ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Dissertation zur Erlangung des Doktorgrades
der Fakultät für Angewandte Wissenschaften der
Albert-Ludwigs-Universität Freiburg im Breisgau

Gaussian Processes for Flexible Robot Learning

Christian Plagemann

Dezember, 2008

Betreuer: Prof. Dr. Wolfram Burgard

Dekan der Fakultät für Angewandte Wissenschaften: Prof. Dr. Bernhard Nebel

1. Gutachter: Prof. Dr. Wolfram Burgard, Universität Freiburg
2. Gutachter: Prof. Dr. Martin Riedmiller, Universität Osnabrück

Tag der Disputation: 01.12.2008

Abstract

As robotic systems are becoming more complex and as they are developed for increasingly realistic environments, the role of machine learning within robotics becomes even more central than it has been. Robot learning tasks typically raise a number of hard requirements for learning methods including accurate estimation of uncertainty, dealing with inhomogeneously sampled data, computational efficiency, and high modeling accuracy.

We show in this thesis that Gaussian processes as a machine learning method have the potential to address these issues. Gaussian processes are a Bayesian approach to function regression, that is, they allow to place a prior distribution over the space of functions. The approach has become one of the standard tools for solving non-linear regression problems, which are central to many machine learning problems. While the modeling capabilities of Gaussian processes are as high as for state-of-the-art alternatives, the basic concept is exceptionally clear and easy to implement.

We present Gaussian process-based solutions to a wide variety of problems encountered in robotics. This includes the modeling of laser and gas sensors, the interpretation of camera images, the detection of failures during mobile robot navigation, and the estimation of probabilistic terrain maps from noisy range measurements. Furthermore, we introduce an approach that allows a robot to learn a model of its own body from scratch using only visual self-observation. In each of the studied problem domains, our solutions have been evaluated empirically using real and simulated data.

As a contribution to machine learning in general, we study and improve extensions of the standard Gaussian process model, which can be used flexibly to solve non-robotics learning tasks also. This includes an approach to dealing with spatially-varying observation noise and one for adapting the function smoothness locally. We furthermore derive sparse approximations for these extensions to be able to apply the models to large data sets and in online settings.

Zusammenfassung

Durch die rapide zunehmende Komplexität von Robotersystemen und ihrer Einsatzumgebungen kommt dem maschinellen Lernen als Grundlagentechnologie eine noch höhere Bedeutung innerhalb der Robotik zu als dies in der Vergangenheit der Fall war. Relevante Lernprobleme in der Robotik stellen hohe Anforderungen an die eingesetzten Lernverfahren. Dies beinhaltet ein hohes Maß an Modellierungsgenauigkeit, den verlässlichen Umgang mit Unsicherheiten, Speicher- und Zeiteffizienz, sowie Robustheit gegenüber fehlenden und verrauschten Daten. Mit dieser Arbeit zeigen wir, dass Gauß-Prozesse als Verfahren zur Regressionsanalyse das Potential haben, diesen Anforderungen in relevanten Anwendungen gerecht zu werden. Gauß-Prozesse sind ein etablierter und konzeptionell klarer Ansatz aus der Statistik, mit dessen Hilfe Verteilungen über Funktionen definiert und praktische Regressionsprobleme gelöst werden können.

Ausgehend von den hohen Anforderungen der Robotik leiten wir eine Reihe von Erweiterungen für Gauß-Prozesse her, die sich auch außerhalb der Robotik als leistungfähig erwiesen haben. Dies beinhaltet einen neuen Ansatz zur Modellierung von inhomogenem Meßrauschen sowie einen Ansatz zur lokalen Adaption der Funktionsglattheit an die gegebenen Daten. Darüber hinaus werden Approximationen der erweiterten Modelle zur Effizienzsteigerung behandelt, die das Lernen auf großen Datenmengen und unter Resourcenbeschränkungen möglich macht. Diese Erweiterungen stellen einerseits einen Beitrag zum maschinellem Lernen im Allgemeinen dar und helfen andererseits, eine breitgefächerte Menge an Robotikproblemen zu lösen. Die in dieser Arbeit ausführlich behandelten Probleme umfassen die Modellierung von Laser- und Gassensoren, die Interpretation von Kamerabildern, die automatische Fehlerdetektion für mobile Roboter, sowie die Erstellung von probabilistischen Geländemodellen. Desweiteren stellen wir einen Ansatz vor, durch den ein Roboter seine eigene kinematische Struktur durch visuelle Selbstobservation modellieren und kontrollieren kann. Für jede dieser Problemstellungen wurde ein experimenteller Vergleich zu existierenden Lösungen durchgeführt und eine Diskussion der Vorteile des Gauß-Prozess Ansatzes geführt.

Acknowledgments

I would like to thank all the wonderful people who have contributed to this work and who have made my PhD such a pleasant experience.

First of all, many thanks to Wolfram Burgard for being a great adviser with the right balance of high expectations, practical guidance, and encouragement. He has granted me an exceptional amount of freedom for developing my own ideas and he has always had an open door for discussing and improving them. I will remember Freiburg as a formative and highly enjoyable period of my live.

I would also like to thank Martin Riedmiller for reviewing this thesis and for acting as an external referee.

Many thanks to Kristian Kersting and Cyrill Stachniss for not only being great friends but also mentors in scientific and personal matters. Many thanks also to my other friends and colleagues at the AIS lab for the great atmosphere and the many productive collaborations: Kai Arras, Maren Bennewitz, Barbara Frank, Giorgio Grisetti, Slawomir Grzonka, Dirk Hähnel, Dominik Joho, Andreas Karwarth, Rainer Kümmerle, Boris Lau, Matthias Luber, Óscar Martinez Mozos, Daniel Meyer-Delius, Patrick Pfaff, Axel Rottmann, Bastian Steder, Hauke Strasdat, Jürgen Sturm, Gian Diego Tipaldi, Rudolph Triebel, and Kai Wurm. Many thanks to the exceptional MSc students who I co-supervised in Freiburg: Sebastian Mischke, Hendrik Kretzschmar, Matthias Luber, Tobias Lang, Jürgen Hess, Felix Endres, Slawomir Grzonka, and Alexandru Cocora. The inspiring discussion that we have had about their projects were as useful for my own work as they, hopefully, were for theirs. My thanks also to Susanne Bourjaillat and Michael Keser for providing administrative and technical help in the lab.

I am also grateful to Luc de Raedt, Dieter Fox, and Achim Lilienthal for the fruitful discussions when we met in Freiburg or at conferences, and for the various collaborations.

Many thanks to Nicholas Roy and Leslie Pack Kaelbling for making my stays at MIT possible and for the many stimulating discussions we had there. My thanks also to the

other people I have met and worked with in Boston, especially to Sam Prentice, Alexander Bahr, Katie Byl, Alec Shkolnik, Russ Tedrake, and Christopher Paciorek.

I am thankful to Roland Siegwart and his colleagues at EPFL Lausanne and ETH Zurich for realizing the autonomous car project SmartTer and letting me be part of it. It was a great experience to see such a complex system develop from early stages on, driven by an enthusiastic team and well-organized coordinators. Many thanks also to the collaborators in the EU project CoSy and the German DESIRE project for helping to design, develop, and present our results. These are especially Michael Brenner, Bernhard Nebel, and the researchers of Siemens Corporate Technology in Munich.

Many thanks to Katja Bay for helping me polish the language and style of this thesis and also to my "scientific reviewers" in the later stages of writing: Barbara, Cyrill, Daniel, Giorgio, Jürgen, Kai, Kristian, Sebastian, and Slawomir. Any remaining obscurities and errors in the document are entirely mine.

Finally and most importantly, I would like to thank Stephanie and our family for the enduring love and support. You are really special and I know how much I owe you.

This work has partly been supported by the European Community under contract number FP6-004250-CoSy and by the German Ministry for Education and Research (BMBF) through the DESIRE project. Their support is gratefully acknowledged.

To Stephanie and our family

Contents

Chapter 1

Introduction

Machine learning and probabilistic modeling have a strong impact on robotics research and their importance is increasing further as systems are becoming more complex and as they are deployed to less-constrained environments. State-of-the-art robotic systems are designed to solve tasks such as recognizing and manipulating objects, understanding natural scenes, navigating in populated environments, playing soccer in teams, driving in traffic, or walking on uneven terrain. It is immediately apparent, that such domains cannot be modeled by human experts in all necessary detail. Even seemingly simple subproblems such as the interpretation of range measurements from a precise laser sensor are often hard to solve due to the complexity of realistic environments and the inherent uncertainties in key variables (e.g., the sensor location). Other popular sensors, like cameras or radio-frequency identification (RFID) antennas, additionally complicate the perception task by having higher noise rates and by measuring the quantities of interest only indirectly. For this reason, a current trend in robotics research and engineering is (a) to model dependencies probabilistically taking the uncertainties on multiple levels into account and (b) to employ advanced machine learning techniques for fitting the models to collected data.

The goal of this thesis is to contribute to the understanding of robot learning problems, especially within probabilistic contexts, and to evaluate and improve the Gaussian process (GP) model as a tool for solving them. Many relevant robot learning problems can be posed as regression problems, that is, as learning the mapping from index vectors to target values given a set of observed data points so that generalization to unobserved parts of the space is possible and observation noise is reduced. In this thesis, we argue that the Gaussian process approach to regression provides a particularly flexible and powerful framework for solving robot learning problems. Gaussian processes have a long tradition in statistics and geoscience, but have gained popularity as a practical machine learning tool

just in the last decade [Rasmussen and Williams, 2006b]. GPs are a Bayesian approach to function regression, i.e., they allow to place a prior distribution over the space of functions. While the modeling capabilities of Gaussian processes are as high as for state-of-the-art alternatives, the basic concept is exceptionally clear and easy to understand.

We present Gaussian process-based solutions to a wide variety of learning problems encountered in robotics, including perception, state estimation, self-modeling, and terrain mapping. In general, we take the probabilistic approach in order to deal with the noise and uncertainties inherent in such domains. We show that the Gaussian process model is advantageous especially in this context since it directly provides uncertainty estimates for its predictions covering the observation noise as well as the uncertainty arising from missing observations. This feature enables the tight integration with other probabilistic approaches such as Monte Carlo localization or occupancy grid mapping.

As a contribution to the general field of machine learning, we studied and improved extensions of the standard Gaussian process model, which proved helpful also in a number of non-robotics learning tasks. This includes an extension for dealing with inhomogeneous observation noise and one for learning and representing spatially-varying smoothness. In both cases, we model the hidden, location-dependent variables—noise and smoothness respectively—by means of additional, latent Gaussian processes and give efficient approximations for learning these from training data. We furthermore discuss ways of achieving sparse Gaussian process approximations making the model applicable to large data sets and in online settings.

1.1 Contributions

With this work, we contribute to *robotics research* by developing solutions to a number of relevant and hard robot learning problems that arise in the context of perception, state estimation, self-modeling, and terrain mapping. The diversity of problems to which we have applied Gaussian processes successfully may been seen as an indication for the flexibility of the approach. Specifically, we address in this work:

- Online failure detection for mobile robots (Chapter 3)
- Body scheme learning from visual self-observation (Chapter 4)
- Gas distribution mapping (Chapter 5)
- Robust laser-based Monte Carlo localization (Chapter 7)
- Range sensing from monocular vision (Chapter 8)
- Probabilistic terrain mapping and path planning (Chapters 9 and 10)

Additionally, we contribute to *machine learning research* by developing extensions of the Gaussian process framework within the following areas and by advancing the understanding of the methods in real-world settings:

- Heteroscedastic GPs—modeling input-dependent noise (Chapter 6)
- Nonstationary GPs—modeling input-dependent smoothness (Chapters 9 and 11)
- Sparse Approximations—scaling to large data sets (Chapters 10 and 11)

Based on existing theoretic concepts discussed in the literature, our approaches are designed to meet the requirements of robot learning tasks including accurate estimation of uncertainty, computational efficiency, and high modeling accuracy.

The results of this work furthermore include a software toolbox for `Matlab` and `C/C++` as well as a collection of benchmark data sets. This and additional multimedia material is contained on the website `http://robreg.org/gprl`.

Publications

The work presented in this thesis is based on the following publications in conference proceedings and journals. They are given in reverse chronological order, grouped by their main subject.

Gaussian Processes for Machine Learning

C. Plagemann, K. Kersting, and W. Burgard. Nonstationary gaussian process regression using point estimates of local smoothness. In *Proc. of the European Conference on Machine Learning (ECML)*, Antwerp, Belgium, 2008.

K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic gaussian process regression. In *International Conference on Machine Learning (ICML)*, Corvallis, Oregon, USA, 2007.

Modeling Sensors and Perception

P. Pfaff, C. Stachniss, C. Plagemann, and W. Burgard. Efficiently learning high-dimensional observation models for monte-carlo localization using gaussian mixtures. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

M. Luber, K. Arras, C. Plagemann, and W. Burgard. Tracking and classification of dynamic objects: An unsupervised learning approach. In *Robotics: Science and Systems (RSS)*, Zurich, Switzerland, 2008.

C. Plagemann, F. Endres, J. Hess, C. Stachniss, and W. Burgard. Monocular range sensing: A non-parametric learning approach. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

P. Pfaff, C. Plagemann, and W. Burgard. Gaussian mixture models for probabilistic localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

P. Pfaff, C. Plagemann, and W. Burgard. Improved likelihood models for probabilistic localization based on range scans. In *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

C. Plagemann, K. Kersting, P. Pfaff, and W. Burgard. Gaussian beam processes: A nonparametric bayesian measurement model for range finders. *Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

C. Plagemann, T. Müller, and W. Burgard. Vision-based 3d object localization using probabilistic models of appearance. In *Pattern Recognition, 27th DAGM Symposium, Vienna, Austria*, volume 3663 of *Lecture Notes in Computer Science*, pages 184–191. Springer, 2005.

State Estimation and Mapping

S. Grzonka, C. Plagemann, G. Grisetti, and W. Burgard. Look-ahead proposals for robust grid-based slam with rao-blackwellized particle filters. *International Journal of Robotics Research (IJRR)*, 2008. To appear.

C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard. Learning predictive terrain models for legged robot locomotion. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

H. Kretzschmar, C. Stachniss, C. Plagemann, and W. Burgard. Estimating landmark locations from geo-referenced photographs. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

C. Stachniss, C. Plagemann, A. Lilienthal, and W. Burgard. Gas distribution modeling using sparse gaussian process mixture models. In *Robotics: Science and Systems (RSS)*, Zurich, Switzerland, 2008.

S. Grzonka, C. Plagemann, G. Grisetti, and W. Burgard. Look-ahead proposals for robust grid-based slam. In *Proc. of the International Conference on Field and Service Robotics (FSR)*, Chamonix, France, 2007.

T. Lang, C. Plagemann, and W. Burgard. Adaptive non-stationary kernel regression for terrain modeling. In *Robotics: Science and Systems (RSS)*, Atlanta, Georgia, USA, 2007.

Robotic Self-Perception and Control

J. Sturm, C. Plagemann, and W. Burgard. Adaptive body scheme models for robust robotic manipulation. In *Robotics: Science and Systems (RSS)*, Zurich, Switzerland, 2008.

J. Sturm, C. Plagemann, and W. Burgard. Unsupervised body scheme learning through self-perception. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

A. Rottmann, C. Plagemann, P. Hilgers, and W. Burgard. Autonomous blimp control using model-free reinforcement learning in a continuous state and action space. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

C. Plagemann, D. Fox, and W. Burgard. Efficient failure detection on mobile robots using particle filters with gaussian process proposals. In *Proc. of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007.

C. Plagemann, C. Stachniss, and W. Burgard. Efficient failure detection for mobile robots using mixed-abstraction particle filters. In *European Robotics Symposium 2006*, volume 22 of *STAR Springer tracts in advanced robotics*, pages 93–107. Springer-Verlag Berlin Heidelberg, Germany, 2006.

C. Plagemann and W. Burgard. Sequential parameter estimation for fault diagnosis in mobile robots using particle filters. In *Autonome Mobile Systeme 2005 (AMS)*, pages 197–202. Springer, 2005.

Relational Modeling

K. Kersting, C. Plagemann, A. Cocora, W. Burgard, and L. De Raedt. Learning to transfer optimal navigation policies. *Advanced Robotics. Special Issue on Imitative Robots*, 21(9), 2007.

C. Meyer-Delius, C. Plagemann, G. von Wichert, W. Feiten, G. Lawitzky, and W. Burgard. A probabilistic relational model for characterizing situations in dynamic multi-agent systems. In *Proc. of the 31st Annual Conf. of the German Classification Society on Data Analysis, Machine Learning, and Applications (GfKl)*, Freiburg, Germany, 2007.

A. Cocora, K. Kersting, C. Plagemann, W. Burgard, and L. De Raedt. Learning relational navigation policies. *KI - Künstliche Intelligenz, Themenheft Lernen und Selbstorganisation von Verhalten*, 3:12–18, Böttcher IT Verlag, Bremen, Germany, 2006.

A. Cocora, K. Kersting, C. Plagemann, W. Burgard, and L. De Raedt. Learning relational navigation policies. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

1.2 Collaborations

Parts of this thesis have resulted from collaboration with colleagues and researchers from other institutions. We are grateful for their advice, ideas, and hard work they put into the joint projects. Specifically, *Heteroscedastic GPs* (Chapter 6) and many of the machine-learning–related concepts were developed jointly with Kristian Kersting. *Terrain regression*, especially the application to path planning and legged robot locomotion (Chapter 10), was studied jointly with Nicholas Roy. Together with Dieter Fox, we developed the concepts underlying the learning of sampling models for dynamic Bayesian networks introduced in Chapter 3. *Nonstationary covariance functions* and their applications to terrain modeling (Chapter 9) were originally addressed in the co-supervised MSc thesis of Tobias Lang. *Sparse approximations*, especially the segmentation-based approaches (Chapters 10 and 11), were originally studied in the co-supervised MSc thesis of Sebastian Mischke. Our approach to *monocular range sensing* (Chapter 8) is based on joint work with Felix Endres and Jürgen Hess. *Range sensor models* (Chapter 7) were studied in cooperation with Patrick Pfaff. The approach to *gas distribution modeling* (Chapter 5) was developed in a joint project with Cyrill Stachniss and Achim Lilienthal. *Body scheme learning for manipulation* (Chapter 4) was investigated jointly with Jürgen Sturm.

1.3 Outline

This thesis is structured in three main parts.

In *Part I: Gaussian Process Regression in Robotics*, we first review the Gaussian process model for regression and compare it to related approaches (Chapter 2). Then we show how the model can be used to solve problems concerning sampling models for dynamic Bayesian

networks (Chapter 3) and robotic self-modeling for manipulation (Chapter 4). A Gaussian process-based mixture model for mapping gas distributions is presented in Chapter 5.

In the second part, *Modeling Input-Dependent Noise*, we introduce a novel approach to heteroscedastic regression, i.e., the modeling of input-dependent observation noise (Chapter 6). Based on this model, we present solutions to robot perception problems that arise with range sensors (Chapter 7), and we develop a system for predicting range from monocular camera images (Chapter 8).

In the third part, *Modeling Input-Dependent Smoothness*, we introduce a nonstationary GP approach to terrain modeling, which locally adapts its smoothness prior to the data (Chapter 9). In Chapter 10, we apply an extended model, which employs sparse approximations to deal with large data sets, to the problem of learning terrain models for legged robot locomotion. Finally, in Chapter 11, we derive a general nonstationary Gaussian process model from first principles and present an evaluation on simulated and real-world data.

Chapter 12, finally, recapitulates the contributions of the thesis and discusses its results. We will furthermore discuss future research directions for the individual topics as well as for the Gaussian process approach in general.

1.4 Symbols and Notation

Symbol	Description
x	scalar variable
$\mathbf{x}$	vector
$\mathcal{U}_{\mathbf{x}}$	local neighborhood of $\mathbf{x}$
$[\mathbf{x}]_i$	i-th element of $\mathbf{x}$
$\mathbf{A}$	matrix
$[\mathbf{A}]_{ij}$	element in the i-th row and j-th column of $\mathbf{A}$
$\{\dots\}$	set
$(\dots)$	vector
$\langle\dots\rangle$	tuple (ordered set)
$\mathrm{diag}(\mathbf{x}), \mathbf{x} \in \mathbb{R}^n$	$n \times n$ matrix with $\mathbf{x}$ on the diagonal
$\lvert\mathbf{A}\rvert$	determinant of $\mathbf{A}$
$\mathbf{I}$	identity matrix
$p(x)$	probability density function for the variable x
$P(E)$	probability of an event E
$\mathcal{N}(\mu, \sigma^2)$	normal distribution with mean μ and variance σ^2
$\mathbb{E}(x), \mathbb{V}(x)$	expectation and variance of a random variable x
$\mathrm{cov}(x, y)$	covariance of the random variables x and y
$\mathbf{p}_t = (x_t, y_t, \theta_t) \in \mathbb{R} \times \mathbb{R} \times (-\pi : \pi]$	pose of mobile robot consisting of its location and orientation relative to an environment
$z_t, t \in \mathbb{N}$	time-indexed sensor measurement
$\mathbf{z} = \langle(r_i, \boldsymbol{\alpha}_i)\rangle_{i=1}^m$	range scan that consists of m beam measurements r_i indexed by beam orientations $\boldsymbol{\alpha}_i$.
$z_{s:t} = \{z_s, \dots, z_t\}$	sequence of sensor measurements
u_t	action / motion command issued at time t
$\mathcal{B}_t = \{\langle s_t^{[i]}, w_t^{[i]}\rangle\}_{i=1}^n$	belief state (set of weighted state samples)
$f(\cdot)$	regression function / process.
$f_i = f(\mathbf{x}_i)$	the function evaluated at input location $\mathbf{x}_i$
$\mathcal{GP}_y$	Gaussian process over the variable y
$\mathcal{GP}_\ell, \mathcal{GP}_z$	Latent Gaussian processes over (log) length-scales ℓ and log noise rates $z = \log(\sigma_n^2)$
$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d, y \in \mathbb{R}$	training set consisting of *target* samples y_i, which are indexed by corresponding *input* vectors $\mathbf{x}_i$

Symbol	Description
$\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^d$	set of training points $\mathbf{x}_i$
$\mathcal{D} = \langle \mathbf{X}, \mathbf{y} \rangle$, $\mathbf{X} \in \mathbb{R}^{n \times d}, \mathbf{y} \in \mathbb{R}^n$	alternative notation in matrix-vector form
$y_*, f_*, \mathbf{x}_*$, $\mathbf{y}_* = (y_1^*, \ldots, y_m^*)^\top$	new target to be predicted, the corresponding latent function value, and its input location
$k_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$	covariance function
$\mathbf{K}_{\mathbf{xx}} \in \mathbb{R}^{n \times n}$	covariance matrix
$\mathbf{k}_{\mathbf{x}_*\mathbf{x}} \in \mathbb{R}^n$	vector of covariates between the query target y_* at $\mathbf{x}_*$ and the training targets $\mathbf{y}$ at $\mathbf{X}$
σ_n	noise level which the observed targets are affected by
$\boldsymbol{\theta}$	hyperparameters
$\boldsymbol{\theta}_{SE} = \langle \sigma_f, \ell \rangle$	hyperparameters of the square exponential (SE) covariance function, consisting of the signal variance (or: amplitude) and the characteristic length-scale
$\boldsymbol{\theta}_\ell = \langle \overline{\sigma}_f, \overline{\sigma}_\ell, \overline{\sigma}_n \rangle$	hyperparameters of the latent length-scale process $\mathcal{GP}_\ell$ (using the square exponential (SE) covariance function)
$\mathcal{L}$	objective function for optimization, typically derived from the logarithm of the marginal data likelihood of the training set

Abbreviation	Description
BN	Bayesian network
CV	cross validation
CDF	cummulative distribution function, $F_x(t) := P(\{x < t\})$
CPD	conditional probability distribution, $p(x\|y)$
DBN	dynamic Bayesian network
DF	discrete filter (also: histogram filter)
DOF	degree of freedom
GP	Gaussian process
GPP	Gaussian process proposal
KF	Kalman filter
MCL	Monte Carlo localization
NLPD	negative log predictive density
NN	neural network
PDF	probability density function
PF	particle filter
RBF	radial basis function
RBPF	Rao-Blackwellized particle filter
SE	squared exponential
sMSE	standardized mean squared error
w.l.o.g	without loss of generality

Part I

Gaussian Process Regression in Robotics

Chapter 2

Basics

We review the Gaussian process approach to regression as the central framework for the remainder of the thesis and discuss its relation to alternative models.

As stated in the introduction, the goal of this thesis is to study how to build artifical systems with advanced learning capabilities. One of the central problems that arises in this context is the so-called *regression problem*, that is, learning about a function $\mathbf{x} \mapsto y$ that maps points $\mathbf{x}$ from an index space to real-valued targets y. Realizations of these variables can be given, for example, (a) in form of an explicit training set supplied by a tutor, (b) from own observations, or (c) as realizations of variables in own internal models—such as the value function in a reinforcement learning context. In the following, we first formalize the non-linear regression problem in Sec. 2.1, then we describe the Gaussian process model as an approach to solving it in Sec. 2.2, and discuss the relationship of the model to alternative ones from the literature in Sec. 2.3.

2.1 The Non-Linear Regression Problem

In supervised regression, we are given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, of n observed tuples $(\mathbf{x}_i, y_i)$. The $\mathbf{x}_i$ are used as indices to the set, called *inputs*, and the y_i are values of interest, called *targets*. The task is to make predictions about new targets y_* given the corresponding inputs $\mathbf{x}_*$. As $\mathcal{D}$ does not contain all possible $\mathbf{x}$ in general and there might be more than one possible response y to an input $\mathbf{x}$ (i.e., for a *noisy* relationship between $\mathbf{x}$ and y), we face a problem of induction [Mitchell, 1997]. That is, one has to make additional assumptions about the distribution of data points in order to get a well-posed problem. More visually, we have to assume some relationship

between observed and unobserved variables in order to make predictions about the latter ones. Such additional assumptions are called the *inductive bias* of the learning algorithm. The bias chosen for regression tasks in often based on the concept of *similarity*, that is, similar indices $\mathbf{x}$ are more likely to be associated to the same target y than dissimilar ones.

More formally, non-linear regression analysis[1] aims at learning a model for the noisy relationship

$$y = f(\mathbf{x}) + \epsilon \ , \tag{2.1}$$

which is composed of a *systematic* component $f(\cdot)$ and a *noise* component ϵ. The noise terms are typically assumed i.i.d. Gaussians, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, but this does not have to be the case in general.

The parametric approach to the regression problem is to assume that f comes from a certain family of parametric functions $\{f_{\boldsymbol{\theta}}\}$ and to reason in the space of parameter vectors $\boldsymbol{\theta}$. Such models include for example the *linear model* $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$ in which $\boldsymbol{\theta}$ is the vector of coefficients $\mathbf{w}$ and *artifical neural networks* (ANNs) for which $\boldsymbol{\theta}$ encodes the weights and other parameters of the network (see also Sec. 2.3). In the following section, we introduce Gaussian processes as an approach to the regression problem that does not assume a finite parameterization of the function f. Rather, it allows to place a prior directly over the space of functions, thus defining the inductive bias independently from any structural assumption about f.

2.2 Gaussian Process Models

The goal is to derive a probabilistic model for recovering the systematic component $f(\cdot)$ in Eq. (2.1) from noisy data, that is, we seek a model for $p(f(\mathbf{x})|\mathbf{x})$ given the noise model $p(\epsilon)$. Since $f(\cdot)$ is a continuous function—thus having infinitely many dimensions—and we do not want to make a parametric assumption about it, we need a *distribution over functions* to define the inductive bias. A flexible way of placing distributions over functions is given by the Gaussian process model, which is defined as follows.

Definition: A Gaussian process is a collection of random variables, any of which have a joint Gaussian distribution.

More formally, if we assume that $\{(\mathbf{x}_i, f_i)\}_{i=1}^n$ with $f_i = f(\mathbf{x}_i)$ are samples from a Gaussian process and define $\mathbf{f} = (f_1, \ldots, f_n)^\top$, we have

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}) \ , \quad \boldsymbol{\mu} \in \mathbb{R}^n, \mathbf{K} \in \mathbb{R}^{n \times n} \ . \tag{2.2}$$

[1] The term "regression" is often used with the linear case in mind. Therefore, we explicitely include "non-linear" in our descriptions to make clear that we are dealing with the more general case.

For simplicity of notation, we can assume $\boldsymbol{\mu} = \mathbf{0}$, since the expectation is a linear operator and, thus, for any deterministic mean function $m(\mathbf{x})$, the Gaussian process over $\overline{f}(\mathbf{x}) := f(\mathbf{x}) - m(\mathbf{x})$ has zero mean (see [Rasmussen and Williams, 2006b, Sec. 2.7]).

The interesting part of the model is indeed the covariance matrix $\mathbf{K}$. It is specified by $[\mathbf{K}]_{ij} := \mathrm{cov}(f_i, f_j) = k(\mathbf{x}_i, \mathbf{x}_j)$ using a *covariance function* k which defines the covariance of any two function values $\{f_i, f_j\}$ sampled from the process given their input vectors $\{\mathbf{x}_i, \mathbf{x}_j\}$ as parameters.

Intuitively, the covariance function specifies how *similar* two function values $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ are depending only on the corresponding inputs. Popular covariance functions used in practice include the *squared exponential*

$$k_{SE}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left[-\frac{1}{2}\frac{|\mathbf{x}_i - \mathbf{x}_j|^2}{\ell^2}\right] \tag{2.3}$$

and a variant of the *Matérn* type of covariance function

$$k_M(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \left(1 + \frac{\sqrt{5}|\mathbf{x}_i - \mathbf{x}_j|}{\ell} + \frac{\sqrt{5}|\mathbf{x}_i - \mathbf{x}_j|^2}{3\ell^2}\right) \cdot \exp\left(-\frac{\sqrt{5}|\mathbf{x}_i - \mathbf{x}_j|}{\ell}\right). \tag{2.4}$$

Both functions are *stationary*, that is, they only depend on the relative distance $|\mathbf{x}_i - \mathbf{x}_j|$ of the input vectors rather than on their absolute locations. Thus, the so-called *length-scale* ℓ defines the global smoothness of the function f. Both functions are also *isotropic*, i.e., the different dimensions of $\mathbf{x}$ are all treated in the same way. σ_f denotes the amplitude (or signal variance) parameter. These parameters, along with the global noise variance σ_n^2 that is assumed for the noise component [see Eq. (2.1)], are known as the *hyperparameters* of the process. They are denoted as $\boldsymbol{\theta} = \langle \sigma_f, \ell, \sigma_n \rangle$.

Making Predictions

Given the definition of a Gaussian process as a collection of jointly Gaussian distributed random variables, it is relatively straight-forward to derive a closed-form solution for *predicting* new function values f_* at input locations $\mathbf{x}_*$, that is, estimating

$$p(f_* | \mathbf{x}_*, \mathcal{D}) .$$

Here, $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ denotes the set of observed input/target pairs. For convenience, we denote with $\mathbf{f}_*$ the vector of m function values f_i^* to be predicted and with $\mathbf{X}_*$ the matrix containing the corresponding index vectors $\mathbf{x}_i^*$, $i = 1, \ldots, m$, in the rows. Since *any* set of samples from the process is jointly Gaussian distributed, the set $\{f_1, \ldots, f_n, f_1^*, \ldots, f_m^*\}$ is

also:

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{xx}} & \mathbf{K}_{\mathbf{xx}_*} \\ \mathbf{K}_{\mathbf{xx}_*}^\top & \mathbf{K}_{\mathbf{x}_*\mathbf{x}_*} \end{bmatrix}\right) . \tag{2.5}$$

Here, we used the notation $\mathbf{K}_{\mathbf{xx}} \in \mathbb{R}^{n\times n}$ with $[\mathbf{K}_{\mathbf{xx}}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{K}_{\mathbf{xx}_*} \in \mathbb{R}^{n\times m}$ with $[\mathbf{K}_{\mathbf{xx}_*}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j^*)$ as well as $\mathbf{K}_{\mathbf{x}_*\mathbf{x}_*} \in \mathbb{R}^{m\times m}$ with $[\mathbf{K}_{\mathbf{x}_*\mathbf{x}_*}]_{ij} = k(\mathbf{x}_i^*, \mathbf{x}_j^*)$. As we do not know the latent, noisy-free function values $\mathbf{f}$ in Eq. (2.5), but only have the corresponding noisy observations $\mathbf{y} = (y_1, \ldots, y_n)^\top$ available, we consider

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{xx}} + \sigma_n^2\mathbf{I} & \mathbf{K}_{\mathbf{xx}_*} \\ \mathbf{K}_{\mathbf{xx}_*}^\top & \mathbf{K}_{\mathbf{x}_*\mathbf{x}_*} \end{bmatrix}\right) , \tag{2.6}$$

where $\mathbf{I}$ is the identity matrix. Conditioning Eq. (2.6) on the known $\mathbf{y}$ can be done analytically—applying the marginalization and conditioning rules for the Gaussian distribution, see [Rasmussen and Williams, 2006b, A.2]. This yields the sought-after result for the predictive distribution

$$\mathbf{f}_*|\mathbf{X}_*, \mathcal{D} \;\sim\; \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \tag{2.7}$$

with

$$\boldsymbol{\mu}_* = \mathbf{K}_{\mathbf{xx}_*}(\mathbf{K}_{\mathbf{xx}} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} \tag{2.8}$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{\mathbf{x}_*\mathbf{x}_*} - \mathbf{K}_{\mathbf{xx}_*}(\mathbf{K}_{\mathbf{xx}} + \sigma_n^2\mathbf{I})^{-1}\mathbf{K}_{\mathbf{xx}_*}^\top . \tag{2.9}$$

Note from Eq. (2.8) that the *posterior* mean of the Gaussian processes can take arbitrary values depending on the training data set $\mathcal{D}$ and the query locations $\mathbf{X}_*$—although we have assumed a *zero-mean prior* process. In practice, the mean of the GP prior is often irrelevant, at least as long as sufficiently many training points have been observed in the proximity of the query locations. Also note that predicting new observations $\mathbf{y}_*$ rather than function values $\mathbf{f}_*$ is easy. Due to the assumption of independent additive noise, one just has to add the noise variance to the diagonal entries of $\boldsymbol{\Sigma}_*$,

$$\mathbf{y}_*|\mathbf{X}_*, \mathcal{D} \;\sim\; \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_* + \sigma_n^2\mathbf{I}) , \tag{2.10}$$

with $\boldsymbol{\mu}_*$ and $\boldsymbol{\Sigma}_*$ as defined in Eq. (2.8) and (2.9). For simplicity of notation, we will w.l.o.g. often consider *single* predictions only in the remainder of this thesis. The predictive distribution in this case is

$$y_*|\mathbf{x}_*, \mathcal{D} \;\sim\; \mathcal{N}(\mu_*, \sigma_*^2) \tag{2.11}$$

with

$$\mu_* = \mathbf{k}_{\mathbf{x}\mathbf{x}_*}^\top (\mathbf{K}_{\mathbf{x}\mathbf{x}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \tag{2.12}$$

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_{\mathbf{x}\mathbf{x}_*}^\top (\mathbf{K}_{\mathbf{x}\mathbf{x}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_{\mathbf{x}\mathbf{x}_*} + \sigma_n^2 \,, \tag{2.13}$$

where $\mathbf{k}_{\mathbf{x}\mathbf{x}_*} \in \mathbb{R}^n$ with $[\mathbf{k}_{\mathbf{x}\mathbf{x}_*}]_i = k(\mathbf{x}_i, \mathbf{x}_*)$.

2.2.1 Properties and Remarks

Gaussian processes belong to the so-called *nonparametric* methods. This does, of course, not imply that the model completely lacks parameters. Rather, the term "nonparametric" means that the hypothesis space cannot be parameterized by a finite set of parameters. This is clearly not possible for GPs as we cannot give a finite representation for a *single* hypothesis, i.e., for a function. Note that for these reasons, the term "hyperparameters" is typically used for the parameters of the covariance function.

Alternative to the *function-space view* on Gaussian processes, which was outlined above, the model can be derived from a different starting point as we now briefly restate (see also [Rasmussen and Williams, 2006b, Kuss, 2006]). Following the so-called *weight-space view*, one can project the inputs $\mathbf{x}$ of the linear model $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$ into a high-dimensional feature space $\phi(\mathbf{x})$, then place a Gaussian prior on the weights, and marginalize out the weights to effectively remove the explicit representation of the functions. This yields the same regression model as derived above while highlighting that the GP's covariance function subsumes both the feature transform as well as the priors on the weights.

