t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}\right)$ is Gaussian, then $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ is also Gaussian given that the following conditions hold:

- The function $\mathbf{f}_t\left(\mathbf{x}_{t-1}, \mathbf{c}_t, \mathbf{v}_t\right)$ is a known linear function of $\mathbf{x}_{t-1}$, $\mathbf{c}_t$, and $\mathbf{v}_t$.
- The function $\mathbf{h}_t\left(\mathbf{x}_t, \mathbf{n}_t\right)$ is a known linear function of $\mathbf{x}_t$ and $\mathbf{n}_t$.
- $\mathbf{v}_t$ and $\mathbf{n}_t$ are drawn from independent Gaussian distributions of known means and covariances.

Let $\mathbf{F}_t$ and $\mathbf{G}_t$ be matrices that define the linear function $\mathbf{f}_t\left(\mathbf{x}_{t-1}, \mathbf{c}_t, \mathbf{v}_t\right)$, i.e. Equation 2.1 can be rewritten as

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{G}_t\mathbf{c}_t + \mathbf{v}_t \ . \tag{2.9}$$

Similarly, let $\mathbf{H}_t$ be a matrix that defines the linear function $\mathbf{h}_t\left(\mathbf{x}_t, \mathbf{n}_t\right)$, i.e. Equation 2.2 can be rewritten as

$$\mathbf{s}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{n}_t \ . \tag{2.10}$$

Let $\mathbf{v}_t$ and $\mathbf{n}_t$ be independent zero-mean Gaussians with covariances $\mathbf{Q}_t$ and $\mathbf{R}_t$, respectively.

At initialization, $\mu_{0|0} = \mathbb{E}[\mathbf{x}_0]$ with covariance matrix $\mathbf{P}_{0|0}$. Given the mean $\mu_{t-1|t-1}$ and covariance $\mathbf{P}_{t-1|t-1}$ of $p\left(\mathbf{x}_{t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}\right)$ at time $t-1$, the Kalman filter computes the mean $\mu_{t|t}$ and covariance $\mathbf{P}_{t|t}$ of $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ at time t in two steps: prediction and update. In the prediction step, the Kalman filter computes the mean $\mu_{t|t-1}$ and covariance $\mathbf{P}_{t|t-1}$ of

the prior pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)$ at time t as follows.

$$\mu_{t|t-1} = \mathbf{F}_t \mu_{t-1|t-1} + \mathbf{G}_t \mathbf{c}_t \tag{2.11}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^T + \mathbf{Q}_t \ . \tag{2.12}$$

As can be seen in Equation 2.11, to compute the mean $\mu_{t|t-1}$ of the prior pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)$ at time t, the Kalman filter uses the deterministic version of the linear system function in Equation 2.9 (i.e. $\mathbf{x}_t = \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{G}_t \mathbf{c}_t$) but with $\mu_{t-1|t-1}$ substituted for $\mathbf{x}_{t-1}$ and the control input $\mathbf{c}_t$ received at time t. The covariance matrix $\mathbf{P}_{t|t-1}$ of the pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)$ is computed by multiplying the linear matrix $\mathbf{F}_t$ into the previous covariance $\mathbf{P}_{t-1|t-1}$ twice (since the next state depends on the previous state through the matrix $\mathbf{F}_t$ and the covariance is a quadratic matrix) and adding the covariance $\mathbf{Q}_t$ of the process noise $\mathbf{v}_t$. As a result, the uncertainty represented by the predicted covariance $\mathbf{P}_{t|t-1}$ is larger than the uncertainty represented by the prior covariance $\mathbf{P}_{t-1|t-1}$.

In the update step, upon receipt of the measurement $\mathbf{s}_t$, the Kalman filter first computes the innovation or measurement residual $\mathbf{r}_t$ and its associated covariance $\mathbf{S}_t$ at time t:

$$\mathbf{r}_t = \mathbf{s}_t - \mathbf{H}_t \mu_{t|t-1} \tag{2.13}$$

$$\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_{t|t-1} \mathbf{H}_t^T + \mathbf{R}_t \ . \tag{2.14}$$

The measurement residual $\mathbf{r}_t$ reflects the discrepancy between the actual measurement $\mathbf{s}_t$ and the predicted measurement $\mathbf{H}_t\mu_{t|t-1}$ (which is the best prediction of what the measurement will be at time t). The Kalman filter then computes the Kalman gain $\mathbf{K}_t$:

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T\mathbf{S}_t^{-1} \ . \tag{2.15}$$

The Kalman gain $\mathbf{K}_t$ specifies the degree to which the measurement $\mathbf{s}_t$ is incorporated into the new state estimate $\mu_{t|t}$ and affects the covariance $\mathbf{P}_{t|t}$ at time t. The Kalman filter computes the mean $\mu_{t|t}$ and covariance $\mathbf{P}_{t|t}$ of the posterior pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ at time t as follows.

$$\mu_{t|t} = \mu_{t|t-1} + \mathbf{K}_t\mathbf{r}_t \tag{2.16}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}_t\mathbf{H}_t\mathbf{P}_{t|t-1} \ . \tag{2.17}$$

Note that in Equation 2.15, the Kalman gain $\mathbf{K}_t$ is proportional to the predicted covariance $\mathbf{P}_{t|t-1}$ and inversely proportional to the covariance $\mathbf{S}_t$ of the measurement. Thus, in Equation 2.16, if the measurement $\mathbf{s}_t$ is much more uncertain than the predicted estimate $\mu_{t|t-1}$, the correction term $\mathbf{K}_t\mathbf{r}_t$ is very small, and as such, the measurement $\mathbf{s}_t$ has little effect on the computation of the new state estimate $\mu_{t|t}$, and $\mu_{t|t}$ is mainly equal to the predicted estimate $\mu_{t|t-1}$. On the other hand, if the predicted estimate $\mu_{t|t-1}$ is much more uncertain than the measurement $\mathbf{s}_t$, the measurement $\mathbf{s}_t$ is used to a larger degree in updating the predicted estimate $\mu_{t|t-1}$ to derive the new state estimate $\mu_{t|t}$.

Now, we consider how the measurement $\mathbf{s}_t$ affects the covariance $\mathbf{P}_{t|t}$ at time t. Using Equation 2.15 in Equation 2.17, we get

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\mathbf{H}_t^T\mathbf{S}_t^{-1}\mathbf{H}_t\mathbf{P}_{t|t-1} \ . \tag{2.18}$$

From Equation 2.18, we observe that if the measurement $\mathbf{s}_t$ is very uncertain (i.e. $\mathbf{S}_t$ is large), the covariance $\mathbf{P}_{t|t}$ is decreased from $\mathbf{P}_{t|t-1}$ by only a small amount. In other words, the measurement contributes very little in reducing the estimation uncertainty if it is highly uncertain (or unreliable). On the other hand, if the measurement is very precise (i.e. $\mathbf{S}_t$ is small), the covariance $\mathbf{P}_{t|t}$ is decreased from $\mathbf{P}_{t|t-1}$ considerably. This means that precise (or reliable) measurement contributes significantly in reducing the estimation uncertainty.

Algorithm 2 shows the Kalman filter algorithm using Equations 2.11 to 2.17. For linear Gaussian systems such as that described by Equations 2.9 and 2.10, the Kalman filter provides the optimal state estimate for the tracking problem and it is therefore an exact implementation of the Bayes' filter. As can be seen, at every time step t, we only need to keep track of the mean vector $\mu_{t|t}$ and the associated covariance matrix $\mathbf{P}_{t|t}$ of the state estimate as they are the parameters of the Gaussian distribution $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$.

2.3 Extended Kalman Filter

The assumption that the system model in Equation 2.1 and the measurement model in Equation 2.2 are linear functions (called the *linearity assumption*) is crucial for the correctness

Algorithm 2 Kalman Filter

Input:

$\mu_{t-1|t-1}$, $\mathbf{P}_{t-1|t-1}$: mean and covariance of $p\left(\mathbf{x}_{t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}\right)$, respectively

$\mathbf{c}_t$: control at time t

$\mathbf{s}_t$: measurement at time t

Output:

$\mu_{t|t}$, $\mathbf{P}_{t|t}$: mean and covariance of $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$, respectively

Process:

$\mu_{t|t-1} \leftarrow \mathbf{F}_t \mu_{t-1|t-1} + \mathbf{G}_t \mathbf{c}_t$

$\mathbf{P}_{t|t-1} \leftarrow \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^T + \mathbf{Q}_t$

$\mathbf{r}_t \leftarrow \mathbf{s}_t - \mathbf{H}_t \mu_{t|t-1}$

$\mathbf{S}_t \leftarrow \mathbf{H}_t \mathbf{P}_{t|t-1} \mathbf{H}_t^T + \mathbf{R}_t$

$\mathbf{K}_t \leftarrow \mathbf{P}_{t|t-1} \mathbf{H}_t^T \mathbf{S}_t^{-1}$

$\mu_{t|t} \leftarrow \mu_{t|t-1} + \mathbf{K}_t \mathbf{r}_t$

$\mathbf{P}_{t|t} \leftarrow \mathbf{P}_{t|t-1} - \mathbf{K}_t \mathbf{H}_t \mathbf{P}_{t|t-1}$

of the Kalman filter. However, the linearity assumption can be limiting and may not hold in practice (as many non-trivial real-world systems are not linear). The EKF relaxes the linearity assumption, i.e. the system and observation models need not be linear functions of the state, but requires that the models be differentiable functions.

In the EKF, the system and measurement models are given in Equations 2.19 and 2.20, respectively. [4]

$$\mathbf{x}_t = \mathbf{f}_t\left(\mathbf{x}_{t-1}, \mathbf{c}_t\right) + \mathbf{v}_t \tag{2.19}$$

$$\mathbf{s}_t = \mathbf{h}_t\left(\mathbf{x}_t\right) + \mathbf{n}_t \ . \tag{2.20}$$

[4] In general, $\mathbf{x}_t$ and $\mathbf{s}_t$ can also depend nonlinearly on $\mathbf{v}_t$ and $\mathbf{n}_t$, respectively, although we do not do so here for ease of discussion.

The pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ at time t is approximated by a Gaussian with mean $\mu_{t|t}$ and covariance $\mathbf{P}_{t|t}$, i.e. $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right) \approx \mathcal{N}\left(\mathbf{x}_t; \mu_{t|t}, \mathbf{P}_{t|t}\right)$, in the EKF. Similar to the Kalman filter, at initialization, $\mu_{0|0} = \mathbb{E}[\mathbf{x}_0]$ with covariance $\mathbf{P}_{0|0}$. Given the mean $\mu_{t-1|t-1}$ and covariance $\mathbf{P}_{t-1|t-1}$ of $p\left(\mathbf{x}_{1:t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}\right)$ at time $t-1$, the EKF computes the mean $\mu_{t|t}$ and covariance $\mathbf{P}_{t|t}$ of $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ at time t in two steps (i.e. prediction and update). In the prediction step, it computes the mean $\mu_{t|t-1}$ and covariance $\mathbf{P}_{t|t-1}$ of $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1}\right)$ at time t as follows.

$$\mu_{t|t-1} = \mathbf{f}_t\left(\mu_{t-1|t-1}, \mathbf{c}_t\right) \tag{2.21}$$

$$\mathbf{P}_{t|t-1} = \hat{\mathbf{F}}_t \mathbf{P}_{t-1|t-1} \hat{\mathbf{F}}_t^T + \mathbf{Q}_t \ , \tag{2.22}$$

where

$$\hat{\mathbf{F}}_t = \left.\frac{\partial \mathbf{f}_t}{\partial \mathbf{x}_{t-1}}\right|_{\mu_{t-1|t-1}, \mathbf{c}_t} . \tag{2.23}$$

In the update step, it computes the measurement residual $\mathbf{r}_t$ and its associated covariance $\mathbf{S}_t$, the Kalman gain $\mathbf{K}_t$, and the mean $\mu_{t|t}$ and covariance $\mathbf{P}_{t|t}$ of $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ at time t as follows:

$$\mathbf{r}_t = \mathbf{s}_t - \mathbf{h}_t\left(\mu_{t|t-1}\right) \tag{2.24}$$

$$\mathbf{S}_t = \hat{\mathbf{H}}_t \mathbf{P}_{t|t-1} \hat{\mathbf{H}}_t^T + \mathbf{R}_t \tag{2.25}$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \hat{\mathbf{H}}_t^T \mathbf{S}_t^{-1} \tag{2.26}$$

$$\mu_{t|t} = \mu_{t|t-1} + \mathbf{K}_t \mathbf{r}_t \tag{2.27}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}_t \hat{\mathbf{H}}_t \mathbf{P}_{t|t-1} \ , \tag{2.28}$$

where

$$\hat{\mathbf{H}}_t = \left. \frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_t} \right|_{\mu_{t|t-1}} \ . \tag{2.29}$$

$\hat{\mathbf{F}}_t$ and $\hat{\mathbf{H}}_t$ are the Jacobian matrices of the system and observation functions, respectively. The EKF is given in Algorithm 3. As can be seen in Algorithm 3, the EKF algorithm is very similar to the Kalman filter algorithm in Algorithm 2 except that it uses the nonlinear system and measurement models for computing the predicted mean $\mu_{t|t-1}$ and predicted (or expected) measurement $\mathbf{h}_t\left(\mu_{t|t-1}\right)$, respectively, and utilizes the Jacobian matrices $\hat{\mathbf{F}}_t$ and $\hat{\mathbf{H}}_t$ in place of the matrices $\mathbf{F}_t$ and $\mathbf{G}_t$ in the Kalman filter.

Algorithm 3 Extended Kalman Filter

Input:
$\mu_{t-1|t-1}$, $\mathbf{P}_{t-1|t-1}$: mean and covariance of $p\left(\mathbf{x}_{t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}\right)$, respectively
$\mathbf{c}_t$: control at time t
$\mathbf{s}_t$: measurement at time t
Output:
$\mu_{t|t}$, $\mathbf{P}_{t|t}$: mean and covariance of $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$, respectively
Process:
$\mu_{t|t-1} \leftarrow \mathbf{f}_t\left(\mu_{t-1|t-1}, \mathbf{c}_t\right)$
$\mathbf{P}_{t|t-1} \leftarrow \hat{\mathbf{F}}_t \mathbf{P}_{t-1|t-1} \hat{\mathbf{F}}_t^T + \mathbf{Q}_t$
$\mathbf{r}_t \leftarrow \mathbf{s}_t - \mathbf{h}_t\left(\mu_{t|t-1}\right)$
$\mathbf{S}_t \leftarrow \hat{\mathbf{H}}_t \mathbf{P}_{t|t-1} \hat{\mathbf{H}}_t^T + \mathbf{R}_t$
$\mathbf{K}_t \leftarrow \mathbf{P}_{t|t-1} \hat{\mathbf{H}}_t^T \mathbf{S}_t^{-1}$
$\mu_{t|t} \leftarrow \mu_{t|t-1} + \mathbf{K}_t \mathbf{r}_t$
$\mathbf{P}_{t|t} \leftarrow \mathbf{P}_{t|t-1} - \mathbf{K}_t \hat{\mathbf{H}}_t \mathbf{P}_{t|t-1}$

The EKF described uses the first-order Taylor series expansion[5] of the nonlinear functions, i.e.

$$\mathbf{f}_t\left(\mathbf{x}_{t-1}, \mathbf{c}_t\right) \approx \mathbf{f}_t\left(\mu_{t-1|t-1}, \mathbf{c}_t\right) + \left.\frac{\partial \mathbf{f}_t}{\partial \mathbf{x}_{t-1}}\right|_{\mu_{t-1|t-1}, \mathbf{c}_t} \left(\mathbf{x}_{t-1} - \mu_{t-1|t-1}\right) \tag{2.30}$$

$$= \mathbf{f}_t\left(\mu_{t-1|t-1}, \mathbf{c}_t\right) + \hat{\mathbf{F}}_t\left(\mathbf{x}_{t-1} - \mu_{t-1|t-1}\right) \tag{2.31}$$

$$\mathbf{h}_t\left(\mathbf{x}_t\right) \approx \mathbf{h}_t\left(\mu_{t|t-1}\right) + \left.\frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_t}\right|_{\mu_{t|t-1}} \left(\mathbf{x}_t - \mu_{t|t-1}\right) \tag{2.32}$$

$$= \mathbf{h}_t\left(\mu_{t|t-1}\right) + \hat{\mathbf{H}}_t\left(\mathbf{x}_t - \mu_{t|t-1}\right) \ . \tag{2.33}$$

Of course, the accuracy of the above approximations is dependent on the degree of nonlinearity of the functions involved. Unlike the Kalman filter, the EKF is, in general, not an optimal estimator and it is not an exact implementation of the Bayes' filter. Nevertheless, the EKF has become the *de facto* technique for dynamic state estimation problems and it performs reasonably well in practice. Note that if the system and measurement functions are linear, then the EKF becomes the standard Kalman filter.

---

[5]The Taylor series expansion of a function $\mathbf{f}\left(\mathbf{x}\right)$ that is infinitely differentiable in a neighborhood of $\mathbf{a}$ is $\mathbf{f}\left(\mathbf{a}\right) + \frac{\mathbf{f}^{(1)}\left(\mathbf{a}\right)}{1!}\left(\mathbf{x}-\mathbf{a}\right) + \frac{\mathbf{f}^{(2)}\left(\mathbf{a}\right)}{2!}\left(\mathbf{x}-\mathbf{a}\right)^2 + \frac{\mathbf{f}^{(3)}\left(\mathbf{a}\right)}{3!}\left(\mathbf{x}-\mathbf{a}\right)^3 + ...$, or more compactly as $\sum_{n=0}^{\infty} \frac{\mathbf{f}^{(n)}\left(\mathbf{a}\right)}{n!}\left(\mathbf{x}-\mathbf{a}\right)^n$, where $\mathbf{f}^{(n)}\left(\mathbf{a}\right)$ is the nth derivative of $\mathbf{f}$ evaluated at $\mathbf{a}$, $n!$ is the factorial of n, $\mathbf{f}^{(0)}\left(\mathbf{a}\right) = \mathbf{f}\left(\mathbf{a}\right)$, and $0! = 1$.

2.4 Particle Filter

The particle filter is an alternative implementation of the Bayes' filter. Unlike the Kalman filter and EKF, it does not assume that the pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ at every time step is a Gaussian. The key idea of the particle filter is to represent the pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ by a set of random samples (also called *particles*, hence the name) with associated weights. Let $\mathcal{X}_t = \left\{ \left(\mathbf{x}_t^{[i]}, w_t^{[i]}\right) : i = 1, 2, ..., N_\mathrm{s} \right\}$ be a random measure characterizing the pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$, where $\left\{\mathbf{x}_t^{[i]} : i = 1, 2, ..., N_\mathrm{s}\right\}$ is a set of support points (samples) with associated weights $\left\{w_t^{[i]} : 0 \leq w_t^{[i]}, i = 1, 2, ..., N_\mathrm{s}\right\}$, and N_s is the number of samples or particles in the particle filter. Each particle $\mathbf{x}_t^{[i]}$ is a concrete instantiation (or hypothesis) of the state at time t. The weights are (usually) normalized, i.e. $\sum_{i=1}^{N_\mathrm{s}} w_t^{[i]} = 1$. The pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ at time t is approximated as

$$p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right) \approx \sum_{i=1}^{N_\mathrm{s}} w_t^{[i]} \delta\left(\mathbf{x}_t - \mathbf{x}_t^{[i]}\right) \ , \tag{2.34}$$

where $\delta(.)$ is the Dirac delta function[6]. The set $\mathcal{X}_t$, therefore, is a discrete weighted approximation of $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$. The weights of the particles $w_t^{[i]}$ are computed using the principle of importance sampling [Arulampalam et al., 2002, Thrun et al., 2005] which we discuss in Section 2.4.1.

Similar to the Kalman filter and EKF, the particle filter computes the pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ at time t from the pdf $p\left(\mathbf{x}_{t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}\right)$ at time $t-1$. At initialization, we assume that

[6]The Dirac delta function $\delta\left(x\right) = 0$ if $x \neq 0$ and $\delta\left(x\right) = \infty$ if $x = 0$ with $\int_a^b \delta\left(x\right) dx = 1$ if $0 \in [a, b]$ and $\int_a^b \delta\left(x\right) dx = 0$ if $0 \notin [a, b]$.

the prior $p(\mathbf{x}_0)$ is represented by the set of weighted particles $\mathcal{X}_0 = \left\{ \left(\mathbf{x}_0^{[i]}, w_0^{[i]} \right) : w_0^{[i]} = \frac{1}{N_\mathrm{s}}, i = 1, 2, ..., N_\mathrm{s} \right\}$ at time 0.[7] Since the pdfs are represented as sets of weighted particles, the particle filter computes the particle set $\mathcal{X}_t$ at time t from the set $\mathcal{X}_{t-1}$ at time $t-1$ with the following steps.

1. *Particle Updating*. For each particle $\mathbf{x}_{t-1}^{[i]}$ at time $t-1$, generate a particle $\mathbf{x}_t^{[i]}$ for time t by sampling from the probabilistic system model $p\left(\mathbf{x}_t \left| \mathbf{x}_{t-1}^{[i]}, \mathbf{c}_t \right.\right)$ given the control input $\mathbf{c}_t$. This step corresponds to the prediction step in the Bayes' filter, Kalman filter, and EKF. The set of particles $\left\{ \mathbf{x}_t^{[i]} : i = 1, 2, ..., N_\mathrm{s} \right\}$ obtained in this step approximates the prior pdf $p(\mathbf{x}_t | \mathbf{c}_{1:t}, \mathbf{s}_{1:t-1})$ at time t.

2. *Particle Weighting*. For each particle $\mathbf{x}_t^{[i]}$ at time t, compute its weight $w_t^{[i]}$ (also called the *importance factor*). The weight $w_t^{[i]}$ is computed as the likelihood of the measurement $\mathbf{s}_t$ given the particle $\mathbf{x}_t^{[i]}$ using the probabilistic observation model, i.e. $w_t^{[i]} = w_{t-1}^{[i]} p\left(\mathbf{s}_t \left| \mathbf{x}_t^{[i]} \right.\right)$. The weights, therefore, are used to incorporate the measurement $\mathbf{s}_t$ at time t into the particle filter. This step corresponds to the update step in the Bayes' filter, Kalman filter, and EKF. The set of weighted particles $\left\{ \left(\mathbf{x}_t^{[i]}, w_t^{[i]} \right) : i = 1, 2, ..., N_\mathrm{s} \right\}$ obtained at this step approximates the posterior pdf $p(\mathbf{x}_t | \mathbf{c}_{1:t}, \mathbf{s}_{1:t})$.

3. *Particle Resampling*. Given the set of weighted particles $\left\{ \left(\mathbf{x}_t^{[i]}, w_t^{[i]} \right) : i = 1, 2, ..., N_\mathrm{s} \right\}$, draw N_s particles with replacement from the set, with each particle $\mathbf{x}_t^{[i]}$ having the probability of being selected proportional to its importance weight $w_t^{[i]}$. Set the weights of

[7] At initialization, the particles can also be assigned with non-uniform weights to encode the prior knowledge that some initial states are more likely than the other states.

the particles drawn to $\frac{1}{N_s}$. Resampling transforms the set of N_s particles before the resampling step (representing the prior pdf $p(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1})$) into another set of N_s particles (representing the posterior pdf $p(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t})$). The resulting set of particles usually contain many duplicates.

Algorithm 4 provides the particle filter algorithm. The particular version of the particle filter shown in Algorithm 4 is called the *sequential importance resampling (SIR) particle filter* [Arulampalam et al., 2002]. It performs resampling at every iteration. The idea behind the resampling step is to remove particles that have small weights and replicate particles that have large weights. Particles that have large weights represent likely ("good") particles while those that are assigned small weights represent unlikely ("bad") particles.[8] However, performing resampling often can lead to loss of diversity among the particles (also called the *particle depletion problem* [Grisetti et al., 2005] or *sample impoverishment* [Arulampalam et al., 2002]). In order to reduce the risk of particle depletion, we perform resampling only when a severe degeneracy is observed in the particle filter: where after a few iterations, all but one particle has negligible weight. One way of measuring the degeneracy of the particle filter is the estimated effective sample size $\hat{N}_{\text{eff}}$ defined as

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N_s} \left(w_t^{[i]\prime}\right)^2}, \tag{2.35}$$

[8] "Good" particles are those that are consistent with the observations or evidence while "bad" particles are those that are not consistent with the observations or evidence.

where $w_t^{[i]'} = \dfrac{w_t^{[i]}}{\sum_{i=1}^{N_s} w_t^{[i]}}$ is the normalized weights of the ith particle at time t [Doucet et al., 2000a, Arulampalam et al., 2002, Grisetti et al., 2005]. Note that $\hat{N}_{\text{eff}} \leq N_s$ and we resample only when $\hat{N}_{\text{eff}}$ falls below a certain threshold N_τ. The particle filter with this selective resample scheme is given in Algorithm 5.

Algorithm 4 Particle Filter

Input:

$\mathcal{X}_{t-1} = \left\{ \left(\mathbf{x}_{t-1}^{[i]}, w_{t-1}^{[i]} \right) : i = 1, 2, ..., N_s \right\}$: Set of weighted particles at time $t-1$

$\mathbf{c}_t$: control at time t

$\mathbf{s}_t$: measurement at time t

Output:

$\mathcal{X}_t = \left\{ \left(\mathbf{x}_t^{[i]}, w_t^{[i]} \right) : i = 1, 2, ..., N_s \right\}$: Set of weighted particles at time t

Process:

$\bar{\mathcal{X}}_t \leftarrow \emptyset$, $\mathcal{X}_t \leftarrow \emptyset$

for $i \leftarrow 1$ to N_s do

  sample $\mathbf{x}_t^{[i]} \sim p\left(\mathbf{x}_t \middle| \mathbf{x}_{t-1}^{[i]}, \mathbf{c}_t \right)$

  $w_t^{[i]} \leftarrow p\left(\mathbf{s}_t \middle| \mathbf{x}_t^{[i]} \right)$

  $\bar{\mathcal{X}}_t \leftarrow \bar{\mathcal{X}}_t \cup \left(\mathbf{x}_t^{[i]}, w_t^{[i]} \right)$

end

for $i \leftarrow 1$ to N_s do

  draw j with probability $\propto w_t^{[j]}$

  $\mathcal{X}_t \leftarrow \mathcal{X}_t \cup \left(\mathbf{x}_t^{[j]}, \dfrac{1}{N_s} \right)$

end

The particle filter is also known by various names including the *sequential Monte Carlo method* [Doucet et al., 2000a], the *bootstrap filter* [Gordon et al., 1993], and the *CONDENSATION* (short for conditional density propagation) algorithm [Isard and Blake, 1998] in the computer vision community. Similar to the EKF, the particle filter is an approximate nonlinear Bayesian filter. However, as the number of samples $N_s \rightarrow \infty$, the approxima-

Algorithm 5 Particle Filter with Selective Resampling

Input:

$\mathcal{X}_{t-1} = \left\{ \left(\mathbf{x}_{t-1}^{[i]}, w_{t-1}^{[i]} \right) : i = 1, 2, ..., N_{\text{s}} \right\}$: Set of weighted particles at time $t-1$

$\mathbf{c}_t$: control at time t

$\mathbf{s}_t$: measurement at time t

Output:

$\mathcal{X}_t = \left\{ \left(\mathbf{x}_t^{[i]}, w_t^{[i]} \right) : i = 1, 2, ..., N_{\text{s}} \right\}$: Set of weighted particles at time t

Process:

$\bar{\mathcal{X}}_t \leftarrow \emptyset$, $\mathcal{X}_t \leftarrow \emptyset$

for $i \leftarrow 1$ to N_{s} do

  sample $\mathbf{x}_t^{[i]} \sim p\left(\mathbf{x}_t \middle| \mathbf{x}_{t-1}^{[i]}, \mathbf{c}_t \right)$

  $w_t^{[i]} \leftarrow w_{t-1}^{[i]} p\left(\mathbf{s}_t \middle| \mathbf{x}_t^{[i]} \right)$

  $\bar{\mathcal{X}}_t \leftarrow \bar{\mathcal{X}}_t \cup \left(\mathbf{x}_t^{[i]}, w_t^{[i]} \right)$

end

Compute $\hat{N}_{\text{eff}}$

if $\hat{N}_{\text{eff}} < N_\tau$

  for $i \leftarrow 1$ to N_{s} do

    draw j with probability $\propto w_t^{[j]}$

    $\mathcal{X}_t \leftarrow \mathcal{X}_t \cup \left(\mathbf{x}_t^{[j]}, \frac{1}{N_{\text{s}}} \right)$

  end

else

  $Z \leftarrow \sum_{i=1}^{N_{\text{s}}} w_t^{[i]}$

  for $i \leftarrow 1$ to N_{s} do

    $\mathcal{X}_t \leftarrow \mathcal{X}_t \cup \left(\mathbf{x}_t^{[i]}, Z^{-1} w_t^{[i]} \right)$

  end

end

tion (Equation 2.34) approaches the true posterior pdf $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ and thus the Bayesian filtered estimate. In practice, the number of particles used is often a large number (e.g. 1,000). Unlike the Kalman filter and EKF, the particle filter is not limited to Gaussian or uni-modal distributions as the set of particles can represent multi-modal or arbitrary distributions which can be important in some applications. Such a representation is approximate and nonparametric and it can represent a much broader space of distributions than Gaussians. Additionally, the particle filter can be applied to any system and measurement models and it is not restricted to the linearity or Gaussian noise assumptions of the Kalman filter and EKF. In fact, one of the advantages of the particle filter over the Kalman filter and EKF is its ability to model nonlinear transformation of random variables.

2.4.1 Importance Sampling

In this section, we briefly discuss the concept of importance sampling and how it is used for the weight computation in the particle filter.

