TOWARDS DEPENDABLE ROBOTIC PERCEPTION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Anna Petrovskaya

June 2011

© Copyright by Anna Petrovskaya 2011
All Rights Reserved

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Oussama Khatib) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Mark Cutkosky)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Vaughan Pratt)

Approved for the University Committee on Graduate Studies

Abstract

Reliable perception is required in order for robots to operate safely in unpredictable and complex human environments. However, reliability of perceptual inference algorithms has been poorly studied so far. These algorithms capture uncertain knowledge about the world in the form of probabilistic belief distributions. A number of Monte Carlo and deterministic approaches have been developed, but their efficiency depends on the degree of smoothness of the beliefs. In the real world, the smoothness assumption often fails, leading to unreliable perceptual inference results.

Motivated by concrete robotics problems, we propose two novel perceptual inference algorithms that explicitly consider local non-smoothness of beliefs and adapt to it. Both of these algorithms fall into the category of iterative divide-and-conquer methods and hence scale logarithmically with desired accuracy. The first algorithm is termed Scaling Series. It is an iterative Monte Carlo technique coupled with annealing. Local non-smoothness is accounted for by sampling strategy and by annealing schedule. The second algorithm is termed GRAB, which stands for Guaranteed Recursive Adaptive Bounding. GRAB is an iterative adaptive grid algorithm, which relies on bounds. In this case, local non-smoothness is captured in terms of local bounds and grid resolution. Scaling Series works well for beliefs with sharp transitions, but without many discontinuities. GRAB is most appropriate for beliefs with many discontinuities. Both of these algorithms far outperform the prior art in terms of reliability, efficiency, and accuracy. GRAB is also able to guarantee that a quality approximation of the belief is produced.

The proposed algorithms are evaluated on a diverse set of real robotics problems: tactile perception, autonomous driving, and mobile manipulation. In tactile perception, we localize objects in 3D starting with very high initial uncertainty and estimating all 6 degrees of freedom. The localization is performed based on tactile sensory data. Using Scaling Series, we obtain highly accurate and reliable results in under 1 second. Improved tactile object localization contributes to manufacturing applications, where tactile perception is widely used for workpiece localization. It also enables robotic applications in situations where vision can be obstructed, such as rescue robotics and underwater robotics.

In autonomous driving, we detect and track vehicles in the vicinity of the robot based on 2D and 3D laser range finders. In addition to estimating position and velocity of vehicles, we also model and estimate their geometric shape. The geometric model leads to highly accurate estimates of pose and velocity for each vehicle. It also greatly simplifies association of data, which are often split up into separate clusters due to occlusion. The proposed Scaling Series algorithm greatly improves reliability and ensures that the problem is solved within tight real time constraints of autonomous driving.

In mobile manipulation, we achieve highly accurate robot localization based on commonly used 2D laser range finders using the GRAB algorithm. We show that the high accuracy allows robots to navigate in tight spaces and manipulate objects without having to sense them directly. We demonstrate our approach on the example of simultaneous building navigation, door handle manipulation, and door opening. We also propose hybrid environment models, which combine high resolution polygons for objects of interest with low resolution occupancy grid representations for the rest of the environment. High accuracy indoor localization contributes directly to home/office mobile robotics as well as to future robotics applications in construction, inspection, and maintenance of buildings.

Based on the success of the proposed perceptual inference algorithms in the concrete robotics problems, it is our hope that this thesis will serve as a starting point for further development of highly reliable perceptual inference methods.

Acknowledgements

I would like to thank the many people I have worked with during my PhD program. First of all, thanks to my advisor and friend Oussama Khatib, who has gone above and beyond anything I could have expected from an advisor. Oussama's guidance in life and research have proven invaluable during these years.

Thanks to my reading committee: Mark Cutkosky and Vaughan Pratt, whose insightful comments and enthusiasm made finishing the dissertation an enjoyable experience. Thanks to Bernie Roth and Ken Salisbury, who served on my defense committee. Your keen interest in my work and deep questions made the defense discussion memorable and worthwhile.

Also thanks to all the other faculty I had the privilege to work with at Stanford: Sebastian Thrun, Daphne Koller, Andrew Y. Ng, and Rajeev Motwani. You have greatly contributed to my research and my development as a scientist.

Thanks to my lab mates Jaeheung Park, Irena Paschenko, Dongjun Shin, Peter Thaulad, Francois Conti, Luis Sentis, Vincent De Sapio, Vincent Padois, Irene Sardellitti, Steve Burion, Jinsung Kwon, Emel Demircan, Ellen Klingbell, Torsten Kroeger, Roland Phillipsen, Taizo Yoshikawa, Takeo Maruyama, Nicolas Tournier, Nicolas Mansard. You made my experience in the lab special and unforgettable and I cherish all the good times that we have spent together.

Thanks to my team mates from the Stanford Racing Team: Mike Montemerlo, David Stavens, Jesse Levinson, Dirk Langer, Ganny Stanek, Hendrik Dahlkamp, Gabe Hoffmann, Dmitri Dolgov, Jan Becker, Scott Ettinger, Pamela Mahoney, Anthony Levandowski, Dirk Haehnel, David Orenstein, and others. Building Stanford's autonomous vehicle Junior was a unique once-in-a-lifetime endeavor.

Thanks to my office mates: Morgan Quigley, Adam Coates, Alan Asbeck, Ashutosh Saxena, Jamie Schulte, and Quoc V. Le, who were always enthusiastic and interested in my work and were a pleasure to share an office with.

Thanks to Stanford colleagues with whom I had the pleasure to work with over the years: Pieter Abbeel, Christian Plagemann, Alex Segal, Barrett Heyneman, Christian Theobalt, Jimmy Zhang, and others.

Also thanks to the many friends and colleagues that I have met around the world during these years: Cyrill Stachniss, Maren Bennewitz, Ana Cristina Murillo Arnal, Eve Coste-Maniere, Kaijen Hsiao, Michel Parent, Gary Bradski, Jose Neira, Christian Laugier, Philippe Martinet, Aurélie Clodic, Antonio Morales, Antonio Frisoli, Nicola Dioliati, Sung-Chul Kang, Chungmook Chun, and many others. Your friendship and stimulating discussions make me always look forward to seeing you at one of the international or local events.

Finally, thanks to my parents: Tatiana and Vladimir Petrovsky, who have raised me to have a keen interest in science and research. Thanks to my husband Peter Varvak for his unconditional love and enthusiasm. Thanks to my daughter Sophia, who had to endure many nights and weekends away from her mother. And thanks to our nanny Alla Salova, without whom my progress would not be possible.

to Peter and Sophia

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Clouds are not spheres, mountains are not cones, coastlines are not circles, and bark is not smooth, nor does lightning travel in a straight line.

B. Mandelbrot,
The Fractal Geometry of Nature.

In uncertain environments, robots need to understand the world around them based on sensory inputs. Robots achieve this via a process known as perception, which transforms sensory data into useful information. Clearly, in order for robots to operate reliably in unstructured settings, perception needs to be dependable. However, to improve efficiency, many perceptual algorithms make simplifying smoothness assumptions about the world. Since the real world is anything but smooth, this discrepancy makes it challenging to achieve reliable perception. In this thesis, we consider these challenges and propose two novel inference algorithms, which address the issues without requiring the use of more complex models. Further, we illustrate the algorithms on a number of real robotics problems from a diverse set of domains: tactile perception, autonomous driving, and mobile manipulation.

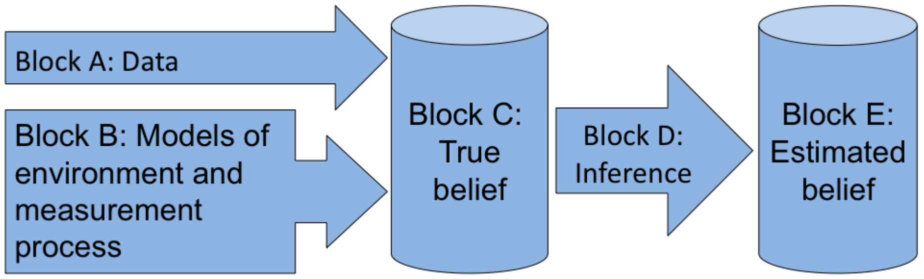

Block A: Data
Block B: Models of environment and measurement process
Block C: True belief
Block D: Inference
Block E: Estimated belief

Figure 1.1: The inner workings of perception.

1.1 Perception and Challenges Therein

The basic building blocks of perception are shown in Fig. 1.1. Perception relies on physical sensors (e.g., a stereo camera or a tactile array), which provide the robot with sensory data (block A). The robot interprets this data based on models of the environment and the *measurement process* (block B). Since the data are noisy and the models are imprecise, it is not possible to extract exact information: e.g., the object is located exactly 50cm ahead. Instead the information is captured using *Bayesian statistics* in the form of a probability distribution (block C), often referred to as the *belief* or the *posterior* distribution. This distribution represents uncertain information: e.g., the object is most likely located 50cm ahead, but could also be a little closer or further away.

Once models and data are given, the belief distribution is uniquely defined. To emphasize this fact, we will often refer to this distribution as the *true belief*. Although the true belief is uniquely defined, it is in general not known to us or the robot. Hence, we need to estimate this distribution using *approximate inference* algorithms (block D). The result is an estimated belief (block E), which can be used by decision algorithms: e.g., if the most likely position of the object is 50cm ahead, then the robot may decide to move its arm to this position in an attempt to grasp the object.

1.1.1 Making Perception Robust

In order to make perception robust, we first need to examine which parts of the diagram in Fig. 1.1 could fail and under what conditions. In principle, blocks A, B, and D could fail. Block A fails when there is a failure within the physical sensor or communications with the sensor, which lead to faulty sensory data. Block B could fail, if the environment or the measurement process are not modeled correctly. Since it is not possible to make exact models of the physical world, there are always discrepancies between models and reality. Finally, even if there are no failures in models or data, block D could still fail when the approximate inference algorithm fails to produce an accurate enough estimate of the belief.

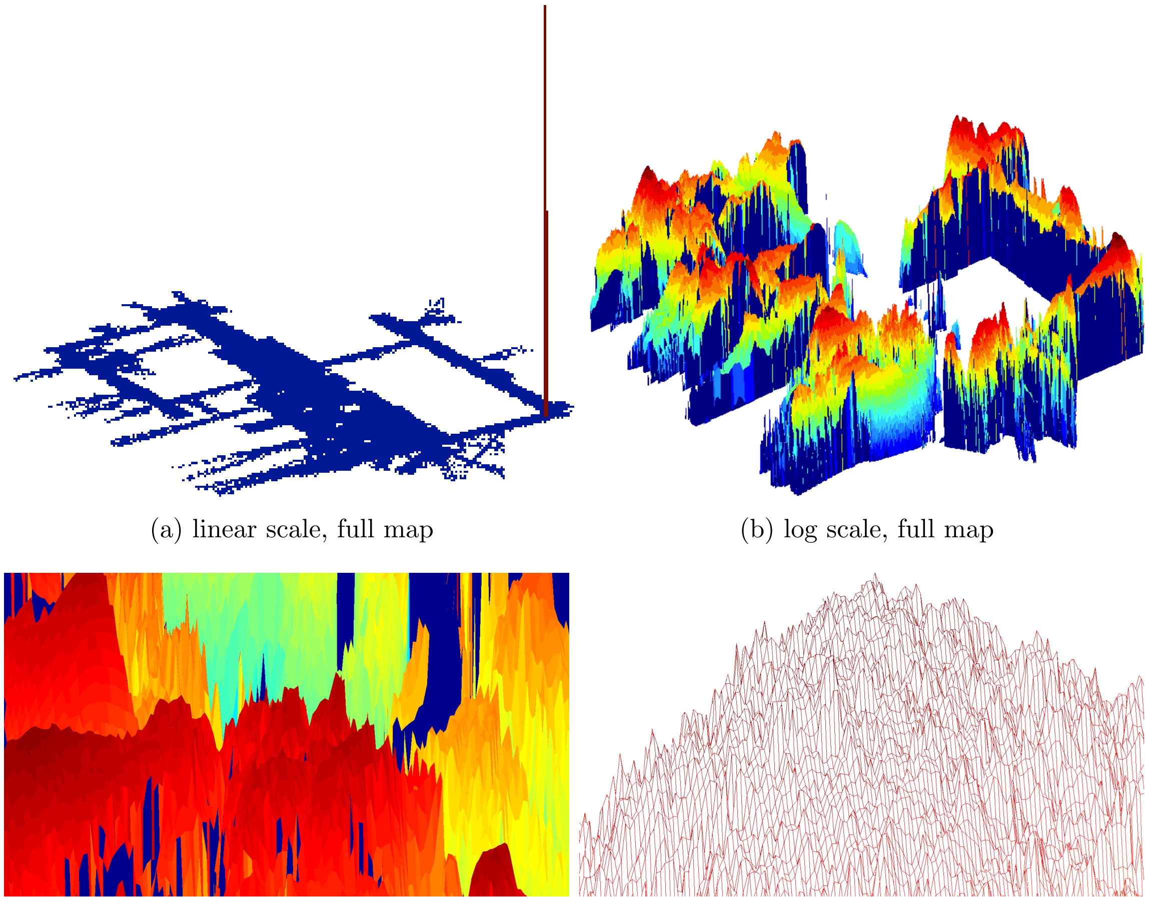
A lot of work in robotics goes into improving the physical sensors and into devising better models. However, in this thesis, we are primarily focused on eliminating failures in block D (approximate inference). In the next subsection, we examine the challenges that arise within this block.

1.1.2 Challenges In Perceptual Inference

The goal of approximate inference is to produce an estimate of the belief. The process can be viewed as numerical estimation of a real-valued function over a multi-dimensional space. However, it turns out that the functions that arise in robotics are among some of the most challenging functions for numerical estimation. The beliefs in robotics are characterized by multiple very narrow peaks with many discontinuities (see Fig. 1.2). Moreover, the peaks can form ridges of complex shape. We will refer to beliefs with these properties as *high roughness* beliefs, due to their similarity to rough terrain with frequent sharp transitions. We use the term roughness here in a sense very similar to the notion used in *fractal geometry* by Mandelbrot [1982]. As Mandelbrot points out, roughness is a very natural property of the world we live in: tree bark, mountain ranges, and coastlines are all non-smooth. In robotics, roughness of belief distributions is a direct consequence of the properties of the physical world and physical sensors.

Causes of High Roughness

Consider the example of sensing an object with a laser range finder in Fig. 1.3. For simplicity, we will assume that the laser range finder only has one ray. As long as



(a) linear scale, full map

(b) log scale, full map

(c) log scale, zoom to 10m

(d) log scale, zoom to 1m

Figure 1.2: An example of a real belief in robotics: indoor localization with laser range finders. Linear scale plot of the belief shows a single dominant peak, which is very narrow. Log scale plots reveal local structure. Note the high roughness persists even as we zoom in.

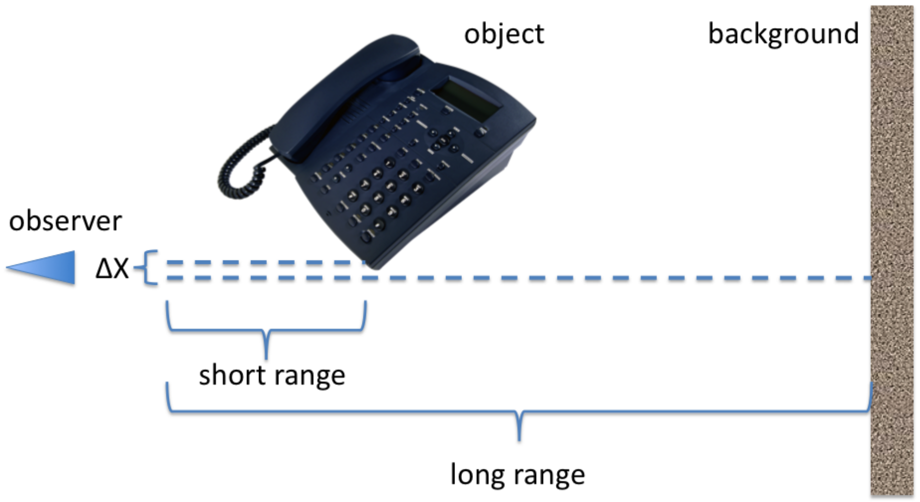


Figure 1.3: Sensing an object with a laser range finder — an illustration of sensing discontinuity. Note that even a small change in the observer's position can cause a large and abrupt change in the measured range.

the laser ray hits the surface of the object, we read a short range. However, if the observer shifts so that the ray just misses the object's surface — even by a very tiny amount — then we read a long range. So there is a discontinuity in the range we read based on the pose of the observer relative to the object. This discontinuity leads directly to a discontinuity in the belief distribution: i.e., for a short reading, poses in which the ray hits the object are likely, whereas poses in which the ray misses the object are very unlikely. This simple example carries over to other sensors and other objects. Hence, *discontinuities in the belief are caused by occlusion boundaries in the physical world.*

Multiple peaks and ridges result from *ambiguities* (or *symmetries*) in the environment. Since we are not able to distinguish between these states based on sensory data, all of these states are equally likely. The width of the peaks depends directly on the accuracy of the sensors. *The more accurate the sensors, the narrower the peaks.*

Relative vs. Global Sensors

An additional challenge comes from the fact that most sensors we use in robotics are *relative sensors* rather than *global sensors*. Global sensors give a direct (but usually noisy) reading of the state. One obvious example is localization with the

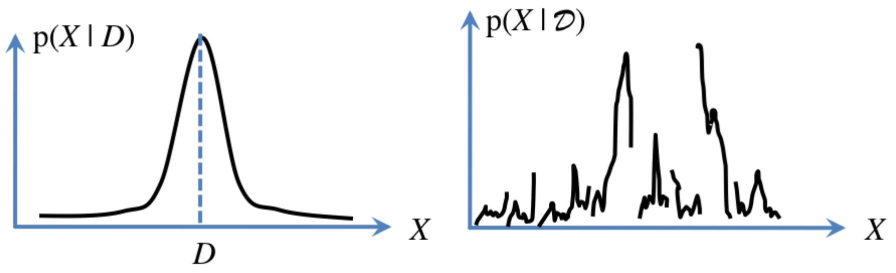


Figure 1.4: Beliefs for relative (right) vs. global (left) sensors. Note the higher complexity for relative sensors.

Global Positioning System (*GPS*), which gives us coordinates on a map. In contrast, relative sensors give readings that depend on the state, but do not directly give the state itself. In most situations, vision, laser range finders, and tactile sensors are relative sensors. For example, a laser range finder is a relative sensor, when used to determine a robot's location on a map. Similarly, vision and tactile are relative sensors, when trying to localize an object.

So why are relative sensors more challenging than global ones? The reason is simple, with a global sensor we immediately know where the peak of the belief distribution is located (see Fig. 1.4). However, if we use a relative sensor, then we do not know where the peak is. In fact, there are likely to be multiple peaks. Hence, we need to search the state space to find the peaks, which makes the problem challenging especially when the peaks are very narrow compared to the width of the state space. The more accurate the (relative) sensor, the more difficult it is to estimate the belief because the peaks become narrower and therefore more difficult to find. This leads to a situation non-obvious at first: *more accurate sensors make perception more difficult.*

Sensor Accuracy vs. Estimation Efficiency

Due to high roughness of the beliefs, parametric methods (such as Kalman filters) are not well suited for inference in many robotics problems. Therefore, we are forced to rely on point-wise estimation methods, where the function is estimated by its values at a set of points. These points are either deterministically placed (as in grids or histogram filters), or randomly sampled from across the state space (as in importance sampling and particle filters). However, all of these methods have the same drawback:

they can miss peaks of the belief, thereby producing entirely inaccurate estimates. Of course, the narrower the peaks, the more likely they are to be missed during estimation. In order to find the narrower peaks reliably, the point-wise estimation methods need more points. Hence, *efficiency of these methods actually falls with increased sensor accuracy.* In the extreme, for a perfectly accurate sensor, these methods completely break mathematically.

On the other hand, it is not like we want noisier sensors! More accurate sensors provide more information than less accurate sensors. Hence, the estimation difficulties are not the fault of the sensors, but rather the estimation algorithms. Moreover, robots in real life applications need to make the most of their existing sensors, because each additional sensor costs money, consumes energy, requires additional computational power, and increases overall system complexity. From the point of view of perceptual inference, this means that we need to devise optimal algorithms, which extract as much information as possible within the time and resource constraints.

1.1.3 Goals of this Work

To summarize the above discussion, our goal is to develop perceptual inference algorithms that:

(a) extract as much information as possible,
(b) can cope with high roughness of robotics beliefs,
(c) work for relative sensors,
(d) can handle arbitrarily accurate sensors,
(e) do not require us to compromise models.

1.2 Related Work

In this section, we give an overview of perceptual inference methods used in robotics. Application specific related work is discussed in later chapters. We shall differentiate robotics problems along two axes (Fig. 1.5): dimensionality and roughness. Dimensionality of a problem is the number of parameters that need to be estimated. For

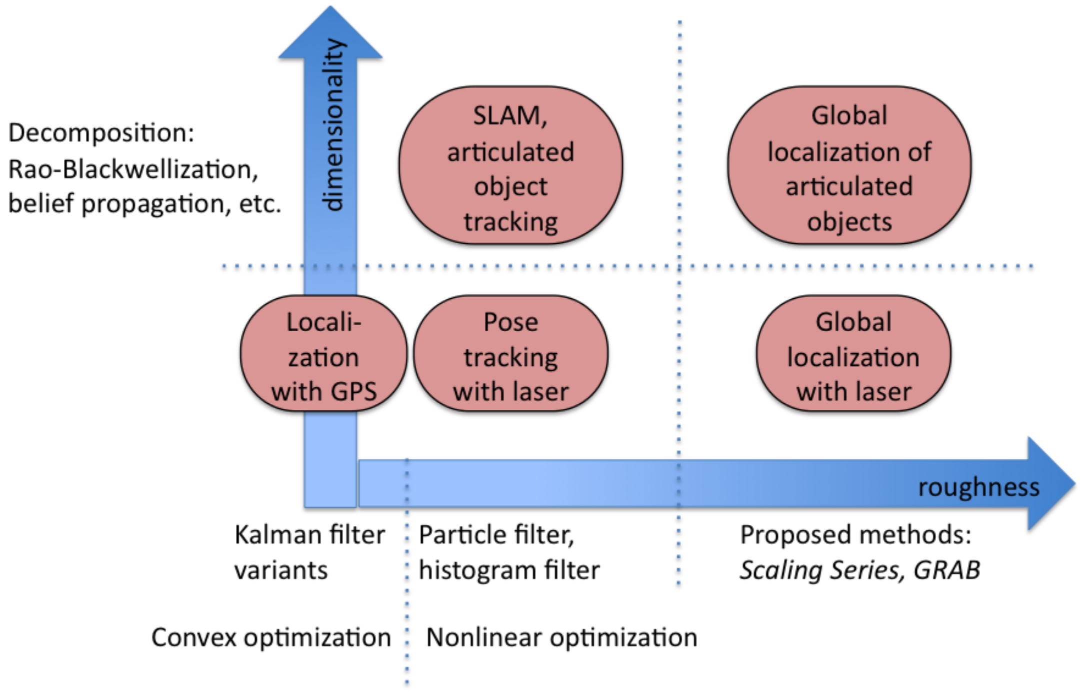


Figure 1.5: Classification of perceptual inference methods in robotics.

example, localizing a rigid object in 3D space requires estimation of 6 parameters: 3 for position and 3 for orientation. Roughness measures how rough the resulting belief is for a given problem. For problems in which the location of the main peak is known a priori, we will assume that roughness is zero. This tends to be the case for state estimation with global sensors, e.g., localization with a GPS. When the main peak location is unknown, we can compute roughness as the ratio of initial uncertainty width to the main peak width. Below we describe inference methods based on where the particular robotics problem fits into in the dimensionality/roughness classification. We illustrate each category of problems with examples.

1.2.1 Parametric Inference Methods

State estimation based on global sensors can typically be solved using *parametric inference methods*, e.g., *Kalman filter* (*KF*) or *extended Kalman filter* (*EKF*). A good example is localization based on GPS sensor [Agrawal and Konolige, 2006, Rezaei and Sengupta, 2007]. In these problems, the location of the main peak of the belief is known based on the sensor readings. Thus, the problem is reduced to approximating the peak with a parametric function. The main limitation of these methods is that they can not represent arbitrary functions (and in particular rough functions).

1.2.2 Non-Parametric Inference Methods

Non-parametric methods approximate the belief by its value at a number of points. In *grid based methods*, these points are placed deterministically. In *Monte Carlo methods*, these points are placed randomly and usually called *particles*. Examples of deterministic methods are *grids* and *histogram filters* [Thrun et al., 2005, Hsiao et al., 2010]. Examples of Monte Carlo methods are *particle filters*, *importance sampling*, and *Markov Chain Monte Carlo* (*MCMC*) [Fishman, 1996, Doucet and De Freitas, 2001]. Although these methods can in principle represent arbitrary functions, the number of points required for accurate representation depends on roughness of the function. Hence, these methods perform reasonably well for low-to-medium roughness problems, e.g., robot pose tracking with a relative sensor (laser or vision) or refinement of a rigid object pose based on tactile sensors (with initial pose estimate already available). However, as roughness increases these methods perform poorly.

1.2.3 Techniques for High Dimensional Problems

Many robotics problems are relatively low dimensional because positioning a rigid object in space requires at most 6 parameters. However, high dimensional problems arise when dealing with deformable, articulated or multiple objects, or whenever other properties need to be estimated (e.g., object shape). Since computational complexity of non-parametric methods goes up exponentially with dimensionality (and polynomially for parametric methods), usually high dimensional problems are solved by decomposing them into several lower dimensional sub-problems. A variety of decomposition techniques exist, including *Rao-Blackwellization* [Doucet et al., 2000, Murphy and Russell, 2001], *belief propagation* [Yedidia et al., 2003], *partitioned sampling* [MacCormick and Blake, 2000], and direct splitting into sub-problems. Rao-Blackwellization is customary in *SLAM* problems [Montemerlo, 2003, Grisetti et al., 2007]. Whereas the other decomposition techniques are commonly used in articulated object tracking [Sudderth et al., 2003, MacCormick and Isard, 2000]. Direct splitting is common in multi-target tracking applications [Dellaert and Thorpe, 1998, Dietmayer et al., 2001, Leonard et al., 2008].

1.2.4 Optimization Techniques

In situations where only the most likely state needs to be estimated, the goal is to estimate the location of the main peak. In this case optimization search techniques are used. Linear least squares problems can be solved in closed form, e.g., by use of *singular value decomposition* (*SVD*) [Yeung et al., 2002]. For non-linear but convex problems, *convex optimization* methods can be used [Boyd and Vandenberghe, 2004]. However, as roughness increases, the problems become non-convex, in which case general *non-linear optimization* techniques are used [Chu, 1999, Sidky et al., 2007]. Still, when roughness is very high, the state space is saturated with local optima making application of optimization techniques very difficult.

1.2.5 High Roughness Problems

High roughness problems arise when global state estimation has to be performed based on relative sensors. In this case, the main peak is very hard to find in the large state space. These problems usually have to be solved in order to initialize tracking methods. State estimation methods that are not capable of solving global uncertainty problems are thought of as ”fragile” [Lepetit and Fua, 2005], because they are unable to recover from localization failures. However, due to high roughness of global uncertainty problems, most state-of-the-art methods do not perform well. Current methods for coping with high roughness include (a) limiting initial uncertainty and (b) smoothing the belief. Method (a) means limiting problems to tracking rather than global localization. The initialization has to be done manually or semi-manually, e.g., in indoor localization, the initial robot pose may be set by hand and, in human posture tracking, the human is asked to assume a specific pose before the tracking can begin. Method (b) means that information is being discarded by smoothing out the sharp transitions in the belief. A number of methods have been developed from simplistic — e.g., sub-sample the data [Thrun et al., 2005]— to more sophisticated — e.g., using Gaussian processes [Plagemann, 2008]. However, since any type of smoothing discards information, the amount of smoothing that can be applied is limited and leads to increased ambiguity and lower accuracy.

1.3 Contributions and Thesis Organization

In this thesis, we propose two perceptual inference algorithms for dealing with high roughness problems: *Scaling Series* and *GRAB*. We illustrate these algorithms on global localization of objects by touch, high accuracy global indoor localization for mobile manipulation, and vehicle detection and tracking for autonomous driving. We also consider the problem of whole body contact estimation. In addition, we propose and evaluate application specific models for each of the applications.

1.3.1 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2 introduces the Scaling Series algorithm and illustrates it on the example of global localization of objects by touch. The approach relies on geometric object models and proposes an efficient model of the measurement process.

Chapter 3 covers estimation of whole-body contacts for robots that do not possess sensory skin, introduces environment and robot models suitable for this application, and solves simple cases using optimization search.

Chapter 4 covers vehicle detection and tracking for autonomous driving. This is a high dimensional problem, which is solved via a combination of direct splitting, Rao-Blackwellization, and Scaling Series.

Chapter 5 deals with high accuracy indoor localization for mobile manipulation.

It is illustrated with operation of door handles and doors for autonomous building navigation. The approach relies on Scaling Series and introduces a combined articulated model of objects and environment. A hybrid representation is used to combine high resolution and low resolution regions.

Chapter 6 introduces a guaranteed inference algorithm, GRAB, which is particularly well suited for discontinuous beliefs. It is illustrated on tactile object manipulation and high accuracy indoor localization and compared with Scaling Series.

Chapter 7 concludes with a summary, discussion of applications, and thoughts about future directions of research.

1.3.2 Publications

The research comprising this thesis has been presented at a number of international conferences and workshops as well as published in international journals. The relevant publications are listed below grouped by application.

Touch Based Perception

- Petrovskaya, A. and O. Khatib (2011). Global localization of objects via touch. *IEEE Transactions on Robotics 27*(3), (forthcoming).

- Petrovskaya, A., J. Park, and O. Khatib (2007, April). Probabilistic estimation of whole body contacts for multi-contact robot control. In *IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, pp. 568–573.

- Petrovskaya, A., O. Khatib, S. Thrun, and A.Y. Ng (2007, June). Touch based perception for object manipulation. In *Robotics: Science and Systems (RSS), Robot Manipulation Workshop*, Atlanta, GA, USA.

- Petrovskaya, A., O. Khatib, S. Thrun, and A.Y. Ng (2006, May). Bayesian estimation for autonomous object manipulation based on tactile sensors. In *IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, Florida, USA, pp. 707–714.

Vehicle Tracking

- Petrovskaya, A. and S. Thrun (2009a, April). Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots 26*(2), 123–139.
- Petrovskaya, A. and S. Thrun (2009b, May). Model based vehicle tracking in urban environments. In *IEEE International Conference on Robotics and Automation (ICRA), Workshop on Safe Navigation*, Volume 1, Kobe, Japan, pp. 1–8.
- Montemerlo, M., J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Hähnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun (2008). Junior: The stanford entry in the urban challenge. *Journal of Field Robotics 25*(9), 569–597.
- Petrovskaya, A. and S. Thrun (2008a, July). Efficient techniques for dynamic vehicle detection. In *International Symposium on Experimental Robotics (ISER)*, Athens, Greece.
- Petrovskaya, A. and S. Thrun (2008b, June). Model based vehicle tracking for autonomous driving in urban environments. In *Robotics: Science and Systems (RSS)*, Zurich, Switzerland.

Mobile Manipulation

- Petrovskaya, A., S. Thrun, D. Koller, and O. Khatib (2010a, June). Guaranteed Inference for Global State Estimation in Human Environments. In *Robotics: Science and Systems (RSS), Mobile Manipulation Workshop*, Zargoza, Spain.
- Petrovskaya, A., S. Thrun, D. Koller, and O. Khatib (2010b, June). Towards dependable perception: Guaranteed inference for global localization. In *Dependable Robots in Human Environments (DRHE), 7th IARP Workshop*, Toulouse, France.

- Petrovskaya, A. and A.Y. Ng (2007, January). Probabilistic mobile manipulation in dynamic environments, with application to opening doors. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India.

1.4 Summary of Notation

Generic notation

Notation	Descripton
$\mathcal{R}(\cdot)$	radius
$Vol(\cdot)$	volume
$d(\cdot,\cdot)$	Euclidean distance metric or generic distance metric
$d_{\mathcal{M}}(\cdot,\cdot)$	Mahalonobis distance
$const$	a constant
$\mathcal{N}(\mu,\sigma^2)$	normal distribution

Bayesian notation

bel	Bayesian belief distribution (posterior)
$\overline{bel}$	prior belief distribution
π	data probability
$\mathcal{D}$	data set
D_k	data point
K	number of data points in data set
t	time step
$\mathcal{D}_{1:t}$ or $\mathcal{D}^t$	data for t steps
$dimX$	dimensionality of state space
$dimD$	dimensionality of measurement space
u	measurement error
ϕ	energy potential
η	normalization constant
Z	partition function

Scaling Series and GRAB

Notation	Descripton
V_0, V_n	uncertainty region (initial and at iteration n)
n, N	index and number of iterations (refinements)
δ_*	desired final radius of δ-neighborhood for Scaling Series or grid resolution for GRAB
$\mathcal{S}_\delta$	δ-neighborhood
M	number of particles per δ-neighborhood
τ	temperature
r, s	relaxation and strengthening
$\mathcal{X}$	particle set
$w, \mathcal{W}$	particle weight, set of particle weights
$zoom$	zoom factor
r_i	δ-neighborhood shape parameters
$\lambda_\pi, \lambda_{u,x}$	Lipschitz constants
ξ	pruning parameter, ξ-mode
G, G_a, G_c	grid cells
$\mathcal{G}, \mathcal{G}^n_{prune}, \mathcal{G}^n_{keep}$	collections of grid cells
$m_k(G), M_k(G)$	min and max range for a grid cell
L^n_i, U^n_i	lower and upper bounds on π
$\widehat{\pi}, \widehat{bel}$	approximations of distributions
$\varepsilon, \varepsilon_{prune}, \varepsilon_{keep}$	errors
ϱ	peak to uncertainty ratio

Whole body contacts

$\mathbf{q}$, $\mathbf{q}^T$	joint angles, joint angles from time 1 to time T
$\mathbf{s}$	environment parameters
$\mathbf{r}$	robot parameters
$dist(\mathbf{r}, \mathbf{s} \vert \mathbf{q})$	distance between environment and robot in configuration $\mathbf{q}$
bel_{geo}, bel_{env}	robot geometry and environment beliefs

Tactile localization

Notation	Descripton
$X = (x, y, z, \alpha, \beta, \gamma)$	object pose
$\mathcal{O}, \hat{\mathcal{O}}$	object and its representation in measurement space
$\hat{o} = (\hat{o}^{\text{pos}}, \hat{o}^{\text{nor}})$	object point in measurement space consisting of position and normal components
$D = (D^{\text{pos}}, D^{\text{nor}})$	tactile measurement consisting of position and normal components
$\sigma_{\text{pos}}, \sigma_{\text{nor}}, \sigma_{\text{met}}, \sigma_{\text{ang}}$	variance
$r_{\text{pos}}, r_{\text{ori}}$	δ-neighborhood shape parameters

Vehicle tracking

$X = (x, y, \theta)$	vehicle pose
v	vehicle forward velocity
$\Omega = (W, L, C_x, C_y)$	vehicle geometry
c_{occ}, c_b, c_s, c_p	measurement model costs
ρ_{min}, ρ_{max}	min and max range measurements
d_{free}	width of free space region
d_{sur}	width of vehicle surface region
$R_t, \bar{R}_t$	vehicle motion belief, motion prediction distribution
S_t	vehicle geometry belief
$\mathcal{X}_t$	collection of particles

Mobile manipulation

$\mathcal{M}$	map of a building
$X = (x, y, \theta)$	robot pose
$D_k := (\rho_k, \alpha_k)$	range measurement consisting of range and bearing components
ctl_t	robot control
Ω	door opening angle (or object parameters)
$\zeta, A_0, \ldots, A_N$	opacity, transition points
R_t, S_t	robot trajectory belief and object state belief

Chapter 2

Tactile Perception

2.1 Introduction

In order to carry out manipulation tasks in real world environments, robots need to perceive objects around them based on sensory information. Although for robots the use of vision has been studied in more depth [Kragic and Christensen, 2002], humans rely heavily on the sense of touch for manipulation tasks [Flanagan et al., 2006]. In fact humans are capable of manipulating objects based solely on the sense of touch. Working towards this ability in robots, we consider global localization of solid objects via touch (Fig. 2.1). Gaining this ability would allow robots to operate in environments where vision is not available, such as smoke filled rooms or muddy water, or it could be used in combination with vision to improve overall perception.

Early tactile perception algorithms date back to the 1980s (e.g., Grimson and Lozano-Perez [1983], Faugeras and Hebert [1983], Shekhar et al. [1986]) as we discuss in the next section. Recent work has focused on tactile perception in uncertain environments. However, in uncertain conditions object localization requires the estimation of a probability distribution over the space of all 6DOF[(1)] poses of the object. When initial uncertainty is high, this estimation is very expensive computationally. For this reason, most approaches limit the DOFs and/or initial uncertainty [Gadeyne and Bruyninckx, 2001, Chhatpar and Branicky, 2005, Hsiao et al., 2010].

[(1)]6DOF stands for six degrees of freedom.

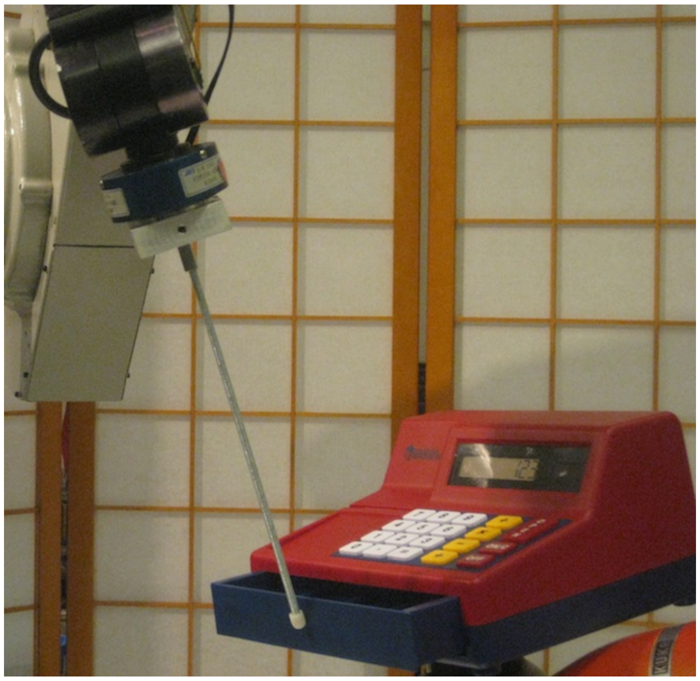


Figure 2.1: In our experiments robots manipulate objects based solely on the sense of touch. Global initial uncertainty is assumed in 6DOF. The photo shows the robot interacting with one of the five objects used in our experiments: the cash register.

To overcome the computational challenge, we propose a principled approach — termed *Scaling Series* (*SS*) — that solves the full global 6DOF localization problem efficiently ($\sim$ 1 second) and reliably ($\geq$ 99%). The approach is a Bayesian Monte Carlo technique coupled with annealing. It performs multiple iterations over the data, gradually scaling the precision from low to high. For each iteration, the number of particles is selected automatically based on the complexity of the annealed belief.

We show that Scaling Series works in both fully-constrained uni-modal scenarios and under-constrained multi-modal scenarios. The latter arise at early stages of tactile exploration, when insufficient data have been collected to fully constrain the problem. We also consider free-standing objects, which can move during tactile exploration. To our knowledge, full 6DOF Bayesian estimation for this case has not been addressed in prior art.

In addition, we present an analytical measurement model for tactile perception that can be used for any object represented as a polygonal mesh. Unlike sampling based models, this model can be computed quickly at run time and does not require training ahead of time. Due to its differentiability, the presented model allows for efficient estimation.

Our approach is easily applicable to any object represented as a polygonal mesh. We demonstrate its portability on five common rigid objects (Fig. 2.2). High initial uncertainty is assumed in the experiments: 400mm in position with unrestricted orientation. The presented approach produces highly accurate results ($\sim$ 1mm) quickly and reliably, enabling the robots to safely manipulate the objects. We also provide extensive empirical evaluation of Scaling Series properties and provide comparisons to other methods, including particle filters, importance sampling, and APF.

The chapter is organized as follows. The next section discusses related work. Sect. 2.3 provides the necessary mathematical background. Sect. 2.4 presents the algorithm together with its discussion and analysis. Experimental results are presented in Sect. 2.5. We conclude in Sect. 2.6. Mathematical derivations are provided in the Appendix.

2.2 Related Work

Touch based perception has not been studied in as much depth as vision because standardized touch sensors are not as easily available. In many situations tactile sensors have to be hand crafted specifically for the robot and the task. This complicates comparisons between methods and slows progress in tactile perception. However, recently there has been a surge of interest in the field due to the necessity of touch based perception in service applications [Kemp et al., 2007, Jain and Kemp, 2009, Hsiao et al., 2010, Prats et al., 2010].

2.2.1 Single Hypothesis Methods

Early methods for tactile object localization generally ignore the sensing process uncertainties and focus on finding a single hypothesis that best fits the measurements. For example, Gaston and Lozano-Perez [1983] used interpretation trees to efficiently find the best match for 3DOF object localization. Grimson and Lozano-Perez [1983] extended the approach to 6DOF. Faugeras and Hebert [1983] used least squares to perform geometrical matching between primitive surfaces. Shekhar et al. [1986] solved systems of weighted linear equations to localize an object held in a robotic hand.

Single hypothesis methods are also widely used to solve the *workpiece localization* problem in manufacturing applications for dimensional inspection [Yau and Menq, 1992], machining [Gunnarsson and Prinz, 1987], and robotic assembly [Gunnarsson, 1987]. In these applications, the measurements are taken by a *coordinate measurement machine* (*CMM*) [Pahk and Ahn, 1996] or by on-machine sensors [Cho and Seo, 2002]. Workpiece localization makes a number of restrictive assumptions, which make it inapplicable to autonomous robot operation in unstructured environments. One important restriction is that there is a known correspondence between each measured data point and a point or patch on the object surface (called *home point* or *home surface* respectively) [Xiong, 2002]. In semi-automated settings the correspondence assumption is satisfied by having a human direct the robot to specific locations on the object. In fully-automated settings the object is placed on the measurement table with low uncertainty to make sure each data point lands near the corresponding home point.

Further restrictions include assumptions that the data are sufficient to fully constrain the object, the object does not move, and there are no unmodeled effects (e.g., vibration, deformation, or temperature variation). All of these parameters are carefully controlled for in the structured manufacturing environments.

The workpiece localization problem is usually solved in least squares form using iterative optimization methods, including the Hong-Tan method [Hong and Tan, 1993], the Variational method [Horn, 1987], and the Menq method [Menq et al., 1992]. Since these methods are prone to getting trapped in local minima, low initial uncertainty is usually assumed to make sure the optimization algorithm is initialized near the solution. Some attempts have been made to solve the global localization problem by re-running the optimization algorithm multiple times from pre-specified and random initial points [Chu, 1999]. Recent work has focused on careful selection of the home points to improve localization results [Xiong et al., 2004, Huang and Qian, 2008, Zhu et al., 2009] and on improving localization efficiency for complex home surfaces [Zhu et al., 2004, Sun et al., 2009].

2.2.2 Bayesian Methods in Tactile Perception

In the last decade, there has been increased interest in Bayesian state estimation for the tactile object localization problem [Gadeyne and Bruyninckx, 2001, Chhatpar and Branicky, 2005, Corcoran and Platt, 2010, Hsiao et al., 2010]. These methods estimate the probability distribution over all possible states (the belief), which captures the uncertainty resulting from noisy sensors, inaccurate object models, and other effects present during the sensing process. Thus, estimation of the belief enables planning algorithms that are resilient to the uncertainties of the real world. Unlike workpiece localization, these methods do not assume known correspondences. In contrast to single hypothesis methods, belief estimation methods can handle the under-constrained scenario, in which the data are insufficient to fully localize the object. These methods can also work with moving objects and answer important questions, such as: "have we localized the object completely?" and "where is the best place to sense next?".

The main challenge faced by belief estimation approaches is computational complexity, which goes up exponentially with the number of DOFs and the size of the initial uncertainty region. For this reason, all of the approaches in this category (except our work) restrict the number of DOFs and/or initial uncertainty.

The earliest known work in this category was Gadeyne and Bruyninckx [2001], who considered localization of a rectangular box based on measurements taken by a force controlled robot. The localization was performed in 3DOF with initial uncertainty of 300mm in position and 360° in orientation. They used a sampled measurement model that was stored in a look-up table.

Chhatpar and Branicky [2005] used particle filters for contact based object localization during peg-in-hole assembly tasks. They considered 20mm initial uncertainty in 3DOF and utilized a measurement model based on sampling the object in advance. Chhatpar and Branicky also considered active localization, where the most optimal next sensing action is chosen based on information from prior steps.

The earliest version of our work was published in 2006 [Petrovskaya et al., 2006]. We considered 6DOF localization with large uncertainty: 400mm in position and 360° in orientation. We also introduced an analytical measurement model and proposed the Scaling Series method.

Corcoran and Platt [2010] used the annealed particle filter (APF) to estimate 4DOF pose and radius of cylindrical objects. Initial uncertainty of up to 250mm in position with unrestricted orientation was considered. They also extended the analytical measurement model we proposed in 2006 to include some negative information and to integrate over object surface. Later in 2010, Platt Jr et al. [2010] introduced sample based models suitable for localization of deformable objects.

Most recently, Hsiao et al. [2010] used grids to estimate the belief in 3DOF with low-to-medium initial uncertainty (up to 50mm Gaussian). The contribution of their approach was in optimizing data collection strategies and considering free standing objects that could potentially move during data collection. The measurement model used in their work is similar to the one we proposed in 2006, except that it also takes negative information into account.

We should also mention the rich literature on object shape reconstruction using tactile sensors [Charlebois et al., 1996, Bicchi et al., 1999, Kaneko and Tsuji, 2001, Moll, 2002]. Although this work does not address localization of known objects, some authors explicitly consider sensor uncertainties using Bayesian methods [Slaets et al., 2004, Schaeffer and Okamura, 2003].

2.2.3 Bayesian Methods in Other Applications

Bayesian methods have been used in a variety of robotic applications with great success. For example a recent book on practical applications includes analysis of planetary ring structure, shape estimation, and target tracking to name a few [Doucet and De Freitas, 2001]. A recent textbook by Thrun et al. [2005] provides an in-depth study of indoor robot localization and mapping, which bear some resemblance to the problems considered in this chapter. However, the global localization problems considered in the textbook are relatively low dimensional: 3DOF. The only high dimensional problem considered in the textbook is simultaneous localization and mapping (SLAM), where global uncertainty does not need to be resolved. Moreover, many SLAM methods effectively reduce dimensionality by utilizing problem structure [Montemerlo, 2003]. These techniques do not apply to the 6DOF object state estimation problem, where this structure is not present.

There has been a lot of work on 6DOF object localization in the vision community. See [Lepetit and Fua, 2005] for a recent survey. The most popular methods have been least-squares minimization [Harris, 1993, Drummond and Cipolla, 2002], RANSAC [Fischler and Bolles, 1981], Kalman filter variants [Rahimi and Darrell, 2002, Davison, 2003], and particle filters [Isard and Blake, 1998]. These approaches tend to rely on manual initialization and assume small initial uncertainty. As Lepetit and Fua point out, methods incapable of dealing with global uncertainty tend to be inherently fragile because they can not recover from tracking failures.