For a third way of deriving the GP model, which is similar to the weight-space approach, consider an artifical neural feedforward network with one hidden layer. If we place Gaussian prior distributions on the network's parameters, let the number of hidden units go to infinity, and simultaneously scale down the prior variances of the output weights, then we achieve a distribution over output functions, in which each set of function values is jointly Gaussian distributed, i.e., which is a Gaussian process [MacKay, 2003, Ch. 45].

The relationship between the Gaussian process model and parametric models can be studied using the so-called *equivalent kernel* (EK) [Sollich and Williams, 2004]. Here, the GP mean predictions [see Eq. (2.12)] are expressed as dot products $\mu(\mathbf{x}_*) = \mathbf{h}(\mathbf{x}_*)^T \mathbf{y}$ (here for the 1D case) of the vector of target values $\mathbf{y}$ and weight vectors $\mathbf{h}(\mathbf{x}_*)$. The weight function $\mathbf{h}(\mathbf{x}_*)$, aka the equivalent kernel, depends on the query location $\mathbf{x}_*$, on the covariance function k, and also on the actual distribution and density of points in $\mathcal{D}$. Due to the inversion of the covariance matrix in Eq. (2.12), the EK is hard to estimate even for the comparably simple case of the stationary squared exponential k_{SE}. Sollich and Williams [2004] derive

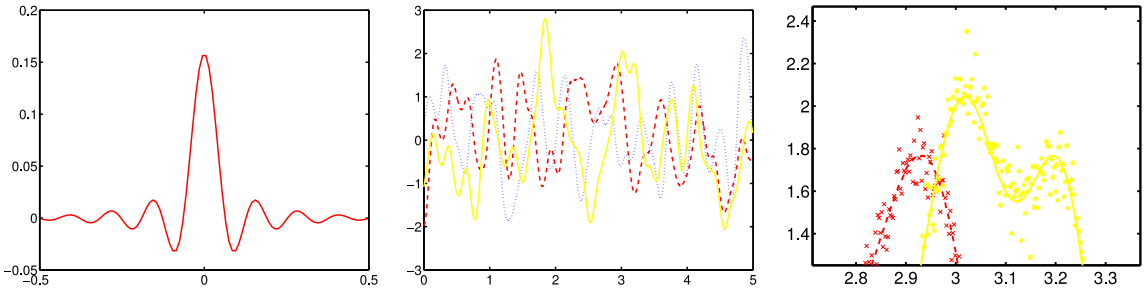


Figure 2.1: Left: The *equivalent kernel* (EK) for computing the GP mean prediction μ_* at $x_* = 0$ using the squared exponential covariance function and equidistantly sampled data. Middle: Functions f drawn randomly from the prior corresponding to the kernel on the left. Right: Enlarged section of the middle diagram including noisy samples y drawn from the GP.

an approximation for this case, which we have used to visualize the EK in the left diagram in Fig. 2.1. The diagram shows the EK computed for the query location $x_* = 0$ using the squared exponential covariance function k_{SE} and an equidistantly sampled $\mathcal{D}$. It can be seen that the shape of the EK differs considerably from the shape of the covariance function [i.e., a Gaussian, see Eq. (2.3)], but some properties, like being strongly localized, remain.

To give an intuition for GP priors over functions in practice, we sampled three functions using the same covariance function and plotted the results in the middle diagram in Fig. 2.1. The right diagram depicts an enlarged section including *noisy* samples from the prior.

2.2.2 Learning the Model from Data

In the GP model, we encode the prior knowledge about the distribution of target functions in the covariance function and the assumption about the observation noise. Inference of the posterior distribution for a given data set can then be performed analytically as described above. *Learning* a Gaussian process model thus amounts to finding an appropriate kernel for the problem, i.e., the form and parameterization of the covariance function. This process is termed *model selection*. Note that some literature is rather vague on what "learning a GP model" actually refers to. While most works—including this one—refer to this as the model selection problem, some others assume given, fixed kernel parameters and discuss only the analytic posterior inference step (inverting the covariance matrix, etc.) as "learning".

The *learning objective* is to find the model that explains the observed data best. Concretely, given a parametric covariance function, we seek to find the vector of hyperparameters $\boldsymbol{\theta}$

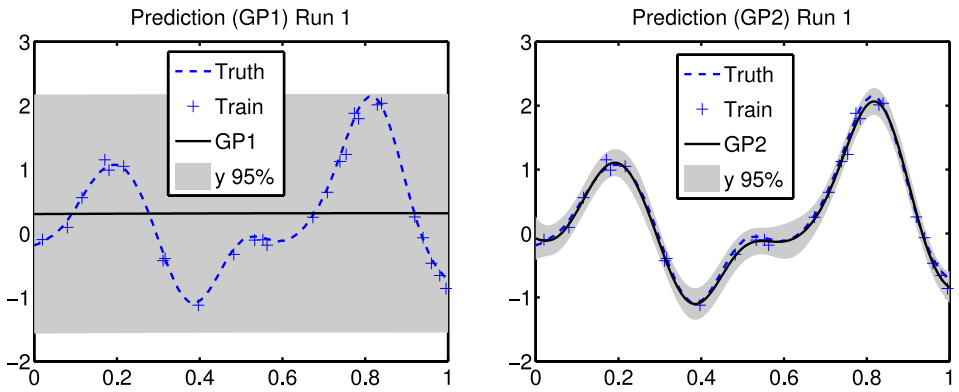


Figure 2.2: Two alternative GP models fitted to a 1D data set. The learning procedure that resulted in the model on the right found a noise rate hyperparameter close to the true value. The log predictive likelihood of this model is significantly higher than the one for the left model, which overestimates the noise.

that maximizes the conditional evidence

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) \,. \tag{2.14}$$

Since the elements of $\mathbf{y}$ in $\mathcal{D}$ are independent samples from the Gaussian process, $p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ takes the convenient form of a multivariate Gaussian density,

$$p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) \quad = \quad \left[(2\pi)^{\frac{n}{2}} |\mathbf{K}_\mathbf{y}|^{\frac{1}{2}}\right]^{-1} \cdot \exp\left(-\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu}_*)^\top \mathbf{K}_\mathbf{y}^{-1} (\mathbf{y} - \boldsymbol{\mu}_*)\right) , \tag{2.15}$$

with $\mathbf{K}_\mathbf{y} = \mathbf{K}_{\mathbf{xx}} + \sigma_n^2 \mathbf{I}$. In logarithmic space, we get

$$\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) \quad = \quad -\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu}_*)^\top {\mathbf{K}_\mathbf{y}}^{-1} (\mathbf{y} - \boldsymbol{\mu}_*) - \frac{1}{2}\log|\mathbf{K}_\mathbf{y}| - \frac{n}{2}\log(2\pi) \,. \tag{2.16}$$

In the remainder of the thesis, unless noted otherwise, we learn the GP models by maximizing this function w.r.t. the hyperparameters $\boldsymbol{\theta}$ using conjugate gradient-based line searches with random restarts. This procedure is part of the Gaussian processes toolbox by Rasmussen and Williams [2006a]. Details about how the partial derivatives of the objective function can be derived are given in Chapter 11 as well as in [Rasmussen and Williams, 2006b, Ch. A.3].

Figure 2.2 visualizes two GP models fitted to a simulated data set. While the hyperparameters of both models are at local optima of the marginal data likelihood [Eq. (2.15)], the right model achieves a significantly higher value.

For an alternative way of performing learning and inference in the Gaussian process model that does not assume *fixed* hyperparameters, Neal [1999] places prior distributions on the

hyperparameters and describes an approximation of the integrals over hyperparameters based on Markov chain Monte Carlo (MCMC) sampling.

2.3 Related Approaches

Gaussian processes are related to various other machine learning techniques, either because they aim at solving the same problems or by building on similar foundations. The history of the idea behind the Gaussian process model dates back to Wiener [1964], Kolmogoroff [1941], O'Hagan [1978], and others (see also [Rasmussen and Williams, 2006b, Sec. 2.8]). In the following, we discuss the most-closely related approaches and describe their relationship to the GP model. More details about the individual methods can be found, for example, in [Hastie *et al.*, 2001], [Rasmussen and Williams, 2006b, Sec. 7.5, Ch. 6], or in [MacKay, 2003]. For a detailed and quantitative comparison of GPs with a subset of the following methods, we refer to Rasmussen [1996].

Kernel smoothing predicts function values by computing the weighted average of training targets in the local neighborhood of the query location. The weights defining the size and "fuzziness" of the neighborhood are given by a kernel function, which is typically parameterized by the distance in input space. The GP model—although it may appear similar at first sight—differs from this approach fundamentally since it considers the *joint* distribution of the target values, which is (n+1)–dimensional for a single prediction and n training points. Whereas the GP mean prediction can also be interpreted as a weighted sum of the target values, there is a notable difference to kernel smoothing in that the weight vector is not built from the kernel function directly. Rather, it is formed in a more complicated way [see Eq. (2.8)] involving, e.g., to invert the covariance matrix. As can be seen from the *equivalence kernel* of a given GP model, which was discussed above briefly, this weight vector has interesting properties, such as automatically adapting to the density of data points in the neighborhood and to the proximity of the query location to the boundary of the training set.

Linear regression is a parametric approach, which fits a linear function to the whole training set according to an error measure. If the coefficients (or "weights") of the linear function are given a prior distribution rather than being fixed, one obtains the *Bayesian linear model*, which is the natural starting point for developing the Gaussian process model from first principles [Rasmussen and Williams, 2006b, Ch. 2].

Local regression [Cleveland and Devlin, 1988] fits linear (or polynomial) models to the local neighborhood around the query location as opposed to the whole data set. As for kernel smoothing, the extent of the neighborhood is defined by a weight function. The

role of the distance metric used for defining the local neighborhoods corresponds to the covariance function in the GP model, but not in a direct way, as outlined above for the kernel smoothing approach. A recent extension of the local regression approach is *locally weighted projection regression* [Vijayakumar *et al.*, 2005b], which was specifically designed for learning in high-dimensional spaces. Note, that kernel smoothing can be seen as a special case of local regression, in which the local linear/polynomial models are constrained to be constant functions. Also note that for this reason, some authors do not make the distinction and subsume both methods under the term *kernel smoothing*.

Radial basis function (RBF) networks [Orr, 1996] basically take the weighted averaging approach of kernel smoothing, but place the local kernels (here: "basis functions") according to a separate set of center points, rather than centering them around the training points in kernel smoothing. The model thus has the number of center points and their locations as additional parameters. Example 45.4 in [MacKay, 2003] nicely shows that the (parametric) RBF model converges to a (nonparametric) Gaussian process, if one lets the number of basis functions per unit length approach infinity while reducing their widths accordingly.

Smoothing spline regression [Wahba, 1990, Schoenberg, 1964] is an interpolation method which fits a piecewise polynomial function to the data set by minimizing both function roughness and interpolation accuracy. Spline smoothing can also be used in a Bayesian setting [Biller and Fahrmeir, 1997] and, as shown for instance by MacKay [2003, Ch. 45.1], the posterior mean of the GP model can be interpreted as a spline.

Artificial neural networks (ANNs) [Haykin, 1998] can also be applied for solving non-linear regression problems. Feedforward networks are directed graphs that pass information from an input layer of nodes through one or several hidden layers to an output layer. In general, the value at each node is computed from its predecessors by computing the weighted sum of their values according to the set of edge weights followed by applying a non-linear transformation to the result. Standard ways of learning the weights in a given network have been presented, e.g., by Werbos [1994], Rumelhart and McClelland [1986], and Riedmiller and Braun [1992]. Putting ANNs in a Bayesian framework, in which the network's parameters are not fixed but given prior distributions, Neal [1996] made the interesting observation that certain ANNs with one hidden layer converge to a GP prior over functions [Rasmussen and Williams, 2006b, Sec. 7.5] as the number of hidden nodes approaches infinity.

Kriging [Krige, 1951, Matheron, 1973, Opsomer *et al.*, 1997] is a different expression for Gaussian process regression used in geostatistics and spatial statistics. The model is equivalent to Gaussian process regression as formalized in Sec. 2.2.

Chapter 3

Learning Sampling Models for Dynamic Bayesian Networks

We present an approach to learning sampling functions for dynamic Bayesian networks. Our approach can handle continuous as well as discrete state variables and it is easy to integrate, e.g., into a particle filter. We motivate and evaluate our approach on the task of online failure detection for mobile robots.

Dynamic Bayesian networks (DBNs) [Murphy, 2002] are widely used to model the dynamic behavior of artifical and natural systems. In a DBN, the variables of interest are indexed by time and related to one another using conditional probability distributions (CPDs), which in general span multiple time frames to describe the dynamics. The result is a (typically sparse) network of local dependencies for which efficient learning and inference mechanisms have been developed. A particularly flexible and often-used inference technique is the application of the so-called particle filters (PFs). They represent the state of the system by a finite set of weighted samples and perform inference by updating the sample set iteratively using the DBN structure, the conditional models, and the set of observed (thus constant) variables.

In this chapter, we present a novel way of improving the generation of state samples in particle filters. This step has to be performed frequently and it has a major influence on the efficiency and robustness of the filter. In particular, we consider data-driven sampling, that is, the generation of state hypotheses at locations that are most promising given recent sensor measurements. Our contribution is to show how sampling policies can be learned from past experience—using Gaussian process regression for continuous state variables and

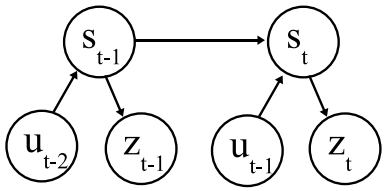


Figure 3.1: Dynamic Bayesian network for the mobile robot domain.

classification for discrete ones. Given the GP's ability to also estimate the uncertainty of its predictions, we can derive a sound way of integrating these *informed proposals* into the particle filtering framework. As a result, the estimation process is more robust since the sampling density is higher in the important parts of the state space while it is also more efficient because less samples are "wasted" in unlikely regions.

As a running example and test-bed for our approach, we consider the problem of online failure detection and recovery for a mobile robot. Concretely, we face the task of (a) localizing an autonomous, wheeled robot relative to a known environment, (b) to detect collisions with (unseen, movable) obstacles online and (c) to estimate their physical parameters (e.g., location and weight) such that they can be dealt with safely. Our system does not require additional hardware such as inertial sensors or bumper rings, which may be expensive or otherwise unfavorable in certain domains, e.g., for aerial blimps. The proposed algorithm has been implemented on a real robot and can be executed online. We discuss results from several test runs in realistic scenarios in which we compared the achievable detection rates and pose tracking quality of our system with an optimized standard particle filter implementation without informed proposals.

In the following, we first formalize the sequential state estimation problem in Sec. 3.1 and describe how to solve it using particle filters. We then introduce the concept of Gaussian process proposals and focus specifically on the mobile robot localization and collision detection problem outlined above. In Sec. 3.3, we present experimental results and discuss related work in Sec. 3.4.

3.1 Sequential State Estimation

One of the fundamental problems in robotics and engineering is to estimate the state of a system given a sequence of acquired sensor measurements and action commands that have been executed. In order to perform this task sequentially, i.e., in an iterative way whenever new information is available, the system is typically modeled using a dynamic Bayesian network (DBN). Assuming a discretization of time—either by modeling iterations of equal duration or by taking the finite set of measurement and action events—there is a

finite set of time-indexed variables that has to be modeled. The DBN factorizes the joint posterior over these variables to a (typically sparse) network of dependent variables and their corresponding conditional probability distributions. Here, these 'local models' can be limited to a specific time frame or they can span several to model the system's behavior over time.

Mobile Robot Localization

In the case of mobile robot localization, for instance, the (dynamic) state of the system is the 3D pose $s_t = (x_t, y_t, \theta_t)$ of the robot relative to its environment. Additional variables include the sensor measurements $z_{1:t}$ as well as the action commands $u_{0:t-1}$ that have been issued—w.l.o.g. assuming time-synchronous state, action, and measurement variables for ease of notation. A typical DBN for such a system is composed of a *prior* distribution over poses $p(s_0)$, a *transition model* $p(s_{t+1}|s_t, u_t)$ and an *observation model* $p(z_t|s_t)$. Figure 3.1 shows how these CPDs are connected to yield the typical DBN for state estimation problems. In models defined by such a graph, each variable is independent from all others given its direct predecessors (see [Murphy, 2002] for a general discussion of graphical models). Given the modeled independence assumptions, the state posterior can be formulated recursively as

$$\begin{aligned} p(s_t \mid z_{1:t}, u_{0:t-1}) &= \int p(s_t \mid s_{t-1}, z_t, u_{t-1})\, p(s_{t-1} \mid z_{0:t-1}, u_{0:t-2})\, ds_{t-1} \\ &= \eta_t \cdot \underbrace{p(z_t \mid s_t)}_{\text{obs. model}} \cdot \int \underbrace{p(s_t \mid s_{t-1}, u_{t-1})}_{\text{transition model}} \underbrace{p(s_{t-1} \mid z_{0:t-1}, u_{0:t-2})}_{\text{recursive term}}\, ds_{t-1} \end{aligned} \tag{3.1}$$

where η_t is a normalizing factor, which is equal for all s_t and, thus, has little importance here. In order to achieve a tractable approximation of Eq. (3.1), one has to make additional assumptions about the distributions involved. Important classes of such approximations include

Discrete filters (DF) where the state space is discretized—typically using a regular grid—which turns the integral in Eq. (3.1) into a sum over possible predecessors,

Kalman filters (KF) for which the distributions over states $p(s_t|\ldots)$ and the observation model are assumed Gaussian and the transition model is linear, and

Particle filters (PF) which represent the distributions over states by a weighted set of state hypotheses which are manipulated individually according to the transition and observation models.

Particle filters are flexible in terms of distributions that can be represented, they are easy to implement, and they can convert heuristics into provably correct algorithms through the concept of *proposal distributions* (see Chapter 5 in [Murphy, 2002]).

3.1.1 Particle Filters for Bayesian Filtering

Particle filters seek to approximate the integral in Eq. (3.1) using Monte Carlo integration. Concretely, they represent distributions over states $p(s_t|\ldots)$ by sets of weighted samples $\mathcal{X}_t = \{\langle s_t^{[i]}, w_t^{[i]}\rangle\}_{i=1}^n$. Starting from a prior distribution $\mathcal{X}_0 \sim p(s_0)$, the particle set is updated sequentially by executing the following steps in each iteration:

1. *Sampling*: The next generation of particles $\{s_t^{[i]}\}$ is obtained from the generation $\{s_{t-1}^{[i]}\}$ by sampling from a proposal distribution π.

2. *Importance Weighting*: Importance weights $w_t^{[i]}$ are assigned to the individual particles according to

$$w_t^{[i]} = \frac{\text{target}}{\text{proposal}} = \frac{p(s_{1:t}^{[i]} \mid z_{1:t}, u_{1:t})}{\pi(s_{1:t}^{[i]} \mid z_{1:t}, u_{1:t})} \propto \frac{p(z_t \mid s_t^{[i]}) p(s_t^{[i]} \mid s_{t-1}^{[i]}, u_t)}{\pi(s_t \mid s_{1:t-1}^{[i]}, z_{1:t}, u_{1:t})} \cdot w_{t-1}^{[i]} \qquad (3.2)$$

   The weights account for the fact that the proposal distribution π is in general not equal to the target distribution of successor states. It can be shown that for all proposal functions π with $p(x) > 0 \Rightarrow \pi(x) > 0$ and the weight update (3.2), the PF algorithm is guaranteed to approximate the correct posterior for $n \to \infty$ [Doucet *et al.*, 2001].

3. *Resampling*: Particles are drawn with replacement proportional to their importance weight.

In the special case of $\pi \equiv p(s_t|s_{t-1}, u_t)$, that is, if sampling is performed according to the transition model, the relationship between Eq. (3.1) and the filtering procedure can be seen directly. Therefore, the recursive weight update (3.2) simplifies to a multiplication of all particle weights with the respective observation likelihoods $p(z_t|s_t^{[i]})$. For a formal derivation of the particle filtering principle, see for example [Thrun *et al.*, 2005] or [Doucet *et al.*, 2001]. Note that resampling exchanges *likelihoods* by *frequencies* without altering the represented distribution. In the limit of infinitively many and densely-sampled particles, the representation of a distribution by non-constant importance weights is equivalent to the one where all weights are equal, but the particle density varies accordingly. Thus, the purpose of the resampling step is to distribute more particles to the probable areas of the state space and less to the improbable ones.

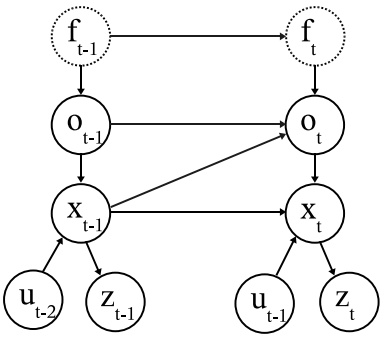


Figure 3.2: DBN for the mobile robot domain extended to include a discrete failure mode f_t and the failure-dependent continuous variables o_t.

The robustness and efficiency of particle filters strongly depends on the proposal distribution π that is used to sample the new state hypotheses in the sampling step. If the proposal distribution differs too much from the true posterior, there is a high risk of filter divergence. On the other hand, if π is too flat (e.g., uniform in the extreme case) many particles are drawn in areas of low likelihood and, thus, computational resources are wasted.

In the following section, we extend the mobile robot DBN to include a discrete failure mode variable and additional continuous variables for the failure's parameters. As this increases the dimensionality of the state space and also adds highly unlikely transitions (failures occur only rarely), the transition model becomes a sub-optimal choice as proposal distribution. For this reason, we then introduce an *informed proposal* distribution, which utilizes information about the failure mode contained in the most recent sensor measurement to yield better pose estimates.

3.1.2 Modeling Failures using Hybrid DBNs

In realistic environments, there typically exist external influences that can cause drastic changes to the behavior of a dynamic system. Consider for example a mobile robot colliding with an undetected obstacle while executing a motion command. Such situations can be modeled by extending the DBN by a discrete failure mode variable f_t and additionally continuous failure-dependent state variables o_t (see Fig. 3.2). We assume the commonly used constant failure rate model (see [Ng *et al.*, 2005]), in which failure events do not depend on the other state variables, but rather occur according to an individually specified prior distribution. Since the observations are independent of the failure state (f_t, o_t) given the state s_t of the robot, the observation model remains $p(z_t \mid s_t)$. The state transition

model can be factorized as

$$p(s_t, f_t, o_t \mid s_{t-1}, f_{t-1}, o_{t-1}, u_{t-1}) = \underbrace{p(f_t \mid f_{t-1})}_{\text{failure event model}} \cdot \underbrace{p(o_t \mid f_t, s_{t-1}, o_{t-1})}_{\text{failure parameter model}} \cdot \underbrace{p(s_t \mid o_t, s_{t-1}, u_{t-1})}_{\text{transition model}} . \tag{3.3}$$

The constant failure rate model states that failure events are distributed exponentially depending on a failure rate parameter λ, i.e.,

$$p(f_t) = 1 - e^{-\lambda \cdot (t - \tilde{t})} , \tag{3.4}$$

where $\tilde{t}$ denotes the time of the last failure event. For such a model, the mean time between failures becomes $MTBF = \frac{1}{\lambda}$. For realistic failure rates λ, this model results in extremely low failure probabilities per filter iteration. Assume, for example, a mean time of 30 minutes between collisions of a service robot with unseen obstacles. This implies $\lambda = \frac{1}{1\,800s} = 0.000\,\overline{5}$ and with a filter frequency of $\delta t = 0.1$ seconds yields a failure probability of $p(f_t \mid \neg f_{t-1}) \approx 0.000\,056$ within one iteration. For such a small value, just one of 20 000 particles would be sampled to a failure mode on average, if the transition model was used directly as proposal distribution for this variable. Thus, one would either need an extremely large particle set or would risk that failures remain undetected. This problem is amplified by the fact that not only the discrete failure mode has to be sampled, but also the unknown continuous failure parameters. Since in general, there is no prior knowledge about the parameters of randomly occurring failures, we assume a uniform distribution

$$p(o_t \mid f_t, x_{t-1}, o_{t-1}) = \mathcal{U}_{[o_{\min}, o_{\max}]}(o_t) \tag{3.5}$$

over a certain interval. Note that this model applies only to the case where the system transitions into a failure state. The evolution of failure parameters within a failure state is typically governed by a much more peaked distribution similar to the motion model of the robot. In Sec. 3.3, we describe our model for the evolution of collision parameters based on rigid body dynamics. This model is able to track collisions sufficiently accurate, if the initial collision parameters have been estimated well enough. The goal here is to improve the detection of failure events and to estimate the initial parameters efficiently.

3.2 Gaussian Processes Proposals

To address the problem of low sampling probabilities for important parts of the state space, Thrun *et al.* [2001] introduced the *risk sensitive particle filter* that incorporates a learned risk function to force the filter into less likely but important states. While this approach

ensures a reasonable amount of samples in the important failure modes, it cannot adapt to the specific situation the robot is in when the sampling decision is made. In contrast, we propose to use *learned* proposal distributions that provide informed guesses about what the states with higher posterior probabilities are.

Data-driven Proposal Distributions

In sequential importance sampling [Doucet, 1998], an arbitrary proposal distribution can be used to sample the relevant areas of the state space as long as (a) all possible states have a non-zero possibility of being chosen and (b) the importance weights of the particles are adjusted appropriately. Proposal distributions that depend on the most recent sensor measurements or on features extracted by separate algorithms are typically denoted as *data-driven proposals* or *detector-mediated proposals* [Khan *et al.*, 2004]. Such proposals aim at approximating the optimal proposal $p(s_t \mid s_{t-1}, z_t, u_{t-1})$ which includes the most current sensor measurement z_t. It can be shown [Murphy, 2002] that such a *fully informed proposal* minimizes the variance of the importance weights making it the optimal choice.

Weight correction [see Eq. (3.2)] is easy to implement also in the failure detection model, if one assumes a proposal π that factorizes according to the transition model (3.3). In this case, the (corrected) weight of a particle i at time t becomes

$$\begin{aligned}
w_t^{[i]} &= \frac{p(s_{1:t}^{[i]} \mid z_{1:t})}{\pi(s_{1:t}^{[i]} \mid z_{1:t})} = \frac{p(z_t \mid s_{1:t}^{[i]}, z_{1:t-1})\; p(s_{1:t}^{[i]} \mid z_{1:t-1})}{\underbrace{p(z_t \mid z_{1:t-1})}_{=:1/\eta}\; \pi(s_{1:t}^{[i]} \mid z_{1:t})} \\
&= \eta \cdot \frac{p(z_t \mid s_t^{[i]})\; p(s_t^{[i]} \mid s_{t-1}^{[i]})}{\pi(s_t^{[i]} \mid s_{1:t-1}^{[i]}, z_{1:t})} \cdot \underbrace{\frac{p(s_{1:t-1}^{[i]} \mid z_{1:t-1})}{\pi(s_{1:t-1}^{[i]} \mid z_{1:t-1})}}_{=w_{t-1}^{[i]}} \\
&= \eta \cdot w_{t-1}^{[i]} \cdot p(z_t \mid s_t^{[i]}) \cdot \frac{p(x_t^{[i]} \mid o_t^{[i]}, x_{t-1}^{[i]})}{\pi(x_t^{[i]} \mid s_{1:t-1}^{[i]}, z_{1:t}^{[i]})} \cdot \\
&\quad \frac{p(f_t^{[i]} \mid f_{t-1}^{[i]})}{\pi_f(f_t^{[i]} \mid s_{1:t-1}^{[i]}, z_{1:t}^{[i]})} \cdot \frac{p(o_t^{[i]} \mid f_t^{[i]}, x_{t-1}^{[i]}, o_{t-1}^{[i]})}{\pi_o(o_t^{[i]} \mid s_{1:t-1}^{[i]}, z_{1:t}^{[i]})} \; .
\end{aligned} \tag{3.6}$$

Here, we left out the control variables u for the sake of brevity. The normalizing factor η is constant for all particles i and, thus, can be neglected. π_f denotes the proposal for the failure event and π_o the one for the failure parameters. Visually speaking, Eq. (3.6) states, that after each filter iteration, the particle weights have to be multiplied with the current observation likelihood $p(z_t \mid s_t^{[i]})$ and with two correction terms for the two learned proposal distributions π_f and π_o. To calculate these correction terms for a specific sample $s_t^{[i]}$, we

follow the importance sampling principle and divide the likelihoods defined in Eq. (3.4) and (3.5) by the likelihoods according to which the state variables $f_t^{[i]}$ and $o_t^{[i]}$ have been drawn from π_f and π_o.

In our approach, we propose to learn π_f and π_o from data to accommodate for the fact that z_t might carry useful information about the current f_t and o_t despite of their extremely low a priori likelihoods. Formally, the task is to learn a mapping from a feature vector F_t extracted from $\langle s_{1:t-1}^{[i]}, z_{1:t}^{[i]} \rangle$ to the variables f_t and o_t. In theory, any feature vector F_t as well as any learned models π_f and π_o can be used as a proposal as long as the assumptions

- $\pi_f(f_t \mid F_t) \neq 0$ for all f with $p(f \mid s_{1:t}, z_{1:t-1}) \neq 0$
- $\pi_o(o_t \mid f_t, F_t) \neq 0$ for all o with $p(o \mid s_{1:t}, z_{1:t-1}) \neq 0$.

hold, which means that all possible failure states are assigned a non-zero probability of being chosen. Another precondition for the learned proposals therefore is that a full predictive distribution for proposed states is available which can be sampled from.

Gaussian process predictions, being normal distributions, naturally meet these requirements for proposal distributions: They can be sampled from directly, they supply the likelihoods of sampled values, they have an infinite support, and they therefore assign non-zero likelihoods to all possible states.

3.2.1 Learning Collision-Proposals from Data

We place Gaussian process priors on the functions π_f and π_o that we seek to learn from data and use the feature vectors F_t (explained in the following) as the corresponding indices. In principle, we could use all available filter information as the index feature: $F_t = \langle s_{1:t-1}^{[i]}, z_{1:t}^{[i]} \rangle$. In practice, however, learning is more efficient, if we extract lower-dimensional features F_t from $\langle s_{1:t-1}^{[i]}, z_{1:t}^{[i]} \rangle$ so that there is a high correlation between F_t and the variables to be predicted, i.e., f_t and o_t.

In our test scenario, we consider a mobile robot navigating in an office environment that eventually collides with obstacles, that are not detectable by its sensors directly (e.g., a low doorstep or a smaller object on the ground). Here, f_t denotes the (binary) collision event variable and o_t is a real-valued variable representing the point of impact on the front of the robot. We found that a suitable feature vector F_t for learning proposals for these hidden failure states is given by the discrepancy $(\Delta v_t, \Delta v_r)$ between the current motion estimate in s_t (velocity and heading) w.r.t. a purely sensor-driven estimate of these quantities from z_t. The sensor-driven estimate of velocity and heading, for instance, can be obtained using an inertial sensor or a laser scan-matcher. We have used the latter in our experiments. Intuitively, $F_t := (\Delta v_t, \Delta v_r)$ captures the amount of translational and rotational deviation

Figure 3.3: Left: The system learns the relationship between failure variables and sensor measurements in simulation. Right: The system implemented on a real robot utilizes this knowledge to detect collisions with obstacles that remain undetected by its sensors.

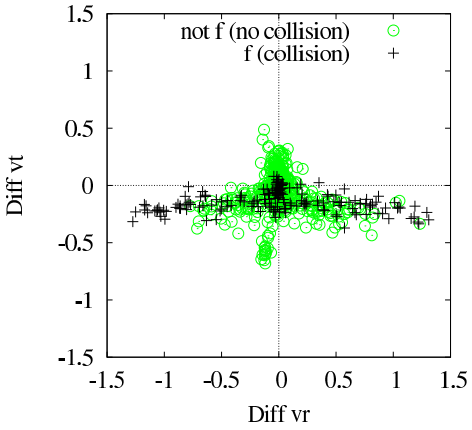
of the robots' belief about its motion state from the measures one. It can be expected that this feature carries information about whether a collision with an undetected obstacle has occurred or not.

For gathering training data, we simulated 500 automatically labeled trajectories of a mobile robot colliding with different, randomly placed obstacles using the 3D, high-fidelity simulator Gazebo [Koenig and Howard, 2004]. The left image in Fig. 3.3 depicts a typical randomly generated situation during training. Given this training set, we learned the Gaussian process models for the functions $F_t \mapsto f_t$ and $F_t \mapsto o_t$. Since f_t is a binary variable rather than a real-valued one, the learning and inference mechanisms described in Sec. 2.2 cannot be applied directly. Rather, such classification problems can be solved in the GP framework by including for every binary target t_i a real-valued latent variable l_i, such that

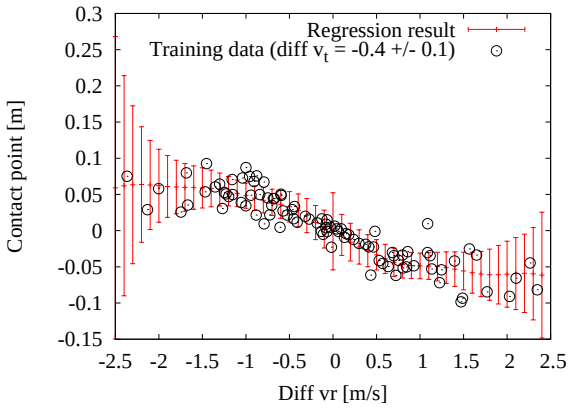
$$p(t_i = 1) = \frac{1}{1 + e^{-l_i}}\,. \tag{3.7}$$

This is known as the logistic model, which links class probabilities to real values, see [Neal, 1997]. The latent variables can now be given a Gaussian process prior as in the regression setting and predictions of class probabilities can be performed by predicting the corresponding latent variables and evaluating Eq. (3.7) (see [Rasmussen and Williams, 2006b, Ch. 3] for a detailed discussion of Gaussian process classification).

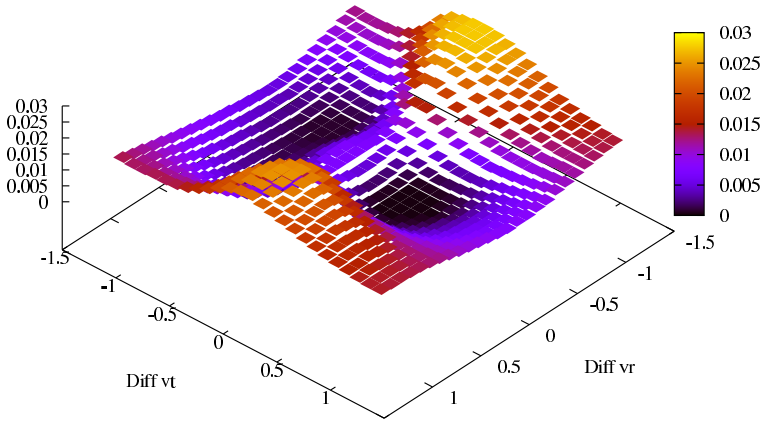
For learning the classification model for f_t as well as the regression model for o_t, we used the MCMC-based implementation of Gaussian process regression and classification of Neal [1999] and employed the squared exponential covariance function k_{SE} (see Chapter 2). Neal's toolbox implements the fully Bayesian approach, in which the hyperparam-



(a) Failure classification data set



(b) Failure parameter data set and regression result



(c) Learned class probabilities for the failure classification task

Figure 3.4: The data sets and learning results for the failure models. Plot (a) shows the collision event training set, where two feature values are mapped to class instances. Plot (c) shows the the learned class probabilities $\pi_f(F_t)$ using Gaussian process classification and (b) plots the learned regression model for the collision parameters visualized by a cut through the 2D regression function that maps velocity deviations to contact points for a collision.

eters are not fixed but integrated over according to prior distributions assigned to them. Other implementations, like the `Matlab`-based one by Rasmussen and Williams [2006b], achieved comparable results but were harder to integrate into our `C/C++`-based implementation of particle filtering for state estimation.