The idea behind importance sampling is as follows. Suppose $p\left(x\right)$ is a probability density from which we want to obtain samples. However, sampling directly from $p\left(x\right)$ is difficult or impossible, but it can be evaluated (up to a proportionality constant). $p\left(x\right)$ is often called the *target distribution*. Now, suppose that there is a distribution $q\left(x\right)$ from which we can easily generate samples $\{x^{[i]} : x^{[i]} \sim q\left(x\right), i = 1, 2, ..., N_{\mathrm{s}}\}$. $q\left(x\right)$ is often referred to as the *proposal distribution* or *importance density*. The proposal $q\left(x\right)$ can be any arbitrary distribution as long as $p\left(x\right) > 0$ implies $q\left(x\right) > 0$. The preceding ensures that there is a

non-zero probability of generating a sample from $q(x)$ for any state that might be generated by sampling from $p(x)$. $p(x)$ can now be approximated by the set of weighted particles $x^{[i]}$ as follows.

$$p(x) \approx \frac{1}{W} \sum_{i=1}^{N_s} w^{[i]} \delta\left(x - x^{[i]}\right) \ , \tag{2.36}$$

where $W = \sum_{i=1}^{N_s} w^{[i]}$, and

$$w^{[i]} = \frac{p\left(x^{[i]}\right)}{q\left(x^{[i]}\right)} \tag{2.37}$$

is the normalized weight for the ith sample $x^{[i]}$.

To see how the weight update equation

$$w_t^{[i]} \propto w_{t-1}^{[i]} p\left(\mathbf{s}_t \left| \mathbf{x}_t^{[i]} \right.\right) \tag{2.38}$$

is derived, we let the particles in the particle filter be samples of state sequences, i.e.

$$\mathbf{x}_{0:t}^{[i]} \triangleq \left(\mathbf{x}_0^{[i]}, \mathbf{x}_1^{[i]}, ..., \mathbf{x}_t^{[i]}\right) \ . \tag{2.39}$$

To generate and maintain these samples of state sequences, we simply append $\mathbf{x}_t^{[i]}$ to the state sequence sample $\mathbf{x}_{0:t-1}^{[i]}$ from which it was generated. The particle filter, in this case, is used to approximate the posterior pdf $p(\mathbf{x}_{0:t}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t})$ over all state sequences by a weighted set of state sequence samples $\mathbf{x}_{0:t}^{[i]}$ instead of the posterior pdf $p(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t})$ over the current state.[9]

[9]Note that the pdf $p(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t})$ can be obtained from $p(\mathbf{x}_{0:t}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t})$ by simply marginalizing out the previous states $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{t-1}$, i.e. $p(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}) = \int\int ... \int p(\mathbf{x}_{0:t}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t})\, d\mathbf{x}_0 d\mathbf{x}_1 ... d\mathbf{x}_{t-1}$.

In the particle filter, the target distribution is $p\left(\mathbf{x}_{0:t}|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right)$ while the proposal distribution is $q\left(\mathbf{x}_{0:t}|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right)$. If we have samples $\mathbf{x}_{0:t}^{[i]} \sim q\left(\mathbf{x}_{0:t}|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right), i = 1, 2, ..., N_{\mathrm{s}}$, then we can approximate the target distribution $p\left(\mathbf{x}_{0:t}|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right)$ (see Equation 2.36)

$$p\left(\mathbf{x}_{0:t}|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right) \approx \frac{1}{W}\sum_{i=1}^{N_{\mathrm{s}}} w_t^{[i]}\delta\left(\mathbf{x}_{0:t} - \mathbf{x}_{0:t}^{[i]}\right) \ , \tag{2.40}$$

where (according to Equation 2.37)

$$w_t^{[i]} = \frac{p\left(\mathbf{x}_{0:t}|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right)}{q\left(\mathbf{x}_{0:t}|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right)} \ . \tag{2.41}$$

Suppose we already have samples drawn from the pdf $p\left(\mathbf{x}_{0:t-1}|\mathbf{c}_{1:t-1},\mathbf{s}_{1:t-1}\right)$ and we want to approximate the pdf $p\left(\mathbf{x}_{0:t}|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right)$ with a set of samples. Suppose further that the proposal distribution can be factorized as

$$q\left(\mathbf{x}_{0:t}|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right) = q\left(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right) q\left(\mathbf{x}_{0:t-1}|\mathbf{c}_{1:t-1},\mathbf{s}_{1:t-1}\right) \ . \tag{2.42}$$

Note that if $\mathbf{x}_{0:t-1}^{[i]} \sim q\left(\mathbf{x}_{0:t-1}|\mathbf{c}_{1:t-1},\mathbf{s}_{1:t-1}\right)$, then $\mathbf{x}_{0:t}^{[i]} \sim q\left(\mathbf{x}_{0:t}|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right)$ can be obtained by appending $\mathbf{x}_t^{[i]} \sim q\left(\mathbf{x}_t \middle| \mathbf{x}_{0:t-1}^{[i]},\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right)$ to $\mathbf{x}_{0:t-1}^{[i]}$. By using the Bayes' rule, the chain rule of probability [10], and the Markov assumption, we can breakdown $p\left(\mathbf{x}_{0:t}|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right)$ in terms of

[10]The chain rule of probability is given by $p\left(x,y|z\right) = p\left(x|z\right)p\left(y|x,z\right)$.

$p(\mathbf{s}_t|\mathbf{x}_t)$, $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)$, and $p(\mathbf{x}_{0:t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1})$ as shown in the following.

$$p(\mathbf{x}_{0:t}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}) = \frac{p(\mathbf{s}_t|\mathbf{x}_{0:t}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t-1})\, p(\mathbf{x}_{0:t}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1})}{p(\mathbf{s}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1})} \tag{2.43}$$

$$= \frac{p(\mathbf{s}_t|\mathbf{x}_{0:t}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t-1})\, p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t-1})\, p(\mathbf{x}_{0:t-1}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1})}{p(\mathbf{s}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1})} \tag{2.44}$$

$$= \frac{p(\mathbf{s}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)\, p(\mathbf{x}_{0:t-1}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1})}{p(\mathbf{s}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t-1})} \tag{2.45}$$

$$\propto p(\mathbf{s}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)\, p(\mathbf{x}_{0:t-1}|\mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1}) \ . \tag{2.46}$$

Expanding the numerator and denominator of Equation 2.37, we get

$$w_t^{[i]} \propto \frac{p\left(\mathbf{s}_t \left| \mathbf{x}_t^{[i]} \right.\right) p\left(\mathbf{x}_t^{[i]} \left| \mathbf{x}_{t-1}^{[i]}, \mathbf{c}_t \right.\right) p\left(\mathbf{x}_{0:t-1}^{[i]} \left| \mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1} \right.\right)}{q\left(\mathbf{x}_t^{[i]} \left| \mathbf{x}_{0:t-1}^{[i]}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t} \right.\right) q\left(\mathbf{x}_{0:t-1}^{[i]} \left| \mathbf{c}_{1:t-1}, \mathbf{s}_{1:t-1} \right.\right)} \tag{2.47}$$

$$\propto w_{t-1}^{[i]} \frac{p\left(\mathbf{s}_t \left| \mathbf{x}_t^{[i]} \right.\right) p\left(\mathbf{x}_t^{[i]} \left| \mathbf{x}_{t-1}^{[i]}, \mathbf{c}_t \right.\right)}{q\left(\mathbf{x}_t^{[i]} \left| \mathbf{x}_{0:t-1}^{[i]}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t} \right.\right)} \ . \tag{2.48}$$

For convenience, we can let $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t}) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t})$ and select the proposal distribution to be the system model, i.e. $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)$. As such, Equation 2.48 becomes Equation 2.38 (see also Algorithm 5). Note that if resampling is performed at every iteration such that the weights of the particles are reset to $\frac{1}{N_\mathrm{s}}$, then the weight update in Equation 2.38 simply becomes (see Algorithm 4)

$$w_t^{[i]} \propto p\left(\mathbf{s}_t \left| \mathbf{x}_t^{[i]} \right.\right) \ . \tag{2.49}$$

2.5 Other Filters

In the preceding three sections, we have discussed three concrete implementations of the Bayes' filter, namely the Kalman filter, EKF, and the particle filter. It should be pointed out that other filters exist such as the unscented Kalman filter (UKF) [Julier and Uhlmann, 1997], information filter (IF) and extended information filter (EIF) [Thrun et al., 2005], and histogram filter [Thrun et al., 2005] or approximate grid-based method [Arulampalam et al., 2002], among others, as well as variants of the particle filter such as the auxiliary particle filter (APF) [Pitt and Shephard, 1999] and Rao-Blackwellized particle filter (RBPF) [Murphy, 2000]. While these filters are interesting, important, and have found themselves being applied in a number of problems, we do not discuss them here as they are not employed in this dissertation. We refer the interested reader to the references given in this section.

2.6 Discussion

The problems of mobile robot localization and mapping are often viewed as dynamic state estimation problems. Probabilistic state estimation algorithms compute probability distributions over the possible values of the state variables using the available sensor data. The Bayes' filter is a recursive (on-line) state estimation algorithm that provides a framework for recursively computing probability distributions over the state variables as they evolve using sensor measurements that are received over time. The Bayes' filter computes the exact pdf at every time step given all available data up to that time.

The Kalman filter is a concrete and exact implementation of the Bayes' filter for linear Gaussian systems. However, many real-life problems are not linear; therefore, the Kalman filter is not directly applicable. For tracking nonlinear systems, the extended version of the Kalman filter or EKF first linearizes the nonlinear functions at the current estimate via a first-order Taylor series expansion of the functions and then applies the Kalman filter equations. Thus, the EKF requires the computation of the Jacobian matrices which may not be feasible or computationally costly in some cases. Unlike the Kalman filter, the EKF is an approximate implementation of the Bayes' filter for nonlinear systems and its performance depends on the degree of non-linearity of the functions involved. Since both the Kalman filter and EKF assume that the pdf is Gaussian, it suffices to only keep track of the mean vector and covariance matrix of the distribution at every iteration. The Kalman filter and EKF are computationally efficient and involve computing matrix inversion and multiplication. The EKF has become the *de facto* technique for dynamic state estimation and it performs reasonably well in practice.

The particle filter is an alternative approximate implementation of the Bayes' filter. Unlike the Kalman filter and EKF, the particle filter does not assume that the pdf is Gaussian. The particle filter represents the required pdf by a set of weighted samples or particles. As the number of particles used tends to infinity, the particle filter estimate approaches that of the Bayes' filter. The particle filter can estimate arbitrary distributions and, more importantly, multi-modal distributions (in cases where there are several distinct modes of hypotheses), which are not possible with the Gaussian distributions used in the Kalman filter and EKF.

Moreover, the particle filter is well-suited for nonlinear system dynamics and measurement functions. In order to implement the particle filter, one needs to be able to sample from the system model and evaluate the measurement model. The particle filter is gaining in popularity and it is being used in increasing number of applications including mobile robotics. It is the filter employed in the next three chapters of this dissertation.

Chapter 3

Mobile Robot Calibration

3.1 Introduction

Mobile robot localization, the problem of estimating the robot's position and orientation (often called the *pose*) within its environment, is often viewed as a dynamic state estimation problem using sensor data. In Chapter 2, we introduced probabilistic state estimation algorithms that compute probability density functions (pdfs) or belief distributions over the possible values of the variables of interest using the available sensor data. Before being able to employ the filters of the previous chapter to solve the mobile robot localization problem, two models must be known: the system model $p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t\right)$ and the observation or measurement model $p\left(\mathbf{s}_t|\mathbf{x}_t\right)$. In the context of mobile robotics, the system model $p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t\right)$ is often called the *kinematic* or *motion model* as it describes the effect the control action $\mathbf{c}_t$ (e.g. translation or rotation) has on the state or configuration of the robot (i.e. its pose) which

is represented as $\mathbf{x}_t$ while the observation model $p(\mathbf{s}_t|\mathbf{x}_t)$ is often called the *perceptual* or *sensor model* as it describes the process by which sensor measurements $\mathbf{s}_t$ (e.g. sonar readings, laser scans, or camera images) are generated given the state $\mathbf{x}_t$ of the robot and the environment. Since robot actuation is not perfect and uncertain and sensor measurements are almost always subject to random noise and error, the motion and sensor models are appropriately represented as probability distributions [Elfes, 1987, Fox et al., 1999, Thrun, 2003, Eliazar and Parr, 2004b].

The probabilistic motion and sensor models explicitly model the inherent uncertainty and noise that exist in robot actuation and perception, respectively. Of course, the specifics of the models depend on the kind of robot and sensors used as well as the robot's operating environment. The models are typically described by a set of parameters that affect the actual shapes and forms of these distributions. These distributions, in turn, influence the quality of the estimate provided by the probabilistic state estimation techniques. Thus, identifying the appropriate values of the parameters used to describe the motion and sensor models is an important step in mobile robotics. As already mentioned in Chapter 1, mobile robot calibration deals with the problem of identifying the parameters that describe the kinematic and perceptual processes of a mobile robot. There are several ways one can go about learning the parameters of the models. Typically, the parameters of the models are provided and hand tuned by a human operator and those are often derived from intensive and careful calibration experiments as well as the operator's knowledge and experience with the robot and its operating environment. The preceding approach is often a tedious and laborious manual process

that needs to be repeated whenever there is a significant change in the robot (e.g. wear and tear) or the environment (e.g. going from static environment to dynamic environment).

In this chapter, we propose an automated technique for learning the parameters of both the motion and sensor models of a mobile robot. The automated technique uses the expectation maximization (EM) [Dempster et al., 1977] approach, a machine learning technique for estimating the parameters of probabilistic models that depend on hidden or latent variables. Our technique assumes that the robot has access to the map of the environment and the historical account of its motion (as provided by the odometric information) and sensing (in our case, range measurements provided by sonar sensors). Odometry and sensor measurements are needed for estimating the likely trajectory of the robot through the environment during its normal operation. We estimate the robot's trajectory using particle filtering [Arulampalam et al., 2002] and smoothing [Doucet et al., 2000b, Godsill et al., 2004]. We provide experimental results demonstrating the effectiveness of the estimation approach and the advantage of learning the parameters of both the motion and sensor models.

3.2 Related Work

Calibration has been an active research area in mobile robotics. UMBmark [Borenstein and Feng, 1995] is a method for the quantitative measurement of systematic odometry errors in a mobile robot that involves performing a series of simple calibration experiments in which the robot traverses a square path in both clockwise and counter-clockwise directions and an

operator manually measures the absolute position of the vehicle to compare with the robot's calculated position based on odometry. Roy and Thrun [1999] proposed a statistical method for the on-line self-calibration of the odometry of mobile robots which eliminates the need for explicit measurements of actual robot motion by a human or some external device. Unlike UMBmark, the method of Roy and Thrun [1999] is automatic and it calibrates a robot's odometers continuously during its everyday operation allowing the robot to adapt to changes that might occur over its lifetime. Eliazar and Parr [2004b] have a similar goal where they proposed a method that can start with a crude motion model and bootstrap itself towards a more refined motion model; thus, allowing the robot to adapt to changing motion parameters. But unlike Roy and Thrun [1999], Eliazar and Parr [2004b] used a more general motion model which incorporates interdependence between motion terms including the influence of turns on lateral movement and vice versa. Instead of dealing with systematic errors, they also estimated non-systematic errors through the variance in the different motion terms.

Other notable work on the automatic calibration of the robot odometry include the self-calibrating extended Kalman filter (EKF_{SC}) approach by Caltabiano et al. [2004] and the observable filter (OF) (used in conjunction with the augmented Kalman filter (AKF)) by Martinelli et al. [2003]. Foxlin [2002] introduced a general architectural framework that enables systems to simultaneously track themselves, construct a map of landmarks in the environment, and calibrate sensor intrinsic and extrinsic parameters. Recently, Stronger and Stone [2005] presented a technique for the simultaneous calibration of the action and sensor models (SCASM) of a mobile robot. However, the models used by Stronger and Stone [2005]

are not probabilistic and their method makes use of careful calibration setup and requires the robot to go through a training phase thereby disrupting the robot from its normal operation.

Our work is similar in spirit with that of Eliazar and Parr [2004b] and Roy and Thrun [1999] but instead of only estimating the motion model parameters, we also aim for the estimation of the sensor model parameters. Unlike Stronger and Stone [2005], our models are expressed probabilistically and our estimation method can be performed by the robot during its normal operation thus skipping the need for a separate training phase. Our results demonstrate the advantage of co-calibrating both models.

3.3 Probabilistic Motion and Sensor Models

In this section, we discuss the probabilistic motion and sensor models we employ for our mobile robot. Note that there are several ways one can describe the motion and sensor models of a mobile robot probabilistically. The specific probabilistic motion and sensor models presented here are chosen because they are found to work well with our current robot and should not be seen as the only models that go with our automated calibration technique. In fact, in this chapter, we propose a general framework for estimating the parameters of a mobile robot that can be used for other motion and sensor models as well.

3.3.1 Motion Model

The purpose of the motion model is to describe the effect the control input $\mathbf{c}_t$ has on the robot's pose $\mathbf{x}_t$. In this chapter, we assume that the robot moves in a planar environment so that its pose at any given time t can be represented as a three-dimensional column vector $\mathbf{x}_t = (x_t, y_t, \theta_t)^T$, where $(x_t, y_t)^T$ is the robot's two-dimensional Cartesian coordinates and θ_t denotes the robot's heading or orientation. Since robot motion is not perfectly repeatable (i.e. the same control command will not generally produce the same effect on the robot's configuration), the motion model is expressed as a probability distribution of the form $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)$, where $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ are the robot's poses for two consecutive time steps $t-1$ and t, respectively, and $\mathbf{c}_t$ is the control command executed by the robot during the time interval $[t-1, t)$. In this work, our robot is equipped with odometry that provides an estimate of the robot pose $\hat{\mathbf{x}}_t = \left(\hat{x}_t, \hat{y}_t, \hat{\theta}_t\right)^T$ at time t by integrating wheel encoder information. Due to drift, wheel slippage, and other factors, the estimated pose $\hat{\mathbf{x}}_t$ gradually differs from the true pose $\mathbf{x}_t$ over time. Thus, using odometry alone is not sufficient for tracking accurately the robot pose over time. Nevertheless, we use the odometry measurements as our basis for calculating the estimated relative motion of the robot over time. Specifically, in the time interval $[t-1, t)$ and given the odometry measurements $\hat{\mathbf{x}}_{t-1}$ and $\hat{\mathbf{x}}_t$, we compute the estimated distance traveled

$\hat{d}_t$ and rotation made $\hat{r}_t$ by the robot during the given time interval as:

$$\hat{d}_t = \sqrt{(\hat{x}_t - \hat{x}_{t-1})^2 + (\hat{y}_t - \hat{y}_{t-1})^2} \tag{3.1}$$

$$\hat{r}_t = \left(\hat{\theta}_t - \hat{\theta}_{t-1}\right) \bmod 2\pi \ . \tag{3.2}$$

The estimated distance traveled and rotation made differ from the true ones and they are assumed to be corrupted by independent noise. While technically speaking the odometry information are sensor measurements, we consider them as control measurements and we let the control command $\mathbf{c}_t$ be the pair $\left(\hat{d}_t, \hat{r}_t\right)^T$. As suggested by the probability distribution $p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t\right)$, the robot's next pose depends stochastically on its previous pose one time step earlier and the control input $\mathbf{c}_t$. The distribution $p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t\right)$ governs how the particles in the particle filter are propagated from one time step to the next. Although not explicitly included as part of the conditioning variables in $p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t\right)$, the map $\mathcal{M}$ of the environment as well as the set of parameters that define the motion model also determine the robot's next pose.

There are several ways one can describe the motion model. Roy and Thrun [1999] suggested the following.

$$x_t = x_{t-1} + d_t \cos\left(\theta_{t-1} + r_t\right) \tag{3.3}$$

$$y_t = y_{t-1} + d_t \sin\left(\theta_{t-1} + r_t\right) \tag{3.4}$$

$$\theta_t = \left(\theta_{t-1} + r_t\right) \bmod 2\pi \ , \tag{3.5}$$

with the assumption that the drive and turn commands are independent. d_t and r_t denote the true distance traveled and rotation made by the robot, respectively. A more complex motion model proposed by Eliazar and Parr [2004b] that can account for simultaneous turning and lateral movement decomposes the movement into two principal components

$$x_t = x_{t-1} + D_t \cos\left(\theta_{t-1} + \frac{T_t}{2}\right) + E_t \cos\left(\theta_{t-1} + \frac{T_t + \pi}{2}\right) \tag{3.6}$$

$$y_t = y_{t-1} + D_t \sin\left(\theta_{t-1} + \frac{T_t}{2}\right) + E_t \sin\left(\theta_{t-1} + \frac{T_t + \pi}{2}\right) \tag{3.7}$$

$$\theta_t = (\theta_{t-1} + T_t) \mod 2\pi \ , \tag{3.8}$$

where $\theta_{t-1} + \frac{T_t}{2}$ is referred to as the major axis of movement, $\theta_{t-1} + \frac{T_t + \pi}{2}$ is the minor axis of movement (orthogonal to the major axis), and E_t is an extra lateral translation term to account for the shift in the orthogonal direction to the major axis. In their motion model, the variables D_t, T_t, and E_t are all independent and conditionally Gaussian given $\hat{d}_t$ and $\hat{r}_t$:

$$D_t \sim \mathcal{N}\left(\hat{d}_t \mu_{D_{\hat{d}}} + \hat{r}_t \mu_{D_{\hat{r}}}, \hat{d}_t^2 \sigma^2_{D_{\hat{d}}} + \hat{r}_t^2 \sigma^2_{D_{\hat{r}}}\right) \tag{3.9}$$

$$T_t \sim \mathcal{N}\left(\hat{d}_t \mu_{T_{\hat{d}}} + \hat{r}_t \mu_{T_{\hat{r}}}, \hat{d}_t^2 \sigma^2_{T_{\hat{d}}} + \hat{r}_t^2 \sigma^2_{T_{\hat{r}}}\right) \tag{3.10}$$

$$E_t \sim \mathcal{N}\left(\hat{d}_t \mu_{E_{\hat{d}}} + \hat{r}_t \mu_{E_{\hat{r}}}, \hat{d}_t^2 \sigma^2_{E_{\hat{d}}} + \hat{r}_t^2 \sigma^2_{E_{\hat{r}}}\right) \ , \tag{3.11}$$

where $\mathcal{N}(a, b)$ is a Gaussian distribution with mean a and variance b, μ_{A_b} is the coefficient for the contribution of the odometry term b to the mean of the distribution over A. Thus, the

twelve parameters $\mu_{D_{\hat{d}}}, \mu_{T_{\hat{d}}}, \mu_{E_{\hat{d}}}, \mu_{D_{\hat{r}}}, \mu_{T_{\hat{r}}}, \mu_{E_{\hat{r}}}, \sigma^2_{D_{\hat{d}}}, \sigma^2_{T_{\hat{d}}}, \sigma^2_{E_{\hat{d}}}, \sigma^2_{D_{\hat{r}}}, \sigma^2_{T_{\hat{r}}}$, and $\sigma^2_{E_{\hat{r}}}$, define this motion model.

In this chapter, we adopt the motion model of Eliazar and Parr [2004b] but with a slightly different noise model. We still use Equations 3.6, 3.7, and 3.8 for our state update equations except that

$$D_t \sim \mathcal{N}\left(\hat{d}_t, \hat{d}_t^2\sigma^2_{D_{\hat{d}}} + \hat{r}_t^2\sigma^2_{D_{\hat{r}}} + \sigma^2_{D_1}\right) \tag{3.12}$$

$$T_t \sim \mathcal{N}\left(\hat{r}_t, \hat{d}_t^2\sigma^2_{T_{\hat{d}}} + \hat{r}_t^2\sigma^2_{T_{\hat{r}}} + \sigma^2_{T_1}\right) \tag{3.13}$$

$$E_t \sim \mathcal{N}\left(0, \hat{d}_t^2\sigma^2_{E_{\hat{d}}} + \hat{r}_t^2\sigma^2_{E_{\hat{r}}} + \sigma^2_{E_1}\right) \ . \tag{3.14}$$

Our noise model is similar to the noise model of Eliazar and Parr [2004b] with $\mu_{D_{\hat{d}}} = 1$, $\mu_{D_{\hat{r}}} = 0$, $\mu_{T_{\hat{d}}} = 0$, $\mu_{T_{\hat{r}}} = 1$, $\mu_{E_{\hat{d}}} = 0$, $\mu_{E_{\hat{r}}} = 0$. We added extra constant terms to the variances using the additional parameters $\sigma^2_{D_1}$, $\sigma^2_{T_1}$, and $\sigma^2_{E_1}$. These parameters are added to account for errors that are not proportional to the translation or rotation of the robot. The motion parameters we wish to estimate from data are $\sigma^2_{D_{\hat{d}}}, \sigma^2_{T_{\hat{d}}}, \sigma^2_{E_{\hat{d}}}, \sigma^2_{D_{\hat{r}}}, \sigma^2_{T_{\hat{r}}}, \sigma^2_{E_{\hat{r}}}, \sigma^2_{D_1}, \sigma^2_{T_1}$, and $\sigma^2_{E_1}$.

3.3.2 Sensor Model

The sensor model constitutes the second probabilistic model needed to implement the probabilistic state estimation techniques discussed in Chapter 2. The sensor model describes the process by which sensor measurements are generated in the physical world. It relates the ac-

tual sensor measurements to the state of the robot or the environment. It is defined as the conditional probability distribution $p(\mathbf{s}_t|\mathbf{x}_t)$, where $\mathbf{s}_t$ is the sensor measurement received at time t and $\mathbf{x}_t$ is the robot pose at time t. Just like in the motion model $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)$, even though we do not explicitly include the environment map $\mathcal{M}$ as part of the conditioning variables in $p(\mathbf{s}_t|\mathbf{x}_t)$, it also determines the sensor measurement. The actual definition of the distribution $p(\mathbf{s}_t|\mathbf{x}_t)$ depends on the type of sensor used by the robot (e.g. cameras, range sensors). In this study, our mobile robot is equipped with a cyclic array of sixteen ultrasound sensors with eight front sonars and eight back sonars. Thus, in our case, $\mathbf{s}_t = \left\{ s_t^{(1)}, s_t^{(2)}, ..., s_t^{(K)} \right\}$, where $s_t^{(k)}$ is the kth sensor measurement received at time t and $K = 16$ is the total number of sensor measurements. We assume that the errors in the sensor measurements are independent.

The distance reported by the rangefinder is often subject to random noise and error. In this chapter, we make use of the sensor model for rangefinders described by Thrun et al. [2005] that represents the distribution as a mixture of four distributions corresponding to the four types of measurement errors typically observed in range readings: small measurement noise, errors due to unexpected objects or obstacles, errors due to failure to detect objects, and random unexplained noise. We let $s_t^{(k)*}$ denote the true distance to an obstacle, $s_t^{(k)}$ denote the recorded measurement, and s_{max} denote the maximum possible reading (e.g. 5000mm).

In order to model small measurement noise associated with range readings, we define a narrow Gaussian distribution p_{hit} over the range $[0, s_{\text{max}}]$ with mean $s_t^{(k)*}$ and standard devia-

tion σ_{hit}. σ_{hit} is an intrinsic parameter of the distribution p_{hit}. Formally

$$p_{\text{hit}}\left(s_t^{(k)}\middle|\,\mathbf{x}_t\right) = \begin{cases} \eta\frac{1}{\sqrt{2\pi}\sigma_{\text{hit}}}e^{-\frac{\left(s_t^{(k)}-s_t^{(k)*}\right)^2}{2\sigma_{\text{hit}}^2}} & \text{if } 0 \le s_t^{(k)} \le s_{\text{max}} \\ 0 & \text{otherwise} \end{cases}, \tag{3.15}$$

where η is a normalizing factor due to clipping.

Although we assume that the map of the environment is static and it does not include moving objects such as people, actual robot environments (such as buildings and hallways) are highly dynamic and they are often populated with dynamic entities other than the robot itself. These dynamic entities often block the robot's range sensors' "line-of-sight" thus causing them to return measurements shorter than the true range. This particular type of measurement error is modeled by a truncated exponential distribution p_{short} with parameter λ_{short}. Specifically,

$$p_{\text{short}}\left(s_t^{(k)}\middle|\,\mathbf{x}_t\right) = \begin{cases} \eta\lambda_{\text{short}}e^{-\lambda_{\text{short}}s_t^{(k)}} & \text{if } 0 \le s_t^{(k)} \le s_t^{(k)*} \\ 0 & \text{otherwise} \end{cases}, \tag{3.16}$$

where $\eta = \dfrac{1}{\left(1-e^{-\lambda_{\text{short}}s_t^{(k)*}}\right)}$.

Sometimes, rangefinders fail to detect obstacles. For sonar sensors, this type of error can happen due to specular reflections when the echo fails to return to the sonar. Thus, the obstacle appears invisible from the robot's perspective. Sonar sensors are typically programmed to return the maximum sensor range s_{max} when this happens. This particular type of mea-

surement error is modeled by a pseudo point-mass distribution p_{max} centered at s_{max}:

$$p_{\text{max}}\left(s_t^{(k)}\middle|\mathbf{x}_t\right) = \begin{cases} 1 & \text{if } s_t^{(k)} = s_{\text{max}} \\ 0 & \text{otherwise} \end{cases} . \tag{3.17}$$

Finally, rangefinders can return totally unexplainable measurements. This can be caused by interference or cross-talk between different sensors or incomplete knowledge about ranging technologies. This type of measurement error is modeled by a uniform distribution p_{rand} over the entire measurement range:

$$p_{\text{rand}}\left(s_t^{(k)}\middle|\mathbf{x}_t\right) = \begin{cases} \dfrac{1}{s_{\text{max}}} & \text{if } 0 \leq s_t^{(k)} \leq s_{\text{max}} \\ 0 & \text{otherwise} \end{cases} . \tag{3.18}$$

The four distributions defined in Equations 3.15 to 3.18 are combined by a weighted average through the mixing parameters $\alpha_{\text{hit}}, \alpha_{\text{short}}, \alpha_{\text{max}}$, and α_{rand}

$$p\left(s_t^{(k)}\middle|\mathbf{x}_t\right) = \alpha_{\text{hit}} p_{\text{hit}}\left(s_t^{(k)}\middle|\mathbf{x}_t\right) + \alpha_{\text{short}} p_{\text{short}}\left(s_t^{(k)}\middle|\mathbf{x}_t\right) + \alpha_{\text{max}} p_{\text{max}}\left(s_t^{(k)}\middle|\mathbf{x}_t\right) + \alpha_{\text{rand}} p_{\text{rand}}\left(s_t^{(k)}\middle|\mathbf{x}_t\right) , \tag{3.19}$$

such that

$$\alpha_{\text{hit}} + \alpha_{\text{short}} + \alpha_{\text{max}} + \alpha_{\text{rand}} = 1 . \tag{3.20}$$

It is often appropriate to think of the mixing parameters as the *a priori* probabilities that a particular sensor reading is caused by one of the four types of measurement errors discussed

above. Note that the mixing parameters, σ_{hit}, and λ_{short} are the intrinsic parameters of this particular sensor model which we wish to learn from actual data.