One of the most successful variants of particle filters, the annealed particle filter (APF), has been introduced by Deutscher et al. in the context of articulated body tracking using vision [Deutscher et al., 2000, Deutscher and Reid, 2005]. As we already mentioned above in Sect. 2.2.2, this method has also been applied to the tactile localization problem [Corcoran and Platt, 2010]. Articulated object tracking is a very high dimensional problem (up to 30DOF). However, usually low initial uncertainty is assumed in these applications, due to the use of manual initialization. Also these approaches do not run in real time. APF tends to outperform the standard particle filter in single-mode scenarios. However, it has been shown to be unstable in multi-modal situations by Balan et al. [2005]. In fact Balan et al. argue for the use of standard particle filters instead of APF for this very reason.

2.3 Mathematical Background

We start out with a quick intuitive summary of the problem: tactile object localization requires estimation of state parameters based on a set of data obtained by touching the object. As we shall see in Sect. 2.3.3 this entails fitting the data to the object model using Mahalonobis distance in the 6D measurement space. In the case of moving objects, the estimation is performed via recursive filtering from one time step to the next.

Instead of producing a single set of parameter values, Bayesian approaches represent the uncertain knowledge by a probability distribution, which records how likely each state is based on sensor measurements. Estimating the entire probability distribution over all the states is important because initially the data are insufficient to disambiguate the object's position. In fact, the shape of the probability distribution (specifically the high likelihood regions, called *modes*) allows us to determine when enough data has been collected in order to manipulate the object safely. The probability distribution is represented numerically by weighted points, called *particles*.(2)

In the remainder of this section we formalize the above intuitive description and introduce the required notation.

2.3.1 Bayesian Problem Statement and Definitions

We consider the class of problems where the state X has to be inferred from a set of sensor measurements $\mathcal{D} = \{D_k\}$. Our goal is to estimate the probability distribution of the state given the measurements, $bel(X) := p(X|\mathcal{D})$, known as the *posterior distribution*, which represents our uncertain *belief* about the state X.

For the general algorithm, we will assume that the state X is a vector of dimensionality $dimX$ in the *state space* $\mathbb{R}^{dimX}$. The measurements are modeled as K random variables D_k, which are drawn independently from *conditional probability distributions* $p(D_k|X)$ with domains in the *measurement space* $\mathbb{R}^{dimD}$. The conditional probability distributions (*CPDs*) encode the *measurement model*, which is a probabilistic law that represents the *measurement process*. The measurement model depends non-linearly on the state X. In many applications, the CPDs are naturally given in the log-linear form via *measurement energy potentials* $\phi_k : \mathbb{R}^{dimX} \times \mathbb{R}^{dimD} \mapsto \mathbb{R}^{+}$. Then the CPD for D_k can be written as

$$p(D_k|X) = \eta \exp\Big(-\phi_k(X, D_k)\Big). \tag{2.1}$$

(2)See [MacKay, 1998] for further information on particle based Bayesian methods.

In the above equation and throughout the chapter η denotes the normalization constant, whose value is such that the expression integrates to 1. We also define the *total measurement energy*

$$\phi(X) := \sum_k \phi_k(X, D_k). \tag{2.2}$$

Via Bayes rule the belief $bel(X)$ can be shown to be proportional to $p(\mathcal{D}|X)p(X)$. The first factor is the *data probability*, which can be shown to be proportionate to $\pi(X) := \exp(-\phi(X))$. The second factor, $\overline{bel}(X) := p(X)$, is called the *prior*, which represents our belief about X before obtaining measurements $\mathcal{D}$. Hence with this notation we can write

$$bel(X) = \eta\pi(X)\overline{bel}(X). \tag{2.3}$$

Stationary systems

In stationary systems with global initial uncertainty the prior $\overline{bel}(X)$ is uniform. Hence, the belief is proportional to the data probability: $bel(X) = \eta\pi(X)$.

Dynamic systems

In dynamic systems the state changes over time. In this case, X_t and $\mathcal{D}_t$ denote the state and the set of sensor measurements for a time step t. The belief, bel_t, is defined as the probability of the current state given all measurements obtained up until this point:

$$bel_t\left(X_t\right) := p\left(X_t|\mathcal{D}_1, \cdots, \mathcal{D}_t\right). \tag{2.4}$$

Measurement CPDs for step t are defined analogously to (2.1). Similarly define $\pi_t(X_t) := \exp(-\phi_t(X_t))$. Also let $\overline{bel}_t(X_t)$ be the prior at time t. For brevity we will drop the argument X_t and write bel_t, π_t, and $\overline{bel}_t$ to denote the values of these functions at time t.

In dynamic systems, the prior is the prediction distribution,

$$\overline{bel}_t := p\left(X_t|\mathcal{D}_1, \cdots, \mathcal{D}_{t-1}\right), \tag{2.5}$$

which predicts the current state X_t before taking into account the most recent sensor

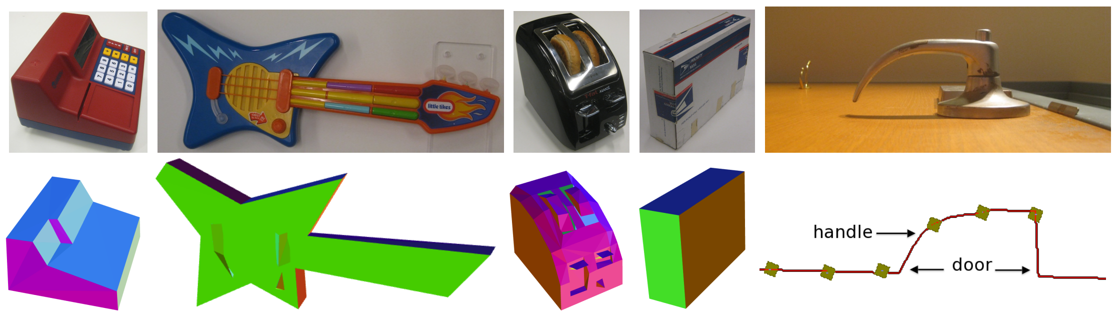


Figure 2.2: The five objects used in our experiments: cash register, toy guitar, toaster, box, and door handle. Bottom row shows polygonal mesh models of the objects. Model complexity ranges from 6 faces (for the box) to over 100 faces (for the toaster).

data $\mathcal{D}_t$. Hence the prior is computed as

$$\overline{bel}_t = \int p\left(X_t | X_{t-1}\right) bel_{t-1} \, \mathrm{d}X_{t-1}. \tag{2.6}$$

Here $p\left(X_t | X_{t-1}\right)$ encodes the dynamics of the system. This probability is called the *motion model*. Combining (2.3) and (2.6) we obtain the *Bayesian recursion* equation:

$$bel_t = \eta \, \pi_t \int p\left(X_t | X_{t-1}\right) bel_{t-1} \, \mathrm{d}X_{t-1}. \tag{2.7}$$

2.3.2 Problem Statement for Tactile Localization

Bayesian tactile localization is an instance of the general Bayesian problem defined in the previous section. Here the robot needs to determine the pose X of a known object $\mathcal{O}$ based on a set of tactile measurements $\mathcal{D}$. The object is typically represented as a polygonal mesh (Fig. 2.2). The state $X := (x, y, z, \alpha, \beta, \gamma)$ is the 6DOF pose of the object — including position (x, y, z) and orientation angles (α, β, γ) — in the manipulator coordinate frame. The measurements $\mathcal{D}$ are obtained by touching the object with the robot's end effector. Each measurement $D_k := (D_k^{\mathrm{pos}}, D_k^{\mathrm{nor}})$ consists of the measured cartesian position of the contact point D_k^{pos} and the measured surface normal D_k^{nor}.

Note that unlike in the workpiece localization problem, here we do not assume known correspondence between measurements and points on the surface of the object. Hence the resulting problem is more complex than workpiece localization.

2.3.3 Measurement Model

To interpret the tactile measurements, we use the *proximity measurement model*, which has been used in stereo vision by Olson [2000] and is known as *likelihood fields* in mobile robotics [Thrun et al., 2005]. In this model, the measurements are considered independent of each other with both position and normal components corrupted by Gaussian noise. For each measurement, the potential depends on the distance between the measurement and the object (hence the name "proximity").

Since the measurements contain both contact coordinates and surface normals, this distance is taken in the 6D space of coordinates and normals (i.e., in the measurement space). Let $\hat{\mathcal{O}}$ be a representation of the object in this 6D space. Let $\hat{o} := (\hat{o}^{\text{pos}}, \hat{o}^{\text{nor}})$ be a point on the object surface, and D be a measurement. Define $d_{\mathcal{M}}(\hat{o}, D)$ to be the Mahalonobis distance between $\hat{o}$ and D:

$$d_{\mathcal{M}}(\hat{o}, D) := \sqrt{\frac{||\hat{o}^{\text{pos}} - D^{\text{pos}}||^2}{\sigma^2_{\text{pos}}} + \frac{||\hat{o}^{\text{nor}} - D^{\text{nor}}||^2}{\sigma^2_{\text{nor}}}}, \tag{2.8}$$

where σ^2_{pos} and σ^2_{nor} are Gaussian noise variances of position and normal measurement components respectively. Then the distance between a measurement D and the object is $d_{\mathcal{M}}(\hat{\mathcal{O}}, D) := \min_{\hat{o}} d_{\mathcal{M}}(\hat{o}, D)$.

Let $\hat{\mathcal{O}}_X$ denote the object in state X. For a measurement D_k, define the *measurement error* to be

$$u_k(X) := d_{\mathcal{M}}(\hat{\mathcal{O}}_X, D_k). \tag{2.9}$$

Then the measurement potential is computed as

$$\phi_k(X, D_k) := \frac{1}{2} u_k^2(X). \tag{2.10}$$

Similarly to total measurement energy, we also define the *total measurement error* to be

$$u(X) := \sqrt{\sum_k d^2_{\mathcal{M}}(\hat{\mathcal{O}}_X, D_k)}. \tag{2.11}$$

Then, we can re-write π as

$$\pi(X) = \exp\left(-\frac{1}{2}u^2(X)\right). \tag{2.12}$$

While early Bayesian tactile localization work used sampled measurement models [Gadeyne and Bruyninckx, 2001, Chhatpar and Branicky, 2005], the model described here is analytical. Hence it can be computed efficiently on the fly and without the need for prior training. As all proximity models, the model assumes that the closest point on the object caused the measurement. This is often referred to as a *hard assignment* meaning that the point causing the measurement is assigned to be the closest point. Alternatively with a *soft assignment*, one considers the contribution from all points to the probability of the measurement. Although the soft assignment model has been used for tactile object localization by Corcoran and Platt [2010], we specifically chose to use the hard assignment model for two reasons. First, the hard assignment model can be efficiently computed explicitly unlike the soft assignment model. Second, for an unbiased application of the soft assignment model, one needs to compute a prior over all surface points, i.e. how likely each surface point is to cause a measurement. However, this prior is usually non-uniform and highly dependent on the object shape, the manipulator shape, and the probing motions.

Like all proximity models, the model described here does not take negative information into account. In other words it does not incorporate information that the robot was able to move through some parts of space without making contact with the object. Negative information has been taken into account by Hsiao et al. [2010] and Corcoran and Platt [2010]. However, incorporation of negative information leads to more complex measurement models and complicates inference. The proposed model is continuous and almost everywhere differentiable. Both of these properties would be lost with incorporation of negative information. Although we did not see a significant impact of negative information on accuracy and reliability of localization, it can be useful for active exploration strategies as in [Hsiao et al., 2010]. In these cases, the negative information can be superimposed on top of the belief computed using the proximity model.

2.3.4 Motion Model

Since free standing objects can move during probing, we need to define a motion model for this dynamic process. We assume the state of the object evolves via addition of Gaussian noise. Hence, $p(X_t|X_{t-1})$ is a Gaussian with mean at X_{t-1} and variances σ^2_{met} and σ^2_{ang} along metric and angular axes respectively.

2.4 Inference Algorithm

We start by introducing the required concepts and providing an intuitive description of the algorithm. A formal description is given in Sect. 2.4.2. Sects. 2.4.3 through 2.4.6 provide detailed analysis of the algorithm's features and properties.

2.4.1 Concepts and Intuition

As we have seen in Sect. 2.2 prior approaches have struggled to solve the full 6DOF object localization problem with global uncertainty. The main challenge is computational complexity, which is proportional to the number of particles used. As we will see below, the number of particles required to solve the problem reliably is exponential in the dimensionality of the problem.

Required number of particles

As an example consider a 1D space [0,1]. We want to find the peak of the belief by sampling particles from the space randomly[3] (see Fig. 2.3 top left). When we sample a particle from the entire space, the probability of it hitting the support of the peak is equal to the ratio between the width of the peak and the width of the entire space. Let's denote this ratio by $1/\varrho$. Hence in expectation we need to sample ϱ particles from the entire space in order to get a particle from the support of the peak.

The same is true for higher dimensional problems: *the ratio between the width of the peak and the width of the initial uncertainty dictates the necessary number of*

[3]More precisely: we want to sample the particles uniformly and independently.

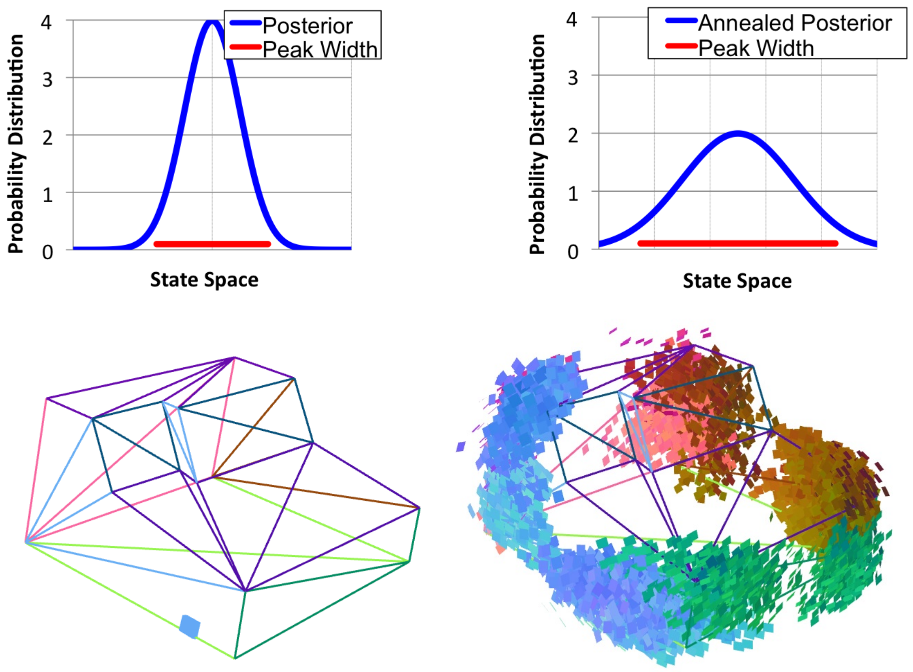


Figure 2.3: Top row: two plots of a simple belief over $[0, 1]$. Top left: true belief. Top right: annealed belief. Note that annealing increases peak width, and therefore improves the ratio of peak width to space width. Bottom row: true (left) and annealed (right) belief for localization of cash register. The cash register model is shown as a wire frame. The small colored squares represent high likelihood particles. Note that annealing makes the problem more ambiguous.

particles required for reliable state estimation.[4]

Unfortunately ϱ goes up exponentially with problem dimensionality. For the 3DOF tactile object localization with 400mm initial uncertainty[5] and sensor accuracy of 1mm, ϱ comes out to be around 6×10^6, whereas for the 6DOF problem it is approximately 3×10^{15}. To put the exponential blowup in perspective, if we assume that the 3DOF problem takes 1 second to solve, then the 6DOF problem would take approximately 1.5 years.

Thinking in terms of peak width also helps understand the following surprising fact about belief estimation: *the problem actually becomes harder with more accurate sensors*. The reason is simple, more accurate sensors produce more narrow peaks, and therefore ϱ increases. In the extreme, when the sensors are perfectly accurate, most Bayesian methods break mathematically.

Smoothing

In order to improve the peak width to uncertainty ratio, many modern methods utilize *smoothing* (also known as *relaxation*) [Thrun et al., 2005]. Smoothing broadens the peaks (Fig. 2.3 top right), and therefore reduces the number of particles required to find it reliably. One of the most common smoothing techniques is *annealing*, which is obtained by exponentiating the measurement model to the power $1/\tau$, where τ is the temperature. Thus for $\tau = 1$ the true measurement model is obtained and for $\tau > 1$ the measurement model is "heated-up". The higher the temperature, τ, the broader the peaks. However, annealing (and any other type of smoothing) comes at a price. The estimates become less accurate and the state estimation becomes more ambiguous (Fig. 2.3 bottom row). Intuitively smoothing is analogous to blurred vision: the more blurry the vision, the harder it is to determine an object's position or to disambiguate objects.

[4]Of course, in 2D the term "width" should be replaced by "area", and in 3D and higher by "volume" of the supporting regions.

[5]Unrestricted orientation uncertainty is assumed.

Broad particles

Estimation of a belief by particles would be impossible without some sort of local smoothness. Indeed, if the value of the belief at one point was completely unrelated to its value at the neighboring points, then no number of particles would be sufficient for accurate estimation. Most particle based methods do not make this assumption explicit and define each particle as a single point. However, we use *broad particles*, which represent regions of space around them. We will call them *δ-neighborhoods*(6), where δ is the radius(7) of the neighborhood. Of course the value of δ depends directly on the smoothness of the belief: the smoother the belief the larger the δ. “Heating-up” the measurement model increases δ as it makes the belief smoother. Thus δ depends on the temperature during annealing.

Intuitive algorithm description

The main idea is to have the whole uncertainty region covered with δ-neighborhoods. This way we are sure that we have a good approximation of the belief. At high temperature this can be easily done with just a few particles because δ is large. Of course this will not produce accurate estimates, so we use an *iterative refinement* approach. First we solve the problem with a few very broad particles at high temperature. Prune out the low probability regions and keep the peaks. Then refine the estimates at a lower temperature. Prune again and repeat until the temperature reaches $\tau = 1$. This way the final estimates will be as accurate as the data and the model allow.

Both the uncertainty region and the peak width change during refinements. The uncertainty region changes due to pruning. The peak width changes due to annealing. Therefore the ratio of peak width to uncertainty width also changes. Hence no single fixed number of particles will work well for all refinement stages. Instead of using a fixed number of particles, we specify the desired *particle density* by setting the number of particles to maintain per δ-neighborhood. This way the algorithm can compute the appropriate number of particles to use at each refinement stage.

(6)In earlier versions of the work, δ-neighborhoods were called δ-spheres.

(7)Radius of a region can be defined as half the diameter, where the diameter is the largest distance between two points contained in the region.

2.4.2 The Scaling Series Algorithm

The goal of the algorithm is to compute an approximation of the belief *bel* by weighted particles. The initial uncertainty is assumed to be uniform over the starting region. In this case, the belief is proportional to the data probability (see Sect. 2.3.1). Hence the weights can be computed via π.

The formal algorithm listing is given in Alg. 2.1. The algorithm takes as input the initial uncertainty region, V_0, the data set, $\mathcal{D}$, and two user-specified parameters: M and δ_*. M specifies the number of particles to maintain per δ-neighborhood. δ_* specifies the terminal value of δ. The refinements stop once the algorithm reaches this value. Selection of appropriate values for the two user-specified parameters is discussed in Sect. 2.4.3.

Lines $1-3$ set initial values. δ_0 is selected so that one δ_0-neighborhood contains the entire initial uncertainty region. The scaling factor *zoom* is set so that the volume of each δ-neighborhood is halved during scaling. The number of iterations N is computed based on the ratio of initial to final volume. $\mathcal{S}_\delta$ denotes a δ-neighborhood, $\mathcal{R}(\cdot)$ denotes the radius and $Vol(\cdot)$ denotes the volume of a region.

The initialization is followed by a loop that performs the refinement iterations in lines 4 – 11. At each iteration n, δ_n is computed by applying the scaling factor to δ_{n-1}. The corresponding temperature, τ_n, is computed based on the assumption that δ_* corresponds to the temperature of $\tau = 1$. Line 7 draws a particle set $\bar{\mathcal{X}}_n$ uniformly from V_{n-1} ensuring the required density of M particles per δ-neighborhood. A listing of this procedure is provided in Alg. 2.2. In line 8 Compute_Normalized_Weights procedure weighs the particles by the annealed data probability, $\pi(X)^{1/\tau_n}$, at temperature τ_n. This procedure also normalizes the weights so that they add up to 1. Line 9 prunes low probability regions. A detailed discussion of this step is provided in Sect. 2.4.3. Line 10 computes the resulting subregion V_n for this iteration. After completion of the refinement steps, lines 12 and 13 draw the final particle set and compute weights at temperature $\tau = 1$.

The algorithm returns an approximation of the belief represented by a weighted particle set $\mathcal{X}$, where the weights $\mathcal{W}$ are set to the data probability at temperature $\tau = 1$.

Alg. 2.1 Scaling Series algorithm for belief estimation.

```
Input: $V_0$ - initial uncertainty region, $\mathcal{D}$ - data set, $M$ - number of particles per
    $\delta$-neighborhood, $\delta_*$ - terminal value of $\delta$.
1: $\delta_0 \leftarrow \mathcal{R}(V_0)$
2: $zoom \leftarrow 2^{-1/dimX}$
3: $N \leftarrow log_2(Vol(\mathcal{S}_{\delta_0})/Vol(\mathcal{S}_{\delta_*}))$
4: for $n = 1$ to $N$ do
5:     $\delta_n \leftarrow zoom \cdot \delta_{n-1}$
6:     $\tau_n \leftarrow (\delta_n/\delta_*)^2$
7:     $\bar{\mathcal{X}}_n \leftarrow$ Even_Density_Cover$(V_{t-1}, M)$
8:     $\mathcal{W}_n \leftarrow$ Compute_Normalized_Weights$(\bar{\mathcal{X}}_n, \tau_n, \mathcal{D})$
9:     $\mathcal{X}_n \leftarrow$ Prune$(\bar{\mathcal{X}}_n)$
10:    $V_n \leftarrow$ Union_Delta_Neighborhoods$(\mathcal{X}_n, \delta_n)$
11: end for
12: $\mathcal{X} \leftarrow$ Even_Density_Cover$(V_N, M)$
13: $\mathcal{W} \leftarrow$ Compute_Normalized_Weights$(\mathcal{X}, 1, \mathcal{D})$
Output: $(\mathcal{X}, \mathcal{W})$ - a weighted particle set approximating the belief.
```

For line 7, we need a procedure to sample uniformly from V_{n-1}, which is represented as a union of δ-neighborhoods. During sampling we need to ensure that we draw M particles from each δ-neighborhood. Thus in effect this is very similar to stratified sampling, except the sets comprising V_{n-1} are not necessarily disjoint. One of the simplest implementations is based on rejection sampling (Alg. 2.2)[8].

2.4.3 Discussion of Algorithm Features and Settings

Even density cover

Although this is one of the most crucial features of Scaling Series, at first it may seem counter-intuitive to call the Even Density Cover procedure (line 7 of Alg. 2.1). Indeed the particle set comprising V_{n-1} is already weighted by the annealed data probability. Why not perform a weighted resample? The weights already resemble the belief distribution, so why should we discard them and sample particles uniformly instead?

[8] A historical note: the original implementation of this step was more complicated. The use of rejection sampling for this purpose was proposed by an anonymous reviewer at ICRA 2006.

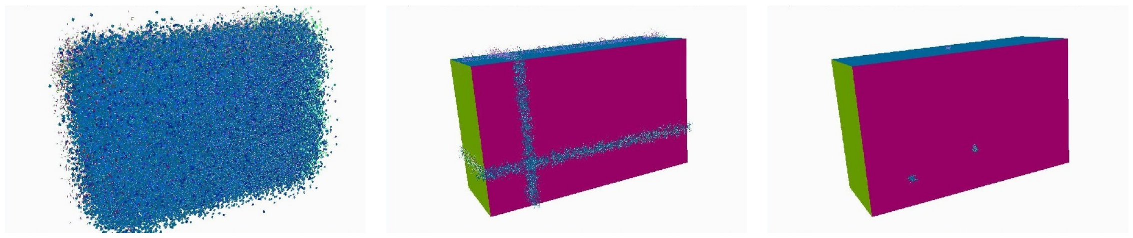
Alg. 2.2 Even Density Cover: procedure for uniform sampling from a region represented as a union of δ-neighborhoods with density M per δ-neighborhood.

```
Input: $V$ - sampling region represented as a union of $\delta$-neighborhoods $\{\mathcal{S}_i\}$, $M$ - number of particles to sample per $\delta$-neighborhood.
1: $\mathcal{X} \leftarrow \{\}$
2: for $i = 1$ to $|\{\mathcal{S}_i\}|$ do
3:     for $m = 1$ to $M$ do
4:         sample a point $X$ from $\mathcal{S}_i$
5:         reject $X$ if it is in $\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_{i-1}$
6:         otherwise add $X$ to $\mathcal{X}$
7:     end for
8: end for
Output: $\mathcal{X}$ - a set of particles that evenly cover $V$.
```

It turns out this step is critical for reliable handling of multi-modal beliefs. This is easiest to understand by considering a simple example. Suppose we have a belief with two modes of even height. We draw two particles: one near each mode. If one of the particles is slightly closer to a mode than the other, the weights will be uneven. Hence during weighted resampling we will favor one mode over the other. If we perform several iterations, this error compounds and hence we are quite likely to discard one of the two modes. Even Density Cover avoids this problem. If a particle survived the pruning step, it will be given full consideration at the next iteration.

The multi-modal case is important for two reasons. First, multi-modal beliefs arise naturally during tactile object exploration because at early stages the number of measurements is insufficient to determine the object's location unambiguously. In fact the belief can even have entire regions of high probability (see Fig. 2.4). Estimating the multi-modal belief at early stages of exploration is important for making safe and informed decisions about future sensing actions. Second, note that most iterations of Alg. 2.1 compute the annealed belief. The higher the temperature the more ambiguous the belief becomes (see bottom row of Fig. 2.3). Hence multiple modes are often present during early iterations of Scaling Series as we show in Sect. 2.5.5 experimentally.

One other important reason for the Even Density Cover step is that without it



(a) after 1 measurement (b) after 2 measurements (c) after 3 measurements

Figure 2.4: During exploration the belief evolves as additional measurements arrive. The particles in this figure approximate high likelihood regions of the evolving belief. Each particle is shown by a small square at the hypothesized position of the first data point on the surface of the object. The normal of each square corresponds to the sensed normal transformed to the object coordinate frame based on the hypothesized object pose.

we would be double-counting the data and hence the estimate would not converge to the true belief.

Pruning

The purpose of the pruning step (line 9 of Alg. 2.1) is to remove low probability regions from consideration. This way the computational resources can be focused on the more interesting high probability regions. This step removes particles with relatively low weights from the particle set. This is achieved via weighted resampling. See [Arulampalam et al., 2002] for a listing of a weighted resampling algorithm. During this step the value of M is ignored. Instead this procedure draws the same number of particles as there were prior to this step. The weights are set to be uniform after the resampling operation. Although weighted resampling is likely to discard low probability particles, from theoretical viewpoint the resulting particle set encodes the same probability distribution as the weighted particle set prior to resampling.

Selecting δ_*

The value for δ_* should be selected so that the belief changes only a small amount within a δ-neighborhood of any particle. This can be done using the *Lipschitz* *constant*.[9] For the global localization case, the belief is proportional to $\pi(X) = \exp(-\frac{1}{2}u^2(X))$ (see (2.12)), so we set

$$\delta_* := \frac{1}{\lambda_\pi}, \tag{2.13}$$

where λ_π is the Lipschitz constant of π. It can be easily shown that λ_π is bounded by $\lambda_u/\sqrt{e}$ (see Appendix A), and so (2.13) relates δ_* to λ_u. Thinking in terms of u gives δ_* a physical meaning: it is the largest radius, within which the total measurement error can change by at most $\sqrt{e}$. Lipschitz constant computations for the measurement model described in Sect. 2.3.3 are provided in Appendix A.[10]

The measurement model described in Sect. 2.3.3 is continuous with bounded derivatives almost everywhere. Thus it is guaranteed to have a Lipschitz constant. However, for some measurement models the Lipschitz constant may not exist or be cumbersome to compute. In these cases, one can set the value of δ_* to a good guess, which works well in most areas of state space. Increasing the value of M will help compensate for an imperfect setting of δ_*, as these two parameters complement each other.

Shape of δ-neighborhood

So far we have not specified what shape a δ-neighborhood takes. In early versions of our work [Petrovskaya et al., 2006], we termed the neighborhoods δ-spheres and defined them to be hyper-spheres of radius δ. However, we also mentioned that when coordinates are not homogenous (e.g. position vs. orientation), scaling factors may be needed. Hence, the obtained shape is actually a hyper-ellipsoid. The scaling factors can have a significant impact on performance. Analogously to using the Lipschitz constant of π, the neighborhood dimensions along each axis can be set based on the partial Lipschitz constants of π, which are defined as the maximum partial derivatives.

[9] For a function $f(X)$, the *Lipschitz constant*, λ_f, is defined to be the maximum slope between any two points.

[10] These derivations provide upper bounds that hold for all objects and data sets. Although these values are not necessarily optimal for a specific object and data set, they serve as a good guide and can be further optimized experimentally (see Sect. 2.5.5).

If $\lambda_{\pi,i} := \sup|\frac{\partial \pi}{\partial x_i}|$, then we set the radius of the neighborhood along i-th axis to be

$$r_i := 1/\lambda_{\pi,i}. \tag{2.14}$$

In this case we assume that $\delta_* := r_1$ to avoid ambiguity. See Appendix A for a derivation of the partial Lipschitz constants for the model described in Sect. 2.3.3.

Annealing schedule

During iterations we compute the annealed data probability

$$\pi^{1/\tau}(X) = \exp(-u^2(X)/\tau), \tag{2.15}$$

and so τ acts on $u^2(X)$. Since δ_* is proportional to change in $u(X)$, τ should be adjusted in proportion to δ^2 rather than linearly with δ. This computation takes place in line 6 of Alg. 2.1.

Selecting M

The number of particles to maintain per δ-neighborhood is a user-specified parameter, which affects reliability, efficiency, and accuracy. As we already mentioned it complements the value of δ_*. The higher the value of M, the higher the accuracy and reliability, and also the higher the computational cost. In practice, if δ_* is chosen as described above, M values between 3 and 6 tend to give good results, although in rare cases M can be set as low as 2. A higher value of M is needed if the Lipschitz constants have been underestimated or if these constants do not exist. An empirical evaluation of dependence on M is provided in Sect. 2.5.5.

Comparison with APF

At a first glance Scaling Series may seem very similar to APF, which also uses iterative annealing. However there are three important distinctions. First, while APF has a fixed number of particles to use at each iteration, Scaling Series selects the number of particles automatically and dynamically for each refinement stage. The selection takes into account the smoothness of the belief, the total uncertainty volume, and the width of the neighborhood each particle can represent. Thus, the optimal number of particles is used at each iteration for efficient and accurate representation.

Second, while APF is known to handle poorly in multi-modal scenarios [Balan et al., 2005], Scaling Series handles these very well due to the use of Even Density Cover. For this reason APF does not converge to the true belief, whereas Scaling Series does as we show in Sect. 2.4.6.

Third, the Scaling Series annealing schedule is derived from the mathematical properties of the belief. This allows for much more efficient and straightforward annealing than APF, which relies on survival rate. Scaling Series also derives the relationship between temperature and δ, which is analogous to the APF diffusion rate. In APF, the diffusion rate is disassociated from the temperature, which can lead to non-optimal diffusion. Empirical comparison to APF is provided in Sects. 2.5.5, 2.5.5, and 2.5.4.

2.4.4 Algorithm Variations

Zoom factor

The standard version of Scaling Series algorithm sets *zoom*, so that the volume of a δ-neighborhood is halved at each iteration. However, it is possible to zoom faster or slower, reducing the volume for example to 10% or 90% each time. Note that if *zoom* factor is changed, the number of iterations also needs to be changed in line 3, where the base of the log is the factor, by which the volume is reduced per iteration. Faster zooming will require fewer iterations, slower will require more. Empirical evaluation in Sect. 2.5.5 shows that *zoom* factor of the original algorithm is optimal.

Alternative pruning strategies

One alternative strategy for pruning is thresholding based on a preset percentage of the top weight in the particle set. Unlike weighted resampling, thresholding can be carried out based on log of the weights (i.e. directly on ϕ). This can significantly improve numerical stability in situations where the data does not match the model very well — a common scenario in the presence of unmodeled effects.[11] A threshold of ξ corresponds to $\ln\xi$ in terms of ϕ. Thus any particle whose ϕ exceeds the minimum ϕ in the particle set by more than $\ln\xi$ can be pruned. A reasonable choice is to prune out everything that is further than one standard deviation away from the solution. Since π is Gaussian in u, this results in $\xi = \frac{1}{\sqrt{e}} \approx 60\%$. Empirical evaluation in Sect. 2.5.5 shows that this is indeed the optimal setting.

Time limit

One practical approach is to limit the amount of time allotted for estimation based on a single data set. This is especially helpful at early stages of exploration, when the belief is highly ambiguous (Fig. 2.4). See Sect. 2.5.2 for an example.

Compensating for object symmetries

Many man-made objects have symmetries that can not be resolved no matter how much data is collected. These objects always produce multi-modal beliefs. In order to reduce the number of particles to represent the modes, a simple strategy is to take each state X modulo the symmetries.

2.4.5 Tracking Dynamic Objects

So far we have only considered estimation of beliefs with a uniform prior. This works well for stationary objects. However, free standing objects can shift during tactile probing. Hence we need a method for tracking the state of dynamic objects. In these cases the prior is not uniform as it encodes the information from prior sensing actions and possible motions of the object. Hence we need a way to extend Scaling Series to tracking of dynamic objects.

First let us consider how a standard particle filter (PF) solves this problem (see [Thrun et al., 2005] for details). At each time step t, PF performs a motion update

[11]Although thresholding in log space does not change the mathematical outcome of the operation in principle, in practice when the probability of the data is extremely low, the weights come out to be zero due to limited floating point exponent range.

followed by a measurement update. The motion update performs a resample followed by application of the motion model. The measurement update incorporates the most recent data by setting importance weights proportional to π_t. Note, that the measurement update is similar to Scaling Series, except the prior is non-uniform in this case.

We consider three possible ways of extending Scaling Series to the tracking problem. The first algorithm, SS-DYN1, simply runs Scaling Series during the first time step (when the prior is uniform), and then follows by standard particle filter updates for the rest of the time steps.

The second algorithm, SS-DYN2, is the same as SS-DYN1, except that it uses Scaling Series during each measurement update. To do so, it uses the particle set generated by the motion update of the previous step and sets δ_0 broad enough to encompass motion noise. Of course, this does not fully take the prior into account, so we end up "forgetting" some information from prior time steps.

The third algorithm, SS-DYN3, runs Scaling Series on each data set using a uniform prior, and then adjusts the weights to capture the motion model via the Bayesian recursion equation (2.7). This way it does not "forget" any information from prior steps. Formal listing of SS-DYN3 is provided in Alg. 2.3. The algorithm takes as input the belief from the prior time step represented as a set of weighted particles. The rest of the parameters are analogous to Alg. 2.1. In line 1 the algorithm approximates π_t with a set of weighted particles using Scaling Series (Alg. 2.1). Lines 4 – 6 compute the integral that appears in (2.7). Line 7 multiplies the weights by the integral. The weights are then normalized in line 9. The algorithm outputs the resulting weighted particle set, which approximates the belief at time step t.

Note that due to efficiency of Scaling Series, SS-DYN3 algorithm is tractable as is. However, two efficiency improvements can be implemented. First, if the prior state is too far away from the proposed current state, the probability of the object transitioning from one state to the other is very low. Thus the contribution of this term to the integral in line 5 is negligible. Hence, the loop in lines 4 – 6 can be restricted to particles X_{t-1} that are close enough to X_t.

Second, we can initialize V_t to the high probability regions of the prior $\overline{bel}_t$. In

Alg. 2.3 SS-DYN3: algorithm for tracking a dynamic state with Scaling Series.

Input: $(\mathcal{X}_{t-1}, \mathcal{W}_{t-1})$ - weighted particle set from prior time step, V_t - initial uncertainty region, $\mathcal{D}_t$ - data set for time step t, M - number of particles per δ-neighborhood, δ_* - terminal value of δ.

```
1: $(\mathcal{X}_t, \mathcal{W}_t) \leftarrow$ Scaling_Series$(V_t, \mathcal{D}_t, M, \delta_*)$
2: for each $(X_t, w_t) \in (\mathcal{X}_t, \mathcal{W}_t)$ do
3:     $s \leftarrow 0$
4:     for each $(X_{t-1}, w_{t-1}) \in (\mathcal{X}_{t-1}, \mathcal{W}_{t-1})$ do
5:         $s \leftarrow s + p(X_t|X_{t-1})w_{t-1}$
6:     end for
7:     $w_t \leftarrow w_t s$
8: end for
9: normalize weights $\mathcal{W}_t$
```

Output: $(\mathcal{X}_t, \mathcal{W}_t)$ - a weighted particle set approximating the belief at time step t.

other words, we can focus on areas, where the object is likely to move based on prior information. We can compute the prior using the motion update step of standard particle filters. Hence we perform a weighed resampling from $(\mathcal{X}_{t-1}, \mathcal{W}_{t-1})$ followed by application of the motion model with randomly sampled noise parameters. The result is an unweighted particle set representing $\overline{bel}_t$. Then V_t can be set to the union of δ-neighborhoods centered at the obtained particles, where δ should be set broad enough to accommodate for the error due to having a finite number of particles. In line 1 of Scaling Series, δ_0 should be set to the value of δ used for V_t. Note that this efficiency improvement does not double-count the prior, due to the Even Density Cover step at the beginning of Scaling Series.

2.4.6 Algorithm Analysis

In this section we analyze convergence of the proposed algorithms. In short, we show that Scaling Series, SS-DYN1 and SS-DYN3 converge to the true belief. However, SS-DYN2 does not converge.

Scaling Series estimates of the belief converge as M tends to ∞. The convergence is understood in the same sense as for importance sampling. Namely, we want to estimate the expected value $\mathbb{E}_{bel}[f]$ of some function of interest $f(X)$ with respect to the belief distribution. Let the estimate produced by importance sampling be denoted by

$$\mathrm{IS}_J(f) := \sum_j f(X_j) w_j, \tag{2.16}$$

where X_j are particles and w_j are normalized importance weights. Then we know that $\mathrm{IS}_J(f) \to \mathbb{E}_{bel}[f]$ *almost surely* (a.s.) as $J \to \infty$ [Geweke, 1989].

Similarly, let the estimate produced by Scaling Series with M particles per δ-neighborhood be denoted by

$$\mathrm{SS}_M(f) := \sum_j f(X_j) w_j, \tag{2.17}$$

where X_j are particles in the final set $\mathcal{X}$ with normalized weights w_j. Then analogously to importance sampling we have the following convergence result.

Theorem 2.1. $\mathrm{SS}_M \to \mathbb{E}_{bel}[f]$ *a.s. as* $M \to \infty$.

Proof. Let us consider the first iteration of Scaling Series (Alg. 2.1). Particles in $\bar{\mathcal{X}}_1$ a.s. completely cover V_0 as $M \to \infty$, and so particles in $\mathcal{X}_1$ also a.s. completely[12] cover V_0. The same reasoning can be applied to all N iterations. Hence, V_N a.s. completely covers V_0 because N does not depend on M. When V_N covers V_0, lines 12 & 13 of Scaling Series are equivalent to importance sampling with a uniform prior and with $J > M$ particles. Thus by convergence of importance sampling we get the desired convergence result for Scaling Series. □

Similarly we can derive convergence of SS-DYN1 and SS-DYN3 from the convergence of particle filters (PF) [Gordon, 1993]. However, SS-DYN2 does not converge to the true belief because the Even Density Cover step after propagating the particles discards some information from the prior, and SS-DYN2 does not compensate for this information loss. This is similar to the behavior of APF, which also does not converge to the true belief due to information loss caused by annealing. In practice, however, these algorithms can be very useful (as we show in Sect. 2.5.4).

[12] This statement is true as long as $bel(X) > 0$ for all X. Without loss of generality we can assume that this is the case. Otherwise, we can simply exclude from V_0 points at which $bel(X) = 0$ as these points do not contribute to the expectation of f.

Theorem 2.2. SS-DYN1$_M$ *converges a.s. as* $M \to \infty$.

Proof. SS-DYN1 consists of SS followed by PF, so the result follows from their convergence. □

Theorem 2.3. SS-DYN3$_M$ *converges a.s. as* $M \to \infty$.

Proof. In SS-DYN3, line 1 computes π_t and lines 4 – 6 compute the prior $\overline{bel}_t$ using (2.6). Line 7 multiplies the weights by the prior, and hence by Bayesian recursion the resulting weights are proportional to the belief bel_t. □

2.5 Experimental Results

We performed extensive evaluation of Scaling Series with both real and simulated data. Two implementations were used: the old and the new one. The old implementation was in Java running on a 1.2GHz laptop computer. The new implementation is in C++ running on a 2GHz laptop computer.

We constructed polygonal mesh models of five everyday objects: cash register, guitar, toaster, box, and door handle (Fig. 2.2). The mesh models of the first three objects were constructed based on measurements taken with the robot's end effector. Models for the last two objects were constructed from ruler measurements. The accuracy of models ranges from 5mm for the first three objects to 1mm for the last two objects. Accuracy of surface normals is quite poor near edges, corners, and other non-flat parts of the objects.

Each object model included feature points: buttons, levers, grasp points, etc. Once localization is performed, the features are transformed into robot coordinates so that the manipulation scenarios could be carried out. Videos of the experiments, code and other supplemental materials are available on our website [Petrovskaya, 2011b].

The remainder of this section is organized as follows. Sects. 2.5.1, 2.5.2, and 2.5.3 cover real robot experiments with the five objects mentioned above. Sects. 2.5.4 and 2.5.5 cover experiments performed in simulation. Sect. 2.5.4 considers tracking of a

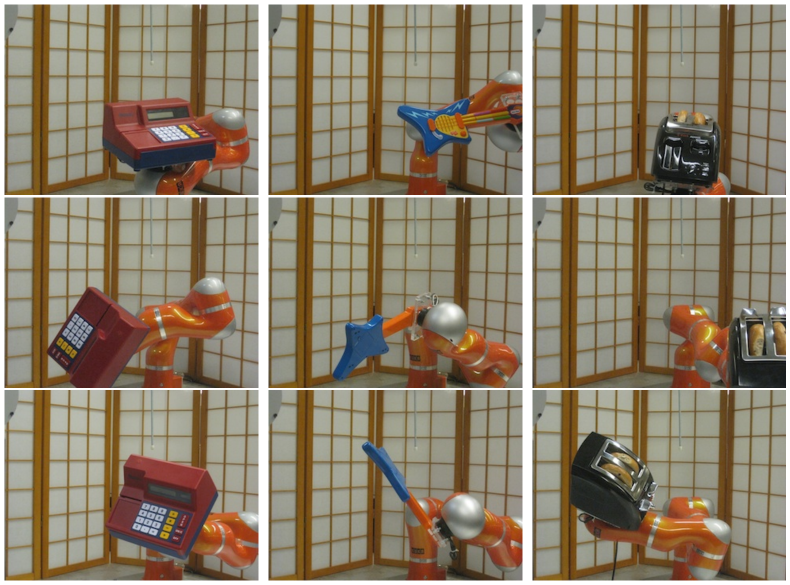


Figure 2.5: The nine poses used during localization experiments (three poses per object). These poses where selected randomly from the uncertainty region. The poses in the top row were also used to run object manipulation scenarios (see videos). All pictures were taken from the same vantage point.

free standing box that moves during tactile exploration. Sect. 2.5.5 provides extensive empirical evaluation of Scaling Series features and parameters.

2.5.1 Experiments with Cash Register, Guitar, and Toaster

In this set of experiments we evaluated the algorithm on three common objects: cash register, guitar, and toaster. The manipulator used was a 6DOF PUMA robot, equipped with a 6D JR3 force/torque sensor at the wrist. In these experiments we used a long end effector of 300mm length and 6mm diameter. Since the initial uncertainty was large, the long end effector was necessary to ensure that the robot always made contact with the tip of the end effector and not some other non-sensing part of the robot. The end effector had a semi-spherical tip of 5mm radius.

The sources of error included: mesh model inaccuracies, object deformation (especially noticeable for the guitar), robot positioning error, end effector deformation (significant due to the long length), and error due to unknown position of the contact on the tip of the end effector. Although it is difficult to determine exact amount of

Table 2.1: Results of the nine experiments with cash register, guitar, and toaster. Cartesian coordinates and errors are listed in millimeters. Orientation angles and errors are listed in degrees. Localization errors are reported with respect to ground truth poses obtained from the Kuka robot. NP denotes the number of probes, ND denotes the number of data points.

		Pose								xyz	$\alpha\beta\gamma$	
	Object	x	y	z	α	β	γ	NP	ND	error	error	δ_*
1	Register	393	542	-285	-60°	22°	16°	24	12	2.9	3.5°	2.4
2	Register	131	635	-350	128°	67°	41°	24	11	6.8	4.3°	2.5
3	Register	364	520	-275	-18°	8°	-33°	26	14	2.1	1.7°	2.2
4	Guitar	468	500	-255	-34°	-32°	-30°	27	10	5.6	2.3°	2.6
5	Guitar	219	528	-335	-166°	-4°	31°	36	11	9.2	5.2°	2.5
6	Guitar	273	678	-186	76°	70°	-68°	60	17	4.8	3.0°	2.0
7	Toaster	380	445	-310	127°	161°	-11°	20	11	4.2	2.4°	2.5
8	Toaster	576	271	-286	-25°	1°	-7°	23	11	6.1	1.2°	2.5
9	Toaster	180	614	-204	85°	101°	39°	22	14	5.5	3.3°	2.2
	Min	131	271	-350	-166°	-32°	-68°	20	10	2.1	1.2°	2.0
	Average	—	—	—	—	—	—	29	12	5.2	3.0°	2.1
	Max	576	678	-186	128°	161°	41°	60	17	9.2	5.2°	2.6

noise produced by all of these errors, we estimated the contact position noise to be roughly $\sigma_{\text{pos}} = 5$mm. Sensed normals were extremely noisy due to polygonal model inaccuracy and long end effector length. We used $\sigma_{\text{nor}} = 50°$. The experiments were carried out using our C++ implementation of Scaling Series with thresholding. We set the threshold to $\xi = 60\%$ and used $M = 6$ particles per neighborhood. The rest of the parameters were set in accordance with the derivations in Sect. 2.4.3 and Appendix A. Specifically, we set δ_*, r_{pos}, and r_{ori} so that:

$$\begin{aligned} \delta_* &= \sigma_{\text{pos}}\sqrt{e/K}, \\ r_{\text{pos}} &= \delta_*, \\ r_{\text{pos}}/r_{\text{ori}} &= \sqrt{\mathcal{R}^2(\mathcal{O}) + \sigma^2_{\text{pos}}/\sigma^2_{\text{nor}}}. \end{aligned} \tag{2.18}$$

The initial uncertainty for all objects was 400mm along x, y, z with unrestricted orientation. We randomly selected nine poses from this uncertainty region: three poses per object (Fig. 2.5). The objects were held in place by a Kuka LWR robot. We used the joint angles of the Kuka robot to generate ground truth for all nine poses.