Figure 3.4(a) depicts the gathered data points for the (binary) failure event variable f_t. As can be seen from the diagram, Δv_t is negative for nearly all "collision" data points, which corresponds to the fact that the robot is decelerated when a collision occurs. The data points for "no collision" are spread widely and do not separate well from the "collision" data points due to noisy sensor measurements and imperfect automatic labeling of the collision events. This makes the classification problem a hard one. Figure 3.4(c) shows

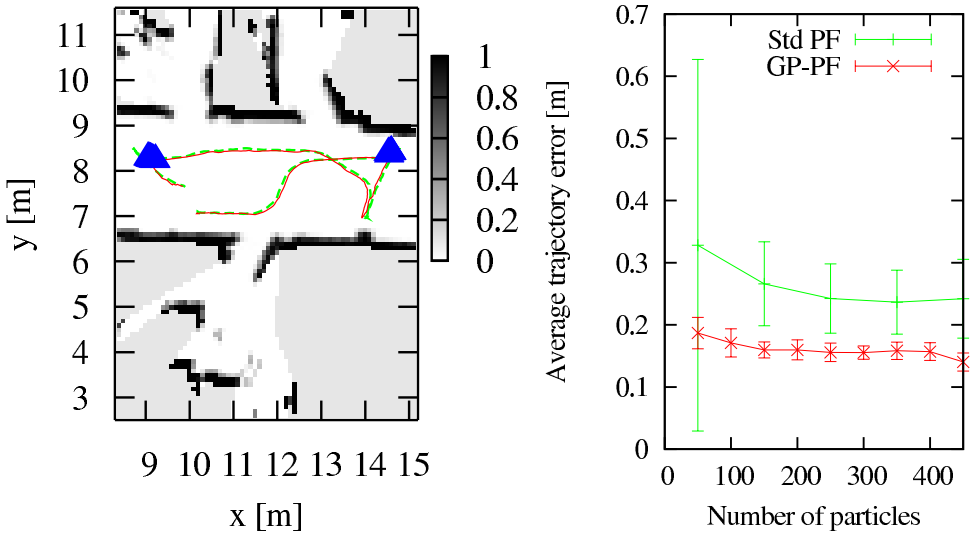
the learned class probabilities depending on the two velocity differences described above. It should be stressed, that this classifier is used as a proposal distribution for collisions in the particle filter rather than as a collision detector directly. It can be seen clearly from, e.g., Fig. 3.4(a) that the available features are too ambiguous to allow for acceptable *instantaneous* classification. Experiments with a real robot (see Sec. 3.3) showed that the sequential state estimation-based approach, in contrast, yields high detection rates with a low number of false alarms.

Given a collision event, the continuous collision parameters o_t are estimated to simulate the effects of the failure on the system and to continue the tracking process. Since the task is not to fully track the pushed obstacle over time, a simple model that abstracts from the obstacle's geometry and exact pose has proven sufficient to describe the effects on the robot. A collision with an unseen obstacle is represented by the obstacle mass m and the contact point c_t on the robot's front. Therefore, the collision parameters are $o_t = (m, c_t)$ in the considered setting. We learn the proposal distribution $\pi_o(o_t|F_t)$ for the parameters o_t using the same velocity-based features and simulated training set as described above and the Gaussian process regression technique. Figure 3.4(b) depicts a cut through the learned 2D distribution for the collision parameter c_t which is the contact point of the obstacle on the front of the robot. The point of contact is measured in meters from the center of the robot's front to the right. It can be seen from the diagram that unexpected clockwise rotations ($\Delta v_r < 0$) of the robot are mapped to positive values for the contact point, which corresponds to a collision on the righthand side of the robot.

3.3 Experimental Results

Our system has been implemented on an ActivMedia Pioneer 3DX robot and tested extensively in an office environment. The right photo in Fig. 3.3 illustrates a typical situation. Before presenting experimental results, we describe the motion model we implemented for our localization system. Since this model is not the focus of this work, we give only a brief overview here.

The most widely used motion model for mobile robots is based on the wheel encoder measurements (see [Thrun *et al.*, 2005]). These, rather than the the actual control commands, are taken as control input u_{t-1}, which under normal circumstances result in comparably accurate predictions of the performed movement. Under the influence of failures like collisions or wheel slip, however, the motion of the wheels is not consistent with the robot's motion any more. A more appropriate model for such situations, that still is efficient enough to be evaluated online, is based on rigid body dynamics (see [Plagemann *et al.*, 2006]). We model the robot as a rigid body in the 2D plane, represented by a set of con-



(a) Two correctly detected collisions (triangles), the estimated trajectory (solid line), and the ground truth (dotted).

(b) Average deviation of the estimated trajectory from the ground truth in meters for a varying number of particles.

Figure 3.5: Results of our experimental evaluation with a real robot colliding with undetected obstacles.

stant values and the variable state vector $x_t = (pos_x, pos_y, pos_\theta, vel_t, vel_r)$ which includes the translational velocity vel_t and the rotational velocity vel_r. In each filter iteration, the wheel thrusts are calculated from the actual velocity command that was sent to the motors. From this, the next state vector is computed by numerical simulation using the physical relationships between forces, acceleration, and speed. Due to space limitations, we refer to Witkin and Baraff [1997] for details about rigid body physics. With this model, collisions with another rigid object at a given point of contact can be simulated using the same type of physical abstraction, namely computing the impulse, the resulting forces, and ultimately the influence on the robot's state vector. At the same time, this model describes how the point of contact between the robot and the obstacle changes over time and therefore defines the transition model for failure parameters of Eq. (3.3). From our experience, this physical model achieves the best balance between accuracy and efficiency. Simpler models fail to handle important test cases while more complex models have too many free parameters to be evaluated in real time.

Quantitative Evaluation of Failure Detection and Tracking Performance

To evaluate the usefulness of our approach quantitatively, we compared it to a particle filter that implements the same process model with the standard uninformed proposals described in Sec. 3.1.2. The parameters of the standard filter were optimized for best tracking

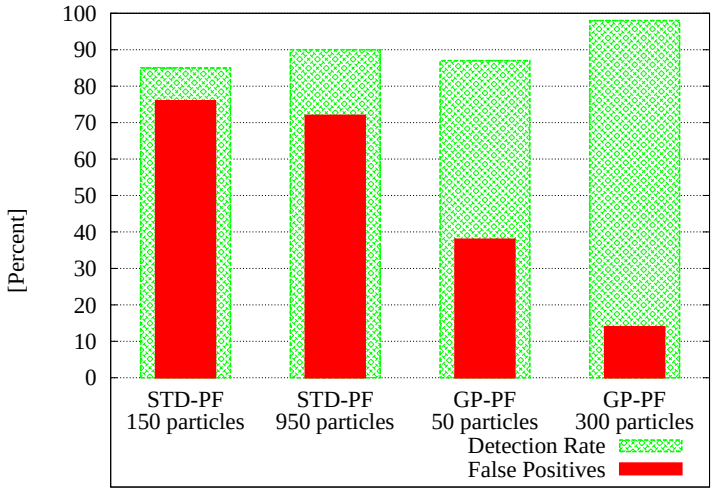


Figure 3.6: Detection results with the optimized standard particle filter (STD-PF) and our approach that uses Gaussian process proposals (GP-PF).

performance and failure detection rates to ensure comparability. We recorded data by manually steering the robot through the environment and arranged for two collisions, one with milk cartons and the other one with a case of lemonade bottles. Both obstacles were placed at arbitrary positions and the obstacles' height were too low for the laser sensor to detect them. Figure 3.5(a) depicts a typical test run where the system successfully tracked the pose of the robot and detected the two collisions. We tested our improved particle filter with Gaussian process proposals on the recorded data set and term this approach GP-PF in the following. As a benchmark, also the standard particle filter (Std PF) was evaluated for different parameter settings. Each filter was executed 50 times for each parameter setting.

Figure 3.6 gives the failure detection performance of the different filters. The detection rate is defined as the number of correctly identified failures relative to the full number. The false positives rate is the amount of false alarms relative to the number of detections. The ground truth collision events were manually entered and a collision was counted as correctly detected, when the marginal failure likelihood exceeded a threshold Θ after a maximum of six filter iterations (0.6 seconds) after the true failure event. The threshold Θ was optimized independently for each filter to ensure unbiased comparison.

The diagram in Fig. 3.5(b) gives the average deviation of the tracked poses of the robot compared to the ground truth trajectory. The ground truth trajectory was computed using a scan matcher. The results visualized in this diagram show that our system stays around ten centimeters closer to the true trajectory and produces less variance in these estimates

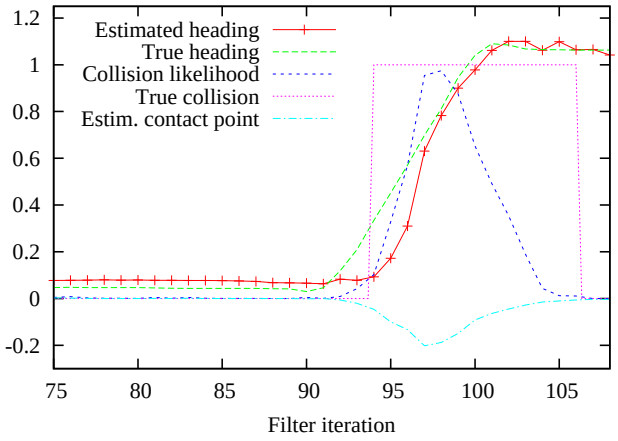


Figure 3.7: Measurements and filter estimates during a typical collision event. Between iteration 90 and 95, the robot hits an undetected obstacle.

than the standard approach. This is mainly due to the fact that the failure parameter (here, the point of contact with the obstacle) is estimated more accurately. To give an impression about the accuracy with which the system estimates the point of contact and the path of the robot, one failure event is depicted in detail in the diagram in Fig. 3.7. It can be seen, that the estimated failure likelihood increases shortly after the labeled failure event and that the heading angle of the robot is estimated correctly.

The detection rates as well as the tracking results show that learned Gaussian process proposals can indeed increase the reliability and efficiency of online state estimation approaches. The time requirements for the improved particle filter are around 10% to 15% higher than for the standard implementation without Gaussian process proposals. Nevertheless, the implemented system with 200 particles is able to process one minute of recorded data in less than 23 seconds on a PC with a 2800 MHz CPU.

3.4 Related Work

The term "fault detection" is commonly referred to as the detection of an abnormal condition that may prevent a functional unit from performing a required function [Leveson, 1995]. Most works in the fault detection and isolation literature deal with internal faults such as defects in hardware or software. Model-based diagnosis has been approached from within the AI community using symbolic reasoning with a focus on large systems with many interacting components and from the control theory community concentrating on fewer components with complex dynamics and higher noise levels [Krysander, 2003]. Krysander proposed a hybrid model consisting of discrete fault modes that switch between differential equations to describe system behavior. The diagnosis system is designed for

large systems with low noise levels, where instantaneous statistical tests are sufficient to identify a faulty component.

The close coupling between a mobile system and its environment makes it hard to detect abnormal behavior only using instantaneous statistical tests without tracking possible failure modes over time [Dearden and Clancy, 2002]. For this reason, probabilistic state tracking techniques have been applied to this problem. Particle filters represent the belief about the state of the system by a set of state samples [Thrun *et al.*, 2005]. In particle filter based approaches to fault diagnosis, the system is typically modeled by a non-linear Markov jump process [Driessen and Boers, 2004] or a dynamic mixture of linear processes [de Freitas *et al.*, 2003]. The look-ahead particle filter introduced in the latter work also approximates the optimal proposal distribution by considering the most recent sensor measurements, but, in contrast to our work, focuses on the case of discrete failure modes without continuous failure parameters. Benazera *et al.* [2004] combine consistency-based approaches, i.e., the Livingstone system, with particle filter based state estimation techniques. Verma *et al.* [2003] introduce the variable resolution particle filter for failure detection. Their approach is to build an abstraction hierarchy of system models. The models of consideration build a partition of the complete state space and the hierarchy is defined in terms of behavioral similarity. A different kind of model abstraction hierarchy based on explicit model assumptions was developed in [Plagemann *et al.*, 2006] to detect failures online.

Tresp [2000] deal with learning the conditional probability distributions (CPDs) in Bayesian networks using Gaussian process mixtures. They do not deal with issues concerning dynamic systems, nonparametric state estimation, and the role of proposal distributions. Approaches that deal with the time efficiency of particle filters include [Kwok *et al.*, 2002] in which real-time constraints are considered for single system models or techniques in which a Rao-Blackwellized particle filter is used to coordinate multiple models for tracking moving objects [Kwok and Fox, 2004].

3.5 Conclusion

In this chapter, we showed that efficient proposal distributions for particle filters can be learned using Gaussian process models and that both discrete as well as continuous state variables can be treated in a consistent manner. We applied the approach to the problem of online failure detection for mobile robots and presented a system for detecting unforseen collisions. Experiments with a real robot demonstrated, that the developed system is able to track the robot's state more reliably through collision events than an optimized version of the standard particle filter with uninformed proposals. Our system does not require any

additional hardware and can be trained conveniently using a simulator.

Furthermore, the approach was presented in a generic way, describing how to learn sampling functions for variables in dynamic Bayesian networks. It is expected, that the general approach can be beneficial also in other applications involving sampling-based estimation.

Chapter 4

Body Scheme Learning for Manipulation

We present an approach to learning the body scheme of a robotic manipulator arm from scratch using self-observations with a single monocular camera. We introduce a flexible model based on Bayesian networks that allows to simultaneously find the robot's kinematic structure and to learn the underlying conditional density functions. Here, Gaussian process regression is used for learning the local kinematic transformations from data.

Kinematic models are widely used in practice, especially in the context of robotic manipulation [Craig, 1989, Choset *et al.*, 2005]. These models are typically derived analytically by an engineer [Rosales and Gan, 2004] and rely heavily on prior knowledge about the robot's geometry and kinematic parameters. As robotic systems become more complex and versatile or are even delivered in a completely reconfigurable way, there is a growing demand for techniques allowing a robot to automatically learn body schemes with no or only minimal human intervention. Clearly, such a capability would not only facilitate the deployment and calibration of new robotic systems but also allow for autonomous re-adaptation when the body scheme changes, e.g., due to deformations of robot parts or due to material fatigue. Additionally, to make proper use of tools, a robot should be able to incorporate the tool into its own body scheme and to adapt the gained knowledge in situations in which the tool is grabbed differently. Finally, components of the robot might get exchanged or replaced by newer parts such that the overall system model no longer complies with original engineering.

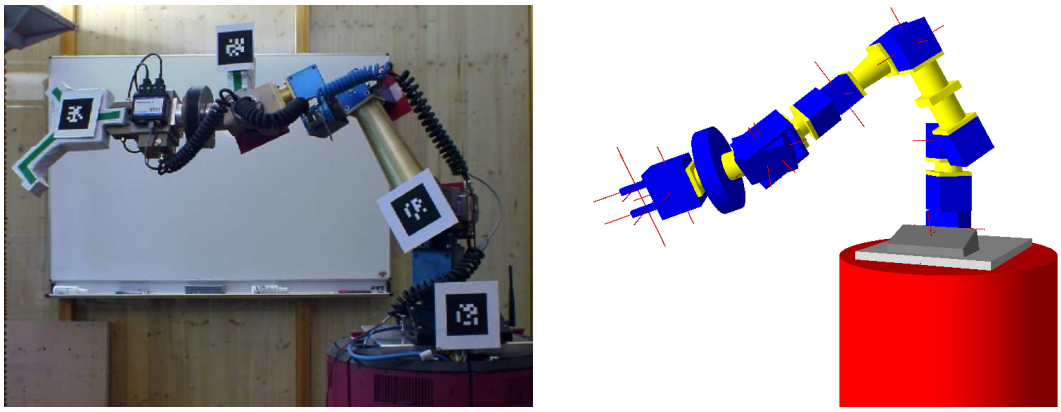


Figure 4.1: Left: Our 6-DOF robotic manipulator arm learns and monitors its own body scheme using an external monocular camera and visual markers. Right: A similar, simulated 7-DOF-manipulator consisting of 10 body parts..

Neuro-physiological evidence indicates that humans as well as higher primates learn and adapt their internal models continuously and autonomously using self-perception [Meltzoff and Moore, 1997]. Brain scan studies of monkeys that have been trained to use tools revealed that the tool itself even gets integrated into their body schemes over time [Maravita and Iriki, 2004]. Mirror neurons, as found in brain area F5, map proprioceptive sensations to tactile and visual ones and thereby seem to serve as a neurological representation of the *body scheme* [Holmes and Spence, 2004]. Moreover, they seem to translate external visual stimuli, for example from a demonstrator, into proprioceptive ones, and thereby play an important role in imitation and imitation learning.

In this chapter, we investigate how to equip robots with the ability to learn and adapt their own body schemes and kinematic models using exploratory actions and self-perception only. Figure 4.1 shows the experimental setups used for developing our approach. The left image depicts the real demonstrator system; the right image shows a visualization of the simulated counter-part. We propose to learn a Bayesian network for the robot's kinematic structure including the forward and inverse models relating action commands and body pose. More precisely, we start with a fully connected network containing all perceivable body parts and available action signals, perform random "motor babbling," and iteratively reduce the network complexity by analyzing the perceived body motion. At the same time, we learn Gaussian process regression models for all individual dependencies in the network, which can later be used to predict the body pose when no perception is available or to allow for gradient-based posture control.

Our approach addresses all of the following practical problems that frequently arise in robotic manipulation tasks in a single framework:

- **Prediction:** If both the structure and the conditional probability distributions (CPDs) of the Bayesian network are known, the robot should be able to predict the expected resulting body configuration for a given action command.
- **Control:** Conversely, given a target body pose, the robot should be able to generate appropriate action commands that will lead to this pose.
- **Model testing:** Given both a prediction and an observation of the current body pose, the robot should be able to estimate the accuracy of its own pose predictions. Model accuracy can, for example, be defined in terms of the prediction error in scene coordinates or by considering the data likelihood.
- **Learning:** Given a sequence of action signals and the corresponding body postures, the Bayesian network and its parameters should be learned from the data.
- **Discovering the network structure:** When the structure of the Bayesian network is unknown, the robot should be able to build it from the available local models which explain the observed data best.
- **Failure detection and model adaptation:** When the robot's physiology changes, e.g., when a joint gets blocked or is deformed, or a visual marker is changed, this should be detected efficiently so that only the affected local parts of the Bayesian network need to be replaced.

We will now introduce our Bayesian framework for representing body schemes, then discuss failure awareness and life-long adaptation in Sec. 4.2, and present experimental results obtained with real and simulated manipulator arms, which demonstrate that our approach is able to quickly learn compact and accurate models and to robustly deal with noisy observations. We finish the chapter with a discussion of related work and conclusions.

4.1 A Bayesian Framework for Kinematic Chains

A robotic body scheme describes the relationship between available action signals $\langle a_1, \ldots, a_m \rangle$, self-observations $\langle \mathbf{Y}_1, \ldots, \mathbf{Y}_n \rangle$, and the configurations of the robot's body parts $\langle \mathbf{X}_1, \ldots, \mathbf{X}_n \rangle$. In our concrete scenario, in which we consider the body scheme of a robotic manipulator arm in conjunction with a stationary, monocular camera, the action signals $a_i \in \mathbb{R}$ are real-valued variables corresponding to the joint angles. Whereas the $\mathbf{X}_i \in \mathbb{R}^6$ encode the 6-dimensional poses (3D Cartesian position and 3D orientation) of the body parts w.r.t. a reference coordinate frame, the $\mathbf{Y}_i \in \mathbb{R}^6$ are observations of the

body parts—generally noisy and potentially missing. Throughout this chapter, we use capital, bold letters to denote the pose variables to highlight that these also uniquely define homogeneous transformation matrices, which can be concatenated and inverted. Note that we do not assume direct feedback/proprioception telling the robot how well joint i has approached the requested target angle a_i. Formally, we seek to learn the probability distribution

$$p(\mathbf{X}_1, \ldots, \mathbf{X}_n, \mathbf{Y}_1, \ldots, \mathbf{Y}_n \mid a_1, \ldots, a_m)\,, \tag{4.1}$$

which in this form is intractable for all but the simplest scenarios. It is therefore typically assumed that each observation variable $\mathbf{Y}_i$ is independent from all other variables given the true configuration $\mathbf{X}_i$ of the corresponding body part and that they can thus be fully characterized by an observation model $p(\mathbf{Y}_i \mid \mathbf{X}_i)$. Furthermore, if the kinematic structure of the robot was known, a large number of pair-wise independencies between body parts and action signals could be assumed, which in turn would lead to the much simpler, factorized model

$$p(\mathbf{X}_1, \ldots, \mathbf{X}_n \mid a_1, \ldots, a_m) = \prod_i p(\mathbf{X}_i \mid \text{parents}(\mathbf{X}_i))\,. \tag{4.2}$$

Here, parents($\mathbf{X}_i$) comprises all body parts and action variables that directly influence $\mathbf{X}_i$. Note, that the actions are given and, thus, do not depend on other variables in this model. We now make the factorized structure of the problem explicit by introducing (hidden) transformation variables $\mathbf{\Delta}_{i\to j} := \mathbf{X}_i^{-1}\mathbf{X}_j$ for all pairs $\langle \mathbf{X}_i, \mathbf{X}_j \rangle$ of body parts. We represent the 6D pose vectors $\mathbf{X}$ as their equivalent homogeneous transformation matrices, which means that $\mathbf{\Delta}_{i\to j}$ reflects the (deterministic) relative transformation between $\mathbf{X}_i$ and $\mathbf{X}_j$. Note that various parameterizations of such transformation matrices are possible (e.g., by means of the Euler angles, quaternions, or over-parameterized as the full 12D matrices) and that we thus assume a d-dimensional parameterization of $\mathbf{\Delta}_{i\to j}$.

Denoting with $\mathbf{Z}_{i\to j} := \mathbf{Y}_i^{-1}\mathbf{Y}_j$ the transformation relating the observations $\mathbf{Y}_i$ and $\mathbf{Y}_j$ that correspond to $\mathbf{X}_i$ and $\mathbf{X}_j$, we define as a *local model* the subgraph of our network that defines the relationship between any two body parts $\mathbf{X}_i$ and $\mathbf{X}_j$ and their dependent variables, if all other body parts are ignored. The right diagram in Fig. 4.2 shows a prototypical local model. Here, we denote with $\mathcal{A}_{i\to j}$ the set of action variables that have a direct influence on $\mathbf{\Delta}_{i\to j}$. Any set of $n-1$ local models which forms a spanning tree over all n body parts defines a model for the whole kinematic structure.

In the following, we explain (a) how to learn local models from data and (b) how to find the best spanning tree built from these local models that explains the whole robot. We

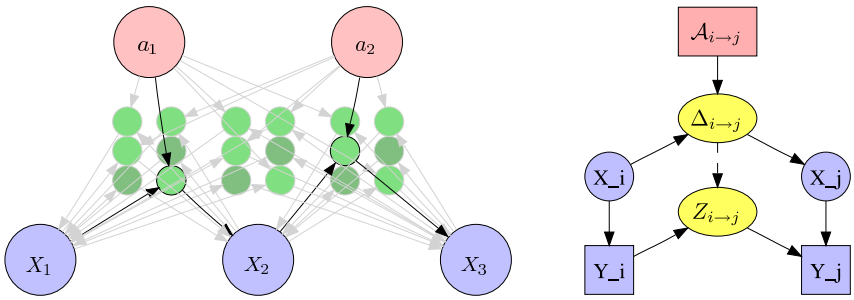


Figure 4.2: Left: Example network model linking action signals a_1 and a_2 to body part locations $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ using local models (green circles). The black arrows visualize the current best minimal spanning tree found during the search for the most compact model. Right: Local model for two body parts $\mathbf{X}_i$ and $\mathbf{X}_j$ as well as their dependent variables. $\mathcal{A}_{i\to j}$ denotes the set of independent action variables that cause a local transformation $\mathbf{\Delta}_{i\to j}$. $\mathbf{Y}_i$ and $\mathbf{Y}_j$ are the observed part locations, and $\mathbf{Z}_{i\to j}$ is their relative geometric transformation.

consider the single best solution only and do not perform model averaging over possible alternative structures. Note that in theory, it would be straight-forward to keep multiple structure hypotheses for *prediction* and to average over them. *Control* under structure uncertainty, however, is a slightly more difficult problem. One would have to consider all possible structures and assess the individual risks and gains for alternative actions. Then, the one action sequence would be selected that maximizes the overall gain while keeping all possible risks low [Stachniss *et al.*, 2005]. In practice, we found that considering only the most-likely structure is sufficient for the relevant tasks. Our approach is conservative in this respect since it requires a certain minimal accuracy from all parts of the body scheme before the model is considered complete.

4.1.1 Local Models

The local kinematic models are the central concept in our body scheme framework. A *local model* $\mathcal{M}$ (see the right diagram in Fig. 4.2) describes the geometric relationship between two body parts i and j given a set of action signals $\mathcal{A}_{i\to j}$. We propose to learn this relationship from data samples acquired while performing random actions and observing their effects. As the learning framework for solving this supervised regression problem, we apply Gaussian processes for regression as described in Sec. 2.2. On the real robotic platform used in our experiments, the action a_i correspond to the target angle requested from joint i. The observations $\mathbf{Y}_i$ of part locations $\mathbf{X}_i$ are obtained by tracking visual markers in 3D space including their 3D orientation [Fiala, 2004] (see the left image in Fig. 4.1). Note that the $\mathbf{Y}_i$'s are inherently noisy and that missing observations are common—for example in the case of (self-)occlusion.

Formally, the task is to learn the local transformations $\mathbf{\Delta}_{i\to j}$, each linking two body parts $\mathbf{X}_i$ and $\mathbf{X}_j$. Considering the right diagram in Fig. 4.2, a straight-forward approach would be to infer the true poses $\mathbf{X}_i$ and $\mathbf{X}_j$ from the noisy observations $\mathbf{Y}_i$ and $\mathbf{Y}_j$, for instance assuming Gaussian white noise on the observations, $\mathbf{Y}_i \sim \mathcal{N}(\mathbf{X}_i, \sigma^2_{\text{sensor}} \cdot \mathbf{I}_d)$. With $\mathbf{I}_d$ we denote the d-dimensional identity matrix and remind that d denotes the chosen dimensionality of the parameterization of tranformation matrices. In other terms, one would be following the full Bayesian pathway $\mathbf{Y}_i \leftarrow \mathbf{X}_i \rightarrow \mathbf{\Delta}_{i\to j} \rightarrow \mathbf{X}_j \rightarrow \mathbf{Y}_j$ to reason about $\mathbf{\Delta}_{i\to j}$.

However, since the *absolute* positions $\mathbf{X}_i$ are irrelevant for describing the *relative* transformations, we take a slightly different approach by concentrating on the transformations $\mathbf{Z}_{i\to j}$ between *observations* $\mathbf{Y}_i$ and $\mathbf{Y}_j$. We model $\mathbf{Z}_{i\to j} \sim \mathcal{N}(\mathbf{\Delta}_{i\to j}, \sigma^2_{\text{sensor-rel}} \cdot \mathbf{I}_d)$, and thus follow the shorter path $\{\mathbf{Y}_i, \mathbf{Y}_j\} \leftarrow \mathbf{Z}_{i\to j} \leftarrow \mathbf{\Delta}_{i\to j}$, which does not include the $\mathbf{X}_i$ explicitely. The problem of learning a single local model now has the form of the noisy regression problem as defined in Eq. (2.1), that is, learning the function

$$\begin{aligned} f_{\mathcal{M}}: \quad \mathbb{R}^{|\mathcal{A}_{i\to j}|} &\rightarrow \mathbb{R}^d, \\ \mathcal{A}_{i\to j} &\mapsto \Delta_{i\to j} \end{aligned} \tag{4.3}$$

from noisy observations $\mathbf{Z}_{i\to j}$. For simplicity, we consider over-parameterized transformation matrices in the following with $d = 12$ independent components and learn the functional mapping for each component separately. Due to this simplification, we cannot guarantee that all predictions correspond to valid, homogeneous transformation matrices. In practice, however, invalid transformations occur only rarely and they lie close to similar, valid transformations, such that a normalization step (orthonormalizing the rotation part using singular value decomposition) resolves the problem. In the future, we might consider more efficient parameterizations that come closer to the actual 6-DOF of the transformations.

For solving the regression problem (4.3), we place individual Gaussian process priors (see Chapter 2) on the 12 variables of the transformation functions $f_{\mathcal{M}}$ for all local models $\mathcal{M}$ and choose the squared exponential covariance function [see Eq. (2.3)] to parameterize the process. Figure 4.3 shows the x, y, and z components of two different local models learned from real data using the GP model. In the situation shown in the left diagram, the action variable (x-axis) physically corresponds to the transformation being measured (y-axis). Thus, the data set is self-consistent and accurate functions with low noise levels can be learned. The higher noise level for the z-component is due to larger measurement error in this direction (i.e. the camera's line of vision). In the situation depicted on the right, a local model has been learned for variables that do not have a direct physical relationship. As a result, the model shows high noise levels and it does not

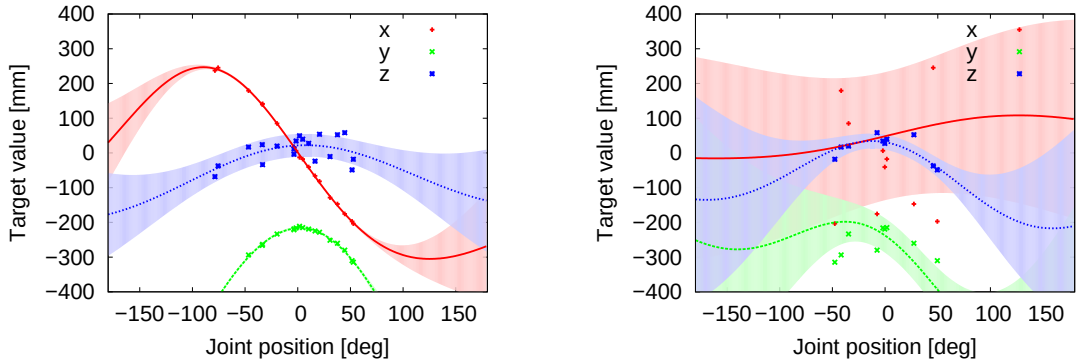


Figure 4.3: Left: Example of an accurate local model learned for two body parts and an action variable. Note the low predictive variance for the x- and y components as well as the higher noise in the z dimension, which is due to higher measurement uncertainty in this direction. Right: Less accurate model learned for the same body parts but a different action variable. Such a local model is less likely to be part of the Bayesian network describing the full kinematic chain of the robot since, on average, its predictions are less accurate.

explain the data well. Such a local model is likely to be discarded during the search for the full body model, which is described in the following.

4.1.2 Learning a Factorized Full Body Model

We seek to find a factorized model for the whole kinematic structure [see Eq. (4.2)] that explains the observed data well and that is not overly complex—such that it can be learned and evaluated online. To limit complexity, we first discard all local models that are overly inconsistent with the observed data. We define a local model $\mathcal{M}$ to be $valid_{\mathcal{M}}(\mathcal{D})$ given a set of observations, if and only if its prediction error is below some threshold θ, i.e., $\epsilon_{\text{pred}}(\mathcal{D}) < \theta$. Our experiments revealed that a good value for θ is 3σ, where σ is the standard deviation of the sensor model. The prediction error $\epsilon_{\text{pred}}(\mathcal{D}|\mathcal{M})$ is defined as

$$\epsilon_{\text{pred}}(\mathcal{D} \mid \mathcal{M}) := \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{Z}_{i\to j}, \mathcal{A}_{i\to j}) \in \mathcal{D}} \epsilon_{\text{pred}}(\mathbf{Z}_{i\to j} \mid \mathcal{A}_{i\to j}, \mathcal{M}) \tag{4.4}$$

with

$$\epsilon_{\text{pred}}(\mathbf{Z}_{i\to j} \mid \mathcal{A}_{i\to j}, \mathcal{M}) := \frac{1}{d} \sqrt{\sum_{z \in \mathbf{Z}_{i\to j}} (z - \mu_z^*)^2}\,. \tag{4.5}$$

Here, μ_z^* is the GP mean prediction of component z of transformation μ_z^*. Denoting with $C(\mathcal{M}) \in \mathbb{N}$ the dimensionality of model $\mathcal{M}$, that is, the number $|\mathcal{A}_{i\to j}|$ of action signals

that the model depends on, we define a *model quality measure* $q(\mathcal{D}|\mathcal{M})$,

$$\log q(\mathcal{D} \mid \mathcal{M}) \quad = \quad \underbrace{\log[1/\epsilon_{\text{pred}}(\mathcal{D} \mid \mathcal{M})]}_{\text{accuracy}} - \underbrace{C(\mathcal{M}) \log 1/\theta}_{\text{complexity}} \tag{4.6}$$

which is proportional to both the model accuracy and to a penalty term for model complexity. Note that quality measures such as the Bayesian information criterion (BIC) are not applicable here directly since GP regression is a *nonparametric* method and, thus, there is no obvious number of parameters for representing the data sets. Note also that the marginal data likelihood [see Eq. (2.15)], which is used to learn the parameters of the covariance function, might serve as an alternative model quality measure. However, since $q(\mathcal{D}|\mathcal{M})$ includes the model dimensionality explicitly and since this makes it easy to order the search for local models by this criterion, we have used this measure in our experiments.

Finding the Network Topology

If no prior knowledge about the robot's body structure exists, we initialize a fully connected network model containing a total of $\sum_{k=0}^{m} \binom{n}{2}\binom{m}{k}$ local models (linking n actions to m transformations). Given a set of self observations, the robot can first eliminate those local models that are highly inconsistent with the data by evaluating $valid_{\mathcal{M}}(\mathcal{D})$ as described above. The remaining set of valid models is typically still large [e.g., see Fig. 4.4]. Certain ambiguities will, for instance, remain even after infinitely many training samples. If, for example, $p_{\mathcal{M}_1}(\mathbf{Z}_{1\to 2} \mid a_1)$ has been determined to be a valid local model, then $p_{\mathcal{M}_2}(\mathbf{Z}_{1\to 2} \mid a_1, a_2)$ will also be. Although these alternative models might not be distinguishable regarding prediction accuracy for $\mathbf{Z}_{1\to 2}$, they differ significantly in their complexity and therefore in their model quality $q(\mathcal{D} \mid \mathcal{M})$.

To resolve such locally ambiguous situations and to also find the best topology on a global level, we seek to select the minimal subset $\mathbb{M} \subset \mathbb{M}_{\text{valid}}$ from the superset of all valid local models $\mathbb{M}_{\text{valid}} = \{\mathcal{M}_1, \ldots\}$ that covers all body part variables and simultaneously maximizes the overall model fit $q(\mathcal{D} \mid \mathbb{M}) := \prod_{\mathcal{M}\in\mathbb{M}} q(\mathcal{D} \mid \mathcal{M})$. $\mathbb{M}$ can be found efficiently by computing the minimal spanning tree of $\mathbb{M}_{\text{valid}}$ taking the model quality measure of the individual local models as the cost function. For our purposes, the spanning tree needs to cover all body parts but not necessarily all action variables, since some of them might not have an influence on the robot.

To connect all n body poses in the Bayesian network, exactly $(n-1)$ local models need to be selected. This yields $\#_{structures} = \binom{\#_{local\ models}}{n-1}$ possible network structures to be considered. In the typical case, where the robot is composed of 1-DOF joints (arbitrarily connected), this number reduces to the order of $O(n^3)$. Regarding the scalability to higher degrees of

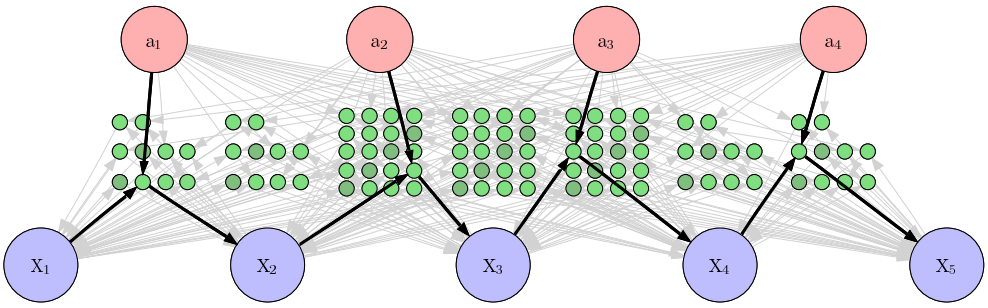


Figure 4.4: In an early learning phase, the robot knows only little about its body structure and, thus, all possible local models need to be considered in parallel (gray arrows). Using the subset of valid local models, a minimal spanning tree can be constructed to form a sparse Bayesian network (dark black arrows). This can subsequently be used as a body scheme for prediction and control.

freedom and longer kinematic chains, the growth of the search space is of less practical importance than other factors like the *observability* of local transformations (from a given camera view point). In practice, straight-forward search heuristics allow us to strongly focus the search on the relevant parts of the structure space, further reducing this number. In our experiments, for instance, we searched by processing the lower dimensional models first. Recall that the quality measure $q(\mathcal{D} \mid \mathcal{M})$ for a local model is composed of the (data-dependent) prediction accuracy and a (data-independent) complexity penalty. If we consider two valid local models, i.e., with $\epsilon_{\text{pred}}(\mathcal{D} \mid \mathcal{M}_{1|2}) < \theta$, then by the definition of $q(\mathcal{D} \mid \mathcal{M})$, the quality of a model with lower complexity is always higher compared to a local model with higher complexity for any $\mathcal{D}$, i.e.,

$$C(\mathcal{M}_1) < C(\mathcal{M}_2) \Longleftrightarrow \forall \mathcal{D} : q(\mathcal{D} \mid \mathcal{M}_1) > q(\mathcal{D} \mid \mathcal{M}_2) \ . \tag{4.7}$$

This is due to the fact that $C(\mathcal{M})$ contains the error threshold θ and that this is also the upper bound of all prediction errors ϵ_{pred} [see Eq. (4.5)]—all models above this threshold are invalid and thus discarded. Due to (4.7), it is sufficient to evaluate only the first k complexity layers of local models in $\mathbb{M}_{\text{valid}}$ until a minimal spanning tree is found for the first time. This spanning tree then corresponds to the global maximum of overall model quality.