3.4 Particle Filtering and Smoothing

In order to learn the parameters of the motion and sensor models of the mobile robot from sensor observations, we need to infer the actual trajectory that the robot took through the environment. Inferring the actual robot trajectory would provide us with an estimate of the actual robot transitions that occurred as well as what robot poses and environmental entities (e.g. walls, doors, or posts) generated the sensor readings. We then use these estimates for learning the parameters of the models. It is noteworthy to point out that learning the model parameters requires the full smoothing inference, rather than just filtering, because it provides better estimates of the states of the process. Learning with filtering can fail to converge correctly [Russell and Norvig, 2003]. In what follows, we briefly review the process of particle filtering and then discuss particle smoothing for estimating the trajectory of the robot through the environment. Particle smoothing requires that particle filtering has already been carried out on the entire sequence of sensor measurements leading to a particle-based approximation of the filtering density at each time step.

3.4.1 Particle Filtering

Given the prior distribution $p(\mathbf{x}_0)$ over the robot's pose at time 0 represented as the weighted set of particles $\mathcal{X}_0$, the motion model $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)$, the sensor model $p(\mathbf{s}_t|\mathbf{x}_t)$, the sequence of control actions $\mathbf{c}_{1:T}$, and the sequence of sensor measurements $\mathbf{s}_{1:T}$ (where T denotes the last time step), we perform particle filtering using Algorithm 4 in Chapter 2 from time $t = 1$ to time $t = T$. After performing particle filtering, we now have a particle representation of the posterior distribution over the state of the robot for each time step t (i.e. $\mathcal{X}_t, t = 0, 1, ..., T$), given the observations up to time t. While particle filtering provides an estimate of the state of the robot for every time step t, we still need to perform particle smoothing as it provides better estimates of the states of the robot.

3.4.2 Particle Smoothing

Smoothing is the task of computing the posterior distribution over a past state, given all evidence up to the present. That is, the objective of smoothing is to compute the distributions $p(\mathbf{x}_t|\mathbf{c}_{1:T}, \mathbf{s}_{1:T})$ for some t with $0 \leq t \leq T$. Hindsight provides a better estimate of the state of the system than was available at the time because it incorporates more (later) evidence [Russell and Norvig, 2003]. Similar to filtering, smoothing can be performed recursively,

although in this case backward in time, using the smoothing formula

$$p\left(\mathbf{x}_t|\mathbf{c}_{1:T},\mathbf{s}_{1:T}\right) = \int p\left(\mathbf{x}_t|\mathbf{x}_{t+1},\mathbf{c}_{1:T},\mathbf{s}_{1:T}\right) p\left(\mathbf{x}_{t+1}|\mathbf{c}_{1:T},\mathbf{s}_{1:T}\right) d\mathbf{x}_{t+1} \tag{3.21}$$

$$= \int p\left(\mathbf{x}_t|\mathbf{x}_{t+1},\mathbf{c}_{1:t+1},\mathbf{s}_{1:t}\right) p\left(\mathbf{x}_{t+1}|\mathbf{c}_{1:T},\mathbf{s}_{1:T}\right) d\mathbf{x}_{t+1} \tag{3.22}$$

$$= \int \frac{p\left(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_{1:t+1},\mathbf{s}_{1:t}\right) p\left(\mathbf{x}_t|\mathbf{c}_{1:t+1},\mathbf{s}_{1:t}\right)}{p\left(\mathbf{x}_{t+1}|\mathbf{c}_{1:t+1},\mathbf{s}_{1:t}\right)} p\left(\mathbf{x}_{t+1}|\mathbf{c}_{1:T},\mathbf{s}_{1:T}\right) d\mathbf{x}_{t+1} \tag{3.23}$$

$$= \int \frac{p\left(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_{t+1}\right) p\left(\mathbf{x}_t|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right)}{p\left(\mathbf{x}_{t+1}|\mathbf{c}_{1:t+1},\mathbf{s}_{1:t}\right)} p\left(\mathbf{x}_{t+1}|\mathbf{c}_{1:T},\mathbf{s}_{1:T}\right) d\mathbf{x}_{t+1} \tag{3.24}$$

$$= \int \eta p\left(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{c}_{t+1}\right) p\left(\mathbf{x}_t|\mathbf{c}_{1:t},\mathbf{s}_{1:t}\right) p\left(\mathbf{x}_{t+1}|\mathbf{c}_{1:T},\mathbf{s}_{1:T}\right) d\mathbf{x}_{t+1} \ , \tag{3.25}$$

where $\eta = p\left(\mathbf{x}_{t+1}|\mathbf{c}_{1:t+1},\mathbf{s}_{1:t}\right)^{-1}$. Note that in Equation 3.25, the second factor in the integral is given by the system model and the third factor can be computed by filtering forward from time 1 to t. As pointed out by Godsill et al. [2004], in many applications, the marginal distributions $p\left(\mathbf{x}_t|\mathbf{c}_{1:T},\mathbf{s}_{1:T}\right), 0 \leq t \leq T$, are of limited interest, because investigation of historical states generally focuses on trajectories and hence requires consideration of collections of states together. Thus, we are interested in the entire joint smoothing density $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T},\mathbf{s}_{1:T}\right)$. Godsill et al. [2001] provided a sequential method for obtaining the maximum *a posteriori* (MAP) estimate of the sequence using dynamic programming and the Viterbi algorithm. However, in Bayesian inference setting, especially when dealing with multi-modal distributions, a single best estimate is rarely appropriate [Godsill et al., 2004].

Thus, we need for random generation of entire historical trajectories drawn from the joint smoothing density $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$.

In this chapter, we rely on a simple and efficient technique presented by Doucet et al. [2000b] and Godsill et al. [2004] for generating samples from the entire joint smoothing density $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$. The technique assumes that particle filtering has already been carried out on the entire data set resulting in a particle approximation of the posterior distribution at each time step t consisting of a weighted particle set $\mathcal{X}_t$. The technique is based on the following factorization of the joint smoothing density

$$p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right) = \prod_{t=0}^{T} p\left(\mathbf{x}_t|\mathbf{x}_{t+1:T}, \mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right) \ , \tag{3.26}$$

where

$$p\left(\mathbf{x}_t|\mathbf{x}_{t+1:T}, \mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right) = p\left(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{c}_{1:t+1}, \mathbf{s}_{1:t}\right) \tag{3.27}$$

$$= \frac{p\left(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{c}_{1:t+1}, \mathbf{s}_{1:t}\right) p\left(\mathbf{x}_t|\mathbf{c}_{1:t+1}, \mathbf{s}_{1:t}\right)}{p\left(\mathbf{x}_{t+1}|\mathbf{c}_{1:t+1}, \mathbf{s}_{1:t}\right)} \tag{3.28}$$

$$= \frac{p\left(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{c}_{t+1}\right) p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)}{p\left(\mathbf{x}_{t+1}|\mathbf{c}_{1:t+1}, \mathbf{s}_{1:t}\right)} \tag{3.29}$$

$$\propto p\left(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{c}_{t+1}\right) p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right) \ . \tag{3.30}$$

Equation 3.30 can be used to generate states recursively backward in time, given future states. Suppose we have a random sample $\widetilde{\mathbf{x}}_{t+1:T} \triangleq \left(\widetilde{\mathbf{x}}_{t+1}, \widetilde{\mathbf{x}}_{t+2}, ..., \widetilde{\mathbf{x}}_T\right)$ drawn from $p\left(\mathbf{x}_{t+1:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$, we then draw $\widetilde{\mathbf{x}}_t$ from $p\left(\mathbf{x}_t|\widetilde{\mathbf{x}}_{t+1:T}, \mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$. Prepending $\widetilde{\mathbf{x}}_t$ to $\widetilde{\mathbf{x}}_{t+1:T}$,

$\widetilde{\mathbf{x}}_{t:T} = (\widetilde{\mathbf{x}}_t, \widetilde{\mathbf{x}}_{t+1:T})$ is then a random sample from $p(\mathbf{x}_{t:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T})$. The question now is: how to draw $\widetilde{\mathbf{x}}_t$ from $p(\mathbf{x}_t|\widetilde{\mathbf{x}}_{t+1:T}, \mathbf{c}_{1:T}, \mathbf{s}_{1:T})$? Notice that $p(\mathbf{x}_t|\widetilde{\mathbf{x}}_{t+1:T}, \mathbf{c}_{1:T}, \mathbf{s}_{1:T})$ depends on the pdf $p(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t})$ for which we have a particle approximation from performing particle filtering. The particle approximation to $p(\mathbf{x}_t|\widetilde{\mathbf{x}}_{t+1:T}, \mathbf{c}_{1:T}, \mathbf{s}_{1:T})$ is obtained using the filtered particles $\mathbf{x}_t^{[i]}$ with modified importance weights $w_{t|t+1}^{[i]} \propto w_t^{[i]} p\left(\widetilde{\mathbf{x}}_{t+1} \middle| \mathbf{x}_t^{[i]}, \mathbf{c}_{t+1}\right)$, i.e.

$$p(\mathbf{x}_t|\widetilde{\mathbf{x}}_{t+1:T}, \mathbf{c}_{1:T}, \mathbf{s}_{1:T}) \approx \sum_{i=1}^{N_s} w_{t|t+1}^{[i]\prime} \delta\left(\mathbf{x}_t - \mathbf{x}_t^{[i]}\right) \ , \tag{3.31}$$

where

$$w_{t|t+1}^{[i]\prime} = \frac{w_{t|t+1}^{[i]}}{\sum_{j=1}^{N_s} w_{t|t+1}^{[j]}} \ . \tag{3.32}$$

Therefore, to draw $\widetilde{\mathbf{x}}_t$, we simply sample from the discrete distribution shown in Equation 3.31 with weights $w_{t|t+1}^{[i]}$.

Algorithm 6 shows the steps for sampling from the joint smoothing density $p(\mathbf{x}_{1:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T})$. The algorithm starts by drawing a particle at the last time step T with probability proportional to its forward filtering weight $w_T^{[i]}$. The algorithm then proceeds backward in time modifying the weights of the particles at each time step t by multiplying their forward filtering weight $w_t^{[i]}$ by the probability that they lead to a transition to the drawn particle at the next time step $t+1$ (i.e. $p\left(\widetilde{\mathbf{x}}_{t+1} \middle| \mathbf{x}_t^{[i]}, \mathbf{c}_{t+1}\right)$). The algorithm draws a particle with probability proportional to its modified weight $w_{t|t+1}^{[i]}$ at every time step t. The sequence of particles $\widetilde{\mathbf{x}}_t$ drawn from

time 0 to time T $(0 \leq t \leq T)$ constitutes a sampled trajectory $\widetilde{\mathbf{x}}_{0:T} \triangleq (\widetilde{\mathbf{x}}_0, \widetilde{\mathbf{x}}_1, ..., \widetilde{\mathbf{x}}_T)$ from the joint smoothing density $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$.

Algorithm 6 Sampling from the entire joint smoothing density $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$

Input:
$\mathcal{X}_t, t = 0, 1, ..., T$: particle approximations to the posterior pdfs $p\left(\mathbf{x}_t|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right), t = 0, 1, .., T$
$\mathbf{c}_{1:T} = (\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_T)$: set of controls from time 1 to time T
Output:
$\widetilde{\mathbf{x}}_{0:T} = (\widetilde{\mathbf{x}}_0, \widetilde{\mathbf{x}}_1, ..., \widetilde{\mathbf{x}}_T)$: a sample from the entire joint smoothing density $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$
Process:
draw i with probability $\propto w_T^{[i]}$
$\widetilde{\mathbf{x}}_T \leftarrow \mathbf{x}_T^{[i]}$
for $t \leftarrow T - 1$ down to 0 do
    for $i \leftarrow 1$ to N_{s} do
        $w_{t|t+1}^{[i]} \leftarrow w_t^{[i]} p\left(\widetilde{\mathbf{x}}_{t+1} \middle| \mathbf{x}_t^{[i]}, \mathbf{c}_{t+1}\right)$
    end
    draw i with probability $\propto w_{t|t+1}^{[i]}$
    $\widetilde{\mathbf{x}}_t \leftarrow \mathbf{x}_t^{[i]}$
end

Algorithm 6 can be repeated many times if several samples of $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$ are needed. We use Algorithm 6 to generate several robot trajectories and use them in estimating the parameters of the motion and sensor models of the robot.

3.5 Expectation Maximization and Parameter Estimation

To estimate the set of motion and sensor model parameters of the robot, we propose using the expectation maximization (EM) algorithm [Dempster et al., 1977], a standard machine learning method.

3.5.1 Expectation Maximization

EM is an iterative optimization method for estimating some unknown parameters ϑ in a probabilistic model, given measurement data $\mathcal{D}$. However, some variables $\mathbf{u}$ (sometimes called *hidden* or *latent* variables) are not observed and that must be integrated out of the likelihood function. The objective of EM is to seek for the parameters ϑ^* that maximizes the posterior probability of the parameters ϑ given the data $\mathcal{D}$, with $\mathbf{u}$ marginalized out, i.e.

$$\vartheta^* = \arg\max_{\vartheta} \int p\left(\vartheta, \mathbf{u}|\mathcal{D}\right) d\mathbf{u} \ . \tag{3.33}$$

Our discussion of the EM algorithm is based on those by Minka [1998] and Dellaert [2002] that explain the EM algorithm as lower bound maximization.

The idea behind the EM algorithm is to alternate between estimating the unknown variables ϑ and the unobserved variables $\mathbf{u}$. However, given an estimate for ϑ, the EM computes a distribution over $\mathbf{u}$ rather than just finding the best estimate for $\mathbf{u}$. While the idea of alternating between estimating the unknown variables ϑ and the unobserved variables $\mathbf{u}$ often serves as a good intuition for the EM algorithm, an alternative and better exposition of the EM algorithm is given by the lower bounding maximization viewpoint. In the lower bounding maximization viewpoint, the EM algorithm alternates between constructing a lower bound of the posterior distribution in Equation 3.33 (this step is often called the *expectation step* or *E-step*) and maximizing (optimizing) the bound (this step is often called the *maximization step* or *M-step*), thus improving the estimate for ϑ.

To maximize the posterior distribution in Equation 3.33, first notice that we can rewrite it as

$$\vartheta^* = \arg\max_{\vartheta} p\left(\vartheta|\mathcal{D}\right) \tag{3.34}$$

$$= \arg\max_{\vartheta} p\left(\vartheta,\mathcal{D}\right) \ , \tag{3.35}$$

since $p\left(\vartheta,\mathcal{D}\right) \propto p\left(\vartheta|\mathcal{D}\right)$. Maximizing $p\left(\vartheta,\mathcal{D}\right)$ is equivalent to maximizing $\log p\left(\vartheta,\mathcal{D}\right)$ since the logarithm is a monotonically non-decreasing function. Thus,

$$\vartheta^* = \arg\max_{\vartheta} \log p\left(\vartheta,\mathcal{D}\right) \tag{3.36}$$

$$= \arg\max_{\vartheta} \log \int p\left(\vartheta,\mathbf{u},\mathcal{D}\right) d\mathbf{u} \tag{3.37}$$

$$= \arg\max_{\vartheta} \log \int q_t\left(\mathbf{u}\right) \frac{p\left(\vartheta,\mathbf{u},\mathcal{D}\right)}{q_t\left(\mathbf{u}\right)} d\mathbf{u} \ . \tag{3.38}$$

where in Equation 3.38, we have introduced some arbitrary probability distribution $q_t\left(\mathbf{u}\right)$ over the hidden variables $\mathbf{u}$ at the tth iteration of the EM algorithm. To maximize the objective in Equation 3.38, EM constructs a tractable lower bound $b_t\left(\vartheta\right)$ at the current estimate of ϑ at the tth iteration (i.e. ϑ_t) and maximizes the bound instead. Using Jensen's inequality[1],

$$b_t\left(\vartheta\right) \triangleq \int q_t\left(\mathbf{u}\right) \log \frac{p\left(\vartheta,\mathbf{u},\mathcal{D}\right)}{q_t\left(\mathbf{u}\right)} d\mathbf{u} \leq \log \int q_t\left(\mathbf{u}\right) \frac{p\left(\vartheta,\mathbf{u},\mathcal{D}\right)}{q_t\left(\mathbf{u}\right)} d\mathbf{u} \ . \tag{3.39}$$

[1]Jensen's inequality (continuous version) states that if $p\left(x\right)$ is a continuous density function and f is a real continuous function that is concave, then $\int p\left(x\right) f\left(g\left(x\right)\right) dx \leq f\left(\int p\left(x\right) g\left(x\right) dx\right)$.

Given the bound $b_t(\vartheta)$, the EM algorithm finds the best (optimal) lower bound that touches the objective function $\log p(\vartheta, \mathcal{D})$ at the current estimate ϑ_t at the tth iteration. This is to ensure that the next estimate ϑ_t is an improvement on ϑ when the bound is locally maximized with respect to ϑ. The optimal lower bound at the current estimate ϑ_t can be obtained by maximizing

$$b_t(\vartheta_t) = \int q_t(\mathbf{u}) \log \frac{p(\vartheta_t, \mathbf{u}, \mathcal{D})}{q_t(\mathbf{u})} d\mathbf{u} \ , \tag{3.40}$$

with respect to the distribution $q_t(\mathbf{u})$. Using a Lagrange multiplier λ for the constraint $\int q_t(\mathbf{u})\, d\mathbf{u} = 1$, the objective is now

$$B(q_t(\mathbf{u})) = \int q_t(\mathbf{u}) \log p(\vartheta_t, \mathbf{u}, \mathcal{D})\, d\mathbf{u} - \int q_t(\mathbf{u}) \log q_t(\mathbf{u})\, d\mathbf{u} + \lambda \left(1 - \int q_t(\mathbf{u})\, d\mathbf{u}\right) \ . \tag{3.41}$$

Taking the partial derivative of Equation 3.41 with respect to $q_t(\mathbf{u})$ and setting to 0

$$\frac{\partial B}{\partial q_t(\mathbf{u})} = \log p(\vartheta_t, \mathbf{u}, \mathcal{D}) - \log q_t(\mathbf{u}) - 1 - \lambda = 0 \ . \tag{3.42}$$

Solving for $q_t(\mathbf{u})$ in Equation 3.42, we get

$$q_t(\mathbf{u}) = \frac{p(\vartheta_t, \mathbf{u}, \mathcal{D})}{\int p(\vartheta_t, \mathbf{u}, \mathcal{D})\, d\mathbf{u}} = p(\mathbf{u}|\vartheta_t, \mathcal{D}) \ . \tag{3.43}$$

Note that if we use the value of $q_t(\mathbf{u})$ in Equation 3.43 to Equation 3.40, the bound $b_t(\vartheta_t)$ indeed touches the objective function $\log p(\vartheta_t, \mathcal{D})$ at the current estimate ϑ_t:

$$b_t(\vartheta_t) = \int p(\mathbf{u}|\vartheta_t, \mathcal{D}) \log \frac{p(\vartheta_t, \mathbf{u}, \mathcal{D})}{p(\mathbf{u}|\vartheta_t, \mathcal{D})} d\mathbf{u} \tag{3.44}$$

$$= \int p(\mathbf{u}|\vartheta_t, \mathcal{D}) \log p(\vartheta_t, \mathcal{D}) \, d\mathbf{u} \tag{3.45}$$

$$= \log p(\vartheta_t, \mathcal{D}) \int p(\mathbf{u}|\vartheta_t, \mathcal{D}) \, d\mathbf{u} \tag{3.46}$$

$$= \log p(\vartheta_t, \mathcal{D}) \quad . \tag{3.47}$$

The preceding is the E-step of the EM algorithm where it finds the optimal lower bound that touches the objective function $\log p(\vartheta, \mathcal{D})$ at the current estimate ϑ_t and this is achieved by setting $q_t(\mathbf{u})$ to be $p(\mathbf{u}|\vartheta_t, \mathcal{D})$.

Having found the optimal lower bound, the EM algorithm then maximizes it with respect to ϑ. Note that Equation 3.39 with $q_t(\mathbf{u}) = p(\mathbf{u}|\vartheta_t, \mathcal{D})$ can be rewritten as

$$b_t(\vartheta) = \int p(\mathbf{u}|\vartheta_t, \mathcal{D}) \log \frac{p(\vartheta, \mathbf{u}, \mathcal{D})}{p(\mathbf{u}|\vartheta_t, \mathcal{D})} d\mathbf{u} \tag{3.48}$$

$$= \int p(\mathbf{u}|\vartheta_t, \mathcal{D}) \log p(\vartheta, \mathbf{u}, \mathcal{D})\, d\mathbf{u} - \int p(\mathbf{u}|\vartheta_t, \mathcal{D}) \log p(\mathbf{u}|\vartheta_t, \mathcal{D})\, d\mathbf{u} \tag{3.49}$$

$$= \mathbb{E}_{p(\mathbf{u}|\vartheta_t, \mathcal{D})}[\log p(\vartheta, \mathbf{u}, \mathcal{D})] - \mathbb{E}_{p(\mathbf{u}|\vartheta_t, \mathcal{D})}[\log p(\mathbf{u}|\vartheta_t, \mathcal{D})] \tag{3.50}$$

$$= \mathbb{E}_{p(\mathbf{u}|\vartheta_t, \mathcal{D})}[\log(p(\mathbf{u}, \mathcal{D}|\vartheta)\, p(\vartheta))] - \mathbb{E}_{p(\mathbf{u}|\vartheta_t, \mathcal{D})}[\log p(\mathbf{u}|\vartheta_t, \mathcal{D})] \tag{3.51}$$

$$= \mathbb{E}_{p(\mathbf{u}|\vartheta_t, \mathcal{D})}[\log p(\mathbf{u}, \mathcal{D}|\vartheta)] + \mathbb{E}_{p(\mathbf{u}|\vartheta_t, \mathcal{D})}[\log p(\vartheta)] - \mathbb{E}_{p(\mathbf{u}|\vartheta_t, \mathcal{D})}[\log p(\mathbf{u}|\vartheta_t, \mathcal{D})] \tag{3.52}$$

$$= \mathbb{E}_{p(\mathbf{u}|\vartheta_t, \mathcal{D})}[\log p(\mathbf{u}, \mathcal{D}|\vartheta)] + \log p(\vartheta) - \mathbb{E}_{p(\mathbf{u}|\vartheta_t, \mathcal{D})}[\log p(\mathbf{u}|\vartheta_t, \mathcal{D})]\ . \tag{3.53}$$

In Equation 3.53, the first term is the expected log-likelihood of the hidden variables $\mathbf{u}$ and data $\mathcal{D}$ given the parameters ϑ, the second term is the logarithm of the prior distribution over the unknown variables ϑ, and the last term is the entropy[2] of the distribution $p(\mathbf{u}|\vartheta_t, \mathcal{D})$. Since the last term does not depend on ϑ, we can maximize the bound with respect to ϑ using

[2]The (continuous) entropy of a probability density function $p(x)$ is given by $\mathbb{E}_{p(x)}[\log p(x)] = -\int p(x) \log p(x)\, dx$.

only the first two terms in Equation 3.53 to get the new improved estimate ϑ_{t+1}, i.e.

$$\vartheta_{t+1} = \arg\max_{\vartheta} b_t(\vartheta) \tag{3.54}$$

$$= \arg\max_{\vartheta} [\mathbb{E}_{p(\mathbf{u}|\vartheta_t, \mathcal{D})}[\log p(\mathbf{u}, \mathcal{D}|\vartheta)] + \log p(\vartheta)] \ . \tag{3.55}$$

To summarize, at initialization, the EM algorithm starts with an initial estimate of the parameters ϑ_0. At every iteration t, it finds an optimal lower bound $b_t(\vartheta)$ of the objective function at the current estimate ϑ_t (see Equation 3.40) (the E-step), and then maximizes this bound to obtain an improved estimate ϑ_{t+1} (see Equation 3.55) (the M-step). The EM algorithm repeatedly alternates between performing the E-step and M-step until convergence or a maximum number of iterations is reached. It is noteworthy to point out that the EM algorithm is a local optimization technique, and as such, it can also be trapped in some local optimum like other optimization methods such as gradient descent and Newton's methods.

3.5.2 Parameter Estimation Framework

In this section, we now discuss how we apply the EM algorithm discussed in the preceding subsection to learn the parameters of the motion and sensor models of the robot.

The parameters we want to learn from data are the set of motion and sensor parameters:

$$\vartheta = \left\{\sigma^2_{D_{\hat{d}}}, \sigma^2_{T_{\hat{d}}}, \sigma^2_{E_{\hat{d}}}, \sigma^2_{D_{\hat{r}}}, \sigma^2_{T_{\hat{r}}}, \sigma^2_{E_{\hat{r}}}, \sigma^2_{D_1}, \sigma^2_{T_1}, \sigma^2_{E_1}, \alpha_{\text{hit}}, \alpha_{\text{short}}, \alpha_{\text{max}}, \alpha_{\text{rand}}, \sigma_{\text{hit}}, \lambda_{\text{short}}\right\} \ . \tag{3.56}$$

The data $\mathcal{D}$ from which we learn the parameters are

$$\mathcal{D} = \{\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\} \ . \tag{3.57}$$

Finally, the hidden variables are the actual robot trajectory through the environment, i.e.

$$\mathbf{u} = \mathbf{x}_{0:T} \ . \tag{3.58}$$

Note that the true robot trajectory $\mathbf{x}_{0:T}$ is not directly observable even with accurate sensors such as global positioning system or GPS.

At initialization, we set the parameters ϑ to some initial values. We then alternate between performing the E-step and M-step. As discussed in the previous subsection, in the E-step, we compute the expectation of $\log p\left(\mathbf{u}, \mathcal{D}|\vartheta\right)$ with respect to the distribution $p\left(\mathbf{u}|\vartheta_t, \mathcal{D}\right)$. The distribution $p\left(\mathbf{u}|\vartheta_t, \mathcal{D}\right)$ is nothing but the entire joint smoothing density $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$ given a particular set of parameters ϑ_t discussed in Section 3.4.2. However, computing $\mathbb{E}_{p(\mathbf{x}_{0:T}|\mathbf{c}_{1:T},\mathbf{s}_{1:T})}[\log p\left(\mathbf{x}_{0:T}, \mathbf{c}_{1:T}, \mathbf{s}_{1:T}|\vartheta\right)]$ is, in general, intractable since the space of all possible robot trajectories $\mathbf{x}_{0:T}$ is infinite and we do not have a closed form for the distribution $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$ but only samples from it. Thus, we approximate the E-step by performing particle filtering and smoothing to obtain sample robot trajectories from the smoothing density $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$ as discussed in Section 3.4. In the maximization step or M-step, we treat the sampled trajectories obtained in the E-step as the ground truth to compute the maximum likelihood parameters of the models. We repeatedly alternate between perform-

ing the E-step and M-step until convergence. It has been shown by Dempster et al. [1977] that the EM algorithm is guaranteed to reach a local optimum. We should emphasize that our E-step is just an approximation of the "true" E-step of the EM algorithm since we are sampling from the posterior distribution $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$ over possible robot paths instead of computing with the exact posterior distribution $p\left(\mathbf{x}_{0:T}|\mathbf{c}_{1:T}, \mathbf{s}_{1:T}\right)$.