Prior to experiments we generated a set of safe probing trajectories, which took joint limits and collisions with the environment into account. During experiments, data collection procedure randomly selected probing trajectories from the pre-generated set. All probing trajectories moved the robot along the direction of the end effector, so that the end effector tip was the first part to make contact. Each probe took approximately 10s. The Scaling Series algorithm was run on all data points collected up to that time step. The algorithm was allowed to compute until the next measurement arrived. Once the algorithm determined that the belief had a single mode and all particles were within 10mm of each other, the probing procedure stopped and the mean pose was used as the estimated pose. In experiments 1, 4, and 7, the localization procedure was followed by a manipulation scenario: using the cash register, playing the guitar, and toasting bagels respectively. Videos are available on our website [Petrovskaya, 2011b].

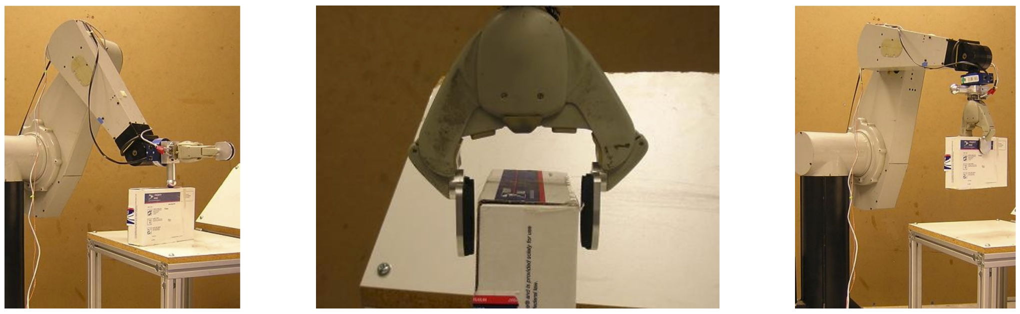
Localization results for the nine experiments are summarized in Tbl. 2.1. Overall, localization was quite accurate: the average localization error was 5.2mm and 3°.

We believe this was an important factor in the success of the manipulation scenarios. Localization was the most accurate for the cash register (3.9mm average error) because its shape consists of planar surfaces that are easy to model accurately. The toaster has many curved surfaces, which are more difficult to model and hence the localization error is slightly higher: 5.3mm on average. Localization was the least accurate for the guitar (6.5mm average error) because this object deformed significantly during probing as can be seen in the videos.

2.5.2 Manipulating a Box

In the second set of experiments, we applied the Java implementation of our approach to the task of localizing, grasping and picking up a rectangular box (see Fig. 2.6). As in the previous set of experiments, we used the PUMA robot with the JR3 sensor. This time the robot's end-effector included a gripper and robotic finger combination, so that the robot could perform both probing and grasping tasks. The finger was much shorter and thicker (75mm length, 25mm diameter) with a spherical end of 15mm radius. This configuration resulted in much more accurate data because the end effector did not deform. Shorter length of the finger also resulted in more accurate measured normals. The rectangular box was 56mm x 159mm x 238mm in size. The size of the mesh model was inflated by the radius of the spherical end-effector, so that the end-effector tip could be reduced to a single point in computations. Due to higher accuracy of the measurements, we set $\sigma_{\text{pos}} = 1\text{mm}$ and $\sigma_{\text{nor}} = 5^\circ$. The rest of the parameters were set as before.

We used the same initial uncertainty region: 400mm in x, y, z with unrestricted orientation. This time the probe was too short to safely explore the large uncertainty region without touching the object with non-sensing surfaces. Therefore for data collection we developed a custom active sensing procedure specific to the box object. Although the procedure restricted the set of poses in which the data collection could be successful, localization was still performed on the full uncertainty region without taking the restrictions into account. The box was fixed using brackets so that it remained relatively stationary during the experiments, although it still shifted and



(a) sensing (b) grasping (c) manipulating

Figure 2.6: The stages of the box manipulation experiment. (a) Sensing the box with a robotic finger. (b) Grasping the box. The position and orientation of the box were estimated from the data obtained during sensing stage. The grasping configuration is defined as part of the box model. Note the precise fit required to perform the grasp. (c) The last stage is manipulation of the box.

deformed during probing as can be seen in the videos.

Fully-constrained case

In fully-constrained experiments, we collected five measurements using the above probing procedure. These five points were used to perform localization of the box using Scaling Series. Two grasp points were manually defined on the box model, each consisting of 3 points: one for each side of the gripper and one for the wrist position. Thus each grasp point fully defined position and orientation of the gripper. After localization, the grasp point with the highest z-coordinate was selected[(13)]. The gripper orientation, position and approach vector were derived from the selected grasp point and estimated parameters. Note the precise fit required for grasping in Fig. 2.6.

We performed 30 trials of fully-constrained experiments on the real robot. The sensing procedure took 30 seconds. Localization was performed in less than 1 second. Out of the 30 trials, the data collection procedure failed in 9 trials[(14)]. These trials were aborted. In all of the remaining 21 trials, the robot successfully localized,

[(13)] z-coordinates increase vertically upwards

[(14)] During these experiments the PUMA robot was experiencing intermittent sudden jolts possibly due to faulty encoders. These jolts resulted in large force measurements registered on the JR3 sensor and hence were interpreted as phantom contact readings. Trials during which these jolts were experienced were aborted.

grasped, and manipulated the box.

Early stages of exploration

To evaluate the algorithm performance at early stages of exploration, we took data sets consisting of 2 - 3 measurements from different sides of the box. These data sets do not fully constrain the problem, and so the modes of the resulting belief form ridges in the state space (Fig. 2.7). For real robot experiments, we took subsets of measurements from our completed real robot trials. We verified that the estimated region included the true state of the object, as it was estimated from complete data sets. We also examined the estimated region visually to make sure it corresponded to the correct solution region in each under-constrained scenario. In addition, we performed 100 simulated trials where ground truth was available. The true state was included in the resulting solution set in all 100 trials.

Since the number of solutions is infinite, high precision settings result in large numbers of particles. However, it is possible to exit out of iterations early based on a time limit setting as discussed in Sect. 2.4.4. For example for a data set consisting of two measurements, Scaling Series generated 4,000 particles for δ =11mm and 29,000 particles for δ =1mm (Fig. 2.7). The running time increases with the number of particles generated. For our Java implementation, operations with a few thousand particles take a few seconds, but 29,000 particles take 40-50 seconds to process. Thus it is possible to trade off precision of estimation for running time. As more measurements arrive, the solution region shrinks and higher precision can be achieved with fewer particles.

2.5.3 Door Handle Operation

In the third set of real robot experiments, we performed door handle manipulation with a mobile manipulator consisting of a Segway platform and a 5DOF Harmonic Arm 6M manipulator (see Fig. 2.8a). Once the robot navigates to the area in front of a door (using its laser sensors for approximate localization), we use tactile feedback to accurately estimate the position and orientation of the door and the door handle.

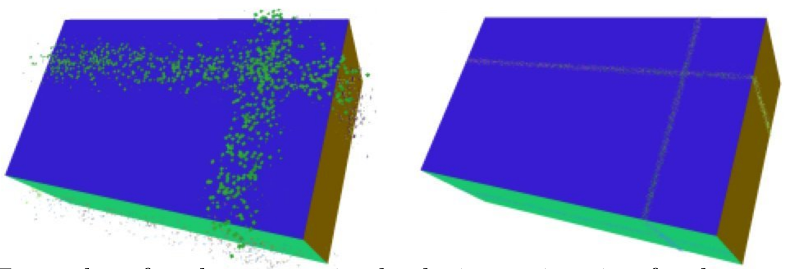
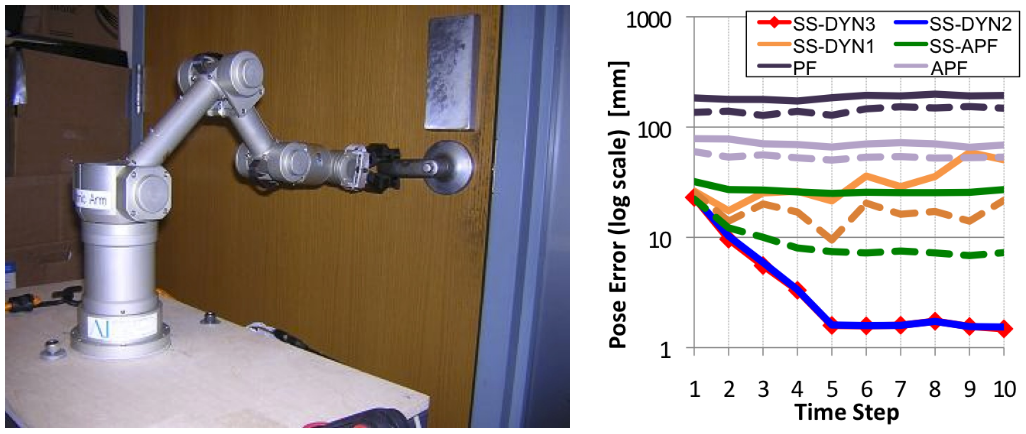


Figure 2.7: Examples of under-constrained solution estimation for data sets consisting of 2 measurements (includes symmetry compensation). Left: With δ =11mm, 4,000 particles were generated by Scaling Series. Right: With δ =1mm 29,000 particles were generated. As before each particle is shown by a square indicating the location of the first data point on object surface. The size of each square is δ.



(a) door handle experiments (b) free standing objects

Figure 2.8: (a) Harmonic Arm robot operating the door handle in one of the experiments. (b) Accuracy of object tracking over 10 time steps starting with global uncertainty. Each algorithm was given 1s of computation time per step. Dashed lines show how the tracking improves if 60s per step are allotted for algorithms of the corresponding color. The results are averaged over 100 runs.

The Harmonic Arm manipulator used in these experiments has about 1mm end-effector positioning precision. Since all door handles in the building are mounted at the same height and always in horizontal position, the height of the handle as well as two orientation angles were fixed, which reduced the localization task to a 3DOF problem. Our algorithm used a 2D model of the door that was constructed by hand using ruler measurements. Specifically, we took door handle depth measurements every 10mm along its length in a horizontal plane through the center of the handle. This resulted in a 2D model consisting of line segments (Fig. 2.2). The grasping point was defined near the tip of the door handle. The sensing used in this experiment gave only position measurements, and did not include surface normals.

For each experimental trial, the robot took 6 measurements in a 30° span (at $0°, 6°, \ldots, 30°$). Each data point thus consisted of range to the contact point and an orientation angle. The sensing procedure took between 1 and 2 minutes. Using these six measurements, our algorithm was able to localize the door and the door handle in a fraction of a second using our Java implementation. In these experiments, we restricted the dimensions of the state space (to 60mm x 60mm x 30°) because of the limited operational range of the manipulator. Out of 100 independent trials, our algorithm successfully completed the sensing in 98 trials. In all of these 98 trials, our algorithm then successfully localized, grasped, and turned the door handle, and opened the door. The two failures during sensing were caused by a hardware glitch in communication with the robot.

2.5.4 Free Standing Objects

When estimating the state of a dynamic system, it is important that the information gained via measurements exceeds the information lost due to noisy motion at each time step. Otherwise the state will only become more uncertain over time making localization impossible. Since in our hardware setup the robot only has one finger, little information is obtained at each time step placing a very tight restriction on the amount of motion allowed. Hence to evaluate tracking of moving objects, we assume that the robot possesses a multi-fingered hand capable of measuring at least three data points per time step[15]. We evaluated this scenario in simulation by sampling three contacts randomly from the surface of the box using the same box model as in Sect. 2.5.2. The object was tracked over ten time steps, starting with global 6DOF uncertainty (400mm in position, 360° in orientation). We simulated measurement noise of $\sigma_{\text{pos}} = 1$mm and $\sigma_{\text{nor}} = 5°$ as well as considerable motion noise: $\sigma_{\text{met}} = 20$mm and $\sigma_{\text{ang}} = 10°$. The rest of the parameters for Scaling Series were set as follows: $\delta_* = 1$mm, $r_{\text{pos}} = 1$mm, $r_{\text{ori}} = 1°$, $M = 6$, $\xi = 60\%$.

Using the C++ implementation, we compared SS-DYN1, SS-DYN2, SS-DYN3, two variants of APF, and PF (Fig. 2.8b). The two APF variants were: APF and SS-APF. The standard APF used 100 layers and survival rate $\alpha = 90\%$ with annealing schedule selected as in [Deutscher et al., 2000]. These settings performed the best for APF. SS-APF used 20 layers and its annealing schedule was selected using Scaling Series methodology. Hence, SS-APF is in between SS and APF algorithms. It uses the same annealing schedule as SS, but like APF it is missing the Even Density Cover step of SS. All algorithms were given 1s of computation time per time step. Dashed lines show how the performance improves with 60s per time step. The results are averaged over 100 runs.

Note, that with three measurements per data set, the belief is multi-modal during the first several time steps. Hence it is not possible to fully localize the object initially. The ambiguity is gradually resolved as additional measurements arrive. Also, note that the average error is to a large extent a function of reliability. In other words whether or not a particular algorithm found the object at each time step. The lower the reliability, the higher the average error. Thus, even if an algorithm has high average error, it may have accurately localized on some of the runs.

PF was unable to locate and/or track the object as the average error is over 140mm even with 60s per update. Still, the average error improves from 1s to 60s, so with more time per update, PF is likely to perform even better. APF converges to 68mm

[15] One possibility is a hand with three fingers, each consisting of three phalanges with a tactile sensor on each phalange. Thus in principle this hand is capable of making nine contacts during a single grasp of the object. If the hand is operated compliantly (either in software or hardware), then it can close around the object without knowing its exact location. As it closes it will make multiple contacts. Since the blind grasp may not be very good, we assume that only three out of possible nine contacts are sensed.

error, which improves to 53mm if 60s per update are allotted. SS-APF ends up with 27mm average error, which improves to 7mm with 60s per update. SS-DYN1 starts off well due to initialization via SS at first time step, however over time SS-DYN1 diverges and approaches APF error. These results are in line with Balan et al. [2005], where APF was compared to well initialized PF.

Both SS-DYN2 and SS-DYN3 performed very well, quickly converging to 1.5mm average error. There was no significant difference in performance of these two algorithms. This is likely because in our case little is known about how the object moves. It is possible that in applications with more informed motion models, SS-DYN3 will show an improvement over SS-DYN2. The difference in performance between SS-DYN3 and SS-APF clearly underscores the importance of the Even Density Cover step for estimation of multi-modal beliefs.

2.5.5 Algorithm Evaluation

In this section we evaluate the impact of Scaling Series features and parameters on performance, as well as compare Scaling Series to other algorithms. These experiments are carried out on simulated data for the box localization problem. The same box model was used as in the real data experiments (Sect. 2.5.2). Unless otherwise noted we used the following settings: target resolution $\delta_* = 1$mm, hyper-ellipsoid neighborhoods with $r_{\text{pos}} = 1$mm and $r_{\text{ori}} = 1°$, $M = 6$ particles per neighborhood, $zoom = 1/\sqrt[6]{2}$, measurement noise $\sigma_{\text{pos}} = 1$mm and $\sigma_{\text{nor}} = 5°$. Fully-constrained data sets (with 5 measurements) were used, unless stated otherwise. Results shown are averages over 100 runs of the algorithms. Most experiments were carried out with our C++ implementation and used Scaling Series with thresholding on $\xi = 60\%$.

In experiments we compare several algorithms, including: Scaling Series (SS), importance sampling (IS), annealed particle filter (APF), and a variant of APF with Scaling Series annealing schedule (SS-APF).

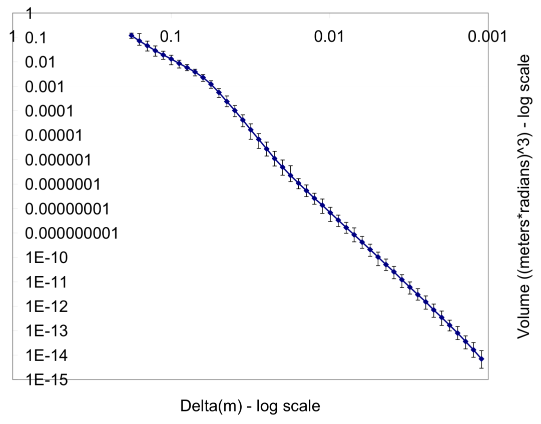
Scaling Series evolution over iterations

First we evaluate how the search space and estimation error change during iterations of Scaling Series (Fig. 2.9). In the plots, the progression of the series is from left to right, with corresponding δ values noted in meters on the horizontal axis (in log scale). The plots clearly show that the volume of the search space shrinks drastically with iterations. At the same time estimation error falls. The number of particles remained small throughout all of the experiments, with the absolute maximum being below 600. The number of particles is highest for δ values between 30 and 100mm. At these settings the distribution is multi-modal, corresponding to 6 possible sides of the box. As these possibilities are ruled out, the number of particles goes down. The multi-modality is particularly noticeable on the orientation error plot (Fig. 2.9d). These experiments used our Java implementation.

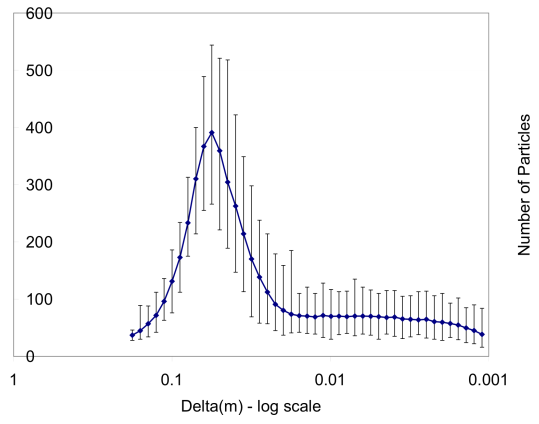
Single mode estimation

In this set of experiments we used fully-constraining data sets (5 measurements from different sides of the box), so that the resulting belief was uni-modal. We compared reliability and accuracy of SS, IS, APF, and SS-APF (Fig. 2.10). For SS-APF, we show performance with 20 layers, which worked the best. For APF, we used 100 layers and survival rate $\alpha = 90\%$, which was optimal. For IS the computation time is controlled by the total number of particles. For SS-APF and APF it is controlled by number of particles per layer. For SS the running time is controlled via setting of M between 3 and 6. Reliability is the percentage of experiments that localized the box successfully, i.e. had at least one particle within 1mm and 1° of the true pose.

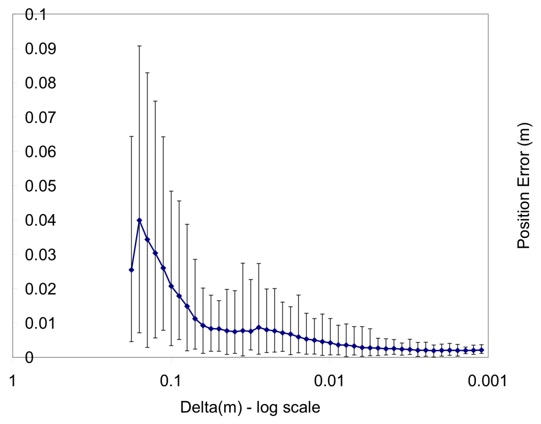
IS was unable to localize the box even after several minutes of computation. SS and SS-APF were able to localize the box within several seconds (0.3s and 5s respectively), with SS being approximately 15 times faster than SS-APF. APF localized the box within 100 seconds, which is approximately 300 times slower than SS and 20 times slower than SS-APF. These comparisons underline the impact of the Even Density Cover step (SS vs. SS-APF) and annealing schedule methodology (SS-APF vs. APF).



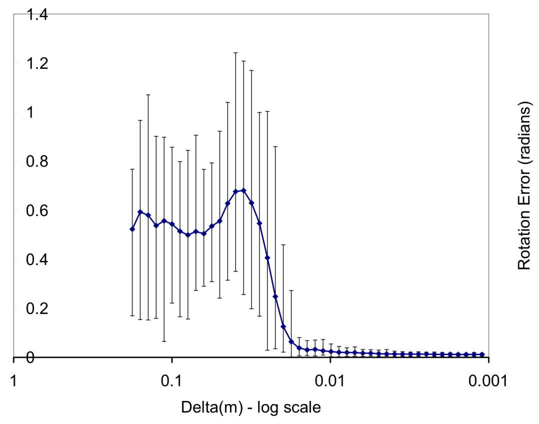
(a) search space volume



(b) number of particles



(c) position error



(d) orientation error

Figure 2.9: Performance of Scaling Series on simulated data set during 100 experiments. Each graph shows progression of the series from left to right. Corresponding value of δ is noted on the horizontal axis in meters, log scale. Vertical bars represent absolute min/max values during all 100 runs.

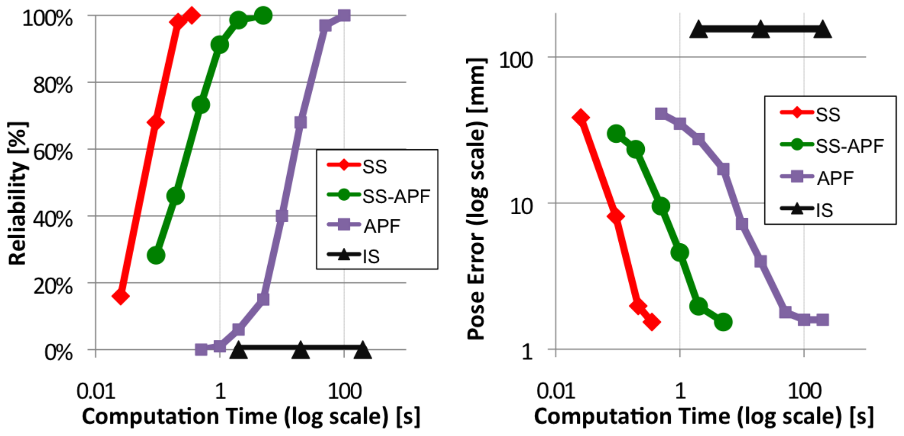


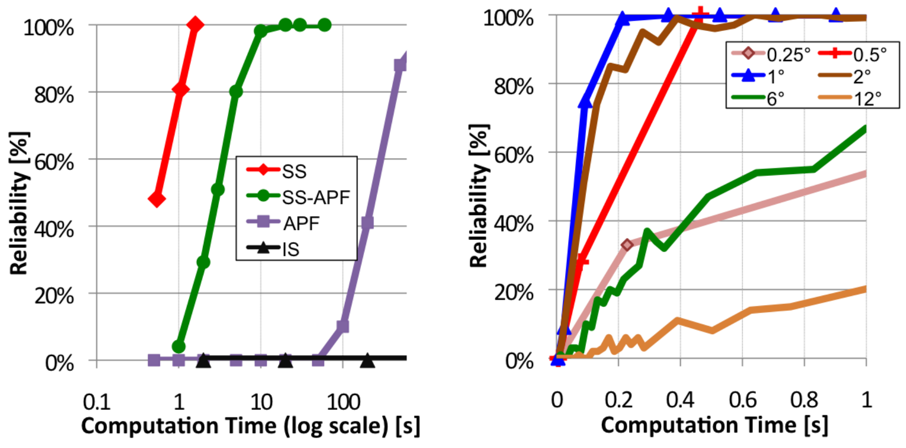
Figure 2.10: Comparison of SS, SS-APF, APF, and IS on single mode beliefs. Left: reliability vs. computation time. Right: accuracy vs. computation time.

Multi-mode estimation

In this set of experiments we used data sets with 3 measurements from three adjacent sides of the box. Such data sets do not fully constrain the problem and the resulting belief has four modes. We evaluated reliability of SS, SS-APF, APF, and IS (Fig. 2.11a). An experiment was considered successful if the approximation had at least one particle within 1mm and 1° of each of the four modes. For SS, the running time was varied by setting $M = 5$ to 7. Again, IS was unable to find all the modes even after several minutes. SS and SS-APF both were able to find all the modes, with SS-APF taking 20s, and SS being approximately 15 times faster (1.5s). APF was not completely reliable even after 10 minutes of computation, but it did reach reliability of 88%. APF was approximately 100 times slower than SS-APF and 1500 times slower than SS. We suspect the difference in performance would be even greater with more modes or whenever multiple modes need to be tracked over time.

Neighborhood size and shape

We evaluated the effect of δ_* and hyper-ellipse shape on the performance of Scaling Series. The hyper-ellipse shape is controlled by the position radius, r_{pos}, and the orientation radius, r_{ori}. We kept $r_{\text{pos}} = \delta_*$ in all experiments. Fig. 2.11b shows



(a) multi-modal (b) neighborhood shape

Figure 2.11: (a) Comparison of SS, SS-APF, APF, and IS on multi-modal beliefs. The plot shows percentage of successful runs, in which each algorithm found all modes. (b) Impact of changing δ-neighborhood shape on reliability of Scaling Series. Hyper-ellipse radius along orientation angles, r_{ori}, was changed during these experiments, while we kept $\delta_* = r_{\text{pos}} = 1$mm. The legend shows r_{ori} values in degrees. Computation time was varied by changing M.

the effect of changing r_{ori}. The value computed via the Lipschitz constant (from Appendix A) was 0.5° (bright red line), with performance close to optimal. Optimal performance was achieved with $r_{\text{ori}} = 1°$ (bright blue line). This is likely due to the fact that measurements tend to land in the interior of box faces, hence the effective radius for Lipschitz constant computations is smaller than the actual box radius.

Fig. 2.12 shows the impact of δ_* and M on accuracy. In the left plot, each curve keeps δ_* constant, and varies running time by changing M. The performance with the predicted $\delta_* = 0.7$mm is optimal and remains optimal with δ_* in the 0.7 to 1.5mm range.

In the right plot, each curve keeps M constant and varies δ_*. $M = 6$ was optimal converging to the minimum average error of 1.5mm with $\delta_* = 1$mm in 0.3s.

Zoom factor

The plot in Fig. 2.13a shows reliability vs. time for varying settings of *zoom*. The results are reported in terms of $Vol(V_n)/Vol(V_{n-1})$ ratio, which is easier to understand

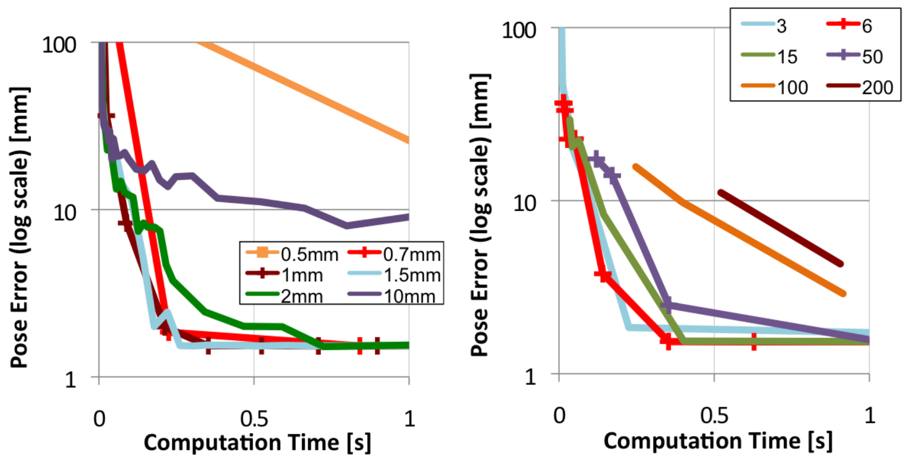


Figure 2.12: Impact of δ_* (left) and M (right) on accuracy of SS.

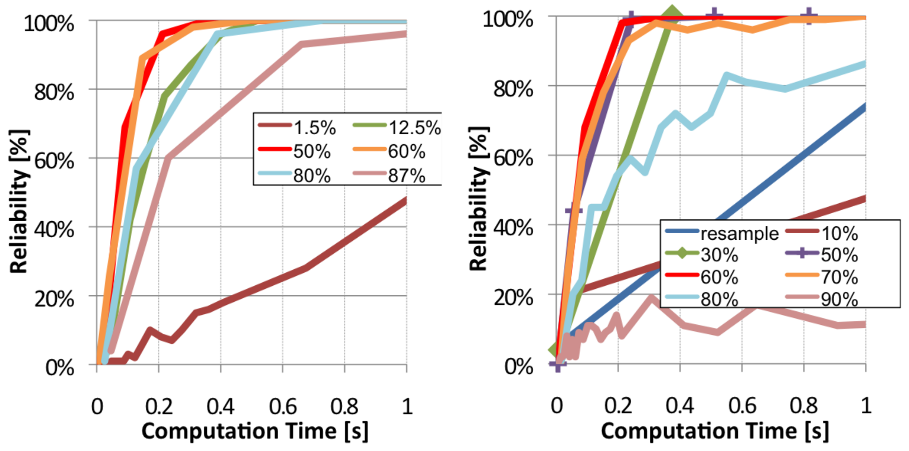
than *zoom* itself. Ratio of 50% was optimal, which corresponds to *zoom* setting shown in Alg. 2.1. However, ratios of 12.5% to 80% worked well.

Pruning

We compared the performance of Scaling Series with resampling and thresholding pruning strategies (Fig. 2.13b). For thresholding the legend shows values of ξ. Although not visible in the figure, different settings of M result in the same running time for the two different strategies. SS with resampling needs somewhere between $M = 2$ and $M = 3$ particles per δ-neighborhood. It is possible to extend Even Density Cover to work with non-integer values of M, which would allow for better performance with resampling pruning strategy. SS with thresholding on $\xi = 30\%$ to 70% needs $M = 3$ to 14 respectively. The optimal threshold was $\xi = 60\%$ (with $M = 6$) as predicted in Sect. 2.4.4.

2.6 Conclusions

We have considered the problem of global object localization via touch. Bayesian belief estimation for objects in 6DOF has been known to be computationally expensive for this problem [Gadeyne and Bruyninckx, 2001]. We have proposed an efficient



(a) zoom factor (b) pruning strategies

Figure 2.13: (a) Effect of *zoom* factor on reliability. Legend shows $Vol(V_n)/Vol(V_{n-1})$ ratio. (b) Reliability for two different pruning strategies: resampling and thresholding. For thresholding, the legend shows the threshold ξ.

approach, termed Scaling Series, that approximates the belief by particles. It performs the estimation by successively refining the high probability region and scaling granularity of estimation from low to high. Our approach does not utilize any special properties of the manipulated objects and can be easily applied to any object represented as a polygonal mesh. We have demonstrated its portability by applying it to five different everyday objects on two robotic platforms.

For fully-constraining data sets, our approach performs the estimation in real time (under 1s) with very high reliability ($\geq 99\%$). At early stages of exploration, when the data set does not fully constrain the object, the resulting belief is multi-modal. Running time in these cases depends on the precision desired and the size of the high probability region. However, our approach allows us to trade off precision of estimation for running time. Coarse estimates can be obtained quickly. As additional measurements arrive, the ambiguities are resolved and so more precise estimates can be obtained in a timely fashion.

We have provided analysis of convergence of the proposed algorithm along with strategies for parameter selection. We have also compared Scaling Series to a number of prior approaches. The results show that the proposed method outperforms prior art and is much more stable in multi-modal cases.

Similarly to Grimson and Lozano-Perez [1983], we expect that our approach can be extended to perform object identification from a set of known objects. Also, due to its stable performance with multi-modal beliefs that arise during exploration, we expect our approach to be particularly well suited for active exploration strategies that derive the optimal next sensing action based on prior data as in [Chhatpar and Branicky, 2005, Hsiao et al., 2010].

The Scaling Series algorithm can be used with other applications and sensors. For example in Chapter 5, we use Scaling Series for mobile manipulation during building navigation based on 2D laser range finders. In Chapter 4, we use Scaling Series for vehicle tracking based on 3D range data. In both cases, Scaling Series provides a significant improvement over state-of-the-art inference methods. In both of these examples, we include additional parameters in the Scaling Series filter. In Chapter 5, we use an articulated model of a door and estimate its opening angle along with robot's position. In Chapter 4, we estimate the number of moving vehicles in a previously unknown environment, as well as estimate vehicles' shape, position, and velocity. Similar techniques can be applied to touch based object localization when less information about the object shape is available a priori or when working with articulated objects.

Although in this chapter we focused on the sense of touch exclusively, the presented approach can be naturally combined with other sensing modalities. For example, if a prior pose estimate is available from a vision system, it can be used to initialize samples of Scaling Series. If several sensing modalities are to be used simultaneously, one can perform sensor updates for each sensor within the same Scaling Series filter.

A number of aspects of the presented approach can be improved upon in future work. The running time of the algorithm depends linearly on the complexity of objects (i.e., number of faces in the mesh model). However, it is possible to implement efficiency improvements that only consider a small subset of faces during each measurement evaluation. So far experiments with moving objects have only been carried out in simulation, and so this aspect warrants further attention, although better hardware is likely to be required. Additional considerations will be needed if the object to be localized is placed into a cluttered environment, where the correspondence problem of measurements to objects will need to be solved. More work can go into devising better sensing procedures in order to reduce sensing time. In particular, it is possible to use compliant motions during exploration to reduce the time the robot has to travel to and from the object. However, more sophisticated sensor configuration will be required to make sure the robot does not contact the object with non-sensing surfaces during exploration.

Chapter 3

Whole Body Contacts

3.1 Introduction

Today robots prefer to avoid contact with the environment along body or links. They strive to interact with the surroundings only by their end effectors. In contrast humans are able to do a great deal by using contact along their limbs and torso. For example, we brace against a desk while handwriting. When stumbling in the dark, we stretch our arms forward to feel the environment. We support ourselves with our knees and forearms while climbing into a tight space.

By learning to utilize contact along manipulator links, robots gain the same advantages as humans (see Fig. 3.1 for illustration). For example, it has been shown by Lew and Book [1994] that bracing increases manipulator precision, and thus, it is desirable to brace for fine manipulation tasks. During exploration, it is much easier to bump into objects if we utilize the entire robot surface as opposed to just the end effector tip. Just like humans, human-like robots also need to be able to support themselves with arms and knees when climbing.

Recently, approaches to control for whole body contact have been proposed in the literature [Liu et al., 1999, Park and Khatib, 2005, Schutter et al., 2005] (see Sect. 3.2.2 for a brief overview). However, absence of contact estimation is a major obstacle holding robots back from using whole body contacts. To our knowledge, there has been no work on estimating contact points along manipulator links, yet this

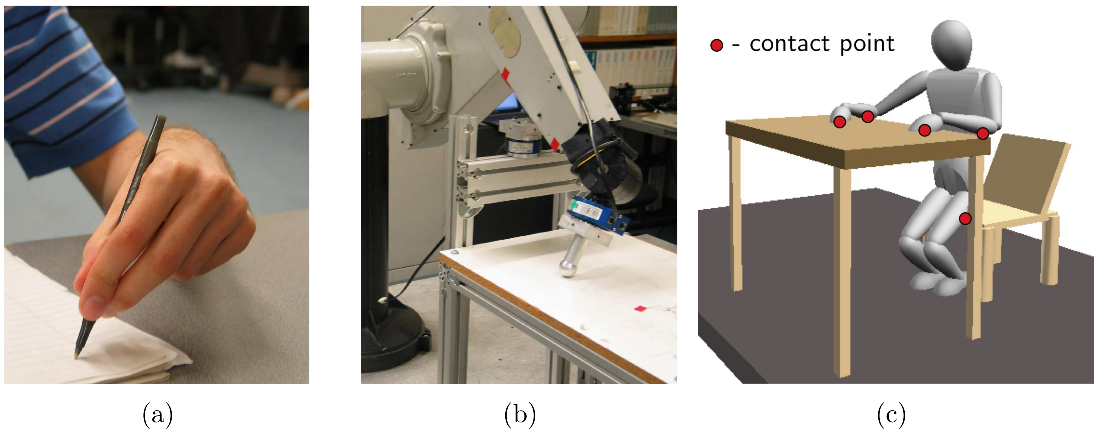


Figure 3.1: Applications of whole body contacts: (a) Humans use bracing to increase precision during fine manipulation tasks such as handwriting. (b) PUMA manipulator in a similar bracing configuration. It has been shown that bracing improves manipulator precision as well. (c) Human-like robots utilize multiple link contacts for bracing and support.

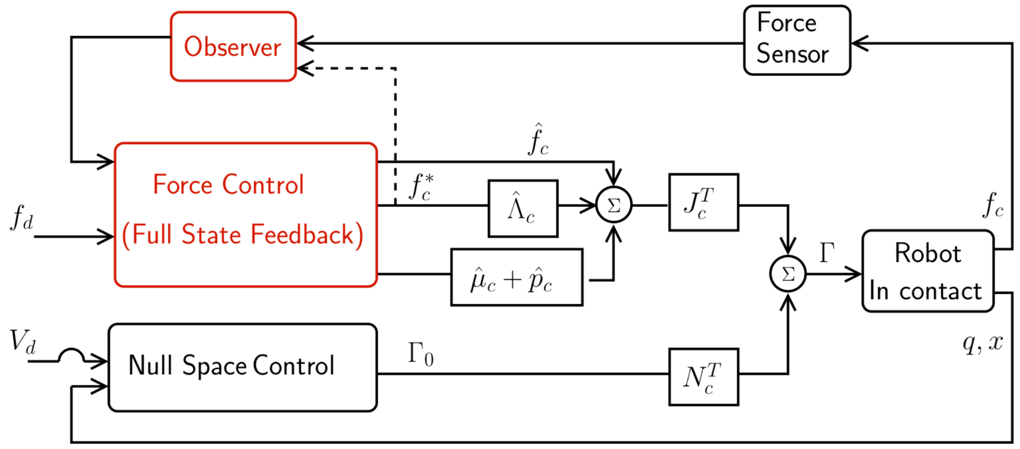


Figure 3.2: A block diagram of the contact control framework for a manipulator, where the Active Observer (AOB) design is implemented for force control. The observer in the AOB design includes a state for input disturbance and the estimate of this state will be directly compensated for in addition to the full state feedback.

information is clearly required for control algorithms.

Estimation of whole body contact points is more difficult for robots than for humans, because most robots today do not possess skin capable of sensing. Human skin is a complex sensor and creation of comparable robotic skin is currently an area of active research (see for example [Someya et al., 2004]). Even once robotic skin is widely available, its complexity and cost may be prohibitive for many applications. In this chapter, we propose an active sensing strategy that robots can use to estimate whole body contacts even without skin.

Since probabilistic techniques have been hugely successful in other areas of robotics (e.g., see a recent book on mobile robotics [Thrun et al., 2005]), we focus on a probabilistic approach to contact point estimation. Our approach results in an efficient online technique that we utilize during our experiments. As exact robot geometry is often unknown, we also provide an offline algorithm for estimating geometric parameters of robot links simultaneously with contact point estimation. Our experimental results demonstrate that estimation of contact points is crucial for control performance.

3.2 Background

3.2.1 Related Work in Perception

To operate in environments built for humans, robots need to estimate environment parameters from sensory information. One popular sensor used for manipulation perception tasks is vision (see [Kragic and Christensen, 2002] for a recent survey). However, due to high precision of manipulators, perception via contact offers very high precision, which is difficult to attain with other sensors. High precision is often required for fine manipulation tasks and for balance control tasks. Traditionally, manipulation approaches do not have probabilistic basis, e.g., [Shekhar et al., 1986, Moll and Erdmann, 2003]. Recently, several groups explored probabilistic techniques. For example, Slaets et al. [2004] used a variant of Kalman filters to estimate environment parameters for cube-in-corner assembly tasks. In Chapter 2, we used a particle filter variant to localize objects by probing them with the end effector.

3.2.2 Related Work in Control

Research on motion and force control strategies has begun with contact at the end-effector level [Siciliano and Villani, 1999, Chiaverini et al., 1999, Yoshikawa, 2000] using *compliant frame selection matrices* [Raibert and Craig, 1981, Khatib, 1987] to describe the decomposition of the end-effector space in the contact frame. Later, more general kinematic contact models were presented by Featherstone et al. [1999], Bruynincks et al. [1995], Lipkin and Duffy [1988], West and Asada [1985], and Park et al. [2004] for non-orthogonal decomposition of motion and force directions.

Control strategies for multiple contact over multiple links were presented by Liu et al. [1999], Park and Khatib [2005], and Schutter et al. [2005]. Liu et al. [1999] present an adaptive control approach for multiple geometric constraints using joint-space orthogonalization and Schutter et al. [2005] propose a constraint-based approach dealing with multiple contacts.

3.2.3 Overview of Control Approach

The control framework in this chapter uses the approach of Park and Khatib [2005], Park [2006] (see Fig. 3.2 for an illustration). This approach defines the operational space coordinates using contact force space, which spans all contacts over the links. The dynamics of the contact forces are composed by projecting the robot dynamics into the operational space and using an environment model. Control torques are chosen to compensate for the dynamics, resulting in linearized second order systems for each contact force [Freund, 1975, Khatib, 1987]. This framework allows for the use of any linear controller at the decoupled level of control. The *Active Observer* (*AOB*) method of Cortesão [2002] is used to deal with unknown disturbances, unmodeled friction, and parameter errors in the environment model. Motion control is composed in the null-space using task consistent dynamics [Khatib et al., 2004], resulting in dynamically decoupled motion and force control structure .

3.3 Contact Estimation

3.3.1 Active Sensing Strategy for Data Collection

To collect data for link contact estimation, we perform compliant motions of the link (see Fig. 3.3 for an illustration). The goal is to maintain contact with the environment throughout the sensing procedure. This way geometric shape of the robot allows us to estimate the environment. To ensure that contact is always maintained, we pick an arbitrary point on the link and initiate force control towards the environment object. We then perform motion control in direction perpendicular to the force control direction.

3.3.2 Model and Notation

Our measurements consist of joint angles $\mathbf{q}$ of the manipulator. We will denote by $\mathbf{q}^T$ the set of all measurements collected from time 1 to time T, i.e., $\mathbf{q}^T = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_T\}$. Since we want to estimate contact along the surface of manipulator links, we need a representation of robot's geometric shape. 3D shapes are usually represented by either polygonal meshes or parametric surfaces (e.g., super-quadrics). Since our PUMA manipulator has polygonal shape, we chose the mesh representation. We denote the set of parameters encoding robot's shape by $\mathbf{r}$. We denote by $\mathbf{s}$ the set of parameters encoding the shape of the environment object and its position in robot's coordinate system.

3.3.3 Probabilistic Inference with Known Robot Geometry

Let us first consider the problem of estimating environment parameters $\mathbf{s}$, when robot geometric parameters $\mathbf{r}$ are known. In probabilistic terms, it means finding values of environment parameters $\mathbf{s}$ that maximize the following probability:

$$bel_{env} := p(\mathbf{s}|\mathbf{q}^T, \mathbf{r}) \tag{3.1}$$

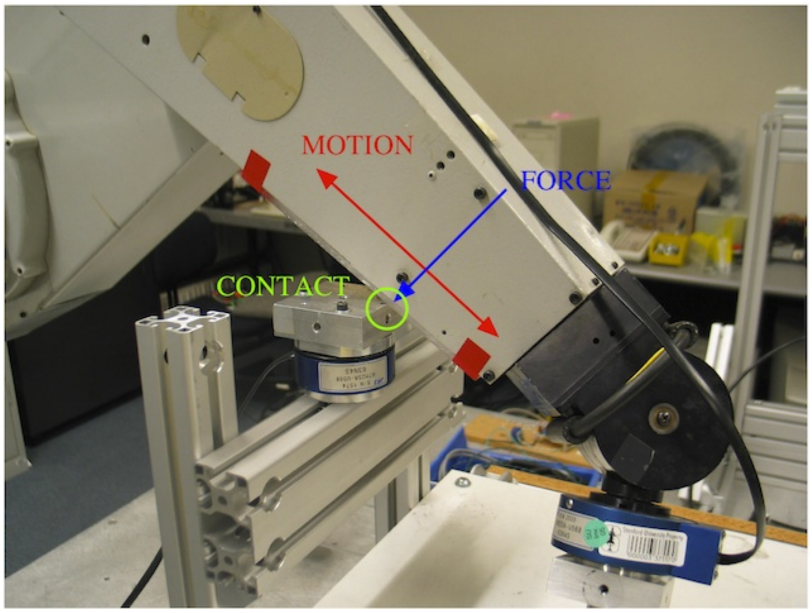


Figure 3.3: Active sensing strategy used to collect data for link contact estimation. Force control is applied towards the environment object to maintain contact. Motion control is applied in perpendicular direction.

Using the Bayes rule, we can re-write the above equation as

$$bel_{env} = p(\mathbf{s}|\mathbf{q}^T, \mathbf{r}) = p(\mathbf{q}^T|\mathbf{r}, \mathbf{s})\frac{p(\mathbf{s}|\mathbf{r})}{p(\mathbf{q}^T|\mathbf{r})}. \tag{3.2}$$

Here, $p(\mathbf{s}|\mathbf{r})$ and $p(\mathbf{q}^T|\mathbf{r})$ are prior beliefs about object shape and robot configuration given robot shape. Since $\mathbf{s}$ does not appear in $p(\mathbf{q}^T|\mathbf{r})$, this prior is constant with respect to $\mathbf{s}$. The object shape prior, $p(\mathbf{s}|\mathbf{r})$, encodes many factors, e.g., that it is not possible for the object to overlap with non-movable parts of the robot or that symmetric objects are more likely than asymmetric in human-made environments. Since it is difficult to obtain an exact representation of this prior, it is convenient to let the robot be unaware of these effects and assume the prior to be uniform. Under this assumption, bel_{env} becomes proportionate to $p(\mathbf{q}^T|\mathbf{r}, \mathbf{s})$. It is also common to assume separate measurements to be independent of each other, which allows us to factor the belief as follows:

$$bel_{env} \propto p(\mathbf{q}^T|\mathbf{r}, \mathbf{s}) = \prod_{t=1}^{T} p(\mathbf{q}_t|\mathbf{r}, \mathbf{s}) \tag{3.3}$$

Here, $p(\mathbf{q}_t|\mathbf{r}, \mathbf{s})$ is the probability of the measurement at time t given state parameters. This probability is called the measurement likelihood. We model this probability as a Gaussian distribution of the distance $dist(\mathbf{r}, \mathbf{s}|\mathbf{q})$ between the robot and the surface of the environment with variance σ^2. We define this distance as follows. If the robot and the environment object do not overlap, it is the minimum Euclidean distance between the surface of the robot in configuration $\mathbf{q}$ and the surface of the object. If the robot and the object overlap, then it is the minimum distance the object has to be moved in order to not overlap with the robot. Then the measurement likelihood can be written as

$$p(\mathbf{q}_t|\mathbf{r}, \mathbf{s}) = \eta \exp\left(-\frac{dist(\mathbf{r}, \mathbf{s}|\mathbf{q}_t)^2}{2\sigma^2}\right), \tag{3.4}$$

where η is a normalization constant. Intuitively, this makes sense because our active sensing strategy specifically collects data when the robot is in contact with the object. Thus, configurations, in which the distance between the robot and the object is very small, are likely, while configurations, for which the distance is large, are very unlikely.

Using (3.4) for measurement likelihood, allows us to transform the belief estimation into a least squares problem. By taking log of the belief, we obtain

$$\begin{aligned}
\log bel_{env} &= \log \prod_{t=1}^{T} p(\mathbf{q}_t|\mathbf{r}, \mathbf{s}) + const \\
&= \sum_{t=1}^{T} \log p(\mathbf{q}_t|\mathbf{r}, \mathbf{s}) + const \\
&= \sum_{t=1}^{T} \log\left(\eta \exp\left(-\frac{dist(\mathbf{r}, \mathbf{s}|\mathbf{q}_t)^2}{2\sigma^2}\right)\right) + const \\
&= -\frac{1}{2\sigma^2} \sum_{i=1}^{t} dist(\mathbf{r}, \mathbf{s}|\mathbf{q}_t)^2 + const,
\end{aligned} \tag{3.5}$$

where *const* denotes a generic constant. In the last line of (3.5), we collect all constant terms into a new constant. Thus, maximizing bel_{env} with respect to $\mathbf{s}$ is equivalent to minimizing $\sum_{t=1}^{T} dist(\mathbf{r},\mathbf{s}|\mathbf{q}_t)^2$, which is a least squares problem. In general, this problem is non-linear in the unknowns. Thus, non-linear optimization search techniques can be applied to obtain a solution.