4.1.3 Prediction and Control

Having discussed the learning of local models and the selection of the network structure, we now show how the resulting model can be used to predict the configuration of the robot for a given action signal (forward modeling) and how to select actions to achieve a given configuration (inverse modeling).

The *kinematic forward model* can be constructed directly from the local models contained in $\mathbb{M}$, since these form a tree over all body part variables $\mathbf{X}_i$. We can write

$$\begin{aligned} p(\mathbf{X}_1, \ldots, \mathbf{X}_n \mid a_1, \ldots, a_m) &= \prod_i p(\mathbf{X}_i \mid \text{parents}(\mathbf{X}_i)) \\ &= p(\mathbf{X}_{\text{root}}) \prod_{\mathcal{M}_{i\to j} \in \mathbb{M}} p_{\mathcal{M}_{i\to j}}(\Delta_{i\to j} \mid \mathcal{A}_{i\to j}) \\ &= p(\mathbf{X}_{\text{root}}) \prod_{\mathcal{M}_{i\to j} \in \mathbb{M}} p_{\mathcal{M}_{i\to j}}(\mathbf{X}_i^{-1}\mathbf{X}_j \mid \mathcal{A}_{i\to j}) \ , \end{aligned} \tag{4.8}$$

where $\mathbf{X}_{\text{root}}$ is the position of the robot's trunk, which serves as the reference frame for all other body parts. We denoted with $\mathcal{M}_{i\to j}$ the local model of $\mathbb{M}$ which describes the transformation between $\mathbf{X}_i$ and $\mathbf{X}_j$. From $p(\mathbf{X}_1, \ldots, \mathbf{X}_n | a_1, \ldots, a_m)$ in the factorized form, we can now approximate the maximum likelihood (ML) estimate of the resulting body posture given an action signal by concatenating the geometric transformations of the individual geometric transformations. See [Ware and Lad, 2003] for how products of Gaussians can be approximated by a single Gaussian.

Although the *inverse kinematic model* can in principle be derived by applying Bayes' rule,

$$p(a_1, \ldots, a_m \mid \mathbf{X}_1, \ldots, \mathbf{X}_n) = \frac{p(a_1, \ldots, a_m)}{p(\mathbf{X}_1, \ldots, \mathbf{X}_n)} \ p(\mathbf{X}_1, \ldots, \mathbf{X}_n \mid a_1, \ldots, a_m)$$

it is difficult in practice to determine the maximum likelihood (ML) solution for the action signal $a_1, \ldots, a_m$. This is due to the fact that the goal configuration is typically not fully specified for all body parts, but rather for the root part and the end-effector only. Thus, the Bayesian network is constrained at both "ends" only, which results in a high-dimensional optimization problem.

For this reason, we resort to the following well-known iterative approach, which applies small changes to the *current* action signal such that the body posture $\mathbf{X}_1, \ldots, \mathbf{X}_n$ approaches the target configuration. Since all individual functions $f_{\mathcal{M}_i}$ are continuous, and so is the ML posture estimate f of the forward kinematic model, we can compute the Jacobian $\nabla f(\mathbf{a})$ of the forward model as

$$\nabla f(\mathbf{a}) \ = \ \left[\frac{\partial f(\mathbf{a})}{\partial a_1}, \ldots, \frac{\partial f(\mathbf{a})}{\partial a_m} \right]^T . \tag{4.9}$$

Given $\nabla f(\mathbf{a})$, it is straight-forward to implement a gradient descent-based algorithm that continuously minimizes the distance function and, thus, controls the manipulator towards the target configuration. While such a "greedy" controller may get trapped in local minima of the distance function and fails to plan around obstacles in general, it nevertheless

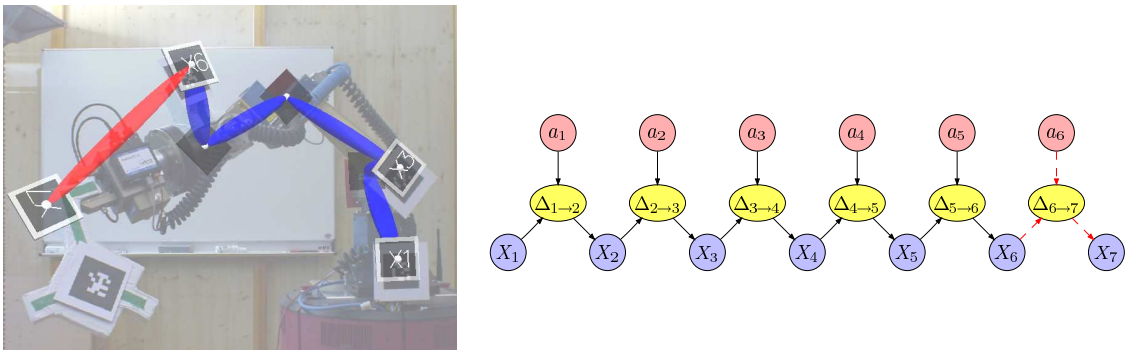


Figure 4.5: Left: After a different tool is placed in the robot's end-effector, the model predictions do not fit the current observations anymore. Right: The current body scheme linking action signals a_i and body parts $\mathbf{X}_j$ using local models $\mathbf{\Delta}_{j\to k}$. Here, a mismatch between the internal model and recent self-observation has been detected at $\mathbf{\Delta}_{6\to7}$.

solves many important control tasks and it builds the basis for higher-level path-planning algorithms, such as probabilistic road-maps.

4.2 Failure Awareness and Life-Long Adaptation

Until now, we have assumed that the robot's physiology remains unchanged during its whole life-time. It is clear, however, that in real-world applications, the robot will change in the course of time. This requires that the robot revises parts of its experience over time, allowing it to discriminate between earlier and more recent observations. We would like the robot to detect changes in its physiology by testing the validity of its local models at different points in time and at different temporal scales. It might even be useful for the robot to maintain multiple body schemes at different time scales. Consider, for example, a robot that uses an accurate pre-programmed model over a long period of time, but which is also able to create and use a new models that takes over as soon as the body structure of the robot changes (which could be as little as the displacement of one visual marker). Such a situation is depicted in Fig. 4.5 and Fig. 4.6. In this experiment, we changed the end-effector body part without notifying the system. The task then was, to automatically detect the change and to learn a replacement for the non-fitting local model.

For dealing with model changes over time, we consider *temporal local models* $\mathcal{M}^T$ that describe the geometric relationship $p^T_{\mathcal{M}}(\mathbf{Z}_{i\to j} \mid \mathcal{A}_{i\to j}, T)$ between two observed body parts $\mathbf{Y}_i$ and $\mathbf{Y}_j$ given a subset of the action signal $\mathcal{A}_{i\to j} \subset \{a_1, \ldots, a_n\}$ and a particular time interval T. However, the size of the learning problem now also grows exponentially in time yielding the immense upper bound of $\sum_{k=0}^{m} \binom{n}{2}\binom{m}{k} 2^{|T|}$ local models to be considered. As it would be practically infeasible to evaluate all of these local models even for small periods of time, three additional assumptions can be made such that an efficient algorithm

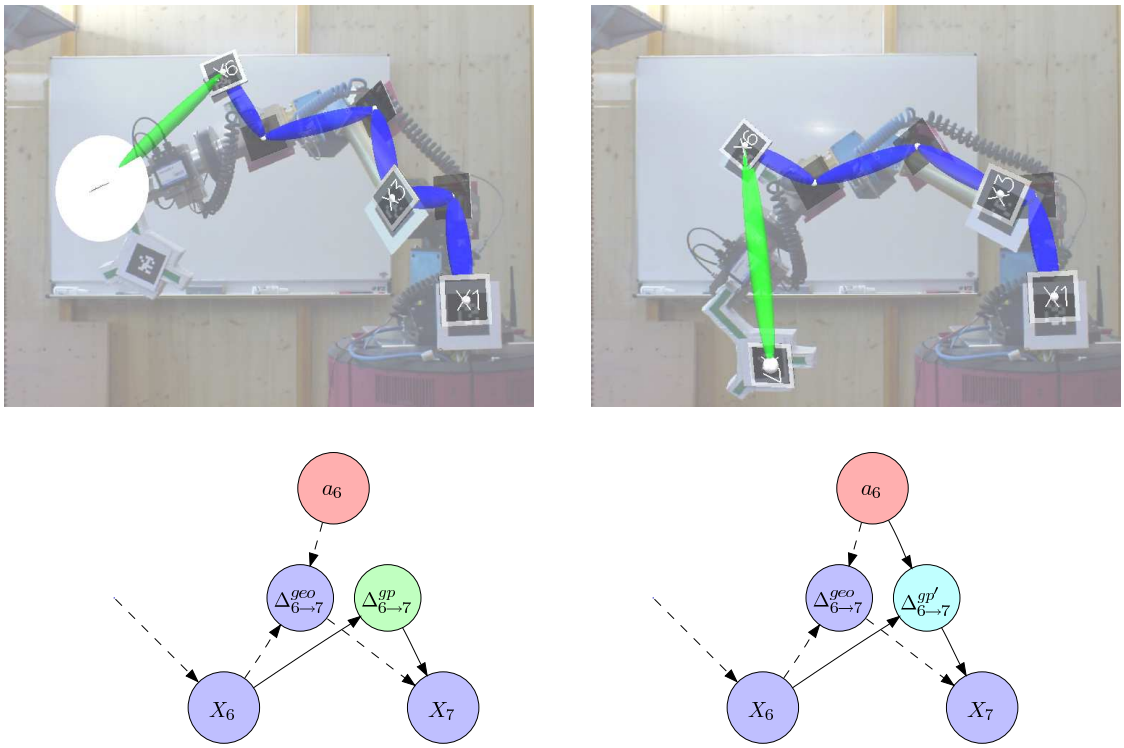


Figure 4.6: Continued experiment from Fig. 4.5. The robot samples a local model as replacement for the mismatching component $\mathbf{\Delta}_{6\to7}$. Left: The first newly sampled model ($\mathbf{\Delta}^{gp}_{6\to7}$) has high uncertainty, because of the missing dependency on action a_6. Right: The second sampled model ($\mathbf{\Delta}^{gp'}_{6\to7}$) is a more suitable replacement for the mismatching component.

for online application can be implemented:

1. Changes in body physiology are relatively rare events.
2. Changes in physiology happen incrementally.
3. Whatever local models were useful in the past, it is likely that similar—or even the same—local models will be useful in the future.

Due to Assumption 1, we do not have to re-learn the local models continuously and re-optimize the network, but rather it is sufficient to monitor the prediction accuracies of the models until one of them is not evaluated as *valid* any more. In this case, Assumption 2 states that the network cannot change completely at a given time step, but that we can recover the new structure by exchanging non-valid local models by re-learned ones individually. Furthermore, according to Assumption 3, it is reasonable to begin the search for new models with those that are similar to previously useful models, i.e., to keep a history of successful local models and to start searching within this history before learning new models from scratch.

These rules were incorporated into an integrated system, that is able to learn a body scheme from scratch and to exchange local models at a later stage, whenever a misfit is detected. For rating and ordering alternative local models, we consider the *structural proximity* $C(\mathcal{M}_2 \mid \mathcal{M}_1)$ of two local models which we define as the ratio of shared nodes in the Bayesian network. This way, models that depend on a similar set of variables are given preference in the search. We now present an experimental evaluation of the integrated system in simulation and on two real robotic manipulators.

4.3 Experimental Results

We tested our approach in a series of experiments both on a real robot and in simulation. The goal of our experiments was to verify that

1. physiological changes are detected confidently (blocked joints / deformations),
2. the body scheme is updated automatically without human intervention, and
3. the resulting body scheme can be used for accurate prediction and control.

The two real robots used to carry out the experiments were equipped with a 2-DOF and with a 6-DOF manipulator, respectively, composed of Schunk PowerCube modules (see the left picture in Fig. 4.1). With nominal noise values of ($\sigma_{\text{joints}} = 0.02°$), the reported joint positions of the encoders were considered to be sufficiently accurate to compute the ground truth positions of the body parts from the known geometrical properties of the robot. Visual perception was implemented using a Sony DFW-SX900 FireWire camera at a resolution of 1280x960 pixels. Seven black-and-white markers were attached to the robot's joints and the ARToolkit vision module [Fiala, 2004] was used to continuously estimate their 6D poses. The standard deviation of the camera noise was measured to $\sigma_{\text{markers}} = 44$ mm in 3D space, which is acceptable considering that the camera was located two meters apart from the robot. The prediction errors and errorbars reported in the following were estimated on an independent test set of $|\mathcal{D}_{\text{testing}}| = 15$ data samples.

Evaluation of Model Accuracy

To quantitatively evaluate the accuracy of the kinematic models learned from scratch as well as the convergence behavior of our learning approach, we generated random action sequences and analyzed the intermediate models using the 2-DOF robot of which the kinematic model is perfectly known. Figure 4.7 gives the absolute errors of prediction and control after certain numbers of observations have been processed. For a reference, we also give the average observation noise, i.e. the absolute localization errors of the visual markers. As can be seen from the diagram, the body scheme converges robustly within

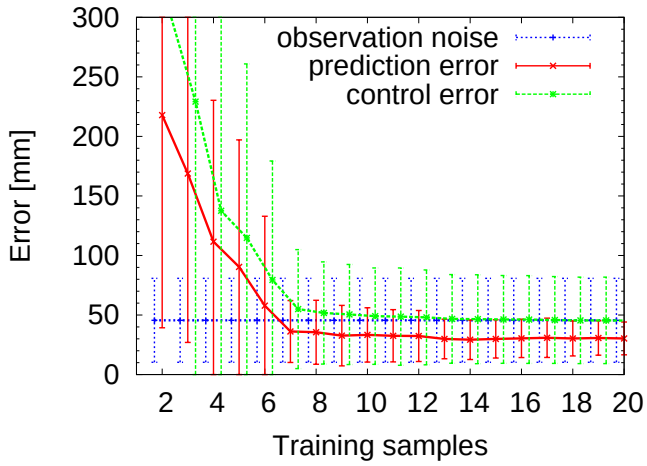


Figure 4.7: Prediction and control errors for a kinematic model that is learned from scratch. Already after 7 samples, the average prediction error is lower than the average localization error of the visual markers.

the first 10 observations. After about 15 training samples, the accuracy of the predicted body part positions becomes even higher than the accuracy of the direct observations. The latter is a remarkable result as it means that, although all local models are learned from noisy observations, the system is able to "blindly" estimate its configuration more accurately than immediate perception. The figure also gives the accuracy of the gradient-based control algorithm. Here, we used an additional marker for defining a target location for the robot's end effector. We learned the full body scheme model from scratch as in the previous experiment and used the gradient-based control algorithm to bring the end effector to the desired target location. The average positioning error is in the order of the perception noise (approx. 50 mm, see Fig. 4.7), which is slightly higher than the prediction error alone.

Scenario 1: Blocked Joint

In a second experiment we used the 6-DOF robot (see the left picture in Fig. 4.1) to evaluated how well the proposed system can detect a stuck joint and repair its model accordingly. To this aim, we initialized the body scheme with an accurate, manually calibrated model. Upon detection of a model-mismatch, new local models were trained from $|\mathcal{D}_{\text{training}}| = 30$ consecutive training samples recorded after the model was instantiated. In order for a local model to be *valid*, its translational and rotational error on the test set was required to be below a threshold of $\theta_{\text{trans}} = 3\sigma_{\text{trans}} = 150$ mm and $\theta_{\text{rot}} = 3\sigma_{\text{rot}} = 45°$, with σ_{trans} and σ_{rot} the standard deviations of the translational and rotational observation noise, respectively. New local models were only sampled when no valid spanning tree could

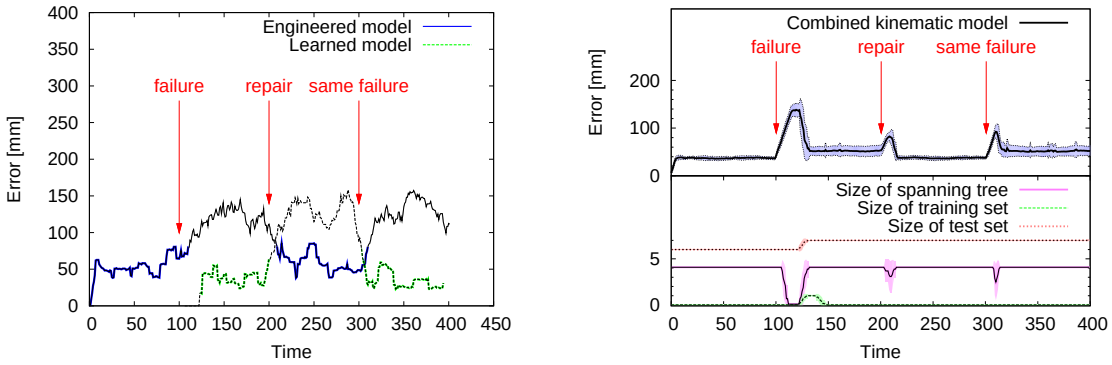


Figure 4.8: Left: At $t = 100$, a joint gets blocked, which causes the initial local model $p_{\text{engineered}}(\mathbf{Z}_{6\to7} \mid a_4)$ to produce substantially larger prediction errors. At $t = 126$, the robot samples a new local model $p_{\text{learned}}(\mathbf{\Delta}_{6\to7})$ as replacement. Right: The absolute prediction error of the combined kinematic model $p(\mathbf{Z}_{1\to7} \mid a_1, \ldots, a_4)$ of our 6-DOF manipulator. This model is composed of 6 individual local models of which one is replaced by a newly learned model at $t = 126$. As can be seen from the plot, the prediction accuracy recovers quickly after each of the three external events.

be constructed for $|\mathcal{D}_{\text{testing}}|$ consecutive time steps, as this is the time it takes to replace most or all data samples of the test set.

We generated a large sequence of random motor commands $\langle a_1, \ldots, a_m \rangle$. Before accepting a pose, we checked that the configuration would not cause any (self-)collisions and that the markers of interest would potentially be visible on the camera image. This sequence was sent to the robot and after each motion command, the observed marker positions $\langle \mathbf{Y}_1, \ldots, \mathbf{Y}_n \rangle$ were recorded. We allowed for arbitrary 3D motion (just constrained by the geometry of the manipulator) and thus do not assume full visibility of the markers. In the rare case of an anticipated or actual (self-)collision during execution, the robot stopped and the sample was rejected. Analysis of the recorded data revealed that, on average, the individual markers were visible only in 86.8% of the time with the initial body layout. In a second run, we blocked the robot's end-effector joint a_4, so that it could not move, and again recorded a log-file. An automated test procedure was then used to evaluate the performance and robustness of our approach. For each of 20 recorded runs, a new data set was sampled from the log-files, consisting of 4 blocks with $N = 100$ data samples each. The first and the third block were sampled from the initial body shape, while the second and the fourth block were sampled from the log-file where the joint got blocked.

The left diagram in Fig. 4.8 shows the absolute errors of the local models predicting the end-effector pose. As expected, the prediction error of the engineered local model increases significantly after the end-effector joint gets blocked at $t = 100$. After a few samples, the robot detects a mismatch in its internal model and starts to learn a new dynamic model

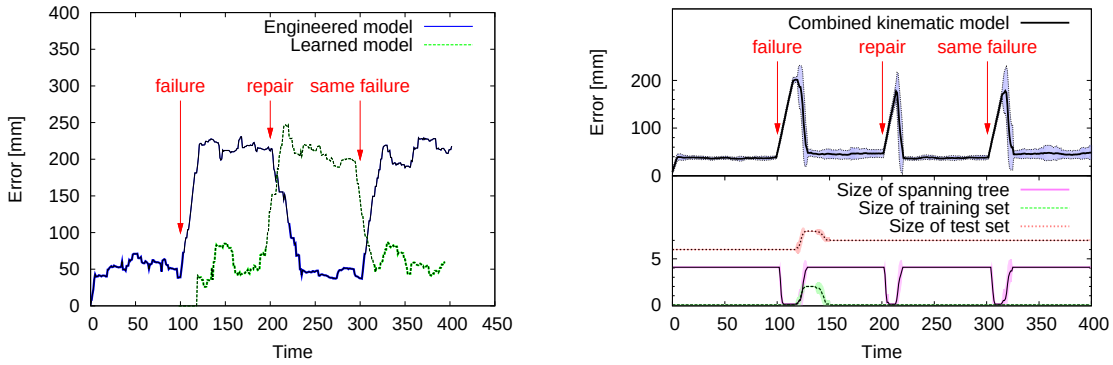


Figure 4.9: After the limb length and orientation of the end-effector were changed, the robot had to sample two local models on average before the spanning tree of the body scheme could be restored.

(around $t = 130$), which quickly reaches the same accuracy as the original, engineered local model. At $t = 200$, the joint gets repaired (unblocked). Now the estimated error of the newly learned local model quickly increases while the estimated error of the engineered local model decreases rapidly towards its initial accuracy. Later, at $t = 300$, the joint gets blocked again in the same position, the accuracy of the previously learned local model increases significantly, and thus the robot can re-use this local model instead of having to learn a new one.

The precision of the combined model—i.e. the engineered one fused with the one learned after having detected the failure—are given in the right diagram in Fig. 4.8 for 20 reruns of the experiment. The hand-tuned initial geometrical model evaluates to an averaged error at the end-effector of approx. 37 mm. After the joint gets blocked at $t = 100$, the error in prediction increases rapidly. After $t = 115$, a single new local models gets sampled, which already is enough to bring down the overall error of the combined kinematic model to approximately 51 mm. Training of the new local model is completed at around $t = 135$. Later, at $t = 200$, when the joint gets un-blocked, the error estimate of the combined kinematic model increases slightly, but returns much faster to its typical accuracy: Switching back to an already known local model requires less data samples than learning a new model (see Tab. 4.1). At $t = 300$, the same quick adaption can be observed when the joint gets blocked again.

Scenario 2: Deformed limb

In a third experiment, we changed the end-effector limb length and orientation and applied the same evaluation procedure as in the previous subsection. This was accomplished by placing a tool with an attached marker in the robot's gripper and changing its configuration

during the experiment (see Fig. 4.5 and Fig. 4.6). The quantitative results for 20 runs are given in Fig. 4.9. After the tool gets displaced at $t = 100$, two local models have to be sampled on average to repair the kinematic model. The prediction accuracy of the whole system closely resembles the levels that were obtained in the case of the blocked joint: On average, we measured an accuracy of 47 mm after recovery. In Tab. 4.1, we summarize the recovery times for this and the previous experiment. As can be seen from the results, the system adapts to a blocked joint quicker than to a deformed limb, and recalling a previously successful model—i.e. the engineered one after "repair" or the newly learned one after "same failure"—is significantly faster than learning from scratch (after "failure").

Controlling a Deformed Robot

Finally, we ran a series of experiments to verify that dynamically maintained body schemes can be used for accurate positioning and control. The experiments were performed on a simulated 4-DOF manipulator. We defined a 3D trajectory consisting of 30 way-points that the manipulator should approach using the inverse kinematics derived from its current body scheme, see Fig. 4.10. When the initial geometric model was used to follow the trajectory by using the undamaged manipulator, a positioning accuracy of 7.03mm was measured. When the middle limb was deformed by 45°, the manipulator with a static body scheme was significantly off course, leading to an average positioning accuracy of 189.35mm. With dynamic adaptation enabled, the precision settled at 15.24mm. These results are also summarized in Tab. 4.2 including the two standard deviations of the errors obtained on 20 runs. The results show that dynamic model adaption enables a robot to maintain a high positioning accuracy even after substantial changes to its body.

4.4 Related Work

The problem of learning kinematics of robots has been investigated heavily in the past. For example, Kolter and Ng [2007] enable a quadruped robot to learn how to follow

Table 4.1: Evaluation of the recovery time required after being exposed to different types of failures. We give the mean recovery times for 20 runs, respectively, and the (±) standard deviation spread.

Visibility rate	Failure type	Recovery time after		
		failure	repair	same failure
91.9%	Joint blocked	16.50 ± 1.20	0.45 ± 0.86	0.65 ± 1.15
79.0%	Limb deformed	20.20 ±1.96	11.10 ± 0.83	12.10 ± 1.64

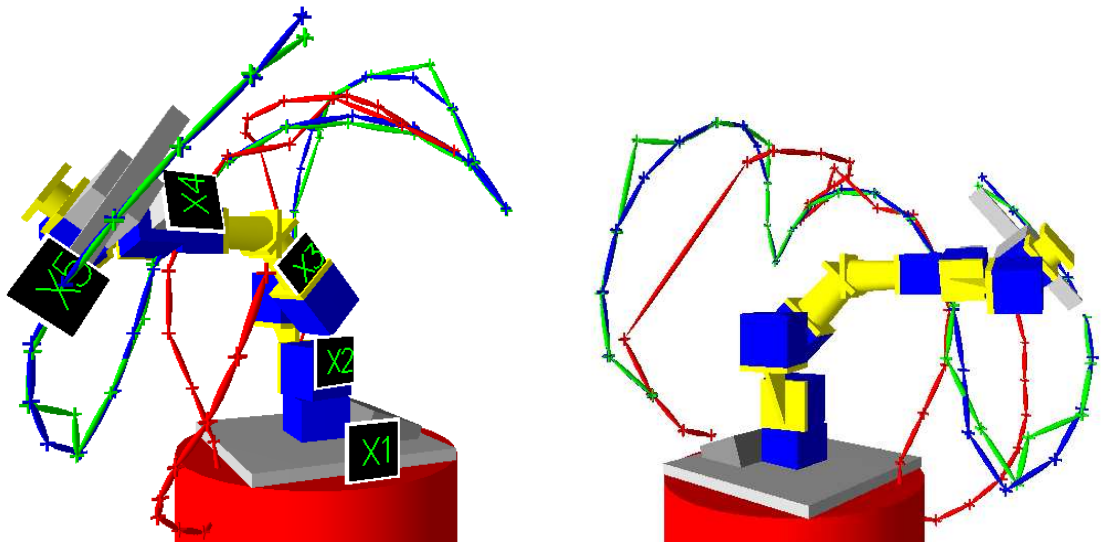


Figure 4.10: The manipulator robot with a deformed limb has to follow the blue target trajectory. With a static body model, it suffers from strong derivation (red trajectory). By using our approach, the body scheme is dynamically adapted, and the trajectory is very well approached (green trajectory).

omnidirectional paths using dimensionality reduction techniques and based on simulations. Their key idea is to use the simulator for identifying a suitable subspace for policies and then to learn with the real robot only in this low-dimensional space. A similar direction has been explored by Dearden and Demiris [2005], who applied dimensionality reduction techniques to unveil the underlying structure of the body scheme. Similar to this work, their approach is formulated as a model selection problem between different Bayesian networks. Another instance of approaches based on dimensionality reduction is the work by Grimes *et al.* [2006] who applied the principal component analysis (PCA) in conjunction with Gaussian process regression for learning walking gaits on a humanoid robot.

Yoshikawa *et al.* [2004a] used Hebbian networks to discover the body scheme from self-occlusion or self-touching sensations. Later, Yoshikawa *et al.* [2004b] learned classifiers for body/non-body discrimination from visual data. Other approaches used for example nearest-neighbor interpolation [Morasso and Sanguineti, 1995] or neural networks [Natale,

Table 4.2: Evaluation of the inverse kinematics experiment in simulation. By sampling new local models upon the detection of a failures, the robot is able to regain high positioning accuracy.

Body shape	Model type	Control error
initial (unchanged)	static	7.03 ± 10.87
deformed	static	189.35 ± 28.36
deformed	adaptive	15.24 ± 1.86

2004]. Recently, Ting *et al.* [2006] developed a Bayesian parameter identification method for non-linear dynamic systems, such as a robotic arm or a 7-DOF robotic head.

The approach presented in this chapter is also related to the problem of self-calibration which can be understood as a subproblem of body scheme learning. When the kinematic model is given in a parametric form, the parameters can be estimated efficiently, in certain cases, by maximizing the likelihood of the model given the data [Roy and Thrun, 1999]. Genetic algorithms have been used by Bongard *et al.* [2006a] for parameter optimization when no closed form is available. To a certain extend, such methods can also be used to calibrate a robot that is temporarily using a tool [Nabeshima *et al.*, 2005]. In contrast to the work presented here, such approaches require a parameterized kinematic model of the robot.

To achieve continuous self-modeling, Bongard *et al.* [2006b] recently described a robotic system that continuously learns its own structure from actuation-sensation relationships. In three alternating phases (modeling, testing, prediction), their system generates new structure hypotheses using stochastic optimization, which are validated by generating actions and by analyzing the following sensory input. In a more general context, Bongard and Lipson [2007] studied structure learning in arbitrary non-linear systems using similar mechanisms. In contrast to all the approaches described above, we propose an algorithm that learns both the structure as well as functional mappings for the individual building blocks. Furthermore, our model is able to revise its structure and component models on-the-fly.

4.5 Conclusion

In this chapter, we presented a novel approach to body scheme learning and life-long adaptation for a robotic manipulation system. Our central idea is to continuously learn a large set of local kinematic models using nonparametric regression and to search for the best arrangement of these models to represent the full system. To the best of our knowledge, this is the first time that such complex kinematic structures have been learned from scratch using visual self-observation only.

In experiments carried out with a real robot and in simulation, we demonstrated that our system is able to deal with missing and noisy observations, operates in full 3D space, and is able to perform relevant tasks like prediction, control, and online adaptation after failures.

Chapter 5

Mapping Gas Distributions

We develop a Gaussian process mixture model for representing and learning spatial models of gas distribution. We discuss sparse approximations for large data sets and give an efficient EM-based learning procedure to fit both the mixture components as well as the gating function that assigns the data points to them.

Gas distribution modeling has important applications in industry, science, and every-day life. Mobile robots equipped with gas sensors are deployed, for example, for pollution monitoring in public areas [DustBot, 2008], surveillance of industrial facilities producing harmful gases, or inspection of contaminated areas within rescue missions. Although humans have comparably precise odor sensors allowing to distinguish between around 10 000 odors, it is hard for us to build spatial representations of sensed gas distributions.

Building gas distribution maps is a challenging task in principle due to the chaotic nature of gas dispersal and, because only point measurements of gas concentration are available. The complex interaction of gas with its surroundings is dominated by two physical effects. First, on a comparably large timescale, *diffusion* mixes the gas with the surrounding atmosphere achieving a homogeneous mixture of both in the long run. Second, turbulent air flow fragments the gas emanating from a source into intermittent *patches* of high concentration with steep gradients at their edges [Roberts and Webster, 2002]. This chaotic system of localized patches of gas makes the modeling problem a hard one. In addition, gas sensors provide information about a small spatial region only since gas sensor measurements require direct interaction between the sensor surface and the molecules to be analyzed. This makes gas sensing different to perceiving the environment with other popular robotic sensors like

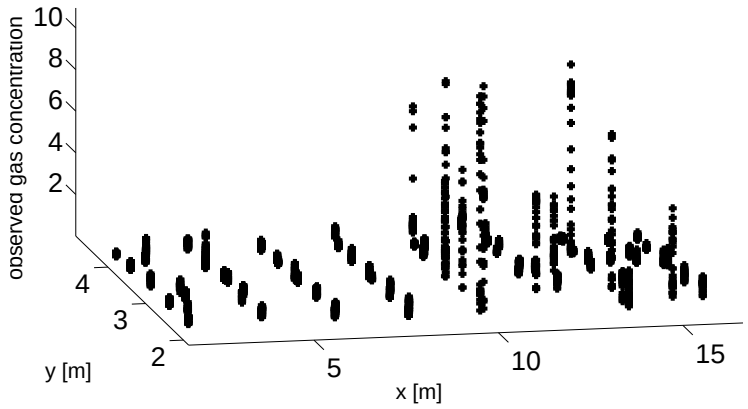


Figure 5.1: Gas concentration measurements acquired by a mobile robot in a corridor. The distribution consists of a rather smooth "background" signal and several peaks, which indicate high gas concentrations.

laser range finders, with which a larger area can be measured directly.

Figure 5.1 illustrates actual gas concentration measurements recorded with a mobile robot along a corridor containing a single gas source. The distribution consists of a rather smooth "background" signal and several peaks, which indicate high gas concentrations. The challenge in gas distribution mapping is to model this background signal while being able to cover also the areas of high concentration and their sharp boundaries. Since it is comparably costly to aquire measurements, one is also interested in reducing the number of samples needed to build a representation. It is important to note that the noise is dominated by the large fluctuations of the instantaneous gas distribution and not by the electronic noise of the gas sensors. From a probabilistic point of view, the task of modeling a gas distribution can be described as finding a model that best explains the observations and that is able to accurately predict new ones. Thus, the data likelihood of an independent test set (i.e. cross validation) is the standard criterion to evaluate such a model.

Simple spatial averaging, which represents a straight-forward approach to the modeling problem, disregards the different nature of the background noise and the peaks resulting from areas of high gas concentrations and, thus, achieves only limited prediction accuracy. On the other hand, precise physical simulation of the gas dynamics in the environment would require immense computational resources as well as precise knowledge about the physical conditions, which is not known in most practical scenarios.

To achieve a balance between model accuracy and efficiency, we treat gas distribution mapping as a supervised regression problem. We derive a solution by means of a sparse mixture model of Gaussian processes [Tresp, 2000] that is able to handle both physical

phenomena highlighted above. Formally, we interpret gas sensor measurements obtained from static sensors or from a mobile robot at locations as noisy samples from a time-constant distribution. This implies that the gas distribution in fact exhibits a time-constant structure, an assumption that is often made in unventilated and un-populated indoor environments [Wandel *et al.*, 2003].

While existing approaches to gas distribution mapping, such as local averaging [Ishida *et al.*, 1998, Purnamadjaja and Russell, 2005], kernel extrapolation [Lilienthal and Duckett, 2004], or standard GP models represent the average concentration per location only, our mixture model distinguishes explicitly different components of the distribution, i.e. concentration layers varying smoothly due to dispersion processes versus those containing localized patches of gas. This leads to a more accurate prediction of the gas concentration. Our model actually allows us to do both, computing the average gas concentration per location (as existing models supply) as well as the multi-modal, predictive densities. As a by-product, we present a generic algorithm that learns a GP mixture model and at the same time reduces the model complexity in order to achieve an efficient representation even for large data sets.

We demonstrate in experiments carried out with real mobile robots that our model has a lower mean squared error and a higher data likelihood on test data sets than other methods. Thus, it allows to predict gas concentration at query locations more accurately.

This chapter is organized as follows. After introducing our mixture model in Sec. 5.1, we explain how to learn the model components from data and how to achieve a sparse approximation. We then present experimental results involving real mobile platforms in Sec. 5.3 and discuss related work in Sec. 5.4.

5.1 A Mixture Model for Gas Distributions

The general gas distribution mapping problem given a set of concentration measurements $y_{1:n}$ acquired at locations $\mathbf{x}_{1:n}$, is to learn a predictive model $p(y_* \mid \mathbf{x}_*, \mathbf{x}_{1:n}, y_{1:n})$ for gas concentrations y_* at a query location $\mathbf{x}_*$. Recalling Sec. 2.2, we approach this problem in a nonparametric way, i.e., not assuming a parametric form of the underlying function $f : \mathbf{x} \mapsto y$, using Gaussian processes. For real-valued functions f, the model is defined by a mean function $m(\cdot)$ and a covariance function $k(\cdot, \cdot)$

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{5.1}$$

$$k(\mathbf{x}_p, \mathbf{x}_q) = \mathbb{E}[(f(\mathbf{x}_p) - m(\mathbf{x}_p))(f(\mathbf{x}_q) - m(\mathbf{x}_q))] \ . \tag{5.2}$$

In the following, we apply the squared exponential covariance function k_{SE} as defined in Sec. 2.2. Observations y obtained from the process are assumed to be affected by Gaussian noise, $y \sim \mathcal{N}(m(\mathbf{x}), \sigma_n^2)$. The variables $\boldsymbol{\theta} = \{\sigma_f, \ell, \sigma_n\}$ are the so-called hyper-parameters of the process which have to be learned from data.