To compute the maximum likelihood values of the motion model parameters in the M-step, we calculate the motion errors $\epsilon_{D_t}^{[j]}$, $\epsilon_{T_t}^{[j]}$, and $\epsilon_{E_t}^{[j]}$ for $t = 0, 1, ..., T-1$ (based from the jth sampled robot trajectory $\widetilde{\mathbf{x}}_{0:T}^{[j]}$ obtained in the E-step) as well as the contributions of the odometry values $\hat{d}_t$ and $\hat{r}_t$ to the variances of these errors. Let $\epsilon_{D_t}^{[j]}$ be the translational error, $\epsilon_{T_t}^{[j]}$ the rotational error, and $\epsilon_{E_t}^{[j]}$ the lateral translation given by

$$\epsilon_{T_t}^{[j]} = \left(\widetilde{\theta}_{t+1}^{[j]} - \widetilde{\theta}_t^{[j]} - \hat{r}_t\right) \mod 2\pi \tag{3.59}$$

$$\epsilon_{D_t}^{[j]} = \left(\widetilde{x}_{t+1}^{[j]} - \widetilde{x}_t^{[j]}\right)\cos\left(\widetilde{\theta}_t^{[j]} + \frac{\hat{r}_t + \epsilon_{T_t}^{[j]}}{2}\right) + \left(\widetilde{y}_{t+1}^{[j]} - \widetilde{y}_t^{[j]}\right)\sin\left(\widetilde{\theta}_t^{[j]} + \frac{\hat{r}_t + \epsilon_{T_t}^{[j]}}{2}\right) - \hat{d}_t \tag{3.60}$$

$$\epsilon_{E_t}^{[j]} = -\left(\widetilde{x}_{t+1}^{[j]} - \widetilde{x}_t^{[j]}\right)\sin\left(\widetilde{\theta}_t^{[j]} + \frac{\hat{r}_t + \epsilon_{T_t}^{[j]}}{2}\right) + \left(\widetilde{y}_{t+1}^{[j]} - \widetilde{y}_t^{[j]}\right)\cos\left(\widetilde{\theta}_t^{[j]} + \frac{\hat{r}_t + \epsilon_{T_t}^{[j]}}{2}\right) . \tag{3.61}$$

Note that

$$\epsilon_{D_t}^{[j]} \sim \mathcal{N}\left(0, \hat{d}_t^2\sigma_{D_{\hat{d}}}^2 + \hat{r}_t^2\sigma_{D_{\hat{r}}}^2 + \sigma_{D_1}^2\right) \tag{3.62}$$

$$\epsilon_{T_t}^{[j]} \sim \mathcal{N}\left(0, \hat{d}_t^2\sigma_{T_{\hat{d}}}^2 + \hat{r}_t^2\sigma_{T_{\hat{r}}}^2 + \sigma_{T_1}^2\right) \tag{3.63}$$

$$\epsilon_{E_t}^{[j]} \sim \mathcal{N}\left(0, \hat{d}_t^2\sigma_{E_{\hat{d}}}^2 + \hat{r}_t^2\sigma_{E_{\hat{r}}}^2 + \sigma_{E_1}^2\right) \ . \tag{3.64}$$

Given the following likelihood functions

$$\mathcal{L}_{\epsilon_D}\left(\sigma_{D_{\hat{d}}}^2, \sigma_{D_{\hat{r}}}^2, \sigma_{D_1}^2\right) = p\left(\left\{\epsilon_{D_t}^{[j]}\right\} \middle| \mathbf{c}_{1:T}, \left\{\widetilde{\mathbf{x}}_{0:T}^{[j]}\right\}\right) \tag{3.65}$$

$$= \prod_j \prod_{t=0}^{T-1} \frac{1}{\sqrt{2\pi\left(\hat{d}_t^2\sigma_{D_{\hat{d}}}^2 + \hat{r}_t^2\sigma_{D_{\hat{r}}}^2 + \sigma_{D_1}^2\right)}} \times \exp\left(-\frac{\left(\epsilon_{D_t}^{[j]}\right)^2}{2\left(\hat{d}_t^2\sigma_{D_{\hat{d}}}^2 + \hat{r}_t^2\sigma_{D_{\hat{r}}}^2 + \sigma_{D_1}^2\right)}\right) \tag{3.66}$$

$$\mathcal{L}_{\epsilon_T}\left(\sigma_{T_{\hat{d}}}^2, \sigma_{T_{\hat{r}}}^2, \sigma_{T_1}^2\right) = p\left(\left\{\epsilon_{T_t}^{[j]}\right\} \middle| \mathbf{c}_{1:T}, \left\{\widetilde{\mathbf{x}}_{0:T}^{[j]}\right\}\right) \tag{3.67}$$

$$= \prod_j \prod_{t=0}^{T-1} \frac{1}{\sqrt{2\pi\left(\hat{d}_t^2\sigma_{T_{\hat{d}}}^2 + \hat{r}_t^2\sigma_{T_{\hat{r}}}^2 + \sigma_{T_1}^2\right)}} \times \exp\left(-\frac{\left(\epsilon_{T_t}^{[j]}\right)^2}{2\left(\hat{d}_t^2\sigma_{T_{\hat{d}}}^2 + \hat{r}_t^2\sigma_{T_{\hat{r}}}^2 + \sigma_{T_1}^2\right)}\right) \tag{3.68}$$

$$\mathcal{L}_{\epsilon_E}\left(\sigma^2_{E_{\hat{d}}}, \sigma^2_{E_{\hat{r}}}, \sigma^2_{E_1}\right) = p\left(\left\{\epsilon^{[j]}_{E_t}\right\}\middle|\, \mathbf{c}_{1:T}, \left\{\widetilde{\mathbf{x}}^{[j]}_{0:T}\right\}\right) \tag{3.69}$$

$$= \prod_j \prod_{t=0}^{T-1} \frac{1}{\sqrt{2\pi\left(\hat{d}_t^2\sigma^2_{E_{\hat{d}}} + \hat{r}_t^2\sigma^2_{E_{\hat{r}}} + \sigma^2_{E_1}\right)}} \times \exp\left(-\frac{\left(\epsilon^{[j]}_{E_t}\right)^2}{2\left(\hat{d}_t^2\sigma^2_{E_{\hat{d}}} + \hat{r}_t^2\sigma^2_{E_{\hat{r}}} + \sigma^2_{E_1}\right)}\right), \tag{3.70}$$

our objective is to get the maximum likelihood estimates of the motion parameters, i.e.

$$\sigma^{2*}_{D_{\hat{d}}}, \sigma^{2*}_{D_{\hat{r}}}, \sigma^{2*}_{D_1} = \underset{\sigma^2_{D_{\hat{d}}}, \sigma^2_{D_{\hat{r}}}, \sigma^2_{D_1}}{\arg\max}\ \mathcal{L}_{\epsilon_D}\left(\sigma^2_{D_{\hat{d}}}, \sigma^2_{D_{\hat{r}}}, \sigma^2_{D_1}\right) \tag{3.71}$$

$$\sigma^{2*}_{T_{\hat{d}}}, \sigma^{2*}_{T_{\hat{r}}}, \sigma^{2*}_{T_1} = \underset{\sigma^2_{T_{\hat{d}}}, \sigma^2_{T_{\hat{r}}}, \sigma^2_{T_1}}{\arg\max}\ \mathcal{L}_{\epsilon_T}\left(\sigma^2_{T_{\hat{d}}}, \sigma^2_{T_{\hat{r}}}, \sigma^2_{T_1}\right) \tag{3.72}$$

$$\sigma^{2*}_{E_{\hat{d}}}, \sigma^{2*}_{E_{\hat{r}}}, \sigma^{2*}_{E_1} = \underset{\sigma^2_{E_{\hat{d}}}, \sigma^2_{E_{\hat{r}}}, \sigma^2_{E_1}}{\arg\max}\ \mathcal{L}_{\epsilon_E}\left(\sigma^2_{E_{\hat{d}}}, \sigma^2_{E_{\hat{r}}}, \sigma^2_{E_1}\right)\ . \tag{3.73}$$

Taking the logarithm of the likelihood functions, we get

$$\mathcal{L}^*_{\epsilon_D}\left(\sigma^2_{D_{\hat{d}}}, \sigma^2_{D_{\hat{r}}}, \sigma^2_{D_1}\right) = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1}\Big[\log 2\pi + \log\left(\hat{d}_t^2\sigma^2_{D_{\hat{d}}} + \hat{r}_t^2\sigma^2_{D_{\hat{r}}} + \sigma^2_{D_1}\right) + \frac{\left(\epsilon^{[j]}_{D_t}\right)^2}{\hat{d}_t^2\sigma^2_{D_{\hat{d}}} + \hat{r}_t^2\sigma^2_{D_{\hat{r}}} + \sigma^2_{D_1}}\Big] \tag{3.74}$$

$$\mathcal{L}^*_{\epsilon_T}\left(\sigma^2_{T_{\hat{d}}}, \sigma^2_{T_{\hat{r}}}, \sigma^2_{T_1}\right) = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1}\Big[\log 2\pi + \log\left(\hat{d}_t^2\sigma^2_{T_{\hat{d}}} + \hat{r}_t^2\sigma^2_{T_{\hat{r}}} + \sigma^2_{T_1}\right) + \frac{\left(\epsilon^{[j]}_{T_t}\right)^2}{\hat{d}_t^2\sigma^2_{T_{\hat{d}}} + \hat{r}_t^2\sigma^2_{T_{\hat{r}}} + \sigma^2_{T_1}}\Big] \tag{3.75}$$

$$\mathcal{L}^*_{\epsilon_E}\left(\sigma^2_{E_{\hat{d}}}, \sigma^2_{E_{\hat{r}}}, \sigma^2_{E_1}\right) = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1} \Big[\log 2\pi + \log\left(\hat{d}_t^2\sigma^2_{E_{\hat{d}}} + \hat{r}_t^2\sigma^2_{E_{\hat{r}}} + \sigma^2_{E_1}\right) + \frac{\left(\epsilon^{[j]}_{E_t}\right)^2}{\hat{d}_t^2\sigma^2_{E_{\hat{d}}} + \hat{r}_t^2\sigma^2_{E_{\hat{r}}} + \sigma^2_{E_1}} \Big] . \tag{3.76}$$

We then maximize the logarithm of the likelihood functions via conjugate gradient[3] ascent with respect to the motion model parameters. Conjugate gradient is a method for finding the closest local minimum or maximum of a function of several variables and it assumes that the gradient of the function can be computed. Unlike the gradient descent or ascent method that takes steps proportional to the gradient of the function at the current estimate for finding the local minimum or maximum, conjugate gradient uses conjugate directions for locating the optimum more efficiently. Typically, in conjugate gradient, the gradient of the function is taken as the first search direction while the succeeding search directions are chosen such that they are mutually conjugate (or orthogonal) to all previous search directions, thus avoiding searching in directions that have been previously searched (this is the case in gradient descent or ascent method). Since conjugate gradient requires the gradient of the function to be optimized, we show how the gradient of the log-likelihood functions are computed. The

[3]For the interested reader, Shewchuk [1994] provides an excellent introduction to the method of conjugate gradient.

gradients of the log-likelihood functions with respect to the motion parameters are

$$\frac{\partial \mathcal{L}^*_{\epsilon_D}}{\partial \sigma^2_{D_{\hat{d}}}} = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1} \Big[\frac{\hat{d}_t^2}{\hat{d}_t^2\sigma^2_{D_{\hat{d}}} + \hat{r}_t^2\sigma^2_{D_{\hat{r}}} + \sigma^2_{D_1}} - \frac{\left(\epsilon^{[j]}_{D_t}\right)^2 \hat{d}_t^2}{\left(\hat{d}_t^2\sigma^2_{D_{\hat{d}}} + \hat{r}_t^2\sigma^2_{D_{\hat{r}}} + \sigma^2_{D_1}\right)^2} \Big] \tag{3.77}$$

$$\frac{\partial \mathcal{L}^*_{\epsilon_D}}{\partial \sigma^2_{D_{\hat{r}}}} = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1} \Big[\frac{\hat{r}_t^2}{\hat{d}_t^2\sigma^2_{D_{\hat{d}}} + \hat{r}_t^2\sigma^2_{D_{\hat{r}}} + \sigma^2_{D_1}} - \frac{\left(\epsilon^{[j]}_{D_t}\right)^2 \hat{r}_t^2}{\left(\hat{d}_t^2\sigma^2_{D_{\hat{d}}} + \hat{r}_t^2\sigma^2_{D_{\hat{r}}} + \sigma^2_{D_1}\right)^2} \Big] \tag{3.78}$$

$$\frac{\partial \mathcal{L}^*_{\epsilon_D}}{\partial \sigma^2_{D_{\hat{1}}}} = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1} \Big[\frac{1}{\hat{d}_t^2\sigma^2_{D_{\hat{d}}} + \hat{r}_t^2\sigma^2_{D_{\hat{r}}} + \sigma^2_{D_1}} - \frac{\left(\epsilon^{[j]}_{D_t}\right)^2}{\left(\hat{d}_t^2\sigma^2_{D_{\hat{d}}} + \hat{r}_t^2\sigma^2_{D_{\hat{r}}} + \sigma^2_{D_1}\right)^2} \Big] \tag{3.79}$$

$$\frac{\partial \mathcal{L}^*_{\epsilon_T}}{\partial \sigma^2_{T_{\hat{d}}}} = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1} \Big[\frac{\hat{d}_t^2}{\hat{d}_t^2\sigma^2_{T_{\hat{d}}} + \hat{r}_t^2\sigma^2_{T_{\hat{r}}} + \sigma^2_{T_1}} - \frac{\left(\epsilon^{[j]}_{T_t}\right)^2 \hat{d}_t^2}{\left(\hat{d}_t^2\sigma^2_{T_{\hat{d}}} + \hat{r}_t^2\sigma^2_{T_{\hat{r}}} + \sigma^2_{T_1}\right)^2} \Big] \tag{3.80}$$

$$\frac{\partial \mathcal{L}^*_{\epsilon_T}}{\partial \sigma^2_{T_{\hat{r}}}} = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1} \Big[\frac{\hat{r}_t^2}{\hat{d}_t^2\sigma^2_{T_{\hat{d}}} + \hat{r}_t^2\sigma^2_{T_{\hat{r}}} + \sigma^2_{T_1}} - \frac{\left(\epsilon^{[j]}_{T_t}\right)^2 \hat{r}_t^2}{\left(\hat{d}_t^2\sigma^2_{T_{\hat{d}}} + \hat{r}_t^2\sigma^2_{T_{\hat{r}}} + \sigma^2_{T_1}\right)^2} \Big] \tag{3.81}$$

$$\frac{\partial \mathcal{L}^*_{\epsilon_T}}{\partial \sigma^2_{T_{\hat{1}}}} = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1} \Big[\frac{1}{\hat{d}_t^2\sigma^2_{T_{\hat{d}}} + \hat{r}_t^2\sigma^2_{T_{\hat{r}}} + \sigma^2_{T_1}} - \frac{\left(\epsilon^{[j]}_{T_t}\right)^2}{\left(\hat{d}_t^2\sigma^2_{T_{\hat{d}}} + \hat{r}_t^2\sigma^2_{T_{\hat{r}}} + \sigma^2_{T_1}\right)^2} \Big] \tag{3.82}$$

$$\frac{\partial \mathcal{L}^*_{\epsilon_E}}{\partial \sigma^2_{E_{\hat{d}}}} = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1} \Big[\frac{\hat{d}_t^2}{\hat{d}_t^2\sigma^2_{E_{\hat{d}}} + \hat{r}_t^2\sigma^2_{E_{\hat{r}}} + \sigma^2_{E_1}} - \frac{\left(\epsilon^{[j]}_{E_t}\right)^2 \hat{d}_t^2}{\left(\hat{d}_t^2\sigma^2_{E_{\hat{d}}} + \hat{r}_t^2\sigma^2_{E_{\hat{r}}} + \sigma^2_{E_1}\right)^2} \Big] \tag{3.83}$$

$$\frac{\partial \mathcal{L}^*_{\epsilon_E}}{\partial \sigma^2_{E_{\hat{r}}}} = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1} \Big[\frac{\hat{r}_t^2}{\hat{d}_t^2\sigma^2_{E_{\hat{d}}} + \hat{r}_t^2\sigma^2_{E_{\hat{r}}} + \sigma^2_{E_1}} - \frac{\left(\epsilon^{[j]}_{E_t}\right)^2 \hat{r}_t^2}{\left(\hat{d}_t^2\sigma^2_{E_{\hat{d}}} + \hat{r}_t^2\sigma^2_{E_{\hat{r}}} + \sigma^2_{E_1}\right)^2} \Big] \tag{3.84}$$

$$\frac{\partial \mathcal{L}^*_{\epsilon_E}}{\partial \sigma^2_{E_{\hat{1}}}} = -\frac{1}{2}\sum_j \sum_{t=0}^{T-1} \Big[\frac{1}{\hat{d}_t^2\sigma^2_{E_{\hat{d}}} + \hat{r}_t^2\sigma^2_{E_{\hat{r}}} + \sigma^2_{E_1}} - \frac{\left(\epsilon^{[j]}_{E_t}\right)^2}{\left(\hat{d}_t^2\sigma^2_{E_{\hat{d}}} + \hat{r}_t^2\sigma^2_{E_{\hat{r}}} + \sigma^2_{E_1}\right)^2} \Big] . \tag{3.85}$$

To compute the maximum likelihood values of the sensor model parameters in the M-step, we first calculate soft assignments of each individual sensor reading s_k to the four components of our sensor model. That is, we calculate the probability $\rho_{k,\text{ccc}}$ that the sensor reading s_k was generated by component ccc of our sensor model, where ccc $\in \{\text{hit}, \text{short}, \text{max}, \text{rand}\}$. That is

$$\rho_{k,\text{hit}} = \eta p_{\text{hit}}\left(s_k|\mathbf{x}_k\right) \tag{3.86}$$

$$\rho_{k,\text{short}} = \eta p_{\text{short}}\left(s_k|\mathbf{x}_k\right) \tag{3.87}$$

$$\rho_{k,\text{max}} = \eta p_{\text{max}}\left(s_k|\mathbf{x}_k\right) \tag{3.88}$$

$$\rho_{k,\text{rand}} = \eta p_{\text{rand}}\left(s_k|\mathbf{x}_k\right) \quad , \tag{3.89}$$

where $\mathbf{x}_k$ is the location where the sensor reading was taken and η is a normalization constant to ensure that the above four values sum to one. We then calculate the maximum likelihood values of the parameters (where $\mathcal{S}$ denotes the set of all sensor measurements s_k)

$$\alpha_{\text{hit}} = \frac{\sum_k \rho_{k,\text{hit}}}{|\mathcal{S}|} \tag{3.90}$$

$$\alpha_{\text{short}} = \frac{\sum_k \rho_{k,\text{short}}}{|\mathcal{S}|} \tag{3.91}$$

$$\alpha_{\text{max}} = \frac{\sum_k \rho_{k,\text{max}}}{|\mathcal{S}|} \tag{3.92}$$

$$\alpha_{\text{rand}} = \frac{\sum_k \rho_{k,\text{rand}}}{|\mathcal{S}|} \tag{3.93}$$

$$\sigma_{\text{hit}} = \sqrt{\frac{\sum_k \rho_{k,\text{hit}} \left(s_k - s_k^*\right)^2}{\sum_k \rho_{k,\text{hit}}}} \tag{3.94}$$

$$\lambda_{\text{short}} = \frac{\sum_k \rho_{k,\text{short}}}{\sum_k \rho_{k,\text{short}} s_k} \ , \tag{3.95}$$

where s_k^* is the true range of the object which can be easily computed by performing ray tracing on the map $\mathcal{M}$. See Thrun et al. [2005] for a detailed derivation of the above formulas. We should point out that the process of computing the maximum likelihood values of the sensor model parameters constitute an EM algorithm itself. In our experiments, we compute the maximum likelihood sensor parameters only once per iteration of our EM framework using the above formulas rather than having another EM algorithm nested within our EM framework. The derivation of the EM algorithm in [Neal and Hinton, 1998] justifies this partial optimization step as well as the approximate E-step; our EM algorithm can still be viewed as optimizing the likelihood of the robot's trajectory. Figure 3.1 shows the block diagram for the parameter estimation framework we used.

3.6 Experimental Results

To demonstrate the effectiveness of the estimation approach discussed in the previous section, we present the results of robotic experiments we conducted. In our experiments, we used an ActivMedia Robotics P3-DX as our testbed robot. The robot is equipped with a front sonar array with eight sensors, one on each side and six forward at 20^{o} intervals. It also has a rear

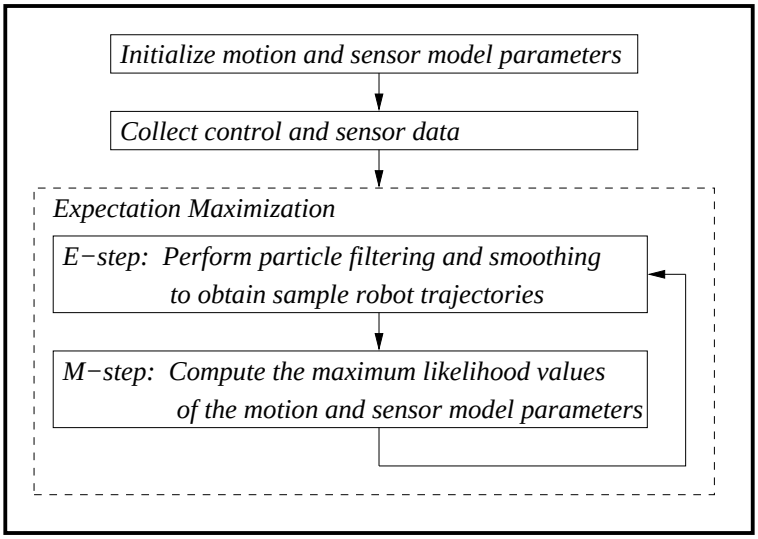


Figure 3.1: The block diagram for the parameter estimation framework. Figure taken from [Yap and Shelton, 2008] © [2008] IEEE.

sonar array with eight sensors, one on each side and six rear at 20^{o} intervals. It is controlled by an IBM ThinkPad X32 notebook computer.

In our experiments, we considered two test environments (see Figure 3.2 and Figure 3.3). The first test environment is a makeshift environment we set up that represents a scaled-down version of a typical office environment. The associated map of the environment has an approximate size of $6.7\text{m} \times 6.7\text{m}$. The second test environment is a portion of the South wing of the third floor of our Computer Science building. Compared to the first test environment, the second test environment is significantly bigger with a map size of approximately $38\text{m} \times 30\text{m}$. The second test environment is more realistic than the first test environment in that it is dynamic and it contains many unmodeled objects and obstacles (e.g. people walking around, trash bins). For both test environments, we instructed the robot to autonomously navigate through the environment by visiting predefined waypoints and returning to the start-

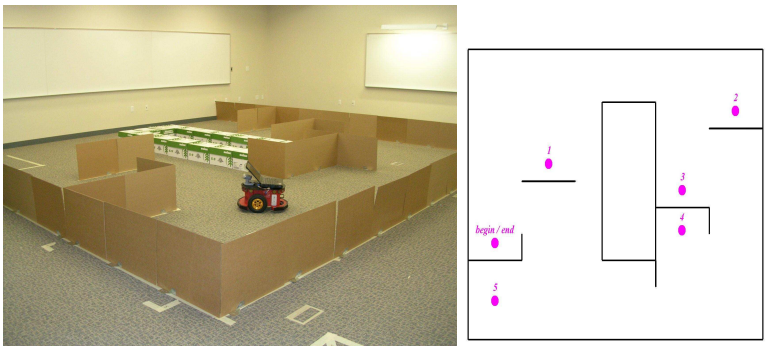


Figure 3.2: The first test environment (left) and its associated map with waypoints (right) for the parameter estimation. Figure taken from [Yap and Shelton, 2008] © [2008] IEEE.

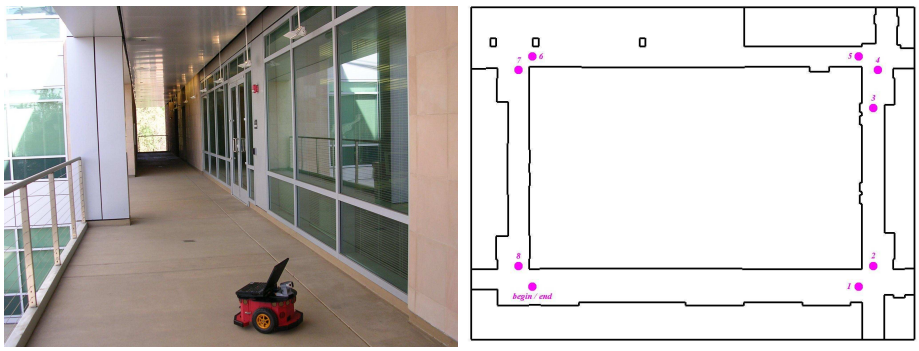


Figure 3.3: The second test environment (left) and its associated map with waypoints (right) for the parameter estimation. Figure taken from [Yap and Shelton, 2008] © [2008] IEEE.

ing position while collecting control and sensor data along the way. In our experiments, the parameter estimation routine is carried out off-line on a different machine for computational reasons. Table 3.1 provides summary information about our experiments. Notice that the parameter estimation routine currently runs approximately ten times slower than real time (i.e. data collection time) but with further advances in processor speedup and code optimization we can expect the parameter estimation routine to achieve real-time execution. Another way to speed up the estimation process is to consider only portions of the collected data set instead of using the entire data set.

Table 3.1: Parameter estimation experiments summary.

	Test Environment 1	Test Environment 2
Map Sizes	$\approx 6.7\text{m} \times 6.7\text{m}$	$\approx 38\text{m} \times 30\text{m}$
Data Collection Times (1 round)	≈ 5minutes	≈ 13minutes
Total Time Steps T	601	1008
Number of Sonar Measurements	9616	16128
Parameter Estimation Times	53m38s	2h7m52s

Table 3.2 shows the values of the parameters we obtained after performing our estimation method starting from some initial crude (uncalibrated) model parameters and using the historical account of the robot's motion and perception. As can be seen in Table 3.2, the estimated values of the motion parameters for both test environments are similar except for the values of the parameters $\sigma^2_{T_1}$, $\sigma^2_{D_1}$, and $\sigma^2_{E_1}$. These are all related to the non-proportional variance in the robot's translational and rotational motions. The drastic difference between the estimated values in the two environments is natural given the differences in floor material: carpet in the first test environment and concrete with expansion joints in the second test environment.

The differences in sensor model parameters are similarly explainable. Recall that the parameter α_{short} represents the probability that a particular sensor measurement is caused by unexpected objects in the environment such as people walking along the corridors and other unmodeled objects (e.g. trash bins) not included in the map. Since the second test environment is highly dynamic and contains many unexpected objects, the learned value for parameter α_{short} is 3.4 times larger than the value obtained from the first test environment.

Table 3.2: The initial and estimated values of the parameters.

		Estimated Parameter Values	
Initial Parameter Values		Test Environment 1	Test Environment 2
$\sigma^2_{T_{\hat{r}}}$	0.01	0.338267	0.348720
$\sigma^2_{T_{\hat{d}}}$	0.01	0.000345	0.000318
$\sigma^2_{T_1}$	0.01	**0.666048**	**0.007811**
$\sigma^2_{D_{\hat{r}}}$	0.01	0.010731	0.007965
$\sigma^2_{D_{\hat{d}}}$	0.01	0.021869	0.044325
$\sigma^2_{D_1}$	0.01	**0.000001**	**0.010533**
$\sigma^2_{E_{\hat{r}}}$	0.01	0.013427	0.026958
$\sigma^2_{E_{\hat{d}}}$	0.01	0.008588	0.005744
$\sigma^2_{E_1}$	0.01	**0.000014**	**0.001121**
α_{hit}	0.30	0.434601	0.319870
α_{short}	0.20	**0.029356**	**0.100010**
α_{max}	0.30	**0.348269**	**0.408525**
α_{rand}	0.20	0.187774	0.171595
σ_{hit}	500.00	31.18050	23.65530
λ_{short}	0.15	0.001094	0.000651

We also notice that the value of α_{max}, the probability that the range finder would fail to detect obstacles, is higher by approximately 17% in the second test environment than the value in the first test environment. The second test environment also contains smooth glass walls along the corridors that can cause specular reflections thus making the walls invisible to the sensor.

After obtaining the estimated parameters, we also tested their effect on the speed of robot navigation. The robot in this experiment is commanded by a controller module that uses the estimated robot pose for issuing control commands (such as the amount of translational and rotational velocities) for navigating from one waypoint to the other. The estimated robot pose is provided by a localizer module that uses the motion and sensor models discussed in

Table 3.3: The navigation times of the robot. Please see text for discussion.

	Navigation Times		
	A	B	C
Test Environment 1 (5 rounds)	25m47s	25m41s	**22m43s**
Test Environment 2 (2 rounds)	26m19s	25m58s	**25m3s**

Sections 3.3.1 and 3.3.2, respectively. The correctness of the control commands issued by the controller module is dependent on the accuracy of the estimated robot pose according to the localizer module. The time to navigate is in turn dependent on the correctness of the control commands given by the controller module. In these experiments, we compared the times it took the robot to navigate through the same path in both environments when the localizer module uses: A) the uncalibrated motion and sensor models, B) only the calibrated motion model, and C) both calibrated motion and sensor models. Table 3.3 shows the navigation times for these experiments. As can be seen in Table 3.3, using both calibrated motion and sensor models led to noticeable speedups in robot navigation. This result is in line with the preceding argument: the model parameters affect the accuracy of the robot pose, the accuracy of the robot pose in turn affects the correctness of the control signals for driving the robot to its goal or destination, and finally, the correctness of the control signals affects the amount of time the robot needs to navigate to its destination. The results of this experiment support the fact that calibrated robot performs better than uncalibrated ones.

Finally, to demonstrate the appropriateness of the estimated parameter values in estimating the robot path, we constructed sonar maps by plotting the endpoints of sonar readings

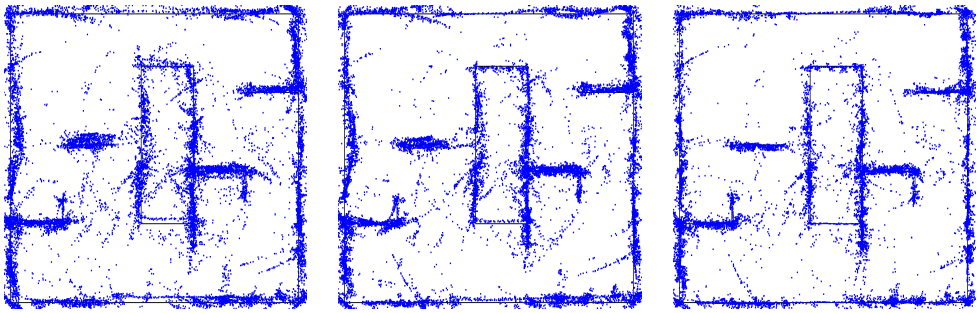


Figure 3.4: The generated sonar maps using the uncalibrated motion and sensor models (left), calibrated motion model only (middle), calibrated motion and sensor models (right). The (blue) dots represent the endpoints of sonar readings with respect to the estimated path of the robot. The true map is displayed for easier visual inspection. Figure taken from [Yap and Shelton, 2008] © [2008] IEEE.

with respect to the estimated path of the robot.[4] Figure 3.4 shows the generated sonar maps displayed on top of the true map of the environment for easier visual inspection. It is easy to see that the sonar map generated using both the calibrated motion and sensor models is better than those generated using the uncalibrated motion and sensor models or using only the calibrated motion model since the sonar readings are well aligned with the actual map of the environment. For a more quantitative comparison, we generated the cumulative plot in Figure 3.5 which shows the percentage of sonar endpoints that are closer than a given distance from the nearest wall in the environment. As Figure 3.5 shows, over 50% of the sonar endpoints are closer than $d = 50$mm to the nearest wall for the sonar map generated using both the calibrated motion and sensor models while only about 33% of the sonar endpoints are closer than $d = 50$mm to the nearest wall for the sonar map generated with the uncalibrated motion and sensor models.