3.3.4 Simultaneous Estimation of Robot Geometry and Contact

In the most general case, both robot's shape parameters and object parameters can be unknown. Thus, we need to estimate both $\mathbf{r}$ and $\mathbf{s}$ (collectively, state parameters) based on the collected data $\mathbf{q}^T$. In probabilistic terms, it means finding values of state parameters that maximize the following probability:

$$bel_{geo} := p(\mathbf{r},\mathbf{s}|\mathbf{q}^T). \tag{3.6}$$

Repeating similar derivations for bel_{geo} as for bel_{env}, we reduce it to least squares form:

$$\log bel_{geo} = -\frac{1}{2\sigma^2}\sum_{t=1}^{T} dist(\mathbf{r},\mathbf{s}|\mathbf{q}_t)^2 + const. \tag{3.7}$$

The only difference from bel_{env} is that here the robot's shape $\mathbf{r}$ is unknown. Greater number of unknown parameters requires longer computation times. Luckily, robot shape parameters only need to be estimated once, as the robot's geometry does not change from one experiment to another.

3.4 Experimental Results

In our experiments, we used a PUMA robotic manipulator equipped with a JR3 force and torque sensor at the wrist (Fig. 3.4). No additional sensors were placed at the manipulator joints or links. To evaluate performance of control strategies, we placed a second JR3 sensor in the environment. It is important to note that this sensor is not required for operation of our method. Its readings are not used within control or estimation algorithms.

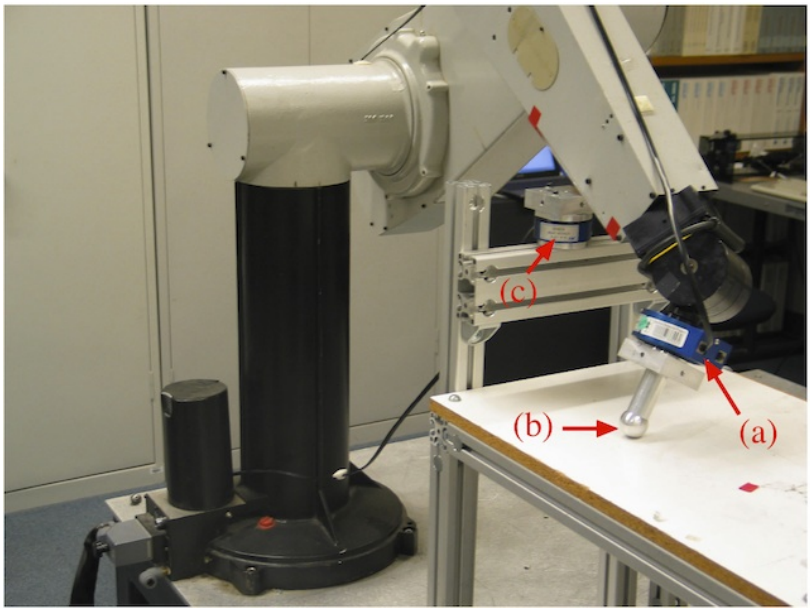


Figure 3.4: We used this 6DOF PUMA robotic manipulator in our link contact estimation experiments. The manipulator is equipped with (a) 6D JR3 force/torque sensor at the wrist, and (b) a robotic finger with a spherical end. For evaluation purposes only, we placed a force/torque sensor in the environment (c). This sensor is not required for operation of our method. We use it solely to evaluate performance.

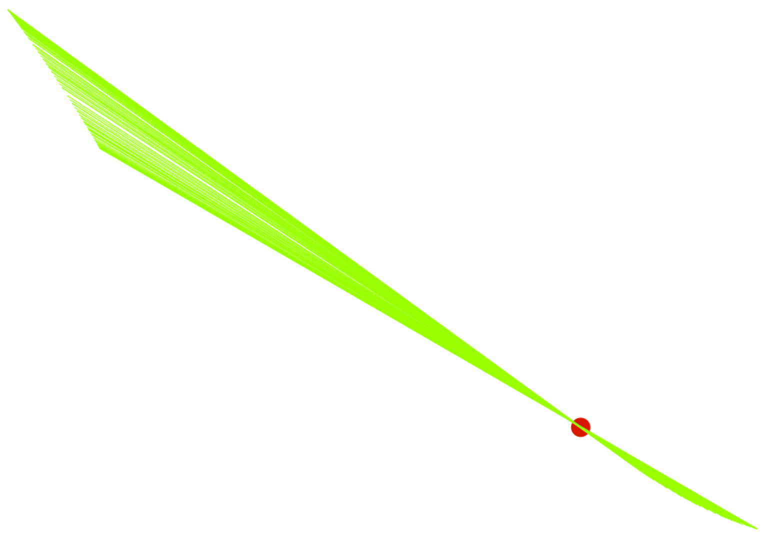


Figure 3.5: Plot of all positions of PUMA's third link edge during active sensing procedure. The estimated contact point is shown as a red circle.

3.4.1 Experiments on Contact Estimation

In our experiments, the third link of the PUMA robot comes in contact with an edge of an armrest (Fig. 3.3). Due to the shape of the robot, this creates a single point contact against one of the edges of the third link.

Once the robot comes in contact with the environment, the active sensing procedure described in Sect. 3.3.1 is initiated. The manipulator motion during this procedure is constrained within one plane. Therefore, the contact point in the environment remains the same throughout the procedure, while the contact point on the link moves.

Since the contact is a single stationary point within the environment, we can reduce our environment representation $\mathbf{s}$ to single point coordinates in the global frame. Thus for our experimental setup, the distance between the robot and the environment is simply the distance between a point and a line. Moreover, when robot geometry is known, the squared distance is a second degree polynomial in the unknown parameters. Thus, the least squares problem (derived in (3.5)) is linear in this case and can be solved in constant time using the eigenvalue method. Fig. 3.5 plots motion of the third link during active sensing and the resulting estimated contact point.

Our initial estimates of the robot's geometry came from measuring the robot with a ruler. However, it turned out that the shape of the link is unobviously asymmetric and, thus, our initial estimates were several centimeters off. For comparison, we measured the environment contact point with the end-effector of the PUMA manipulator. The error resulting from our estimation algorithm was 3.4cm. This is in part due to incorrect geometry of the link and in part to imprecise measurement of the contact point with the spherical end of the end-effector.

To obtain better estimates, we solve the full estimation problem that simultaneously considers the robot's shape **r** and the environment contact point **s**. In this case, the distance between the robot and the environment is non-linear in the unknown parameters. This problem can be solved using a variety of non-linear optimization techniques. We used a Matlab implementation based on the method described by Coleman and Li [1996].

As it is widely known, optimization search for non-linear least squares is prone to getting stuck in local minima. To overcome this problem, we first obtain initial estimate of the contact point with the ruler-measured geometry of the robot using linear least squares as described above (known robot geometry case). Then, we use the obtained contact point estimate together with the ruler-measured robot geometry as a starting point for non-linear optimization search to estimate **r** and **s**. Obtaining robot geometric parameters in this fashion improved contact point estimation precision from 3.4cm to 0.4cm.

3.4.2 Control Using Estimated Contacts

Intuitively, accurately estimated contact parameters should be important for control performance. To verify the significance in practice, we performed a series of comparison experiments. In these experiments, we performed open loop force control with and without link contact estimation. As before, the robot maintained a point

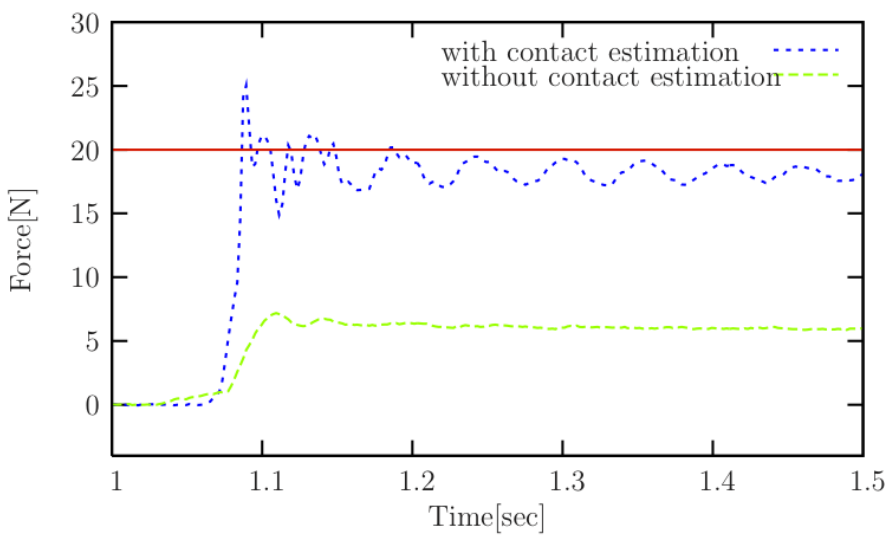


Figure 3.6: Comparison of response to step force command with and without link contact estimation. For the experiments without estimation, the assumed contact point was offset from actual position by 20cm. In the experiment shown, a 20N step command was commanded to the robot. The target response is denoted by the solid red line.

contact between the third link and the armrest. For the experiments without contact estimation, the assumed the contact point was displaced by 20cm along the link edge from the actual contact point. We performed a series of experiments with force step commands ranging from 20 to 50N. Fig. 3.6 shows comparison of force response in one of the step command experiments with and without link contact estimation. To measure these results, we placed a JR3 force sensor under the armrest. This sensor is not needed for the estimation and control algorithms, only for evaluation of the results. Overall, in our experiments with contact estimation, the error was less than 20% and without contact estimation it was over 50%.

3.4.3 Multi-Contact with Estimated Contacts

We also conducted multi-contact environment interaction experiments to demonstrate capabilities similar to human handwriting behavior. These experiments are illustrated in Fig. 3.7. During these experiments, the robot moves and makes contact with the third link against the armrest. It then initiates the active sensing procedure to

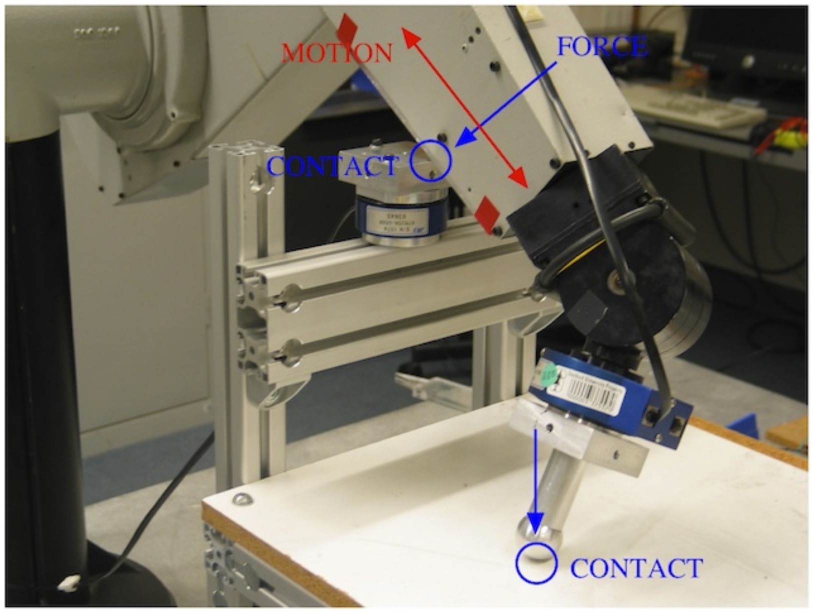


Figure 3.7: Performance of multi-contact control using the estimated environment contact point. Two point contacts are maintained during these experiments: third link contact with the armrest and end-effector contact with the desk. The robot maintains both contact forces and moves at the same time.

estimate the contact point. Once the point is estimated, the PUMA moves to make contact with its end-effector against the desk, while still maintaining contact of the third link with the armrest. In this multi-contact configuration, the robot performs the task of force control and motion control simultaneously. For force control, we command the normal direction forces at the third link contact and the end-effector contact. The normal contact force at the end-effector is feedback controlled using the measured force information from the wrist mounted force sensor. However, the contact force on the third link is controlled in an open loop and there is no feedback from a force sensor.

Using the remaining degrees of freedom of the robot, the third link was controlled to move in a tangential direction, which created a motion toward and away from the robot in a sinusoidal form. Note that even though we chose to estimate link contact point prior to making a second contact with the end effector, the motion of the robot after making the second contact could be used to estimate the link contact. Thus, the experiment demonstrates the possibility of estimating link contacts using this procedure while maintaining a multi-contact configuration.

Videos of our experiments are available at the website

http://cs.stanford.edu/people/petrovsk/manips.html

3.5 Discussion and Conclusions

Perception of link contact enables a wide range of applications including bracing to improve manipulation precision, exploration of environment, and support during climbing. For robots that do not have skin, we proposed a probabilistic approach for approximation of whole body contacts based on an active sensing strategy. Since robot shape is often known only approximately, we have also proposed an approach for simultaneously estimating the robot shape and the contact point. Our experiments clearly demonstrate the impact of contact estimation on control accuracy.

It is worth noting that estimation of whole body contacts is only possible, when the robot has sufficient degrees of freedom to carry out motion around each contact point. For example, estimation via motion is theoretically impossible for the second link of the PUMA manipulator. In these cases, artificial skin or some other type of sensor is necessary to estimate the contact. However, when degrees of freedom are sufficient, the proposed active sensing strategy provides estimates with good accuracy.

Although the described approach considers one contact at a time, it applies even if the robot maintains multiple contacts with the environment. Moreover, the approach can be easily extended to estimate multiple contacts simultaneously, provided the robot has the freedom to carry out simultaneous exploratory motions around all of the contacts. On the other hand, estimation of multiple contact points on the same link is unlikely to be possible using this method, because the link will be unable to move around each contact point independently.

There is ample room for future work on whole body contact estimation. In our experiments, we considered only simple robot and environment geometries. For more complex geometries (including complex polygonal mesh and curved representations), more sophisticated active exploration strategies will likely be needed. While the derivations of this algorithm apply to arbitrarily complex geometries, in practice these geometries will entail significantly higher numbers of parameters and lead to non-linear estimation problems. Hence, better search algorithms will be required. In addition, distance computation is more difficult for complex objects and, in fact, the notion of distance may need to be redefined depending on the representation.

Chapter 4

Vehicle Detection and Tracking

4.1 Introduction

Autonomously driving cars have been a long-lasting dream of robotics researchers and enthusiasts. Self-driving cars promise to bring a number of benefits to society, including prevention of road accidents, optimal fuel usage, comfort, and convenience. In recent years the *Defense Advanced Research Projects Agency* (*DARPA*) has taken a lead on encouraging research in this area and organized a series of competitions for autonomous vehicles. In 2005, autonomous vehicles were able to complete a 131 mile course in the desert [Buehler et al., 2007]. In the 2007 competition, the *Urban Grand Challenge* (*UGC*), the robots were presented with an even more difficult task: safe autonomous navigation in urban environments. In this competition, the robots had to drive safely with respect to other robots, human-driven vehicles, and the environment. They also had to obey the rules of the road as described in the California rulebook (see [DARPA, 2007] for a detailed description of the rules). One of the most significant changes from the 2005 competition is the need for situational awareness of both static and dynamic parts of the environment. Several successful approaches have been developed in parallel by the UGC participants [Leonard et al., 2008, Urmson et al., 2008]. Our robot, *Junior*, won second prize in the 2007 competition. An overview of Junior's software and hardware architecture is given in [Montemerlo et al., 2008]. In this chapter, we describe the approach we developed for detection and tracking of moving vehicles.

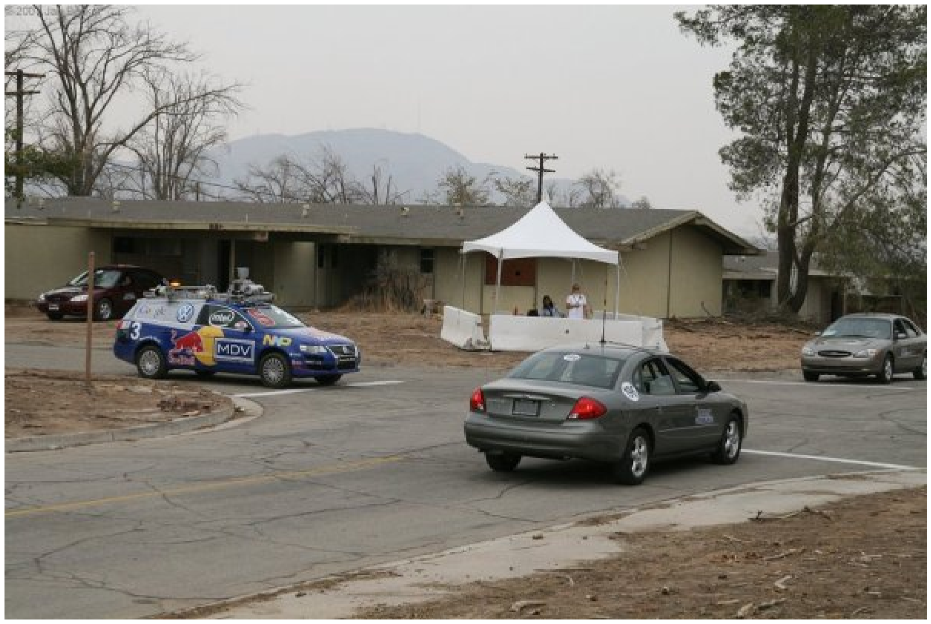
Vehicle tracking has been studied for several decades. A number of approaches focused on the use of vision exclusively [Zielke et al., 1993, Dickmanns, 1998, Dellaert and Thorpe, 1998], whereas others utilized laser range finders [Zhao and Thorpe, 1998, Streller et al., 2002, Wang et al., 2007] sometimes in combination with vision [Wender and Dietmayer, 2008]. We give an overview of prior art in Sect. 4.2.

For our application, we are concerned with laser based vehicle tracking from the autonomous robotic platform Junior, to which we will also refer as the *ego-vehicle* (see Fig. 4.1). In contrast to prior art, we propose a model based approach, which encompasses both geometric and dynamic properties of the tracked vehicle in a single Bayes filter. The approach eliminates the need for separate data segmentation and association steps. We show how to properly model the dependence between geometric and dynamic vehicle properties using *anchor point coordinates*. The geometric model allows us to naturally handle the disjoint point clusters that often result from partial occlusion of vehicles (see Fig. 4.2). Moreover, the estimation of geometric shape leads to accurate prediction of dynamic parameters (see Fig. 4.3).

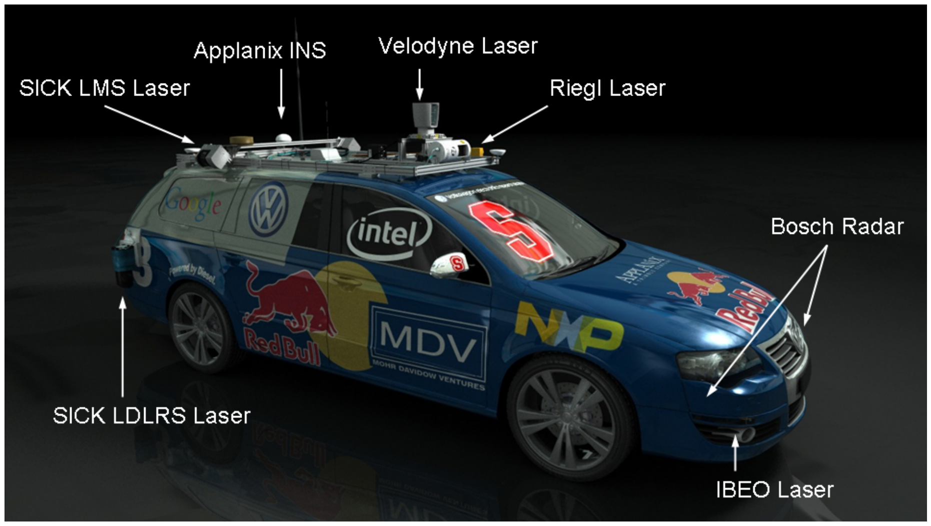
Further, we introduce an abstract sensor representation, called the *virtual scan*, which allows for efficient computation and can be used for a wide variety of laser sensors. We present techniques for building consistent virtual scans from 3D range data and show how to detect poorly visible black vehicles in laser scans. To battle the low signal-to-noise ratio during rapid detection of vehicles in noisy urban settings, we introduce the notion of *motion evidence*, which allows us to quickly prune false positives caused by noise. Our approach runs in real time with an average update rate of 40Hz, which is 4 times faster than the common sensor frame rate of 10Hz. The results show that our approach is reliable and efficient even in the challenging traffic situations presented at the UGC.

4.2 Background

A number of vehicle tracking approaches have been developed over the past few decades (e.g., Zhao and Thorpe [1998], Streller et al. [2002], Wang [2004], Wender

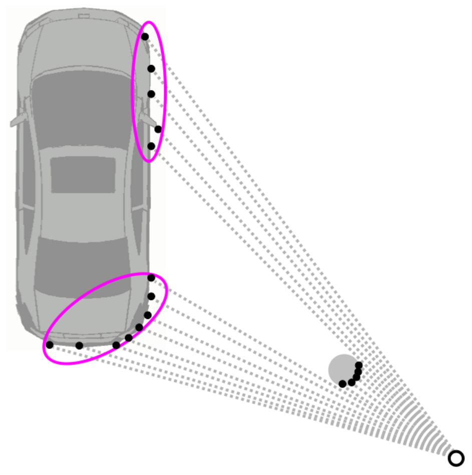


(a)



(b)

Figure 4.1: (a) Our robot Junior (blue) negotiates an intersection with human-driven vehicles at the qualification event for the Urban Grand Challenge in November 2007. (b) Junior, is equipped with five different laser measurement systems, a multi-radar assembly, and a multi-signal inertial navigation system.


(a) without geometric model

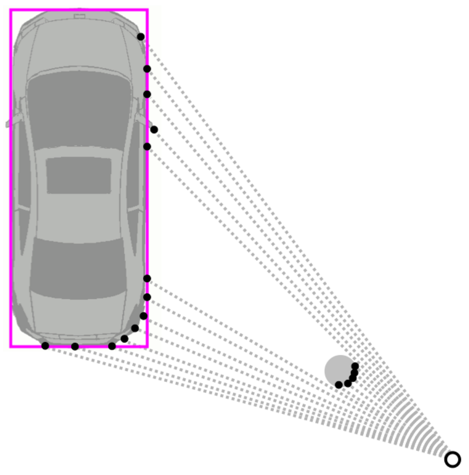
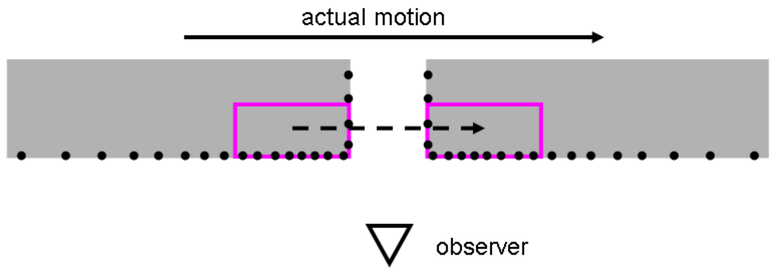

(b) with geometric model

Figure 4.2: Scans from vehicles are often split up into separate clusters by occlusion. Geometric vehicle model helps interpret the data properly. In (a), purple ovals group together points that have been associated together. In (b), the purple rectangle denotes the geometric vehicle model. Gray areas are objects. Gray dotted lines represent laser rays. Black dots denote laser data points.

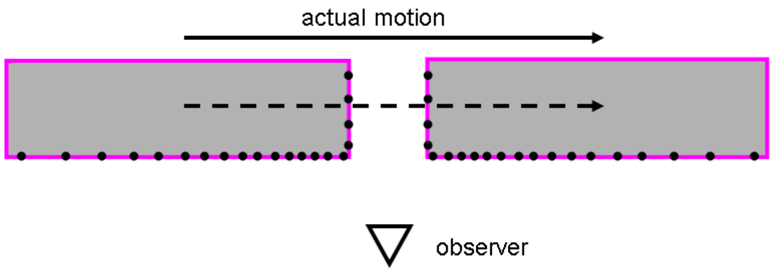
and Dietmayer [2008]) including most recent developments by the UGC participants [Darms et al., 2008, Leonard et al., 2008]. Typically, these approaches proceed in three stages: *data segmentation*, *data association*, and *Bayesian filter* update. During data segmentation, the sensor data are divided into meaningful pieces — usually line features [Zhao and Thorpe, 1998, Darms et al., 2008] or clusters [Wender and Dietmayer, 2008, Leonard et al., 2008]. During data association, these pieces are assigned to tracked vehicles. Next, a Bayesian filter update is performed to track the centroids of the targets.

The second stage — data association — is generally considered the most challenging stage of the vehicle detection and tracking problem because of the association ambiguities that arise. Typically this stage is carried out using variants of the *multiple hypothesis tracking* (*MHT*) algorithm (e.g., Streller et al. [2002], Wang et al. [2007]).

In the third stage, the filter update is usually carried out using variants of *Kalman filter* (*KF*), which is augmented by the *interacting multiple model* method in some



(a) without shape estimation



(b) with shape estimation

Figure 4.3: Vehicles come in different sizes. Accurate estimation of geometric shape helps obtain a more precise estimate of the vehicle dynamics. Solid arrows show the actual distance the vehicle moved. Dashed arrows show the estimated motion. Purple rectangles denote the geometric vehicle models. Black dots denote laser data points.

cases [Zhao and Thorpe, 1998, Wang et al., 2007].

Although vehicle tracking literature primarily relies on variants of KF, there is a great body of *multiple target tracking* (*MTT*) literature for other applications, where parametric, sample-based, and hybrid filters are used. Blackman et al. [2004] provides a summary. For example, Särkkä et al. [2007] uses a *Rao-Blackwellized particle filter* (*RBPF*) for multiple target tracking on simulated data. A popular alternative to MHT for data association is the *joint probabilistic data association* (*JPDA*) method, which was used by Schulz et al. [2001] to track multiple targets from an indoor mobile robot platform.

Vehicle detection is a pre-requisite for vehicle tracking. One of the challenges in vehicle detection is poor *signal-to-noise ratio* due to the fact that a vehicle moves only a small amount between consecutive frames. In prior art, the detection problem has been solved by addition of vision sensors (e.g., Wender and Dietmayer [2008]), although visual classification does not help distinguish moving vehicles from stationary. Another approach is to sample frames at lower rates to overcome the low signal-to-noise ratio [Wang et al., 2007], although this approach increases the time it takes to detect a new moving vehicle. Other described approaches detect vehicles by scan shape [Zhao and Thorpe, 1998, Streller et al., 2002] or by location [Wang et al., 2007]. Due to possible ambiguities in the range data, these approaches tend to have lower detection accuracy.

4.3 Representation

In this chapter, we shall assume that a reasonably precise pose of the ego-vehicle is always available. On our robot, the pose estimates are provided by the localization module, which is described in detail in [Montemerlo et al., 2008]. Here we provide a brief summary. The robot is outfitted with an *Applanix POS LV 420 inertial navigation system* (*INS*), which provides pose localization with 1m accuracy. Due to periodic GPS measurement updates, the INS pose estimate can suddenly shift by up to 1m. The sudden shifts are very undesirable for vehicle tracking as they greatly increase tracking uncertainty. For the purposes of vehicle tracking, the ego-vehicle pose estimate should evolve smoothly over time. For this reason, we implemented *smooth coordinates*, which provide a locally consistent estimate of the ego-vehicle motion by integrating the velocity estimates from the INS. Although the smooth pose estimate can drift over time, it does not experience sudden shifts. To map from smooth coordinates to globally consistent GPS coordinates, one simply needs to add an offset, which is periodically updated to reflect the mismatch between the smooth and GPS coordinate systems. A similar smooth coordinate system was independently developed by the MIT UGC team [Leonard et al., 2008]. In the remainder of this chapter, all operations will be carried out in the smooth coordinate frame, which we will also call the *world frame*. The transformation from smooth to GPS coordinates will only be needed when dealing with global features, such as the digital road map.

Following the common practice in vehicle tracking [Dellaert and Thorpe, 1998, Dietmayer et al., 2001, Leonard et al., 2008], we will represent each vehicle by a separate Bayesian filter, and represent dependencies between vehicles via a set of local spatial constraints. Specifically, we will assume that no two vehicles overlap, that all vehicles are spatially separated by some free space, and that all vehicles of interest are located on or near the road.

4.3.1 Probabilistic Model and Notation

For each vehicle, we estimate its 2D position and orientation $X_t = (x_t, y_t, \theta_t)$ at time t, its forward velocity v_t, and its geometry Ω (further defined in Sect. 4.3.2). Also, at each time step, we obtain a new set of measurements $\mathcal{D}_t$. A dynamic Bayes network representation of the resulting probabilistic model is shown in Fig. 4.4. The dependencies between the parameters involved are modeled via probabilistic laws discussed in detail in Sects. 4.3.3 and 4.3.5. For now, we briefly note that the velocity evolves over time according to

$$p(v_t|v_{t-1}). \tag{4.1}$$

The vehicle moves based on the evolved velocity according to a dynamics model

$$p(X_t|X_{t-1}, v_t). \tag{4.2}$$

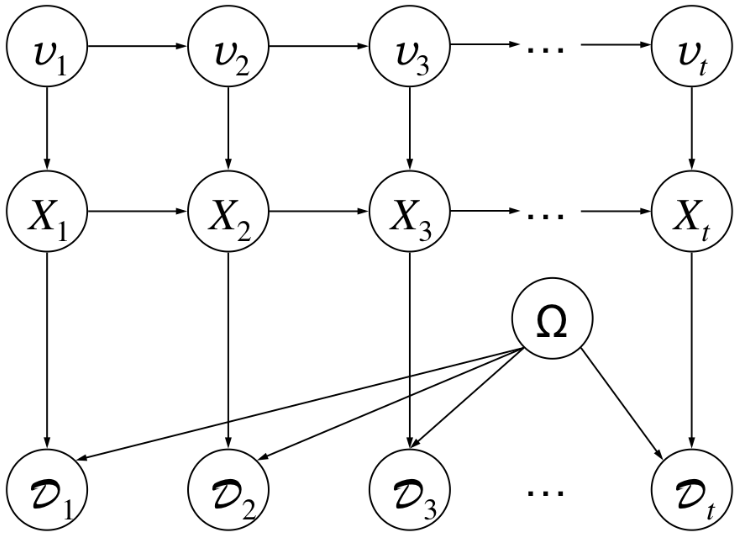


Figure 4.4: Dynamic Bayesian network model of the tracked vehicle pose X_t, forward velocity v_t, geometry Ω, and measurements $\mathcal{D}_t$.

The measurements are governed by a measurement model

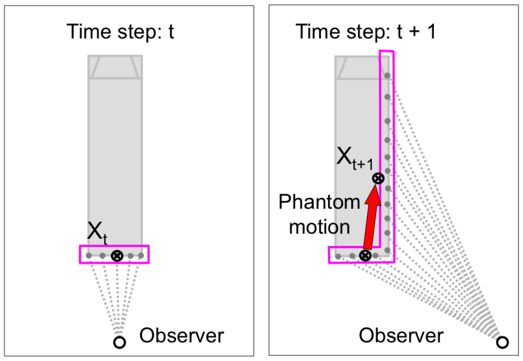
$$p(\mathcal{D}_t|X_t, \Omega). \tag{4.3}$$

For convenience, we will write $X^t = (X_1, X_2, ..., X_t)$ for the vehicle's trajectory up to time t. Similarly, v^t and $\mathcal{D}^t$ will denote all velocities and measurements up to time t.

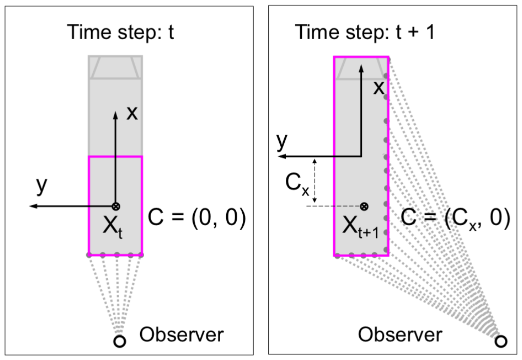
4.3.2 Vehicle Geometry

The exact geometric shape of a vehicle can be complex and difficult to model precisely. For simplicity, we approximate it by a rectangular shape of width W and length L. The 2D representation is sufficient because the height of tracked vehicles is not important for driving applications.

During vehicle tracking, the state variable X_t usually represents the position of the vehicle's center in the world coordinate frame. However, there is an interesting dependence between our belief about the vehicle's shape and its position. As we observe the object from a different vantage point, we change not only our belief of its



(a) without geometric model



(b) with geometric model

Figure 4.5: As we move to observe a different side of a stationary bus, our belief of its shape changes and so does the belief about position of the vehicle's center point. (a) Without geometric vehicle model, the geometric mean of observed points shifts resulting in phantom motion (red arrow) of X_t. (b) With geometric model, we compensate for the effect by introducing local anchor point coordinates $C = (C_x, C_y)$ so that we can keep the anchor point X_t stationary in the world coordinates.

shape, but also our belief of the position of its center point. Allowing X_t to denote the center point can lead to the undesired effect of obtaining a non-zero velocity for a stationary vehicle, simply because we refine our knowledge of its shape as Fig. 4.5 illustrates.

To overcome this problem, we view X_t as the pose of an *anchor point*, whose position with respect to the vehicle's center can change over time. Initially, we set the anchor point to be the center of what we believe to be the vehicle's shape and thus its coordinates in the vehicle's *local* coordinate system are $C = (0, 0)$. We assume that the vehicle's local coordinate system is tied to its center with the x-axis pointing directly forward. As we revise our knowledge of the vehicle's shape, the local coordinates of the anchor point will also need to be revised accordingly to $C = (C_x, C_y)$. Thus, the complete set of geometric parameters is $\Omega = (W, L, C_x, C_y)$.

4.3.3 Vehicle Dynamics Model

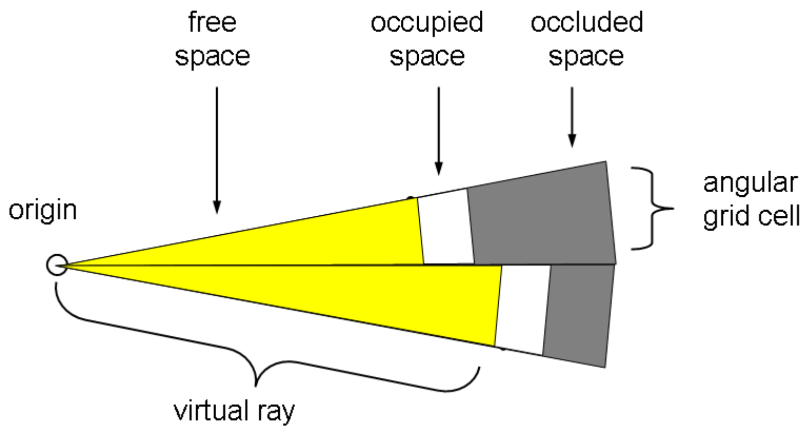
In vehicle tracking literature, it is common to use a *constant velocity model* [Dellaert and Thorpe, 1998], a *constant acceleration model* [Dietmayer et al., 2001], or a *switching dynamics model* [Wang, 2004, Darms et al., 2008]. We use the constant velocity model and assume that velocity of each tracked vehicle stays constant for the duration of each time interval from $t-1$ to t. It also instantaneously evolves at each time step t via addition of random bounded noise based on the maximum allowed acceleration a_{max} and the time delay Δt from the previous time step $t-1$. Specifically, we sample Δv uniformly from $[-a_{max}\Delta t,\ a_{max}\Delta t]$.

The pose evolves via *linear motion* [Thrun et al., 2005, Sec. 5.4] — a motion law that is often utilized when exact dynamics of the object are unknown. The motion consists of perturbing orientation by $\Delta\theta_1$, then moving forward according to the current velocity by $v_t\Delta t$, and making a final adjustment to orientation by $\Delta\theta_2$. Again we sample $\Delta\theta_1$ and $\Delta\theta_2$ uniformly from $[-d\theta_{max}\Delta t,\ d\theta_{max}\Delta t]$ for a maximum allowed orientation change $d\theta_{max}$.

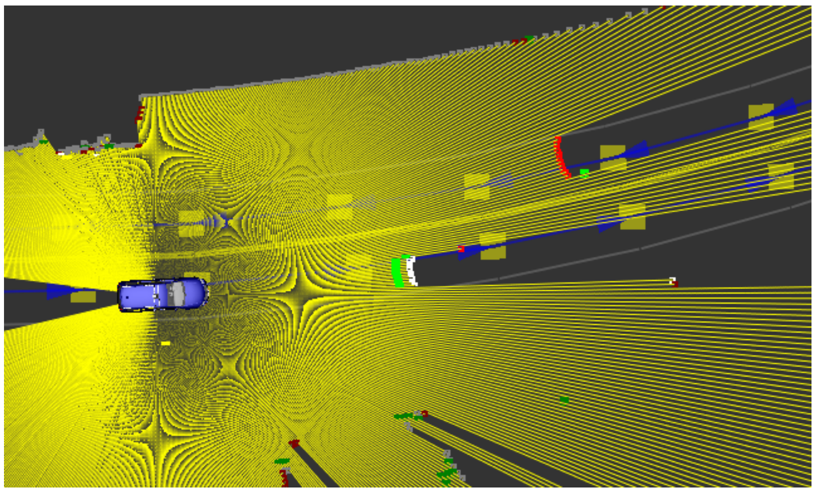
4.3.4 Sensor Data Representation

In this chapter, we focus on laser range finders for sensing the environment. Recently, these sensors have evolved to be more suitable for driving applications. For example, *IBEO Alasca* sensors allow for easy *ground filtering* by collecting four parallel horizontal scan lines and marking which of the readings are likely to come from the ground [Ibeo Automobile Sensor GmbH, 2008]. *Velodyne HDL-64E* sensors do not provide ground filtering, however, they take a 3D scan of the environment at high frame rates (10Hz) producing 1,000,000 readings per second [Velodyne Lidar, Inc., 2008]. Given such rich data, the challenge has become to process the readings in real time as vehicle tracking at 10 - 20Hz is desirable for driving decision making.

A number of factors make the use of raw sensor data inefficient. As the sensor rotates to collect the data, each new reading is made from a new vantage point due to ego-motion. Ignoring this effect leads to significant sensor noise. Taking this effect into account makes it difficult to quickly access data that pertains to a specific region of space. Much of the data come from surfaces uninteresting for the purpose of vehicle tracking, e.g., ground readings, curbs and tree tops. Finally, the raw 3D data wastes a lot of resources as vehicle tracking is a 2D application where the cars are restricted



(a) anatomy of a virtual scan



(b) a virtual scan constructed from Velodyne data

Figure 4.6: In (b), yellow line segments represent virtual rays. Colored points show the results of a scan differencing operation. Red points are new obstacles, green points are obstacles that disappeared, and white points are obstacles that remained unchanged or appeared in previously occluded areas.

to move on the ground surface. Therefore, it is desirable to pre-process the data to produce a *virtual sensor* representation tailored for vehicle tracking.

Virtual sensors have been employed in the past for a wide range of applications. For example, in neuroimaging, virtual sensors have been created from fMRI data using machine learning techniques for diagnosis of mental processes in patients with brain injuries [Mitchell et al., 2002]. In sensor networks, virtual sensors have been implemented to abstract data from multiple non-homogeneous sensors [Kabadayi et al., 2006]. In geoscience, virtual sensors have been constructed using models trained on spectrally rich data to "fill in" unmeasured spectral channels in spectrally poor data for improved detection of clouds over snow and ice [Srivastava et al., 2005]. In artificial intelligence and robotics, virtual sensors are commonplace in simulated environments, often used as a testbed for perception, planning, and control algorithms [Thalmann et al., 1997, Gerkey et al., 2003].

To create a virtual sensor for our application, we construct a grid in polar coordinates — a *virtual scan* — which subdivides 360° around a chosen origin point into *angular grid cells* (see Fig. 4.6). In each angular grid cell, we record the range to the closest obstacle within that cell. Hence, each angular grid cell contains the following information: the space from origin up to the recorded range is free, at the recorded range — occupied, and beyond the recorded range — occluded. We will often refer to the cone of an angular grid cell from the origin up to the recorded range as a *virtual ray* (or simply *ray*) due to its similarity to a laser ray. We will also treat each angular grid cell as a single range measurement in the virtual scan.

Virtual scans simplify data access by providing a single point of origin for the entire data set, which allows constant time look-up for any given point in space. As we mentioned earlier, it is important to compute correct world coordinates for the raw sensor readings. However, once the correct positions of obstacle points have been computed, adjusting the origin of each ray to be at the common origin for the virtual scan produces an acceptable approximation. To minimize the error due to approximation, we select the common origin to be the average sensor pose during scan collection. Constructed in this manner, a virtual scan provides a compact representation of the space around the ego-vehicle classified into free, occupied and occluded.

The classification helps us properly reason about what parts of an object should be visible as we describe in Sect. 4.3.5.

One important parameter of a virtual scan is the angular resolution. Although coarser resolutions can speed up computations because fewer rays need to be examined, it is desirable to set the resolution as fine as possible in order to capture more detail about objects at long range.(1) For this reason, we set the resolution as fine as possible in our implementation. For the IBEO lasers, we set the resolution to $0.5°$, which is the highest resolution the sensor provides.

For the purpose of vehicle tracking, it is crucial to determine what changes take place in the environment over time. With virtual scans, these changes can be easily computed in spite of the fact that ego-motion can cause two consecutive virtual scans to have different origins. The changes are computed by checking which obstacles in the old scan are cleared by rays in the new scan and vice versa. This computation takes time linear in the size of the virtual scan and only needs to be carried out once per frame. Fig. 4.6b shows results of a *virtual scan differencing* operation with red points denoting new obstacles, green points denoting obstacles that disappeared, and white points denoting obstacles that remained in place or appeared in previously occluded areas.

Virtual scans are a suitable representation for a wide variety of laser range finders. While this representation is easy to build for 2D sensors such as IBEO, 3D range sensors require additional considerations to produce consistent 2D representations. We describe these techniques in Sect. 4.6.

4.3.5 Measurement Model

This section describes the measurement model $p(\mathcal{D}|X, \Omega)$ used in our approach. Here, $\mathcal{D}$ is a virtual scan representation of a single frame of range data from a laser range finder.

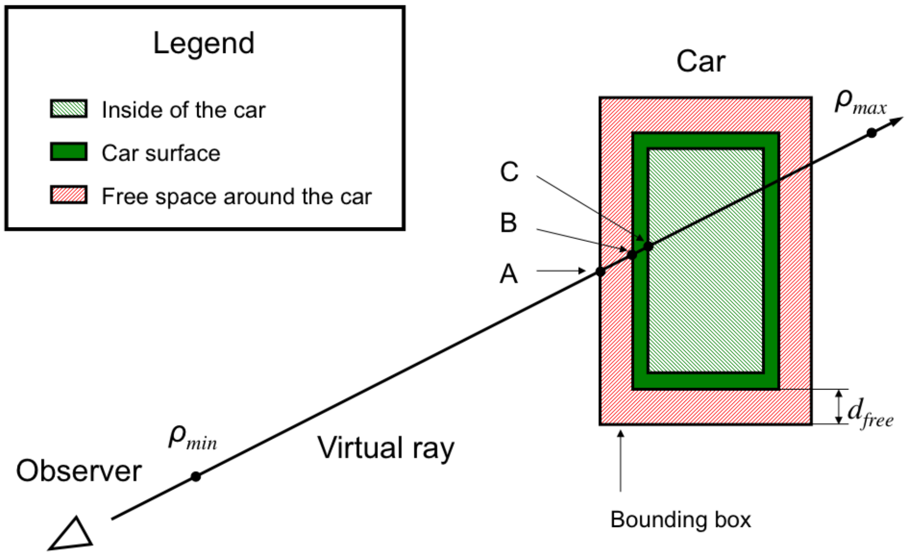
(1)In principle, it is possible to get the best of both worlds by constructing several virtual scans of varying resolution for the same laser data. Lower resolution virtual scans can be used to examine close range objects, while higher resolution scans can be used for long range operations.

Prior Art on Measurement Models

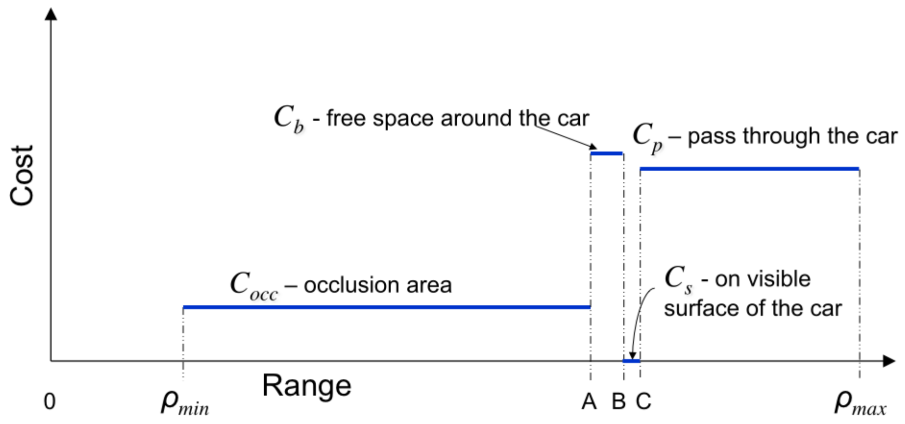
To our knowledge, range scan likelihood models have not been proposed for vehicle tracking, as most of the vehicle tracking literature is concerned with tracking point targets — usually centers of clusters [Wang, 2004, Leonard et al., 2008] or features extracted from the range data [Streller et al., 2002, Wender and Dietmayer, 2008, Darms et al., 2008]. In contrast to the prior art, we are able to provide a direct interpretation of the range measurements because we model geometry of the tracked vehicles. Measurement models for range finders in the presence of a geometric environment model have been proposed in mobile robot localization and mapping literature, where the environment is commonly represented by an *occupancy grid map* (see Thrun et al. [2005, Ch. 6] for an overview). The two most common models are the *independent beam model* (*IB*) [Moravec, 1988, Burgard et al., 1996, Fox et al., 1999] and the *likelihood field model* (*LF*) [Thrun, 2001].

The IB model treats each ray in the scan as an independent measurement of range to the closest obstacle along the ray corrupted by Gaussian noise. One drawback of the IB model is that rays are represented by lines. This assumption does not work well at longer ranges ($50 - 100$m) typical in outdoor environments. Outdoors, it is better to represent rays by cones because the laser spot light is of non-negligible radius ($20 - 40$cm). Another drawback is that the IB model does not leave room for possible unmodeled occlusions of the geometric model — a very common scenario in vehicle tracking.

The LF model also treats laser rays as independent of each other. The end point of each ray is compared to the closest obstacle point (not necessarily on the ray itself) under the assumption of Gaussian noise. The LF model is more appropriate for cone representation of rays. It also handles unmodeled occlusions very well. However, the LF model allows rays to go through obstacles without any penalty. This is undesirable for vehicle tracking because rays going through a candidate vehicle provide strong evidence that these points may not belong to the same physical object.



(a)



(b)

Figure 4.7: Measurement likelihood computations. (a) shows the geometric regions involved in the likelihood computations. (b) shows the costs assignment for a single ray.