Given a set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of training data where $\mathbf{x}_i \in \mathbb{R}^d$ are the inputs and $y_i \in \mathbb{R}$ the targets, the goal in regression is to predict target values $y_* \in \mathbb{R}$ at a new input point $\mathbf{x}_*$. Let $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_n]^\top$ be the $n \times d$ matrix of the inputs and $\mathbf{X}_*$ be defined analogously for multiple test data points. In the GP model, any finite set of samples is jointly Gaussian distributed

$$\begin{bmatrix} \mathbf{y} \\ f(\mathbf{X}_*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} & k(\mathbf{X}, \mathbf{X}_*) \\ k(\mathbf{X}_*, \mathbf{X}) & k(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right), \tag{5.3}$$

where $k(\cdot, \cdot)$ refers to the matrix with the entries given by the covariance function $k(\cdot, \cdot)$ and $\mathbf{y}$ the vector of the (observed) targets y_i. To make predictions at $\mathbf{X}_*$, we obtain the predictive mean

$$\bar{f}(\mathbf{X}_*) := \mathbb{E}[f(\mathbf{X}_*)] = k(\mathbf{X}_*, \mathbf{X}) \left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} \mathbf{y} \tag{5.4}$$

and the (noise-free) predictive variance

$$\mathbb{V}[f(\mathbf{X}_*)] = K(\mathbf{X}_*, \mathbf{X}_*) - k(\mathbf{X}_*, \mathbf{X}) \left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} k(\mathbf{X}, \mathbf{X}_*) \ , \tag{5.5}$$

where $\mathbf{I}$ is the identity matrix. The corresponding (noisy) predictive variance for an observation $\mathbf{y}_*$ can be obtained by adding the noise term σ_n^2 to the individual components of $\mathbb{V}[f(\mathbf{X}_*)]$.

The standard GP model recapitulated above has two major limitations in our problem domain. First, the computational complexity is high, since to compute the predictive variance given in Eq. (5.5), one needs to invert the matrix $K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I$. This introduces a complexity of $\mathcal{O}(n^3)$ where n is the number of training examples. As a result, an important issue for GP-based solutions to practical problems is the reduction of this complexity. This can, as we will show in Sec. 5.2, be achieved by artificially limiting the training data set in a way that introduces small loss in the data likelihood of the whole training set while at the same time minimizing the runtime. As a second limitation, the standard GP model generates a *uni-modal* distribution per input location $\mathbf{x}$. This assumption hardly fits our application domain in which a relatively smooth "background" signal is typically *mixed* with high-concentration "packets" of gas. In the following, we address this issue by deriving a *mixture model* of Gaussian processes.

Mixtures of Gaussian Process Models

The GP mixture model [Tresp, 2000] constitutes a locally weighted sum of several Gaussian process models. For simplicity of notation, we consider without loss of generality the case of single predictions only ($\mathbf{x}_*$ instead of $\mathbf{X}_*$). Let $\{\mathcal{GP}_1, \ldots, \mathcal{GP}_m\}$ be a set of m Gaussian processes representing the individual mixture components. Let $P(z(\mathbf{x}_*) = i)$ be the probability that $\mathbf{x}_*$ is associated with the i-th component of the mixture. Let $\bar{f}_i(\mathbf{x}_*)$ be the mean prediction of $\mathcal{GP}_i$ at $\mathbf{x}_*$. The likelihood of observing y_* is thus given by

$$h(\mathbf{x}_*) := p(y_* \mid \mathbf{x}_*) = \sum_{i=1}^{m} P(z(\mathbf{x}_*) = i) \cdot \mathcal{N}_i(y_*; \mathbf{x}_*)\,, \tag{5.6}$$

where we define $\mathcal{N}_i(y; \mathbf{x})$ as the Gaussian density function with mean $\bar{f}_i(\mathbf{x})$ and variance $\mathbb{V}[f_i(\mathbf{x})] + \sigma_n^2$ evaluated at y. One can sample from such a mixture by first sampling the mixture component according to $P(z(\mathbf{x}_*) = i)$ and then sampling from the corresponding Gaussian. For some applications such as information-driven exploration missions, it is practical to estimate the mean and variance for this multi-modal model. The mean $\mathbb{E}[h(\mathbf{x}_*)]$ of the mixture model is given by

$$\bar{h}(\mathbf{x}_*) := \mathbb{E}[h(\mathbf{x}_*)] = \sum_{i=1}^{m} P(z(\mathbf{x}_*) = i) \cdot \bar{f}_i(\mathbf{x}_*) \tag{5.7}$$

and the corresponding variance is computed as

$$\mathbb{V}[h(\mathbf{x}_*)] \;=\; \sum_{i=1}^{m} \left[\mathbb{V}[f_i(\mathbf{x}_*)] + (\bar{f}_i(\mathbf{x}_*) - \bar{h}(\mathbf{x}_*))^2\right] \cdot\; P(z(\mathbf{x}_*) = i)\,. \tag{5.8}$$

5.2 Learning the Model from Data

Given a training set $\mathcal{D} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{n}$ of gas concentration measurements y_j and the corresponding sensing locations $\mathbf{x}_j$, the task is to jointly learn the assignment $z(\mathbf{x}_j)$ of data points to mixture components and, given this assignment, the individual regression models $\mathcal{GP}_i$. Tresp [2000] describes an approach based on Expectation Maximization (EM) for solving this task. We take his approach, but also seek to minimize the model complexity to achieve a computationally tractable model even for large training data sets $\mathcal{D}$. This is of major importance in our application, since typical gas concentration data sets easily exceed $n = 1000$ data points and the standard GP model (see Sec. 5.1) is of cubic complexity $\mathcal{O}(n^3)$. Different solutions have been proposed for lowering this upper bound, such as dividing the input space into different regions and solving these problems individually or

by deriving sparse approximations for the whole space. Sparse GPs [Smola and Bartlett, 2000, Snelson and Ghahramani, 2006a] use a reduced set of inputs to approximate the full GP model. This new set can be either a subset of the original inputs [Smola and Bartlett, 2000] or a set of r new *pseudo-inputs* [Snelson and Ghahramani, 2006a] which are obtained using an optimization procedure. This reduces the complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(nr^2)$ with $r \ll n$, which in practice results in a nearly linear complexity. In this section, we describe a greedy forward-selection algorithm integrated into the EM-learning procedure which achieves a sparse mixture model while also maximizing the data likelihood of the whole training set $\mathcal{D}$.

5.2.1 Initializing the Mixture Components

In a first step, we subsample n_1 data points and learn a standard GP for this set. This model $\mathcal{GP}_1$ constitutes the first mixture component. To cover areas of gas concentration that are poorly modeled by this initial model, we learn an "error model", termed $\mathcal{GP}_\Delta$, that captures the absolute differences between a set of target values and the predictions of $\mathcal{GP}_1$. We then sample points according to $\mathcal{GP}_\Delta$ and use them as initialization for the next mixture component. In this way, the new mixture is initialized with the data points that are poorly approximated by the first one. This process is repeated until the desired number of model components is reached. For typical gas modeling scenarios, we found that two mixture components are sufficient to achieve good results. In our experiments, the converged mixture models nicely reflect the bimodal nature of gas distributions, having one smooth "background" component and a layer of locally concentrated structures.

5.2.2 Iterative Learning via Expectation-Maximization

The Expectation Maximization (EM) algorithm can be used to obtain a maximum likelihood estimate when hidden and observable variables need to be estimated. It consists of two steps, the so-called estimation (E) step and the maximization (M) step which are executed alternately.

In the E-step, we estimate the probability $P(z(\mathbf{x}_j) = i)$ that data point j corresponds to model i. This is done by computing the marginal likelihood of each data point for all models individually. Thus, the new $P(z(\mathbf{x}_j) = i)$ is computed given the previous estimate as

$$P(z(\mathbf{x}_j) = i) \;\leftarrow\; \frac{P(z(\mathbf{x}_j) = i) \cdot \mathcal{N}_i(y_j; \mathbf{x}_j)}{\sum_{k=1}^{m} P(z(\mathbf{x}_j) = k) \cdot \mathcal{N}_k(y_j; \mathbf{x}_j)} \;. \tag{5.9}$$

In the M-step, we update the components of our mixture model. This is achieved by inte-

grating the probability that a data point belongs to a model component into the individual GP learning steps (see also [Tresp, 2000]). This is achieved by modifying Eq. (5.4) to

$$\bar{f}_i(\mathbf{X}_*) \quad = \quad k(\mathbf{X}_*, \mathbf{X}) \left[k(\mathbf{X}, \mathbf{X}) + \boldsymbol{\Psi}^i \right]^{-1} \mathbf{y} \,, \tag{5.10}$$

where $\boldsymbol{\Psi}^i$ is a matrix with

$$[\boldsymbol{\Psi}^i]_{jj} \quad = \quad \frac{\sigma_n^2}{P(z(\mathbf{x}_j) = i)} \tag{5.11}$$

and zeros in the off-diagonal elements. Equation (5.5) is updated accordingly. The matrix $\boldsymbol{\Psi}^i$ allows us to consider the probabilities that the individual inputs belong to the corresponding components. Figuratively speaking, the contribution of an unlikely data point to a model is reduced by increasing the data point specific noise term. If the assignment probability, on the other hand, is one, only σ_n^2 remains and the point is fully included as in the standard GP model.

Learning a GP model also involves the estimation of its hyperparameters $\boldsymbol{\theta} = \{\sigma_f, \ell, \sigma_n\}$. To estimate them for $\mathcal{GP}_i$, we first apply a variant of the hyperparameter heuristic used by Snelson and Ghahramani [2006a] in their open-source implementation. We extended it to incorporate the correspondence probability $P(z(\mathbf{x}_k) = i)$ into this initial guess

$$\ell \quad \leftarrow \quad \max_{\mathbf{x}_j} \; P(z(\mathbf{x}_j) = i) \; ||\mathbf{x}_j - \bar{\mathbf{x}}|| \tag{5.12}$$

$$\sigma_f^2 \quad \leftarrow \quad \frac{\sum_{j=1}^n P(z(\mathbf{x}_j) = i) \; (y_j - \mathbb{E}[y])^2}{\sum_{j=1}^n P(z(\mathbf{x}_j) = i)} \tag{5.13}$$

$$\sigma_n^2 \quad \leftarrow \quad 0.25 \cdot \sigma_f^2 \,, \tag{5.14}$$

where $\bar{\mathbf{x}}$ refers to the weighted mean of the inputs, each $\mathbf{x}_j$ having a weight of $P(z(\mathbf{x}_j) = i)$. To optimize the hyperparameters further given this initial estimate, one could apply, for example, Rasmussen's conjugate-gradient–based approach [Rasmussen, 2006] to minimize the negative log marginal likelihood. In our experiments, however, this approach lead to serious overfitting and we therefore resorted to cross validation-based optimization. Concretely, we randomly sample the hyperparameters and evaluate the model accuracy according to Sec. 5.2.2 on a separate validation set. As a sampling strategy, we draw in each even iteration new parameters from an uninformed prior and in each odd iteration, we improve the current best parameters $\boldsymbol{\theta}'$ by sampling from a Gaussian with mean $\boldsymbol{\theta}'$. The standard deviation of that Gaussian is decreased with the iteration number.

5.2.3 Learning the Gating Function

In our mixture model, the gating function defines for each data point the likelihood of being assigned to the individual mixture components. The EM algorithm learns the assignment probabilities for all training inputs $\mathbf{x}_j$, maximizing the overall data likelihood. To generalize these assignments to the whole input space (to form a proper gating *function*), we place another GP prior on the gating variables. Concretely, we learn a gating GP for each component i that uses the $\mathbf{x}_j$ as inputs and the $z(\mathbf{x}_j)$ obtained from the EM algorithm as targets. Let $\bar{f}_i^z(\mathbf{x})$ be the prediction of z for $\mathcal{GP}_i$. Given this set of m GPs, we can compute the correspondence probability for a new test point $\mathbf{x}_*$ as

$$P(z(\mathbf{x}_*) = i) \quad = \quad \frac{\exp(\bar{f}_i^z(\mathbf{x}_*))}{\sum_{j=1}^{m} \exp(\bar{f}_j^z(\mathbf{x}_*))} \ . \tag{5.15}$$

Illustrating Example

To visualize our approach, we now give a simple, one-dimensional example. The left diagram of Fig. 5.2 shows simulated data points, of which most were sampled uniformly from the interval $[2, 2.5]$ and some are distributed with a larger spread at two distinct locations. The same diagram also shows a standard GP model learned on this set, which is not able to explain the data well. The right diagram of the figure shows $\mathcal{GP}_\Delta$, i.e. the resulting error model, which characterizes the local deviations of the model predictions from the data points. Based on this model, a second mixture component is initialized and used as input to the EM algorithm.

The individual diagrams in Fig. 5.3 illustrate the iterations of the EM algorithm (to be read from left to right and from top to bottom). They depict the two components of the mixture model. The learned gating function after convergence of the algorithm is depicted in the left diagram of Fig. 5.4. The right diagram in the same figure gives the final GP mixture model. It is clearly visible that the mixture model better represents this data set than the standard GP model, which assumes a smooth, uni-modal process (see the left diagram of Fig. 5.2). As a quantitative measure of model quality, we noted the average negative log likelihood of the data points as -1.70 for the mixture model vs. -0.24 for the standard GP.

5.3 Experimental Results

We carried out pollution monitoring experiments in a real-world setting, in which a mobile robot followed a predefined sweeping trajectory covering the area of interest. Along its

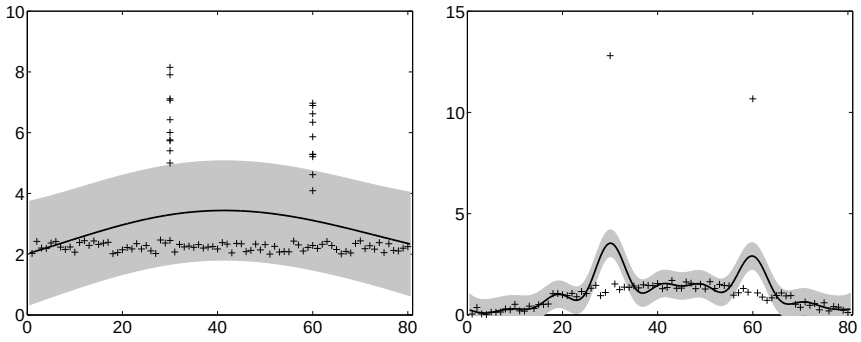


Figure 5.2: Left: The standard GP used to initialize the first mixture component. Right: The error GP used to initialize the next mixture component.

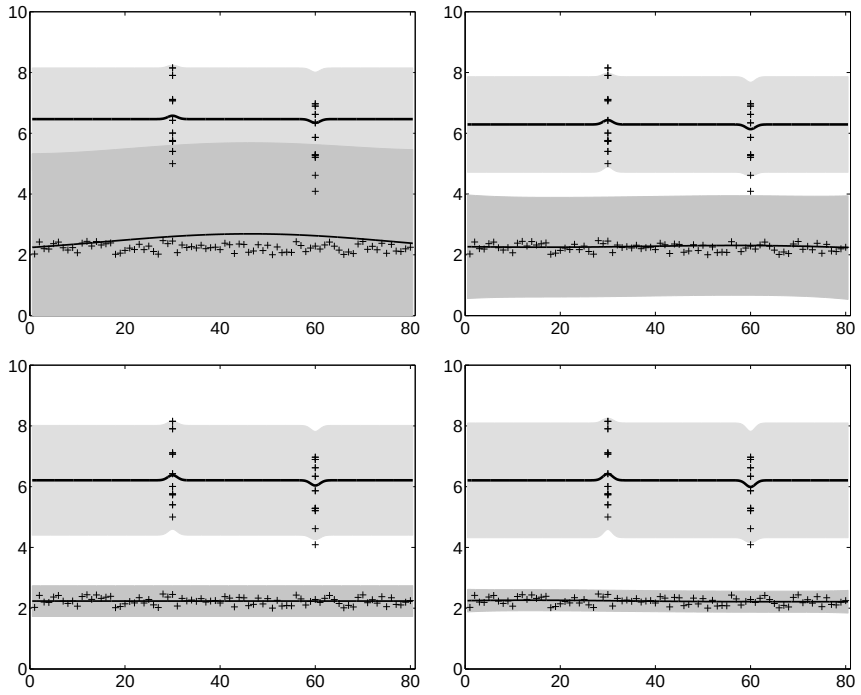


Figure 5.3: Components during different iterations of the EM algorithm.

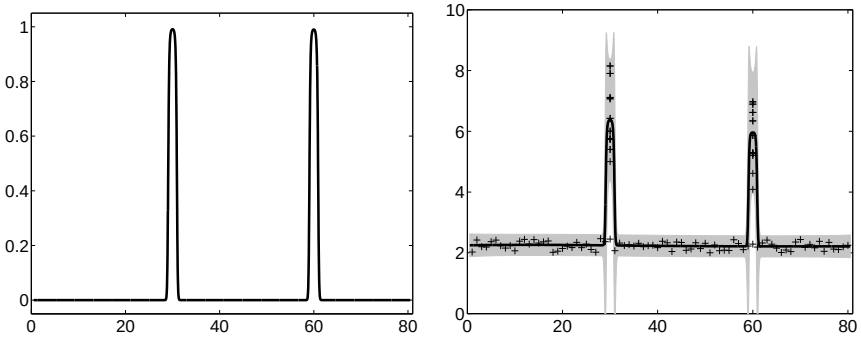


Figure 5.4: Left: The learned gating function. Right: Resulting distribution of the GP mixture model.

path, the robot was stopped for several seconds, 10 s (outdoors) and 30 s (indoors), at predefined points on a regular grid to acquire measurements. The spacing between the grid points was set to values between 0.5 m to 2.0 m depending on the topology of the available space. The sweeping motion was performed twice in opposite directions, and the robot was driven at a maximum speed of 5 cm/s in between the stops. The gas source was a small cup filled with ethanol.

The robot was equipped with a SICK laser range scanner used for pose correction, with an electronic nose, and an anemometer. The electronic nose comprises six Figaro gas sensors (2 $\times$ TGS 2600, TGS 2602, TGS 2611, TGS 2620, TGS 4161) enclosed in an aluminum tube. This tube was mounted horizontally at the front side of the robot (see also Fig. 5.5). The electronic nose is actively ventilated through a fan that creates a constant airflow towards the gas sensors. This lowers the effect of external airflow and the movement of the robot on the sensor response.

Note that in this work, we concentrate only on the gas concentration measurements and do not consider the pose uncertainty of the vehicle. One can apply one of the various SLAM systems available to account for the uncertainty in the robot's pose.

Three environments with different properties were selected for the pollution monitoring experiments. The first experiment, termed *3-rooms*, was carried out in an enclosed indoor area that consists of three rooms which are separated by slightly protruding walls in between them. The area covered by the robot is approximately 14 m $\times$ 6 m. There is little exchange of air with the "outer world" in this environment. The gas source was placed in the central room and all three rooms were monitored by the robot. The second location was a part of a *corridor* with open ends and a high ceiling. The area covered by the trajectory of the robot is approximately 14 m $\times$ 2 m. The gas source was placed on the floor in the middle of the investigated corridor segment. Finally, an *outdoor* scenario was considered. Here, the experiments were carried out in an 8 m $\times$ 8 m region that is part of a much bigger open area.

We used the raw sensor readings in all three environments as training sets and applied our approach to learn the gas distribution models. In all three settings, the robot moved

Table 5.1: Average negative log likelihoods of test data points given the different models. All results in this table differ significantly (10 repetitions, $\alpha = 5\%$)

Dataset	GP	GPM	GPM avg
3-rooms	-1.22	-1.54	-1.50
corridor	-0.98	-1.60	-1.58
outdoor	-1.01	-1.77	-1.69

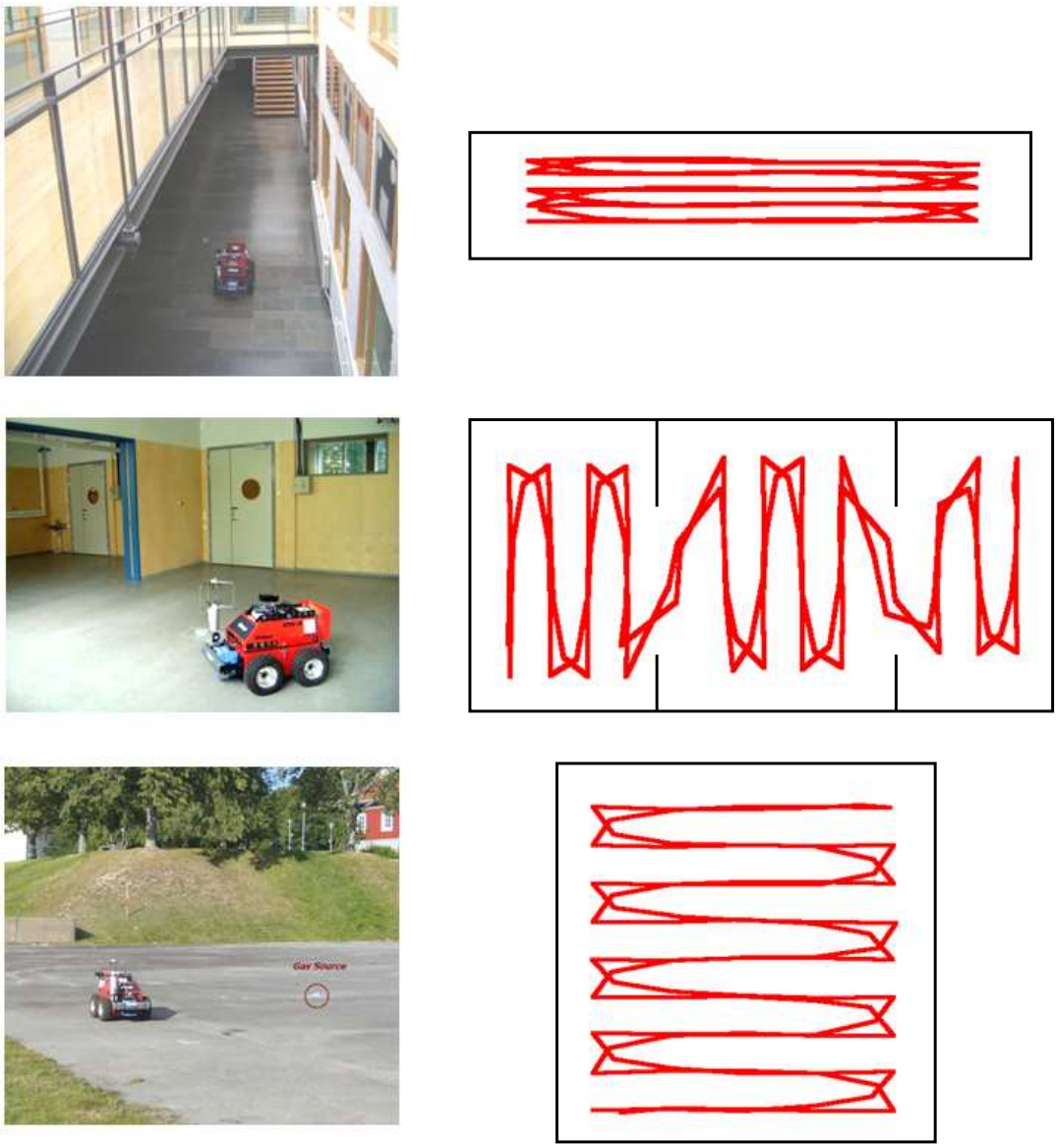
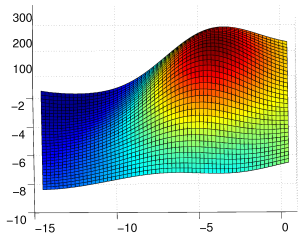
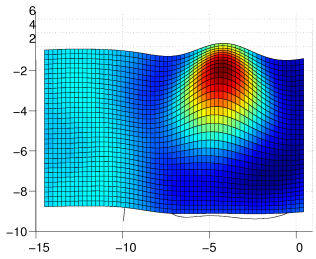


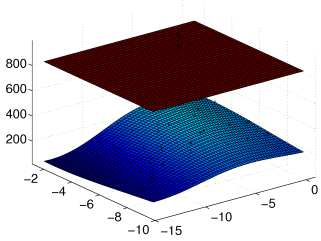
Figure 5.5: Pictures of the robot inspecting three different environments as well as the corresponding sweeping trajectories.



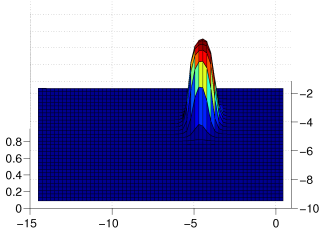
Initial, uni-modal model



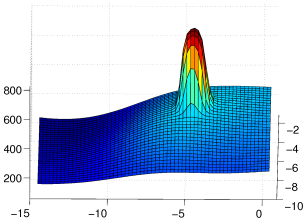
Error model



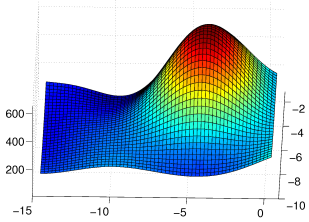
Means of the mixture components



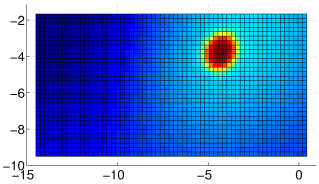
Gating function



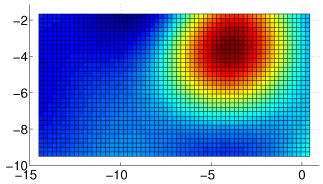
GPM mean (3D view)



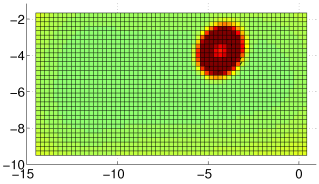
Standard GP mean (3D view)



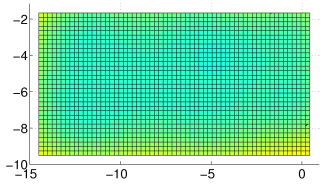
GPM mean (2D view)



Standard GP mean (2D view)



GPM Variance (2D view)



Standard GP variance (2D view)

Figure 5.6: The *3-rooms* dataset with one ethanol gas source in the central room. The room structure itself is not visualized here. In all plots, blue represents low, yellow reflect medium, and red refers to high values.

through the environment twice. We used the first run for learning the model and the second one for evaluating it. To benchmark our results, we compare against gas distribution models learned using (a) standard GP regression, (b) a grid-based interpolation approach, and (c) using kernel extrapolation.

Figure 5.6 shows the learned models for the *3-room* dataset. The left plot in the first row illustrates the mean prediction for the standard GP on the subsampled training set that defines the first mixture component. The right diagram depicts the error GP representing the differences between the initial prediction and a set of observations. Based on the error GP, a new mixture component is initialized and the EM algorithm is carried out. The means of the two mixture components after convergence are shown in the left diagram of the seconds row and the learned gating function is visualized in the adjacent diagram on the right. The left diagram in the third row shows the mean prediction of the final mixture model. As can be seen, the model consists of a smooth "background" distribution and a peak of gas concentration—close to the gas source—with a sharp boundary. In contrast to this, the standard GP (right diagram in the third row) learned using the same data is overly smooth for this dataset, especially in proximity to the gas source.

Table 5.1 summarizes the negative log likelihoods of the test data (second part of the dataset, which was not used for training) given our mixture model (GPM) as well as the standard GP model (GP). We provide two likelihoods for our model, the one given in Eq. (5.6) (called "GPM" in the table) and the one computed based on the averaged prediction specified in Eq. (5.7) and Eq. (5.8) (called "GPM avg"). As can be seen, our GPM method outperforms the standard GP model in all three settings. A t-test on 10 repetitions of the experiments revealed that these results are significant ($\alpha = 5\%$). Two reasons for the increased model accuracy of GPM w.r.t. standard GPs can be seen in the 2D plots in the last two rows of Fig. 5.6. First, as already mentioned before, the standard GP overly smoothes the area close to the gas source and, second, its variance estimates around the source are too low (since standard GPs assume a constant noise rate for the whole domain).

Figure 5.7 visualizes the final result for the *corridor* experiment for the GPM model (means of the mixture components in the left diagram and the gating function on the right). The raw dataset from this experiment is plotted in Fig. 5.1. In this experiment, the area of high gas concentration was mapped appropriately also by the standard GP, but again the variance close to the area of high gas concentration was too small. This can seen by comparing the two rows in Fig. 5.8, which shows the standard GP results on top and GPM below. Similar results are also obtained in the *outdoor* dataset. Mean and variance predictions of the standard GP and our model are provided in Fig. 5.9.

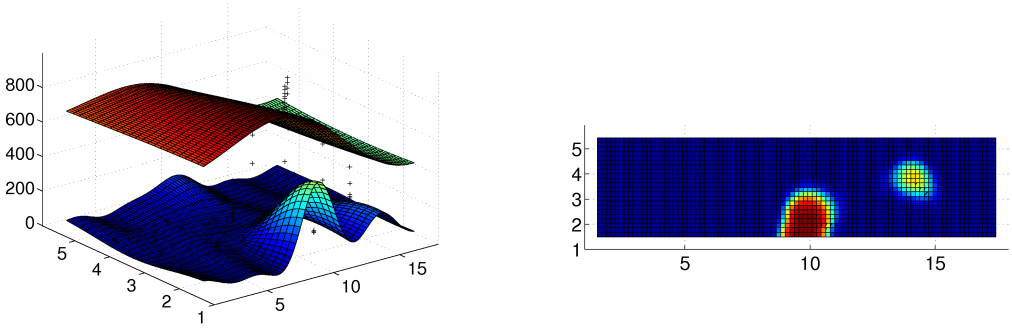
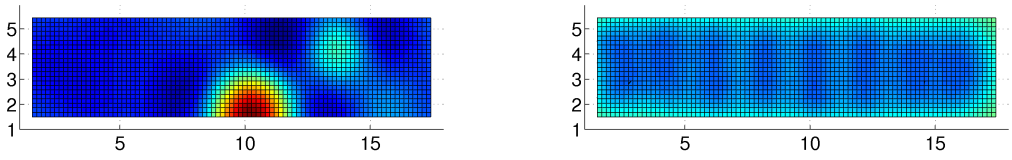
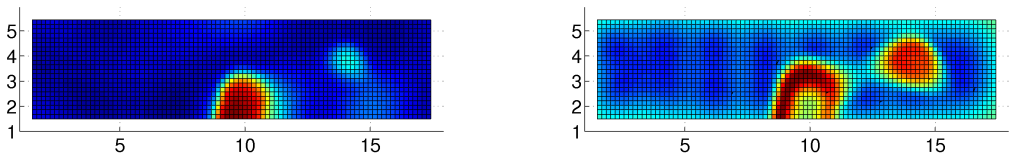


Figure 5.7: GPM results in the *corridor* experiment (see Fig. 5.1 for the raw data). Left: Means of the GPM components. Right: Learned gating function .

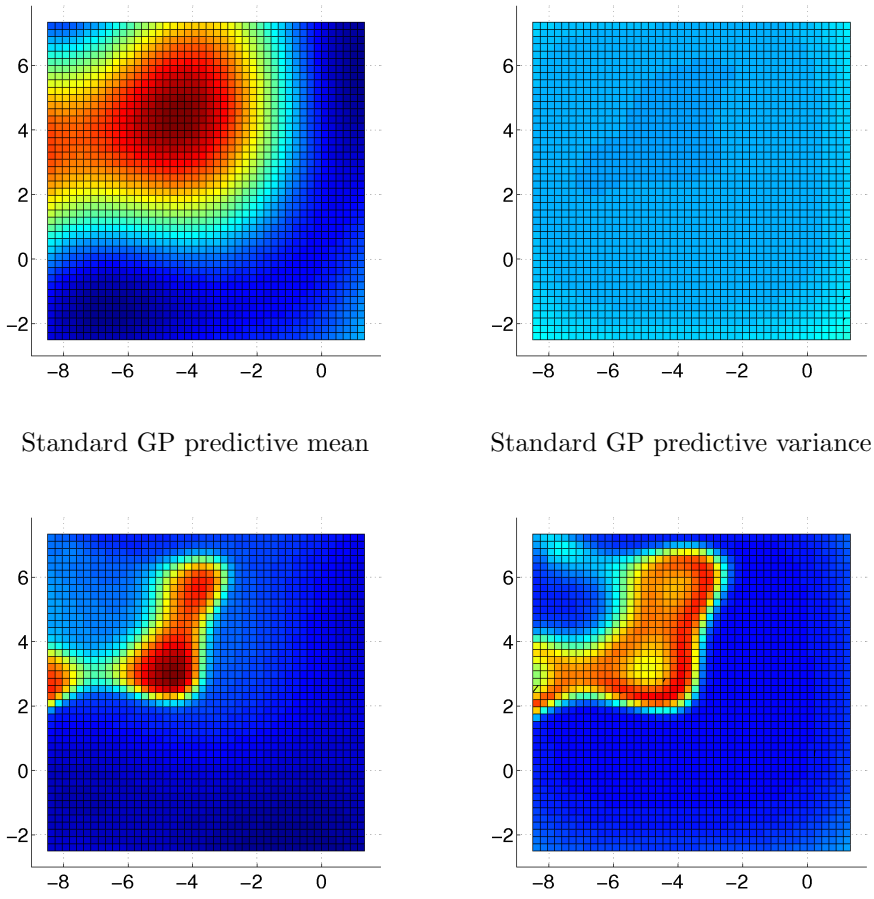


Standard GP predictive mean (left) and variance (right)



GPM predictive mean (left) and variance (right)

Figure 5.8: Models learned from concentration data recorded in the *corridor* environment. The gas source was placed at the location (10, 3). The standard GP and our GPM model provide similar mean estimates. Our approach, however, provides more accurate predictive uncertainties and thus achieves significantly higher data likelihoods (see Tab. 5.1).



Standard GP predictive mean

Standard GP predictive variance

GPM predictive mean

GPM predictive variance

Figure 5.9: Results on the *outdoor* dataset on a 8 m × 8 m area with an ethanol source in the center and under the influence of airflow—approximately from south-east to north-west.

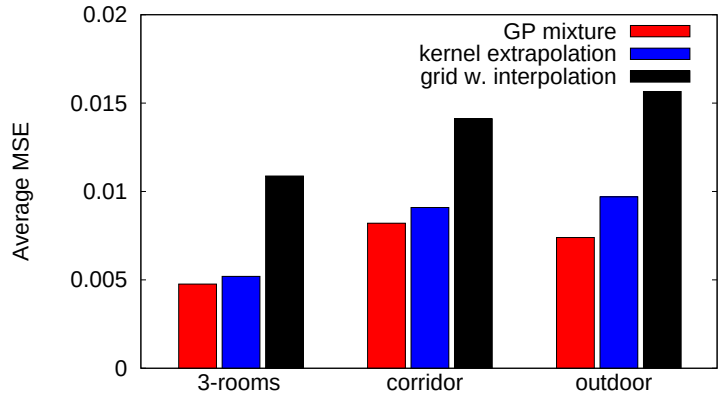


Figure 5.10: Experimental comparison of our GP mixture model with two state-of-the art alternatives in three real-world setting. The bars give the mean squared errors of predicted gas concentration w.r.t. the measured one on a test set.

In all our experiments, we limited the number of data points in the reduced input set to $n_1 = 100$ (taken from the first part of the datasets). The datasets itself contained between 2 500 and 3 500 measurements so our model was able to make accurate predictions with less than 5% of the data. Matrices of that size can be easily inverted and as a result the overall computation time for learning our model including cross validation, running `Matlab` on a standard laptop computer, is around 1 minute for all datasets shown above.

Finally, we compared the mean estimates of our mixture model to the results obtained with the method of Lilienthal and Duckett [2004] as well as with the standard approach of using a grid in combination with linear interpolation. The results of this comparison are shown in Fig. 5.10. As can be seen from the diagram, our method outperforms both alternative methods.

5.4 Related Work

A common approach to creating representations for time-averaged concentration fields is to acquire measurements using a fixed grid of gas sensors over a long period of time. Equidistant gas sensor locations can be used to directly measure and map the average concentration values according to a given grid approximation of the environment. This approach was taken by Ishida *et al.* [1998]—additionally considering partially simultaneous measurements. A similar method was used in [Purnamadjaja and Russell, 2005], but instead of the average concentration, the *peak* concentration observed during a sampling period of 20 s was considered to create the map.