[4] We do not include the maximum sonar readings $s_{\max}$ in the plot since they provide us with little information as to the location of the objects or obstacles.

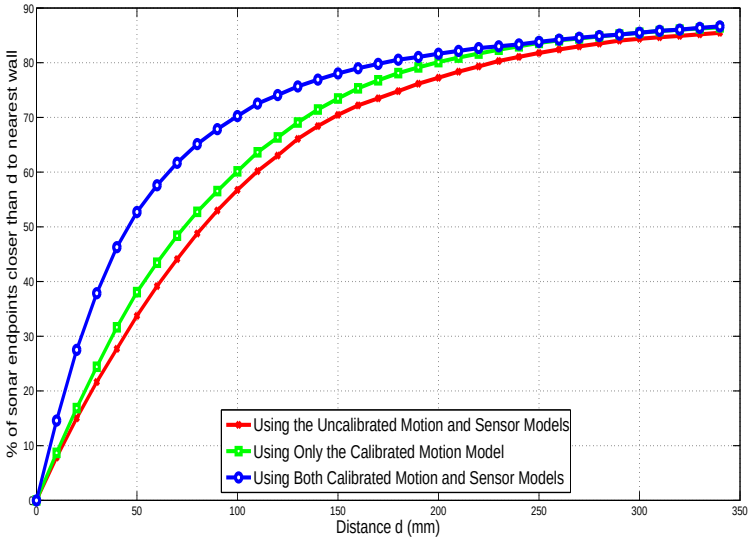


Figure 3.5: Percentage of sonar endpoints that are less than a certain distance from the nearest wall in the environment. Figure taken from [Yap and Shelton, 2008] © [2008] IEEE.

3.7 Summary

Robot calibration is an important activity in mobile robotics. However, current calibration techniques often involve intensive and carefully set up calibration experiments as well as a significant amount of human effort. In this chapter, we presented an automated technique for calibrating a mobile robot's motion and sensor models based from the control and sensor data obtained naturally during robot operations. Our method is based on a standard machine learning method called the EM algorithm. Starting from some initial parameter values, our method iteratively optimizes the parameters based from the data collected during normal robot operation. Unlike other calibration techniques, our method does not require any special calibration setup and it can be performed by the robot as it operates with little or no human intervention. Since the parameters are estimated from actual data, the robot can learn a more

appropriate set of parameter values. The estimation procedure can be readily invoked by the robot at a later time when there is a need to recalibrate the models. The results of some actual robotic experiments are presented that show the effectiveness of our approach. As pointed out, our parameter estimation framework is general in that it is not tied to any particular motion and sensor models and it can be used for other motion and sensor models as well.

Chapter 4

Mobile Robot Localization

4.1 Introduction

For nearly two decades now, mobile robot localization, the problem of estimating the robot's pose in a given environment from sensor measurements, has been the subject of investigation in the field of robotics [Crowley, 1989, Cox, 1991, Leonard and Durrant-Whyte, 1991a, Weißet al., 1994, Fox et al., 1999]. Researchers have made great strides and progress in tackling this fundamental problem. To date, a variety of sensors have already been considered and demonstrated for pose estimation including wheel encoders, inertial sensors, rangefinders (such as infrared, ultrasonic, and laser), cameras (e.g. omnidirectional, monocular, binocular, trinocular, and multi-camera), microphone arrays, WiFi sensors, global positioning system (GPS), and directional antennas. While it is evident that high-end sensors are capable of delivering superior localization performance than low-end ones, their high costs, power con-

sumption, hardware requirements, and computational demands do not justify their application to small-scale robots such as domestic, educational, service, and entertainment robots. We are interested in the use of low-cost sensors for small-scale robots while still achieving good real-time localization accuracy in an unmodified indoor environment.

In this chapter, we present an approach to real-time pose estimation for a small-scale indoor mobile robot equipped with wheel encoders for its odometry and aided by a standard perspective camera without an *a priori* map of the environment. Virtually all wheeled robotic vehicles (and legged robots, to some extent) come equipped with wheel encoders as their primary proprioceptive sensors for ego-motion estimation. While wheel encoders are only reliable for very short distances of travel as they can quickly suffer from accumulated errors over time, their extremely low cost and wide default availability in most robotic platforms could still render them as the sensor of choice for the navigation of small-scale indoor robots. In order to compensate for the limitations of wheel encoders for pose estimation, additional sensors are usually employed. Whereas there are several equally lightweight and low-cost sensors for aiding odometry such as sonars and infrared rangefinders, we prefer to use a single standard front-looking perspective camera for the following reasons: it is not intrusive, it is becoming more pervasive and common, it provides rich information, and it is versatile in that it can also be used for other robotic tasks (e.g. object or place detection and recognition, situation understanding, etc.). We use vision for detecting naturally occurring static three-dimensional point features or landmarks from the environment and utilize the information for correcting the pose as suggested by the odometry. Using vision to estimate the three-

dimensional position of a static point feature requires that it be observed in at least two different camera poses. Since our robot is equipped with only a single camera, it has to rely on small motions and observe various features from several vantage points. As already pointed out by Mourikis and Roumeliotis [2006, 2007], the observation of a static feature from several camera poses defines geometric constraints involving all those poses. They used an extended Kalman filter (EKF) to estimate a history of up to N_{p} recent camera poses for processing the feature measurements as the measurements of each tracked feature are utilized for imposing constraints between all the camera poses from which the feature was observed.

Instead of using an EKF, we use a particle filter to sample over the space of the most recent N_{p} robot poses (and camera poses, assuming we know the fixed transformation from the robot frame to the camera frame). The most recent N_{p} camera poses are used for processing the feature measurements. With a particle filter, we avoid the linearization errors associated with the EKF. We introduce a measurement model for assigning weights to the particles based on their recent trajectory and the measurements of the features observed along the trajectory. We validate the effectiveness of the particle filter approach extensively with both simulation as well as real-world data and compare its performance against that of the EKF. Results from the tests show that the particle filter is better than the EKF in terms of the root mean squared error (RMSE) for the simulation data and it is capable of good real-time localization accuracy in unmodified indoor environments.

4.2 Related Work

The use of single camera setup for aiding robotic navigation has already been considered by a number of researchers in the past. As opposed to stereo or multi-camera systems, a single camera is inherently a bearing-only sensor, i.e. each image in itself does not contain sufficient information for determining the locations of features or landmarks that are observed. Chenavier and Crowley [1992] used an EKF to fuse the position estimation from odometry with observations of fixed objects in the environment from a camera mounted on the robot. Fixed objects serve as landmarks and they are assumed to be listed in a database. The system calculates the angle to each landmark and then orients the camera. The EKF is used to correct the pose from the error between the observed and estimated angle to each landmark. In our work, we use the camera to detect and temporarily estimate naturally occurring static three-dimensional point features in the environment for localization. We do not assume the existence of an *a priori* database of features in the environment to aid in localization.

In order to determine the locations of static point features with a single camera, they have to be observed from multiple view points. Inferring the positions of features using images taken from multiple view points is widely known in the computer vision community as the structure-from-motion (SFM) problem. Methods for solving the SFM problem commonly work off-line and process all the obtained images in a batch fashion. For example, Royer et al. [2005, 2007] presented a method for computing the localization of a mobile robot with reference to a learning video sequence. Their method follows a three-step approach. In the

learning step, the robot is manually guided on a path while a video sequence is recorded with its front-looking camera. In the second step, a 3D reconstruction of the path and the environment is computed off-line from the learning sequence. In the navigation step, the robot uses the 3D reconstruction for its pose estimation in real time and follows the learning path or a slightly different path in autonomous navigation. Although such methods can provide very accurate and robust reconstruction of the path and the environment (in light of the fact that such methods utilize all of the available information at once), they may not be suitable in cases (such as ours) where on-line and real-time operation is called for. Moreover, using a learning sequence limits the operational applicability of the robot to that of the training environment. Thus, rather than computing the 3D reconstruction off-line, Mouragnon et al. [2006] described a method for computing in real time the robust estimates of the camera poses as well as the 3D map of the environment. The 3D map is constantly refined via a fast and local bundle adjustment method producing accurate and reliable results. However, being an SFM method, it does not assume feedback from information sources such as odometry which is commonly (and in our case, readily) available in filtering approaches in robotics.

In robotics, the problem of jointly estimating the robot's (or camera's) pose and the landmarks' positions from sensor measurements is referred to as the simultaneous localization and mapping (SLAM) problem as discussed in Chapter 1. The SLAM problem is customarily tackled with the use of recursive and probabilistic filtering framework such as the EKF and particle filter. Davison [2003] used the EKF estimation framework for achieving real-time SLAM with a single camera. The overall system state vector of the EKF includes the

state of the camera (its current position and orientation) and the states of the feature points (their 3D locations in the environment). The state vector is accompanied by a full covariance matrix to represent the uncertainty (to the first order) in all the entries in the state vector.

Pupilli and Calway [2005] described a particle filter (instead of an EKF) method for vision-based tracking of a hand-held calibrated camera in real time. The particle filter is used to track the 3D motion parameters of the camera. The main motivation for the use of the particle filter over the EKF is its ability to deal with non-linearities and non-Gaussianity in the system (as mentioned in Chapter 2), thus providing improved robustness in tracking. Instead of tracking a camera undergoing general 3D motions, Karlsson et al. [2005] proposed the vSLAM algorithm for SLAM of a camera-equipped robot moving on a planar surface. The algorithm is odometry- and vision-based and applies the Rao-Blackwellized particle filter [Murphy, 2000] for estimating the full posterior distribution over the robot pose and landmark locations. Similar to FastSLAM [Montemerlo et al., 2002], it is based on an exact factorization of the posterior into a product of conditional landmark distributions and a distribution over robot paths. Thus, the entire problem is broken down into a robot localization problem and a number of landmark estimation problems that are conditioned on the robot pose estimate. A particle filter is used for estimating the posterior over robot paths. Each particle possesses a number of Kalman filters, each one to estimate one of the landmark locations conditioned on the path estimate. A FastSLAM-type particle filter is also used in a monocular SLAM system by Eade and Drummond [2006]. The main advantage of SLAM-based algorithms is that they maintain the correlations between the robot (or camera)

pose and the locations of the features observed. However, maintaining those correlations is computationally expensive thus prohibiting current SLAM-based algorithms to be applied in large environments with thousands of features.

In order to reduce the complexity and achieve real-time operation, some algorithms only estimate the robot or camera poses and forgo including the feature positions in the state estimation. In these approaches, visual feature observations are used to initialize temporary landmarks, the landmarks are used for localization, and then they are discarded. Mourikis and Roumeliotis [2006, 2007] introduced the multi-state constraint Kalman filter (MSCKF) for vision-aided inertial navigation of a vehicle localizing in a large-scale urban environment. In the MSCKF, the overall state vector consists of the state of the inertial measurement unit as well as the N_{p} camera poses. They introduced a measurement model capable of expressing the geometric constraints resulting from the observations of static features from the camera poses in the state vector without requiring the inclusion of the 3D feature positions in the state vector. As such, their algorithm is fast with a computational complexity that is linear in the number of features processed and it is capable of achieving accurate pose estimates in large-scale real-world environments.

In this chapter, we investigate the use of a particle filter to sample over the space of the most recent N_{p} robot or camera poses instead of an EKF. This is motivated by the fact that the particle filter is able to deal with non-linearities and non-Gaussianity in the system. We avoid the linearization errors associated with the EKF and introduce a measurement model for assigning weights to the particles based on their recent trajectory and the measurements

of the features observed along the trajectory. We validate the particle filter approach using simulation and real-world data and compare its performance against that of the MSCKF.

4.3 Particle Filtering Approach

The objective of our particle filter is to track the pose of a wheeled robotic vehicle navigating around an indoor environment with the use of odometry and the observations of static and naturally occurring point features or landmarks in the environment through its front-looking camera. As before, our robot operates in a planar environment so we represent the robot pose at time t as a column vector $\mathbf{x}_t = (x_t, y_t, \theta_t)^T$, where $(x_t, y_t)^T$ are its two-dimensional Cartesian coordinates and θ_t is its heading or orientation with respect to some global reference frame G. Here, we assume that the global reference frame coincides with the initial robot frame R_0 at time 0.

Each particle in our particle filter is an estimate of the recent N_p robot poses (i.e. the ith particle at time t is $\mathbf{x}^{[i]}_{t-N_\text{p}+1:t} \triangleq \left(\mathbf{x}^{[i]}_{t-N_\text{p}+1}, \mathbf{x}^{[i]}_{t-N_\text{p}+2}, ..., \mathbf{x}^{[i]}_{t}\right)$. Thus, the particles collectively sample the space of the recent N_p robot poses. Similar to Mourikis and Roumeliotis [2006, 2007], multiple states are necessary for processing the feature measurements as the measurements of the features are utilized for imposing constraints between the different poses from which those features are observed. Feature measurements are incrementally processed for features that are observed or tracked for at least two and up to a maximum of N_p frames. A feature that is tracked for more than N_p frames will be considered as a new feature starting at

the $(N_\mathrm{p}+1)$th frame. Each particle in the particle filter is assigned a weight $w_t^{[i]}$ based on the feature measurements. Let $\mathcal{X}_t = \left\{ \left(\mathbf{x}_{t-N_\mathrm{p}+1:t}^{[i]}, w_t^{[i]} \right) : i = 1, 2, ..., N_\mathrm{s} \right\}$ be the set of weighted particles at time t, where N_s is the total number of particles in the particle filter. Note that we do not represent the position of landmarks in the particle filter unlike SLAM techniques. In the next subsections, we discuss how the particles are propagated from one time step to the next and how the particles are weighted based on the estimates and the observed feature measurements.

4.3.1 Motion Model

As in Chapter 3, the robot is equipped with odometry that provides an estimate of the robot pose at every time step by integrating wheel encoder information. We use the estimated robot poses $\hat{\mathbf{x}}_{t-1}$ and $\hat{\mathbf{x}}_t$ to compute the relative motion (i.e. $\hat{d}_t$ and $\hat{r}_t$) of the robot during the time interval $[t-1, t)$ (see Equations 3.1 and 3.2) and let the probabilistic motion model $p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{c}_t\right)$ be conditioned on the control command $\mathbf{c}_t = \left(\hat{d}_t, \hat{r}_t\right)^T$. In this chapter, we use the motion model by Roy and Thrun [1999] which we repeat here for convenience:

$$x_t = x_{t-1} + d_t \cos\left(\theta_{t-1} + r_t\right) \tag{4.1}$$

$$y_t = y_{t-1} + d_t \sin\left(\theta_{t-1} + r_t\right) \tag{4.2}$$

$$\theta_t = \left(\theta_{t-1} + r_t\right) \bmod 2\pi \ , \tag{4.3}$$

where d_t and r_t denote the robot's true translation and rotation, respectively. The true translation and rotation are related to the estimated translation and rotation from the odometry via

$$d_t = \hat{d}_t + \delta_{\text{trans}} \left| \hat{d}_t \right| + \epsilon_{\text{trans}} \tag{4.4}$$

$$r_t = \hat{r}_t + \delta_{\text{rot}} \left| \hat{d}_t \right| + \epsilon_{\text{rot}} \ , \tag{4.5}$$

where δ_{trans} and δ_{rot} describe the systematic errors, and $\epsilon_{\text{trans}} \sim \mathcal{N}(0, \sigma^2_{\text{trans}})$ and $\epsilon_{\text{rot}} \sim \mathcal{N}(0, \sigma^2_{\text{rot}})$ are the additive random variables representing the random noise. In our experiments, $\delta_{\text{trans}} = 1 \times 10^{-3}$, $\delta_{\text{rot}} = 1 \times 10^{-3}$, $\sigma_{\text{trans}} = 0.002$m, and $\sigma_{\text{rot}} = 0.5^\circ$.

Given the ith particle $\mathbf{x}^{[i]}_{t-N_\text{p}:t-1} = \left(\mathbf{x}^{[i]}_{t-N_\text{p}}, \mathbf{x}^{[i]}_{t-N_\text{p}+1}, ..., \mathbf{x}^{[i]}_{t-1}\right)$ at time $t-1$, we use the motion model and the latest pose $\mathbf{x}^{[i]}_{t-1}$ at time $t-1$ to sample the pose $\mathbf{x}^{[i]}_t$ at time t (i.e. $\mathbf{x}^{[i]}_t \sim p\left(\mathbf{x}_t \left| \mathbf{x}^{[i]}_{t-1}, \mathbf{c}_t \right.\right)$). We then append the pose $\mathbf{x}^{[i]}_t$ to $\mathbf{x}^{[i]}_{t-N_\text{p}:t-1}$ to obtain the intermediate[1] particle $\mathbf{x}^{[i]}_{t-N_\text{p}:t}$ at time t.

4.3.2 Sensor Model

After propagating the particles, we now assign weights to the particles based on their estimates and the observed feature measurements along the trajectory hypothesized by the particles. In the particle filtering framework, the weight of each particle is proportional to the likelihood of the observations given the particle estimates (see Chapter 2 Section 2.4). In this

[1]The particle is intermediate as it still has an extra pose needed for processing the feature measurements in the update stage of the filter.

chapter, our robot is equipped with a standard perspective front-looking camera for observing naturally occurring static three-dimensional point features or landmarks in the environment. Viewing a static feature from multiple camera poses results in constraints involving all those poses. Camera observations are grouped by tracked features and they are used for assigning weights to the particles.

First, we consider a single feature f_j that is observed in $M_{f_j,t}$ camera poses by time t and we use the observations of feature f_j to define constraints on the consecutive robot poses where it is observed. The weight of the ith particle at time t based from observing feature f_j is

$$w_t^{(j)[i]} = p\left(\mathcal{O}_t^{(j)} \middle| \mathbf{x}_{t-N_\mathrm{p}:t}^{[i]}\right) \ , \tag{4.6}$$

where $\mathcal{O}_t^{(j)} \triangleq \left\{\mathbf{o}_k^{(j)} : k \in \mathcal{S}_{f_j,t}\right\}$ is the set of normalized image coordinates of feature f_j in the camera frames C_k and $\mathcal{S}_{f_j,t}$ is the set of camera frames $\left\{C_k\right\}$ where feature f_j is observed by time t. Averaging over the potential position of feature f_j, denoted as $\mathbf{f}_j$, we get

$$w_t^{(j)[i]} = \int p\left(\mathcal{O}_t^{(j)} \middle| \mathbf{x}_{t-N_\mathrm{p}:t}^{[i]}, \mathbf{f}_j\right) p\left(\mathbf{f}_j \middle| \mathbf{x}_{t-N_\mathrm{p}:t}^{[i]}\right) d\mathbf{f}_j \ . \tag{4.7}$$

Our noise model is captured by

$$p\left(\mathcal{O}_t^{(j)} \middle| \mathbf{x}_{t-N_\mathrm{p}:t}^{[i]}, \mathbf{f}_j\right) \propto \prod_{k \in \mathcal{S}_{f_j,t}} e^{-\frac{1}{2}\frac{\left(\mathbf{o}_k^{(j)} - \psi_k\left(\mathbf{f}_j\right)\right)^T \left(\mathbf{o}_k^{(j)} - \psi_k\left(\mathbf{f}_j\right)\right)}{\sigma_{\mathrm{im}}^2}} \ , \tag{4.8}$$

where $\psi_k(\mathbf{f}_j)$ is the projection of $\mathbf{f}_j$ in the camera frame $\left\{C_k\right\}$ at time t. Note that in Equation 4.8, we have assumed that the errors in the feature observations in the image planes are independent and normally distributed with mean 0 and variance σ_{im}^2. We also assume $p\left(\mathbf{f}_j \left| \mathbf{x}_{t-N_\text{p}:t}^{[i]} \right.\right) = p(\mathbf{f}_j)$: the feature position in the environment is independent of the actual robot or camera trajectory. Hence, Equation 4.7 becomes

$$w_t^{(j)[i]} = \int p\left(\mathcal{O}_t^{(j)} \left| \mathbf{x}_{t-N_\text{p}:t}^{[i]}, \mathbf{f}_j \right.\right) p(\mathbf{f}_j)\, d\mathbf{f}_j \ . \tag{4.9}$$

We select $p(\mathbf{f}_j)$ in Equation 4.9 to be the uninformative[2] distribution over the potential position $\mathbf{f}_j$ of feature f_j in the environment. We can therefore safely drop the term $p(\mathbf{f}_j)$ from the above:

$$w_t^{(j)[i]} \propto \int p\left(\mathcal{O}_t^{(j)} \left| \mathbf{x}_{t-N_\text{p}:t}^{[i]}, \mathbf{f}_j \right.\right) d\mathbf{f}_j \ . \tag{4.10}$$

In order to compute the integral in Equation 4.10, we introduce a proposal distribution $q(\mathbf{f}_j)$ over the potential feature position $\mathbf{f}_j$ from which to sample:

$$w_t^{(j)[i]} = \int \frac{p\left(\mathcal{O}_t^{(j)} \left| \mathbf{x}_{t-N_\text{p}:t}^{[i]}, \mathbf{f}_j \right.\right)}{q(\mathbf{f}_j)} q(\mathbf{f}_j)\, d\mathbf{f}_j \tag{4.11}$$

$$= \mathbb{E}_{q(\mathbf{f}_j)}\left[\frac{p\left(\mathcal{O}_t^{(j)} \left| \mathbf{x}_{t-N_\text{p}:t}^{[i]}, \mathbf{f}_j \right.\right)}{q(\mathbf{f}_j)}\right] \ . \tag{4.12}$$

[2]This means that without any information, the feature can be located equally likely anywhere in the environment.

While the proposal $q(\mathbf{f}_j)$ can be any distribution, we choose $q(\mathbf{f}_j)$ to be $\mathcal{N}\left(\hat{\mathbf{f}}_j; \hat{\mathbf{f}}_j^*, \mathbf{C}^{(j)}\right)$, where $\hat{\mathbf{f}}_j^*$ is the estimated position of feature f_j and $\mathbf{C}^{(j)}$ is the associated covariance matrix of the estimate. There are several ways the above expectation can be evaluated and we choose the technique of the unscented transform [Julier and Uhlmann, 1997, Ito and Xiong, 2000] which we briefly discuss in Section 4.3.6.

Equation 4.12 provides the contribution of feature f_j to the weight of the ith particle at time t. Considering all the features that have been observed at time t, the weight $w_t^{[i]}$ of the ith particle can be updated as

$$w_t^{[i]} = w_{t-1}^{[i]} \prod_j \frac{w_t^{(j)[i]}}{w_{t-1}^{(j)[i]}} \ , \tag{4.13}$$

assuming that observed features are independent of each other. Note that in Equation 4.13, we divided $w_t^{(j)[i]}$ by $w_{t-1}^{(j)[i]}$ (i.e. the contribution of feature f_j to the weight of the ith particle at time $t-1$) in order to avoid "double counting" the contribution of feature f_j to the weight of the ith particle in case the observations of feature f_j were processed at the previous time step $t-1$. If the observations of feature f_j are processed for the first time at time t, then $w_{t-1}^{(j)[i]} = 1$. With this weighting scheme, feature observations are processed incrementally at every time step without waiting for the features to be lost or be tracked for a maximum number of frames before their measurements are processed. Note that while $w_{t-1}^{[i]}$ may have been changed (due to the resampling step of the particle filter), the individual feature weights, $w_{t-1}^{(j)[i]}$, were not. Therefore, Equation 4.13 does not directly reduce to $\prod_j w_t^{(j)[i]}$.

4.3.3 Feature Parametrization

The parametrization of feature f_j is critical to the success of using a Gaussian proposal distribution $q(\mathbf{f}_j)$. In this section, we discuss the parametrization we use for feature f_j.

We represent camera frame C_k as $\left({}^G_{C_k}\mathbf{R}, {}^G\mathbf{p}_{C_k}\right)$, where ${}^A_B\mathbf{R}$ is the 3×3 matrix describing the rotation between frames A and B, ${}^A\mathbf{p}_B$ is the 3D position of the origin of frame B with respect to frame A, and G is the global reference frame. Each of the $M_{f_j,t}$ observations of feature f_j is described by the pinhole camera model:

$$\mathbf{o}_k^{(j)} = \frac{1}{{}^{C_k}z_{f_j}} \begin{pmatrix} {}^{C_k}x_{f_j} \\ \\ {}^{C_k}y_{f_j} \end{pmatrix} + \mathbf{n}_k^{(j)} \ , \tag{4.14}$$

where $\left({}^{C_k}x_{f_j}, {}^{C_k}y_{f_j}, {}^{C_k}z_{f_j}\right)^T = {}^{C_k}\mathbf{p}_{f_j}$ is the position of feature f_j in camera frame C_k, $\mathbf{n}_k^{(j)}$ is the 2×1 image noise vector with known covariance matrix $\sigma_{\text{im}}^2\mathbf{I}_2$, and $\mathbf{I}_2$ is the 2×2 identity matrix. The position of feature f_j in camera frame C_k is given by

$${}^{C_k}\mathbf{p}_{f_j} = \begin{pmatrix} {}^{C_k}x_{f_j} \\ \\ {}^{C_k}y_{f_j} \\ \\ {}^{C_k}z_{f_j} \end{pmatrix} = {}^C_R\mathbf{R}\,{}^G_{R_k}\mathbf{R}^T({}^G\mathbf{p}_{f_j} - {}^G\mathbf{p}_{R_k}) + {}^C\mathbf{p}_R \ , \tag{4.15}$$

where

$$
{}^{G}_{R_k}\mathbf{R} = \begin{pmatrix} \cos(\theta_k) & -\sin(\theta_k) & 0 \\ \sin(\theta_k) & \cos(\theta_k) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.16}
$$

$$
{}^{G}\mathbf{p}_{R_k} = \begin{pmatrix} x_k \\ y_k \\ h \end{pmatrix} , \tag{4.17}
$$

h is the known height of the robot, ${}^{G}\mathbf{p}_{f_j}$ is the three-dimensional position of feature f_j in the global frame G, and ${}^{C}_{R}\mathbf{R}$ and ${}^{C}\mathbf{p}_{R}$ are the known extrinsic parameters of the camera frame C with respect to the robot frame R. Since the three-dimensional position ${}^{G}\mathbf{p}_{f_j}$ of the feature in the global frame G is unknown, we need to first estimate it using the measurements $\mathbf{o}_k^{(j)}$, $k \in \mathcal{S}_{f_j,t}$, and the $M_{f_j,t}$ camera poses where the feature was observed. However, instead of directly computing an estimate of the three-dimensional position of the feature in the global frame, we compute an estimate of the inverse-depth parametrization of the feature with respect to the last camera frame in which the feature was observed. Specifically, suppose C_n is the last camera frame the feature was observed, then the feature coordinates with respect to the kth camera frame are

$$
{}^{C_k}\mathbf{p}_{f_j} = {}^{C_k}_{C_n}\mathbf{R}\,{}^{C_n}\mathbf{p}_{f_j} + {}^{C_k}\mathbf{p}_{C_n} , \qquad k \in \mathcal{S}_{f_j,t} . \tag{4.18}
$$

Rewriting the above equation:

$$
{}^{C_k}\mathbf{p}_{f_j} = {}^{C_n}z_{f_j} \left({}^{C_k}_{C_n}\mathbf{R} \begin{pmatrix} \frac{{}^{C_n}x_{f_j}}{{}^{C_n}z_{f_j}} \\ \frac{{}^{C_n}y_{f_j}}{{}^{C_n}z_{f_j}} \\ 1 \end{pmatrix} + \frac{1}{{}^{C_n}z_{f_j}} {}^{C_k}\mathbf{p}_{C_n} \right) \tag{4.19}
$$

$$
= {}^{C_n}z_{f_j} \left({}^{C_k}_{C_n}\mathbf{R} \begin{pmatrix} \alpha_{f_j} \\ \beta_{f_j} \\ 1 \end{pmatrix} + \rho_{f_j} {}^{C_k}\mathbf{p}_{C_n} \right) \tag{4.20}
$$

$$
= {}^{C_n}z_{f_j} \begin{pmatrix} \psi_{k1}(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}) \\ \psi_{k2}(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}) \\ \psi_{k3}(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}) \end{pmatrix} , \tag{4.21}
$$

where α_{f_j}, β_{f_j}, and ρ_{f_j} are the inverse-depth parameters of feature f_j with respect to the last camera frame C_n and are defined as

$$
\alpha_{f_j} \triangleq \frac{{}^{C_n}x_{f_j}}{{}^{C_n}z_{f_j}} , \quad \beta_{f_j} \triangleq \frac{{}^{C_n}y_{f_j}}{{}^{C_n}z_{f_j}} , \quad \rho_{f_j} \triangleq \frac{1}{{}^{C_n}z_{f_j}} , \tag{4.22}
$$

and ψ_{k1}, ψ_{k2}, and ψ_{k3} are scalar functions of α_{f_j}, β_{f_j}, and ρ_{f_j}. We can now express the observation $\mathbf{o}_k^{(j)}$ of feature f_j in camera frame C_k as functions of the inverse-depth parameters

α_{f_j}, β_{f_j}, and ρ_{f_j}:

$$\mathbf{o}_k^{(j)} = \frac{1}{\psi_{k3}(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j})} \begin{pmatrix} \psi_{k1}(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}) \\ \\ \psi_{k2}(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}) \end{pmatrix} + \mathbf{n}_k^{(j)} \tag{4.23}$$

$$= \psi_k \left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right) + \mathbf{n}_k^{(j)} \ , \tag{4.24}$$

where

$$\psi_k \left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right) = \frac{1}{\psi_{k3}(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j})} \begin{pmatrix} \psi_{k1}(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}) \\ \\ \psi_{k2}(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}) \end{pmatrix} = \psi_k \left(\mathbf{f}_j\right) \ . \tag{4.25}$$

We let the parametrization of feature f_j be $\mathbf{f}_j = (\alpha_{f_j}, \beta_{f_j}, \rho_{f_j})^T$.