Our Measurement Model

Given a vehicle's pose X, geometry Ω, and a virtual scan $\mathcal{D}$, we compute the measurement likelihood $p(\mathcal{D}|\Omega, X)$ as follows. We position a rectangular shape representing the vehicle according to X and Ω. Then, we build a bounding box to include all points within a predefined distance d_{free}[(2)] around the vehicle (see Fig. 4.7). Assuming that there is an actual vehicle in this configuration, we would expect the points within the rectangle to be occupied or occluded, and points in its vicinity to be free or occluded because vehicles are spatially separated from other objects in the environment.

Like the IB and LF models for laser range finders, we consider measurements obtained along each ray to be conditionally independent of each other given vehicle pose and geometry. Thus, if we have a total of K rays in the virtual scan $\mathcal{D}$, the measurement likelihood factors as follows

$$p(\mathcal{D}|\Omega, X) = \prod_{k=1}^{K} p(D_k|\Omega, X). \tag{4.4}$$

Following the IB and LF models, we use a Gaussian form for each ray's likelihood. Specifically, we model it as a zero-mean Gaussian of variance σ_k computed with respect to a cost c_k selected based on the relationship between the ray and the vehicle (η_k is a normalization constant):

$$p(D_k|\Omega, X) = \eta_k \ \exp\Big(-\frac{c_k^2}{2\sigma_k^2} \Big). \tag{4.5}$$

The costs are set to constants that depend on the region in which the ray's end point falls (see Fig. 4.7 for illustration). c_{occ} is the cost for range readings that fall short of the bounding box and thus represent situations when another object is occluding the vehicle. c_b is the cost for range readings that fall short of the vehicle but inside of the bounding box. c_s is the cost for readings on the vehicle's visible surface which we assume to be of non-zero depth d_{sur}. c_p is used for rays that extend beyond the vehicle's surface. Assigning likelihood based on the region of space in which a ray's

(2) We used the setting of $d_{free} = 1$m in our implementation.

end point falls bears resemblance to the LF model. It is more appropriate for cone representation of rays than the IB model. Like the LF model, our measurement model gives little penalty to occlusions by other objects, but unlike the LF model, we assign a large penalty to rays passing through the candidate vehicle. We also enforce our assumption of free space around each vehicle by assigning a large penalty to rays that terminate in this region.

The domain for each range reading is between the minimum range ρ_{min} and the maximum range ρ_{max} of the sensor. Since the costs we select are piece-wise constant, it is easy to integrate the unnormalized likelihoods to obtain the normalization constants η_k. Note that for the rays that do not target the vehicle or the bounding box, the above logic automatically yields uniform distributions as these rays never hit the bounding box.

Note that the above measurement model naturally handles partially occluded objects, including objects that are "split up" by occlusion into several point clusters (see Fig. 4.2). In contrast to our approach, these cases are often challenging for approaches that utilize separate data segmentation and correspondence methods.

4.4 Vehicle Tracking

Most vehicle tracking methods described in the literature apply separate methods for data segmentation and correspondence matching before filtering via an *extended Kalman filter* (*EKF*). In contrast, we use a single Bayesian filter to fit model parameters from the start. This is possible because we model both geometric and dynamic properties of the vehicles and because our measurement model interprets the range data directly. Since EKF does not work well when *relative sensors* are interpreted directly[(3)], we use a particle filter for Bayesian estimation.

[(3)]The laser range finders are relative sensors for the vehicle tracking problem because the information about vehicles is contained in the data implicitly. The data consist of individual range measurements, which can belong to the same or different targets. Hence, the positions and velocities of moving vehicles can not be easily determined directly from the data. See the discussion on relative vs. global sensors in Sect. 1.1.2 and on parametric inference methods in Sect. 1.2.1.

Unlike the MHT method commonly used in the literature, the computational complexity for our method grows linearly with the number of vehicles in the environment because vehicle dynamics dictates that vehicles can only be matched to data points in their immediate vicinity. The downside, of course, is that two targets can in principle merge into one. In practice, we have found that this happens very rarely and only in situations where one of the targets is lost due to complete occlusion. In these situations, target merging is acceptable for our application.

We have a total of eight parameters to estimate for each vehicle: $X = (x, y, \theta)$, v, $\Omega = (W, L, C_x, C_y)$. Computational complexity grows exponentially with the number of parameters for particle filters. Thus, to keep computational complexity low, we turn to RBPFs [Doucet et al., 2000]. We estimate X and v by particles and keep Gaussian estimates for Ω within each particle. Below we give a brief derivation of the required update equations.

4.4.1 Derivation of Update Equations

At each time step t, we produce an estimate of a Bayesian belief about the tracked vehicle's trajectory, velocity, and geometry based on a set of measurements

$$bel_t := p(X^t, v^t, \Omega | \mathcal{D}^t). \tag{4.6}$$

The derivation provided below is similar to the one used by Montemerlo [2003]. We split the belief into two conditional factors:

$$bel_t = p(X^t, v^t | \mathcal{D}^t) \;\; p(\Omega | X^t, v^t, \mathcal{D}^t). \tag{4.7}$$

The first factor represents the belief about vehicle's motion,

$$R_t := p(X^t, v^t | \mathcal{D}^t). \tag{4.8}$$

The second factor represents the belief about vehicle's geometry, conditioned on its motion,

$$S_t := p(\Omega | X^t, v^t, \mathcal{D}^t). \tag{4.9}$$

The factor R_t is approximated using a set of particles; the factor S_t is approximated using a Gaussian distribution (one Gaussian per particle).

Updating Vehicle's Motion Belief

Let $\mathcal{X}_t$ denote the set of particles at time t. We compute $\mathcal{X}_t$ recursively from $\mathcal{X}_{t-1}$. Suppose that at time $t-1$, particles in $\mathcal{X}_{t-1}$ are distributed according to R_{t-1}. We compute an intermediate set of particles $\bar{\mathcal{X}}_t$ by sampling a guess of the vehicle's pose and velocity at time t from the dynamics model (described in detail in Sect. 4.3.3). Thus, particles in $\bar{\mathcal{X}}_t$ are distributed according to the vehicle motion prediction distribution

$$\bar{R}_t := p(X^t, v^t | \mathcal{D}^{t-1}). \tag{4.10}$$

To ensure that particles in $\mathcal{X}_t$ are distributed according to R_t (asymptotically), we generate $\mathcal{X}_t$ by sampling from $\bar{\mathcal{X}}_t$ with replacement in proportion to importance weights given by $w_t = R_t / \bar{R}_t$. We compute the weights later in this section, but first, we need to derive the update equations for the geometry belief.

Updating Vehicle's Geometry Belief

We use a Gaussian approximation for the geometry belief S_t. Thus, we keep track of the mean μ_t and the co-variance matrix Σ_t of the approximating Gaussian in each particle. We have

$$\begin{aligned} S_t &= p(\Omega | X^t, v^t, \mathcal{D}^t) \\ &\propto p(\mathcal{D}_t | \Omega, X^t, v^t, \mathcal{D}^{t-1}) \;\; p(\Omega | X^t, v^t, \mathcal{D}^{t-1}) \\ &= p(\mathcal{D}_t | \Omega, X_t) \;\; p(\Omega | X^{t-1}, v^{t-1}, \mathcal{D}^{t-1}). \end{aligned} \tag{4.11}$$

The first step above follows from the Bayes' rule; the second step follows from the conditional independence assumptions of our model (Fig. 4.4). The expression in

(4.11) is a product of the measurement likelihood and the geometry prior S_{t-1}. To obtain a Gaussian approximation for S_t, we linearize the measurement likelihood as will be explained in Sect. 4.4.3. Once the linearization is performed, the mean and the co-variance matrix for S_t can be computed in closed form because S_{t-1} is already approximated by a Gaussian (represented by a Rao-Blackwellized particle from the previous time step).

Computing Importance Weights

Now we are ready to compute the importance weights. Briefly, following the derivation by Montemerlo [2003], it is straightforward to show that the importance weights w_t should be:

$$w_t = R_t/\bar{R}_t = \frac{p(X^t, v^t|\mathcal{D}^t)}{p(X^t, v^t|\mathcal{D}^{t-1})} = \mathbb{E}_{S_{t-1}}\Big[\ p(\mathcal{D}_t|\Omega, X_t)\ \Big]. \tag{4.12}$$

In words, the importance weights are the expected value (with respect to the vehicle geometry prior) of the measurement likelihood. Using Gaussian approximations of S_{t-1} and $p(\mathcal{D}_t|\Omega, X_t)$, this expectation can be expressed as an integral over a product of two Gaussians, and can thus be carried out in closed form. See Appendix B for details.

4.4.2 Motion Inference

As we mentioned in Sect. 4.3.1, a vehicle's motion is governed by two probabilistic laws: $p(v_t|v_{t-1})$ and $p(X_t|X_{t-1}, v_t)$. These laws are related to the motion prediction distribution as follows:

$$\begin{aligned}
\bar{R}_t &= p(X^t, v^t|\mathcal{D}^{t-1}) \\
&= p(X_t, v_t|X^{t-1}, v^{t-1}, \mathcal{D}^{t-1})\ p(X^{t-1}, v^{t-1}|\mathcal{D}^{t-1}) \\
&= p(X_t|X^{t-1}, v^t, \mathcal{D}^{t-1})\ p(v_t|X^{t-1}, v^{t-1}, \mathcal{D}^{t-1})\ R_{t-1} \\
&= p(X_t|X_{t-1}, v_t)\ p(v_t|v_{t-1})\ R_{t-1}.
\end{aligned} \tag{4.13}$$

The first and second steps above are simple conditional factorizations; the third step follows from the conditional independence assumptions of our model (Fig. 4.4).

Note that since only the latest vehicle pose and velocity are used in the update equations, we do not need to actually store entire trajectories in each particle. Thus memory storage requirements per particle do not grow with t.

4.4.3 Shape Inference

In order to maintain the vehicle's geometry belief in a Gaussian form, we need to linearize the measurement likelihood $p(\mathcal{D}_t|\Omega, X_t)$ with respect to Ω. Clearly, the measurement likelihood does not lend itself to differentiation in closed form. Thus, we turn to *Laplace's method* to obtain a suitable Gaussian approximation. We provide details in Appendix B. In short, the method involves fitting a Gaussian at the global maximum of a function. Since the global maximum is not readily available, we search for it via local optimization starting at the current best estimate of geometry parameters. Due to construction of our measurement model (Sect. 4.3.5), the search is inexpensive as we only need to recompute the costs for the rays directly affected by a local change in Ω.

The dependence between our belief of the vehicle's shape and its position (discussed in Sect. 4.3.2) manifests itself in a dependence between the local anchor point coordinates C and the vehicle's width and length. The vehicle's corner closest to the vantage point is a very prominent feature that impacts how the sides of the vehicle match the data. When revising the belief of the vehicle's width and length, we keep the closest corner in place. Thus, a change in the width or the length leads to a change in the global coordinates of the vehicle's center point, for which we compensate with an adjustment in C to keep the anchor point in place. This way, a change in geometry does not create phantom motion of the vehicle.

4.4.4 Initializing and Discontinuing Tracks

New tracks are initialized in areas where scan differencing detects a change in data that is not already explained by existing tracks. New tracks are fitted using the same measurement and motion models that we use for vehicle tracking (Sects. 4.3.5 and 4.3.3). The candidates are vetted for three frames before they can become "real tracks". Detection of new vehicles is the most computationally expensive part of vehicle tracking. In Sect. 4.5, we describe the techniques we used to achieve reliable vehicle detection in real time.

We discontinue tracks if the target vehicle gets out of sensor range or moves too far away from the road.(4) We also discontinue tracks if the unnormalized weights have been low for several turns. Low unnormalized weights signal that the sensor data is insufficient to track the target, or that our estimate is too far away from the actual vehicle. This logic keeps the resource cost of tracking occluded objects low, yet it still allows for a tracked vehicle to survive bad data or complete occlusion for several turns. Since new track acquisition only takes three frames, it does not make sense to continue tracking objects that are occluded for significantly longer periods of time.

4.5 Vehicle Detection

Accurate moving vehicle detection in laser range data requires three frames. The first two frames are required to detect motion of an object. The third frame is required to check that the motion is consistent over time and follows the vehicle dynamics law. Thus, for a 10Hz sensor the minimum vehicle detection time is 0.3 seconds.

Note that detection based on three frames allows for accurate results because we can observe two consecutive motion updates and verify that the observed motion is consistent with a moving vehicle. For some applications, it may be acceptable to sacrifice accuracy in favor of faster detection based on just one or two frames. For example, Wang et al. [2007] detects as "moving" all objects that appear in areas previously seen as empty. Often, this approach is adopted when the intention is to filter out moving obstacles to build a static map. However, for vehicle tracking, this approach leads to many false positives.

(4) A digital street map was available for our application in the *Road Network Definition Format* (*RNDF*).

4.5.1 The Basic Detection Algorithm

Our vehicle detection method proceeds in three stages:

1 First, a vehicle is fitted using importance sampling in an area where a change in the environment has been detected by scan differencing. The scoring is performed using the measurement model described in Sect. 4.3.5.

2 Next, the vehicle's velocity is estimated by performing a particle filter update step and scoring using the measurement model in the next frame.

3 During the last stage, another particle filter update is performed and scored against a third frame.

4.5.2 Challenges in Vehicle Detection

The range data in outdoor urban environments contains large amounts of noise that adds up from a number of sources. The limitations of horizontal scan resolution ($0.5°$ for IBEO and $0.1°$ for Velodyne) and vertical scan resolution ($0.4°$ for Velodyne) produce $40 - 50$cm noise at 60m range. Another source of noise is the laser beam spot size, which can exceed the scan resolution [Sick Optics, 2003]. Scanning the same vehicle at a slightly different height can result in $1-2$m range discrepancy. Additional noise comes from the virtual scan approximation (25cm at 60m range for $0.5°$ angular resolution) and the box model approximation of the vehicle's shape ($20 - 40$cm). Internal sensor construction, circuitry, and messaging time delays also produce noise, which is in general difficult to quantify. Studies have been performed for older sensors [Mäkynen, 2000, Blais, 2004], but this information is not yet available for the newer models of range finders. Finally, environmental factors such as dust and rain cause false readings many meters off the actual target.

For the driving application, we need to detect vehicles moving at 5mph to 35mph with a 10Hz sensor. Thus, a vehicle moves $20 - 150$cm per frame. This signal can be easily overwhelmed by noise especially in the lower range of the velocities. The poor signal-to-noise ratio makes it difficult to accurately tell a moving object apart from noise in just three frames.

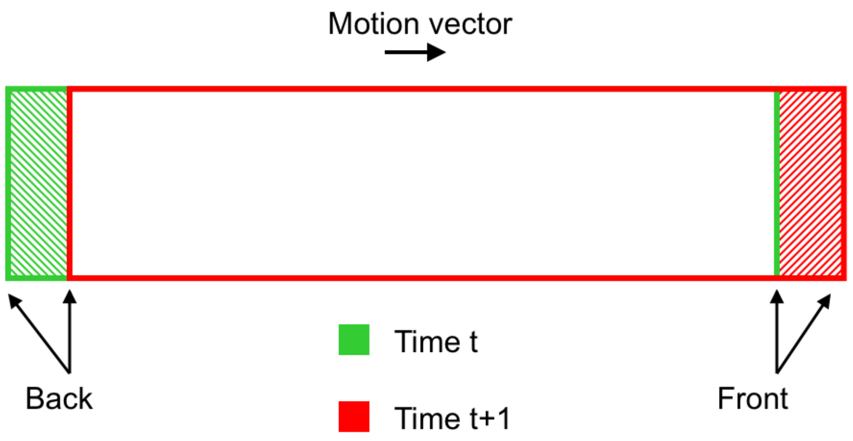


Figure 4.8: Diagram representing forward motion of a bus. Green color represents the position of the bus at time t. Red color represents its position at time $t + 1$. The green shaded area in the back of the bus frees up as the bus moves forward. The red shaded area in the front of the bus becomes occupied. Note that these changes are small compared to the overall area taken up by the bus, which remains occupied in both frames.

Although the signal is easier to detect if we use more than three frames, this solution is undesirable because it increases the detection time and takes up more computational resources. A more efficient approach, proposed by Wang et al. [2007], is to sample the frames at a lower rate (e.g., 1Hz), so that the signal is prevalent over the noise. However, this method also increases the total time required for detection of a vehicle and therefore it is unsuitable for our application.

4.5.3 Motion Evidence

To overcome the poor signal-to-noise ratio, we turn to the method used by humans to detect moving vehicles in noisy data. Consider a long bus moving forward at 5mph (Fig. 4.8). From one frame to the next, it travels 20cm — a negligible distance compared to the noise and overall size of the vehicle. Since the middle of the bus appears stationary, a human trying to discern motion will focus on the front and back of the bus, to see if there is at least a tad of motion.

To take advantage of the same method for vehicle detection, we define a score we call *motion evidence*. To compute this score, we consider the regions cleared by the vehicle as it moves. The cleared area behind the vehicle should be occupied in the prior frame and free in the current frame. Similarly, the area in front of the moving vehicle should be free in the prior frame and occupied in the current frame. Often, we can only observe the front or the back of the vehicle, thus only half of the evidence is available due to self-occlusion. To allow for self-occlusion and partial occlusions by other objects, we threshold the motion evidence score at 25%.

Note that the motion evidence score is different from the probabilities obtained by fitting a vehicle using a particle filter. The particle filter computes the probability that motion *"could have"* happened, whereas the motion evidence scores the motion that *"must have"* happened. In the bus example given above, the motion evidence score would ignore the entire bus except 20cm in the front and in the back.

The motion evidence score can be computed for any pair of consecutive frames. In our approach, we compute it for the first and the second pairs of frames and filter out vehicle candidates for which the score is below the threshold. Doing so provides a very dramatic decrease in false positives, without affecting the false negatives rate.

4.5.4 Optimizations

Since new vehicle detection is computationally expensive, we developed several optimizations to achieve reliable real time performance. We describe the optimization techniques below and evaluate their impact on the performance of vehicle detection in Sect. 4.7.2.

Scaling Series

The first step of vehicle detection involves fitting the geometric vehicle model to a virtual scan under conditions of large uncertainty: several meters in position and $360°$ in orientation of the vehicle. Using simple importance sampling with three state parameters makes the problem intractable within real time constraints.

To improve performance, we turn to the Scaling Series algorithm we described in Chapter 2. Briefly, the algorithm works by performing a series of successive refinements, generating an increasingly informative proposal distribution at each step of the series. The successive refinements are performed by gradually annealing the measurement model from artificially relaxed to realistic.

For the vehicle detection problem, we applied the Scaling Series algorithm to choose the proposal distribution for the initial importance sampling step. We obtained measurement model relaxations by inflating the width, d_{sur}, of the vehicle surface region (see Fig. 4.7). The normal setting for d_{sur} is 0.25m. The most relaxed model was obtained by padding the region by 1m on the inside and outside, resulting in an d_{sur} setting of 2.25m. At this setting the vehicle surface region expands to consume the free space region, and thus the penalty c_b is not applied. However, even with this coarse model, the algorithm quickly rules out vehicle candidates placed more than 1m away from the actual vehicle location. The resulting high likelihood region includes a region of 1m radius around the true position of the vehicle. As d_{sur} is gradually annealed(5) from 2.25m to 0.25m, the high likelihood region shrinks, resulting in a more and more informed proposal distribution.

In Sect. 4.7.2, we show that using this method, we obtained a very significant improvement in the reliability of the search and reduced the time it takes to detect a new moving vehicle by a factor of 10.

Road Masking

Since a digital road map is available in our application, one simple optimization is to restrict the search to the road regions. We do this by marking each data point as “close to road” or “far from road”. Only the points near the road are considered for new vehicle detection. This optimization greatly improves the efficiency of the vehicle detection algorithm.

Cleared Area

As we already discussed above, a change in the data can be caused by either noise or motion. Ultimately the motion evidence score will help disambiguate motion from noise. However, the motion evidence score can only be used after the vehicle model has already been fitted to data. To make the search more efficient, we would like to

(5)The annealing is done over $N = 10$ iterations with $zoom = 2^{-1/3}$.

distinguish between noise and motion before performing any model fittings.

When a vehicle moves forward with a minimum velocity v_{min} for a time interval Δt, it clears an area of approximately $v_{min}\Delta tW$. Thus, we can examine each data point to see if enough space has been cleared around it to allow for motion of a vehicle. If the vehicle is moving away from us, the cleared area will be in the current frame with respect to the prior frame. If the vehicle is approaching us, the cleared area will be in the prior frame with respect to the current frame. Thus, we can find both types of cleared area by performing a symmetric clearing operation between the two frames.

Even though cleared area logic is not as powerful as the motion evidence score, it provides a significant speed-up when used as a fast data pre-processing step.

Backward Search

Since vehicle detection takes three frames, the minimum detection time is 0.3 seconds for a sensor with a frame rate of 10Hz. It turns out that if we only search forward in time, then the minimum detection time is 0.4 seconds for approaching vehicles because the first frame is only used to detect dynamic data points in the second frame. However, if we fit the vehicle in the second frame and then move it backwards in time, we can utilize the first frame as well. In this case we use frame number two for the initial vehicle fitting and frame number one for velocity estimation. As before, the third frame is used to check motion consistency.

4.6 Working with 3D Range Data

As we explained in Sect. 4.3.4, vehicle tracking is a 2D problem, for which compact 2D virtual scans are sufficient. However, for 3D sensors, such as Velodyne, it is non-trivial to build consistent 2D virtual scans. These sensors provide immense 3D data sets of the surroundings, making computational efficiency a high priority when processing the data. In our experience, the hard work pays off and the resulting virtual scans carry more information than 2D sensor data.

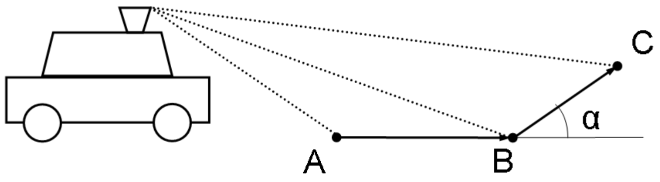


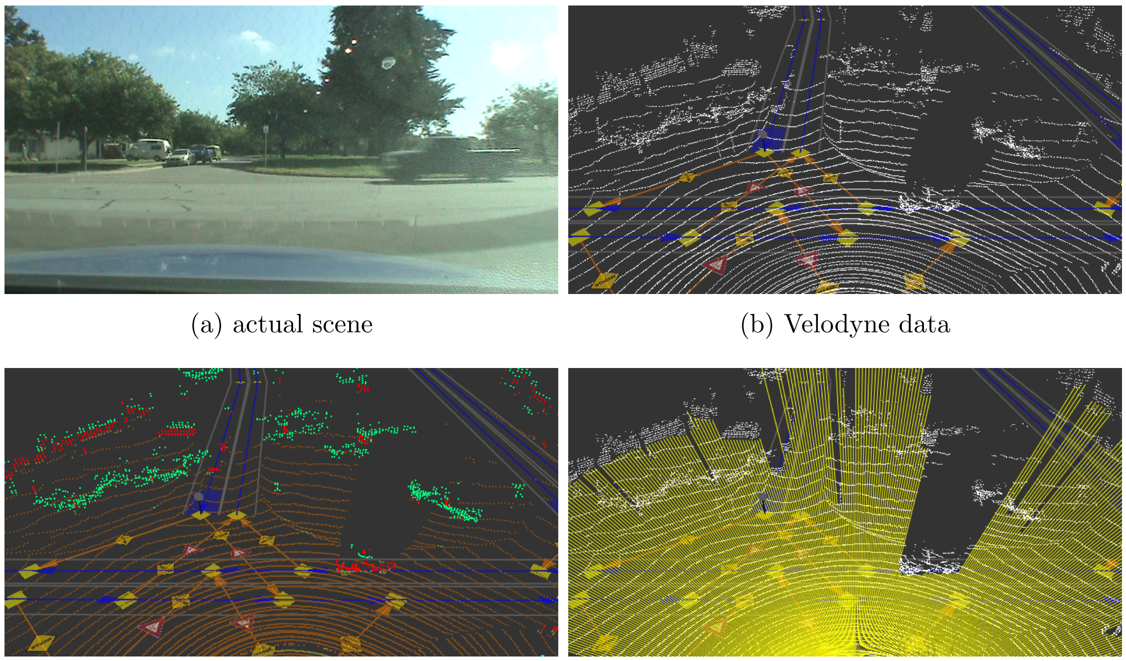
Figure 4.9: We determine ground readings by comparing angles between consecutive readings. If A, B, C are ground readings, then α is close to 0 and thus $\cos\alpha$ is close to 1.

To produce consistent 2D virtual scans, we need to understand which of the 3D data points should be considered obstacles. From the perspective of driving applications, we are interested in the slice of space directly above the ground and up to 2m high, as this is the space that a vehicle would actually have to drive through. Objects elevated more than 2m above ground — e.g., tree tops or overpasses — are not obstacles. The ground itself is not an obstacle (assuming the terrain is drivable). Moreover, for tracking applications, low obstacles such as curbs should be excluded from virtual scans because they can prevent us from seeing more important obstacles beyond them. The remaining objects in the 2m slice of space are obstacles for a vehicle, even if these objects are not directly touching the ground.

In order to classify the data into the different types of objects described above, we first build a 3D grid in spherical coordinates. Similarly to a virtual scan, it has a single point of origin and stores actual world coordinates of the sensor readings. Just as in the 2D case, this grid is an approximation of the sensor data set because the actual laser readings in a scan have varying points of origin. In order to downsample and reject outliers for each spherical grid cell we compute the median range of the readings falling within.(6) This gives us a single obstacle point per grid cell. For each spherical grid cell, we will refer to the cone from the grid origin to the obstacle point as a virtual ray.

The first classification step is to determine ground points. For this purpose, we select a single slice of vertical angles from the spherical grid (i.e., rays that all have the same bearing angle). We cycle through the rays in the slice from the lowest vertical

(6)In our implementation, the angular grid resolution for Velodyne based virtual scans is 0.5°, which results in three readings per angular grid cell on average — just enough to reject outliers.



(a) actual scene (b) Velodyne data

(c) after classification (d) generated virtual scan

Figure 4.10: In (c), Velodyne data is colored by type: orange — ground, yellow — low obstacle, red — medium obstacle, green — high obstacle. In (d), yellow lines denote the virtual scan. Note the truck crossing the intersection, the cars parked on a side of the road and the white van parked on a driveway. On the virtual scan, all of these vehicles are clearly marked as obstacles, but ground, curbs and tree tops are ignored.

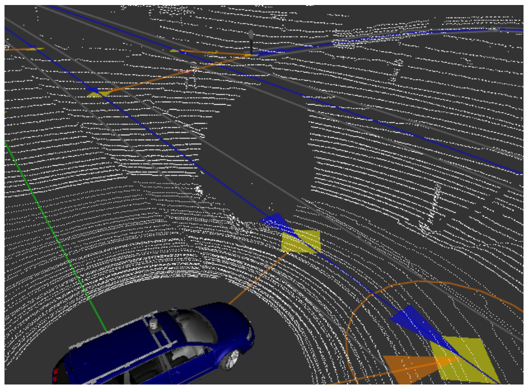
angle to the highest. For three consecutive readings A, B, and C, the slope between AB and BC should be near zero if all three points lie on the ground (see Fig. 4.9 for illustration). If we normalize AB and BC, their dot product should be close to 1. Hence, a simple thresholding of the dot product allows us to classify ground readings and to obtain estimates of local ground elevation. Thus, one useful piece of information we can obtain from 3D sensors is an estimate of ground elevation. A similar ground estimation method was independently developed by the MIT Urban Challenge team [Leonard et al., 2008].

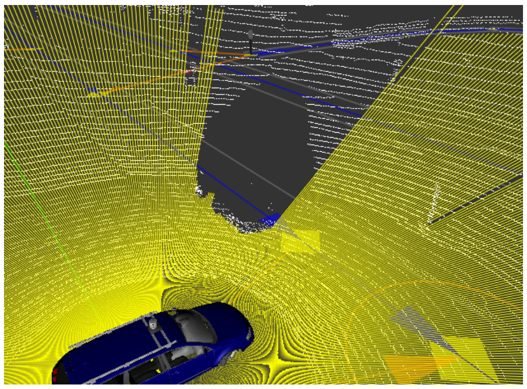
Using the elevation estimates, we can classify the remaining non-ground readings into low, medium and high obstacles, out of which we are only interested in the medium ones (see Fig. 4.10). It turns out that there can be medium height obstacles that are still worth filtering out: birds, insects and occasional readings from cat-eye reflectors. These obstacles are easy to filter because the BC vector tends to be very long (greater than 1m), which is not the case for normal vertical obstacles such as buildings and cars. After identifying the interesting obstacles, we simply project them on the 2D horizontal plane to obtain a virtual scan.

4.6.1 Detection of Black Obstacles

Laser range finders are widely known to have difficulty seeing black objects. Since these objects absorb light, the sensor never gets a return. Clearly, it is desirable to "see" black obstacles for driving applications. Other sensors could be used, but they all have their own drawbacks. Here, we present a method for detecting black objects in 3D laser data. Fig. 4.11 shows the returns obtained from a black car. The only readings obtained are from the license plate and wheels of the vehicle, all of which get filtered out as low obstacles. Instead of looking at the little data present, we can detect the black obstacle by looking at the absent data. If no readings are obtained along a range of vertical angles in a specific direction, we can conclude that the space must be occupied by a black obstacle. Otherwise the rays would have hit some obstacle or the ground. To provide a conservative estimate of the range to the black obstacle, we place it at the last reading obtained in the vertical angles just before the

(a) actual appearance of the vehicle


(b) the vehicle gives very few laser returns


(c) virtual scan with black object detection

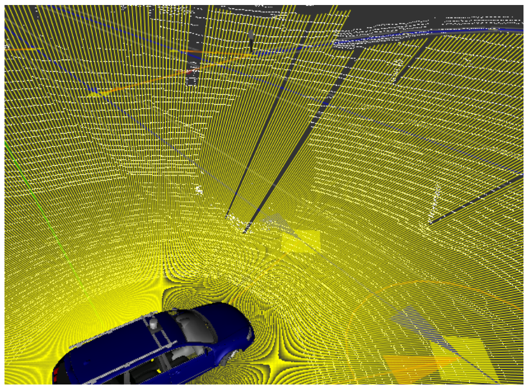
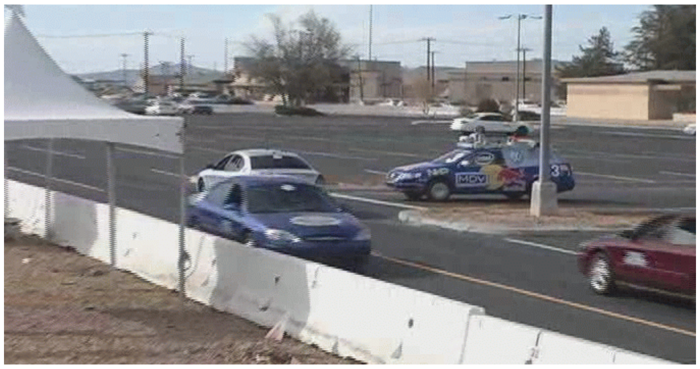
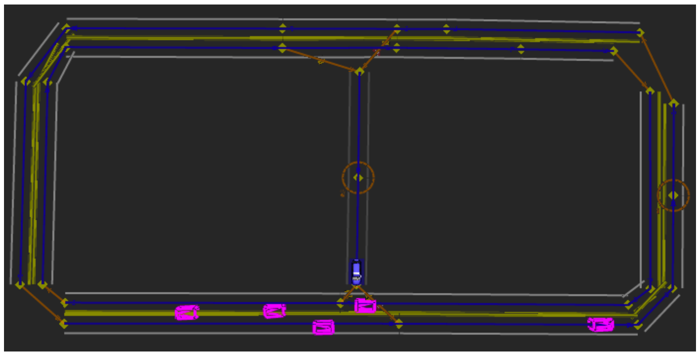

(d) virtual scan without black object detection

Figure 4.11: Detecting black vehicles in 3D range scans. White points represent raw Velodyne data. Yellow lines represent the generated virtual scans.



(a) traffic density



(b) course A outline

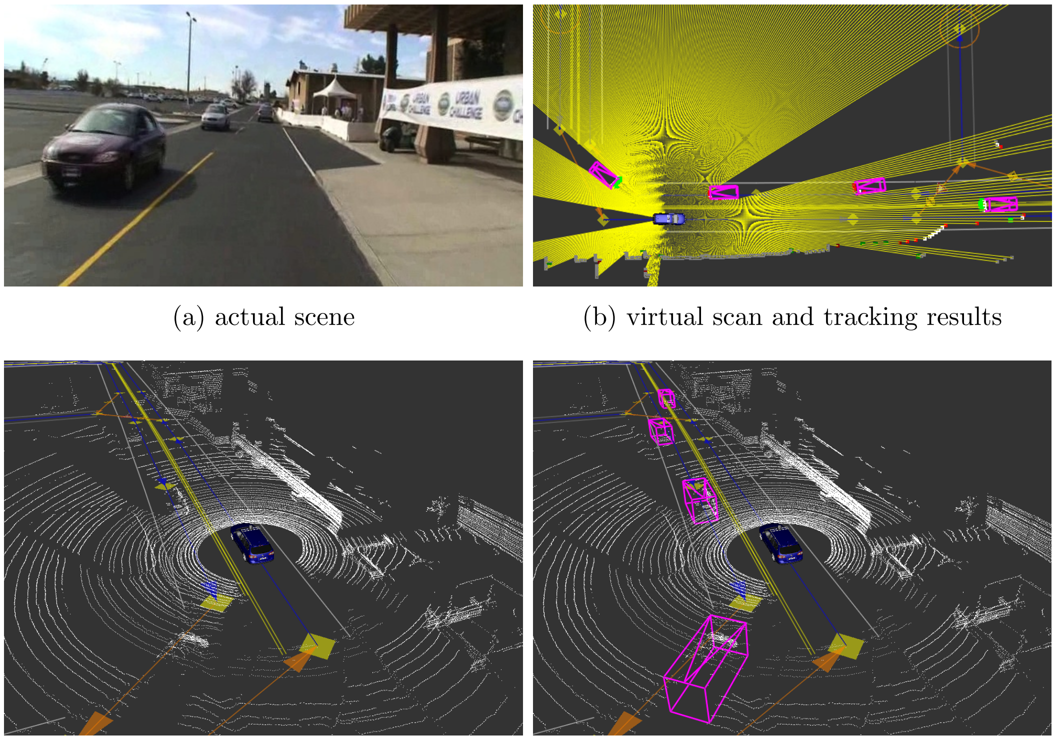
Figure 4.12: Test conditions on course A at the Urban Grand Challenge. The test consisted of repeated merges into dense traffic (a) on a course with an outline resembling the Greek letter θ (b).

absent readings. We note that this method works well as long as the sensor is good at seeing the ground. For the Velodyne sensor the range within which the ground returns are reliable is about $25 - 30$m, beyond this range the black obstacle detection logic does not work.

4.7 Experimental Validation

4.7.1 Tracking Results

The most challenging traffic situation at the Urban Grand Challenge was presented on course A during the qualifying event (Fig. 4.12) . The test consisted of dense human



(a) actual scene

(b) virtual scan and tracking results

(c) Velodyne data

(d) Velodyne scan and tracking results

Figure 4.13: Tracking results on course A at the UGC. In (b) yellow line segments represent the virtual scan and red/green/white points show results of scan differencing. The purple boxes denote the tracked vehicles.

Table 4.1: Tracker performance on data sets from three urban environments. Max TP is the theoretically maximum possible true positive percent for each data set. TP and FP are the actual true positive and false positive rates attained by the algorithm.

Data Sets	Total Frames	Total Vehicles	Correctly Id'ed	Falsely Id'ed	Max TP (%)	TP (%)	FP (%)
UGC Area A	1,577	5,911	5,676	205	97.8	96.02	3.35
Stanford	2,140	3,581	3,530	150	99.22	98.58	4.02
Alameda 1	1,531	901	879	0	98.22	97.56	0
Overall	5,248	10,393	10,085	355	98.33	97.04	3.3

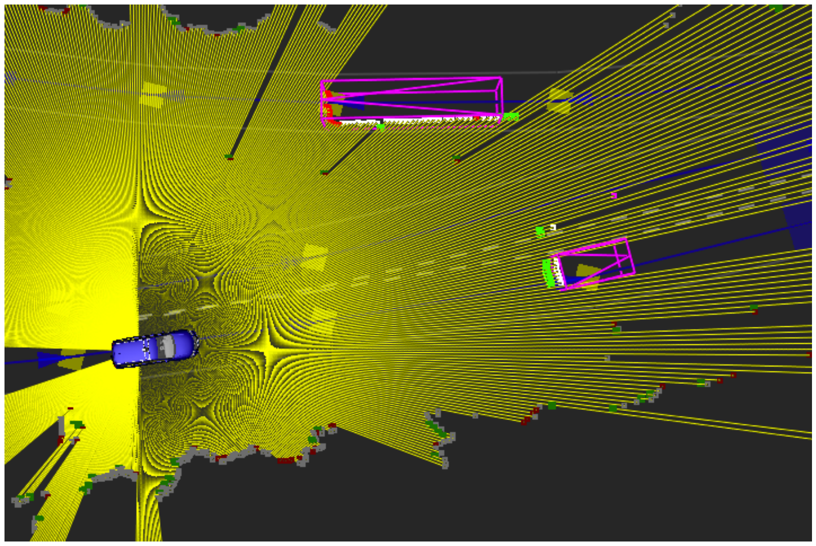


Figure 4.14: Shape estimation results on Stanford campus. Vehicles of different sizes are successfully estimated and tracked.

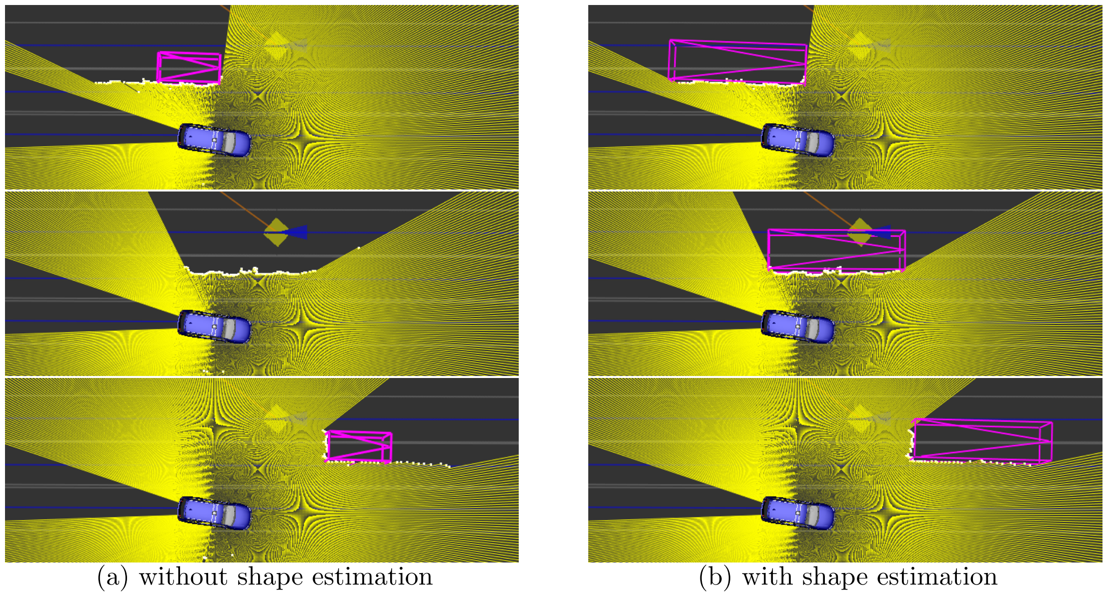


Figure 4.15: Shape estimation on the example of a passing bus from a data set taken in Alameda. Without shape estimation (a) the tracking results are poor because the geometric model does not fit the data well. Not only is the velocity estimated incorrectly, but the track is lost entirely when the bus is passing. With shape estimation (b) the bus is tracked successfully and the velocity is properly estimated.

driven traffic in both directions on a course with an outline resembling the Greek letter θ. The robots had to merge repeatedly into the dense traffic. The merge was performed using a left turn, so the robots had to cross one lane of traffic each time. In these conditions, accurate estimates of positions and velocities of the vehicles are very useful for determining a gap in traffic large enough to perform the merge safely. Vehicles passed in close proximity to each other and to stationary obstacles (e.g., signs and guard rails) providing plenty of opportunity for false associations. Partial and complete occlusions happened frequently due to traffic density. Moreover, these occlusions often happened near merge points which complicated decision making.

During extensive testing, the performance of our vehicle tracking module has been very reliable and efficient (see Fig. 4.13). Geometric shape of vehicles was properly estimated (see Figs. 4.14 and 4.15), which increased tracking reliability and improved motion estimation. The tracking approach proved capable of handling complex traffic situations such as the one presented on course A of the UGC. The computation time of our approach averages at 25ms per frame, which is faster than real time for most modern laser range finders.

We also gathered empirical results of the tracking module performance on data sets from several urban environments: course A of the UGC, Stanford campus and a port town in Alameda, CA. In each frame of data, we labeled the vehicles a human is able to identify in the laser range data. The vehicles had to be within 50m of the ego-vehicle, on or near the road, and moving with a speed of at least 5mph. We summarize how the tracker performed on the labeled data sets in Tbl. 4.1. Note that the maximum theoretically possible true positive rate is lower than 100% because three frames are required to detect a new vehicle. On all three data sets, the tracker performed very close to the theoretical bound. Overall, the true positive rate was 97% compared to the theoretical maximum of 98%.

4.7.2 Detection Results

To evaluate the performance of the vehicle detection algorithm empirically, we forced the tracking module to drop each target as soon as it was detected. We then ran

Table 4.2: Vehicle detector performance on data sets from three urban environments. For each car we counted how many frames it took to detect it. By construction of the algorithm, at least three frames are required. We also counted the number of false detections. The '% Detected' columns give the percentages of cars detected by frame three, four and five. 'FP %' is the false positive rate attained by the vehicle detection algorithm.

	Total	Detected in Frame			False	% Detected by Frame			FP
Data Sets	Cars	3	4	5	Det.	3	4	5	%
UGC Area A	713	596	103	14	1	83.6	98.0	100.0	0.1
Stanford	679	645	32	2	2	95.0	99.7	100.0	0.3
Alameda 2	532	485	45	2	5	91.2	99.6	100.0	0.9
Overall	1,924	1,726	180	18	8	89.7	99.1	100.0	0.4

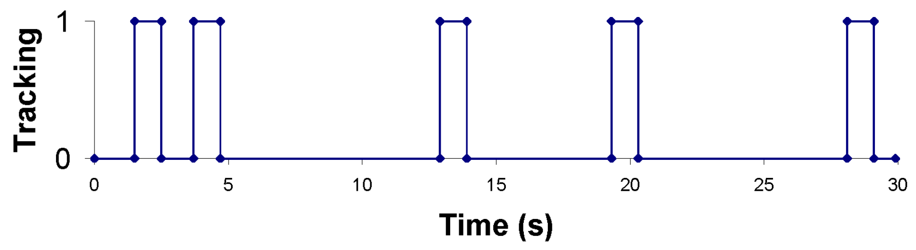
vehicle detection on data sets from three different urban environments: Area A of the Urban Grand Challenge qualifiers, the Stanford campus, and a port town in Alameda, CA (see Tbl. 4.2). In each frame of data, we labeled all vehicles identifiable by a human in the range data. The vehicles had to be within 50m of Junior, on or near the road, and moving with a speed of at least 5mph. For each vehicle, we counted how many frames it took to detect it. We also counted false positives. Overall, all vehicles were detected in five frames or less and the false positive rate was 0.4%.(7)

To evaluate motion evidence contribution, we ran the algorithm with and without motion evidence logic on labeled data sets. The use of motion evidence brought false discovery rate from 60% down to 0.4%. At the same time the rate of false negatives did not increase.

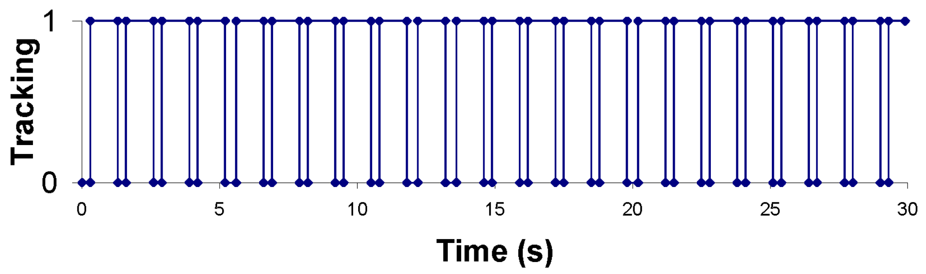
We used prerecorded data sets to evaluate performance gains from the optimization techniques. We compared the computation time of the algorithm with and without road masking. Road masking sped up the algorithm by a factor of eight. We also ran the algorithm with and without cleared area logic. The speed up from this optimization was approximately a factor of three. The backward search optimization reduced the minimum detection delay for oncoming traffic by 25%.

To evaluate improvements from Scaling Series, we used a 30 second data set of our ego-vehicle following another car. For evaluation purposes, we modified the tracker

(7)Note that the false positive rate is much lower for detection than for tracking. When a stationary object is falsely identified as moving by the detector, the tracker may keep it for many turns because we allow tracking of vehicles that came to a stop.



(a) standard PF



(b) Scaling Series

Figure 4.16: Comparison of standard PF with Scaling Series for new vehicle detection. The horizontal axis denotes time in seconds. The vertical axis has two states: 0 — target is not tracked, 1 — target is tracked. To verify target acquisition, the code was specifically modified to discontinue tracking a target after 1 second. By construction of the algorithm, the minimum possible time spent in non-tracking state is 0.3 seconds. (a) standard PF has a long target acquisition time — too dangerous for autonomous driving. (b) Scaling Series method has nearly perfect acquisition time.

to drop each target after tracking it for 1 second. Fig. 4.16 presents comparison of results obtained using a standard particle filter and Scaling Series. Vehicle detection with the standard particle filter took 4.44 seconds on average and 13.7 seconds in the worst case, which can easily result in a collision in a real life situation. In contrast the Scaling Series algorithm took 0.32 seconds on average to detect the vehicle, with the worst case being 0.5 seconds. Thus, the Scaling Series approach performs very close to the theoretical minimum of 0.3 seconds.

Several videos of vehicle detection and tracking using the techniques presented in this chapter are available at the website

http://cs.stanford.edu/people/petrovsk/uc.html

4.8 Conclusions

We have presented the vehicle detection and tracking module developed for Stanford's autonomous driving robot Junior. Tracking is performed from a high-speed moving platform and relies on laser range finders for sensing. Our approach models both dynamic and geometric properties of the tracked vehicles and estimates them with a single Bayes filter per vehicle. In contrast to prior art, separate data segmentation and association steps do not need to be carried out prior to the filtering step. The approach has proven to be reliable, efficient and capable of handling challenging traffic situations, such as the ones presented at the Urban Grand Challenge.

Our approach explicitly models tracked vehicle's geometric shape, which is estimated simultaneously with the vehicle's motion using an efficient RBPF method. The introduced anchor point notion allows us to correctly model the shape vs. motion ambiguity, previously unaddressed in vehicle tracking literature. This reduces motion uncertainty and improves the estimation of vehicle dynamics.

Unlike prior vehicle tracking approaches, which relied on features for tracking, we introduced a direct measurement model for range scans. This approach eliminates the need for data segmentation and association steps. Moreover, it naturally handles partial occlusions of the tracked vehicles, including situations where the vehicle scan is split up into multiple disjoint clusters due to occlusion.