Consecutive measurements with a single sensor and time-averaging over 2 minutes for each sensor location were used by Pyk *et al.* [2006] to create a map of the distribution of

ethanol. Methods, which aim at determining a map of the instantaneous gas distribution from successive concentration measurements, rely on the assumption of a time-constant distribution profile, i.e. on uniform delivery and removal of the gas to be analyzed as well as on stable environmental conditions. Thus, the experiments of Pyk *et al.* were performed in a wind tunnel with a constant airflow and a homogeneous gas source. To make predictions about locations outside of the directly observed regions, the same authors apply bi-cubic interpolation in the case of equidistant measurements and triangle-based cubic filtering in the case, in which the measurement points are not equally distributed. A problem with such interpolation methods is that there is no means of "averaging out" instantaneous response fluctuations at measurement locations. Even if response values are measured close to each other, they will appear independently in the gas distribution map with interpolated values in between. Consequently, interpolation-based maps tend to become more and more jagged the more new measurements are added [Lilienthal *et al.*, 2006].

Histogram-based methods approximate the continuous distribution of gas concentration by means of binning according to regular grids. Hayes *et al.* [2002] for instance suggest using two-dimensional histograms over the number of "odor hits" received in the corresponding area. "Odor hits" are counted whenever the response level of a gas sensor exceeds a defined threshold. In addition to the dependency of the gas distribution map on the selected threshold, a disadvantage of processing binary information only is that useful information contained in the (continuous) sensor readings is discarded. Further disadvantages of histogram-based methods for gas distribution modeling are their dependency on a properly chosen bin size and the lack of generalization across bins or beyond the inspection area.

Gas distribution mapping based on kernel extrapolation can be seen as an extension of the histogram-based approach. The idea was introduced by Lilienthal and Duckett [2004]. In this model, spatial integration is carried out by convolving sensor readings and modeling the information content of the point measurements with a Gaussian kernel. As discussed in [Lilienthal *et al.*, 2006], this method is related to nonparametric estimation using Parzen windows.

Model-based approaches [Ishida *et al.*, 1998] infer the parameters of an analytical gas distribution model from the measurements. They naturally depend on the characteristics of the assumed model. Complex numerical models based on the simulation of fluid dynamics are computationally expensive and require accurate knowledge of the state of the environment (boundary conditions) which are typically not available in practice. Simpler analytical models, on the other hand, often make rather unrealistic model assumptions which hardly fit the real situation. Model-based approaches also rely on well-calibrated

gas sensors and an established understanding of the sensor-environment interaction. The majority of approaches proposed in the literature create a two-dimensional representation and represent time-constant structures in the gas distribution. The complexity of model-free approaches for converging to a stable representation—either in terms of time consumption or the number of sensors—scales quadratically with the size of the environment. To the best of our knowledge, none of the existing approaches jointly models the variance and the time-average concentration values of the field.

In contrast to the above-mentioned approaches, we apply a Gaussian process-based mixture model to the problem of learning probabilistic gas distribution maps. GPs allow us to model the dependencies between measurements by means of a covariance function. They enable us to make predictions at locations not observed so far and do not only provide the mean gas distribution but also the predictive uncertainty. Our mixture model is furthermore able to model sharp boundaries around areas of high gas concentration. Technically, we build on Tresp [2000]'s mixture model of GP experts to better deal with spatially varying properties of the data. Extensions of this technique using infinite mixtures have been proposed by Rasmussen and Ghahramani [2002] and Meeds and Osindero [2006].

5.5 Conclusion

We considered the problem of modeling gas distributions from sensor measurements by means of sparse Gaussian process mixture models. Gaussian processes are an attractive modeling technique in this context since they do not only provide a gas concentration estimate for each point in the space but also the predictive uncertainty. Our approach learns a GP mixture model and simultaneously decreases the model complexity by reducing the training set in order to achieve an efficient representation even for a large number of observations. The mixture model allows us to explicitly distinguish the different components of the spatial gas distribution, namely areas of high gas concentration from the smoothly varying background signal. This improves the accuracy of the gas concentration prediction.

Our method has been implemented and tested using gas sensors mounted on a real robot. With our method, we obtain gas distribution models that better explain the sensor data compared to techniques such as the standard GP regression for gas distribution mapping. Our approach and the one of Lilienthal and Duckett [2004] provide similar mean gas concentration estimates, their approach as well as the majority of techniques in the field, however, lack the ability of also estimating the corresponding predictive uncertainties.

Part II

Modeling Input-Dependent Noise

Chapter 6

Heteroscedastic Gaussian Processes

We relax the assumption of constant observation noise and present a novel Gaussian process model that can adapt its noise rates locally. As experiments show, this model is significantly more accurate than the standard GP model in many real-world learning tasks, while it still retains its main benefits.

The standard Gaussian process model for regression as discussed and applied in the previous chapters constitutes a principled Bayesian framework for learning, model selection, and density estimation while it still requires only relatively basic matrix algebra. An important practical problem, that has been addressed in recent literature, is to relax the assumption of constant noise made in the standard GP model. In many real-world problems, the local noise rates are important features of data distributions that have to be modeled accurately. Consider for example the *Motorcycle* benchmark dataset depicted in Fig. 6.1. While the standard GP regression model quite accurately estimates the mean of the sought after distribution, it clearly overestimates the data variance in some areas and underestimates it in others. In contrast, taking the input-dependent noise into account the variance in the flat regions becomes low. The main contribution of this chapter is a novel GP treatment of input-dependent noise. More precisely, we follow Goldberg *et al.*'s [1998] approach and model the noise variance using a second GP in addition to the GP governing the noise-free output value. In contrast to Goldberg *et al.* , however, we do not apply a time consuming Markov chain Monte Carlo method to approximate the posterior noise variance but replace it with an alternative approximation that seeks to find the most likely local noise rates. This treatment allows us to develop a fast (hard) EM-like procedure for learning both the hidden noise variances and, in contrast to other approaches, also the kernel parameters. Experiments on synthetic and real-world data sets show that our most

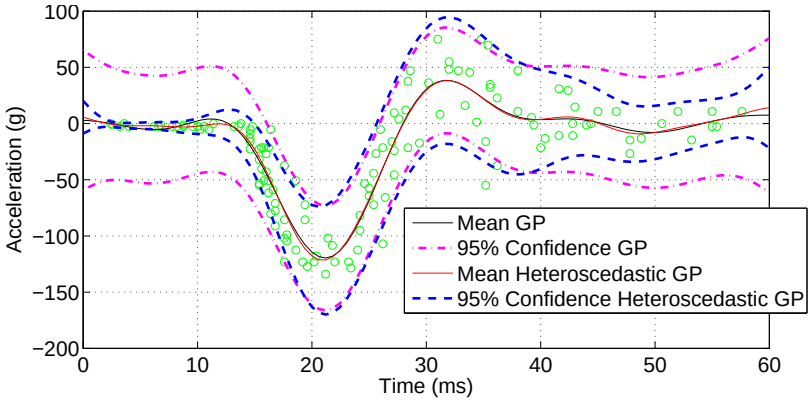


Figure 6.1: is an example for input dependent noise. It consists of a sequence of accelerometer readings through time following a simulated motor-cycle crash.

likely noise approach clearly outperforms standard GP regression and is competitive with existing heteroscedastic regression approaches. At the same time, our approach is substantially less complex than previous ones and has the additional advantage of fully staying within the GP regression framework. Extensions to standard GPs such as online learning, dependent outputs, nonstationary covariance functions, and sparse approximation can be adapted directly. We will exemplify this by combining our model with the *projected process* approximation [Rasmussen and Williams, 2006b], which only represents a small subset of the data for parameter estimation and inference. As our experiments show, this can keep memory consumption low and speed up computations tremendously.

We proceed as follows: After first developing our most likely heteroscedastic GP regression model in Sec. 6.1, we discuss parameter adaptation in Sec. 6.2. Afterwards, we show how to achieve sparse approximations in Sec. 6.2.1 and present the results of an extensive set of experiments in Sec. 6.3.

6.1 The Model

Recalling the definitions in Chapter 2.2, the non-linear regression problem is to recover a functional dependency $y_i = f(\mathbf{x}_i) + \epsilon_i$ from n observed data points $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. Here, $y_i \in \mathbb{R}$ are the (noisy) observed output values at input locations $\mathbf{x}_i \in \mathbb{R}^d$. The task is to learn a model for $p(y_*|\mathbf{x}_*, \mathcal{D})$, i.e., the predictive distribution of new target values y_* indexed by $\mathbf{x}_*$ depending on the observed data set $\mathcal{D}$. If we assume independent, normally distributed noise terms $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$, where the noise variances σ_i^2 are modeled by $\sigma_i^2 = r(\mathbf{x}_i)$, i.e., by a function of $\mathbf{x}$, we get a heteroscedastic regression problem as studied by Goldberg *et al.* [1998], in which the noise rate is not assumed constant on the domain. By placing a Gaussian process prior on f and assuming a noise rate function $r(\mathbf{x})$,

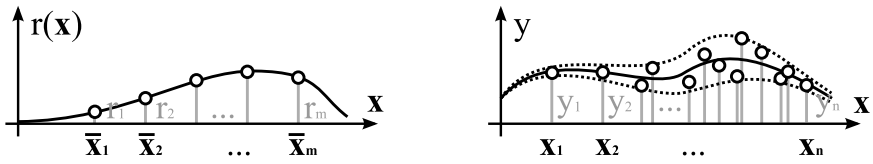


Figure 6.2: By using a GP prior over the latent local noise rates (left), heteroscedastic GP models (right) are able to adapt their predictive variances to the data.

the predictive distribution $P(\mathbf{y}_*|\mathbf{x}_1^*, \ldots, \mathbf{x}_q^*)$ at the query points $\mathbf{x}_1^*, \ldots, \mathbf{x}_q^*$ is a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$ with mean

$$\boldsymbol{\mu}_* = \mathbb{E}[\mathbf{y}_*] = \mathbf{K}_{\mathbf{x}_*\mathbf{x}_*} \left(\mathbf{K}_{\mathbf{xx}} + \mathbf{R}\right)^{-1} \mathbf{y} \tag{6.1}$$

and covariance matrix

$$\boldsymbol{\Sigma}_* = \mathbb{V}[\mathbf{y}_*] = \mathbf{K}_{\mathbf{x}_*\mathbf{x}_*} + \mathbf{R}_* - \mathbf{K}_{\mathbf{x}_*\mathbf{x}} \left(\mathbf{K}_{\mathbf{xx}} + \mathbf{R}\right)^{-1} \mathbf{K}_{\mathbf{x}_*\mathbf{x}}^\top . \tag{6.2}$$

In these equations, we have $\mathbf{K}_{\mathbf{xx}} \in \mathbb{R}^{n\times n}$, $[\mathbf{K}_{\mathbf{xx}}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{K}_{\mathbf{x}_*\mathbf{x}} \in \mathbb{R}^{q\times n}$, $[\mathbf{K}_{\mathbf{x}_*\mathbf{x}}]_{ij} = k(\mathbf{x}_i^*, \mathbf{x}_j)$, $\mathbf{K}_{\mathbf{x}_*\mathbf{x}_*} \in \mathbb{R}^{q\times q}$, $[\mathbf{K}_{\mathbf{x}_*\mathbf{x}_*}]_{ij} = k(\mathbf{x}_i^*, \mathbf{x}_j^*)$, $\mathbf{y} = (y_1, y_2, \ldots, y_n)^\top$, $\mathbf{R} = \mathrm{diag}(\mathbf{r})$ with $\mathbf{r} = (r(\mathbf{x}_1), r(\mathbf{x}_2), \ldots, r(\mathbf{x}_n))^\top$, and $\mathbf{R}_* = \mathrm{diag}(\mathbf{r}_*)$ with $\mathbf{r}_* = (r(\mathbf{x}_1^*), r(\mathbf{x}_2^*), \ldots, r(\mathbf{x}_q^*))^\top$.

An integral part of this model is the covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$ that specifies the covariance $\mathrm{cov}(y_i, y_j)$ of the corresponding targets. In this chapter, we will make use of two common choices, the squared exponential covariance function k_{SE} and the Matérn form k_M (see Chapter 2). These two covariance functions are called *stationary*, since they only depend on the distance $|\mathbf{x}_i - \mathbf{x}_j|$ between input locations $\mathbf{x}$ and $\mathbf{x}'$. In the definitions above, σ_f denotes the amplitude (or signal variance) and ℓ is the characteristic length-scale, see [Rasmussen and Williams, 2006b] for a detailed discussion. These parameters are called hyper-parameters of the process. They are typically denoted as $\boldsymbol{\theta} = (\sigma_f, \ell)$.

Goldberg *et al.* [1998] do not specify a functional form for the noise level $r(\mathbf{x})$ but place a prior over it. More precisely, an independent GP is used to model the logarithms of the noise levels, denoted as $z(\mathbf{x}) = \log(r(\mathbf{x}))$. This z-process is governed by a different covariance function k_z, parameterized by $\boldsymbol{\theta}_z$. We will use the notation $\overline{\cdot}$ to denote variables of the latent log noise process $\mathcal{GP}_z$ to make the distinction to their counterparts in the observed process $\mathcal{GP}_y$ clearly visible. For instance, we denote with $\overline{\mathcal{X}} = \overline{\mathbf{x}}_1, \ldots \overline{\mathbf{x}}_n$ the locations of the "training data points" $\overline{\mathbf{z}} = \{\overline{z}_1, \overline{z}_2, \ldots, \overline{z}_n\}$ for the z-process. $\overline{\mathcal{X}}$ can be chosen in various ways, e.g., by uniformly drawing points from the convex hull of $\mathcal{X}$. In our experiments and here for notational convenience, we set them to coincide with $\mathcal{X}$ of $\mathcal{GP}_y$. Figure 6.2 sketches a prototypical situation with the latent noise process in the left

diagram and $\mathcal{GP}_y$ on the right.

Since the noise rates z_i are now independent latent variables in the combined regression model, the predictive distribution for $\mathbf{y}_*$, i.e., the vector of regressands at query points $\mathcal{X}_* = \{\mathbf{x}_1^*, \ldots, \mathbf{x}_q^*\}$, changes to $P(\mathbf{y}_*|\mathcal{X}_*, \mathcal{D}) =$

$$\int\int p(\mathbf{y}_* \mid \mathcal{X}_*, \mathbf{z}, z_*, \mathcal{D}) \cdot p(\mathbf{z}, z_*|\mathcal{X}_*, \mathcal{D})\, d\mathbf{z}\, dz_* \,. \tag{6.3}$$

Given $(\mathbf{z}, z_*)$, the prediction $p(\mathbf{y}_*|\mathcal{X}_*, \mathbf{z}, z_*, \mathcal{D})$ is Gaussian with mean and variance as defined in Eq. (6.1) and Eq. (6.2). The problematic term is indeed $p(\mathbf{z}, z_*|\mathcal{X}_*, \mathcal{D})$ as it makes the integral difficult to handle analytically. Therefore, Goldberg *et al.* proposed a Monte Carlo approximation. More precisely, given a representative sample $\{(\mathbf{z}_1, z_1^*), \ldots, (\mathbf{z}_k, z_k^*)\}$ of (logarithmic) noise rates, the integral in Eq. (6.3) can be approximated by $\frac{1}{k}\sum_{j=1}^{k} p(y_*|\mathcal{X}_*, \mathbf{z}_j, z_j^*, \mathcal{D})$. The sampling is quite time consuming and the expectation can be approximated by the most likely noise levels $(\tilde{\mathbf{z}}, \tilde{z}_*)$. That is, we approximate the predictive distribution by $p(\mathbf{y}_*|\mathcal{X}_*, \mathcal{D}) \approx p(\mathbf{y}_*|\mathcal{X}_*, \tilde{\mathbf{z}}, \tilde{z}_*, \mathcal{D})$, where $(\tilde{\mathbf{z}}, \tilde{z}_*) = \arg\max_{(\tilde{\mathbf{z}}, \tilde{z}_*)} p(\tilde{\mathbf{z}}, \tilde{z}_*|\mathcal{X}_*, \mathcal{D})$. This will be a good approximation if most of the probability mass of $p(\mathbf{z}, z_*|\mathcal{X}_*, \mathcal{D})$ is concentrated around $(\tilde{\mathbf{z}}, \tilde{z}_*)$. Moreover, computing the most likely noise level and $p(\mathbf{y}_*|\mathcal{X}_*, \mathcal{D})$ now requires only standard GP inference, which is faster than the fully Bayesian treatment.

6.2 Learning Local Noise

So far, we have described our model and how to make predictions assuming that we have the parameters $\boldsymbol{\theta}_z$ of the z-process and the parameters $\boldsymbol{\theta}$ of the noise-free y-process, which uses the predictions of the z-process as noise variances at the given points. In practice, we are unlikely to have these parameters a priori and, instead, we would like to estimate them from data.

The basic observation underlying our approach is very similar to the one underlying the (hard) EM algorithm: Learning would be easy if we knew the noise level values for all the data points. Therefore, we iteratively perform the following steps to find the parameters:

1. Given the input data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, we estimate a standard, homoscedastic $\mathcal{GP}_1$ maximizing the likelihood for predicting y from $\mathbf{x}$.

2. Given G_1, we estimate the empirical noise levels for the training data, i.e., $z_i' = \log(\mathbb{V}[y_i, G_1(\mathbf{x}_i, \mathcal{D})])$, forming a new data set $\mathcal{D}' = \{(\mathbf{x}_1, z_1'), (\mathbf{x}_2, z_2'), \ldots, (\mathbf{x}_n, z_n')\}$.

3. On $\mathcal{D}'$, we estimate a second, also homoscedastic GP_z

4. Now, we estimate the combined, heteroscedastic $\mathcal{GP}_y$ on $\mathcal{D}$ using $\mathcal{GP}_z$ to predict the (logarithmic) noise levels r_i.

5. If not converged, we set $\mathcal{GP}_1 = \mathcal{GP}_y$ and go to step 2.

To summarize the procedure, we take the current noise model and complete the data, i.e., make the noise levels observed. We then fix the completed data cases and use them to compute the maximum likelihood parameters of $\mathcal{GP}_y$. This process is repeated until convergence. Like the hard EM, the algorithm is not guaranteed to improve the likelihood in each step and can start oscillating as it considers most-likely completions of the data only. In our experiments, however, this happened only occasionally and only at reasonably accurate estimates. Step 2, i.e., the empirical estimation of the noise levels is the most crucial step of the procedure: *Given* the data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and the predictive distribution of the current GP estimates, *find* an estimate of the noise levels $\mathbb{V}[y_i, \mathcal{GP}_1(\mathbf{x}_i, \mathcal{D})]$ at each $\mathbf{x}_i$. Consider a sample y_i^j from the predictive distribution induced by the current GP at $\mathbf{x}_i$. Viewing y_i and y_i^j as two independent observations of the same noise-free, unknown target, their arithmetic mean $(y_i - y_i^j)^2/2$ is a natural estimate for the noise level at $\mathbf{x}_i$. Indeed, we can improve the estimate by taking the expectation with respect to the predictive distribution. This yields

$$\mathbb{V}[y_i, \mathcal{GP}_1(\mathbf{x}_i, \mathcal{D})] \approx s^{-1} \sum_{j=1}^{s} \frac{1}{2} \cdot (y_i - y_i^j)^2 \ , \tag{6.4}$$

where s is the sample size and the y_i^j are samples from the predictive distribution induced by the current GP at $\mathbf{x}_i$. This minimizes the average distance between the predictive distribution and the prototype value y_i. For a large enough number of samples ($s > 100$), this will be a good estimate for the noise levels.

Note, that there are several alternatives to the way the latent noise levels of our model are fitted to the data. First, the noise function could be assumed given in an analytical form, in which case the latent noise GP vanishes and $\mathbf{R}$ is constructed analytically instead. As a second alternative, if the training data set is sufficiently dense, the local noise rates can be estimated directly from the data using binning or kernel smoothing. In this case, the local noise estimates directly serve as the "training points" of $\mathcal{GP}_z$. The empirical way is especially robust and easy to implement when multiple target samples y_i are given *for each* input location $\mathbf{x}_i$. In the next chapter, in which we will apply the model to the problem of regularizing range measurement likelihoods for mobile robot localization, we encounter a situation where this naturally occurs. As a last alternative, the latent noise rates used in $\mathcal{GP}_y$ could be fixed to the *mean predictions* of $\mathcal{GP}_z$ instead of integrating over all possible local noise rates in Eq. (6.3). In this case, we get a tractable approximation of Eq. (6.3), for

which we can *jointly* learn $\boldsymbol{\theta}_y$, $\boldsymbol{\theta}_z$, and $\mathbf{z}$, e.g. using gradient-based optimization. We will discuss a similar approach for learning a model dealing with input-dependent smoothness in Chapter 11.

6.2.1 Sparse Approximations

The heteroscedastic regression model presented in the previous section can be combined directly with various extensions of the GP model, like online learning, dependent outputs, nonstationary covariance functions, and sparse approximations. To exemplify this, we discuss how the *projected process* approximation of Rasmussen and Williams [2006b] can be applied to our model to increase its efficiency for large data sets. Section 6.3.3 also gives experimental results for this extension.

Several approximative models have been proposed for GPs in order to deal with the high time and storage requirements for large training data sets. In general, existing approaches select a subset $\mathcal{I}$, $|\mathcal{I}| = m$, of data points (the support set) from the full training set $\mathcal{D}$, $|\mathcal{D}| = n$, to reduce the complexity of learning, model representation, and inference. In contrast to simpler approaches that discard $\mathcal{D} \setminus \mathcal{I}$ completely, the so-called *projected process* (PP) approximation considers a projection of the m-dimensional space of $\mathcal{I}$ up to n dimensions in order to be able to involve all available data points. The key idea is to only represent $m < n$ latent function values, denoted as $\mathbf{f}_m$ with $f_i = f(\mathbf{x}_i), \mathbf{x}_i \in \mathcal{I}$, which leads to smaller matrices that have to be stored and calculated. Then, in the homoscedastic case where a constant noise level σ_n^2 is assumed, the "discarded" points $\mathbf{t}_{n-m}$ from $\mathcal{D} \setminus \mathcal{I}$ are modeled by

$$\mathbf{t}_{n-m} \sim \mathcal{N}(\mathbb{E}[\mathbf{f}_{n-m}|\mathbf{f}_m], \sigma_n^2 \mathbf{I}) \; . \tag{6.5}$$

As detailed in [Rasmussen and Williams, 2006b], this leads to an easy to implement modification of the predictive distribution $p(y_*|\mathbf{x}_*, \mathcal{D})$. For our heteroscedastic model, we replace $\sigma_n^2 \mathbf{I}$ in Eq. (6.5) by the input noise rate matrix $\mathbf{R}$ (as defined in Sec. 6.1), which leads to a straightforward modification of the approximated predictive distribution of the homoscedastic case. An issue not discussed so far is how to select the active set $\mathcal{I}$ from $\mathcal{D}$. While existing approaches make informed selection decisions based on the information gain or on the predictive variance at prospective points in a greedy fashion, we employed a simple random sampling strategy for the experiments reported in Sec. 6.3.3. Due to this, the results reported there can be seen as a lower bound for the performance of our model under the PP approximation. More importantly, it takes only $\mathcal{O}(m^2 n)$ to carry out the necessary matrix computations. For a fixed m, this is linear in n as opposed to $\mathcal{O}(n^3)$ for standard GPs.

6.3 Experimental Results

The goal of our experimental evaluation was to investigate to which extent most likely heteroscedastic GP regression is able to handle input-dependent noise:

(Q1) Is there a gain over standard GP regression?

(Q2) Can it rediscover the hidden noise function?

(Q3) Can it deal with non-smooth noise?

(Q4) Can sparse GP techniques be employed?

We conducted several experiments on benchmark data sets. We implemented our approach in `Matlab` using Rasmussen's and William's GP toolbox [Rasmussen and Williams, 2006b] as well as in `C++`. The benchmark data set experiments were run on a PowerBook G4 using `Matlab 7.2` and a squared exponential covariance function. The parameters were always initialized randomly. As performance measures, we used two different losses. For traditional reasons, we report on the *standardized mean squared error*

$$\text{sMSE} = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \mu_i^*)^2}{\mathbb{V}(y)} \,, \tag{6.6}$$

where μ_i^* is the mean of the estimated predictive distribution $p(y_i|x_i)$ and $\mathbb{V}(y)$ is the empirical variance of the data, which only takes a point prediction into account that minimizes squared errors. A better loss for our task is the average *negative log predictive density*

$$\text{NLPD} = \frac{1}{n}\sum_{i=1}^{n} -\log p(y_i|\mathbf{x}_i) \,, \tag{6.7}$$

which penalizes over-confident predictions as well as under-confident ones.

6.3.1 Benchmark Data Sets

We evaluated most likely heteroscedastic GP regression on the following benchmark data sets known from the literature, which have been used to empirically investigate other heteroscedastic regression methods:

GB: The synthetic data originally used by Goldberg *et al.* [1998]: 100 points x_i were sampled uniformly from the interval $[0, 1]$ and the targets $y_i = 2\sin(2\pi x_i)$ were corrupted with Gaussian noise, which increases its standard deviation linearly from 0.5 at $x = 0$ to 1.5 at $x = 1$.

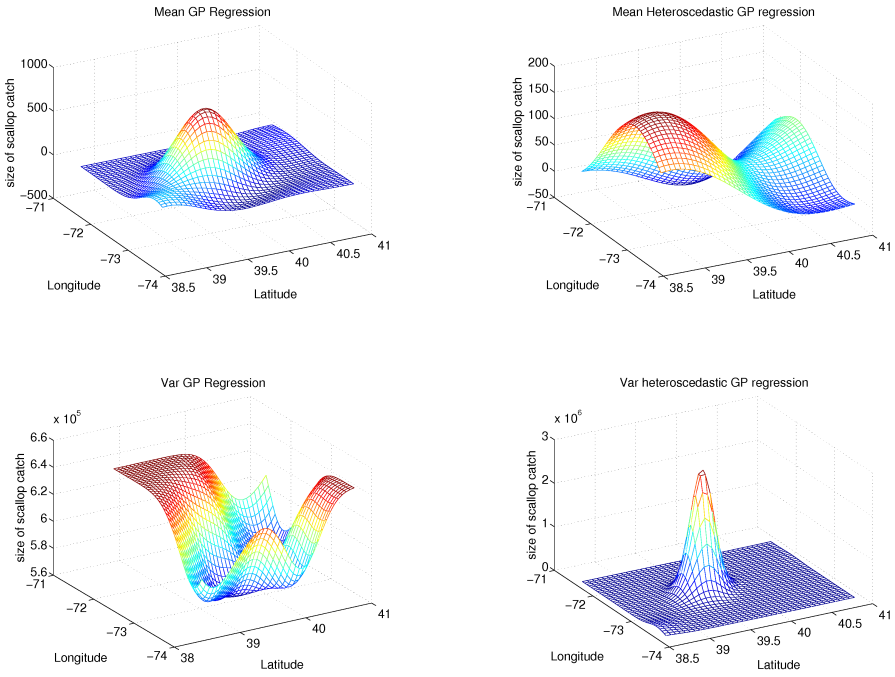


Figure 6.3: Size of scallop catch prediction. Left column: Mean and variance estimate for standard GP regression. Right column: Mean and variance prediction for most likely heteroscedastic GP regression. Note the difference in both the mean and the variance prediction. Standard GP regression is unable to adapt to the increase in noise at the location of higher variance.

YW: The synthetic data originally used by [Yuan and Wahba, 2004]: 200 points x_i were sampled uniformly from $[0, 1]$. The targets were sampled from a Gaussian $y_i \sim \mathcal{N}(\mu(x_i), \exp(g(x_i)))$ with mean $\mu(x_i) = 2[\exp(-30(x_i - 0.25)^2) + \sin(\pi x_i^2)] - 2$ and the logarithm of the standard deviation $g(x_i) = \sin(2\pi x_i)$.

WI: The synthetic data originally used by [Williams, 1996]: 200 input points x_i are drawn from a uniform distribution on $[0, \pi]$. The targets y_i are distributed according to a Gaussian with mean $\sin(2.5x_i) \cdot \sin(1.5x_i)$ and standard deviation $0.01 + 0.25(1 - \sin(2.5x_i))^2$.

LI: The LIDAR data set [Sigrist, 1994] consists of 221 observations from a light detection and ranging experiment. The logarithm of the ratio of received light from two laser sources are given for several distances traveled before the light is reflected back to its source.

For each data set, we performed 10 independent runs. In each run, we randomly split the data into 90% for training and 10% for testing. Table 6.1 summarizes the experimental results on the test sets. As one can see, most likely heteroscedastic GP regression is

always at least as good as GP regression and always significantly improves the estimated predictive distribution. We observed the same when we investigated Ecker and Heltshe's scallop data set [Ecker and Heltshe, 1994]. The set consist of 148 data points concerning scallop abundance and it is based on a 1990 survey cruise in the Atlantic continental shelf off Long Island, New York, USA. The input specifies the location (latitude and longitude) and the target is the size of scallop catch at this location. We performed a 20 times estimate on 129 randomly selected data points for training and tested the model on the remaining 19 points. On average, GP regression achieved an MSE of 1.93 ± 2.0 and an NLPD of 8.16 ± 0.64. Our heteroscedastic GP regression approach achieved an MSE of 1.03 ± 0.16 and an NLPD of 7.73 ± 1.78. The difference in NLPD is significant (t-test, $p = 0.07$), the one in MSE not. To summarize, the results clearly answer (Q1) in an affirmative way.

To investigate (Q2), we ran experiments on all generated data sets, i.e., data sets GB, YW, and WI. In Fig. 6.4 (top and bottom-left), the average standard deviations of the inferred noise levels are given. Notice how in all cases the estimated noise is in close agreement with the data generator. Moreover, they are also in the range of the ones reported in the literature. Thus, our method is competitive with other heteroscedastic regression methods. This is clearly an affirmative answer to (Q2).

To summarize, these results show that our method indeed improves GP regression, that it is able to rediscover the hidden noise function, and that it is competitive with other heteroscedastic regression approaches.

6.3.2 Non-Smooth Noise

Most likely heteroscedastic Gaussian processes assume the noise function to be smooth. Here, we will experimentally investigate (Q3), i.e., to which extent they can handle non-smooth noise functions. To this aim, we followed Cawley *et al.* [2006] and considered the step function on $[-1, 1]$: $f(x) = 1$ if $x > 0$ and 0 otherwise. 100 points uniformly spaced in the interval $[-1, 1]$ have been chosen and the targets have been corrupted with

Table 6.1: Mean test set MSE and NLPD over 10 reruns. In each run, the data set was randomly split into 90% training and 10% test data points. A '•' denotes a significant improvement (t-test, $p = 0.05$) over the corresponding value. Values are rounded to the second digit.

Data set	MSE		NLPD	
	GP	Het GP	GP	Het GP
GB	0.40 ± 0.20	0.40 ± 0.19	1.57 ± 0.31	1.46 ± 0.30 •
YW	0.88 ± 0.19	0.89 ± 0.18	1.66 ± 0.21	1.37 ± 0.26 •
WI	0.49 ± 0.30	0.49 ± 0.30	0.78 ± 0.35	0.35 ± 0.39 •
LI	0.49 ± 0.30	0.49 ± 0.30	0.78 ± 0.36	0.35 ± 0.39 •

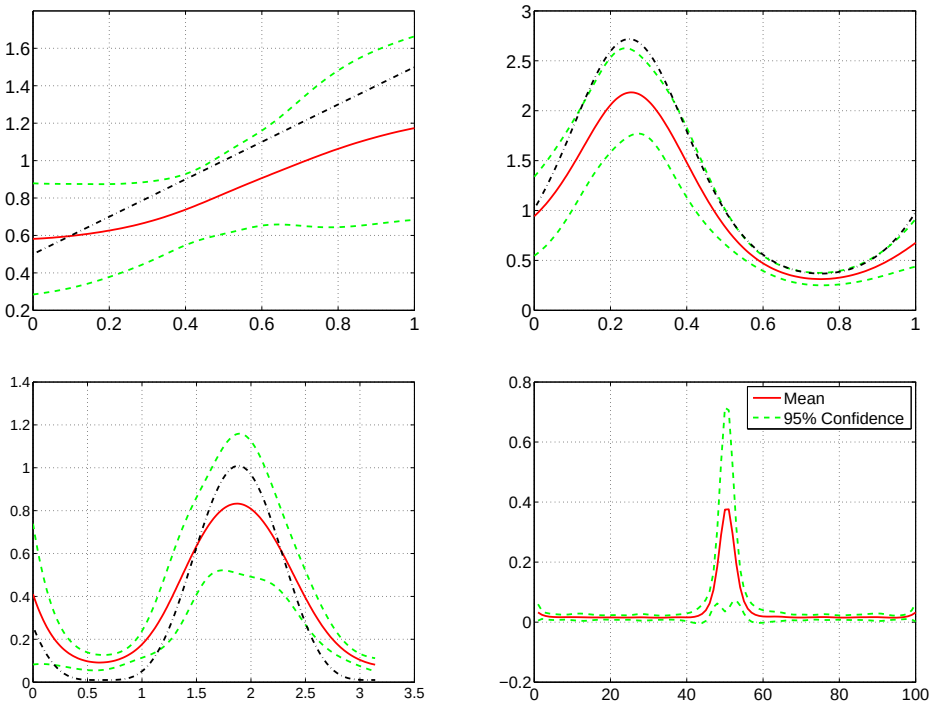


Figure 6.4: Comparison of the estimated hidden noise rates with the true ones. Solid curves give the average estimated standard deviation of the noise and dashed curves the corresponding 2σ confidence intervals. Dashed-dotted curves show the true noises. (Top-left) **GB** data set: 30 runs à 60 samples. (Top-right) **YW** data set: 10 runs, 200 samples. (Bottom-left) **WI** data set: 10 runs, 100 samples. (Bottom-right) Average variance for the step function (20 runs, 100 samples).

a Gaussian noise of standard deviation 0.1. The optimal predictive variance is very large around 0. A standard GP with stationary covariance function is in fact unable to model this. In contrast, the predictive variance of a most likely heteroscedastic GP captures the misfit around 0 well. Figure 6.4 (bottom-right) shows the estimated variance averaged over 20 reruns. The peak is at zero and the average of 0.4 is the same as Cawley *et al.* 's result [Cawley *et al.*, 2006] using "leave-one-out heteroscedastic kernel regression." The non-zero variance in the flat regions is related directly to the noise in the targets. This affirmatively answers (Q3).

6.3.3 Sparse Approximations

In order to investigate (Q4), i.e., sparse approximation techniques within most likely heteroscedastic GP regression, we ran three sets of experiments.

First, we reconsidered the benchmark data sets from Sec. 6.3.1. For the synthetic data sets, we sampled 1000 examples in each run; for the LI data set, we used the original data

set. The data was randomly split into 90% training and 10% test points. 100 random samples of the training set were used as support set. Table 6.2 summarizes the results. As one can see, most likely heteroscedastic GP regression is again always at least as good as GP regression and always significantly improves the estimated predictive distribution.

Second, we investigated the `kin-8nh` data set. This data set was generated synthetically from a realistic simulation of the forward dynamics of an 8 link all-revolute robot arm[1]. The task is to predict the distance of the end-effector from a target. The inputs are 8 features describing quantities like joint positions, twist angles, etc. In total, there are 2000 training examples. We ran our approach 10 times and each time randomly selected a subset of 200 as support set. Standard GP regression achieved an MSE of 0.52 ± 0.03 and an NLPD of -0.23 ± 0.023 on the whole data set; the most likely heteroscedastic GP regression an MSE of 0.49 ± 0.03 and a NLPD of -0.26 ± 0.024. Both differences are significant (t-test, $p = 0.05$).

Third, we considered the Spatial Interpolation Comparison (SIC) 2004 competition. The target variable is ambient radioactivity measured in Germany. More precisely, the data are gamma dose rates reported by means of the national automatic monitoring network[2]. There are two scenarios: the "normal" and the "anomaly", which contains an anomaly in radiation at a specific location. We have focused on the "anomaly" scenario. As Le *et al.* [2005] point out, there is no reason to believe that radioactivity would exhibit highly non-uniform behavior. GP regression, however, is unable to cope with local noise due to the "step-like" anomaly. In contrast, heteroscedastic GP regression should adapt locally to the noise. To investigate this, we performed 10 random estimates using 400 of the 808 given examples as support set. The initial parameters were selected on the "normal" data set. On the complete data set, the standard GP achieved an MSE of 24.72 ± 8.51 and an NLPD of 6.84 ± 3.63, both averaged over the 10 runs. Our heteroscedastic approach achieved an

Table 6.2: Mean test set MSE and NLPD over 10 runs of sparse approximation. A '•' denotes a significant improvement (t-test, $p = 0.05$) over the corresponding value. Values are rounded to the second digit.