4.3.4 Feature Estimation Using Gauss-Newton Minimization

In this section, we discuss the Gauss-Newton minimization technique we used for estimating the inverse-depth parametrization $\mathbf{f}_j = \left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right)^T$ of feature f_j given the set of normalized image coordinates $\mathcal{O}^{(j)} \triangleq \left\{\mathbf{o}_k^{(j)} : k \in \mathcal{S}_{f_j,t}\right\}$ in the camera frames $C_k, k \in \mathcal{S}_{f_j,t}$. The output of this step will be the estimated inverse-depth parametrization $\hat{\mathbf{f}}_j^*$ and its associated covariance matrix $\mathbf{C}^{(j)}$. Note that $\hat{\mathbf{f}}_j^*$ is computed for each particle in the particle filter since each particle represents an estimate of the robot or camera trajectory.

With reference to Equation 4.8, the likelihood function of the parametrization $\hat{\mathbf{f}}_j$ of feature f_j is given by

$$\Lambda\left(\hat{\mathbf{f}}_j\right) = p\left(\mathcal{O}_t^{(j)} \middle| \mathbf{x}_{t-N_\mathrm{p}:t}^{[i]}, \hat{\mathbf{f}}_j\right) \tag{4.26}$$

$$\propto \prod_{k \in \mathcal{S}_{f_j,t}} e^{-\frac{1}{2}\left(\mathbf{o}_k^{(j)} - \psi_k\left(\hat{\mathbf{f}}_j\right)\right)^T \left(\mathbf{Q}_k^{(j)}\right)^{-1} \left(\mathbf{o}_k^{(j)} - \psi_k\left(\hat{\mathbf{f}}_j\right)\right)} , \tag{4.27}$$

where $\mathbf{Q}_k^{(j)} = \sigma_{\mathrm{im}}^2 \mathbf{I}_2$. Taking the logarithm of the likelihood function above, we get

$$\Lambda^*\left(\hat{\mathbf{f}}_j\right) = \text{const.} - \frac{1}{2} \sum_{k \in \mathcal{S}_{f_j,t}} \left(\mathbf{o}_k^{(j)} - \psi_k\left(\hat{\mathbf{f}}_j\right)\right)^T \left(\mathbf{Q}_k^{(j)}\right)^{-1} \left(\mathbf{o}_k^{(j)} - \psi_k\left(\hat{\mathbf{f}}_j\right)\right) . \tag{4.28}$$

Our objective is

$$\hat{\mathbf{f}}_j^* = \arg\max_{\hat{\mathbf{f}}_j} \Lambda^*\left(\hat{\mathbf{f}}_j\right) . \tag{4.29}$$

Let

$$\mathcal{J}\left(\hat{\mathbf{f}}_j\right) = \frac{1}{2} \sum_{k \in \mathcal{S}_{f_j,t}} \left(\mathbf{o}_k^{(j)} - \psi_k\left(\hat{\mathbf{f}}_j\right)\right)^T \left(\mathbf{Q}_k^{(j)}\right)^{-1} \left(\mathbf{o}_k^{(j)} - \psi_k\left(\hat{\mathbf{f}}_j\right)\right) . \tag{4.30}$$

Maximizing $\Lambda^*\left(\hat{\mathbf{f}}_j\right)$ with respect to $\hat{\mathbf{f}}_j$ is equivalent to minimizing $\mathcal{J}\left(\hat{\mathbf{f}}_j\right)$ with respect to $\hat{\mathbf{f}}_j$. Taking the derivative of $\mathcal{J}\left(\hat{\mathbf{f}}_j\right)$ with respect to $\hat{\mathbf{f}}_j$ and setting to 0:

$$0 = \nabla_{\hat{\mathbf{f}}_j} \mathcal{J}\left(\hat{\mathbf{f}}_j\right) \tag{4.31}$$

$$= -\sum_{k \in \mathcal{S}_{f_j,t}} \left(\mathbf{o}_k^{(j)} - \psi_k\left(\hat{\mathbf{f}}_j\right)\right)^T \left(\mathbf{Q}_k^{(j)}\right)^{-1} \mathbf{\Psi}_k^{(j)} , \tag{4.32}$$

where $\mathbf{\Psi}_k^{(j)} = \nabla_{\hat{\mathbf{f}}_j} \psi_k \left(\hat{\mathbf{f}}_j \right)$ is the gradient of the measurement function $\psi_k \left(\hat{\mathbf{f}}_j \right)$ with respect to $\hat{\mathbf{f}}_j$. Note that Equation 4.32 cannot be solved in closed form and we solve it using the Gauss-Newton iterative minimization.

The Gauss-Newton algorithm is an iterative method for solving nonlinear least-squares problem (e.g. Equation 4.32). Starting with an initial guess $\hat{\mathbf{f}}_j^{(0)}$ for the minimum, the method iteratively updates the guess via

$$\hat{\mathbf{f}}_j^{(l+1)} = \hat{\mathbf{f}}_j^{(l)} + \Delta \ , \tag{4.33}$$

where $\hat{\mathbf{f}}_j^{(l)}$ is the estimate for $\mathbf{f}_j$ at the lth iteration and Δ is the correction term. In the following, we derive the correction term Δ needed for iteratively updating the estimate $\hat{\mathbf{f}}^{(l)}$.

Computing the Taylor series expansion of $\mathcal{J} \left(\hat{\mathbf{f}}_j \right)$ up to the first-order around the estimate $\hat{\mathbf{f}}_j^{(l)}$ of the Gauss-Newton iteration, we have

$$\hat{\mathcal{J}} \left(\hat{\mathbf{f}}_j \right) \approx \mathcal{J} \left(\hat{\mathbf{f}}_j^{(l)} \right) + \nabla_{\hat{\mathbf{f}}_j} \mathcal{J} \left(\hat{\mathbf{f}}_j^{(l)} \right) \left(\hat{\mathbf{f}}_j - \hat{\mathbf{f}}_j^{(l)} \right) \ . \tag{4.34}$$

Taking the derivative of Equation 4.34 with respect to $\hat{\mathbf{f}}_j$ and setting to 0 yields

$$\nabla_{\hat{\mathbf{f}}_j} \mathcal{J} \left(\hat{\mathbf{f}}_j^{(l)} \right)^T + \left(\hat{\mathbf{f}}_j - \hat{\mathbf{f}}_j^{(l)} \right)^T \nabla_{\hat{\mathbf{f}}_j}^2 \mathcal{J} \left(\hat{\mathbf{f}}_j^{(l)} \right) = 0 \ , \tag{4.35}$$

where $\nabla^2_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)$ is the Hessian of $\mathcal{J}$ with respect to $\hat{\mathbf{f}}_j$ evaluated at $\hat{\mathbf{f}}_j^{(l)}$. Solving for $\hat{\mathbf{f}}_j$ in Equation 4.35

$$\nabla_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)^T + \left(\hat{\mathbf{f}}_j - \hat{\mathbf{f}}_j^{(l)}\right)^T \nabla^2_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right) = 0 \tag{4.36}$$

$$\left(\hat{\mathbf{f}}_j - \hat{\mathbf{f}}_j^{(l)}\right)^T \nabla^2_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right) = -\nabla_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)^T \tag{4.37}$$

$$\nabla^2_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)\left(\hat{\mathbf{f}}_j - \hat{\mathbf{f}}_j^{(l)}\right) = -\nabla_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)^T \tag{4.38}$$

$$\hat{\mathbf{f}}_j - \hat{\mathbf{f}}_j^{(l)} = -\left(\nabla^2_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)\right)^{-1}\nabla_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)^T \tag{4.39}$$

$$\hat{\mathbf{f}}_j = \hat{\mathbf{f}}_j^{(l)} - \left(\nabla^2_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)\right)^{-1}\nabla_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)^T . \tag{4.40}$$

Thus, the update rule for the Gauss-Newton minimization is

$$\hat{\mathbf{f}}_j^{(l+1)} = \hat{\mathbf{f}}_j^{(l)} - \left(\nabla^2_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)\right)^{-1}\nabla_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)^T , \tag{4.41}$$

where $\Delta = -\left(\nabla^2_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)\right)^{-1}\nabla_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)^T$.

For completeness, we also show how to compute the gradient $\mathbf{\Psi}_k^{(j)}$ of the measurement function $\psi_k\left(\hat{\mathbf{f}}_j\right)$ and the Hessian $\nabla^2_{\hat{\mathbf{f}}_j}\mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)$. Recall from Equation 4.25 that

$$\psi_k\left(\mathbf{f}_j\right) = \psi_k\left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right) \tag{4.42}$$

$$= \begin{pmatrix} \dfrac{\psi_{k1}\left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right)}{\psi_{k3}\left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right)} \\ \dfrac{\psi_{k2}\left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right)}{\psi_{k3}\left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right)} \end{pmatrix} . \tag{4.43}$$

Let

$$\psi_{k13}\left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right) = \frac{\psi_{k1}\left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right)}{\psi_{k3}\left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right)} \tag{4.44}$$

$$\psi_{k23}\left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right) = \frac{\psi_{k2}\left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right)}{\psi_{k3}\left(\alpha_{f_j}, \beta_{f_j}, \rho_{f_j}\right)} \ . \tag{4.45}$$

$$\tag{4.46}$$

Thus,

$$\mathbf{\Psi}_k^{(j)} = \begin{pmatrix} \frac{\partial \psi_{k13}}{\partial \alpha_{f_j}} & \frac{\partial \psi_{k13}}{\partial \beta_{f_j}} & \frac{\partial \psi_{k13}}{\partial \rho_{f_j}} \\ \frac{\partial \psi_{k23}}{\partial \alpha_{f_j}} & \frac{\partial \psi_{k23}}{\partial \beta_{f_j}} & \frac{\partial \psi_{k23}}{\partial \rho_{f_j}} \end{pmatrix} \ . \tag{4.47}$$

Computing the Hessian $\nabla^2_{\hat{\mathbf{f}}_j} \mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)$:

$$\nabla^2_{\hat{\mathbf{f}}_j} \mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right) = \nabla_{\hat{\mathbf{f}}_j}\left(\nabla_{\hat{\mathbf{f}}_j} \mathcal{J}\left(\hat{\mathbf{f}}_j^{(l)}\right)\right) \tag{4.48}$$

$$= \nabla_{\hat{\mathbf{f}}_j}\left(-\sum_{k \in \mathcal{S}_{f_j,t}} \left(\mathbf{o}_k^{(j)} - \psi_k\left(\hat{\mathbf{f}}_j\right)\right)^T \left(\mathbf{Q}_k^{(j)}\right)^{-1} \mathbf{\Psi}_k^{(j)}\right) \tag{4.49}$$

$$= -\sum_{k \in \mathcal{S}_{f_j,t}} -\left(\mathbf{\Psi}_k^{(j)}\right)^T \left(\mathbf{Q}_k^{(j)}\right)^{-1} \mathbf{\Psi}_k^{(j)} \tag{4.50}$$

$$= \sum_{k \in \mathcal{S}_{f_j,t}} \left(\mathbf{\Psi}_k^{(j)}\right)^T \left(\mathbf{Q}_k^{(j)}\right)^{-1} \mathbf{\Psi}_k^{(j)} \ . \tag{4.51}$$

To summarize, given an initial estimate $\hat{\mathbf{f}}_j^{(0)}$ of the inverse-depth parametrization of feature f_j, we repeatedly apply Equation 4.41 to obtain a new estimate for $\hat{\mathbf{f}}_j$ for a specified number of iterations or until convergence. Let $\hat{\mathbf{f}}_j^*$ be the final estimate of the Gauss-Newton

minimization. Its associated covariance $\mathbf{C}^{(j)}$ is given by the inverse of the Hessian, i.e.

$$\mathbf{C}^{(j)} = \left(\nabla^2_{\hat{\mathbf{f}}_j} \mathcal{J} \left(\hat{\mathbf{f}}^*_j \right) \right)^{-1} . \tag{4.52}$$

4.3.5 Summary of Particle Weighting

The following summarizes the steps needed to assign weights to the particles.

- For each particle $\mathbf{x}^{[i]}_{t-N_\mathrm{p}:t}, i = 1, 2, ..., N_\mathrm{s}$
  - For each feature f_j tracked at time t
    - Use $\mathcal{O}^{(j)}_t$, $\mathbf{x}^{[i]}_{t-N_\mathrm{p}:t}$, and the Gauss-Newton minimization to obtain an estimate $\hat{\mathbf{f}}^*_j$ of the inverse-depth parametrization of feature f_j and its associated covariance $\mathbf{C}^{(j)}$
    - Compute $w^{(j)[i]}_t$, the contribution of feature f_j to the weight of the ith particle, using Equation 4.12 with $q\left(\mathbf{f}_j\right) = \mathcal{N}\left(\hat{\mathbf{f}}_j; \hat{\mathbf{f}}^*_j, \mathbf{C}^{(j)}\right)$
  - Compute the overall weight $w^{[i]}_t$ using Equation 4.13

4.3.6 Unscented Transform

As discussed in Section 4.3.2, the contribution of feature f_j to the weight of the ith particle at time t is given by

$$w_t^{(j)[i]} = \int \frac{p\left(\mathcal{O}_t^{(j)} \middle| \mathbf{x}_{t-N_\mathrm{p}:t}^{[i]}, \mathbf{f}_j\right)}{q\left(\mathbf{f}_j\right)} q\left(\mathbf{f}_j\right) d\mathbf{f}_j \tag{4.53}$$

$$= \mathbb{E}_{q(\mathbf{f}_j)}\left[\frac{p\left(\mathcal{O}_t^{(j)} \middle| \mathbf{x}_{t-N_\mathrm{p}:t}^{[i]}, \mathbf{f}_j\right)}{q\left(\mathbf{f}_j\right)}\right] , \tag{4.54}$$

where we choose $q\left(\mathbf{f}_j\right) = \mathcal{N}\left(\hat{\mathbf{f}}_j; \hat{\mathbf{f}}_j^*, \mathbf{C}^{(j)}\right)$. Thus,

$$w_t^{(j)[i]} = \mathbb{E}_{\mathcal{N}\left(\hat{\mathbf{f}}_j; \hat{\mathbf{f}}_j^*, \mathbf{C}^{(j)}\right)}\left[\frac{p\left(\mathcal{O}_t^{(j)} \middle| \mathbf{x}_{t-N_\mathrm{p}:t}^{[i]}, \hat{\mathbf{f}}_j\right)}{\mathcal{N}\left(\hat{\mathbf{f}}_j; \hat{\mathbf{f}}_j^*, \mathbf{C}^{(j)}\right)}\right] \tag{4.55}$$

$$= \int \frac{p\left(\mathcal{O}_t^{(j)} \middle| \mathbf{x}_{t-N_\mathrm{p}:t}^{[i]}, \hat{\mathbf{f}}_j\right)}{\mathcal{N}\left(\hat{\mathbf{f}}_j; \hat{\mathbf{f}}_j^*, \mathbf{C}^{(j)}\right)} \mathcal{N}\left(\hat{\mathbf{f}}_j; \hat{\mathbf{f}}_j^*, \mathbf{C}^{(j)}\right) d\mathbf{f}_j \tag{4.56}$$

$$= \int \mathcal{F}\left(\hat{\mathbf{f}}_j\right) \mathcal{N}\left(\hat{\mathbf{f}}_j; \hat{\mathbf{f}}_j^*, \mathbf{C}^{(j)}\right) d\mathbf{f}_j , \tag{4.57}$$

where $\mathcal{F}\left(\hat{\mathbf{f}}_j\right) = \dfrac{p\left(\mathcal{O}_t^{(j)} \middle| \mathbf{x}_{t-N_\mathrm{p}:t}^{[i]}, \hat{\mathbf{f}}_j\right)}{\mathcal{N}\left(\hat{\mathbf{f}}_j; \hat{\mathbf{f}}_j^*, \mathbf{C}^{(j)}\right)}$. One way to approximate the integral in Equation 4.57 is the unscented transform [Julier and Uhlmann, 1997, Ito and Xiong, 2000]. In general, we are faced with the problem of computing the integral of the following form.

$$\mathbb{E}_{\mathcal{N}(\mathbf{m};\mu,\mathbf{P})}[\mathcal{F}\left(\mathbf{m}\right)] = \int \mathcal{F}\left(\mathbf{m}\right) \mathcal{N}\left(\mathbf{m}; \mu, \mathbf{P}\right) , \tag{4.58}$$

where $\mathbf{m}$ is an n-dimensional random variable. The technique of unscented transform to approximate the above integral is to deterministically choose $2n+1$ points

$$\bar{\mathbf{m}}_0 = \mu \tag{4.59}$$

$$\bar{\mathbf{m}}_i = \mu + \left(\sqrt{(n+\kappa)\,\mathbf{P}}\right)_i, \quad 1 \leq i \leq n \tag{4.60}$$

$$\bar{\mathbf{m}}_{i+n} = \mu - \left(\sqrt{(n+\kappa)\,\mathbf{P}}\right)_i, \quad 1 \leq i \leq n\ , \tag{4.61}$$

with corresponding weights

$$w_0 = \frac{\kappa}{n+\kappa} \tag{4.62}$$

$$w_i = \frac{1}{2\,(n+\kappa)}, \quad 1 \leq i \leq n \tag{4.63}$$

$$w_{i+n} = \frac{1}{2\,(n+\kappa)}, \quad 1 \leq i \leq n\ , \tag{4.64}$$

where $\kappa \in \Re$ and $\left(\sqrt{(n+\kappa)\,\mathbf{P}}\right)_i$ is the ith row or column of the matrix square root (Cholesky decomposition) of $(n+\kappa)\,\mathbf{P}$. Then, Equation 4.58 can be approximated as

$$\mathbb{E}_{\mathcal{N}(\mathbf{m};\mu,\mathbf{P})}[\mathcal{F}\,(\mathbf{m})] \approx \sum_{i=0}^{2n} w_i \mathcal{F}\,(\bar{\mathbf{m}}_i)\ . \tag{4.65}$$

4.4 Experimental Results

We implemented the particle filtering approach described in the previous section and tested it extensively with both simulation as well as real-world data. In this section, we present some of the experimental results.

4.4.1 Simulation Results

The goal of the simulation experiments is to test the performance of the particle filtering approach under different conditions by varying certain parameters.

Figure 4.1 shows our simulation environment which is a $12\text{m} \times 12\text{m}$ room with two hundred visual point features randomly placed on the walls. The height of the features ranges from 0.5m to 5m. The robot is moving along a circular path of radius 3m with a constant velocity and angular velocity of 0.1m/s and 0.0333rad/s, respectively.[3] The measured velocity and angular velocity are corrupted by independent zero-mean Gaussian noise with standard deviations 0.01m/s and $1^\circ/\text{s}$, respectively. As the robot moves, its camera records images at 1Hz. We assume that the camera has a field of view of 47.5° and $\sigma_{\text{im}} = 1/400$. We set the total number of time steps for the simulation to be 1000. For the particle filter, we set $N_\tau = N_\text{s}/3$.

For comparison, we computed the average root mean squared error (RMSE) over all 100 Monte Carlo trials and over all time steps of both the EKF and particle filter. We exper-

[3]For the simulation experiments, we used the constant velocity and constant angular velocity motion model instead of the odometry motion model discussed in Section 4.3.1.

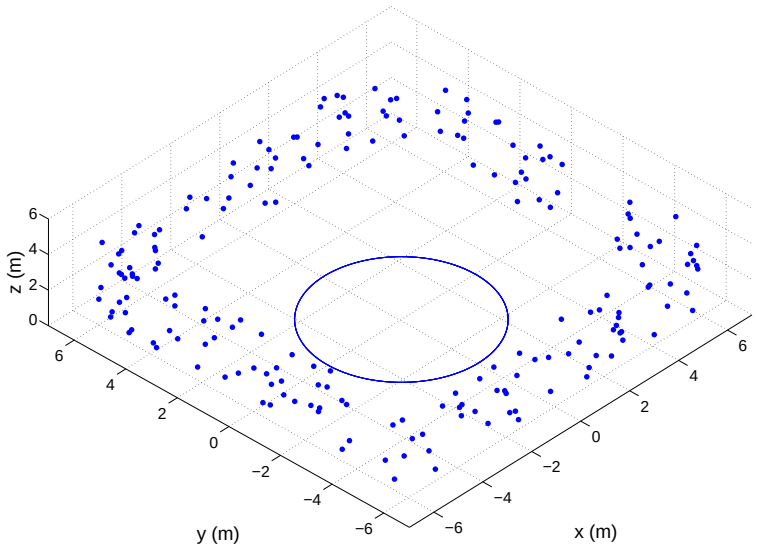


Figure 4.1: The simulation environment and the true trajectory of the robot (circle). Each dot represents a point landmark placed on the wall of the environment.

imented with different values for N_{p} (the maximum number of robot or camera poses for processing feature measurements) and different number of particles N_{s} for the particle filter.

From Table 4.1, it can be seen that the particle filter performs better than the MSCKF for the cases considered. Moreover, as the value of N_{p} decreases, the performance of the particle filter exhibits a more graceful degradation than that of the MSCKF. Notice also that increasing the number of particles in the particle filter does not significantly affect its performance indicating that using only 1000 particles is sufficient for this simulation scenario.

Table 4.1: The mean RMSE of the MSCKF and particle filter over all 100 Monte Carlo trials and over all time steps.

	MSCKF	Particle Filter			
		N_{s}			
N_{p}		1000	5000	10000	
10	0.2449	0.2312	0.2198	0.2297	x
	0.2288	0.2024	0.1951	0.1925	y
	0.0707	0.0248	0.0190	0.0210	θ
5	0.3149	0.2623	0.2446	0.2363	x
	0.3169	0.2526	0.2496	0.2478	y
	0.0962	0.0357	0.0331	0.0358	θ
3	0.4622	0.3365	0.3092	0.3275	x
	0.4678	0.3309	0.3155	0.3218	y
	0.1536	0.0583	0.0627	0.0634	θ
2	0.6795	0.4221	0.4147	0.4103	x
	0.6599	0.4564	0.4284	0.4183	y
	0.2077	0.1000	0.0984	0.0828	θ

4.4.2 Real-World Results

To demonstrate the effectiveness and applicability of the particle filter approach in a real-world setting, we apply the approach on real image sequences collected with our mobile robot. In our experiments, we used an ActivMedia Robotics P3-DX that is equipped with wheel encoders for its odometry and a front-looking Canon VC-C50i camera that provides images of resolution 320×240 pixels. We pre-calibrated the camera using the camera calibration toolbox for Matlab by Bouguet [2008]. For the real-world experiments, we stored all data on an IBM ThinkPad X32 notebook computer that is used to control the robot. Data processing was done off-line. We used the Kanade-Lucas-Tomasi (KLT) tracker [Shi and Tomasi, 1994, Birchfield, 2007] for extracting and tracking fifteen features per image frame.

Table 4.2: Parameter settings for the real-world experiments.

$\sigma_{\text{trans}} = 0.002\text{m}$	$\sigma_{\text{rot}} = 0.5^\circ$	$\delta_{\text{trans}} = 0.001$	$\delta_{\text{rot}} = 0.001$
$\sigma_{\text{im}} = \frac{1}{400}$	$N_{\text{max}} = 5$	$N_\tau = 0.6N_{\text{s}}$	$N_{\text{s}} = 1000$

For the particle filter, we used 1000 particles. Table 4.2 shows the parameter settings we used for our real-world experiments.

The first real-world experiment was conducted inside our laboratory where the robot was driven along a rectangular path of size approximately $3\text{m} \times 7\text{m}$. While traversing the path, the robot collected images at the rate of 4Hz. This led to a total of 3346 images of which sample images are shown in Figure 4.2. If there is not enough motion between two consecutive image frames, computing the feature estimate is an ill-conditioned problem. To avoid this, we automatically selected images from the original sequence of images in which the encoders indicate that the robot translated for at least 0.1m or rotated for at least 1°. From a total of 3346 images from the original sequence of images, 194 images were selected. However, the KLT tracker was ran on the full image sequence.

Figure 4.3 shows the estimated trajectory according to the odometry (broken line) vs. the estimated trajectory according to the particle filter (i.e. the mean particle trajectory) (thick line) vs. the estimated trajectory according to the MSCKF (thin line). The robot started and ended at position $(0, 0)$. Notice that the robot's odometry suffers from drift; its estimate of its pose quickly and significantly deviates from the true robot pose. Here, we can clearly see that using the observations of visual features and the particle filter (or MSCKF) approach can



Figure 4.2: Some sample images taken from the first real-world experiment.

compensate for odometric errors and close the loop (which is generally considered a difficult problem in aided robotic navigation).

We also tested the MSCKF and particle filter approaches in a much larger real-world environment. The second test environment is the rectangular hallway environment of the South wing of the fourth floor of our Computer Science building. The rectangular hallway has an approximate size of $30\text{m} \times 25\text{m}$. Aside from being much larger than the first test environment, the second test environment is more challenging from a vision standpoint as the lighting condition is non-uniform across the environment as can be seen from the sample images taken from this environment shown in Figure 4.4. The entire image sequence for the second test environment consists of a total of 4857 images and image selection resulted in

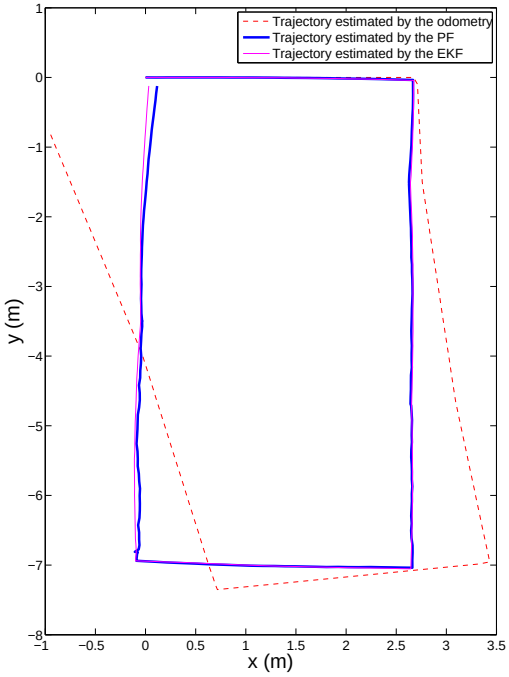


Figure 4.3: The estimated trajectory according to the odometry (broken line), particle filter (thick line), and MSCKF (thin line) for the first test environment. The robot started and ended at position $(0, 0)$.



Figure 4.4: Some sample images taken from the second real-world experiment.

1257 images. As before, we ran the KLT tracker on the original image sequence and we post processed the output of the KLT tracker to determine the feature tracks for the selected images.

Figure 4.5 shows the estimated trajectory based on the odometry (broken line), particle filter (thick line), and MSCKF (thin line). The robot started and ended at position $(0, 0)$. As can be seen from the figure, only the particle filter is capable of closing the loop. This is remarkable as loop closing in environments consisting of long hallways (such as the second test environment) poses a great challenge to filtering algorithms since an incorrect filtered estimate during any part of the journey would cause drifting and prevent the loop from being properly closed. We should also point out that the approach uses no prior map of the envi-

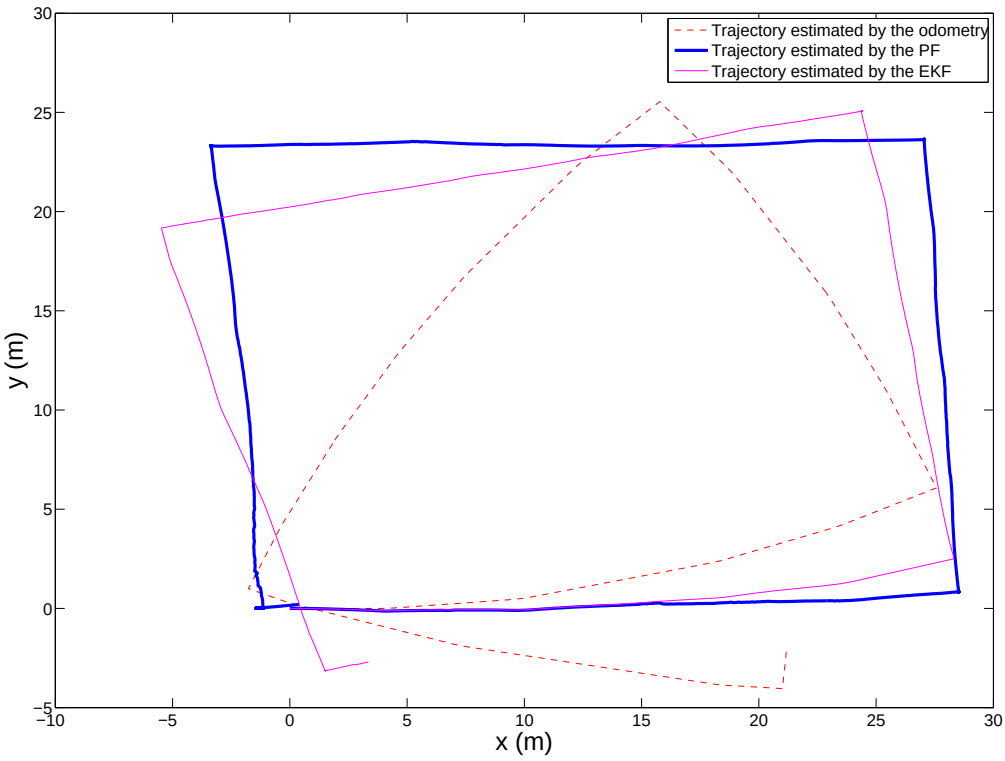


Figure 4.5: The estimated trajectory according to the odometry (broken line), particle filter (thick line), and MSCKF (thin line) for the second test environment. The robot started and ended at position $(0, 0)$.

ronment and that the robot or camera motion is mostly parallel to the camera's optical axis during the experiment which is a difficult scenario for vision-based motion estimation.