We presented a number of optimization techniques to improve accuracy and efficiency of vehicle detection. These techniques are largely independent of each other. To aid the design decisions of future vehicle tracking approaches, we provided an analysis of how each technique influences the end result.

We presented techniques for efficient manipulation of 3D data clouds and construction of 2D virtual sensor models. The method relies on a ground estimation technique, which we expect to be applicable not only in urban environments but also in off-road settings with rugged terrain. The method purposefully ignores short obstacles in an effort to extract data useful specifically for vehicle tracking. As a result, detection and tracking of vehicles is unimpeded by curbs and short foliage present in urban settings, or even small rocks and rough features in completely off-road environments. It also ignores overhanging obstacles — such as trees, signs, and overpasses — if there is sufficient clearance for a vehicle to pass underneath. However, due to the fact that short obstacles are ignored, the presented data extraction method is not suitable for estimation of terrain drivability. For this reason, we used a separate method for detection of small hazards as described by Montemerlo et al. [2008].

We also introduced a method for detection of poorly visible black objects in 3D range data. This method is applicable not only for vehicle tracking but also for static mapping and collision avoidance. Moreover it can be extended to dark object detection using 3D range scanners in indoor settings.

There is ample room for future work in the field of perception for autonomous urban driving. The presented approach does not model pedestrians, bicyclists, or motorcyclists — a prerequisite for driving in populated areas. Another promising direction for future work is fusion of different sensors, including laser, radar and vision.

Chapter 5

Mobile Manipulation

5.1 Introduction

Many believe that general-purpose robots will soon inhabit home/office environments and carry out a large variety of tasks, for example, fetching items, delivering messages, or cleaning up a room. At a bare minimum, such robots must navigate in these environments as well as interact with them. In this chapter, we present a unified probabilistic approach to state estimation for simultaneous manipulation (of objects in the environment) as well as global navigation. Using this approach, we successfully enable a mobile manipulation platform to navigate from far away up to a door, manipulate the door handle so as to open the door, and enter an office while simultaneously continuing to manipulate the door. This work was done as part of the STAIR (STanford Artificial Intelligence Robot) project, which has the long-term goal of building a useful robotic assistant that can carry out home/office tasks such as those described above.

Over the last decade, probabilistic techniques have found great success in mobile robot navigation (e.g., Fox et al. [1999]). In much of this literature, the environment is modeled as static (unchanging), and there is no interaction between the robot and the environment. More recently, a number of authors have developed models for non-static environments. For example, Biswas et al. [2002], Anguelov et al. [2004, 2002], and Hähnel et al. [2003] use an off-line EM algorithm to differentiate between

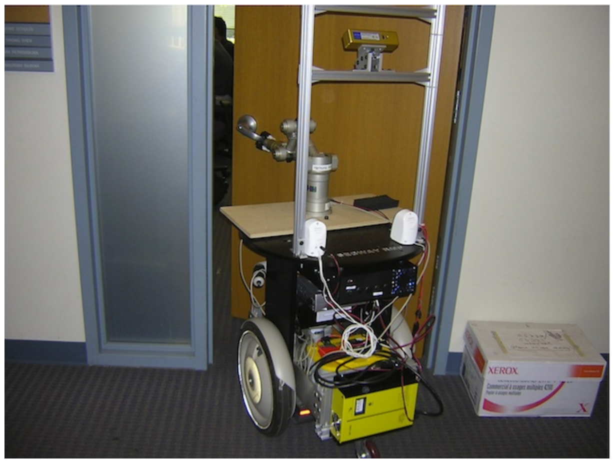


Figure 5.1: STAIR robot platform manipulating a door during one of our experiments.

static and non-static parts of an environment. A few algorithms also perform on-line mapping while taking into account non-static information. Wolf and Sukhatme [2004] use separate occupancy grids for dynamic obstacles (e.g., moving people) and static obstacles; Stachniss and Burgard [2005] maintain clusters of local grid maps corresponding to different observed configurations of the environment; and Biber and Duckett [2005] model temporal changes of local maps.

Robots typically interact with the environment using manipulators. Most work on manipulation focuses on properties of specific objects to be manipulated, rather than on moving in or understanding the global environment (e.g., Shekhar et al. [1986], Moll and Erdmann [2003]). With a few exceptions (e.g., Slaets et al. [2004] and our own work), most of this literature also does not have probabilistic basis, and thus at first glance, it appears difficult to derive a single unifying model that seamlessly integrates navigation and manipulation.

The task of mobile manipulation combines both navigation and manipulation. Most current work in mobile manipulation treats these as two tasks to be solved separately: first, mobile robotics techniques are used to navigate to a specific point; then, a separate set of techniques is used to localize objects to be manipulated. For example, in the context of door opening, navigation and manipulation of the door handle were considered by Rhee et al. [2004] and Petersson et al. [2000]. In both approaches, navigation to the door was performed as a separate task. The door handle is then localized only after the robot is already in front of the door, and a combination of visual servoing and tactile feedback is then used to grasp the door handle, and finally the door is opened by having the robot follow (with some compliance) a pre-scripted set of motions. Localization during door opening or while entering the doorway was not considered.

In this chapter, we present a unified, real-time, algorithm that simultaneously models the position of the robot within the environment, as well as the objects to be manipulated. It allows us to consider manipulation of large objects, such as doors, filing cabinets, and chairs. When the state of these objects changes, it significantly impacts navigation tasks. Thus, our goal is to simultaneously model a dynamic environment as well as localize ourselves within it. Because this objective is reminiscent to that of simultaneous localization and mapping (SLAM), we will find that we can borrow many ideas from SLAM [Thrun et al., 2005]. However, our objective is also different in two significant ways: first, the environment changes very significantly as we interact with (manipulate) it, and second, the precision required (1-5mm) for manipulation is 1-2 orders of magnitude higher than is typical for most SLAM applications.

Tested successfully on multiple doors, our approach enables our robot to navigate towards, manipulate, and move through a door (Fig. 5.1). In contrast to prior art on door opening, we are able to estimate parameters with high precision even during motion of the robot. Thus, no additional delay is required to locate the door handle once the robot reaches the door. Further, using the same, seamlessly integrated probabilistic model, the robot is able to precisely estimate the position of the door even while the robot and/or door are in motion, so that the robot can continuously manipulate the door even while it is passing through it.

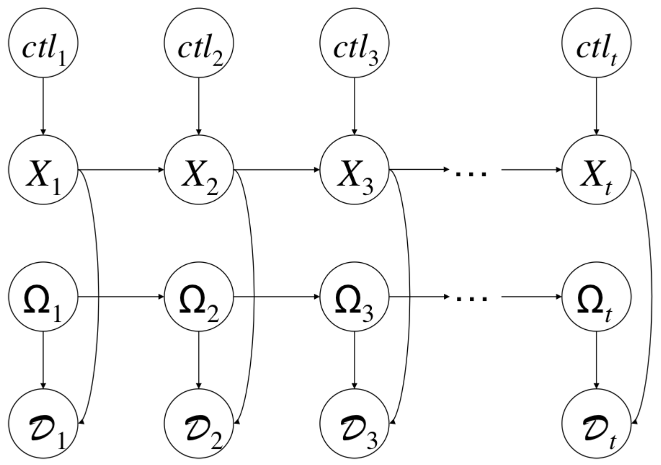


Figure 5.2: Dynamic Bayesian network model of the robot pose X_t, object state Ω_t, measurements $\mathcal{D}_t$, and controls ctl_t.

5.2 Representation

5.2.1 Probabilistic Model and Notation

One of our tasks is to determine the robot's position and orientation within an environment. We denote the robot's pose by $X = (x, y, \theta)$. In this chapter we will restrict our attention to manipulation of a single dynamic object placed in the environment. Concretely, consider an example where the position of the object is known (a reasonable assumption for doors, filing cabinets, elevators, etc.), but whose shape is governed by an object state parameter Ω. For example Ω could be the angle at which a door is open, or the extent to which a drawer is pulled out of a filing cabinet.(1)

At each time step t, we give the robot a new motion command, ctl_t, and obtain a new set of measurements $\mathcal{D}_t$ from its sensors. The robot pose and the object state evolve over time, and at time t are denoted by X_t and Ω_t respectively. We model X_t, Ω_t, $\mathcal{D}_t$, and ctl_t jointly using the dynamic Bayesian network shown in Fig. 5.2.

(1) In our door-opening application, $\Omega \in \mathbb{R}$ is a real number denoting the opening angle of the door. More generally, $\Omega \in \mathbb{R}^n$ could be vector-valued, when multiple parameters are needed to describe shape, configuration, position, and orientation of the object being manipulated. Note that this is similar to the vector-valued vehicle geometry description, which we used in Chapter 4. In fact, as we will see in Sect. 5.3, the derivations are almost identical.

In detail, given the robot pose and a new control, the pose evolves according to a probabilistic motion model derived from robot kinematics:

$$p(X_t|X_{t-1}, ctl_t).$$

The object state evolves according to:

$$p(\Omega_t|\Omega_{t-1}).$$

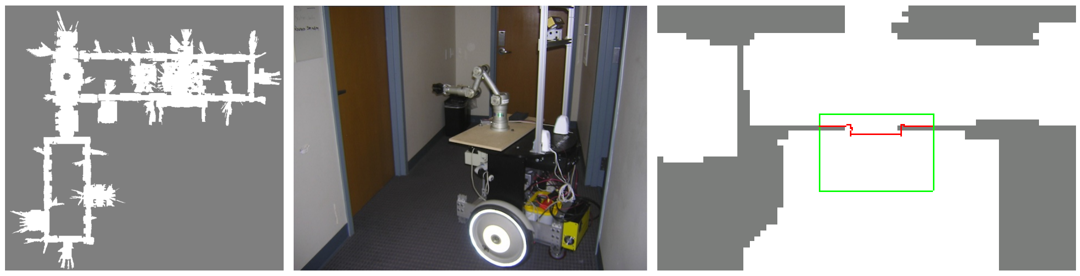
Similarly, sensor measurements are governed by a measurement model (discussed in Sect. 5.2.3 in more detail):

$$p(\mathcal{D}_t|\Omega_t, X_t).$$

We define the robot trajectory to be a vector of all robot poses up to the current time step, written $X^t = (X_1, X_2, ..., X_t)$. Similarly, we write $\mathcal{D}^t$ and ctl^t to denote all measurements and controls up to time t.

5.2.2 Representation of Environment

Following standard practice in mobile robot navigation, we represent the environment using an occupancy grid map (Fig. 5.3a) of the form typically constructed by mobile robots using SLAM. These coarse maps typically use grid cells that are 10cm x 10cm, and are thus well suited to navigation where 10cm resolution is acceptable. However, manipulation tasks require 1-5mm precision, and constructing a 2mm grid map of an entire building is clearly impractical—both from a memory storage point of view, and because these maps are typically built using noisy sensors. Consequently, we choose to use a combination of high and low resolution maps. We use a high resolution map only for the parts of environment (i.e., the objects) we are interested in manipulating. In this chapter, we use models comprising polygonal objects ("polygon models") to represent these objects at high resolution. This representation is well-suited to modeling doors, filing cabinets, straight walls, etc. Fig. 5.3c shows an example polygon model of a door together with the surrounding grid map. Our polygon models also allow us to model the changes in the shape of articulated objects (e.g., opening



(a) (b) (c)

Figure 5.3: (a) Occupancy grid map. (b) Actual environment. (c) Our representation: polygon model is "pasted" onto a grid map. The green box shows the bounding box of the polygon model; and the red lines show the polygon model.

doors or filing cabinets) in a very natural and efficient manner, simply by letting the position or orientation of some of the polygons be parameters.

Thus, a complete representation of the environment consists of a combination of an occupancy grid map and a polygon model. Further, the polygon model's shape is governed by object's state parameter Ω.

Our choice of this combination of models is motivated by our goal of having the robot be able to open any door (and enter any office) in our office building. Since all offices in our building are built on a common theme, all doors are essentially identical, and thus it suffices to build only a single polygon model of a door (via careful measurement). Wherever a door is present in the building, this same polygon model can then be rapidly "pasted" onto a 10cm-resolution grid map that has been built via standard SLAM techniques. This allows us to very rapidly acquire a map of the entire building, including 0.1cm-resolution models of all the doors in it.

5.2.3 Measurement Model

This section describes the measurement model $P(\mathcal{D}_t|X_t, \Omega_t)$ used in our approach. For this work, we focused on using a single sensor: a SICK laser scanner. Because our map comprises both low-resolution (10cm-grid cells) and high-resolution (1-mm, polygon model) components, we desire a measurement model that has a consistent interpretation regardless of the resolution at which the map is represented. Specifically, we know that the 10cm grid cells are inaccurate—the building walls, chairs,

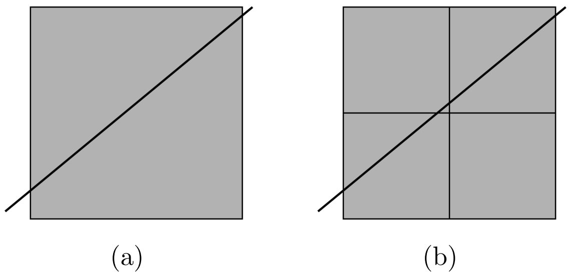


Figure 5.4: Example of a laser ray traveling through grid cells.

etc., are unlikely to be aligned with the global 10cm grid—and thus, we wish to model the 10cm grid map as probabilistically impeding the laser ray in a way that is noisier than the more precise polygon model.

In detail, the 10cm grids are usually only partially occupied, and thus there is a chance of the laser passing through it entirely. Thus, rather than simply modeling each grid cell as occupied or unoccupied, we will instead associate with each grid cell a probability that a laser ray terminates within that grid cell. Because our map has multiple resolutions, it is insufficient to associate with each grid cell a probability of that grid cell impeding the laser. To understand this, consider the toy map shown in Fig. 5.4, which we can choose to represent via either a low-resolution (10cm x 10cm) grid, or a higher-resolution (5cm x 5cm) grid. If we model a grid-cell as having a probability p of impeding a laser, then the chance of the laser being impeded by the 10cm x 10cm region in Fig. 5.4a is p, whereas the chance for the map on the right is p^3 (since it passes through three grid-cells). Clearly, it is undesirable that the measurement model change just because we chose to represent an object at a different resolution.

There are a variety of solutions to this, but we consider the most natural one to be the probabilistic interpretation of occupancy grid maps proposed by Eliazar and Parr [2004]. Their interpretation was motivated by the observation that the more naive model (using a fixed probability p for each grid cell) shows anomalous effects depending on the angle at which a laser travels relative to the grid lines, even if all the grid cells are the same resolution. However, the same interpretation turns out to also elegantly address our problem of using multi-resolution maps.

In this model, each cell of a grid map is associated with an *opacity* ζ.[2] The probability of a laser ray being interrupted by this cell depends both on the opacity and on the distance the ray travels within the cell. In detail, the probability of a ray terminating (i.e., being interrupted) while traveling from point A_1 to point A_2 in a medium of opacity ζ is defined as

$$P(terminate(\zeta, A_1, A_2)) = 1 - \exp\left(-\frac{||A_1 - A_2||}{\zeta}\right). \tag{5.1}$$

Immediately, we see that the probabilities of the ray terminating under the maps in Fig. 5.4a or 5.4b are the same, since the total distance is the same in either case (assuming that all the grid-cells have the same opacity ζ). Thus, this model allows us to give a consistent probabilistic interpretation to multi-resolution maps.

More generally, suppose a laser travels in a direction that (if it were unimpeded) would take it through N different regions in the map. Here, a "region" can be a grid cell (from the low-resolution map) or a polygonal region (from the polygon map), such as a polygon that represents the shape of a door, or one that represents part of the door-frame. We let $A_0, A_1, \ldots, A_N$ denote the points at which the laser ray would transition from one region to another (if it were to pass through all regions unimpeded), with A_0 denoting the origin of the laser ray. We also let $\zeta_1, \ldots, \zeta_N$ denote the opacities of these regions. The probability of the ray terminating within the i-th region is then

$$\begin{aligned} & P\left(terminate(\zeta_i, A_{i-1}, A_i)\right) \prod_{k=1}^{i-1} \left(1 - P(terminate(\zeta_k, A_{k-1}, A_k))\right) && (5.2) \\ &= \left(1 - \exp\left(-\frac{||A_{i-1} - A_i||}{\zeta_i}\right)\right) \prod_{k=1}^{i-1} \exp\left(-\frac{||A_{k-1} - A_k||}{\zeta_k}\right). && (5.3) \end{aligned}$$

This allows us to define the probability that the laser terminates at any specific range ρ. Finally, if the laser terminates at a certain range ρ, we model the actual observed

[2] The chance of a laser terminating/being interrupted is a decreasing function of ζ, so this parameter is perhaps better thought of as "transparency" rather than "opacity." However, for consistency we will use Eliazar and Parr [2004]'s terminology.

laser measurement D to be the range corrupted by Gaussian noise:

$$D = \rho + \epsilon, \tag{5.4}$$

where ϵ is a random variable $\sim \mathcal{N}(0, \sigma_r^2)$.

5.3 Inference

5.3.1 Rao-Blackwellization

We now describe an inference algorithm that reasons about the robot trajectory and the object state based on a set of measurements and controls. Specifically, we compute the following belief:

$$bel_t := p(\Omega_t, X^t | \mathcal{D}^t, ctl^t). \tag{5.5}$$

Note that the belief includes the entire robot trajectory up to time t, but only the latest object state. This choice turns out to be important for deriving an efficient filter. (This is a consequence of the fact that the current belief about the state of the door depends on the entire past trajectory. For example, this belief incorporates information about which of the past sensor measurements were directed at the door. A similar derivation is also used by Montemerlo [2003] and Murphy and Russell [2001].)

In detail, we apply a *Rao-Blackwellized particle filter* (*RBPF*), where each particle represents a guess of the entire robot trajectory and a belief about the object state S_t. Thus, we split up the full belief into two conditional factors:

$$bel_t = p(X^t | \mathcal{D}^t, ctl^t) p(\Omega_t | X^t, \mathcal{D}^t, ctl^t). \tag{5.6}$$

The first factor represents the robot trajectory belief:

$$R_t := p(X^t | \mathcal{D}^t, ctl^t). \tag{5.7}$$

The second factor represents object state belief, conditioned on the robot trajectory:

$$S_t := p(\Omega_t | X^t, \mathcal{D}^t, ctl^t). \tag{5.8}$$

The factor R_t will be approximated using a set of particles; the factor S_t, which estimates the angle of the door, will be approximated using a Gaussian distribution (one Gaussian per particle).

5.3.2 Robot Trajectory Estimation

Within each particle, we record a guess of the robot trajectory X^t, and a Gaussian approximation S_t to the object state. Let $\mathcal{X}_t$ denote the collection of particles at time t. We compute $\mathcal{X}_t$ recursively from $\mathcal{X}_{t-1}$. Suppose that at time step t, particles in $\mathcal{X}_{t-1}$ are distributed according to R_{t-1}. We compute an intermediate set of particles $\bar{\mathcal{X}}_t$ by sampling a guess of robot pose at time t from the motion model. Thus, particles in $\bar{\mathcal{X}}_t$ are distributed according to the robot trajectory prediction distribution

$$\bar{R}_t := p(X^t | \mathcal{D}^{t-1}, ctl^t). \tag{5.9}$$

To ensure that particles in $\mathcal{X}_t$ are distributed according to R_t (asymptotically), we generate $\mathcal{X}_t$ by sampling from $\bar{\mathcal{X}}_t$ with replacement in proportion to importance weights given by $w_t = R_t / \bar{R}_t$. Sect. 5.3.4 explains how these weights are computed. For now, we note that since only the latest robot pose is used in the update equations, we do not need to actually store entire trajectories in each particle. Thus the memory storage requirements per particle do not grow with t.

5.3.3 Object State Estimation

We use a Gaussian/extended Kalman filter (EKF) approximation to estimate the object state belief, S_t. Thus we keep track of the mean μ_t and variance σ_t of the approximating Gaussian in each particle.

Since S_t involves only the latest object state Ω_t (and not the entire object state history Ω^t), storage and computation requirements here also do not grow with t. We have:

$$\begin{aligned} S_t &= p(\Omega_t | X^t, \mathcal{D}^t, ctl^t) \\ &\propto p(\mathcal{D}_t | \Omega_t, X^t, \mathcal{D}^{t-1}, ctl^t) \;\; p(\Omega_t | X^t, \mathcal{D}^{t-1}, ctl^t) \\ &= p(\mathcal{D}_t | \Omega_t, X_t) \;\; p(\Omega_t | X^t, \mathcal{D}^{t-1}, ctl^t). \end{aligned} \tag{5.10}$$

The first step above follows from the Bayes' rule; the second step follows from the conditional independence assumptions of our model (Fig. 5.2). The final expression in (5.10) is a product of a measurement likelihood term and an object state (dynamical model) prediction term, which is defined (similarly to $\bar{R}_t$) as

$$\begin{aligned} \bar{S}_t &:= p(\Omega_t | X^t, \mathcal{D}^{t-1}, ctl^t) \\ &= \int_{\Omega_{t-1}} p(\Omega_t | \Omega_{t-1}) p(\Omega_{t-1} | X^{t-1}, \mathcal{D}^{t-1}, ctl^t) \, \mathrm{d}\Omega_{t-1}. \end{aligned} \tag{5.11}$$

Because $p(\Omega_{t-1} | X^{t-1}, \mathcal{D}^{t-1}, ctl^t)$ is already approximated as a Gaussian (represented by a Rao-Blackwellized particle from the previous timestep) and we use a linear-Gaussian model for $p(\Omega_t | \Omega_{t-1})$, we can easily compute the mean and variance of $\bar{S}_t$ above in closed form. Similarly, by using a Laplace approximation to obtain a Gaussian approximation to the measurement model $p(\mathcal{D}_t | \Omega_t, X_t)$ and using (5.10), we can also compute the mean and variance of S_t in closed form.

5.3.4 Computing Importance Weights

Briefly, following the derivation by Montemerlo [2003], it is straightforward to show that the importance weights w_t should be

$$w_t \;=\; R_t / \bar{R}_t = \frac{p(X^t | \mathcal{D}^t, ctl^t)}{p(X^t | \mathcal{D}^{t-1}, ctl^t)} = \mathbb{E}_{\bar{S}_t} \left[\, p(\mathcal{D}_t | \Omega_t, X_t) \,\right]. \tag{5.12}$$

In words, the importance weights are the expected value (over the object state prediction distribution) of the measurement likelihood. Since the two terms $p(\mathcal{D}_t | \Omega_t, X_t)$ and $\bar{S}_t$ are already approximated as Gaussians (as described in Sect. 5.3.3), this expectation can be expressed as an integral over a product of two Gaussians, and can thus be carried out in closed form.

5.3.5 Using Scaling Series to Increase Precision

For mobile robot navigation, 10cm precision is usually acceptable and gives reasonable performance. Thus in deployed implementations, it is fairly common practice to assume a large laser noise variance and use only a few laser rays per measurement update, which results in an approximation to the next-state R_t that has fairly large variance. However, to perform manipulation tasks, we require sub-centimeter precision in localization. Achieving this requires that we use most of the laser measurements in every update, and assume a realistic (small) variance in the laser readings. This results in a very peaked measurement model, in which most of the probability mass of our robot's position estimate is supported by a very small region (of perhaps 1-5mm diameter). A consequence of this is that it becomes difficult during the usual importance sampling step to draw a sufficient number of particles from this region to represent it well.

In the tactile localization application of Chapter 2, we also had the similar problem of a very sharply peaked measurement model. We proposed the Scaling Series algorithm to efficiently produce a much more informed proposal distribution, one that is concentrated around the areas of high probability mass. We refer the reader to Chapter 2 for details on Scaling Series, but briefly, the algorithm works by performing a series of successive refinements, generating an increasingly informative proposal distribution at each step of the series. The successive refinements are performed by gradually annealing the noise variance parameter within the measurement model from an artificially high value down to a realistic variance setting.

In the mobile manipulation setting, we applied the Scaling Series algorithm to choose the proposal distribution on each step of importance sampling in our particle filter. To do this, we annealed the measurement noise variance parameter σ_r in (5.4) and performed a series of successive refinements. This resulted in a much more informed proposal distribution, which allowed us to perform localization using only about 100 particles per step.

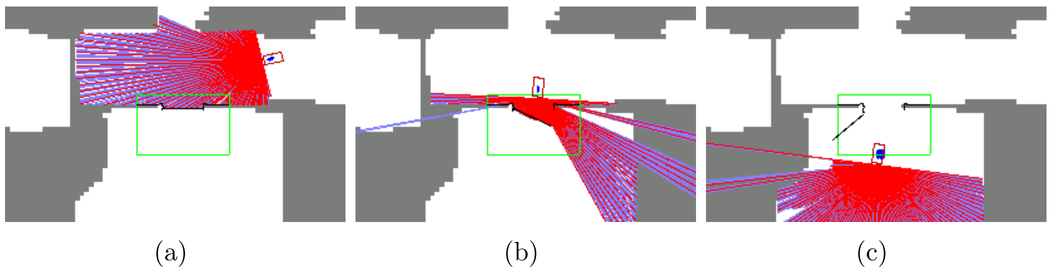


Figure 5.5: Robot localization with estimation of door state. The robot is denoted via the small rectangle; the rays emanating from the robot show the laser range scans; and the estimated door angle is also shown in the figures. The sequence of three images show the robot approaching, opening, and having passed through the door.

5.4 Experimental Results

We apply our algorithm to the task of manipulating door handles and doors. The STAIR (STanford Artificial Intelligence Robot) project is an ambitious, long-term (10-15 year) project that seeks to build a useful home/office robotic assistant. Thus, the ability to use doors and enter offices and conference rooms is of great practical importance to the robot.

We obtained 10cm resolution occupancy grid map by using standard SLAM algorithms with a mobile robot equipped with a laser. (See Fig. 5.3a.) Further, as discussed in Sect. 5.2.2, because all doors in our building are essentially identical, it is possible to build a single precise polygon model of a door, and then rapidly "paste" the same model into the grid map at all places where a door exists. Our polygon model includes the door itself, the door frame, and a small surrounding region, and also encodes the position of the door hinge. (Figs. 5.3b and 5.3c show a door and its polygon model representation.) Although not part of the polygon model, we note that the door handle is also at a fixed (known) position relative to the surface of the door, and can thus be straightforwardly computed if the position of the door (including the opening angle Ω of the door) is known.

The STAIR mobile manipulation platform comprises a Segway mobile platform, a Harmonic Arm manipulator, and a SICK laser. The arm has a fairly limited operational range (workspace), and has barely enough power to turn the door handles—it is able to do so only from certain configurations, where the load is spread more evenly among its motors—and thus there is only a very small 3cm x 3cm region from where the robot is physically capable of opening the door. Even within this region, localization accuracy of about 1-5mm is necessary to correctly grasp, turn, and manipulate the handle.

Several videos showing results of the robot opening a door are available at the website

http://cs.stanford.edu/people/petrovsk/stair/

The robot navigates to the door, turns the handle and opens the door slightly; then as it is moving through the door, the arm continues to manipulate the door by continuously pushing it open in front of the robot. Our state estimation algorithm is used in real-time to continuously estimate the position of the robot and the opening angle of the door, and thereby control the arm to continuously push the middle of the door. Even though the map changes drastically each time the robot opens the door and moves through it (see Fig. 5.5), in our experiments, the proposed approach invariably gives precise state estimates and results in successfully manipulating and navigating through the door. Tested 12 times (3 times on each of 4 doors), the algorithm succeeded each time in giving sufficiently accurate state estimates to open, continuously manipulating, and move through the door.

Although our algorithm allowed us to solve the practical problem of going through doors in our building, we now also present a more formal evaluation of its performance.

For the experiments presented below, we collected several minutes of laser and odometry data of the robot's approach towards a door in twelve distinct test situations. We considered 4 different doors (in different parts of the building); we considered each door in 3 different positions (closed, open, half-open). The purpose of this set of experiments was to test our approach against others in identical conditions. To ensure fairness of comparison, the same real-time computation requirements were imposed on all three algorithms. Since the algorithms considered are non-deterministic, we ran each algorithm 10 times for each dataset. To give a quantitative evaluation, we computed (for each algorithm) the root-mean-square (RMS) error with respect to ground truth among 10 runs for each test situation, and then averaged over all

	Algorithms		
Datasets	AMCL	RBPF	RBPF-SS
1	20.158cm	1.639cm	0.239cm
2	29.557cm	0.599cm	0.392cm
3	52.167cm	1.335cm	0.272cm
4	04.582cm	1.263cm	0.115cm
5	14.956cm	1.668cm	0.446cm
6	88.873cm	2.597cm	0.550cm
7	04.653cm	1.481cm	0.253cm
8	08.925cm	1.329cm	0.102cm
9	08.993cm	1.610cm	0.172cm
10	13.589cm	3.052cm	0.499cm
11	05.921cm	1.524cm	0.273cm
12	98.063cm	1.962cm	0.560cm
Overall RMS	42.975cm	1.779cm	0.358cm

Table 5.1: Positioning RMS error comparison of three localization algorithms: Adaptive MCL (AMCL), Rao-Blackwellized Particle Filter (RBPF), and the full proposed algorithm (RBPF-SS). Each of the 12 experiments shown was an average over 10 runs of each algorithm; the final row shows the overall RMS error.

twelve test situations (summarized in Tbl. 5.1). For the ground truth, we used the maximum a posteriori estimate of the door and robot position using a fine (2mm) grid within a 10cm area around the robot's final position in each test case.

To provide a baseline comparison, we used the *Adaptive MCL* (*AMCL*) localization algorithm implemented in Player by Gerkey et al. [2003]. This implements the *KLD MCL* method proposed by Fox [2001], and uses a 10cm occupancy grid map. In agreement with results reported by Stachniss and Burgard [2005], we noticed that if the actual door state does not correspond to the mapped door state, the robot gets "lost," which manifests itself as increased positioning error. If the door state does correspond to the map, the robot is able to localize with a RMS error of 12.6cm.

We also tested an "intermediate" algorithm that uses the polygon map and grid map combination, also using a Rao-Blackwellized particle filter, but without the Scaling Series algorithm to choose its proposal distributions. Empirically, this algorithm is able to localize and estimate the door state fairly accurately. The RMS error of robot pose in this scenario was 1.78cm, which is insufficient accuracy for manipulating the door handle.

Our full algorithm, using the Scaling Series proposal distributions (and the same update rate as the intermediate algorithm) is able to estimate robot pose with an RMS error of about 3mm. Using this algorithm, we were able to reliably open multiple doors.

5.5 Discussion and Conclusions

One frequently discussed difficultly of Rao-Blackwellized algorithms, specifically of FastSLAM, is that of extinction of particles. In FastSLAM, if a robot does not visit part of a map for a long time, then because the map is *static*, through the normal death of particles there will be very little diversity in its representation of the belief about that part of the map. Less formally, the algorithm becomes overly confident in its estimate of the map, which makes it difficult for the robot to accurately estimate that part of the map if it later returns to it. (See discussion by Montemerlo [2003].) In contrast, because we are estimating a *dynamic* parameter—namely the opening angle of the door, which is modeled as a dynamic, changing, variable—this is not a problem for our algorithm. Specifically, if the robot wanders away from the door for a long time, then its belief of the door state will converge to its stationary (uncertain) distribution, and thus the particle filter will correctly capture the fact that we should be very uncertain about the state of a door that we have not seen for some time.

One possible direction for future work is consideration of highly crowded environments. While we did have occasional passersby during our experiments (and our algorithm was resilient to these effects), overall the amount of unmodeled effects was low and the map provided a good representation of the environment. Unmodeled effects can be considerably more frequent in highly crowded or cluttered environments, e.g., high traffic public areas such as a museum or a cinema theater. These environments have been considered (from a mobile robot localization perspective) by Fox et al. [1999], who proposed a range data filtering technique to improve robustness of localization. Similar techniques can be added to our algorithm to increase robustness for mobile manipulation in these environments.

While designing the algorithm, we also had in mind the applications of manipulating (opening/closing) a filing cabinet sliding drawer, and moving a piece of furniture (e.g., a chair). Either of these fit into our framework very naturally (Ω = drawer position; or Ω = chair position), and we believe our approach will extend straightforwardly to such applications as well.

In summary, we have presented a single, unified, probabilistic model for simultaneously localizing a mobile manipulator robot and estimating the state of an object being manipulated. Our algorithm uses a combination of a high- and a low-resolution map, and was successfully applied to door manipulation, a task which requires very precise state estimation.

Chapter 6

Guaranteed Inference

6.1 Introduction

Accurate and reliable state estimation is required for safe and dependable robot operation in human environments, where uncertainty is high due to inherent unpredictability. However, reliable estimation can be challenging in this setting because the state has to be inferred from non-linear *relative sensors*. These sensors include the most common types used in robotics: laser, vision, and tactile. In these cases, the underlying probability distribution (the belief) is highly complex with many discontinuities and narrow peaks. The complexity is due to the properties of the sensors: occlusion boundaries cause discontinuities and high sensor accuracy leads to narrow peaks. Since the sensors are relative, the belief estimation problem becomes highly non-trivial whenever large initial uncertainty must be considered. In Chapter 1, we termed beliefs exhibiting this type of complexity *high-roughness beliefs* due to their similarity to rough terrain with many sharp transitions and narrow peaks. Although a variety of Monte Carlo and deterministic methods have been developed for non-linear problems [Doucet and De Freitas, 2001, Evans and Swartz, 2000], the reliability of these methods for high-roughness beliefs degrades as initial uncertainty increases. Methods that can provide guaranteed results for these beliefs are virtually nonexistent.

Due to the challenge posed by high-roughness beliefs, two main paths have been pursued in order to simplify the problem. The first method is to reduce the roughness by smoothing the measurement model. This method is popular in the indoor localization literature [Thrun et al., 2001, Gerkey et al., 2003, Plagemann, 2008]. However, sharp transitions in the belief actually contain very accurate information, which is discarded by smoothing, thus, leading to inaccurate estimates and increased ambiguity. The second method is to reduce the initial uncertainty. This method is very popular in visual tracking systems, where the state has to be initialized manually as documented in a recent study of 3D visual tracking literature by Lepetit and Fua [2005]. However, as the study concludes, methods incapable of dealing with global uncertainty tend to be inherently fragile because they can not recover from tracking failures.

In this chapter, we present an inference method suitable for moderately dimensional problems, in which the state has to be recovered from global uncertainty based on a set of non-linear relative sensor measurements. The method is a variation of an *adaptive grid algorithm* [Genz and Kass, 1997, Burgard et al., 1998], in which estimates of belief variation are used to drive the grid refinement process. Unlike [Genz and Kass, 1997, Burgard et al., 1998], we rely on analysis of the measurement model to produce *sound* bounds on the variation of the belief. As a result, our approach *guarantees* that all global optima are found and provides *provable* approximation error bounds. The method is capable of handling arbitrarily rough beliefs without discarding information. It can even handle the extreme case of perfect sensors, known to break most popular belief estimation methods. Thus, the method allows the robot to make the most of its sensors by extracting maximum amount of information contained in the sensor data.

We have termed the approach *Guaranteed Recursive Adaptive Bounding* (*GRAB*). It is applicable to a variety of pose estimation problems with relative sensors. We demonstrate its generality on two basic competency tasks: indoor navigation and tactile manipulation, where it drastically outperforms state-of-the-art. In our experiments, GRAB increased decision safety to 100%. Empirical evaluation shows that the performance of GRAB scales logarithmically with the desired approximation precision, allowing for efficient high-accuracy estimation. This property is due to the fact that the approach falls within the class of *divide-and-conquer methods*.

In indoor localization experiments, the approach led to 1mm accuracy of pose estimation based on the commonly used laser range finders. This high accuracy is useful for accurate maneuvering in tight spaces and is sufficient for reliable manipulation of stationary objects of interest within the environment (e.g., door handles, elevator buttons, light switches, etc.) as we have demonstrated in Chapter 5. It also opens up new potential applications during building construction, inspection and maintenance. In the tactile manipulation setting, the method results in efficient and reliable 6DOF object pose estimation with sub-millimeter accuracy, allowing for reliable manipulation.

Compared to the Scaling Series algorithm presented in Chapter 2, GRAB provides guaranteed results and can handle discontinuous beliefs, which commonly arise in indoor localization. On the other hand, Scaling Series may perform better for differentiable beliefs, such as the tactile object localization problem. We provide empirical comparisons between GRAB and Scaling Series in Sect. 6.5.

6.2 Related Work

Methods for belief estimation with non-linear measurements can be divided into deterministic and Monte Carlo classes. Deterministic methods include *uniform grid* (UG) and, for problems with dynamics, *histogram filter* [Thrun et al., 2005]. Adaptive grid methods have been developed to improve efficiency by concentrating samples near narrow modes. One of the earliest adaptive grid methods is *subregion adaptive integration* (*SAI*), which has been shown by Genz and Kass [1997] to outperform Monte Carlo methods in moderate dimensions. SAI selectively subdivides regions based on their estimated contribution to the cumulative integration error. The key difficulty here (and in other adaptive grid methods) is estimating the variation of the belief within a subregion. SAI estimates the variation by evaluating the function at selected points. This method is error prone and tends to underestimate the variation. Genz and Kass specifically warn that for functions with narrow peaks it is possible to miss the peaks entirely. An adaptive grid algorithm has also been applied to robot localization with laser range finders by Burgard et al. [1998], but similarly to SAI it did not provide any guarantees. A guaranteed adaptive grid algorithm has been developed by Olson [2000] for robot localization based on stereo sensors in the context of planetary exploration. However, it did not estimate the full belief — only its maximum. Olson's approach restricted estimation to just x and y coordinates, obtaining robot orientation from a compass. Moreover, it utilized a *likelihood field measurement model* [Thrun et al., 2005], which discards *negative information*. This model is smooth and induces a smooth belief, but can lead to frequent mis-localizations in the cluttered indoor environments.

Monte Carlo methods include variants of *importance sampling* and, for problems with dynamics, *particle filter* [Doucet and De Freitas, 2001]. In the mobile robot navigation domain, these algorithms are called *Monte Carlo localization* (*MCL*) [Thrun et al., 2005]. One of the most widely used adaptive Monte Carlo methods is *Adaptive Monte Carlo Localization* (*AMCL*) by Fox [2003]. It has been implemented in modern mobile navigation suites by Gerkey et al. [2003] and Willow Garage [2009]. AMCL adapts the number of particles over time by considering the KL-divergence of the resulting approximation. Recent work by Liu et al. [2008] has improved AMCL to better maintain multi-modality during pose tracking by adding spatial clustering. For reliable global localization, all variants of MCL require a large number of samples (at least initially). For this reason, some approaches suggest injecting samples directly from the measurement model [Lenser and Veloso, 2000, Thrun et al., 2001], although these methods require the availability of a measurement model from which one can easily and efficiently draw samples. A number of smoothing techniques are often applied: e.g., inflation of the measurement noise and/or subsampling of sensor data [Thrun et al., 2001, 2005]. Since smoothing discards information, recent approaches improve global localization by learning more sophisticated smoothed models with Gaussian processes [Plagemann, 2008].

The progress is much slower in tactile manipulation, due to complexity of experimental setup and relatively poor availability of tactile sensors. Belief estimation literature for this problem relies primarily on variants of particle filters [Gadeyne and Bruyninckx, 2001, Chhatpar and Branicky, 2005, Petrovskaya et al., 2006]. Since solving a global uncertainty problem in 6DOF with a particle filter is computationally expensive, some authors restrict the problem to 3DOF thus reducing the initial uncertainty [Gadeyne and Bruyninckx, 2001, Chhatpar and Branicky, 2005]. In 2006, we introduced the Scaling Series algorithm, which is described in detail in Chapter 2. Scaling Series is similar to the technique we describe in this chapter. It relies on graduated annealing to eventually estimate an un-smoothed belief. Although, the Scaling Series algorithm is not able to provide guarantees, it may work better for differentiable problems.

Belief propagation (*BP*) and *message passing* methods have been developed to take advantage of the factorization structure present in some belief estimation problems. These methods can often be more efficient than direct belief estimation methods and are capable of handling high dimensional problems. Kozlov and Koller [1997] proposed a *non-uniform discretization BP* method (*NUBP*), which combines BP with adaptive gridding . Similar to SAI, NUBP estimates function variation over subregions by random sampling. As for SAI, this can lead to entirely inaccurate belief approximations. Recently, the interest in adaptive gridding has been renewed by Isard et al. [2008]. Isard's approach is similar to NUBP, but requires the integration for BP message computation to be easily tractable, which makes the method unsuitable for the applications we consider. Another recent promising structured method is *non-parametric BP* (*NBP*) by Sudderth et al. [2003], combining regularization with BP. It turns out that in the problems we consider, local beliefs have even more complex distributions than the full belief itself. Thus, as we show, BP variants perform poorly.

6.3 Mathematical Background

In this section, we provide mathematical background and introduce the notions necessary for the proposed algorithm.

6.3.1 Problem Statement and Notation

We consider the class of problems where the state X has to be inferred from a set of sensor measurements $\mathcal{D} = \{D_k\}$. Our goal is to estimate the probability distribution of the state given the measurements, $p(X|\mathcal{D})$, known as the *belief distribution*, which we will denote by $bel(X)$.

For the general algorithm, we will assume that the state X is a $dimX$-dimensional vector in a bounded rectangle V in $\mathbb{R}^{dimX}$. The measurements are modeled as K random variables D_k, which are drawn independently from *conditional probability distributions* $p(D_k|X)$ with domains in $\mathbb{R}^{dimD}$. The conditional probability distributions (*CPDs*) encode the measurement model and often depend non-linearly on the state X. In many applications, the CPDs are naturally given in the log-linear form via *measurement energy potentials* $\phi_k : \mathbb{R}^{dimX} \times \mathbb{R}^{dimD} \mapsto \mathbb{R}^{+}$. Then the CPD for D_k can be written as

$$p(D_k|X) = \eta \exp\Big(-\phi_k(X, D_k)\Big), \tag{6.1}$$

where η is the normalizing constant. Let $\phi(X) := \sum_k \phi_k(X, D_k)$ be the *total measurement energy*.

We will primarily focus on problems with global initial uncertainty as this is often the more challenging case in robotics. In this case, the prior $p(X)$ is uniform. With this assumption, using the Bayes rule and conditional independence of D_k, the belief can be shown to be proportionate to $\prod_k p(D_k|X)$. Using the log-linear form of measurement CPDs, we can express this fact as $bel(X) \propto \exp(-\phi(X))$. Since the normalization constant is usually unavailable directly, it is more convenient to work with the unnormalized belief $\pi(X) := \exp(-\phi(X))$. Then the belief can be obtained by normalizing: $bel(X) = \frac{1}{Z}\pi(X)$, where Z is called the partition function. When the prior is non-uniform, we will write $\overline{bel}(X)$ to denote the prior, then the belief is $bel(X) = \frac{1}{Z}\overline{bel}(X)\pi(X)$.

In the indoor localization problem, the robot needs to determine its position on a known map $\mathcal{M}$ from laser range measurements $\mathcal{D}$. The map is commonly represented by an *occupancy grid*, which can be produced using *SLAM* techniques [Thrun et al., 2005]. The state $X := (x, y, \theta)$ is the robot's pose comprised of map coordinates

(x, y) and orientation angle θ. The measurements $\mathcal{D}$ consist of a single scan from a laser range finder. $D_k := (\rho_k, \alpha_k)$ denotes a single ray in the scan and consists of range ρ_k and bearing α_k components. To interpret the measurements, we rely on the most widely used *independent beam* (*IB*) measurement model for range finders [Thrun et al., 2005]. In this model, all rays are considered as independent measurements of range to obstacles in the environment, corrupted by Gaussian noise. The expected range to the closest obstacle along ray k is computed by ray tracing on the map. If $\mu_k(X)$ is the expected range along ray k, the measurement potential is given by

$$\phi_k(X, D_k) := \frac{1}{2\sigma^2}(\mu_k(X) - \rho_k)^2, \tag{6.2}$$

where σ^2 is the Gaussian noise variance.

In the tactile manipulation problem, the robot needs to determine the position X of a known stationary object $\mathcal{O}$ based on a set of tactile measurements $\mathcal{D}$. The object is typically represented as a polygonal mesh. The state $X := (x, y, z, \alpha, \beta, \gamma)$ is the 6DOF pose of the object — including position (x, y, z) and orientation angles (α, β, γ) — in the manipulator coordinate frame. The measurements $\mathcal{D}$ are obtained by touching the object with the robot's end effector. Each measurement $D_k := (D_k^{\text{pos}}, D_k^{\text{nor}})$ consists of the measured cartesian position of the contact point D_k^{pos} and the measured surface normal D_k^{nor}. To interpret the tactile measurements, we use the *proximity measurement model*, which was described in detail in Chapter 2. In this model, the measurements are considered independent of each other with both position and normal components corrupted by Gaussian noise. For each measurement, the potential depends on the Mahalonobis distance between the measurement and the object. The distance is taken in the 6D measurement space.

6.3.2 Insight into the Measurement Model

Each measurement model contains a wealth of domain knowledge about the application, the sensor, and the measurement process. In order to construct a more efficient inference algorithm, we make two properties of the measurement model available to the algorithm at runtime. The first property is a *relaxation* of the measurement model, representing a more optimistic interpretation of the measurements. The second property is a *strengthening*, representing a more pessimistic interpretation of the measurements. Thus we define r_k to be a *relaxation* and s_k to be a *strengthening* of the measurement potential ϕ_k, if for all X in V we have

$$r_k(X) \leq \phi_k(X, D_k) \leq s_k(X). \tag{6.3}$$

We define $r(X) := \sum_k r_k(X)$ and $s(X) := \sum_k s_k(X)$. Then for all X in V

$$r(X) \leq \phi(X) \leq s(X). \tag{6.4}$$

From relaxations and strengthenings, we obtain bounds on the unnormalized belief π. Let $\pi_s(X) := \exp(-s(X))$ and $\pi_r(X) := \exp(-r(X))$. Using (6.4), for all X in V we obtain

$$\pi_s(X) \leq \pi(X) \leq \pi_r(X). \tag{6.5}$$

To extend the above equation to the case of a non-uniform prior, let $\overline{bel}_r$ and $\overline{bel}_s$ be relaxation and strengthening of the prior, then we have $\overline{bel}_s(X)\pi_s(X) \leq \overline{bel}(X)\pi(X) \leq \overline{bel}_r(X)\pi_r(X)$.

Note that relaxations and strengthenings defined here are simply lower and upper bounds on the measurement potentials. However, we feel it is helpful to have an intuitive understanding of these bounds, because they are central to the proposed algorithm. For relaxations and strengthenings to be useful, they need to be easy to evaluate on the rectangular grid regions that arise during adaptive gridding. For this reason, we construct relaxations and strengthenings that are piece-wise constant over the grid regions.

In the sections that follow, we show how to build useful relaxations and strengthenings for tactile manipulation and robot localization. These two applications represent two of the most common measurement model types in robotics. Tactile manipulation utilizes a proximity model, which is commonly used for many sensor types: e.g., for stereo [Olson, 2000] and laser range finders (under the name of *likelihood fields*) [Thrun et al., 2005]. This model is smooth, almost everywhere differentiable and omits *negative information*. The indoor localization example utilizes the independent beam model, which takes negative information into account. This is a more complex model that results in an erratic discontinuous belief, but captures more information from the measurement process.

6.3.3 Relaxations and Strengthenings for Tactile Manipulation

From Sect. 2.3.3, we have $\phi(X) = \frac{1}{2}u^2(X)$. In Appendix A, we derive partial Lipschitz constants for u. The relaxations and strengthenings can be built straightforwardly using these Lipschitz constants (in fact, this derivation generalizes to any other Lipschitz measurement model).