Data Set	MSE		NLPD	
	GP	Het GP	GP	Het GP
GB	0.73 ± 0.18	0.73 ± 0.17	2.02 ± 0.12	1.92 ± 0.16 •
YW	0.88 ± 0.05	0.84 ± 0.05 •	1.88 ± 0.14	1.46 ± 0.13 •
WI	0.59 ± 0.09	0.56 ± 0.11	0.90 ± 0.11	0.41 ± 0.18 •
LI	0.08 ± 0.04	0.08 ± 0.03	-1.03 ± 0.33	-1.35 ± 0.32 •

[1] See `http://www.cs.toronto.edu/\~delve`

[2] See `http://www.ai-geostats.org/events/sic2004`

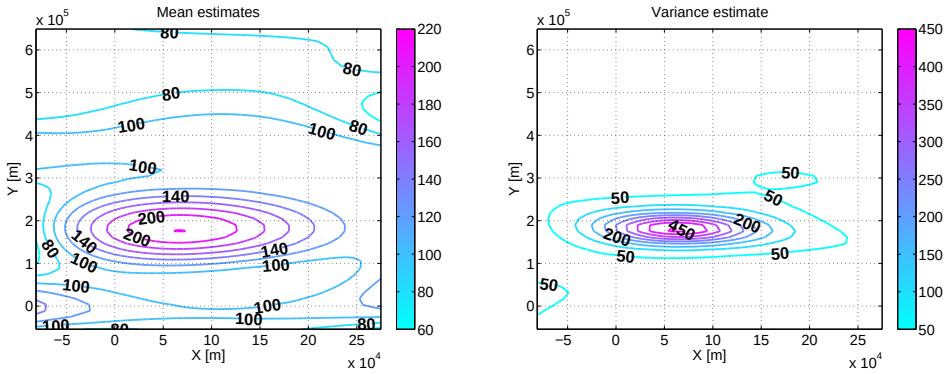


Figure 6.5: Results on data from the Spatial Interpolation Comparison (SIC) 2004. Mean estimates (left) and variance estimates (right) for most likely heteroscedastic GP regression. Note the peak in variance at the location of the outbreak.

MSE of 58.27 ± 29.17 and an NLPD of 4.21 ± 0.25. Thus, the most likely heteroscedastic GP models the predictive distribution significantly better (t-test, $p = 0.05$), but achieve a significantly worse MSE measure (t-test, $p = 0.05$). This is because the outbreak was identified as noise as shown in Fig. 6.5, which depicts a typical radioactivity prediction using our method. Actually, the estimated variance was only high at the location of the outbreak. This contrasts with standard GPs, which cannot adapt to the local noise.

To summarize, the results of all three experiments affirmatively answer (Q4), the SIC experiment also (Q3). Furthermore, they confirmed the drop in runnning time from $\mathcal{O}(n^3)$ for standard GPs to $\mathcal{O}(m^2 n)$ for the projective process approximation.

6.4 Related Work

The non-linear regression problem has been extensively studied in research areas such as machine learning, statistics, or engineering. While many existing approaches to the problem assume constant noise throughout the domain, there is also a growing body of work addressing heteroscedasticity, i.e., varying levels of noise. Schoelkopf *et al.* [2000] present an SVM based algorithm that takes a known variance function into account. Nott [1996] propose a Bayesian model based on penalized splines and give an MCMC algorithm for inferring the posterior. Chan *et al.* [2006] derive a similar model for the Gaussian case, which adapts the noise variances and also requires MCMC for inference. Edakunni *et al.* [2007] presents a mixture of local linear regression models that can be learned using variational Bayesian EM. Opsomer *et al.* [1997] present an iterative procedure for dealing with heteroscedasticity in the context of kriging. They assume a linear model for the mean that is fitted using generalized least squares. Snelson *et al.* [2003] propose a non-linear transformation of the output space to model output-dependent noise variances. Yuan and

Wahba [2004] also jointly estimate the mean and noise variances but do not deal with the problem of selecting the kernel function. Le *et al.* [2005] also estimate the variance non-parametrically along with the mean of the distribution. In contrast to other approaches, they propose a maximum-a-posteriori estimation of the natural parameters in the exponential family. This yields, for the case of given kernel parameters, a convex optimization problem that can be solved efficiently. Recently, Snelson and Ghahramani [2006b] proposed to utilize the dependency of the predictive uncertainty on the density of input data points.

6.5 Conclusion

This chapter has shown that effective Gaussian process regression with input-dependent noise can be fully implemented using standard GP techniques and without having to resort to MCMC approximations.

In experimental tests, most likely heteroscedastic GP regression, the resulting approach, produces estimates that are significantly better than for standard GPs and competitive with other heteroscedastic regression approaches.

Chapter 7

Learning Observation Models for Range Sensors

We apply the heteroscedastic Gaussian process model, which was introduced in the previous chapter, to the problem of mobile robot localization using 2D range sensors. This approach allows to learn in the space of continuous range functions directly rather than being restricted to discrete sets of independent beam measurements. As one distinct practical benefit, this offers a natural and sound way of dealing with highly peaked observation models in Monte Carlo-based estimation approaches.

Acquiring, interpreting, and manipulating information from sensors is one of the fundamental tasks within mobile robotics. For instance, based on models for the robot's kinematics and perception, a robot might be asked to perform tasks such as building a map of the environment, determining its precise location within a map, or navigating to a particular place. In designing robots to operate in the real world, one cannot avoid dealing with the issue of uncertainty. Uncertainty arises from sensor limitations, noise, uncertain actions, and the fact that most complex environments can only be represented and perceived in a limited way. As also discussed in Chapter 3, state-of-the-art approaches build on probabilistic foundations and model the robot's perception as a probability density $p(\mathbf{z}|\mathbf{x})$, where $\mathbf{z}$ is an observation and $\mathbf{x}$ denotes the state of the robot, e.g., its position relative to the environment.

Among the most widely used types of sensors are range sensors such as laser range finders and sonar sensors. The popularity of range sensors in research and industry is due to

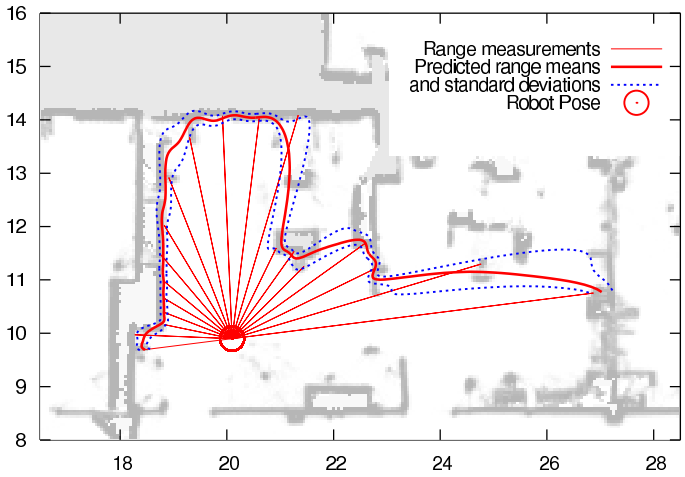


Figure 7.1: The predictive distribution of range measurements for an uncertain robot pose within an office environment (heading angle varies by $\pm 5°$). Scales are given in meters. The straight, red lines depict one possible range scan in this setting.

the fact that spatial information about the environment can be acquired directly and that state-of-the-art sensor hardware is accurate and reliable. Range sensors measure distances r_i to nearby objects along certain directions $\boldsymbol{\alpha}_i$ (possibly multi-dimensional bearing angles) relative to the sensor. Hence, for a vector $\mathbf{r} = (r_1, \ldots, r_m)$ of distance measurements with corresponding bearing angles $\mathcal{A} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m)$ and for a given sensor pose $\mathbf{x}$, the sensor model is given by $p(\mathbf{r}|\mathcal{A}, \mathbf{x})$. Such a model is used, for example, to update a map of the environment according to an newly acquired range scan $\mathbf{z} = \langle (r_i, \boldsymbol{\alpha}_i) \rangle_{i=1}^m$ or to evaluate the likelihood of a range measurement for localization purposes.

In this chapter, we propose a novel, generative model for $p(\mathbf{r}|\mathcal{A}, \mathbf{x})$, termed *Gaussian beam processes* (GBP), which takes a nonparametric Bayesian regression view on the measurement modeling task. We treat the measurements $\mathbf{z} = \langle (r_i, \boldsymbol{\alpha}_i) \rangle_{i=1}^m$ as sets of samples from a stochastic process $p(r|\boldsymbol{\alpha})$ and assume the process to be a Gaussian process [Williams and Rasmussen, 1995], i.e., any finite collection of random variables r_j is assumed to have a joint Gaussian distribution. Learning in this framework means recording or simulating a training set of range scans and adjusting a predefined covariance function accordingly. This can be done online, while the robot is operating, or offline. We put special emphasis on the application of GBPs to mobile robot localization, but nevertheless present the model in a general form that should also be useful in many other applications for range sensor.

A major benefit of our model in localization tasks is that it naturally allows to estimate the distribution $p(\mathbf{r}|\mathcal{A}, \mathcal{U}_\mathbf{x})$ of range measurements for a whole *region* $\mathcal{U}_\mathbf{x}$ of the pose space. As we will show, this property can be used to deal with highly peaked observation likelihood

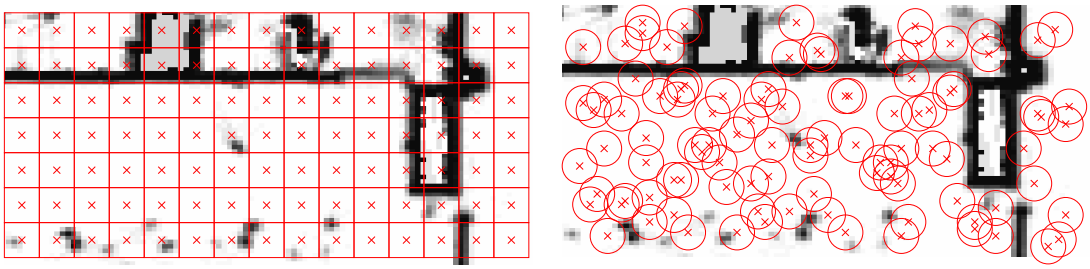


Figure 7.2: The posterior distribution of robot poses is typically approximated by discretizing the pose space using a regular grid (left) or by sampling it (right).

functions—which are a well-known practical problem in mobile robot localization and which typically occur when using precise sensors such as laser range finders.

As an example, consider Fig. 7.1. It shows the predictive distribution of range measurements for a mobile robot with an uncertain heading angle, i.e., its location is fixed, the orientation angle, however, is known only up to $\pm 5°$. It can be clearly seen from the visualized standard deviations of the range predictions that the model accurately identifies the distances that can be predicted with high confidence despite the angular uncertainty in the sensor pose. The ability to learn and represent such distributions is of particular value in applications in which the posterior is approximated using a discrete set of pose hypotheses. In histogram filtering, for example, the pose space is partitioned into a finite set of grid cells (see the left diagram in Fig. 7.2). With the GBP model, we can estimate the observation likelihood $p(\mathbf{z}|\mathcal{U}_\mathbf{x})$ for a whole grid cell $\mathcal{U}_\mathbf{x}$ (centered at $\mathbf{x}$) directly rather than having to numerically approximate $\frac{1}{|\mathcal{U}_\mathbf{x}|} \int_{\mathcal{U}_\mathbf{x}} p(\mathbf{z}|\mathbf{x})\ d\mathbf{x}$ using point estimates of the likelihood function. This ability is also useful for particle filtering in which the posterior is represented by a finite set of weighted samples (see the right diagram in Fig. 7.2). It is a well-known fact that highly peaked likelihood functions have to be regularized in practical applications, because the number of particles is limited and, thus, the probability of sampling close enough to the high-likelihood peaks can be extremely low.

Furthermore, the Gaussian process treatment offers the following distinct benefits:

- The model is fully predictive as it is able to predict ranges at arbitrary bearing angles, i.e., also for angles in between two beams of an actual scan and for beams that have been classified as erroneous. For all predictions, the model also yields the predictive uncertainties.
- Neither the number of range measurements per scan nor their bearing angles have to be fixed beforehand.
- By representing correlations between adjacent beams using parameterized covari-

ance functions, only few recorded or simulated range scans are required to learn an accurate model.

- Gaussian processes are mathematically well-established. There exists a large pool of methods for learning, likelihood evaluation, and prediction.

The chapter is organized as follows. In Sec. 7.1, we discuss the problem of highly peaked observation models and possible remedies. Section 7.2 presents our Gaussian beam process (GBP) model. In Sec. 7.3, we describe its application to Monte Carlo localization and present the results of extensive evaluations on real world and synthetic data.

7.1 Regularizing Observation Likelihoods

It was already noted in [Gordon *et al.*, 1993]—one of the earliest references to the particle filtering principle—that low observation noise variances can lead to serious practical problems. Range sensors, for instance, have extremely small noise rates and provide a large amount of beam measurements per scan $\mathbf{z}$. Thus, already small deviations of a sampled sensor pose $\mathbf{x}^{[i]}$ from the true pose $\mathbf{x}$ can lead to a significant drop of the observation likelihood, $p(\mathbf{z}|\mathbf{x}^{[i]}) \ll p(\mathbf{z}|\mathbf{x})$. In such cases, either

- the state space has to be sampled very densely to ensure that hypotheses are created close enough to the true state or
- the observation likelihoods have to be regularized appropriately.

The first solution often is intractable for computational reasons or because appropriately focusing proposal distributions are not available (see Chapter 3). For deriving the second solution, consider the right diagram in Fig. 7.2. We would like to compute the weight $w^{[i]}$ of each state sample $\mathbf{x}^{[i]}$ based on a local neighborhood $\mathcal{U}_{\mathbf{x}^{[i]}}$ rather than on its center point only:

$$w^{[i]} \propto p(\mathbf{z} \mid \mathcal{U}_{\mathbf{x}^{[i]}}) = \int_{\mathcal{U}_{\mathbf{x}^{[i]}}} p(\tilde{\mathbf{x}})\, p(\mathbf{z} \mid \tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \;. \tag{7.1}$$

Here, $p(\tilde{\mathbf{x}})$ characterizes how the neighboring states of $\mathbf{x}^{[i]}$ should influence the regularized likelihood estimate. It can be set, e.g., to a uniform distribution over a circular region $\mathcal{U}_{\mathbf{x}^{[i]}}$ or to a Gaussian centered at $\mathbf{x}^{[i]}$. This way, we can account explicitly for the sampling density of the filter in the regularization step by adapting the extent of the local neighborhoods accordingly. We have conducted simulation studies (see Sec. 7.3.1) to confirm that this regularization procedure significantly reduces the KL-divergence of the posterior distribution from the ground truth. In effect, this reduces the risk of "missing" the peaks of the true posterior and, thus, increases the robustness of the approximation.

The hard task indeed is to represent and estimate the typically complex distribution of observations $p(\mathbf{z}|\mathcal{U}_\mathbf{x})$ and to perform density estimation for it. In our range sensing application, for instance, Eq. (7.1) contains an integral over high-dimensional range scans $\mathbf{z}$, which causes the individual components z_i to become statistically dependent. To capture this dependency, we model the joint distribution of range measurements

$$p(\mathbf{z} \mid \mathcal{U}_\mathbf{x}) = p(r_1, \ldots, r_m \mid \mathcal{A}, \mathcal{U}_\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{7.2}$$

by an m-dimensional Gaussian with $\boldsymbol{\mu} \in \mathbb{R}^m$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times m}$. Note that this is a generalization of existing beam-based models that assume independent, normally distributed r_i, which corresponds to setting $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_n^2)$ with a constant, real-valued measurement noise parameter σ_n^2. By also taking the covariances off the diagonal into account and by estimating these parameters depending on the locations $\mathbf{x}$, we achieve a less biased likelihood model that also takes the dependencies between beams into account.

We seek to estimate the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ from a set of L given range scans $\{\tilde{\mathbf{z}}_1, \ldots, \tilde{\mathbf{z}}_L\}$, e.g., simulated at poses $\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_L\}$ sampled from $\mathcal{U}_\mathbf{x}$. This can either be done directly using

$$\boldsymbol{\mu} = \frac{1}{L} \sum_{i=1}^{L} \tilde{\mathbf{z}}_i \tag{7.3}$$

$$\boldsymbol{\Sigma} = \frac{1}{L} \sum_{i=1}^{L} (\tilde{\mathbf{z}}_i - \boldsymbol{\mu}) (\tilde{\mathbf{z}}_i - \boldsymbol{\mu})^\top \tag{7.4}$$

or by regarding the $\tilde{\mathbf{z}}_i$ as samples from a continuous range function $\boldsymbol{\alpha} \mapsto r$ that maps the orientations of the individual beams to the corresponding range measurements. In the experimental evaluation, we will refer to this approach as *Estimated Covariance* (EC). If we place a Gaussian process prior on this function and apply the heteroscedastic extension introduced in the previous chapter, we are able to estimate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ reasonably accurate from a number L of scans that is only about 5% of the one required for direct estimation according to Eq. (7.3) and (7.4). This approach, which we term *Gaussian beam processes* (GBP) is introduced in the following section.

7.2 Gaussian Beam Processes

We pose the task of estimating $p(\mathbf{r}|\mathcal{A})$ as a regression problem and model the function that maps beam orientations $\boldsymbol{\alpha}$ to range measurements r as a stochastic process. In other words, we regard the individual measurements r_i as a collection of random variables indexed by the respective beam orientations $\boldsymbol{\alpha}_i$. By placing a Gaussian process prior over

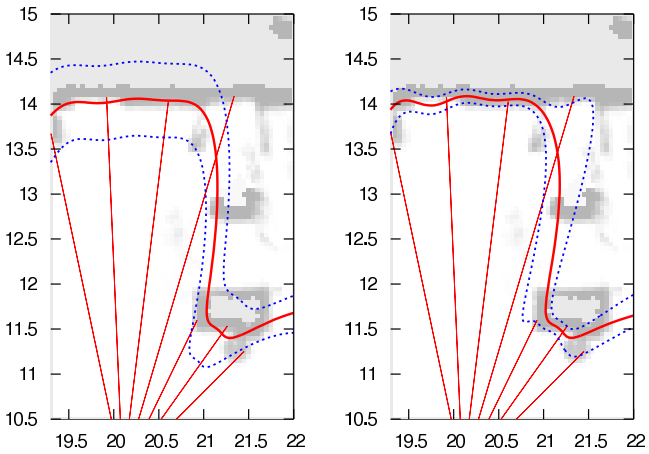


Figure 7.3: The effect of modeling non-constant noise on a data set of range measurements simulated for the case of an uncertain sensor orientation ($\pm 5°$). Standard Gaussian process regression (left) assumes constant noise for all bearing angles. Modeling heteroscedasticity (our model, on the right) yields lower predictive uncertainties at places with low expected noise levels such as the wall in front. The straight, red lines depict one possible range scan in this setting.

this function, we get an easy-to-implement yet powerful model for likelihood estimation for range measurements as well as for prediction. In the following, we will derive the general model for d-dimensional angular indices $\boldsymbol{\alpha}_i$ (e.g., $d = 1$ for planar sensing devices, $d = 2$ for 3D sensors). Also note that we leave out dependencies on the sensor pose $\mathbf{x}$ in this section to simplify notation.

Given a training set $\mathcal{D}$ of range and bearing samples, we want to learn a model for the non-linear and noisy functional dependency $r_i = f(\boldsymbol{\alpha}_i) + \epsilon_i$ with independent, normally distributed error terms ϵ_i. For this task, we apply the heteroscedastic Gaussian process model introduced in the previous chapter. Thus, we model all range samples r_i as jointly Gaussian distributed $p(r_1, \ldots, r_n | \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ with a mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{K}$.

The mean $\boldsymbol{\mu}$ is typically assumed $\mathbf{0}$ and $\mathbf{K}$ is defined by $k_{ij} := k(\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j) + \sigma_n^2 \delta_{ij}$, depending on a covariance function k and the global noise variance parameter σ_n. The covariance function represents the prior knowledge about the underlying function f and does not depend on the target values $\mathbf{r}$ of $\mathcal{D}$. In this work, we make use of the often-used squared exponential and the Matérn form (see Sec. 2.2). Both covariance functions are called *stationary*, since they only depend on the distance Δ_{ij} between input locations $\boldsymbol{\alpha}_i$ and $\boldsymbol{\alpha}_j$. The parameters of the covariance functions are called hyperparameters of the process. They are typically denoted as $\boldsymbol{\Theta} = (\sigma_f, \ell, \sigma_n)$ (e.g., for the squared exponential).

As discussed in the previous chapter, standard Gaussian processes assume a constant noise term, i.e., identically distributed error terms ϵ_i over the domain. For modeling range sensor

measurements, however, we have two sources of variance in beam lengths, i.e.,

- measurement noise, which is relatively small for state-of-the-art sensors such as laser range finders and
- inter-beam correlations due to an uncertain sensor pose (see Fig. 7.1). This situation arises, for instance, if one applies the likelihood regularization scheme introduced in the previous section.

Especially in the latter case, the variance of range values in each beam direction is, along with its mean value, an important feature of the sought-after distribution of range measurements. In order to account for it, we apply the heteroscedastic GP regression framework introduced in the previous chapter rather than standard GPs. Figure 7.3 illustrates the effect of this treatment on the predictive distribution for range values. The left diagram depicts the standard procedure that assumes a constant noise term for all bearings $\boldsymbol{\alpha}$. Our heteroscedastic treatment, depicted in the right diagram, achieves a significantly better fit to the data set while still not over-fitting to the individual samples.

As in Chapter 6, we follow the approach of Goldberg *et al.* [1998] and condition a standard Gaussian processes $\mathcal{G}_r$ for the range function on latent noise variables sampled from a separate *noise* process $\mathcal{G}_n$. Let $\mathbf{v} \in \mathbb{R}^n$ be such noise variances at the n given data points and $\mathbf{v}_* \in \mathbb{R}^m$ those for the m locations to be predicted, then the predictive distribution for a new range value $\mathbf{r}_*$ at angular index $\mathcal{A}_*$ changes to

$$\boldsymbol{\mu}_* = \mathbf{K}_{\mathbf{x}_*\mathbf{x}_*} \left(\mathbf{K}_{\mathbf{xx}} + \mathbf{R}\right)^{-1} \mathbf{r} \,, \tag{7.5}$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{\mathbf{x}_*\mathbf{x}_*} + \mathbf{R}_* - \mathbf{K}_{\mathbf{x}_*\mathbf{x}} \left(\mathbf{K}_{\mathbf{xx}} + \mathbf{R}\right)^{-1} \mathbf{K}_{\mathbf{x}_*\mathbf{x}}^\top \,, \tag{7.6}$$

where $\mathbf{R} = \mathrm{diag}(\mathbf{v})$ and $\mathbf{R}_* = \mathrm{diag}(\mathbf{v}_*)$. Now, as the noise variances $\mathbf{v}$ and $\mathbf{v}_*$ cannot be known a-priori, they have to be integrated over,

$$p(\mathbf{r}_*|\mathcal{A}_*, \mathcal{D}) = \int \underbrace{p(\mathrm{r}_*|\mathcal{A}_*, \mathbf{v}, \mathbf{v}_*, \mathcal{D})}_{p_r} \cdot \underbrace{p(\mathbf{v}, \mathbf{v}_*|\mathcal{A}_*, \mathcal{D})}_{p_v} \, d\mathbf{v} d\mathbf{v}_* \,. \tag{7.7}$$

Given the variances $\mathbf{v}$ and $\mathbf{v}_*$, the prediction p_r in Eq. (7.7) is a Gaussian with mean and variance as discussed above. The problematic term is indeed p_v as it makes the integral difficult to handle analytically. As discussed in detail in the previous chapter, we therefore consider the approximation

$$p(\mathbf{r}_*|\mathcal{A}_*, \mathcal{D}) \approx p(\mathbf{r}_*|\mathcal{A}_*, \tilde{\mathbf{v}}, \tilde{\mathbf{v}}_*, \mathcal{D}) \,, \tag{7.8}$$

with $(\tilde{\mathbf{v}}, \tilde{\mathbf{v}}_*) = \arg\max_{(\tilde{\mathbf{v}}, \tilde{\mathbf{v}}_*)} p(\tilde{\mathbf{v}}, \tilde{\mathbf{v}}_*|\mathcal{A}_*, \mathcal{D})$ and use the following EM-like procedure for

learning the noise levels $(\tilde{\mathbf{v}}, \tilde{\mathbf{v}}_*)$: (1) fix the noise levels and learn $\mathcal{G}_r$ using a standard maximum likelihood estimator; (2) fix $\mathcal{G}_n$, estimate the empirical noise levels of $\mathcal{G}_r$ on the training data and estimate $\mathcal{G}_n$ using them as target data. Initially, the noise levels are set to the empirical noise levels of a constant-noise Gaussian process induced on the training data.

As covariance functions, we use the Matérn type as stated in Eq. (2.4) for the range process and the squared exponential one [Eq. (2.3)] for the noise process. This matches the intuition that the noise process should exhibit more smoothness than the range process. As preliminary experiments revealed, however, this is not a critical choice. With properly learned hyperparameters, the squared exponential function for both processes yields a nearly as high performance in our application.

Evaluating the Joint Data Likelihood of Observations

Having learned a model for the distribution over range functions $\boldsymbol{\alpha} \mapsto r$ as discussed above, we now want to estimate the likelihoood of observing a *new* scan $\mathbf{z} = \langle(\boldsymbol{\alpha}_i, r_i)\rangle_{i=1}^m$ indexed by the beam orientations $\boldsymbol{\alpha}_i$. In other terms, we seek to estimate $p(\mathbf{z}|\mathcal{D}, \boldsymbol{\Theta})$ given the training data $\mathcal{D}$ and the learned covariance parameters $\boldsymbol{\Theta}$. We solve this by considering the predictive distribution for range measurements $\mathbf{r}_*$ at the very same beam orientations $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m$, which is an m-dimensional Gaussian distribution as defined by Eq. (7.5) and Eq. (7.6). As this predictive distribution is a multivariate Gaussian, we can directly calculate the observation likelihood for the data vector $\mathbf{z}$ by evaluating the density function

$$p(\mathbf{z} \mid \boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \quad = \quad \left[(2\pi)^{\frac{m}{2}} |\boldsymbol{\Sigma}_*|^{\frac{1}{2}}\right]^{-1} \cdot \exp\left(-\frac{1}{2}(\mathbf{z} - \boldsymbol{\mu}_*)^\top {\boldsymbol{\Sigma}_*}^{-1}(\mathbf{z} - \boldsymbol{\mu}_*)\right) \ ,$$

or, in a more convenient form

$$\log p(\mathbf{z}|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \quad = \quad -\frac{1}{2}(\mathbf{z} - \boldsymbol{\mu}_*)^\top {\boldsymbol{\Sigma}_*}^{-1}(\mathbf{z} - \boldsymbol{\mu}_*) - \frac{1}{2}\log|\boldsymbol{\Sigma}_*| - \frac{m}{2}\log(2\pi) \ . \qquad (7.9)$$

7.2.1 Regression over Periodic Spaces

In our application, we have to account for the fact that the input vectors $\boldsymbol{\alpha}_i$ are angular quantities rather than unconstrained real valued vectors. This means that an angular distance metric has to be incorporated into the covariance function to avoid discontinuities

at $\pm\pi$. For the one-dimensional case (for planar sensing devices), we use

$$d_{\text{angle}}(\alpha, \beta) := \begin{cases} |\alpha - \beta| & \text{if } |\alpha - \beta| \leq \pi \\ 2\pi - |\alpha - \beta| & \text{otherwise.} \end{cases}$$

Indeed, we also have to adapt the covariance functions themselves to the *periodic* structure of the input space. For example, a periodic variant of the squared exponential covariance function on the unit circle is

$$k(\alpha_i, \alpha_j) = \sigma_f^2 \sum_{p=-\infty}^{\infty} \exp\left(-\frac{|(\alpha_i + 2\pi p) - \alpha_j|^2}{2\ell^2}\right), \tag{7.10}$$

which takes infinitively many influences of a data point on itself into account. The squared exponential covariance function, however, has a strong locality for relevant values of σ_f^2 and ℓ. All summands with $|\alpha_i - \alpha_j| >= 2\pi$ in Eq. (7.10) cannot even be represented using double precision, because their value is too close to zero. We can therefore safely ignore the periodicity in practice and only use the standard covariance function with the modified distance metric defined above.

7.2.2 Efficient Inference by Exploiting Locality

The covariance functions employed in this work are stationary, i.e., they assign small covariance values to those pairs of input points which lie far apart. With the given machine precision, this implies that the resulting covariance matrices are effectively band limited and only have non-zero entries close to the diagonal. This property can be exploited to speed up the computations by using optimized algorithms for sparse matrix operations. In this work, we used the UMFPACK package [Davis, 2004], an optimized solver for sparse linear systems, which resulted in significantly reduced computation times as shown in Fig. 7.4. The run-times are given in seconds for a full iteration of simulating the scan, building the heteroscedastic model, and evaluating the observation likelihood for a given scan with 31 beams. The gain in speed depicted in this figure is due to the sparsity induced by the limitations of machine precision only, that is, exactly the same results are obtained. In addition to this, the covariances could be truncated actively to much tighter bounds before a notable loss of precision occurs. See also Chapter 11 for a more detailed discussion of sparse approximations of Gaussian processes.

7.2.3 GBPs for Monte Carlo Localization

In the beginning of the chapter, we argued that likelihood functions for highly accurate sensors should be regularized when used in sampling based estimation schemes. Then, we

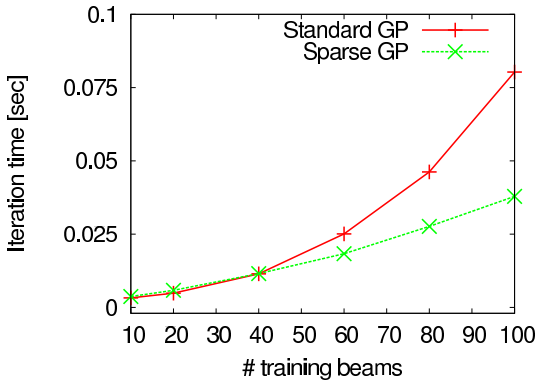


Figure 7.4: Exploiting the sparsity of the covariance matrix reduces the iteration times drastically, especially for larger problem sizes.

proposed Gaussian beam processes as a means of learning and representing the complex distributions over range functions that arise in this way. In this section, we will now join both pieces and describe how they fit into the Monte Carlo localization (MCL) framework.

The task of mobile robot localization is to sequentially estimate the pose $\mathbf{x}$ of a moving robot in its environment. The key idea of Monte Carlo localization (MCL) [Thrun *et al.*, 2000], which belongs to the class of particle filtering algorithms, is to maintain a sampled approximation of the probability density $p(\mathbf{x})$ of the robot's own location. This belief distribution is updated sequentially according to

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{0:t-1}) = \eta \cdot p(\mathbf{z}_t|\mathbf{x}_t) \cdot \int p(\mathbf{x}_t|\mathbf{u}_{t-1}, \mathbf{x}_{t-1}) \cdot p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{0:t-2})\, d\mathbf{x}_{t-1} \; .$$

Here, η is a normalization constant containing the prior observation likelihood $p(\mathbf{z}_t)$, which is equal for the whole sample set and can thus be neglected. The term $p(\mathbf{x}_t|\mathbf{u}_{t-1}, \mathbf{x}_{t-1})$ describes the probability that the robot is at position $\mathbf{x}_t$ given it executed the action $\mathbf{u}_{t-1}$ at position $\mathbf{x}_{t-1}$. Furthermore, $p(\mathbf{z}_t|\mathbf{x}_t)$ denotes the probability of making observation $\mathbf{z}_t$ given the robot's current location is $\mathbf{x}_t$. Concretely, the update of the belief is realized by the following two alternating steps:

1. In the **prediction step**, we propagate each sample to a new location according to the robot's dynamics model $p(\mathbf{x}_t|\mathbf{u}_{t-1}, \mathbf{x}_{t-1})$ given the action $\mathbf{u}_{t-1}$ executed since the previous update.

2. In the **correction step**, the new observation $\mathbf{z}_t$ is integrated into the sample set. This is done by adjusting the weight of each sample according to the likelihood $p(\mathbf{z}_t|\mathbf{x}_t)$ of sensing $\mathbf{z}_t$ given the robot pose $\mathbf{x}_t$.

The measurement model $p(\mathbf{z}|\mathbf{x})$ plays a crucial role in the correction step of the particle

filter. Typically, very peaked models require a large number of particles and induce a high risk of filter divergence. Even when the particles populate the state space densely, the likelihoods of an observation might differ by several orders of magnitude. As the particles are drawn proportionally to the importance weights, which themselves are calculated as the likelihood of $\mathbf{z}_t$ given the pose $\mathbf{x}_t$ of the corresponding particle, a minor difference in $\mathbf{x}_t$ can already result in a large difference in likelihoods. This, in turn, would result in the depletion of such a particle in the re-sampling step. As a consequence, the "peakedness" of a measurement model should depend on the number of particles available and the size of the space to be covered. To the best of our knowledge, all observation models used in practice have parameters for controlling their smoothness. No information is lost in this regularization step, as long as the "peakedness" of the model is not reduced below sensible levels. In the following, we describe how GBPs can be applied to MCL and how the smoothness of the model can be defined in terms of an easy to interpret parameter.

As mentioned in Sec. 7.2, we propose to estimate $p(\mathbf{z}|\mathbf{x})$ by building a GBP model for the robot pose $\mathbf{x}$ online and evaluating the data likelihood of $\mathbf{z}$ according to Sec. 7.2. For building the GBP model, we construct a training set $\mathcal{D}$ of simulated range measurements relative to $\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \boldsymbol{\sigma}_\mathbf{x})$ using an occupancy grid map of the environment. The random perturbations added to $\mathbf{x}$ account for the desired smoothness of the model as motivated above. Indeed, the pose variance parameter $\boldsymbol{\sigma}_\mathbf{x}$ introduced here more naturally quantifies the level of regularization of GBPs compared to other models, as it is directly specified in the space of robot locations. Note that no sensory information is discarded at this point. For sufficiently high sampling densities, one could set $\sigma_\mathbf{x} = 0$ to get the fully peaked model. The MCL measurement update step for the whole filter using GBPs can be summarized as follows:

Algorithmus 1 GBP-based Measurement Update for MCL

for all particles $\mathbf{x}^{[i]}$ **do**
  Generate $\mathcal{D}$ using ray casting in the given map at robot locations sampled from $\mathcal{N}(\mathbf{x}^{[i]}, \boldsymbol{\sigma}_\mathbf{x})$.
  Build local GBPs using $\mathcal{D}$ and the global covariance C.
  Estimate all $\log p(\mathbf{z}|\mathbf{x}^{[i]})$ using the GBP models and weight the particles accordingly.
end for

7.3 Experimental Results

The intention of the experimental evaluation is to determine how well the proposed GBP model performs compared to state-of-the-art probabilistic measurement models for laser range finders. We implemented our approach in `C/C++` and tested it in simulation as well

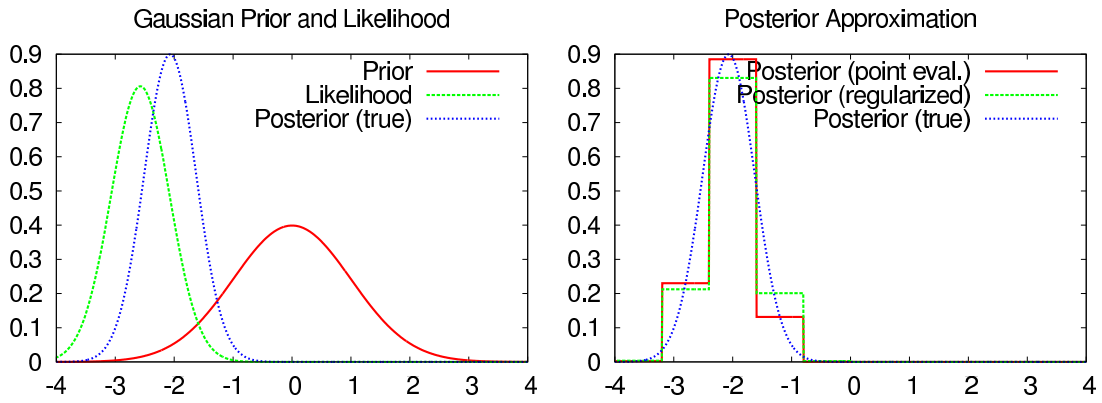


Figure 7.5: Evaluating the proposed regularization procedure in simulation. Left: A Gaussian prior is combined with a Gaussian likelihood function using Bayes' rule. Right: If the posterior is approximated using a coarse histogram, the regularized approximation lies closer to the true posterior in terms of KL-divergence than the one based on point estimates of the likelihood function in the middle of the histogram bins.

as with an ActivMedia Pioneer PII DX8+ robot equipped with a laser range finder in a typical office environment. As the primary sensor is a planar sensing device, we only have to deal with one-dimensional bearing angles. To ensure a fair comparison, we independently optimized the parameters of all models using different data in all experiments.