4.5 Discussion and Summary

In this chapter, we have presented a particle filtering approach for monocular vision-aided odometry for the real-time localization of a mobile robot navigating in indoor environments. Monocular vision is used for detecting naturally occurring static three-dimensional point

features or landmarks from the environment and the information is utilized for correcting the pose as suggested by the odometry. In order to process feature measurements, several robot or camera poses are maintained by each particle in the particle filter. We have introduced a sensor model for assigning weights to the particles based on their trajectory and the measurements of features observed along the trajectory. The approach has been validated using both simulation as well as real-world data and compared against that of the MSCKF.

While the results of the tests show the effectiveness of the particle filter approach for monocular vision-aided odometry, there are several open issues that have to be addressed:

- It is interesting to investigate the performance of the approach in the case of dynamic environments. We should point out that the real-world experiments we have conducted took place in nearly static environments.

- Since the success of the approach highly depends on the performance of the visual tracker employed (in our case, we used the KLT tracker), it is worthwhile to study the robustness of the approach against wrong data associations (i.e. mistakingly treating different features in two consecutive images to be the same feature) produced by the tracker.

- Since it is impossible to have a perfect visual tracker and we cannot always assume that the robot's operating environment is static, it is necessary to incorporate outlier detection and rejection into the approach. We believe that outlier detection and rejection can be useful for dealing with the problems mentioned above.

Chapter 5

Simultaneous Localization and Mapping

5.1 Introduction

Since it was introduced by Smith and Cheeseman [1986] and Smith et al. [1990], the simultaneous localization and mapping (SLAM) problem has become one of the mainstream research areas in mobile robotics. SLAM involves estimating the position and orientation of the robot while building a map of the environment in parallel. In order to accomplish SLAM, the robot is usually equipped with sensors (e.g. wheel encoders, range sensors, cameras) that allow it to observe (though only partially and inaccurately) the state of the world including itself. The SLAM problem is inherently difficult and complex although the specific sensor used also contributes to the hardness of SLAM.

SLAM is a well-studied problem in mobile robotics and a number of techniques have been proposed. However, the majority of the proposed techniques for SLAM rely on the

use of accurate and dense measurements provided by laser rangefinders to correctly localize the robot and produce accurate and detailed maps of complex environments (see e.g. Montemerlo et al. [2002], Eliazar and Parr [2003], Montemerlo and Thrun [2003], Hähnel et al. [2003], Eliazar and Parr [2004a], Brunskill and Roy [2005], Grisetti et al. [2005], Garulli et al. [2005], Nguyen et al. [2006], Eliazar and Parr [2006]). Relatively little work has been done on the use of low-cost but noisy and sparse sonar sensors for SLAM in large indoor environments involving large loops. Obviously, the SLAM problem is much more difficult and challenging in the case of sonar sensors than laser rangefinders (which have become the *de facto* range sensors for SLAM). Despite the difficulty, we believe that it is interesting to investigate the extent to which sonar sensors can be used for SLAM especially in mapping large indoor environments with large loops.

Although sonar sensors are not as accurate and do not provide as dense observations as laser rangefinders, they are still an attractive alternative to laser rangefinders when it comes to cost, power consumption, size and weight, and computational requirements. Compared to laser rangefinders, sonars cost several orders of magnitude less, consume less power, are small and lightweight, and impose minimal computational requirements. As such, sonar sensors are well suited for use in inexpensive consumer-oriented and minimally-equipped robots that are typically limited in both power and computational capability.

In this chapter, we present our approach to SLAM with sonar sensors by applying particle filtering and an orthogonal line-segment-based map representation [Nguyen et al., 2006] to map indoor environments much larger and more challenging than those previously con-

sidered with sonar sensors. Rather than employing a grid-based representation [Elfes, 1987, Moravec, 1988, Elfes, 1989] for the map, we use a feature-based representation where the features are represented as line segments. Line segments are suitable for compactly describing most structured indoor environments that are usually composed of walls, doors, glass windows, etc. that are either parallel or perpendicular to each other. Similar to Nguyen et al. [2006], we also make use of the *orthogonality* assumption about the shape of the environment in order to reduce the complexity, mapping only lines that are either parallel or perpendicular to each other.

A major difficulty in using line segments as an environment's representation is extracting them from noisy and sparse sensors such as sonars (in our case, an array of sixteen sonar transducers) compared to a single dense scan of a 180° laser rangefinder. Thus, in this work, we adopt the *multiscan* approach [Beevers and Huang, 2006] to group consecutive sparse scans so that measurements from multiple time frames can be used to extract line segments, although our work differs on how the sparse scans are collected and the frequency at which the feature extraction is performed.

The contributions of our work can be summarized as follows. We show through extensive experiments that it is possible to produce good-quality maps of large indoor environments with large loops even with noisy and sparse sonar sensors. To our knowledge, the environments we consider in our experiments are much larger and more challenging than those previously reported in the literature for SLAM with sonars. The results provide significant supportive evidence for the potential viability of sonars for large-scale indoor SLAM. Our

method employs a particle filtering technique where each particle carries a single map rather than a distribution over possible maps. Moreover, we apply the orthogonality assumption to reduce the complexity. Finally, we discuss how we extract line segments from sonar data and we introduce a simple sensor model for computing the likelihood of the observations.

5.2 Related Work

While much of the work on SLAM with proximity sensors has focused on laser scanners, some researchers have used sonars. In this section, we briefly describe some of the techniques proposed for carrying out SLAM with sonars. There are two main criteria that can be used to categorize existing sonar-based SLAM techniques: the representation used to model the environment and the technique used to estimate the state belief distribution.

Zunino and Christensen [2001] described an algorithm for SLAM based on the extended Kalman filter (EKF). The EKF approach was used to build and maintain the map of the environment. Their method used point features as landmarks (representing corners, edges, and thin poles detectable by standard sonar sensors through a triangulation technique). They also presented a method for detecting the failures of the EKF approach and recovering from such failures. Experiments were performed using a Nomadic SuperScout mobile robot in a room of size $5\text{m} \times 9\text{m}$.

Tardós et al. [2002] described a technique for the creation of feature-based stochastic maps using standard Polaroid sonar sensors. In their work, they used the Hough transform

[Duda and Hart, 1972] for detecting point features (representing corners and edges) and line segment features (representing walls) from sonar data acquired from multiple uncertain vantage points. Instead of building one global map from the start, they generated a sequence of local maps of limited size, and then joined them together, to obtain the global map. Techniques for joining and combining several stochastic maps were presented. The locations of geometric features in the environment and the position of the robot were jointly estimated in a stochastic framework via the EKF. Experiments were carried out using a B21 mobile robot equipped with a ring of twenty-four Polaroid sensors in a 12m $\times$ 12m environment.

A similar approach to that of Tardós et al. [2002] was proposed by Leonard et al. [2002] except that they incorporated past robot positions to the state vector and explicitly maintained the estimates of the correlations between current and previous robot states. By doing so, it became possible to consistently initialize new map features by combining data from multiple vantage points. Experiments were conducted in a testing tank of size 10m $\times$ 3m $\times$ 1m for underwater sonar-based SLAM, a simple "box" environment made of plywood, and along a 25m long corridor.

A much earlier work along the same lines of feature-based stochastic mapping using the EKF was given by Rencken [1993] although experiments were only performed in a simulation of a 5m $\times$ 5m room.

Instead of jointly estimating the robot state and the locations of line segment features via the EKF, Lorenzo et al. [2004] used the EKF to estimate only the robot state. The environment was described by a global segment-based map that was built using local sonar-based

occupation maps to identify obstacle boundaries. The Hough transform was used to extract the segments that represent the obstacle boundaries. The segments were then incorporated to the global segment-based map. The Hough-based approach provided a correction of the estimated robot pose which was integrated with odometric information via the EKF. The approach was tested with a Nomad 200 mobile robot equipped with a ring of sixteen sonar sensors traversing a corridor environment.

More recently, Schröter et al. [2007] presented a combination of map-matching with a Rao-Blackwellized particle filter (RBPF) [Murphy, 2000] which enabled them to solve the SLAM problem with low-resolution sonar range sensors. They introduced a simple and fast but very efficient shared representation of grid maps which reduced the memory cost overhead caused by inherent redundancy between the particles. Experimental results were presented with a SCITOS A5 robot platform navigating in a home store environment.

Other notable recent related work on SLAM with sparse sensors include the work of Beevers and Huang [2006] on SLAM with sparse sensing using the multiscan approach and RBPF, Abrate et al. [2007] on experimental EKF-based SLAM for mini-rovers with IR sensors only, and Choi et al. [2008] on a line-based SLAM with infrared sensors using geometric constraints and active exploration.

The focus of our work reported in this chapter is on SLAM in large structured indoor environments involving large loops using sonar sensors. The largest environment we consider has an approximate area of $70.0\text{m} \times 53.7\text{m}$ containing two major loops and it is made up of various types of obstacles (e.g. brick walls, tiled walls, glass windows and doors, and cable

railings). Our work differs from Schröter et al. [2007] in that we use a line-segment-based instead of a grid-based representation for the map. By adopting a line-segment-based map representation, we can avoid the usual problems associated with a grid-based representation such as data smearing [Tardós et al., 2002], the strong independence assumption between the grid cells, and the considerable amount of memory required for storage. However, a major difficulty with line segments is extracting them from sparse and noisy sonars. To overcome the sparseness, we apply the multiscan approach [Beevers and Huang, 2006] to group consecutive sparse scans. In order to reduce the complexity of SLAM, similar to Nguyen et al. [2006], we make use of the orthogonality assumption about the shape of the environment by mapping only lines that are parallel or perpendicular to each other. Unlike Rencken [1993], Tardós et al. [2002], Leonard et al. [2002], Lorenzo et al. [2004], we use particle filters to sample the distribution over the most recent robot poses. Each particle in our particle filter is associated with a single map rather than a distribution over possible maps.

5.3 Localization Approach

The basis of our approach is a particle filter, where each particle is an estimate of the recent N_{p} robot poses. Thus, the particles collectively sample the space of the recent N_{p} robot poses. Here, as in previous chapters, we assume that the robot moves in a planar environment so that its pose at time t can be represented as a column vector $\mathbf{x}_t = (x_t, y_t, \theta_t)^T$, where (x_t, y_t) is its two-dimensional Cartesian coordinates and θ_t is its heading or orientation with respect to

some global reference frame G. Each particle carries an estimated map of the environment that is represented as a set of line segments. Because of the orthogonality assumption about the shape of the environment, the map will contain only two types of lines: horizontal and vertical. Horizontal (vertical) lines are assigned to be parallel to the x-axis (y-axis) of the global reference frame G. Extracted line segments that are not close to being either horizontal or vertical are simply discarded and not placed in the map.

In the next two subsections, we discuss the motion and sensor models we use for our particle filter.

5.3.1 Motion Model

Let $\mathbf{x}_{t-N_\mathrm{p}+1:t} \triangleq \left(\mathbf{x}_{t-N_\mathrm{p}+1}, \mathbf{x}_{t-N_\mathrm{p}+2}, ..., \mathbf{x}_t\right)$ be the sequence of the recent N_p robot poses at time t. We denote by $\mathbf{x}^{[i]}_{t-N_\mathrm{p}+1:t} \triangleq \left(\mathbf{x}^{[i]}_{t-N_\mathrm{p}+1}, \mathbf{x}^{[i]}_{t-N_\mathrm{p}+2}, ..., \mathbf{x}^{[i]}_t\right)$ the trajectory estimate of the ith particle. $\mathcal{M}^{[i]}_t$ is the set of line segments representing the map of ith particle at time t. The particles move according to the probabilistic motion model $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{c}_t)$, where $\mathbf{c}_t$ is the control command executed by the robot during the time interval $[t-1, t)$, to account for the inherent uncertainty in robot motion. The control command $\mathbf{c}_t$ is given by the pair $\left(\hat{d}_t, \hat{r}_t\right)$, where $\hat{d}_t$ is the distance traveled and $\hat{r}_t$ is the rotation made by the robot, according to the wheel encoders. Due to the probabilistic nature of the motion model, the particles spread and generate different possible robot trajectories. Here we use the motion model in Roy and

Thrun [1999] and we repeat it here for convenience:

$$x_t = x_{t-1} + d_t \cos(\theta_{t-1} + r_t) \tag{5.1}$$

$$y_t = y_{t-1} + d_t \sin(\theta_{t-1} + r_t) \tag{5.2}$$

$$\theta_t = (\theta_{t-1} + r_t) \mod 2\pi, \tag{5.3}$$

where d_t and r_t denote the robot's true translation and rotation, respectively. The true translation and rotation both differ from the translation and rotation measured by the robot due to systematic and random errors. Specifically,

$$d_t = \hat{d}_t + \delta_{\text{trans}} \left|\hat{d}_t\right| + \epsilon_{\text{trans}} \tag{5.4}$$

$$r_t = \hat{r}_t + \delta_{\text{rot}} \left|\hat{d}_t\right| + \epsilon_{\text{rot}} \tag{5.5}$$

where δ_{trans} and δ_{rot} describe the systematic errors and $\epsilon_{\text{trans}} \sim \mathcal{N}(0, \sigma^2_{\text{trans}})$ and $\epsilon_{\text{rot}} \sim \mathcal{N}(0, \sigma^2_{\text{rot}})$ are the additive random variables representing the random noise. In our experiments, $\delta_{\text{trans}} = 1.0 \times 10^{-2}$, $\delta_{\text{rot}} = 1.0 \times 10^{-5}$, $\sigma_{\text{trans}} = 2$mm, and $\sigma_{\text{rot}} = 2^\circ$.

5.3.2 Sensor Model

For SLAM, the weight assigned to the ith particle is directly proportional to the likelihood of the observations $p\left(\mathbf{s}_t \left| \mathbf{x}_t^{[i]}, \mathcal{M}_t^{[i]} \right.\right)$ where $\mathbf{s}_t = \left\{s_t^{(1)}, s_t^{(2)}, ..., s_t^{(K)}\right\}$ is the set of sensor measurements taken by the K available sensors at time t. From the preceding discussion, it

is evident that using an accurate and robust sensor model is therefore crucial to the success of the particle filter and other state estimation techniques in general. However, deriving an accurate and robust model for sensors is generally a difficult problem.

In this chapter, we propose a heuristic but simple way to calculate the weights of the particles which is inspired by the work of Schröter et al. [2007] on map matching. In [Schröter et al., 2007], they need to match a local map to a global one. They calculate the weight of ith particle as

$$w_t^{[i]} \propto e^{\frac{m_t^{[i]}}{f}} \tag{5.6}$$

where f is a free parameter that influences the spread of the particle weights and $m_t^{[i]}$ is a measure of the quality of the match. Schröter et al. [2007] employs a grid-based map, so we replace their definition of $m_t^{[i]}$ with our own. For each sonar range measurement $s_t^{(k)}$ obtained by sensor k at time t, we perform ray tracing along the facing direction of sensor k twice: a) using the current map $\mathcal{M}_t^{[i]}$ and b) using the current map but with the line segments extended by a certain length of L_{ext} (e.g. 200mm) at both ends.[1] Each ray tracing operation for sensor k at time t, calculates the "true" (expected) range measurement[2] $s_t^{(k)*}$ for sensor k at time t. If the absolute difference between the actual and expected range measurements is less than or equal to the standard deviation of the sonar measurements σ_{d} (50mm) (i.e.

[1] We exclude those measurements $s_t^{(k)}$ that are equal to the maximum sonar range $s_{\max}$ (e.g. 5000mm) since they provide us with little information as to the location of objects or obstacles.

[2] The "true" (expected) range measurement $s_t^{(k)*}$ is intended to be the distance to the nearest obstacle in the map along the facing direction of the sensor. If the ray doesn't intersect with any obstacle in the map, $s_t^{(k)*} = +\infty$.

$\left|s_t^{(k)} - s_t^{(k)*}\right| \leq \sigma_{\mathrm{d}}$), we count it as $+1$. Otherwise, we count it as -1. We then take the average of the counts obtained from the two ray tracing operations for sensor k and use that as the sensor's contribution. We sum these contributions to obtain $m_t^{[i]}$. Figure 5.1 illustrates how the match value is computed. In this way, a particle whose map can explain the current measurements well will be assigned a higher weight than one that cannot. We should point out that the above procedure to compute the match value is just a simple heuristic that is found to work well in our experiments.

5.3.3 Particle Filtering

For particle filtering, we use Algorithm 5 (see Section 2.4) but we compute the particle weights using Equation 5.6 and we perform resampling only if $\hat{N}_{\mathrm{eff}} < N_\tau$ where $N_\tau = N_{\mathrm{s}}/2$. We then extract line segment features from the recent N_{p} sonar scans $\mathbf{s}_{t-N_{\mathrm{p}}+1:t}$ for each particle $\mathbf{x}_{t-N_{\mathrm{p}}+1:t}^{[i]}$ and incorporate the extracted line segments into the map $\mathcal{M}_t^{[i]}$.

5.4 Map Building Approach

In this section, we discuss the technique we use to extract line segments from multiple sonar scans and how the lines are added into the maps of the particles as well as how the maps are maintained.

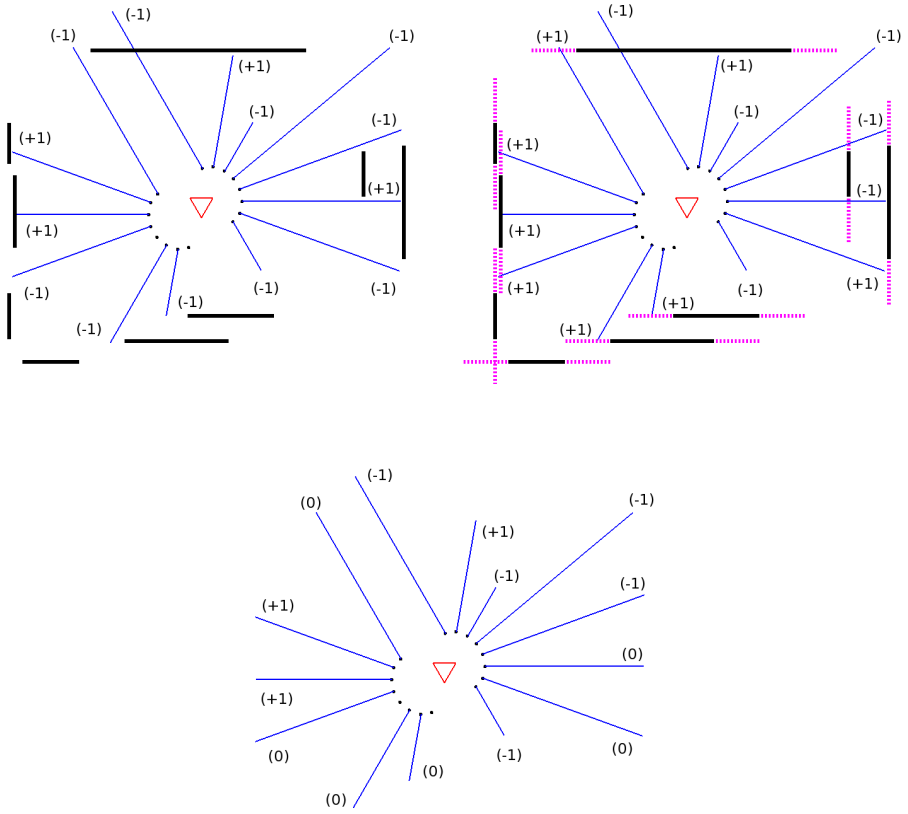


Figure 5.1: Computation of the match value. (Top Left) Result of the first ray tracing operation using the estimated line segments. Count values are shown in parenthesis beside their corresponding sonar measurements. Missing sonar measurements are equal to $s_{\max}$ and are not considered in computing the match value. (Top Right) Result of the second ray tracing operation using the extended line segments. (Bottom) Average count value for each sonar measurement. The match value $m_t = -2$ for this example. Figure taken from [Yap and Shelton, 2009] © [2009] IEEE.

5.4.1 Line Segment Extraction

For extracting line segments from the group of recent N_{p} sonar scans, we use the randomized Hough transform (RHT) technique proposed by Kultanen et al. [1990]. Before line segments can be extracted, we first plot each sonar measurement as a point (in the global reference frame G) that represents the nominal position of the sonar return, computed along the central axis (facing direction) of the transducer, with respect to the robot pose where the measurement was taken. Let $\mathcal{P}$ be the set of such points. We then use the RHT to find groups of (almost) collinear points in $\mathcal{P}$ and extract the parameters of the lines that fit those groups of points.

The Hough transform [Duda and Hart, 1972] is a technique used in digital image processing for extracting features such as lines and curves in binary edge images. It is a voting scheme where each point (pixel) in the image votes for a set of features (lines, curves) that pass through it. Voting is performed in a discretized parametric space, called the *Hough space*, representing all possible feature locations. The most voted cells in the Hough space should correspond to the features actually present in the image.

The RHT is an improvement on the original Hough transform by reducing the computation time and memory usage. The basis of the method lies on the fact that a single parameter space point can be determined uniquely with a pair, triple, or generally n-tuple of points from the image. Such an n-tuple of points can be chosen randomly from the image and hence the name. Unlike in the original Hough transform where each point in the image votes for a set of cells in the Hough space, a group of n randomly chosen points from the image vote for

only one parameter space point in the RHT. The presence of a specific feature in the image is quickly revealed by the accumulation of a small number of parameter space points. Rather than having a fixed-size array structure for implementing the Hough space (also called the *accumulator space*) as in the original Hough transform (thus imposing some predefined accuracy in parameter point location), in RHT, it is possible to use a dynamic tree structure to store the accumulator cells with non-zero votes in the parameter space. This way, one can achieve as high an accuracy as required while at the same time bringing memory usage to near minimal.

To use the RHT for line segment extraction, we use the polar coordinate parametrization (ρ, θ) for lines, where ρ is the length of the normal from the origin to the line and θ is the angle that the normal makes with the positive x-axis. Using this parametrization, the equation of the line can be written as

$$\rho = x \cdot \cos\theta + y \cdot \sin\theta \ . \tag{5.7}$$

To ensure unique parametrization of lines, we impose that $0 \leq \rho \leq \rho_{\text{max}}$ and $-180^\circ \leq \theta < 180^\circ$.

In our implementation, we discretize the Hough space using the resolutions $\Delta\rho = 50$mm and $\Delta\theta = 1^\circ$. Let $\mathcal{A}$ be the fixed-size accumulator array for implementing the discretized Hough space. A point (ρ, θ) in the Hough space corresponds to the accumulator cell $\mathcal{A}_{ij}$ with $i = \lfloor \rho / \Delta\rho \rfloor$ and $j = \lfloor (\theta - \theta_{\text{min}}) / \Delta\theta \rfloor$. Algorithm 7 outlines the RHT to extract line segments from the set $\mathcal{P}$.

Algorithm 7 Randomized Hough Transform

Input:
$\mathcal{P}$: set of points
N_0: minimum number of points
N_{trials}: maximum number of trials
τ: a threshold (e.g. 100)
Output:
Extracted line segments
Process:
while $\mathcal{P}$ contains at least N_0 points and the maximum number of trials has not been reached
    Reset the accumulator cells $\mathcal{A}_{ij}$ to 0
    while the accumulator array $\mathcal{A}$ does not have a global maximum that exceeds τ
        Pick two points p_a and p_b randomly from $\mathcal{P}$
        Solve the line parameters (ρ, θ) from the line equation with points p_a and p_b
        Increment the accumulator cell $\mathcal{A}_{ij}$ corresponding to (ρ, θ) by 1
    end
    Let $\left(\hat{\rho}, \hat{\theta}\right)$ be the line determined by the location of the maximum in $\mathcal{A}$
    Let $\mathcal{Q}$ be the set of points in $\mathcal{P}$ that are close to the line $\left(\hat{\rho}, \hat{\theta}\right)$
    If $\mathcal{Q}$ contains at least N_0 points, use $\mathcal{Q}$ and $\left(\hat{\rho}, \hat{\theta}\right)$ to extract line segments ***
    Remove from $\mathcal{P}$ points in $\mathcal{Q}$ used to generate segments
end

Prior to extracting line segments in the step marked with *** in Algorithm 7, we apply the orthogonality assumption and proceed with the line segment extraction only if the line $\left(\hat{\rho}, \hat{\theta}\right)$ is close to being horizontal or vertical. This is done by testing whether $\hat{\theta}$ is within a certain threshold $\epsilon_\theta = 5^\circ$ from 90° or -90° for horizontal line and -180°, 0°, or 180° for vertical line. If so, we then set $\hat{\theta}$ to be one of $\{-180^\circ, -90^\circ, 0^\circ, 90^\circ\}$ accordingly and update $\hat{\rho}$ so as to best fit the points in $\mathcal{Q}$ in the least-square sense. We denote the line resulting from applying the orthogonality assumption as $\left(\hat{\rho}', \hat{\theta}'\right)$.

Given $\left(\hat{\rho}', \hat{\theta}'\right)$, we project the points in $\mathcal{Q}$ onto the line $\left(\hat{\rho}', \hat{\theta}'\right)$. Let $\mathcal{R} = \{p'_1, p'_2, ...\}$ be the set of projected points arranged sequentially starting from one of the extreme endpoints. We then sequentially partition $\mathcal{R}$ into subsets $\mathcal{S}_h, h = 1, 2, ...$, with each $\mathcal{S}_h$ representing the set of points belonging to a line segment. We break the line into segments if there is a gap greater than a threshold $L_{\text{sep}} = 500$mm. After partitioning $\mathcal{R}$ into subsets $\mathcal{S}_h, h = 1, 2, ...$, the line segments are easily obtained as those that connect the extreme endpoints in each $\mathcal{S}_h$. To increase the reliability of the line segment extraction phase, only those line segments that are made up of at least $N_0 = 8$ points and with length at least $L_{\text{min}} = 200$mm are incorporated into the map.

5.4.2 Line Merging and Map Management

When incorporating a line segment into the map, we first test whether it can be merged with the line segments of the same type already in the map. Two line segments can be merged if

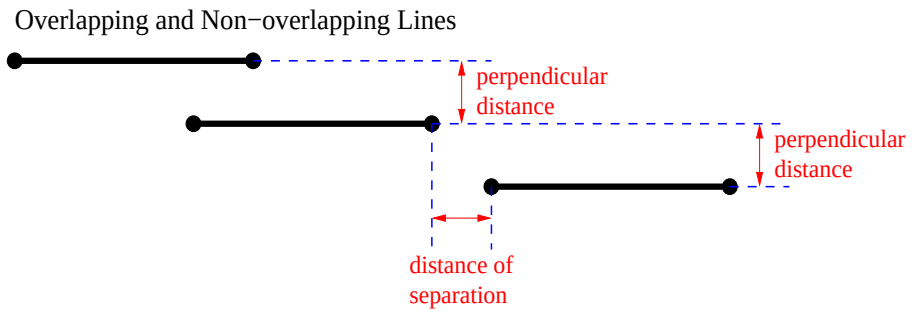


Figure 5.2: The perpendicular distance and distance of separation between overlapping and non-overlapping horizontal lines. Figure taken from [Yap and Shelton, 2009] © [2009] IEEE.

- they overlap and the perpendicular distance between them is less than or equal to L_{dist} (300mm); or

- they do not overlap but the perpendicular distance between them is less than or equal to L_{dist} and their distance of separation is less than or equal to L_{sep}.

Figure 5.2 illustrates the perpendicular distance and distance of separation between overlapping and non-overlapping horizontal lines. A similar interpretation exists for vertical lines.

To merge two line segments, we first compute the position (i.e. the ρ-value) of the resulting line segment. The position of the resulting line segment is the sum of the positions of the given line segments, weighted by their respective number of points. We then project the endpoints of the given line segments onto the resulting line and the two projections that are farthest apart define the endpoints of the resulting line segment. The number of points of the resulting line segment is just the sum of the number of points of the given line segments. If a line segment cannot be merged with the line segments already in the map, it is simply added to the map.

After incorporating the line segments into the map, we continue to merge segments that meet our criteria until no further mergings are possible. This step can have the desirable effect of integrating two distinct line segments representing the same environmental feature into one (e.g. a long stretch of wall occasionally occluded by dynamic obstacles or relatively small and insignificant objects).

5.5 Rao-Blackwellized Particle Filter

SLAM integrates the problems of estimating the robot's pose and landmark positions in the environment. When viewed as a probabilistic state estimation problem, SLAM involves estimating the joint posterior probability over the robot's pose $\mathbf{x}_t$ and map of the (static) environment $\mathcal{M}$ given all the control and sensor data up to time t, i.e. $p\left(\mathbf{x}_t, \mathcal{M} | \mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$. Thus, the overall state vector $\mathbf{\Xi}_t$ includes both the map $\mathcal{M}$ and the pose of the robot $\mathbf{x}_t$, i.e. $\mathbf{\Xi}_t = (\mathbf{x}_t, \mathcal{M})$. Formulated in this way, the various filters discussed in Chapter 2 can be directly applied to solve SLAM. As already mentioned, to implement the filters in Chapter 2, one needs to define the probabilistic system model $p\left(\mathbf{\Xi}_t | \mathbf{\Xi}_{t-1}, \mathbf{c}_t\right) = p\left(\mathbf{x}_t, \mathcal{M} | \mathbf{x}_{t-1}, \mathcal{M}, \mathbf{c}_t\right)$ that describes how the system evolves. Since the environment is static (as is the assumption for most work on SLAM and in this chapter), then the map $\mathcal{M}$ should not change and should not be affected by the control input $\mathbf{c}_t$. Additionally, one also needs to specify the prior distribution $p\left(\mathbf{\Xi}_0\right) = p\left(\mathbf{x}_0, \mathcal{M}\right)$ over the robot's pose $\mathbf{x}_0$ and map $\mathcal{M}$ at time 0.

When using a particle filter directly for SLAM, at initialization, one needs to be able to generate samples for the robot's pose and map of the environment from the prior $p(\mathbf{x}_0, \mathcal{M})$. With the static world assumption, the sampled maps at initialization are fixed and should never change. Herein lies the problem of directly using a particle filter for SLAM: the space of all possible maps is infinite and the dimension of this space is also infinite and thus being able to sample a map that is close to the true map is close to impossible. The particle filter, therefore, is guaranteed to fail since it will (with very high probability) not converge to the true state of the system.