Let Λ_u be the vector of partial Lipschitz constants of u, i.e.,

$$\Lambda_u := (\lambda_{u,x}, \lambda_{u,y}, \lambda_{u,z}, \lambda_{u,\alpha}, \lambda_{u,\beta}, \lambda_{u,\gamma}). \tag{6.6}$$

For a grid cell G, let $\Delta X := (\Delta x, \Delta y, \Delta z, \Delta\alpha, \Delta\beta, \Delta\gamma)$, where $\Delta x, \ldots, \Delta\gamma$ are half the width of G along the respective axes. Let $\Delta u := \Delta X \cdot \Lambda_u$. Then we know that u can change by at most Δu within G. If X_c is the center of G, then for any $X \in G$, we can define

$$u_r(X) := u(X_c) - \Delta u \tag{6.7}$$

and

$$u_s(X) := u(X_c) + \Delta u \tag{6.8}$$

to be the relaxation and strengthening of u, respectively.(1)

Now, define $r(X) := \frac{1}{2}u_r^2(X)$ and $s(X) := \frac{1}{2}u_s^2(X)$ to be the relaxation and strengthening of ϕ, respectively. Clearly, $r(X)$ and $s(X)$ are constant in G by construction.

(1) We should take care to make sure u_r does not become negative, i.e., $u_r(X) := \max(0, u(X_c) - \Delta u)$.

6.3.4 Relaxations and Strengthenings for Robot Localization

Consider a grid cell G of robot's poses $X = (x, y, \theta)$. The k-th measurement potential is proportional to the squared error of measured range ρ_k with respect to expected range $\mu_k(X)$ (see (6.2)). For the moment, assume that $m_k(G)$ is a lower bound on $\mu_k(X)$ for X in G and $M_k(G)$ is an upper bound (we will explain how to obtain these values shortly). Then we can construct a relaxation r_k of the k-th measurement potential by underestimating the squared error as follows. If the range reading ρ_k is within $[m_k(G), M_k(G)]$ interval, we set $r_k(X) := 0$. Otherwise we compute the squared error to the closest of the m_k and M_k values:

$$r_k(X) := \frac{1}{2\sigma^2}\min\{(m_k(G) - \rho_k)^2, (M_k(G) - \rho_k)^2\}. \tag{6.9}$$

A strengthening of the k-th potential can be constructed by overestimating the squared error:

$$s_k(X) := \frac{1}{2\sigma^2}\max\{(m_k(G) - \rho_k)^2, (M_k(G) - \rho_k)^2\}. \tag{6.10}$$

By construction, r_k and s_k are constant in G and, as one can easily check, satisfy inequality in (6.3).

All that remains to see is how to obtain $m_k(G)$ and $M_k(G)$. Without loss of generality, let us assume that the range finder is mounted at the center of the robot facing directly forward. A grid cell G of robot's poses creates a cone of possible k-th rays (see Fig. 6.1). We set $m_k(G)$ to the distance between the closest obstacle within this cone and the boundary of G. An upper bound $M_k(G)$ on the possible range reading can be computed by walking the piece-wise linear map boundary. If a boundary fragment starts on one side of the cone and finishes on the other, then it completely blocks all rays in the cone from passing through. Hence $M_k(G)$ is attained on this boundary fragment.

For small grid cells, both m_k and M_k can be computed efficiently. For larger grid cells, these values can be propagated from the smaller grid cells comprising the larger ones. All of m_k and M_k values can be pre-computed before the robot needs to localize,

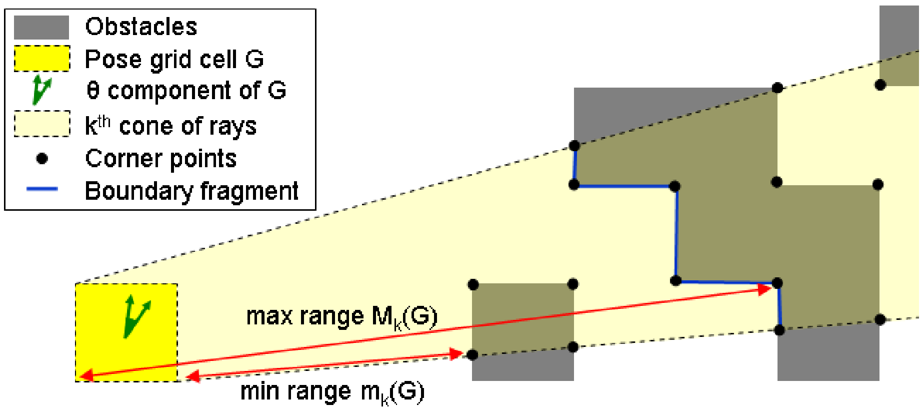


Figure 6.1: Computations of min and max range for a cone of ray poses corresponding to a grid cell G of robot poses. Note that the intersections of the occupied space boundary with the cone boundaries are included as corner points.

because these values are independent of the actual sensor data.

6.4 Belief Approximation Algorithm and Analysis

We start with an intuitive description of the proposed algorithm. The algorithm begins by partitioning the state space into large grid cells. Then it iteratively refines grid cells until the desired final resolution is reached. At each iteration, low probability grid cells are pruned to focus computational resources on the high likelihood areas of the state space. The pruning step relies on relaxations and strengthenings. After the final iteration, the result is a piece-wise constant approximation of the belief: zero in all pruned grid cells and equal to each cell's midpoint value in each of the remaining grid cells. We give full algorithm listing in Alg. 6.1.[2] See Fig. 6.2 for an illustration of the evolving grid regions during one run of the algorithm.

[2] In this section we focus on the case of the uniform prior. For the non-uniform prior case, $\pi_r(X)$, $\pi(X)$, and $\pi_s(X)$ need to be replaced by $\overline{bel}_r(X)\pi_r(X)$, $\overline{bel}(X)\pi(X)$, and $\overline{bel}_s(X)\pi_s(X)$ respectively throughout the section.

Alg. 6.1 Guaranteed Recursive Adaptive Bounding (GRAB): adaptive grid algorithm for belief estimation.

$$\widehat{\pi}_{max} := init\_pi\_max() \tag{6.11}$$

Start with the whole region $\mathcal{G}^0_{keep} := \{V\}$. Until the desired resolution is reached, repeat:

Refine: Construct $\mathcal{G}^n$ by splitting each grid cell in $\mathcal{G}^{n-1}_{keep}$ in halves along each dimension.

Bound: For each G_i in $\mathcal{G}^n$ compute upper bound U^n_i and lower bound L^n_i on π using relaxation r^n and strengthening s^n and compute $\pi(X_i)$. Update $\widehat{\pi}_{max}$, the maximum value of $\pi(X_i)$ observed by the algorithm thus far.

Prune: Add to $\mathcal{G}^n_{prune}$ grid cells G_i with the lowest upper bounds U^n_i, as long as:

$$\sum_i U^n_i Vol(G_i) \leq \xi \ \widehat{\pi}_{max} Vol_* / N. \tag{6.12}$$

Prune the selected grid cells by setting $\mathcal{G}^n_{keep} := \mathcal{G}^n - \mathcal{G}^n_{prune}$.

After the final iteration, estimate the unnormalized belief:

$$\widehat{\pi}(X) := \begin{cases} \pi(X_i), & \text{if } X \in G_i \in \mathcal{G}^N_{keep} \\ 0, & \text{otherwise.} \end{cases} \tag{6.13}$$

Estimate the partition function $\widehat{Z} := \int \widehat{\pi}$ and the normalized belief $\widehat{bel}(X) := \frac{1}{\widehat{Z}}\widehat{\pi}(X)$. Compute approximation error bounds:

$$\varepsilon_{prune} := \sum_{n=1}^{N} \sum_{G_i \in \mathcal{G}^n_{prune}} U^n_i \ Vol(G_i). \tag{6.14}$$

$$\varepsilon_{keep} := \sum_{G_i \in \mathcal{G}^N_{keep}} (U^N_i - L^N_i) \ Vol_*. \tag{6.15}$$

$$\varepsilon := \varepsilon_{prune} + \varepsilon_{keep}. \tag{6.16}$$

Outputs: $\widehat{\pi}, \widehat{Z}, \widehat{bel}$ - estimated unnormalized belief, partition function, normalized belief, ε - approximation error bound.

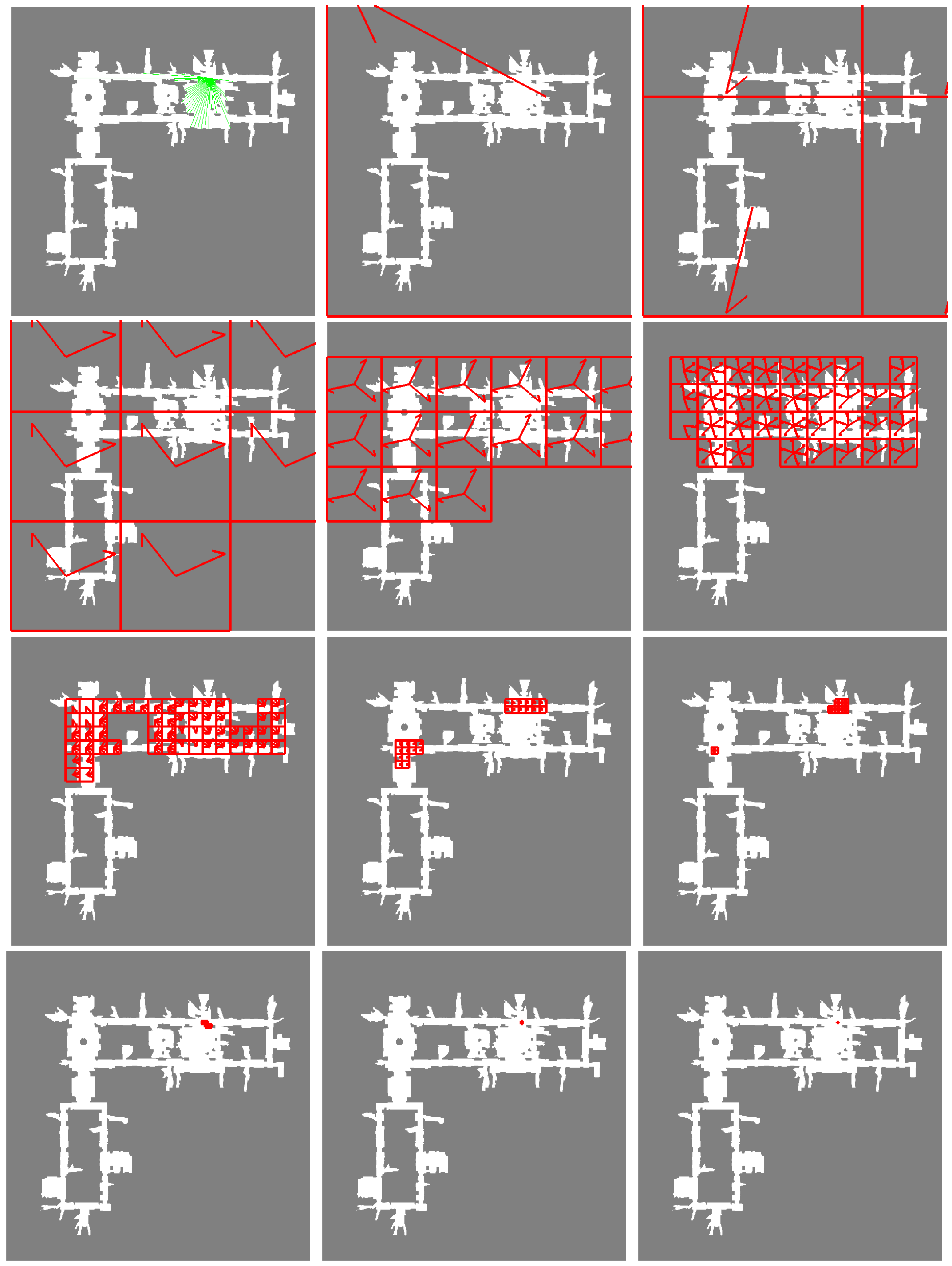


Figure 6.2: GRAB evolution for indoor localization. The top left plot shows the laser scan (green lines) at the true robot pose. The remaining plots show the first 11 iterations of GRAB. Red squares show the remaining grid cells, red arrows show the angular components of the grid cells.

6.4.1 Algorithm Detail

Notation

Let δ_* be the desired final resolution — a user specified parameter, which directly controls the degree of refinement. Then the total number of refinement iterations required is $N = \log \frac{Len_1}{\delta_*}$, where Len_1 is the size of region V along the first axis. A finer resolution δ_* will increase overall running time and improve accuracy of the resulting approximation. Let $Vol(\cdot)$ denote the volume of a region and Vol_* denote the volume of a grid cell at the final resolution δ_*.

For each iteration n, the current set of grid cells will be denoted by $\mathcal{G}^n := \{G_i\}$. All grid cells in $\mathcal{G}^n$ have the same resolution δ^n. During pruning step, we will prune some grid cells $\mathcal{G}^n_{prune} \subset \mathcal{G}^n$ and keep the rest $\mathcal{G}^n_{keep}$. We will denote the center of a grid cell G_i by X_i.

The pruning of low-likelihood grid cells is controlled by a mode sensitivity parameter ξ. It is a user specified parameter, which is the smallest fraction of the maximum value of π that is still considered significant. In other words, if ξ is set to 1%, then all regions where π is at least 1% of its maximum value are considered significant.

Initialization

The initialization of $\widehat{\pi}_{max}$ in (6.11) has an effect on the efficiency of the algorithm. The better the estimate the more can be pruned from the very beginning in (6.12). The simplest implementations can set $\widehat{\pi}_{max} := 0$ or $\widehat{\pi}_{max} := \pi(X)$ for some $X \in V$. A more sophisticated strategy is to run a greedy version of Alg. 6.1, where $U^n_{max} := \max_i U^n_i$ can be used instead of $\widehat{\pi}_{max}$ in (6.12). Then set $\widehat{\pi}_{max} := \max_i \widehat{\pi}(X_i)$ based on the belief estimate produced by the greedy run. The greedy run can not guarantee that a mode will be found, but it tends to provide a good initial estimate for π_{max} very quickly.

Bounding Step

Note that we adaptively change the relaxations and strengthenings from one refinement to the next. In practice, r^n and s^n are chosen so that the bounds on π^n_r and π^n_s

are easy to compute. If L_i^n is a lower bound on π_s^n in a grid cell G_i and U_i^n is an upper bound on π_r^n in G_i, then due to (6.5) L_i^n and U_i^n are also bounds on the variation of π within G_i:

$$L_i^n \leq \pi(X) \leq U_i^n. \tag{6.17}$$

Pruning Step

Inequality (6.12) ensures that the total probability mass discarded after all N refinements is at most ξ portion of the estimated mass of the "heaviest" grid cell. In practice, for peaked beliefs the value of ξ can be set quite low, because π is virtually zero everywhere except in the vicinity of the peak.

Error Computations

We have two sources of approximation error in our algorithm. ε_{prune} upper-bounds the error due to the probability mass of grid cells we pruned. ε_{keep} upper-bounds the error due to the variation of π in the grid cells kept at the final refinement stage. Thus, ε upper-bounds the total L_1 error between π and its approximation $\widehat{\pi}$.

6.4.2 Approximation Analysis

Two important properties of the algorithm can be proven:

1 GRAB will find *all modes* of the belief.

2 We can compute a sound bound on the L_1 distance between the true belief and its approximation.

Hence, we can be absolutely sure that the algorithm did not miss any global optima and we know how good of an approximation was produced.

First, we define what we mean by a mode. Let bel_{max} be the maximum value of the belief bel, then we will consider any point X in V to be a ξ-*mode* of the belief if $bel(X) \geq \xi \; bel_{max}$ for some user-specified parameter ξ.

Theorem 6.1. *Let ξ be the mode sensitivity setting of GRAB. Let X be a ξ-mode of the belief bel. Then X is in one of the grid cells kept at the final iteration of GRAB.*

Proof. We argue by contradiction. If X is not in the final set of grid cells, then at some iteration n a grid cell G_i containing X has been pruned. However, as one can easily check, this assumption violates the condition in (6.12). □

Theorem 6.2. *Let $\widehat{Z}$, $\widehat{\pi}$ and $\widehat{bel}$ be the partition function, unnormalized belief and normalized belief estimates respectively produced by GRAB. Let ε be the error bound computed by GRAB at runtime. Then*

$$|Z - \widehat{Z}| \leq \|\pi - \widehat{\pi}\|_{\mathsf{L}_1} \leq \varepsilon \tag{6.18}$$

and

$$\|bel - \widehat{bel}\|_{\mathsf{L}_1} \leq \frac{2\varepsilon}{\widehat{Z} - \varepsilon}. \tag{6.19}$$

The proof follows from the bounds carefully constructed at runtime. We provide details in Appendix C.

6.5 Experimental Results

6.5.1 Indoor Robot Localization

Indoor environments contain a variety of obstacles invisible to robot's sensors — glass doors, mirrors, staircases — as has been noted in several field studies [Burgard et al., 1998, Montemerlo et al., 2002]. Therefore, for safety, it is important for the robot to localize itself on a known map denoting invisible hazards *prior to moving* in the environment. For this reason, we evaluate the ability of modern algorithms to reliably estimate the global localization belief prior to moving.

We performed several sets of experiments with real and simulated robot data in several different environments.[3] In each experiment, the localization was performed

[3] In our experiments, we used publicly available maps produced by Cyrill Stachniss (Fig. 6.3), Mike Montemerlo (Fig. 6.4), and Ashley Tews (Fig. 6.5). We thank the map authors for making these maps available to the public.

from a single scan of laser data. The real data was produced by a SICK LMS laser range finder. In the experiments, the results are averaged over 100 runs. The error bars in plots represent 95% confidence intervals. For GRAB, the different running times are obtained by varying the desired resolution δ_* with $\xi = 1\%$.

We compared the performance of the proposed algorithm (GRAB), uniform grid (UG), uniform discretization belief propagation (UBP), non-deterministic belief propagation (NBP, proposed by Sudderth et al. [2003]), subregion adaptive integration (SAI, proposed by Genz and Kass [1997]), Scaling Series (SS), and several variants of Monte Carlo localization. Although for Monte Carlo localization with a single scan of laser data it is more correct to perform a single importance sampling update, in some situations multiple updates can perform better. For completeness, we provide both versions: single-update (IS) and 10-update (MCL). For the latter, we injected 50cm position noise between updates.

For GRAB, we pre-computed the min-max ranges for grid cells sized 10cm and greater as described in Sect. 6.3.4. The pre-computation only needs to be carried out once for each map as it is independent of actual sensor measurements. For a 70m x 70m map, the pre-computation takes 3 minutes. For GRAB, UG, and UBP, we also pre-computed expected range scans for grid cells sized 10cm and greater. This optimization allows grid based methods to be more efficient than Monte Carlo approaches, for which such pre-computations are unsuitable.

The BP algorithms were implemented on a cluster graph satisfying the running intersection property. We assigned X CPD and D_1 CPD factors to the first cluster, and each of the remaining D_k CPDs to the other $k-1$ clusters. For UBP, the evidence was applied to the factors prior to propagation. For efficiency reasons, operations during propagation were carried out only on the rows supporting the evidence. Since the evidence was applied prior to propagation, the method is equivalent to integrating measurements one at a time. BP variants performed rather poorly because the belief in this problem is unstructured. While this may seem obvious in retrospect, we still provide the empirical evaluations.

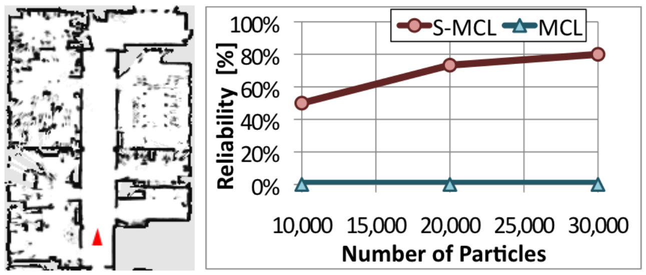


Figure 6.3: Smoothing improves reliability of MCL because it decreases roughness of the belief. Here the reliability of MCL and S-MCL is tested on the 20m x 14m map (left). Red triangle denotes the robot's pose.

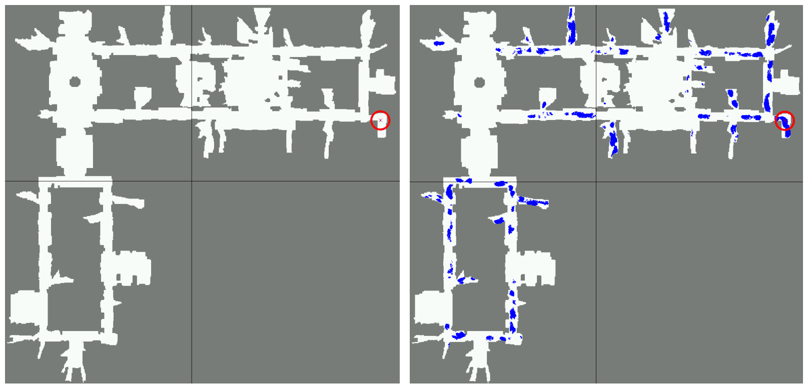


Figure 6.4: Smoothing increases ambiguity of localization. Left: without smoothing the resulting belief has a unique mode (circled). Right: with significant smoothing (50cm noise, 7 rays) the result is highly multi-modal.

Impact of Smoothing

In indoor localization, the most common smoothing techniques are to increase the measurement noise and to decrease the number of rays considered. To evaluate smoothed MCL (S-MCL), we use 20cm noise and 61 rays per scan. In this and later experiments, we consider the localization successful if the mean pose is within 1m and 30° of the true pose. Figure 6.3 shows the reliability of global localization on a 20m x 14m map, where S-MCL exhibits very similar performance to prior art with similar settings [Thrun et al., 2001, Plagemann, 2008]. MCL without smoothing (i.e., 1cm noise, 361 rays) fails to solve the problem. Thus, smoothing has been vital for solving the global localization problem because MCL is unable to solve the problem without it.

On the other hand, smoothing can have a number of negative effects as it discards

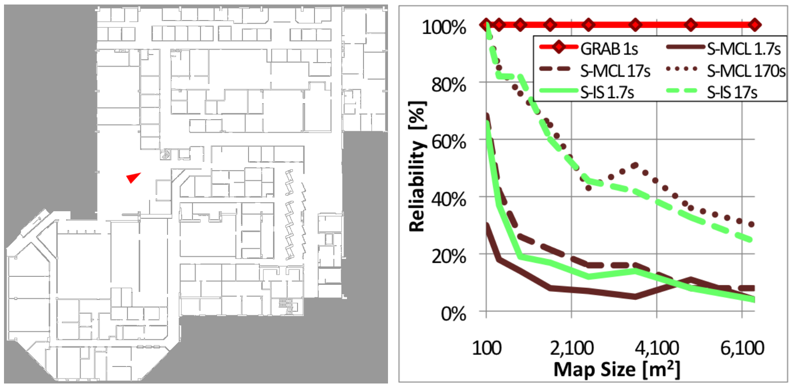


Figure 6.5: Dependence of reliability on initial uncertainty (right). Experiments were performed on sub-maps of the 80m x 80m library map (left). Red triangle denotes the robot's pose.

information contained in the data. Not only can the accuracy go down, but the produced belief estimate is much more ambiguous than without smoothing (Fig. 6.4). Under these conditions, the robot has to travel to resolve the ambiguity. In addition to being unsafe with respect to invisible hazards, this strategy risks losing the robot's location because MCL does not track multiple modes very well. Moreover, from theoretical perspective smoothed belief estimates do not converge to the true belief even as the number of particles tends to infinity.

Dependence on Initial Uncertainty

Even with smoothing, reliability of MCL drops quickly as map size increases (Fig. 6.5). These experiments were performed in simulation by taking sub-maps of a larger map. We show results for S-MCL with 1,000, 10,000, and 100,000 particles (with computation time of 1.7s, 17s, and 170s respectively). For comparison, GRAB solves the problem in under 1s (with δ_*= 5cm) and provides guaranteed results.

Localization Accuracy

To evaluate the accuracy of localization, we performed two sets of experiments on the 70m x 70m map depicted in Fig. 6.4. In the first set, the robot has to localize after being placed in a random position on the map. These experiments used simulated data with 1cm noise, so that exact reference pose is available for evaluation purposes. The

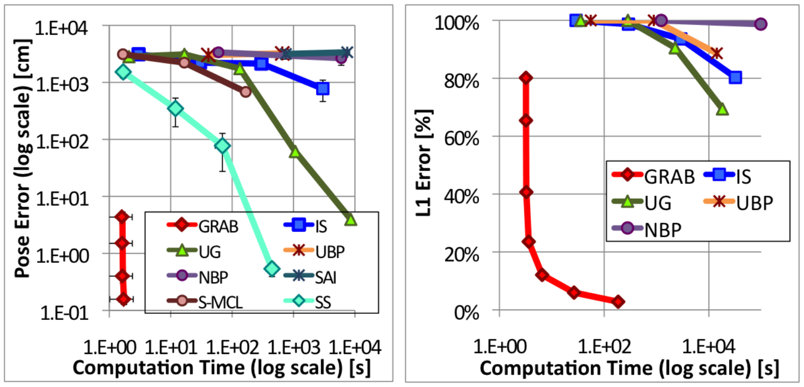


Figure 6.6: Accuracy of pose estimation (left) and belief estimation (right) for global robot localization on the map depicted in Fig. 6.4. The right plot shows $0.5\mathsf{L}_1$ error, which represents dissimilarity percentage of the approximated belief vs. ground truth belief.

results are shown in Fig. 6.6 (left), where we plotted the distance of the estimated pose from the reference pose vs. computation time.[4] GRAB demonstrated logarithmic dependence on localization precision, whereas the other algorithms behaved roughly linearly. GRAB was able to recover the reference pose with 1mm accuracy[5] and outperformed other approaches by several orders of magnitude.

In these experiments, we also evaluated the performance of Scaling Series (SS), which we discussed in detail in Chapter 2 and first proposed in [Petrovskaya et al., 2006]. Scaling Series relies on annealing, which works well for problems with differentiable beliefs. However, in this case, the belief has many discontinuities due to the use of negative information in the IB measurement model. Hence, Scaling Series is at a disadvantage compared to GRAB (see the cyan line in the left plot of Fig. 6.6), but it still performs significantly better than prior art approaches. To obtain the SS line in the plot, we varied M between 3 and 100, while keeping $\delta_* = 1$cm and threshold $\xi = 10\%$. Horizontal bars show 95% confidence in running time for the same setting of M.

The second set of experiments evaluates the accuracy with which the *entire* global localization belief can be estimated. These experiments were carried out with a

[4] Horizontal error bars for GRAB show 95% confidence in running time for the same setting of δ_*.

[5] Although GRAB was able to recover the pose to within 1mm, one should keep in mind that these experiments were carried out in simulation. To achieve comparable accuracy with real data, accurate environment models are required (see Chapter 5 for one such example).

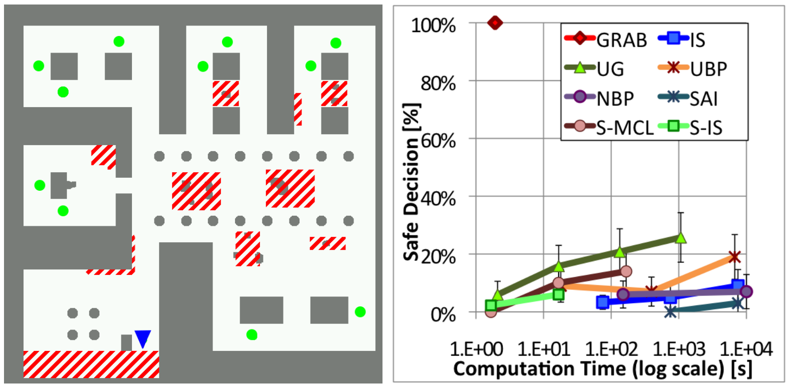


Figure 6.7: Decision safety results (right). On the map (left), the blue triangle denotes the robot's pose in one of the experiments. The green circles denote multiple modes of the belief. Safety zones are shaded in red.

single scan of real robot data. The robot's pose is circled in Fig. 6.4. The reference distribution was obtained using a fine mesh. Prior art approaches were unable to estimate the un-smoothed belief with any degree of accuracy within reasonable time. The right plot in Fig. 6.6 shows the results using a smoothed model (25 rays and 10cm noise) for all algorithms and reference distribution. GRAB dramatically outperformed other approaches. The experiments demonstrate that accurate estimation of the belief takes significantly longer than estimation of only the robot's pose.

Safety During Exploration

This set of experiments evaluates whether a robot is able to make safe decisions during navigation under conditions of multi-modality. The experiments were carried out in a 70m x 70m simulated museum environment (Fig. 6.7). The red shaded areas denote safety zones containing glass encased exhibits. The robot was placed randomly in front of the safety zones and had to plan an exploration route using safe maneuvers. Note that this environment has several look-alike areas leading to multi-modal localization beliefs.

First, the robot estimates a global localization belief. Then it evaluates the proposed action of moving forward by 1m based on the estimated belief. To make a fair comparison of all algorithms, the robot's planner treats all belief estimates the same.

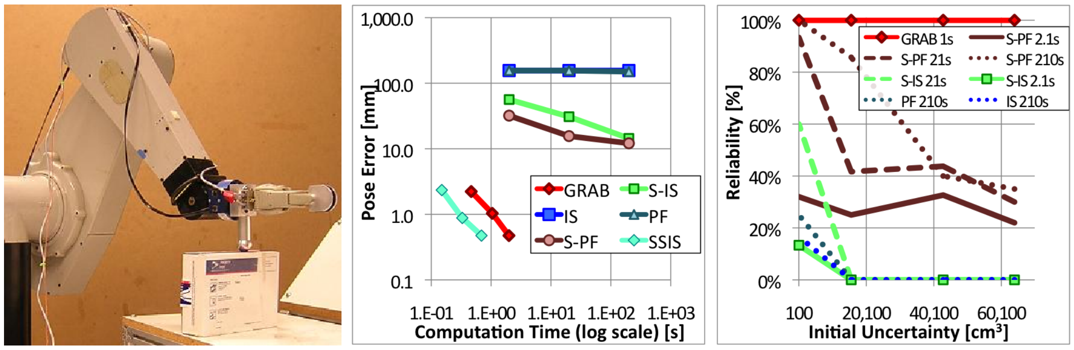


Figure 6.8: Tactile sensing for object localization. Left: experimental setup. Center: pose estimation accuracy vs. computation time. Right: reliability vs. initial uncertainty (with unrestricted orientation).

For each experiment, it draws 100,000 samples from the belief approximation and applies the robot's dynamics model to the samples obtaining a prediction distribution. The prediction distribution is used to compute the expected risk of the maneuver. The risk function is 1 in the safety zones and 0 everywhere else. If the expected risk is higher than 0.01%, the maneuver is considered unsafe. Since the robot is positioned facing safety zones, this maneuver should be identified as unsafe in all experiments.

As expected, GRAB correctly identified all modes of the belief and thus lead to safe decisions in all experiments. Other algorithms often lead to unsafe decisions.

6.5.2 Tactile Manipulation

In these experiments, a stationary object is localized by a robot that explores the object by touching it with its end-effector. The experimental setup consisted of a PUMA manipulator robot equipped with 6D JR3 force/torque sensor at the wrist (see Fig. 6.8). The sensed object was a rectangular box, for which we constructed a polygonal model using careful ruler measurements. The data sets consisted of five data points taken from different sides of the box.

For un-smoothed versions of the algorithms, we used $\sigma_p = 1$mm and $\sigma_n = 2°$. We compared the proposed algorithm (GRAB), Scaling Series (SS), importance sampling (IS), and particle filter (PF). For PF, we performed 10 updates with the same data injecting 1cm noise between updates. We also evaluated S-IS and S-PF using smoothing parameters shown to be optimal in [Petrovskaya et al., 2006]: $\sigma_p = 1$cm and $\sigma_n = 10°$.

Object Localization Accuracy

First, we evaluated how accurately the object could be localized in the robot's workspace. The initial uncertainty in these experiments was 40cm x 40cm x 40cm with unrestricted orientation. The results shown in Fig. 6.8 (center) are averaged over 100 runs with simulated data, although the algorithms exhibited similar performance with real data. GRAB was able to localize the object to 1mm within about 1s. Scaling Series was approximately three times faster than GRAB for this problem. Other algorithms were unable to get average accuracy better than 1cm even after 3 minutes. It should be noted that sub-centimeter accuracy is often required for successful manipulation as we saw in Chapter 5. GRAB quickly achieves sub-millimeter accuracy, which we have shown in Chapter 2 to be sufficient for reliable manipulation of objects.

Reliability vs. Initial Uncertainty

To evaluate how reliability depends on initial uncertainty, we varied the uncertainty from 5cm cube to 40cm cube. Unrestricted orientation was used in all experiments. The localization was considered successful if the pose was recovered within 5mm and 5°. The results of 100 runs with simulated data are shown in Fig. 6.8 (right). GRAB solves the problem in 1s (with $\delta_* = 2$mm) with guaranteed results. We also show results for S-PF with 10,000 to 1,000,000 particles (2.1s to 210s running time) and S-IS with 100,000 to 1,000,000 particles (2.1s to 21s running time). For S-IS and S-PF, the reliability degrades quickly with uncertainty. IS and PF performed even worse.

6.6 Discussion and Conclusions

We have presented a belief estimation algorithm that relies on a simple adaptive gridding technique. Yet, in contrast to state-of-the-art, it is able to solve high-roughness problems with guaranteed results. The reason for the superior performance is that our algorithm utilizes additional domain knowledge, whereas the other algorithms treat the belief as a "black box" function. The downside is that GRAB is less general than "black box" approaches and requires additional work to extract the domain knowledge. However, in situations where "black box" algorithms are insufficient, the extra work is very worthwhile. The domain knowledge is captured in a general way in the form of measurement model bounds, which we have given intuitive names of relaxations and strengthenings. Thus, our algorithm is applicable to problems where such bounds can be constructed. We have demonstrated the algorithm on indoor navigation and tactile manipulation. We expect that it will extend to other robotic applications with accurate relative sensors (laser, vision and tactile), although clearly application-specific relaxations and strengthenings will need to be built in each case. Since the class of domain knowledge algorithms is in principle stronger than the class of "black box" algorithms, we hope the demonstrated success will encourage development of algorithms utilizing domain knowledge in a general way.

A number of extensions of the algorithm can be made. For high dimensional problems, the approach can be combined with belief propagation [Kozlov and Koller, 1997, Isard et al., 2008] or Rao-Blackwellization [Murphy and Russell, 2001, Montemerlo, 2003] — a very promising direction for future work. The refinement strategy could also be tuned to obtain different properties. For example, SAI-like refinement strategy could be beneficial when the belief has regions of similar probability or whenever a better representation of low probability regions is desired. In this chapter, we have only shown that non-uniform priors could be used in principle, but have not performed any empirical evaluation of such scenarios. The non-uniform priors could carry information about prior measurements and dynamic updates. There are several possible methods for using GRAB to solve dynamic problems. One method is an adaptive histogram filter [Burgard et al., 1998]. This could be implemented similarly to SS-DYN3 of Chapter 2, resulting in the same convergence properties. Analogues of SS-DYN1 and SS-DYN2 could also be implemented. Another method is to use GRAB to obtain a representation of the measurement probability distribution for algorithms that sample directly from this distribution [Lenser and Veloso, 2000, Thrun et al., 2001]. Thus, evaluation of how GRAB can be used with non-uniform priors in dynamic applications represents an interesting direction for future work.

Our focus has been solely on belief estimation and we have left many application aspects outside the scope. For indoor localization, combining the presented algorithm with existing methods for handling high traffic areas, doors and other dynamic environment changes, could provide the community with one of the most reliable and accurate global localization solutions to date.[(6)]

We have shown that GRAB is able to achieve 1mm localization accuracy very quickly even in large buildings. In prior art, it has been shown that laser scans contain sufficient information to achieve such accuracy [Censi, 2009]. In Chapter 5, we have shown that 3mm pose estimation accuracy has allowed the robot to reliably manipulate stationary objects not directly observed by its sensors (e.g., door handles, elevator buttons, switches). Moreover, high localization accuracy is useful for maneuvering in tight spaces (e.g., when passing through a doorway). Of course, accuracy of localization depends on accuracy of the environment model. To achieve high accuracy, Censi [2009] matched laser scans directly to each other. In Chapter 5, a highly accurate polygonal model of a door was used in conjunction with a less accurate grid map. Thus, further work on high-accuracy modeling of the environment and sensors will lead to high-accuracy localization, which will open up a host of new applications.

In the past, 1mm accuracy has been reserved for stationary manipulators. With high-accuracy localization, mobile robots will be able to aid in building construction, inspection, and maintenance. They will be able to precisely place fixtures in the environment, accurately drill and paint walls, provide accurate distance measurements, etc. Moreover, high-accuracy localization can easily provide an accurate coordinate transformation between a robot and an off-board sensor (e.g., a ceiling mounted camera or another robot's sensor). Hence, robots will potentially be able to manipulate even dynamic objects invisible to their own sensors.

In summary, we have presented an adaptive grid method suitable for estimating high-roughness beliefs that result from non-linear relative sensors. The method

(6) Development of GRAB library is currently underway. Once complete, the library will be made available at Petrovskaya [2011a].

guarantees that all modes will be found and provides provable approximation error bounds. The strength of the approach comes from utilizing domain knowledge about the measurement process to construct bounds in a general way. We demonstrated the approach on the examples of indoor navigation and tactile manipulation, where it performs significantly better than state-of-the-art and opens up new potential applications.

Chapter 7

Conclusion

7.1 Summary

This thesis has considered the problem of making perceptual inference robust. We identified the weak point of existing methods to be inability to cope with sharp transitions (termed "roughness") in belief distributions. Since roughness of beliefs in robotics is caused by real world phenomena (e.g., range discontinuities), it is important to develop methods capable of handling rough beliefs. To this end, we proposed two novel algorithms: Scaling Series and GRAB. Both of these methods use iterative divide-and-conquer techniques. Scaling Series is a Monte Carlo approach utilizing annealing and GRAB is an adaptive grid approach, which relies on measurement model bounds. Both of these approaches drastically outperform state-of-the-art. GRAB is particularly well suited for problems with discontinuous beliefs, whereas Scaling Series works best for problems with beliefs that are differentiable.

We demonstrated usefulness of the proposed algorithms on a wide range of real robotics problems: touch based perception (Chapter 2), vehicle tracking (Chapter 4), and mobile manipulation (Chapters 5 and 6). In all of these applications, the proposed algorithms outperformed existing methods by several orders of magnitude. In touch based perception, we are able to quickly localize objects in 6DOFs with very large initial uncertainty. This problem has received considerable attention in the literature due to the use of tactile perception for workpiece localization in manufacturing [Chu, 1999, Zhu et al., 2004, Sun et al., 2009]. However, to date, no other method for solving this problem in general form has been proposed.

In vehicle tracking, we demonstrated how the proposed methods can be used to solve high dimensional problems when combined with problem decomposition techniques. Early vehicle tracking approaches date back to the late 1980s and virtually all approaches since then have concentrated on tracking of blobs (clusters of points or features) [Zhao and Thorpe, 1998, Wender and Dietmayer, 2008, Darms et al., 2008]. Unlike the prior art, along with detection and tracking of vehicles, we also estimate their geometric shape, which allows us to accurately estimate each vehicle's motion. The use of geometric models greatly simplifies association of data, which are often split up into separate clusters due to occlusion. Vehicle tracking for autonomous driving is a demanding application, where the results have to be reliably available in a fraction of a second for autonomous decision making. Large amounts of data have to be processed quickly and the processing hardware is limited onboard of the autonomous vehicle. The Scaling Series algorithm has been instrumental in making this approach run in real time with high reliability and accuracy.

In indoor localization, the proposed methods allow for fast and highly accurate (up to a few millimeters) global robot localization in large buildings based on the commonly used laser range finders. Although the field of probabilistic robot localization is over a decade old, to our knowledge we are the first ones to achieve high accuracy global localization. Among other things, the high accuracy allows robots to manipulate objects without having to sense them directly as we demonstrated on the example of navigation with door opening in Chapter 5. We expect that high accuracy localization will find applications in building construction, where accurate measurements are of great importance.

7.2 Directions for Future Work

The presented algorithms are best suited for moderately dimensional problems, where up to 10 parameters need to be inferred. This covers most basic robotics problems such as object or robot localization. However, higher dimensional problems arise when additional properties need to be inferred including object shape, localization of multiple objects, and estimation of state for articulated objects. In these problems, the number of parameters can be very large: 100 or more. As we demonstrated in Chapter 4, the proposed algorithms can be extended to high dimensional problems by combining with decomposition techniques, such as Rao-Blackwellization and parameter splitting. Further exploring the use of proposed algorithms in combination with decomposition techniques is a promising direction for future work as it could potentially address the most difficult problems in robotics: high dimensional problems with global uncertainty.

7.2.1 From Application Perspective

In tactile perception, along with object localization it would be interesting to estimate shape, stiffness, surface friction, and deformations of objects. The whole body contact estimation problem considered in Chapter 3 is another challenging and potentially high dimensional problem that could be addressed by combing proposed algorithms with decomposition techniques. In this problem, shape of the robot and the environment could be estimated along with contact positions on robot links. So far, we have only considered the simplest robot and object shapes. In Chapter 2, we have proposed algorithms for tracking of moving objects. However, so far this approach has only been validated in simulation and warrants additional investigation with real robots. Further, it would be interesting to consider localization and recognition of multiple objects, which can be located in cluttered scenes. In these situations data association becomes an important part of the problem. In addition, the use of multi-fingered hands equipped with multiple tactile pads should be investigated along with planning techniques for optimal sensing actions and contact placement. Promising planning approaches have been proposed in the past for 3DOF problems [Hsiao et al., 2010] and our proposed inference algorithms should help extend these planning approaches to 6DOF problems.

In vehicle tracking (Chapter 4), we have only estimated 2D shape of vehicles so far. However, the proposed anchor point coordinates help track the motion of the

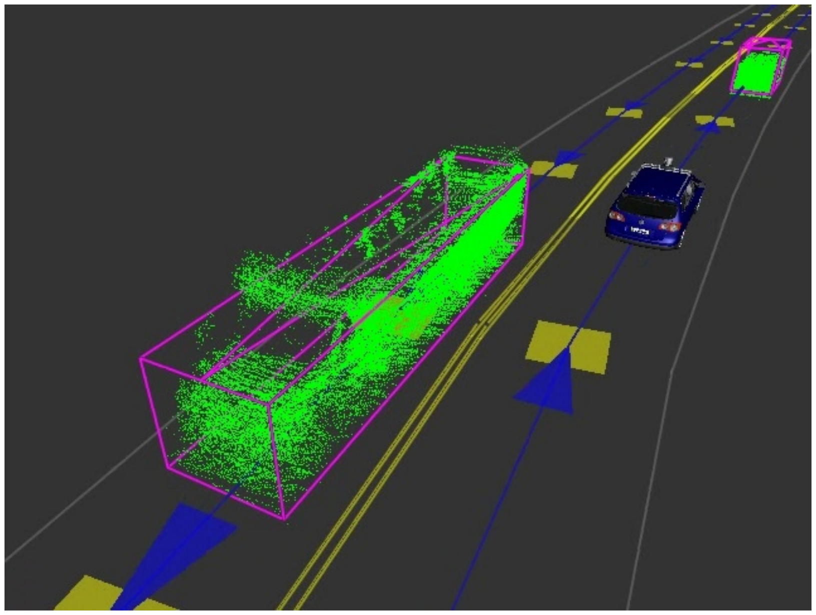


Figure 7.1: 3D shape inference for vehicle tracking. The green points show the accumulated Velodyne points assigned to each vehicle over multiple frames. Tracking with anchor point coordinates allows us to align the points from different frames on per-vehicle basis.

shapes accurately. Hence, full 3D models of the vehicles can be built (see Fig. 7.1 for preliminary results). 3D models will in turn allow for even more accurate motion estimation. This can be particularly useful at intersections with stop signs, where it is crucial to distinguish between a vehicle standing completely still and a vehicle that is moving forward slightly. In addition, the model based approach can be extended to motorcyclists, pedestrians, and other road scene participants. This has been considered by Vu [2009], but with our proposed approach much more accurate models of these participants can be built. Incorporating data from other sensing modalities should also prove useful, especially from omni-directional cameras and radars, which can help detect and track vehicles at a greater range.

In mobile manipulation (Chapters 5 and 6), we have shown that high accuracy global localization can be achieved with commonly used 2D laser range finders. However, high accuracy localization requires highly accurate maps. In Chapter 5, we proposed a hybrid representation, which combines high resolution polygons with low resolution occupancy grid maps built by SLAM techniques. The high resolution polygons were built using hand measurements. Hence, it would be interesting to develop

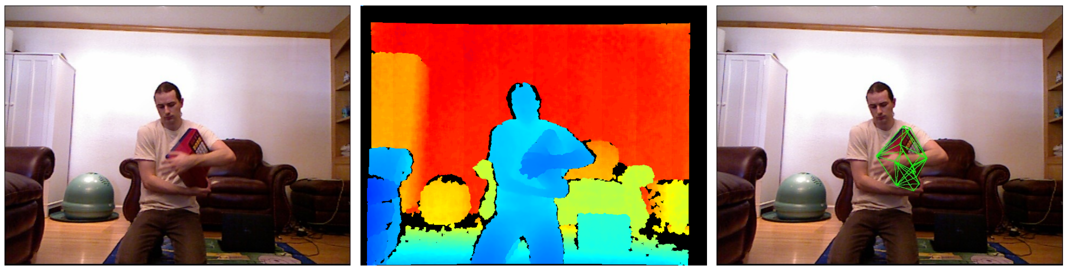


Figure 7.2: Object tracking with Kinect. Left to right: scene, range data, tracking results. The tracking results were obtained using the object, model, and method from Chapter 2.

an approach, which allows robots to build such representations autonomously. High accuracy of pose estimation produced by the proposed algorithms should help build these accurate representations, which could include articulated objects and semantic information. Another interesting direction is to extend the approach to 3D as more and more 3D sensors become available.

Finally, in this thesis, we obtained data from contact sensors, robot joints, or laser range finders. However, the proposed algorithms could be applied to other types of range and non-range sensors. In particular, the recently released Kinect sensor is inexpensive and widely available. It is an RGBD camera that produces a range data cloud, which is in many ways similar to the 3D range finder data we used in Chapter 4. Other RGBD cameras are surely to follow, making applications of the proposed algorithms to these sensors a promising direction for future work. Fig. 7.2 shows preliminary results of applying Scaling Series method to range data generated by Kinect sensor. Using the method, object, and model from Chapter 2, resulted in fast and accurate tracking with Kinect data.

7.2.2 From Theoretical Perspective

We have provided some convergence results for both algorithms. For Scaling Series, we have shown that the estimates converge to the true belief in Sect. 2.4.6. For GRAB, we have shown that the estimates converge, derived the L_1 error between estimated and true belief, and provided best/worst case running time analysis in Sect. 6.4.2. However, it would be interesting to derive more detailed convergence results.

In particular, due to their divide-and-conquer nature, both algorithms can run in log time (in the best case) compared to naive approaches, such as importance sampling or uniform grid. On the other hand, in the worst case, both algorithms can take as long as naive approaches. Intuitively, the worst case is achieved, when the algorithms are unable to prune out any part of space. In the practical problems we have considered, both algorithms perform close to the best case. To better understand applicability, it would be interesting to characterize the types of problems on which the algorithms perform particularly well or particularly poorly. For these problems, we could then prove tighter convergence rate results, which would be useful when deciding what method should be used for a particular problem and what performance is to be expected.

The proposed algorithms have the advantage while searching for modes of the belief, because at this stage large parts of space can be pruned. However, for some problems, sub-mode refinement may be desirable (e.g., for high accuracy numerical integration). Once refinement goes sub-mode, no additional space can be pruned and so from this point on the proposed algorithms are equivalent to naive estimation within the supporting regions of the modes. In these cases, both algorithms allow for a trade-off between running time and accuracy: the iterations can be stopped after a preset amount of time, at which point the estimates are more coarse compared to the final resolution. This is also useful for under-constrained problems, where modes cover large regions of space and obtaining coarse estimates quickly helps plan further sensing actions. Still, performance of both algorithms in these situations warrants further investigation.

One interesting point is that both algorithms provide some notion of accuracy at each refinement stage. Hence, when iterations are stopped early, the user has a useful measure of how good the obtained estimates are. GRAB goes even further and computes L_1 error between the estimated and true beliefs. Thus, it is possible to perform refinements until an acceptable level of L_1 error is reached. In this case, the running time is unknown in advance, but the level of accuracy is guaranteed. This is in direct contrast to naive approaches, where the running time is known in advance (dictated by the number of samples), but the accuracy remains unknown even after the algorithm terminates.

7.3 Applications

The aim of this thesis has been to provide practical algorithms, which enable real world robotics applications and help speed up adoption of robotics in complex unstructured environments. Development of software libraries is currently underway. Once released these libraries will provide the robotics community with access to this new technology.

Tactile object localization is already used in manufacturing for inspection and machining of parts. However, existing approaches require that each contact point is sensed in a specific location on the object. This means that either a human has to be in the loop or the initial uncertainty about the object pose has to be very small. Tasks dependent on humans can lead to human error, whereas fully automated tasks currently require accurate part positioning, which is expensive and makes manufacturing process inflexible. The approach presented in Chapter 2 does not place restrictive assumptions on where each contact has to be sensed. Hence, in fully automated manufacturing lines, the parts no longer have to be accurately positioned as the proposed localization algorithm can provide an accurate estimate of each part's pose. The parts can simply be rolling down the manufacturing line and be accurately localized at each station for processing. Since the proposed algorithm only needs a CAD model of the part, each part produced on a manufacturing line can be custom. Tasks dependent on humans can now be fully automated to reduce training requirements and human error. With all of these improvements, the manufacturing costs will be lower and the manufacturing process will be much more flexible.

Tactile object localization can also extend the reach of robots to situations where vision is obstructed. Such situations often arise in rescue operations, where the environment can be filled with smoke. These situations are also common in underwater applications, where the water can be mixed with mud, oil, or debris.

The field of intelligent vehicles is developing rapidly. More powerful sensors are being developed and deployed on vehicles. The vehicle tracking algorithm proposed in Chapter 4 can be immediately useful on intelligent vehicles to provide advanced driver assistance and can aid in development and adoption of fully autonomous vehicles.

High accuracy localization (Chapters 5 and 6) is important for mobile manipulation applications in service robotics. With highly accurate localization, robots can be maneuvering in tight spaces, manipulating doors, light switches, elevator buttons, even using sensors from other robotic platforms or environment. This opens up a number of applications and opportunities. Robots will be able to build accurate maps and to navigate safely around visible and invisible obstacles and in tight spaces. In addition to in-home service applications, robots will be able to aid in construction, inspection, and maintenance of buildings and other structures.

Appendix A

Lipschitz Constants for Tactile Perception

In this section we provide bounds on Lipschitz constants of π and u. These bounds are used as explained in Sect. 2.4.3 to set the shape of δ-neighborhoods. Theorem A.1 relates the Lipschitz constants of π and u. Theorem A.2 computes bounds on partial Lipschitz constants of u. It relies on a helper Lemma A.1, which is a variant of triangle inequality for sets of points. The statement and proof of Lemma A.1 are provided at the end of the section.

Theorem A.1. *If λ_π and λ_u are Lipschitz constants of π and u respectively, then $\lambda_\pi \leq \lambda_u/\sqrt{e}$. The same relationship holds for partial Lipschitz constants of π and u.*

Proof. We have $\pi(X) = \exp(-u^2(X)/2)$. Hence π can be expressed as a composition: $\pi = g \circ u$, where $g(u) := \exp(-u^2/2)$. Using chain rule, all partial derivatives of π can be written as $\frac{\partial \pi}{\partial x_i} = g'(u(X))\frac{\partial u}{\partial x_i}$. The (partial) Lipschitz constant can be computed as maximum (partial) derivative. For $g(u)$ we have $\lambda_g = \sup_u |g'(u)|$. As one can easily compute, this works out to $1/\sqrt{e}$. Thus using chain rule we obtain the desired result. $\square$

For the next theorem, we restate the definition of $d_{\mathcal{M}}$ from (2.8) more generally for any two points $\hat{o}_1$ and $\hat{o}_2$ in the 6D space of contact coordinates and surface normals:

$$d_{\mathcal{M}}(\hat{o}_1, \hat{o}_2) := \sqrt{\frac{||\hat{o}_1^{\text{pos}} - \hat{o}_2^{\text{pos}}||^2}{\sigma_{\text{pos}}^2} + \frac{||\hat{o}_1^{\text{nor}} - \hat{o}_2^{\text{nor}}||^2}{\sigma_{\text{nor}}^2}}. \tag{A.1}$$

For a vector $\Delta\hat{o}$, we can compute the corresponding norm $||\cdot||_{d_{\mathcal{M}}}$ by setting $\hat{o}_1 = \Delta\hat{o}$ and $\hat{o}_2 = 0$ in the above equation:

$$||\Delta\hat{o}||_{d_{\mathcal{M}}} := \sqrt{\frac{||\Delta\hat{o}^{\text{pos}}||^2}{\sigma_{\text{pos}}^2} + \frac{||\Delta\hat{o}^{\text{nor}}||^2}{\sigma_{\text{nor}}^2}}. \tag{A.2}$$

Theorem A.2. *The partial Lipschitz constants of $u(X)$ are bounded by:*

$$\begin{aligned} \lambda_{u,x}, \lambda_{u,y}, \lambda_{u,z} &\leq \sqrt{K}\tfrac{1}{\sigma_{\text{pos}}} \\ \lambda_{u,\alpha}, \lambda_{u,\beta}, \lambda_{u,\gamma} &\leq \sqrt{K}\sqrt{\tfrac{\mathcal{R}^2(\mathcal{O})}{\sigma_{\text{pos}}^2} + \tfrac{1}{\sigma_{\text{nor}}^2}}. \end{aligned} \tag{A.3}$$

Proof. We split the proof into two parts: differentiating w.r.t. the metric coordinates, and differentiating w.r.t. the angular coordinates.

Compute bounds on $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial y}$ and $\frac{\partial u}{\partial z}$:

Let $\Delta_x u_k$ denote the most u_k can change when x changes by at most Δx. When the object is moved by at most Δx, each point $\hat{o}$ in $\hat{\mathcal{O}}$ moves by a vector $\Delta\hat{o} = (\Delta\hat{o}^{\text{pos}}, 0)$, with $||\Delta\hat{o}||_{d_{\mathcal{M}}} = ||\Delta\hat{o}^{\text{pos}}||/\sigma_{\text{pos}} \leq \Delta x/\sigma_{\text{pos}}$. Hence by Lemma A.1, $\Delta_x u_k \leq \Delta x/\sigma_{\text{pos}}$ for all k.

By definition (2.11), we have $u(X) = \sqrt{\sum_k u_k^2(X)}$. Using triangle inequality we get

$$\Delta_x u \leq ||(\Delta_x u_1, ..., \Delta_x u_K)|| = \sqrt{\sum_k \Delta x^2/\sigma_{\text{pos}}^2}. \tag{A.4}$$

Hence $\Delta_x u \leq \sqrt{K}\Delta x/\sigma_{\text{pos}}$, and so $\frac{\partial u}{\partial x} \leq \sqrt{K}/\sigma_{\text{pos}}$. The same derivation applies for $\frac{\partial u}{\partial y}$ and $\frac{\partial u}{\partial z}$, giving us the top line of (A.3).

Compute bounds on $\frac{\partial u}{\partial \alpha}$, $\frac{\partial u}{\partial \beta}$ and $\frac{\partial u}{\partial \gamma}$:

When we consider partial derivative w.r.t. α, the logic is similar to the above, except that $\Delta\alpha$ acts on all components of $\hat{o}$. $\Delta\alpha$ rotation changes the normal $\hat{o}^{\text{nor}}$ by $\Delta\hat{o}^{\text{nor}}$ with $||\Delta\hat{o}^{\text{nor}}|| = 2\sin\frac{\Delta\alpha}{2}$. It also moves $\hat{o}^{\text{pos}}$ by $\Delta\hat{o}^{\text{pos}}$ with $||\Delta\hat{o}^{\text{pos}}|| \leq 2\mathcal{R}(\mathcal{O})\sin\frac{\Delta\alpha}{2}$, where $\mathcal{R}(\mathcal{O})$ is the radius of the object. Since $2\sin\frac{\Delta\alpha}{2} \leq \Delta\alpha$, we get

$$||\Delta\hat{o}^{\text{pos}}|| \leq \mathcal{R}(\mathcal{O})\Delta\alpha \tag{A.5}$$

and

$$||\Delta\hat{o}^{\text{nor}}|| \leq \Delta\alpha. \tag{A.6}$$

Combining the definition (A.2) of $||\cdot||_{d_{\mathcal{M}}}$ with (A.5) and (A.6), we get

$$||\Delta\hat{o}||_{d_{\mathcal{M}}} \leq \Delta\alpha\sqrt{\left(\mathcal{R}(\mathcal{O})\right)^2/\sigma^2_{\text{pos}} + 1/\sigma^2_{\text{nor}}}. \tag{A.7}$$

Again by Lemma A.1, we have $\Delta_\alpha u_k \leq ||\Delta\hat{o}||_{d_{\mathcal{M}}}$, and so for all k we have

$$\Delta_\alpha u_k \leq \Delta\alpha\sqrt{\left(\mathcal{R}(\mathcal{O})\right)^2/\sigma^2_{\text{pos}} + 1/\sigma^2_{\text{nor}}}. \tag{A.8}$$

Analogous to the derivation for x, we get

$$\frac{\partial u}{\partial\alpha} \leq \sqrt{K}\sqrt{\left(\mathcal{R}(\mathcal{O})\right)^2/\sigma^2_{\text{pos}} + 1/\sigma^2_{\text{nor}}}. \tag{A.9}$$

The same derivation applies for $\frac{\partial u}{\partial\beta}$ and $\frac{\partial u}{\partial\gamma}$, giving us the bottom line of (A.3). $\square$

Lemma A.1. *Let A be a point and S be a set of points in $\mathbb{R}^{dimA}$. Let $d(\cdot,\cdot)$ be a measure of distance and $||\cdot||_d$ the corresponding norm. Let $\Delta s \geq 0$ be a fixed constant. Let S' be a set of points obtained by displacing each point B of S to a point B' such that $||B - B'||_d \leq \Delta s$. Then*

$$d(A, S') \leq d(A, S) + \Delta s, \tag{A.10}$$

$$d(A, S) \leq d(A, S') + \Delta s, \tag{A.11}$$

and

$$|d(A, S') - d(A, S)| \leq \Delta s. \tag{A.12}$$

Proof. Let B be a point in S. By triangle inequality

$$||A - B'||_d \leq ||A - B||_d + ||B - B'||_d. \tag{A.13}$$

Since

$$||B - B'||_d \leq \Delta s \tag{A.14}$$

and

$$d(A, S') := \min_{B' \in S'} ||A - B'||_d, \tag{A.15}$$

we have

$$d(A, S') \leq ||A - B||_d + \Delta s. \tag{A.16}$$

Minimizing the above inequality over B, we obtain (A.10). Inequality (A.11) is obtained by swapping B with B' and S with S' in the above proof. Inequality (A.12) follows from (A.10) and (A.11). $\square$

Appendix B

Math Notes for Vehicle Shape Estimation

For RBPF, the measurement function has to be approximated by a Gaussian. This is done using *Laplace's method*. Then the computations for the weights and geometry belief (in Sect. 4.4.1) can be carried out in closed form. Details are provided below.

B.1 Laplace's Method

First, a quick description of Laplace's method taken from MacKay [2003]. A function $P^*(x)$ is approximated by an unnormalized Gaussian $Q^*(x)$ around a (hopefully) maximum point x_0. The variance of the Gaussian is usually computed via second derivatives, i.e.,

$$\frac{1}{\sigma^2} = -\frac{\partial^2}{\partial x^2} \ln P^*(x)|_{x=x_0}. \tag{B.1}$$

Then the approximation is made with an unnormalized Gaussian, so that the value at the maximum point remains the same:

$$Q^*(x) = P^*(x_0) \exp\left(-\frac{(x-x_0)^2}{\sigma^2}\right). \tag{B.2}$$

In our case, we need to approximate the measurement function $M_t := p(\mathcal{D}_t|\Omega, X_t)$

by a Gaussian *as a function of* Ω. Since it is not a probability distribution over Ω there is no requirement that it integrate to 1 over the domain of Ω. We fit the Gaussian at the maximum point $\Omega^* = \mu_1$. In order to find this maximum point, we perform a greedy search in both directions from the prior estimate of geometry. The search can be done very efficiently due to the construction of the measurement model, as only the new rays need to be analyzed and the rest of the measurement function can be re-used.

Note: using second derivative to estimate the variance of the Gaussian did not work well. Instead, the variance is set to 1m for a "good" observation or 100m otherwise. The observation is considered to be "good" if we are able to see some free space immediately after the end of the vehicle.

B.2 Importance Weight Computation

The importance weights can be shown to be (see Sect. B.2.2 for a derivation):

$$\begin{aligned} w_t &= \mathbb{E}_{S_{t-1}}\Big[\, p(\mathcal{D}_t|\Omega, X_t) \,\Big] \\ &= \int M_t S_{t-1}\, \mathrm{d}\Omega \\ &= M_t(\mu_1)\frac{\sigma}{\sigma_2}\exp\left(-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}\right), \end{aligned} \tag{B.3}$$

where M_t is the measurement function, μ_1 is the "observed" geometry (i.e., geometry that maximizes the measurement function), σ_1^2 is the variance of the Gaussian approximating the measurement function, S_{t-1} is the geometry prior, σ_2^2 is the variance of the geometry prior, and σ^2 is the variance of the geometry belief S_t computed as a product of two Gaussians:

$$S_t = \eta M_t S_{t-1}. \tag{B.4}$$

B.2.1 Product of Gaussians

Formulas for a product of two Gaussians from Smith [2008]:

$$\begin{aligned}\mu &= \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2},\\ \sigma^2 &= \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}.\end{aligned} \tag{B.5}$$

B.2.2 Derivations

First, let's figure out what constant remains when we replace a product of two Gaussians by a single Gaussian:

$$\frac{e^A e^B}{e^C}. \tag{B.6}$$

Taking logarithm, we can concentrate just on the exponents:

$$\begin{aligned} & A + B - C = \\ = \; & -\frac{(x-\mu_1)^2}{2\sigma_1^2} - \frac{(x-\mu_2)^2}{2\sigma_2^2} + \frac{(x-\mu)^2}{2\sigma^2} \\ = \; & -\frac{(x-\mu_1)^2}{2\sigma_1^2} - \frac{(x-\mu_2)^2}{2\sigma_2^2} + \frac{\left(x - \frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} \\ = \; & -\frac{\sigma_2^2x^2 - 2\sigma_2^2\mu_1 x + \sigma_2^2\mu_1^2 + \sigma_1^2x^2 - 2\sigma_1^2\mu_2 x + \sigma_1^2\mu_2^2}{2\sigma_1^2\sigma_2^2} \\ & + \frac{(\sigma_1^2+\sigma_2^2)x^2 - 2(\mu_1\sigma_2^2+\mu_2\sigma_1^2)x + \frac{\left(\mu_1\sigma_2^2+\mu_2\sigma_1^2\right)^2}{\sigma_1^2+\sigma_2^2}}{2\sigma_1^2\sigma_2^2} \\ = \; & \frac{(-\mu_1^2\sigma_2^2 - \mu_2^2\sigma_1^2)(\sigma_1^2+\sigma_2^2) + (\mu_1\sigma_2^2+\mu_2\sigma_1^2)^2}{2\sigma_1^2\sigma_2^2(\sigma_1^2+\sigma_2^2)} \\ = \; & \frac{\mu_1^2\sigma_2^4 + \mu_2^2\sigma_1^4 + 2\mu_1\mu_2\sigma_1^2\sigma_2^2 - \mu_1^2\sigma_2^4 - \mu_2^2\sigma_1^4 - \mu_1^2\sigma_1^2\sigma_2^2 - \mu_2^2\sigma_1^2\sigma_2^2}{2\sigma_1^2\sigma_2^2(\sigma_1^2+\sigma_2^2)} \\ = \; & -\frac{2\mu_1\mu_2 - \mu_1^2 - \mu_2^2}{2(\sigma_1^2+\sigma_2^2)} \\ = \; & -\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}. \end{aligned} \tag{B.7}$$

Now we are ready to derive the weights:

$$\begin{aligned}
w_t &= \int M_t S_{t-1}\,\mathrm{d}\Omega \\
&= \int M_t(\mu_1)\exp\left(-\frac{(\Omega-\mu_1)^2}{2\sigma_1^2}\right)\frac{1}{\sigma_2\sqrt{2\pi}}\exp\left(-\frac{(\Omega-\mu_2)^2}{2\sigma_2^2}\right)\,\mathrm{d}\Omega \\
&= \int M_t(\mu_1)\frac{1}{\sigma_2\sqrt{2\pi}}\exp\left(-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}\right)\exp\left(-\frac{(\Omega-\mu)^2}{2\sigma^2}\right)\,\mathrm{d}\Omega \\
&= M_t(\mu_1)\frac{1}{\sigma_2\sqrt{2\pi}}\exp\left(-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}\right)\int\exp\left(-\frac{(\Omega-\mu)^2}{2\sigma^2}\right)\,\mathrm{d}\Omega \\
&= M_t(\mu_1)\frac{1}{\sigma_2\sqrt{2\pi}}\exp\left(-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}\right)\sigma\sqrt{2\pi} \\
&= M_t(\mu_1)\frac{\sigma}{\sigma_2}\exp\left(-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}\right). \qquad \text{(B.8)}
\end{aligned}$$

In second line, we substitute the Gaussian approximations for M_t and S_{t-1}. In third line, we replace a product of two unnormalized Gaussians by a single Gaussian, which results in the extra constant derived in (B.7). In the forth line, we take constants outside of the integral sign. In the fifth line, integration of the unnormalized Gaussian results in its normalization constant. The sixth line gives the final cleaned up result.

Note: the derivations have been shown for a single real variable Ω. In our case, Ω has two real variables: W and L. However, these variables are independent, so we can simply factor it down to the single variable case. The final expression for the weights used in our implementation is:

$$w_t = M_t(\mu_{L_1},\mu_{W_1})\frac{\sigma_L}{\sigma_{L_2}}\frac{\sigma_W}{\sigma_{W_2}}\exp\left(-\frac{(\mu_{L_1}-\mu_{L_2})^2}{2(\sigma_{L_1}^2+\sigma_{L_2}^2)}\right)\exp\left(-\frac{(\mu_{W_1}-\mu_{W_2})^2}{2(\sigma_{W_1}^2+\sigma_{W_2}^2)}\right). \qquad \text{(B.9)}$$

Appendix C

Proofs for GRAB

Proof of Theorem 6.2. To see the left hand side of (6.18), consider that $|Z - \widehat{Z}| = |\int \pi - \widehat{\pi}|$ and $\|\pi - \widehat{\pi}\|_{\mathsf{L}_1} = \int |\pi - \widehat{\pi}|$. Since $|\int f| \leq \int |f|$ for any integrable function f, we can obtain the inequality by setting $f := \pi - \widehat{\pi}$.

To see the right hand side of (6.18), let V_{prune} be the union of all grid cells we pruned and V_{keep} be the union of grid cells we kept at the last iteration. By definition $\|\pi - \widehat{\pi}\|_{\mathsf{L}_1} = \int |\pi - \widehat{\pi}|$. We can split up this integral into a sum of two terms: an integral over V_{prune} and an integral over V_{keep}. Since $\widehat{\pi}$ is zero everywhere in V_{prune}, the first term is simply $\int_{V_{prune}} \pi$, which is bounded from above by ε_{prune}.

To bound the integral over V_{keep}, consider one of the grid cells G_i kept at the last iteration. Everywhere in G_i, both π and $\widehat{\pi}$ are bounded by the variation bounds $U_i^{(N)}$ and $L_i^{(N)}$. Hence, $|\pi - \widehat{\pi}| \leq U_i^{(N)} - L_i^{(N)}$ everywhere in G_i. Integrating this inequality over all grid cells kept at the last iteration and taking into account (6.15), we obtain $\int_{V_{keep}} |\pi - \widehat{\pi}| \leq \varepsilon_{keep}$. Thus $\int |\pi - \widehat{\pi}| \leq \varepsilon_{prune} + \varepsilon_{keep} = \varepsilon$.

Lastly, to obtain inequality ((6.19)), we have the following derivation:

$$\begin{aligned}
|bel - \widehat{bel}| &= \left|\tfrac{1}{Z}\pi - \tfrac{1}{\widehat{Z}}\widehat{\pi}\right| = \left|\left(\tfrac{1}{Z}\pi - \tfrac{1}{Z}\widehat{\pi}\right) + \left(\tfrac{1}{Z}\widehat{\pi} - \tfrac{1}{\widehat{Z}}\widehat{\pi}\right)\right| \\
&= \left|\tfrac{1}{Z}(\pi - \widehat{\pi}) + \left(\tfrac{1}{Z} - \tfrac{1}{\widehat{Z}}\right)\widehat{\pi}\right| \\
&\leq \tfrac{1}{Z}\left|\pi - \widehat{\pi}\right| + \left|\tfrac{1}{Z} - \tfrac{1}{\widehat{Z}}\right|\widehat{\pi} \\
&= \tfrac{1}{Z}\left|\pi - \widehat{\pi}\right| + \tfrac{|Z - \widehat{Z}|}{Z\widehat{Z}}\widehat{\pi}.
\end{aligned} \tag{C.1}$$

Integrating the obtained inequality over V, we have

$$\begin{aligned}\textstyle\int |bel - \widehat{bel}| &\le \textstyle\int \frac{1}{Z}\left|\pi - \widehat{\pi}\right| + \int \frac{|Z-\widehat{Z}|}{Z\widehat{Z}}\widehat{\pi} \\ &\le \textstyle\frac{1}{Z}\varepsilon + \frac{1}{Z}|Z - \widehat{Z}| \le \frac{2\varepsilon}{Z} \le \frac{2\varepsilon}{\widehat{Z}-\varepsilon}.\end{aligned} \tag{C.2}$$

The last step follows from the fact that $\widehat{Z} - \varepsilon \le Z$ and thus $1/Z \le 1/(\widehat{Z} - \varepsilon)$ as long as $\widehat{Z} - \varepsilon > 0$. □

Bibliography

Agrawal, M. and K. Konolige (2006). Real-time localization in outdoor environments using stereo vision and inexpensive gps. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, Volume 3, pp. 1063–1068. IEEE.

Anguelov, D., R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun (2002). Learning hierarchical object maps of non-stationary environments with mobile robots. In *Proc. of UAI*.

Anguelov, D., D. Koller, E. Parker, and S. Thrun (2004). Detecting and modeling doors with mobile robots. In *Proc. of ICRA*.

Arulampalam, S., S. Maskell, N. Gordon, and T. Clapp (2002). A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*.

Balan, A., L. Sigal, and M. Black (2005). A quantitative evaluation of video-based 3D person tracking. In *ICCCN*, Volume 5, pp. 349–356.

Biber, P. and T. Duckett (2005). Dynamic maps for long-term operation of mobile service robots. In *Proc. of Robotics: Science and Systems (RSS)*.

Bicchi, A., A. Marigo, and D. Prattichizzo (1999). Dexterity through rolling: Manipulation of unknown objects. In *ICRA*, pp. 1583–1588.

Biswas, R., B. Limketkai, S. Sanner, and S. Thrun (2002). Towards object mapping in dynamic environments with mobile robots. In *Proc. of IROS*.

Blackman, S., R. Co, and C. El Segundo (2004). Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine 19*(1 Part 2), 5–18.

Blais, F. (2004). Review of 20 years of range sensor development. *Journal of Electronic Imaging 13*, 231.

Boyd, S. and L. Vandenberghe (2004). *Convex optimization.* Cambridge Univ Pr.

Bruynincks, H., S. Demey, S. Dutré, and J. D. Schutter (1995, Oct). Kinematic models for model-based compliant motion in the presence of uncertainty. *Int. J. of Robotics Research 14*(5), 465–482.

Buehler, M., K. Iagnemma, and S. Singh (2007). *The 2005 Darpa Grand Challenge: The Great Robot Race.* Springer Verlag.

Burgard, W., A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun (1998). The interactive museum tour-guide robot. *AAAI*, 11–18.

Burgard, W., A. Derr, D. Fox, and A. B. Cremers (1998). Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach. In *IROS.*

Burgard, W., D. Fox, D. Hennig, and T. Schmidt (1996). Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 896–901.

Censi, A. (2009). On achievable accuracy for pose tracking. In *ICRA.*

Charlebois, M., K. Gupta, and S. Payandeh (1996). Curvature based shape estimation using tactile sensing. In *IEEE International Conference on Robotics and Automation*, pp. 3502–3507.

Chhatpar, S. and M. Branicky (2005). Particle filtering for localization in robotic assemblies with position uncertainty. In *IROS.*

Chiaverini, S., B. Siciliano, and L. Villani (1999, September). A survey of robot interaction control schemes with experimental comparison. *ASME Transactions on Mechatronics 4*(3), 273–285.

Cho, M. and T. Seo (2002). Inspection planning strategy for the on-machine measurement process based on CAD/CAM/CAI integration. *The International Journal of Advanced Manufacturing Technology 19*(8), 607–617.

Chu, Y. (1999). *Workpiece localization: Theory, algorithms and implementation.* Ph. D. thesis, The Hong Kong University of Science and Technology.

Coleman, T. F. and Y. Li (1996). An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*.

Corcoran, C. and R. Platt (2010). A measurement model for tracking handobject state during dexterous manipulation. In *ICRA*.

Cortesão, R. (2002). *Kalman Techniques for Intelligent Control Systems: Theory and Robotic Experiments.* Ph. D. thesis, University of Coimbra.

Darms, M., P. Rybski, C. Urmson, C. Inc, and M. Auburn Hills (2008). Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments. In *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 1197–1202.

DARPA (2007, Dec.). Urban challenge rules, revision oct. 27, 2007.

Davison, A. (2003). Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *ICCV*.

Dellaert, F. and C. Thorpe (1998). Robust car tracking using kalman filtering and bayesian templates. In *Proceedings of SPIE*, Volume 3207, pp. 72.

Deutscher, J., A. Blake, and I. Reid (2000). Articulated body motion capture by annealed particle filtering. In *CVPR*.

Deutscher, J. and I. Reid (2005). Articulated body motion capture by stochastic search. *International Journal of Computer Vision 61*(2), 185–205.

Dickmanns, E. (1998). Vehicles capable of dynamic vision: a new breed of technical beings? *Artificial Intelligence 103*(1-2), 49–76.

Dietmayer, K., J. Sparbert, and D. Streller (2001). Model based object classification and object tracking in traffic scenes from range images. In *Proceedings of IV 2001, IEEE Intelligent Vehicles Symposium*, pp. 25–30.

Doucet, A. and N. De Freitas (2001). *Sequential Monte Carlo Methods in Practice*. Springer.

Doucet, A., N. d. Freitas, K. Murphy, and S. Russell (2000). Rao-blackwellised filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, San Francisco, CA*, pp. 176–183.

Drummond, T. and R. Cipolla (2002). Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*(7), 932–946.

Eliazar, A. and R. Parr (2004). DP-SLAM 2.0. In *ICRA*.

Evans, M. and T. Swartz (2000). *Approximating Integrals via Monte Carlo and Deterministic Methods*. Oxford University Press.

Faugeras, O. and M. Hebert (1983). A 3-d recognition and positioning algorithm using geometrical matching between primitive surfaces. In *Eighth Intl. Joint Conf. on Artificial Intelligence, Los Altos, CA*, pp. 996–1002.

Featherstone, R., S. S. Thiebaut, and O. Khatib (1999). A general contact model for dynamically-decoupled force/motion control. In *Proc. of the Int. Conf. on Robots and Automation*, pp. 3281–3286.

Fischler, M. and R. Bolles (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*(6), 381–395.

Fishman, G. S. (1996). *Monte Carlo: concepts, algorithms, and applications*. Springer.

Flanagan, J. R., M. C. Bowman, and R. S. Johansson (2006). Control strategies in object manipulation tasks. *Current Opinion in Neurobiology*.

Fox, D. (2001). KLD-sampling: Adaptive particle filters. In *Proc. of NIPS*.

Fox, D. (2003). Adapting the sample size in particle filters through KLD-sampling. *IJRR*.

Fox, D., W. Burgard, F. Dellaert, and S. Thrun (1999). Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence*.

Fox, D., W. Burgard, and S. Thrun (1999). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research 11*(3), 391–427.

Freund, E. (1975). The structure of decoupled nonlinear systems. *Int. J. Contr. 21*(3), 443–450.

Gadeyne, K. and H. Bruyninckx (2001). Markov techniques for object localization with force-controlled robots. In *ICAR*.

Gaston, P. C. and T. Lozano-Perez (1983). Tactile recognition and localization using object models. Technical report, MIT, AIM-705.

Genz, A. and R. Kass (1997). Subregion adaptive integration of functions having a dominant peak. *Journal of Computational and Graphical Statistics*.

Gerkey, B., V. R. T., and A. Howard (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *ICAR*.

Gerkey, B., R. Vaughan, and A. Howard (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, pp. 317–323.

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica: Journal of the Econometric Society*, 1317–1339.

Gordon, N. (1993). *Bayesian methods for tracking.* Ph. D. thesis, University of London.

Grimson, W. E. L. and T. Lozano-Perez (1983). Model-based recognition and localization from sparse range or tactile data. *Journal of Robotics Research.*

Grisetti, G., C. Stachniss, and W. Burgard (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on 23*(1), 34–46.

Gunnarsson, K. (1987, May). *Optimal part localization by data base matching with sparse and dense data.* Ph. D. thesis, Dept. of Mechanical Engineering, Carnegie Mellon Univ.

Gunnarsson, K. and F. Prinz (1987). CAD model-based localization of parts in manufacturing. *Computer;(United States) 20*(8).

Hähnel, D., R. Triebel, W. Burgard, and S. Thrun (2003). Map building with mobile robots in dynamic environments. In *Proc. of ICRA*.

Harris, C. (1993). Tracking with Rigid Objects. *Active Vision*, 59–73.

Hong, J. and X. Tan (1993, May 4). Method and apparatus for determining position and orientation of mechanical objects. US Patent 5,208,763.

Horn, B. (1987). Closed-form solution of absolute orientation using unit quaternions. *JOSA A 4*(4), 629–642.

Hsiao, K., L. Kaelbling, and T. Lozano-Perez (2010, June). Task-driven tactile exploration. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain.

Huang, Y. and X. Qian (2008). An Efficient Sensing Localization Algorithm for Free-Form Surface Digitization. *Journal of Computing and Information Science in Engineering 8*, 021008.

Ibeo Automobile Sensor GmbH (2008, Dec.). IBEO AlascaXT brochure.

Isard, M. and A. Blake (1998). A smoothing filter for condensation. *Computer Vision—ECCV'98*, 767–781.

Isard, M., J. MacCormick, and K. Achan (2008). Continuously-adaptive discretization for message-passing algorithms. In *NIPS*, pp. 737–744. MIT Press.

Jain, A. and C. Kemp (2009). Behavior-based door opening with equilibrium point control. In *RSS Workshop: Mobile Manipulation in Human Environments.*

Kabadayi, S., A. Pridgen, and C. Julien (2006). Virtual sensors: Abstracting data from physical sensors. In *Proceedings of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks*, pp. 587–592. IEEE Computer Society Washington, DC, USA.

Kaneko, M. and T. Tsuji (2001). Pulling motion based tactile sensing. In *Algorithmic and computational robotics: new directions: the fourth Workshop on the Algorithmic Foundations of Robotics*, pp. 157. AK Peters, Ltd.

Kemp, C., A. Edsinger, and E. Torres-Jara (2007). Challenges for robot manipulation in human environments [grand challenges of robotics]. *IEEE Robotics & Automation Magazine 14*(1), 20–29.

Khatib, O. (1987, February). A unified approach for motion and force control of robot manipulators: The operational space formulation. *Int. J. on Robotics and Automation 3*(1), 43–53.

Khatib, O., L. Sentis, J. Park, and J. Warren (2004). Whole-body dynamic behavior and control of human-like robots. *Int. J. of Humanoid Robotics 1*(1), 29–43.

Kozlov, A. and D. Koller (1997). Nonuniform dynamic discretization in hybrid networks. In *Proc. UAI*, pp. 314–325. Morgan Kaufmann.

Kragic, D. and H. I. Christensen (2002). Survey on visual servoing for manipulation. Technical Report ISRN KTH/NA/P–02/01–SE, Royal Institute of Technology (KTH).

Lenser, S. and M. Veloso (2000). Sensor resetting localization for poorly modelled mobile robots. *ICRA 2*.

Leonard, J., J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, et al. (2008). A perception-driven autonomous urban vehicle. *Journal of Field Robotics 25*(10), 727–774.

Lepetit, V. and P. Fua (2005). Monocular model-based 3D tracking of rigid objects. *Foundations and Trends® in Computer Graphics and Vision 1*(1), 1–89.

Lew, J. Y. and W. J. Book (1994). Bracing micro/macro manipulators control. In *ICRA*, pp. 2362–2368.

Lipkin, H. and J. Duffy (1988, Jun). Hybrid twist and wrench control for a robotic manipulator. *ASME Journal of Mechanisms, Transmissions and Automation in Design 110*(2), 138–144.

Liu, Y., K. Kitagaki, T. Ogasawara, and S. Arimoto (1999, January). Model-based adaptive hybrid control for manipulators under multiple geometric constraints. *IEEE Trans. on Control Systems Technology 7*(1), 97–109.

Liu, Z., Z. Shi, M. Zhao, and W. Xu (2008). Adaptive dynamic clustered particle filtering for mobile robots global localization. *Journal of Intelligent and Robotic Systems 53*(1), 57–85.

MacCormick, J. and A. Blake (2000). A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision 39*(1), 57–71.

MacCormick, J. and M. Isard (2000). Partitioned sampling, articulated objects, and interface-quality hand tracking. *Computer Vision—ECCV 2000*, 3–19.

MacKay, D. J. C. (1998). Introduction to Monte Carlo methods. In M. I. Jordan (Ed.), *Learning in Graphical Models*, NATO Science Series, pp. 175–204. Kluwer Academic Press.

MacKay, D. J. C. (2003). *Information theory, inference, and learning algorithms.* Cambridge, UK: Cambridge University Press.

Mäkynen, A. (2000, October). *Position-sensitive devices and sensor systems for optical tracking and displacement sensing applications.* Ph. D. thesis, Department of Electrical Engineering, Oulu University, Oulu, Finland.

Mandelbrot, B. (1982). *The fractal geometry of nature.* Freeman.

Menq, C., H. Yau, and G. Lai (1992). Automated precision measurement of surface profile in CAD-directed inspection. *Robotics and Automation, IEEE Transactions on 8*(2), 268–278.

Mitchell, T., R. Hutchinson, M. Just, S. Newman, R. Stefan, N. Francisco, and P. X. Wang (2002). Machine learning of fmri virtual sensors of cognitive states. In *In The 16th Annual Conference on Neural Information Processing Systems, Computational Neuroimaging: Foundations, Concepts and Methods Workshop, 2002. 87.* MIT Press.

Moll, M. (2002). *Shape Reconstruction Using Active Tactile Sensors.* Ph. D. thesis, University of California.

Moll, M. and M. A. Erdmann (2003). *Reconstructing the Shape and Motion of Unknown Objects with Active Tactile Sensors*, pp. 293–310. Springer Verlag.

Montemerlo, M. (2003). *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association.* Ph. D. thesis, Robotics Institute, Carnegie Mellon University.

Montemerlo, M., J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al. (2008). Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics 25*(9), 569–597.

Montemerlo, M., J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Hähnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer,

A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun (2008). Junior: The stanford entry in the urban challenge. *Journal of Field Robotics 25*(9), 569–597.

Montemerlo, M., J. Pineau, N. Roy, S. Thrun, and V. Verma (2002). Experiences with a mobile robotic guide for the elderly. *AAAI*, 587–592.

Moravec, H. (1988). Sensor fusion in certainty grids for mobile robots. *AI Magazine 9*(2), 61.

Murphy, K. and S. Russell (2001). *Rao-blackwellized particle filtering for dynamic bayesian networks.* Springer.

Olson, C. (2000). Probabilistic self-localization for mobile robots. *IEEE Transactions on Robotics and Automation 16*(1), 55–66.

Pahk, H. and W. Ahn (1996). Precision inspection system for aircraft parts having very thin features based on CAD/CAI integration. *The International Journal of Advanced Manufacturing Technology 12*(6), 442–449.

Park, J. (2006). *Control Strategies for Robots in Contact.* Ph. D. thesis, Stanford University.

Park, J., R. Cortesão, and O. Khatib (2004). Multi-contact compliant motion control for robotic manipulators. In *Proc. of the Int. Conf. on Robotics and Automation*, New Orleans, U.S.A., pp. 4789–4794.

Park, J. and O. Khatib (2005). Mult-link multi-contact force control for manipulators. In *Proc. of the Int. Conf. on Robotics and Automation*, Barcelona, Spain, pp. 3613–3618.

Petersson, L., D. Austine, and D. Kragic (2000). High-level control of a mobile manipulator for door opening. In *IROS*.

Petrovskaya, A. (2011a, May). *Home Page of Anna Petrovskaya*. http://cs.stanford.edu/people/petrovsk.

Petrovskaya, A. (2011b). *Robotic Perception via Contact*. http://cs.stanford.edu/people/petrovsk/tactile.html.

Petrovskaya, A. and O. Khatib (2011). Global localization of objects via touch. *IEEE Transactions on Robotics 27*(3), (forthcoming).

Petrovskaya, A., O. Khatib, S. Thrun, and A. Y. Ng (2006, May). Bayesian estimation for autonomous object manipulation based on tactile sensors. In *IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, Florida, USA, pp. 707–714.

Petrovskaya, A., O. Khatib, S. Thrun, and A. Y. Ng (2007, June). Touch based perception for object manipulation. In *Robotics: Science and Systems (RSS), Robot Manipulation Workshop*, Atlanta, GA, USA.

Petrovskaya, A. and A. Ng (2007, January). Probabilistic mobile manipulation in dynamic environments, with application to opening doors. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India.

Petrovskaya, A., J. Park, and O. Khatib (2007, April). Probabilistic estimation of whole body contacts for multi-contact robot control. In *IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, pp. 568–573.

Petrovskaya, A. and S. Thrun (2008a, July). Efficient techniques for dynamic vehicle detection. In *International Symposium on Experimental Robotics (ISER)*, Athens, Greece.

Petrovskaya, A. and S. Thrun (2008b, June). Model based vehicle tracking for autonomous driving in urban environments. In *Robotics: Science and Systems (RSS)*, Zurich, Switzerland.

Petrovskaya, A. and S. Thrun (2009a, April). Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots 26*(2), 123–139.

Petrovskaya, A. and S. Thrun (2009b, May). Model based vehicle tracking in urban environments. In *IEEE International Conference on Robotics and Automation (ICRA), Workshop on Safe Navigation*, Volume 1, Kobe, Japan, pp. 1–8.

Petrovskaya, A., S. Thrun, D. Koller, and O. Khatib (2010a, June). Guaranteed Inference for Global State Estimation in Human Environments. In *Robotics: Science and Systems (RSS), Mobile Manipulation Workshop*, Zargoza, Spain.

Petrovskaya, A., S. Thrun, D. Koller, and O. Khatib (2010b, June). Towards dependable perception: Guaranteed inference for global localization. In *Dependable Robots in Human Environments (DRHE), 7th IARP Workshop*, Toulouse, France.

Plagemann, C. (2008, December). *Gaussian Processes for Flexible Robot Learning.* Ph. D. thesis, University of Freiburg.

Platt Jr, R., F. Permenter, and J. Pfeiffer (2010). Inferring hand-object configuration directly from tactile data. In *Mobile Manipulation Workshop at ICRA*.

Prats, M., P. Sanz, and A. del Pobil (2010, August). Reliable non-prehensile door opening through the combination of vision, tactile and force feedback. *Autonomous Robots 29*(2), 1–18.

Rahimi, A. and T. Darrell (2002). Location Estimation with a Differential Update Network. In *NIPS*, pp. 1049–1056.

Raibert, M. H. and J. J. Craig (1981, June). Hybrid position/force control of manipulators. *ASME Journal of Dynamic Systems, Measurement and Control 103*(2), 126–133.

Rezaei, S. and R. Sengupta (2007). Kalman filter-based integration of dgps and vehicle sensors for localization. *Control Systems Technology, IEEE Transactions on 15*(6), 1080–1088.

Rhee, C., W. Chung, M. Kim, Y. Shim, and H. Lee (2004). Door opening control using the multi-fingered robotic hand for the indoor service robot. In *Proc. of ICRA*.

Särkkä, S., A. Vehtari, and J. Lampinen (2007). Rao-blackwellized particle filter for multiple target tracking. *Inf. Fusion 8*(1).

Schaeffer, M. and A. M. Okamura (2003). Methods for intelligent localization and mapping during haptic exploration. In *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*.

Schulz, D., W. Burgard, D. Fox, and A. Cremers (2001). Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *IEEE International Conference on Robotics and Automation, 2001. Proceedings 2001 ICRA*, Volume 2.

Schutter, J. D., J. Rutgeerts, E. Aertbelien, F. D. Groote, T. D. Laet, T. Lefebvre, W. Verdonck, and H. Bruyninckx (2005). Unified constraint-based task specification for complex sensor-based robot systems. In *Proc. of the Int. Conf. on Robotics and Automation*, Barcelona, Spain, pp. 3607–3612.

Shekhar, S., O. Khatib, and M. Shimojo (1986). Sensor fusion and object localization. In *Proc. of ICRA*.

Siciliano, B. and L. Villani (1999). *Robot Force Control*. The Kluwer International Series In Engineering and Computer Science. Kluwer Academic Publishers.

Sick Optics (2003). LMS 200/211/220/221/291 laser measurement systems technical description.

Sidky, E., R. Chartrand, and X. Pan (2007). Image reconstruction from few views by non-convex optimization. In *Nuclear Science Symposium Conference Record, 2007. NSS'07. IEEE*, Volume 5, pp. 3526–3530. IEEE.

Slaets, P., J. Rutgeerts, K. Gadeyne, T. Lefebvre, H. Bruyninckx, and J. De Schutter (2004). Construction of a geometric 3-D model from sensor measurements collected during compliant motion. In *Proc. of ISER*.

Smith, J. O. (2008). *Spectral Audio Signal Processing, October 2008 Draft*. http://ccrma.stanford.edu/~jos/sasp/. online book.

Someya, T., T. Sekitani, S. Iba, Y. Kato, H. Kawaguchi, and T. Sakurai (2004, July). A large-area, flexible pressure sensor matrix with organic field-effect transistors for artificial skin applications. In *National Academy of Sciences*.

Srivastava, A., N. Oza, J. Stroeve, N. Aeronaut, S. Center, and C. Moffett Field (2005). Virtual sensors: Using data mining techniques to efficiently estimate remote sensing spectra. *IEEE Transactions on Geoscience and Remote Sensing 43*(3), 590–600.

Stachniss, C. and W. Burgard (2005). Mobile robot mapping and localization in non-static environments. In *Proc. of AAAI*.

Streller, D., K. Furstenberg, and K. Dietmayer (2002). Vehicle and object models for robust tracking in traffic scenes using laser range images. In *The IEEE 5th International Conference on Intelligent Transportation Systems, 2002. Proceedings*, pp. 118–123.

Sudderth, E., A. Ihler, W. Freeman, and A. Willsky (2003). Nonparametric belief propagation. In *CVPR*, Volume 1, pp. 605.

Sun, Y., J. Xu, D. Guo, and Z. Jia (2009). A unified localization approach for machining allowance optimization of complex curved surfaces. *Precision Engineering 33*(4), 516–523.

Thalmann, D., H. Noser, and Z. Huang (1997). Autonomous virtual actors based on virtual sensors. *Lecture Notes In Computer Science 1195*, 25–42.

Thrun, S. (2001). A probabilistic on-line mapping algorithm for teams of mobile robots. *The International Journal of Robotics Research 20*(5), 335.

Thrun, S., W. Burgard, and D. Fox (2005). *Probabilistic Robotics.* MIT Press.

Thrun, S., D. Fox, W. Burgard, and F. Dellaert (2001). Robust monte carlo localization for mobile robots. *Artificial Intelligence 128*(1-2), 99–141.

Urmson, C., J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics 25*(8), 425–466.

Velodyne Lidar, Inc. (2008, Dec.). High definition lidar HDL-64E S2 specifications.

Vu, T. (2009, September). *Vehicle Perception: Localization, Mapping with Detection, Classification and Tracking of Moving Objects.* Ph. D. thesis, INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE.

Wang, C., C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte (2007, Sep.). Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research 26*, 889–916.

Wang, C.-C. (2004, April). *Simultaneous Localization, Mapping and Moving Object Tracking.* Ph. D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Wender, S. and K. Dietmayer (2008). 3d vehicle detection using a laser scanner and a video camera. *Intelligent Transport Systems, IET 2*(2), 105–112.

West, H. and H. Asada (1985). A method for the design of hybrid position/force controllers for manipulators contrained by contact with the environment. In *Proc. of the Int. Conf. on Robotics and Automation*, pp. 251–259.

Willow Garage (2009, May). *ROS — Robot Open Source.* http://www.willowgarage.com/pages/software/ros-platform.

Wolf, D. and G. S. Sukhatme (2004). Online simultaneous localization and mapping in dynamic environments. In *Proc. of ICRA*.

Xiong, Z. (2002). *Workpiece Localization and Computer Aided Setup System.* Ph. D. thesis, The Hong Kong University of Science and Technology.

Xiong, Z., M. Wang, and Z. Li (2004). A near-optimal probing strategy for workpiece localization. *Robotics, IEEE Transactions on 20*(4), 668–676.

Yau, H. and C. Menq (1992). An automated dimensional inspection environment for manufactured parts using coordinate measuring machines. *International journal of production research 30*(7), 1517–1536.

Yedidia, J., W. Freeman, and Y. Weiss (2003). Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium 8*, 236–239.

Yeung, M., J. Tegnér, and J. Collins (2002). Reverse engineering gene networks using singular value decomposition and robust regression. *Proceedings of the National Academy of Sciences of the United States of America 99*(9), 6163.

Yoshikawa, T. (2000). Force control of robot manipulators. In *Proc. of ICRA*.

Zhao, L. and C. Thorpe (1998). Qualitative and quantitative car tracking from a range image sequence. In *1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Proceedings*, pp. 496–501.

Zhu, L., H. Luo, and H. Ding (2009). Optimal Design of Measurement Point Layout for Workpiece Localization. *Journal of Manufacturing Science and Engineering 131*, 011006.

Zhu, L., Z. Xiong, H. Ding, and Y. Xiong (2004). A distance function based approach for localization and profile error evaluation of complex surface. *Journal of manufacturing Science and Engineering 126*, 542.

Zielke, T., M. Brauckmann, and W. von Seelen (1993). Intensity and edge-based symmetry detection with an application to car-following. *CVGIP: Image Understanding 58*(2), 177–190.

Index