In order to avoid the above problem, we use a variant of the particle filter known as the *Rao-Blackwellized particle filter* (RBPF) [Murphy, 2000]. RBPFs have been introduced as an effective way to solve the SLAM problem (e.g. the FastSLAM system of Montemerlo et al. [2002] and Montemerlo and Thrun [2003]). When applied to SLAM, RBPF decomposes the entire problem of jointly estimating the robot's pose and map of the environment into a robot localization problem and a map estimation problem conditioned on the robot pose estimate. That is,

$$p(\mathbf{x}_{0:t}, \mathcal{M}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t}) = p(\mathbf{x}_{0:t}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t})\, p(\mathcal{M}|\mathbf{x}_{0:t}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t}) \tag{5.8}$$

$$= p(\mathbf{x}_{0:t}|\mathbf{c}_{1:t}, \mathbf{s}_{1:t})\, p(\mathcal{M}|\mathbf{x}_{0:t}, \mathbf{s}_{1:t}) \ , \tag{5.9}$$

where $p(\mathcal{M}|\mathbf{x}_{0:t}, \mathbf{c}_{1:t}, \mathbf{s}_{1:t}) = p(\mathcal{M}|\mathbf{x}_{0:t}, \mathbf{s}_{1:t})$ (that is, the map $\mathcal{M}$ is independent of the controls $\mathbf{c}_{1:t}$). Note that the factorization shown in Equation 5.9 is exact. In order to approx-

imate the distribution $p\left(\mathbf{x}_{0:t}, \mathcal{M} | \mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$, a particle filter is used to estimate the posterior $p\left(\mathbf{x}_{0:t} | \mathbf{c}_{1:t}, \mathbf{s}_{1:t}\right)$ over potential robot trajectories and a distribution over map $\mathcal{M}_t$ is associated with every sample $\mathbf{x}_{0:t}^{[i]}$ in the filter.[3] This map distribution associated with the ith particle represents $p\left(\mathcal{M} \left| \mathbf{x}_{0:t}^{[i]}, \mathbf{s}_{1:t} \right.\right)$. Each distribution over $\mathcal{M}_t$ is incrementally built given the robot trajectory $\mathbf{x}_{0:t}^{[i]}$ of the associated particle and the sensor observations $\mathbf{s}_{1:t}$. The robot trajectory evolves according to the robot motion given by the probabilistic motion model in Section 5.3.1 and as such, we can still use Algorithm 5 in Chapter 2. In our work, each distribution over map $\mathcal{M}_t$ is represented as $\mathcal{M}_t^{[i]}$, a set of line segments (to represent the presence of walls, doors, windows, etc.). It is an approximation to the full distribution $p\left(\mathcal{M}_t \left| \mathbf{x}_{0:t}^{[i]}, \mathbf{s}_{1:t} \right.\right)$ over possible maps given the robot trajectory according to the ith particle and the sensor data $\mathbf{s}_{1:t}$. While at first glance, a set of line segments $\mathcal{M}_t^{[i]}$ does not appear to be a distribution over possible maps, we can view it as a distribution over maps by letting it represent the most likely map. The presence of a line in the map represents an obstacle that is most likely to be in the environment while the absence of a line may be interpreted as insufficient or uncertain knowledge of about the presence of an obstacle in the environment. This interpretation explains why we need to perform ray tracing twice, first using the map $\mathcal{M}_t^{[i]}$ and then using the same map but with extended lines segments, taking the average of the results for computing the weight for the ith particle. In doing so, we are taking a heuristic average over other likely maps "near" the most likely map. While this is not a formal implementation of a

[3]Please note that the time subscript t for each individual map is not used to mean that the environment or its map is dynamic but rather to indicate that its posterior distribution is incrementally modified as a result of incorporating sensor measurements.

Rao-Blackwellized particle filter, it is a heuristic that requires little computational effort, and performs well in practice.

5.6 Experimental Results

We have implemented our SLAM approach for sonars and tested it on the ActivMedia robotics P3-DX mobile robot platform. The robot is equipped with a front sonar array with eight sensors, one on each side and six facing forward at 20° intervals. It also has a rear sonar array with eight sensors, one on each side and six facing backward at 20° intervals. Therefore, a single sonar scan yields a total of sixteen sonar measurements. Of the sixteen sonar measurements in a single scan, only a small fraction actually correspond to correct measurements while the rest are unreliable due to angular uncertainty (20° to 30° for sonars), specular and multiple reflections, crosstalk, etc. This is in stark contrast to typical laser rangefinders that produce accurate (with a statistical error of ≈ 5mm in range measurement and angular beam width of $\leq 1^\circ$ for each beam) and dense measurements (e.g. 180 or 360 measurements in each scan) and they are not affected by problems such as specular reflections and crosstalk. As such, SLAM with sonars is much more difficult and challenging than SLAM with laser rangefinders. Despite the shortcomings of sonars, our experimental results show that it is possible to produce good-quality maps of large indoor environments with large loops using sonars.

In our experiments, the robot is controlled by an IBM ThinkPad X32 notebook computer and it navigated around different environments by visiting predefined waypoints while collecting control and sensor data along the way. We considered three test environments of increasing sizes and complexities. The first test environment is a makeshift environment that represents a scaled-down version of a typical office environment. The associated map of the environment has an approximate size of 6.7m $\times$ 6.7m. It is mostly made up of smooth cardboard walls. The second test environment is a portion of the faculty suite on the third floor of our Engineering Building II. It is bigger than the first test environment with an approximate map size of 38.6m $\times$ 12.7m and it contains two major loops. Finally, the third test environment is the entire hallway of the third floor of our Engineering Building II, including the two bridges connecting to the old engineering building. It is the biggest environment in our experiments with a map size of approximately 70.0m $\times$ 53.7m and it contains two major loops. The third test environment is made up of various types of obstacles such as brick walls, tiled walls, cable railing, glass windows, trash bins, etc. Unlike the first two test environments, the third test environment is dynamic with people walking along the corridors during the experiments. Figure 5.3 shows the three test environments and their associated maps while Table 5.1 provides summary information about our experiments.

To show the effectiveness of our SLAM approach using sonars to compensate for odometric errors, we show in Figure 5.3 (right column) the trajectory from the raw encoder readings (broken lines) against the desired path of the robot (solid lines) for all three test environments. It is evident from Figure 5.3 (right column) that the robot's odometry suffers

Table 5.1: SLAM experiments summary.

	Test Environment 1	Test Environment 2	Test Environment 3
Map Size	$\approx 6.7\text{m} \times 6.7\text{m}$	$\approx 38.6\text{m} \times 12.7\text{m}$	$\approx 70.0\text{m} \times 53.7\text{m}$
Data Collection Time	≈ 5 minutes	≈ 9 minutes	≈ 30 minutes
Distance Traveled	26.7m	85.7m	283.5m
Total Time Steps T	545	727	1985
Number of Measurements $\neq s_{\text{max}}$	5915	7577	19348
Number of Particles Used N_{s}	200	300	200

from drift that gets more pronounced in larger environments. Therefore, relying only on the robot's odometry for performing SLAM is not sufficient and sonar measurements taken must be used for correcting the robot's pose. Figure 5.4 (left column) shows the resulting maps and robot trajectories with our approach for the three test environments. Although the generated maps are not exactly the same as the true maps, they do capture the main structure of the environments. Additionally, our approach managed to close the loops properly in all test environments which is generally considered a difficult problem in SLAM. Lines in our maps correspond to actual major obstacles such as walls, glass windows, doors, cable railings, as well as to some minor ones such as trash bins and posts. Because of the Hough transform, our line extraction procedure is quite robust to noise caused by specular and multiple reflections and phantom readings. Also, since we only consider horizontal and vertical lines, our line extraction procedure is effective at filtering out dynamic objects particularly for the third test environment.

We also performed experiments without using the orthogonality assumption and the final maps and robot trajectories are shown in Figure 5.4 (middle column). Without the orthogonality assumption, the resulting maps and robot trajectories are not properly estimated and

corrected. Finally, we implemented a RBPF using grid maps for SLAM based on the map matching technique of Schröter et al. [2007], and the resulting maps for all three test environments are also shown in Figure 5.4 (right column). Note that we used the same set of parameter values for generating the results for all three test environments with our approach (except for the number of particles N_{s}) while we had to use different sets of parameter values to generate the resulting grid maps shown in Figure 5.4 (right column) using RBPF with map matching in order to obtain reasonable results. We can easily see the effect of data smearing when using a cell-based approach to SLAM; measurements are often blurred onto a region of the map to account for angular and distance uncertainty.

5.7 Summary

SLAM has received considerable attention in the mobile robotics community for the last two decades. However, much of the research effort for SLAM has focused on the use of highly accurate and dense measurements provided by laser rangefinders to correctly localize the robot and produce accurate and detailed maps of complex environments. In this chapter, we presented an approach to SLAM for a mobile robot equipped with low-cost but noisy and sparse sonar sensors navigating in large indoor environments involving large loops. The proposed approach applies particle filtering where each particle is an estimate of the recent robot poses and carries a map of the environment represented as a set of line segments. To overcome the sparseness of sonars and to allow for the reliable extraction of line segment features

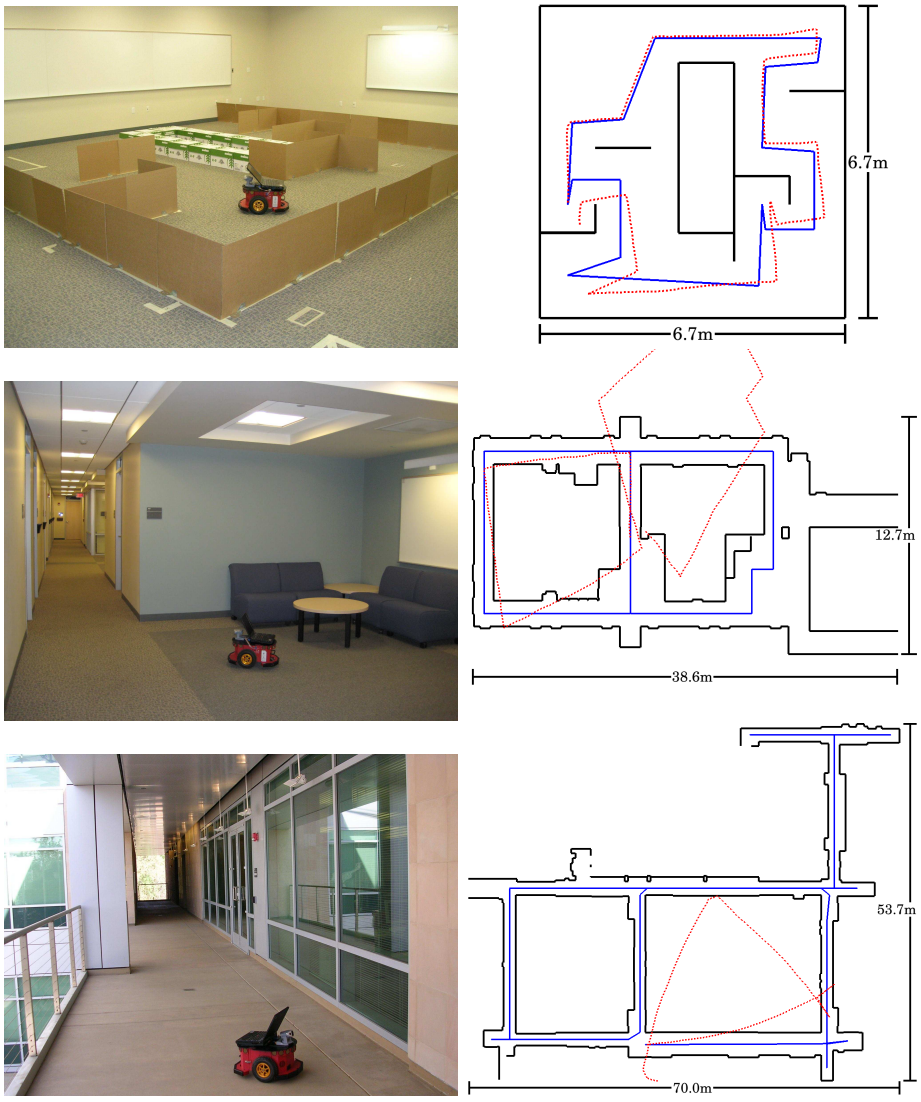


Figure 5.3: The three test environments (left column) and their associated maps (right right). The right column also shows the paths of the robot (solid lines) and the trajectory estimates based from encoder readings (broken lines). Figure taken from [Yap and Shelton, 2009] © [2009] IEEE.

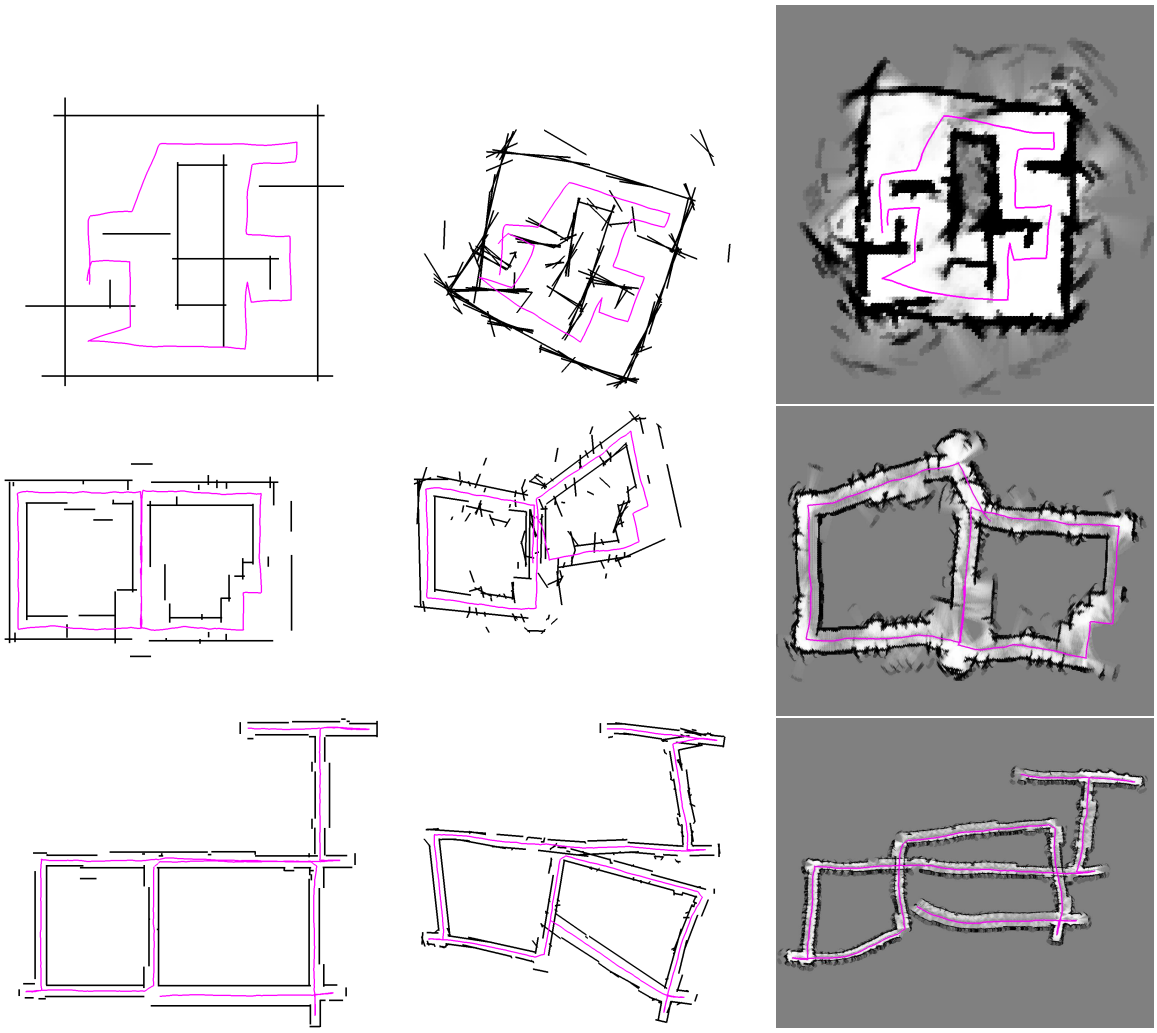


Figure 5.4: The estimated maps and robot trajectories with our approach (left column). Maps and robot trajectories estimated without using the orthogonality assumption (middle column). Maps and robot trajectories estimated using a RBPF using grid maps (right column). For the grid maps, dark regions indicate high level of occupancy, white regions indicate low level of occupancy, and gray regions represent unexplored areas. Figure taken from [Yap and Shelton, 2009] © [2009] IEEE.

from the environment, we used the multiscan approach and grouped consecutive sparse scans into multiscans. To reduce the complexity of SLAM particularly when sonars are used, we applied the orthogonality assumption about the shape of the environment by mapping only lines that are parallel or perpendicular to each other. The orthogonality assumption is reasonable especially for most man-made indoor environments where major structures such as walls, windows, and doors are either parallel or perpendicular to each other. The randomized Hough transform was used to extract line segments from multiscans and it was quite robust to noise caused by specular and multiple reflections and phantom readings, problems usually associated with sonars. Despite the inherent limitations of sonars, results of empirical validation, carried out using a real mobile robot platform navigating in different environments of increasing sizes and complexities, provide supportive evidence for the potential viability of sonars for complex large-scale indoor SLAM.

Chapter 6

Conclusions

Mobile robots are becoming ubiquitous and essential part of our everyday lives. They are increasingly taking their place in service-oriented applications including domestic and entertainment. They open up many potential opportunities but they also come with challenges in terms of their limited sensing capability and accuracy and minimal on-board computing resources. In this dissertation, we have addressed three important problems in mobile robotics and demonstrated our approach to each of the problems with a mobile robot with low-cost and low-end sensors. The problems we have considered are those of mobile robot calibration, mobile robot localization, and simultaneous localization and mapping.

We presented an automated technique for simultaneously calibrating a mobile robot's motion and sensor models based from the control and sensor data obtained naturally during robot operations. Our method is based on a standard machine learning method called the EM algorithm. Starting from some initial parameter values, EM iteratively optimizes the

parameters based from the collected data. The method we proposed does not require any special calibration setup and it can be performed by the robot as it operates with little or no human intervention. The estimation procedure can be readily invoked by the robot at a later time when there is a need to recalibrate the models. Experimental results show the effectiveness of our estimation approach and the advantage of co-calibrating the models.

Monocular vision has long been advertised as an attractive sensor for mobile robot localization. In this dissertation, we have presented a particle filtering approach for monocular-vision aided odometry for the real-time localization of a mobile robot navigating in indoor environments. Monocular vision is used for detecting naturally occurring static three-dimensional point features or landmarks from the environment and the information is utilized for correcting the pose as suggested by the odometry. We introduced a sensor model for assigning weights to the particles based on their trajectory and the measurements of the features observed along the trajectory. The approach has been validated using both simulation as well as real-world data and compared against that of the EKF.

For the last two decades, SLAM has been the focus of much research in the mobile robotics community. However, much of the research effort for SLAM has focused on the use of laser rangefinders to correctly localize the robot and produce accurate and detailed maps of complex environments. In this dissertation, we presented an approach to SLAM for a mobile robot equipped with low-cost but noisy and sparse sonar sensors navigating in large indoor environments with large loops. We used a particle filter where each particle is an estimate of the recent robot poses and it carries a map of the environment represented as a set of

line segments. To overcome the sparseness of sonar scans and reliably extract line segments from the environment, we grouped consecutive scans into multiscans. The orthogonality assumption about the shape of the environment is applied by mapping only horizontal and vertical lines. The randomized Hough transform was used to extract line segments from multiscans and it was quite robust to noise caused by specular and multiple reflections and phantom readings, problems usually associated with sonars. Results of empirical validation using a real mobile robot platform navigating in different environments of increasing sizes and complexities provide supportive evidence for the potential viability of low-cost sonars for complex large-scale indoor SLAM.

Taken together, the results of this dissertation demonstrate that good navigation performance by a small-scale mobile robot can be achieved using low-cost sensors. Such results are especially important for mobile service robots that are equipped with modest sensors.

Bibliography

Fabrizio Abrate, Basilio Bona, and Marina Indri. Experimental EKF-based SLAM for mini-rovers with IR sensors only. In *Proceedings of the Third European Conference on Mobile Robots*, Freiburg, Germany, September 2007.

M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117, September 2006.

Kristopher R. Beevers and Wesley H. Huang. SLAM with sparse sensing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2285–2290, Orlando, Florida, USA, May 2006.

Stan Birchfield. KLT: Kanade-Lucas-Tomasi feature tracker. `http://www.ces.clemson.edu/~stb/klt/installation.html`, August 2007.

Johann Borenstein and Liqiang Feng. UMBmark: A benchmark test for measuring odometry errors in mobile robots. In *Proceedings of the 1995 SPIE Conference on Mobile Robots*, pages 113–124, Philadephia, Pennsylvania, USA, October 1995.

Johann Borenstein, H. R. Everett, and Liqiang Feng. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Wellesley, Massachusetts, USA, 1996.

Jean-Yves Bouguet. Camera calibration toolbox for Matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, June 2008.

Emma Brunskill and Nicholas Roy. SLAM using incremental probabilistic PCA and dimensionality reduction. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 342–347, Barcelona, Spain, April 2005.

Daniele Caltabiano, Giovanni Muscato, and Francesco Russo. Localization and self calibration of a robot for volcano exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 586–591, New Orleans, Louisiana, USA, April/May 2004.

Frédéric Chenavier and James L. Crowley. Position estimation for a mobile robot using vision and odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2588–2593, Nice, France, May 1992.

Young-Ho Choi, Tae-Kyeong Lee, and Se-Young Oh. A line feature based SLAM with low grade range sensors using geometric constraints and active exploration for mobile robot. *Autonomous Robots*, 24(1):13–27, January 2008.

I. J. Cox and G. T. Wilfong, editors. *Autonomous Robot Vehicles*. Springer-Verlag, New York, New York, USA, September 1990.

Ingemar J. Cox. Blanche–an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, April 1991.

James L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 674–680, Scottsdale, Arizona, USA, May 1989.

Andrew J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 2, pages 1403–1410, Nice, France, October 2003.

Frank Dellaert. The expectation maximization algorithm. Technical Report GIT-GVU-02-20, College of Computing, Georgia Institute of Technology, February 2002.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

M. W. M. Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, June 2001.

Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, July 2000a.

Arnaud Doucet, Simon J. Godsill, and Mike West. Monte Carlo filtering and smoothing with application to time-varying spectral estimation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 701–704, Istanbul, Turkey, June 2000b.

Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, January 1972.

Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: Part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110, June 2006.

Ethan Eade and Tom Drummond. Scalable monocular SLAM. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 469–476, New York, New York, USA, June 2006.

Alberto Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, June 1987.

Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989.

Austin Eliazar and Ronald Parr. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1135–1142, Acapulco, Mexico, August 2003.

Austin I. Eliazar and Ronald Parr. DP-SLAM 2.0. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1314–1320, New Orleans, Louisiana, USA, April/May 2004a.

Austin I. Eliazar and Ronald Parr. Learning probabilistic motion models for mobile robots. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 32–39, Banff, Alberta, Canada, July 2004b.

Austin I. Eliazar and Ronald Parr. Hierarchical linear/constant time SLAM using particle filters for dense maps. In *Advances in Neural Information Processing Systems 18*, pages 339–346, 2006.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, November 1999.

Eric M. Foxlin. Generalized architecture for simultaneous localization, auto-calibration, and map-building. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 527–533, Lausanne, Switzerland, October 2002.

Andrea Garulli, Antonio Giannitrapani, Andrea Rossi, and Antonio Vicino. Mobile robot SLAM for line-based environment representation. In *Proceedings of the Forty-Fourth IEEE Conference on Decision and Control and the European Control Conference*, pages 2041–2046, Seville, Spain, December 2005.

Simon Godsill, Arnaud Doucet, and Mike West. Maximum a posteriori sequence estimation using Monte Carlo particle filters. *Annals of the Institute of Statistical Mathematics*, 53 (1):82–96, March 2001.

Simon J. Godsill, Arnaud Doucet, and Mike West. Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168, March 2004.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F Radar and Signal Processing*, 140(2):107–113, April 1993.

Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based SLAM with Rao-Blackwellised particle filters by adaptive proposals and selective resampling. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2432–2437, Barcelona, Spain, April 2005.

Dirk Hähnel, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 206–211, Las Vegas, Nevada, USA, October 2003.

International Federation of Robotics Statistical Department. *World Robotics 2009 Service Robots: Statistics, Market Analysis, Forecasts, and Case Studies*. September 2009.

Michael Isard and Andrew Blake. CONDENSATION–conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, August 1998.

Kazufumi Ito and Kaiqi Xiong. Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control*, 45(5):910–927, May 2000.

Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proceedings of the SPIE International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, volume 3068, pages 182–193, Orlando, Florida, USA, April 1997.

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

Niklas Karlsson, Enrico Di Bernardo, Jim Ostrowski, Luis Goncalves, Paolo Pirjanian, and Mario E. Munich. The vSLAM algorithm for robust localization and mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 24–29, Barcelona, Spain, April 2005.

Pekka Kultanen, Lei Xu, and Erkki Oja. Randomized Hough transform (RHT). In *Proceedings of the Tenth International Conference on Pattern Recognition*, volume 1, pages 631–635, Atlantic City, New Jersey, USA, June 1990.

John J. Leonard and Hugh F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, June 1991a.

John J. Leonard and Hugh F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, volume 3, pages 1442–1447, Osaka, Japan, November 1991b.

John J. Leonard, Richard J. Rikoski, Paul M. Newman, and Michael Bosse. Mapping partially observable features from multiple uncertain vantage points. *The International Journal of Robotics Research*, 21(10–11):943–975, October/November 2002.

J. M. Pérez Lorenzo, R. Vázquez-Martín, P. Núñez, E. J. Pérez, and F. Sandoval. A Hough-based method for concurrent mapping and localization in indoor environments. In *Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics*, volume 2, pages 840–845, Traders Hotel, Singapore, December 2004.

Agostino Martinelli, Nicola Tomatis, Adriana Tapus, and Roland Siegwart. Simultaneous localization and odometry calibration for mobile robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1499–1504, Las Vegas, Nevada, USA, October 2003.

Thomas P. Minka. Expectation-maximization as lower bound maximization. `http://research.microsoft.com/en-us/um/people/minka/papers/em.html`, November 1998.

Michael Montemerlo and Sebastian Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1985–1991, Taipei, Taiwan, September 2003.

Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 593–598, July/August 2002.

Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2): 61–74, July/August 1988.

E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Monocular vision based SLAM for mobile robots. In *Proceedings of the Eighteenth International Conference on Pattern Recognition*, volume 3, pages 1027–1031, Hong Kong, August 2006.

Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. Technical report, Department of Computer Science and Engineering, University of Minnesota, September 2006.

Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3565–3572, Roma, Italy, April 2007.

Kevin P. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems 12*, pages 1015–1021, 2000.

Radford Neal and Geoffrey E. Hinton. *Learning in Graphics Models*, pages 355–368. MIT Press, Cambridge, Massachusetts, USA, November 1998.

Viet Nguyen, Ahad Harati, Agostino Martinelli, Roland Siegwart, and Nicola Tomatis. Orthogonal SLAM: a step toward lightweight indoor autonomous navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5007–5012, Beijing, China, October 2006.

Michael K. Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, June 1999.

Mark Pupilli and Andrew Calway. Real-time camera tracking using a particle filter. In *Proceedings of the British Machine Vision Conference*, pages 519–528, Oxford Brookes University, Oxford, UK, September 2005.

W. D. Rencken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2192–2197, Yokohama, Japan, July 1993.

Nicholas Roy and Sebastian Thrun. Online self-calibration for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2292–2297, Detroit, Michigan, USA, May 1999.

Eric Royer, Maxime Lhuillier, Michel Dhome, and Thierry Chateau. Localization in urban environments: monocular vision compared to a differential GPS sensor. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 114–121, San Diego, CA, USA, June 2005.

Eric Royer, Maxime Lhuillier, Michel Dhome, and Jean-Marc Lavest. Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, 74(3):237–260, September 2007.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc., Upper Saddle River, New Jersey, USA, second edition, 2003.

Christof Schröter, Hans-Joachim Böhme, and Horst-Michael Gross. Memory-efficient gridmaps in Rao-Blackwellized particle filters for SLAM using sonar range sensors. In *Proceedings of the Third European Conference on Mobile Robots*, Freiburg, Germany, September 2007.

Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical Report CS-94-125, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, August 1994.

Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, Washington, USA, June 1994.

Randall Smith, Matthew Self, and Peter Cheeseman. *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, New York, New York, USA, September 1990.

Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986.

Daniel Stronger and Peter Stone. Simultaneous calibration of action and sensor models on a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4563–4568, Barcelona, Spain, April 2005.

Juan D. Tardós, José Neira, Paul M. Newman, and John J. Leonard. Robust mapping and localization in indoor environments using sonar data. *The International Journal of Robotics Research*, 21(4):311–330, April 2002.

Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2):111–127, September 2003.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 321–328, San Francisco, California, USA, April 2000.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, Massachusetts, USA, 2005.

Gerhard Weiß, Christopher Wetzler, and Ewald von Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 595–601, Munich, Germany, September 1994.

Greg Welch and Gary Bishop. An introduction to Kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, USA, July 2006.

Teddy N. Yap, Jr. and Christian R. Shelton. Simultaneous learning of motion and sensor model parameters for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2091–2097, Pasadena, California, USA, May 2008.

Teddy N. Yap, Jr. and Christian R. Shelton. SLAM in large indoor environments with low-cost, noisy, and sparse sonars. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1395–1401, Kobe, Japan, May 2009.

Guido Zunino and Henrik I. Christensen. Navigation in realistic environments. In *Proceedings of the Ninth International Symposium on Intelligent Robotics Systems*, pages 405–413, Toulouse, France, July 2001.