We will proceed as follows. First, we present the results of a simulation study confirming that regularizing the likelihood function reduces the KL-divergence of posterior estimates from the true posterior. Then, we present tracking and localization results with the real robot and, finally, we present results on simulated data to demonstrate the main benefits of our approach.

7.3.1 Simulation Study on Regularization

In Sec. 7.1, we argued that the robustness of approximative Bayes filter implementations, such as the histogram filter or the particle filter, can be increased by regularizing the observation likelihoods according to the discretization of the state space. In order to quantify this effect, we performed a simulation study with the histogram filter and two different likelihood functions. The central question here is the following. If the state space is discretized using regularly spaced bins $\mathbf{x}^{[i]}$ (see the left diagram in Fig. 7.2) and the Bayesian update rule

$$p(\mathbf{x}^{[i]} \mid \mathbf{z}) \propto p(\mathbf{z} \mid \mathbf{x}^{[i]})\ p(\mathbf{x}^{[i]})$$

is applied to compute the posterior likelihood of bin $\mathbf{x}^{[i]}$ after having observed $\mathbf{z}$, does regularization of $p(\mathbf{z} \mid \mathbf{x}^{[i]})$ decrease the approximation error w.r.t. the true posterior $p(\mathbf{x}|\mathbf{z})$? The non-regularized likelihood function is the point estimate of $p(\mathbf{z} \mid \mathbf{x})$ at the center of

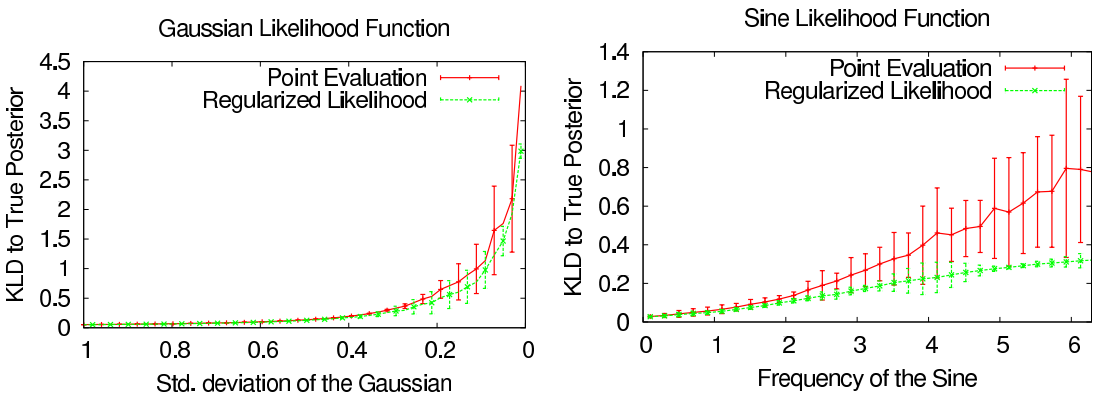


Figure 7.6: Results of the simulation study for Gaussian likelihoods (left) and the Sine function (right). We give the mean KL-divergence (KL-D) with two-standard-deviations errorbars of the true posteriors from the histogram-approximated ones for varying parameters of the likelihood functions (horizontal axis). In both cases and for all parameter settings, regularization yields a significantly lower KL-D (two sample t-test, $\alpha = 5\%$)

the histogram bin, while the regularized version uses a Monte Carlo estimate of Eq. (7.1) over the whole bin $\mathbf{x}^{[i]}$. Figure 7.5 depicts one of the simulated situations with a Gaussian prior and a Gaussian likelihood function. The true posterior is approximated using a fine-grained grid and the approximation quality was assessed on $[-4, 4]$ by calculating the KL-divergence

$$D_{KL} = \sum_j p_{\text{true}}(\mathbf{x}^{[j]} \mid \mathbf{z}) \cdot \log \left(\frac{p_{\text{true}}(\mathbf{x}^{[j]} \mid \mathbf{z})}{p_{\text{approx}}(\mathbf{x}^{[j]} \mid \mathbf{z})} \right)$$

on this grid. The two likelihood functions considered were a *Gaussian* with a standard deviation σ ranging from 1 to 0 and the *Sine* function $1 + \sin(\omega \cdot (x - x_o))$ with a frequency parameter ω ranging from 0 to 20. Figure 7.6 presents the results in both scenarios. For each parameter setting, the offset of the likelihood peak was sampled 100 times from $[-3, 3]$. The error bars give the two standard deviations intervals. For all parameter settings, the regularized likelihood function achieves a significantly lower KL-divergence (two sample t-test, $\alpha = 5\%$). Note that the large errorbars in the results are due to a high variability of the approximation error depending on where the likelihood peak was sampled within $[-3, 3]$.

This experiment provides positive evidence for the claim that our regularization procedure is able to reduce the KL-divergence of the approximated posterior distribution from the true one. As the following experiment show, this leads to measurable improvements in particle filtering-based applications of the likelihood model.

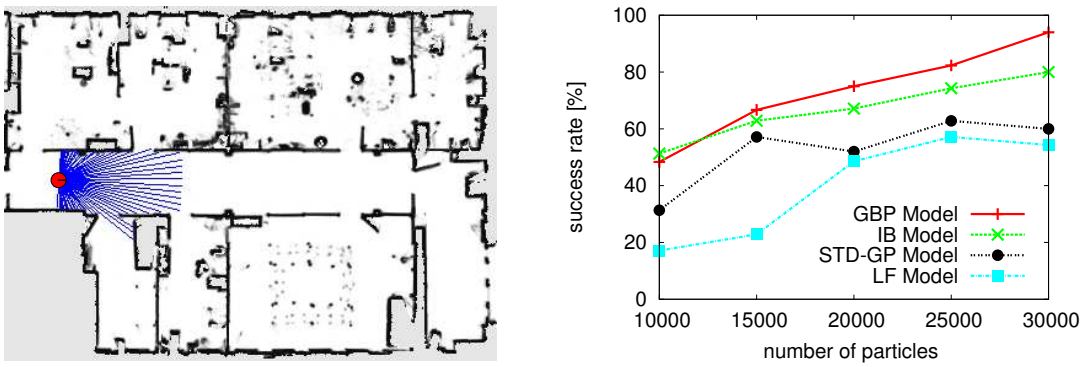


Figure 7.7: Left: Map used for the tracking and global localization experiments. Right: Comparison of the regularized, Gaussian-process–based sensor model (GBP) to the homoscedastic version (STD-GP) and to state-of-the-art sensor models. We give the amount of successful global localizations depending on the number of particles used.

7.3.2 Tracking and Localizing a Real Robot

To evaluate our approach in practically relevant scenarios we performed extensive experiments with a real robot and compared our GBP sensor model to often-used alternatives. Concretely, we compared the performance of:

IB [Independent Beams] The standard beam-based sensor model that assumes independent beams with an additive white noise component [Fox *et al.*, 1999].

LF [Likelihood Fields] Also called the *end-point sensor model* [Thrun, 2001], which calculates the likelihood of a range measurement as a function of the distance of the respective beam's end point to the closest obstacle in the environment.

GBP [Gaussian Beam Processes] Our GBP model presented in Sec. 7.2.

STD-GP [Standard Gaussian Processes] The homoscedastic version of the GBP model, which assumes a constant noise variance for the whole domain.

As described in the previous sections, we build the GBP model online for each pose hypothesis $\mathbf{x}$ by simulating a training set $\mathcal{D}$ of range measurements relative to neighboring poses $\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \boldsymbol{\sigma}_{\mathbf{x}})$ using an occupancy grid map of the environment. The random perturbations added to $\mathbf{x}$ account for the desired smoothness of the model as motivated above. The pose variance parameter $\boldsymbol{\sigma}_{\mathbf{x}}$ introduced here quantifies the level of regularization more naturally compared to other models, as it is directly specified in the space of robot locations. Note that no sensory information is discarded by this operation. For sufficiently high sampling densities, one could set $\boldsymbol{\sigma}_{\mathbf{x}} = \mathbf{0}$ to get the fully peaked model. We then estimate $p(\mathbf{z}|\mathbf{x})$ by evaluating the data likelihood of $\mathbf{z}$ as described in Sec. 7.2.

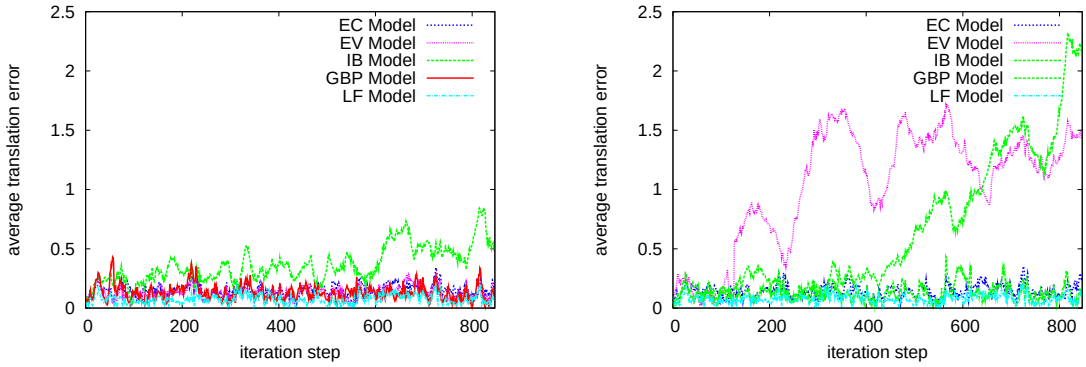


Figure 7.8: Pose tracking results with a real robot using a 180° field of view. The diagrams depict the average localization error for this experiment with 31 (left) and 61 (right) laser beams and give the tracking displacement error (y-axis) in meters for an increasing number of iterations. The errors are averaged over 25 runs on the same trajectory. Due to a lack of proper regularization, the risk of filter divergence increases with a growing number of beams for the IB and LF models.

We implemented Alg. 1 (see Sec. 7.2.3) in `C/C++` and evaluated it using real data acquired with a Pioneer PII DX8+ robot equipped with a laser range scanner in a typical office environment (see the left picture in Fig. 7.7). The experiments described here are designed to investigate how well our GBP approach performs in comparison to the widely used independent-beams model (IB) and the likelihood-fields model (LF), and to quantify its benefits compared to the standard GP (STD-GP), which does not model input-dependent noise. While our approach is computationally more demanding than the alternative ones, it still runs close to real-time for mobile robot tracking. A full iteration including scan simulation and model building takes approximately 0.011 seconds on a standard desktop PC running Linux.

In the first set of experiments, we assess the *position tracking* performance of the MCL filter using the different measurement models. The robot started in the corridor of an office environment (see the left picture in Fig. 7.7) and traversed a path through all adjacent rooms. The explored environment has a long corridor and 8 rooms containing chairs, tables and other pieces of furniture. In total, the map is 20 meters long and 14 meters wide. The two diagrams in Fig. 7.8 depict the average localization errors for this experiment with 31 and 61 laser beams respectively. As can be seen, the GBP model and the likelihood-fields model (LF) show similar, good localization performance and both outperform the independent-beams model (IB). When using more beams for the same task, the difference to IB gets even more pronounced. Due to a lack of proper regularization, the risk of filter divergence increases with a growing number of utilized beams. In another experiment with 181 beams, the GBP model and LF showed a similar behavior as before. IB, however, diverged even earlier than with 61 beams.

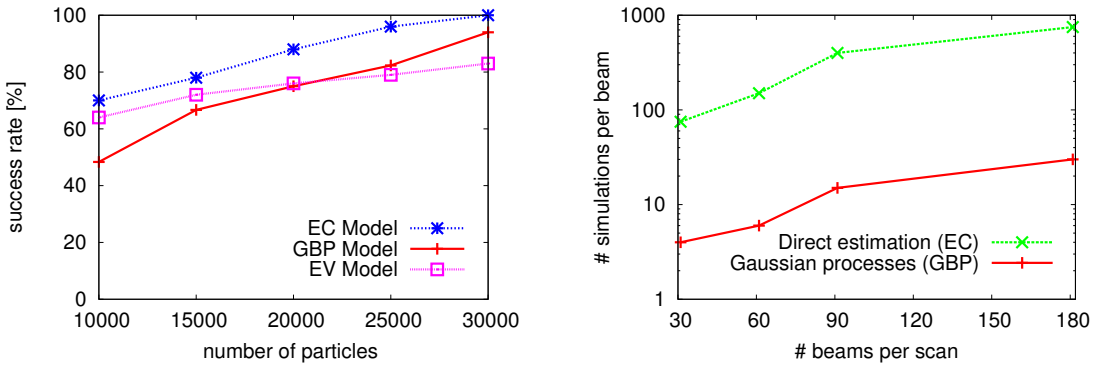


Figure 7.9: Left: Number of successful localizations after 8 integrations of measurements for the three regularization-based likelihood models for different numbers of particles. Right: The Gaussian-process–based model (GBP) requires significantly less training data (note the logarithmic scale) than the EC approach, which directly estimates the covariance matrix from data.

In a second set of experiments we investigated the robustness of the GBP approach during *global localization*. Here, the task is to find a moving robot's pose within the same office environment as for the tracking experiment using a sequence of wheel encoder and laser measurements. The results are given in the right diagram in Fig. 7.7, which shows the number of successful localizations after 8 integrations of measurements of a moving robot for the three measurement models and for different numbers of particles used. In the experiment, we assumed that the localization was achieved when more than 95% percent of the particles differed on average 30 cm at most from the true location of the robot. As can be seen from the diagram, the GBP model performs better than the IB model and both outperform the standard GP and the LF model in this task.

To analyze the effect of the Gaussian process prior placed on the distribution of range functions, we also compared the GBP model against

EC [Estimated Covariance], our regularization scheme as presented in Sec. 7.1, in which the beams per scan are modeled jointly as a Gaussian and the full covariance matrix is estimated from data directly and

EV [Estimated Variance], the *EC* model with cross-correlation components ignored. That is, only the diagonal entries of the covariance matrix are learned.

The left diagram in Fig. 7.9 gives the result of this comparison on the global localization data set. It can be seen that the EC model outperforms both the GBP model as well as EV. In principle, EC takes the same regularization approach as GBP and EV—it estimates a joint Gaussian for the whole range scan based on scans simulated in a local neighborhood of the pose hypotheses. In contrast to GBP, however, it estimates the parameters of this high-dimensional Gaussian from data *directly*. This causes EC to require approximately 20

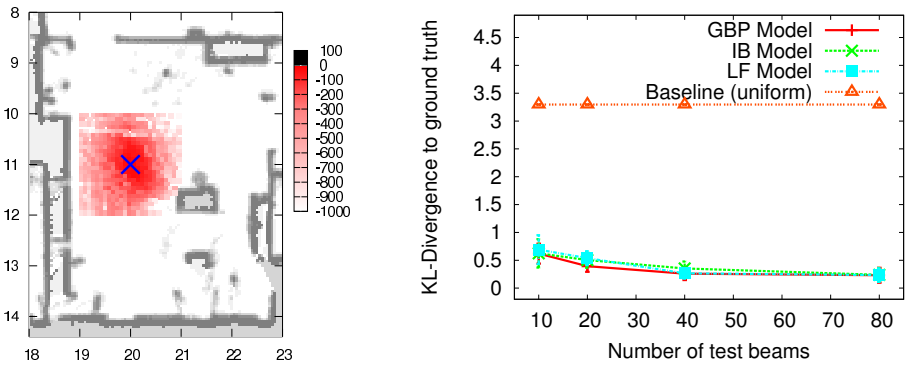


Figure 7.10: Left: Observation log likelihood estimated by the GBP model for a single simulated scan on a discretized pose space in a typical office room. It shows a convex shape and the true robot pose (cross) is in the area of maximal values. Right: In the "nicely structured" office room environment, all methods show similar performance and outperform the uniform (baseline) model, which assigns the same likelihood value to all grid cells. Lower KL-D values are better.

times more training points than GBP. The right diagram in Fig. 7.9 gives the number of simulations per beam required for the two approaches in logarithmic scale. This number was determined by iteratively increasing the number of simulations until the parameters of the estimated Gaussians had reached a steady state. Concretely, to learn models for 31-dimensional scans, GBP requires 4 data points per beam compared to 75 ones in the case of EC. For 181 dimensions, GBP requires 30 simulations per beam vs. 750 for EC.

7.3.3 Results on Simulated Data in a Static Setting

In the previous section, we have evaluated the measurement models in the standard way for mobile robot applications, i.e., we have evaluated their performances in real-world tracking and localization tasks. Although this is closest to the actual application of the models (and should therefore round off any other evaluation strategy), it has also one major drawback: several external factors influence the evaluation, such as the choice of filtering algorithm, the sampling and resampling strategies, and the order in which places are visited along a trajectory. To investigate the strengths and weaknesses of the measurement models independently from specific tracking algorithms, we ran a different set of experiments in a static setting. Here, we use the Kullback-Leibler divergence (KL-D) on a discretized pose space to measure how well the different models are able to reconstruct a pose distribution given just the corresponding laser measurements. More precisely, for each measurement model, we

- discretize the space of robot poses using a three dimensional grid (2D location and heading) and let each grid cell represent one pose hypothesis $\mathbf{x}^{[i]}$,

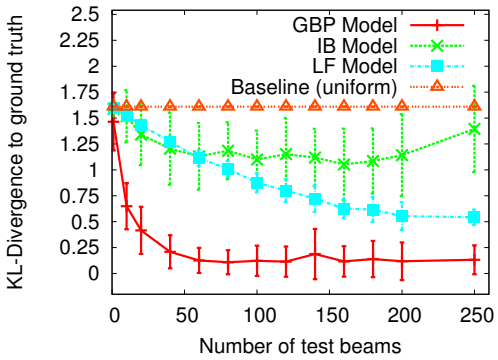


Figure 7.11: In a highly cluttered environment, the IB model performs better than uniform (baseline) but less well than the LF model. Our GBP model significantly outperforms the other methods in this environment.

- select a cell index t to contain the true robot pose $\mathbf{x}$: $\mathbf{x} \in \mathbf{x}^{[t]}$,
- randomly draw m test poses within this cell t and simulate corresponding range measurement vectors $\mathbf{z}_m$ using a given occupancy grid map.
- Now, we evaluate the observation likelihoods $p(\mathbf{z}_m|\mathbf{x}^{[i]})$ for each grid cell and each test observation and sum up the individual observation likelihoods per cell.
- Finally, we normalize the whole likelihood grid and compute the KL-D $D_{KL} = \sum_i p(\mathbf{z}_m|\mathbf{x}^{[i]}) \cdot \log\left(\frac{p(\mathbf{z}_m|\mathbf{x}^{[i]})}{\delta_{i=t}}\right)$ to a binary ground truth grid, where all likelihood mass is concentrated at the cell t, i.e., the true robot pose.

To be able to compute the KL-D measure also when empty cells exist in the grids, we employ the standard trick of adding an extremely small value (10^{-6}) to each cell. The specific choice of this value did not have a notable influence on the measure. The left diagram in Fig. 7.10 depicts such a likelihood grid for the GBP model as well as the true robot location in an office environment. It can be seen that the observation likelihood is nicely peaked around the true robot pose and that the GBP model yields a smooth likelihood function. The KL-D results for this room are given in the right diagram of the same figure. The diagram shows that all three models achieve comparable good performance in the task of recovering the pose distribution in this situation. Additionally, we plot the KL-D for the uniform model taken as a baseline. This model assigns the same, constant likelihood value to all cells.

In contrast to these results for a "nicely structured" office room, the situation is distinctly different in highly cluttered environments such as a laboratory room with many chairs and tables. Here, as documented by the results given in the left diagram in Fig. 7.11, the GBP model clearly outperforms the other two models. The KL-D to the ground truth is always

significantly lower and it decreases with a growing number of laser beams used. The IB model even shows an increasing KL-D with increasing numbers of laser beams due to its lack of proper regularization. In both experiments, we used a relatively coarse grid with grid cell areas of approximately $0.037m^2$.

To summarize, the experiments on real and on simulated data demonstrate that the proposed regularization approach significantly improves the performance of Bayes filters for pose recovery tasks from range measurements. The Gaussian beam process (GBP) model achieves a nearly as high modeling accuracy as regularized models that are estimated directly from data while it requires approximately 20 times less data points to be learned.

7.4 Related Work

Existing range sensor models for can be classified into feature-based, beam-based, and correlation-based models. Feature-based approaches typically extract a set of features from the range scan $\mathbf{z}$ and match them to features contained in an environmental model in order to obtain $p(\mathbf{z}|\mathbf{x})$. Whereas such approaches have been proven to be robust in various applications, they assume that the features are known beforehand and that they can be extracted reliably, which might be hard in unstructured or cluttered environments. Alternative approaches directly operate on the dense measurements and therefore are applicable even in situations in which the relevant features are unknown. As Thrun *et al.* [2005] point out, extracting robust features is hard especially in unstructured and cluttered environments and can result in a loss of information. Like most state-of-the-art approaches, we therefore operate on the dense measurements directly.

Beam-based models consider each value r_i of the measurement vector $\mathbf{z}$ as a separate range measurement and represent its one-dimensional distribution by a parametric function depending on the expected range measurement in the respective beam direction (see Fox *et al.* [1999] for example). Such models are closely linked to the geometry and the physics involved in the measurement process. They are often called *ray-cast models*, because they rely on ray casting operations within an environmental model, e.g., an occupancy grid map, to calculate the expected beam lengths. As a major drawback, the traditional approach assumes independent beams, which leads to highly peaked likelihood functions when one increases the number of beams per measurement (e.g., to increase the spatial resolution). In practice, this problem is dealt with by sub-sampling of measurements, by introducing minimal likelihoods for beams, by inflating the measurement uncertainty [Petrovskaya *et al.*, 2006], or by other means of regularization of the resulting likelihoods (see, e.g., [Arulampalam *et al.*, 2002]). Other work along these lines include the regularized particle filter [Musso *et al.*, 2001] and the Parzen particle filter [Lehn-Schiøler *et al.*, 2004]. Early

references to the general idea include [Liu and West, 2001], in which the authors propose to exchange the delta-Dirac kernels of a particle filters for Gaussian kernels in order to estimate fixed system parameters and dynamic state variables jointly.

Correlation-based methods typically build local maps from consecutive scans and correlate them with a global map [Schiele and Crowley, 1994, Konolige and Chou, 1999]. An intuitive and effective approach also associated with this class of models is the so-called likelihood-fields model or end point model [Thrun, 2001]. Here, the likelihood of a single range measurement is a function of the distance of the beam's respective end point to the closest obstacle in the environment. As in the ray cast model, each beam is treated independently. This model lacks a physical explanation, as it can basically "see through walls", but it is more efficient than ray cast models and works well in practice.

Work that specifically dealt with peaked measurement models include Pfaff *et al.* [2006], who adapt the smoothness of the likelihood model depending on the region covered by the individual particles, Fox [2001], and Kwok *et al.* [2003], who adapt the number of particles depending on the progress of the localization process and computational power. These approaches have been developed independently from specific measurement models and should be applicable directly to GBPs as well. Finally, GBPs are related to Gutierrez-Osuna *et al.*'s [1998] neural networks approach to modeling the measurement of an ultrasonic range sensor, which in contrast to GBPs assumes scans of fixed size.

7.5 Conclusion

In this chapter, we introduced Gaussian beam processes as a novel probabilistic measurement model for range sensors. The key idea of our approach is to view the measurement modeling task as a Bayesian regression problem and to solve it using Gaussian processes. A major benefit of our model in localization tasks is that it allows to reason in the space of range *functions* directly rather than being restricted to discrete sets of independent beam measurements. This offers a sound way of dealing with highly peaked observation likelihood functions in Monte Carlo-based estimation approaches.

As our experiments with real and simulated data demonstrate, Gaussian beam processes provide superior robustness compared to state-of-the-art sensor models for range sensors.

Chapter 8

Monocular Range Sensing

We present a novel approach to estimating depth from single monocular camera images by learning the relationship between range measurements and visual features. Applied to test images, our model not only yields the most likely range value corresponding to a certain visual input but also the predictive uncertainty. This information, in turn, can be utilized to build a consistent occupancy grid map of the environment.

Cameras have become popular sensors in the robotics community. Compared to proximity sensors such as laser range finders, they have the advantage of being cheap, lightweight, and energy efficient. The drawback of cameras, however, is the fact that only a projection of the scene is recorded and, thus, it is not possible to sense depth information directly. From a geometric point of view, one needs at least two images taken from different locations to recover the depth information analytically. An alternative approach that requires just one monocular camera and that we follow here, is to learn from previous experience how visual appearance is related to depth. Such an ability is also highly developed in humans, who are also able to utilize monocular cues for depth perception [Swaminathan and Grossberg, 2002].

As a motivating example, consider Fig. 8.1, which shows the (warped) image of an office environment. Overlayed in white, we visualize the most likely area of free space that is predicted by our approach. We explicitly do not try to estimate a depth map for the whole image, as for example Saxena *et al.* [2007]. Rather, we aim at learning the function that, given an image, maps measurement directions to their corresponding distances to the closest obstacles. Such a function can be utilized to solve various tasks of mo-

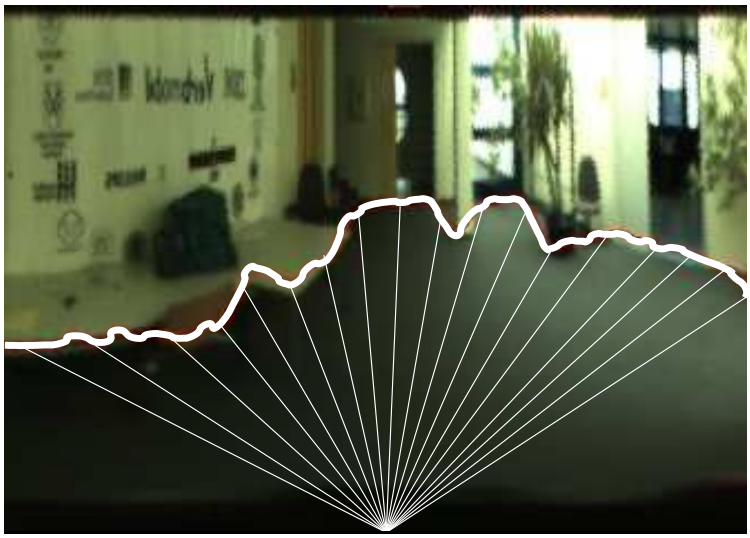


Figure 8.1: Our approach estimates proximity information from a single image after having learned how visual appearance is related to depth.

bile robots including local obstacle avoidance, localization, mapping, exploration, or place classification.

In this chapter, we formulate the range estimation task as a supervised regression problem, in which the training set is build by acquiring images of the environment as well as proximity data provided by a laser range finder. We discuss how to extract appropriate visual features from the images using algorithms for edge detection and dimensionality reduction. We apply Gaussian processes as the learning framework in our proposed system since this technique is able to model non-linear functions, offers a direct way of estimating uncertainties for its predictions, and it has proven successful in previous work involving range functions (see the previous chapter, for instances).

The chapter is organized as follows. First, we discuss appropriate visual features and how they can be extracted from images in Sec. 8.1. We then formalize the problem of predicting range from these features and introduce the proposed learning framework in Sec. 8.2. In Sec. 8.3, we present the experimental evaluation of our algorithm as well as an application to the mapping problem.

8.1 Omnidirectional Vision and Feature Extraction

Our goal is to learn the relationship between visual input and the extent of free space around the robot. Figure 8.2 depicts the configuration of our robot used for data acquisition. An omnidirectional camera system (catadioptric with a parabolic mirror) is mounted on top of a SICK laser range finder. This setup allows the robot to perceive the whole

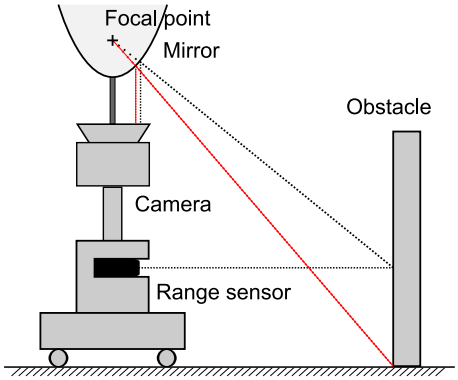


Figure 8.2: Our experimental setup. The training set was recorded using a mobile robot equipped with an omnidirectional camera (monocular camera with a parabolic mirror) as well as a laser range finder.

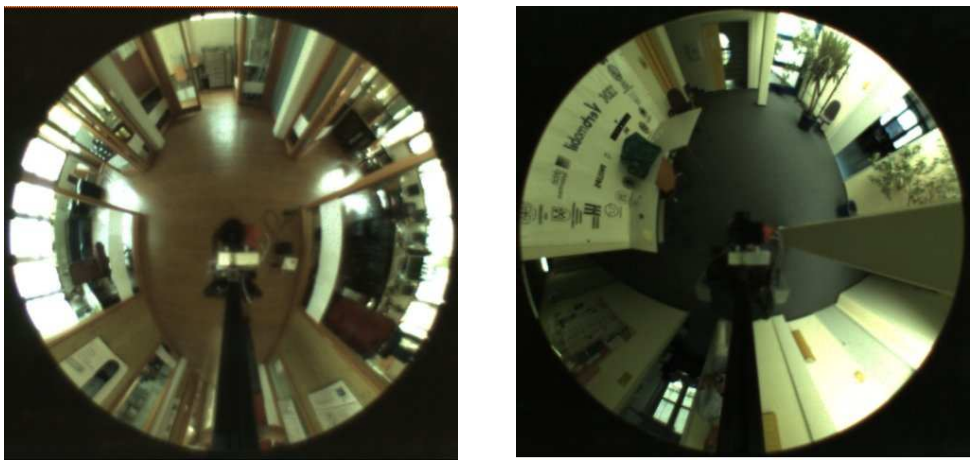


Figure 8.3: Two typical omnidirectional images recorded at the University of Freiburg (left) and at the German Research Center for Artificial Intelligence (DFKI) in Saarbrücken (right).

surrounding area at every time step as the two example images in Fig. 8.3 illustrate. The images in this figure show typical situations from the two benchmark data sets used in this chapter. They have been recorded at the University of Freiburg (left image) and at the German Research Center for Artificial Intelligence (DFKI) in Saarbrücken (right image).

The part of an omnidirectional image which is most strongly correlated with the distance to the nearest obstacle in a certain direction α is the strip of pixels oriented in the same direction covering the area from the center of the image to its margins. With the type of camera used in our experiments, such strips have a dimensionality of 420 (140 pixels, each having a *hue*, *saturation*, and a *value* component). In order to make these strips easier accessible to filter operators, we warp the omnidirectional images (see Fig. 8.3) into panoramic views (e.g., see Fig. 8.1) so that angles in the polar representation now

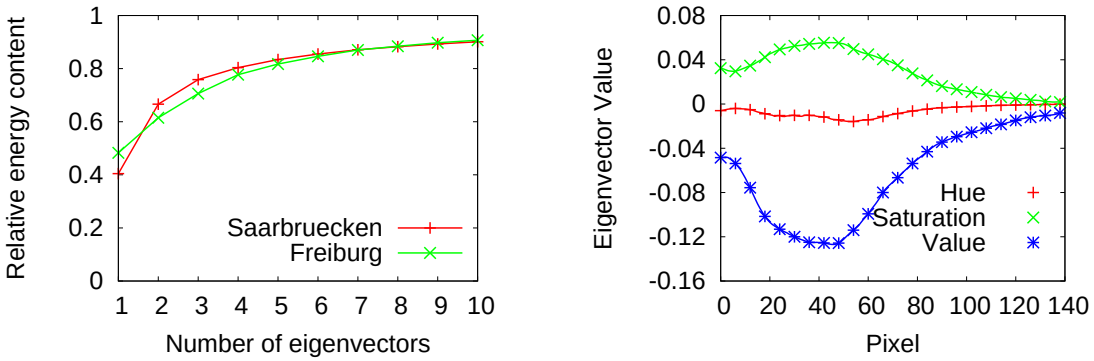


Figure 8.4: Left: The amount of variance explained by the first principle components (eigenvectors) of the pixel columns in the two data sets. Right: The 420 components of the first eigenvector of the Freiburg data set.

correspond to column indices in the panoramic one. This transformation allows us to replace complicated image operations in the polar domain by easier and more robust ones in a Cartesian coordinate system. In the following, we describe several ways of extracting useful low-dimensional feature vectors $\mathbf{v}$ from the 420-dimensional image columns, which can be used to index the training and test sets for learning and evaluation, respectively.

8.1.1 Unsupervised Dimensionality Reduction

As a classic way of reducing the complexity of a data set, one can apply the principle component analysis (PCA) to the raw 420-dimensional pixel vectors that are contained in the columns of the panoramic images. The PCA is implemented using eigenvalue decomposition of the covariance matrix of the training vectors. It yields a linear transformation which brings the input vectors into a new basis so that their dimensions are now ordered by the amount of data set variance they carry. In this way, one can truncate the vectors to a few components without losing a large amount of information. The left diagram in Fig. 8.4 shows the remaining fraction of variance after truncating the transformed data vectors after a certain number of components. The right diagram in the same figure shows the 420 components of the first eigenvector for the Freiburg data set grouped by hue, saturation, and value.

For the experiments reported on in Sec. 8.3, we trained the PCA on 600 input images and retained the first six principle components. Our experiments revealed that the value channel of the visual input is more important than hue and saturation for our task. The GP model learned with these 6D features is termed *PCA-GP* in the experimental section.

8.1.2 Edge-based Features

The PCA is an unsupervised method that does not take into account prior information that might be available about the task to be solved—like the fact that distances to the closest obstacles are to be predicted in our case. Driven by the observation that, especially in indoor environments, there is a strong correlation between the extent of free space and the presence of horizontal edge features in the panoramic image, we also assessed the potential of edge-type features in our approach.

Laws' convolution masks [Davies, 1997] provide an easy way of constructing local feature extractors for discretized signals. The idea is to define three basic convolution masks

- $L_3 = (1, 2, 1)^T$ (Weighted Sum: Averaging),
- $E_3 = (-1, 0, 1)^T$ (First difference: Edges), and
- $S_3 = (-1, 2, -1)^T$ (Second difference: Spots),

each having a different effect on (1D) patterns, and to construct more complex filters by a combination of the basic masks. In our application domain, we obtained good results with the (2D) directed edge filter $E_5 L_5^\top$, which is the outer product of E_5 and L_5. Here, E_5 is a convolution of E_3 with L_3 and L_5 denotes L_3 convolved with itself. After filtering with this mask, we apply an optimized threshold to yield a binary response. This feature type is denoted as *Laws5* in the experimental section. As another well-known feature type, we applied the $E_3 L_3^\top$ filter, i.e. the Sobel operator, in conjunction with Canny's algorithm [Canny, 1986]. This filter yields binary responses at the image locations with maximal gray-value gradients in gradient direction. We denote this feature type as *Laws3+Canny* in Section 8.3. For both edge detectors, *Laws5* and *Laws3+Canny*, we search along each image column for the first detected edge. This pixel index then constitutes the feature value.

To increase the robustness of the edge detectors described above, we applied *lightmap damping* as an optional preprocessing step to the raw images. This means that, in a first step, a copy of the image is converted to gray scale and strongly smoothed with a Gaussian filter, such that every pixel represents the brightness of its local environment. This is referred to as the *lightmap*. The brightness of the original image is then scaled with respect to the lightmap, such that the *value* component of the color is increased in dark areas and decreased in bright areas. In the experimental section, this operation is marked by adding *+LMD* to the feature descriptions. The right image in Fig. 8.5 shows *Laws5+LMD* edge features extracted from an image of the Freiburg data set.

All parameters involved in the edge detection procedures described above were optimized

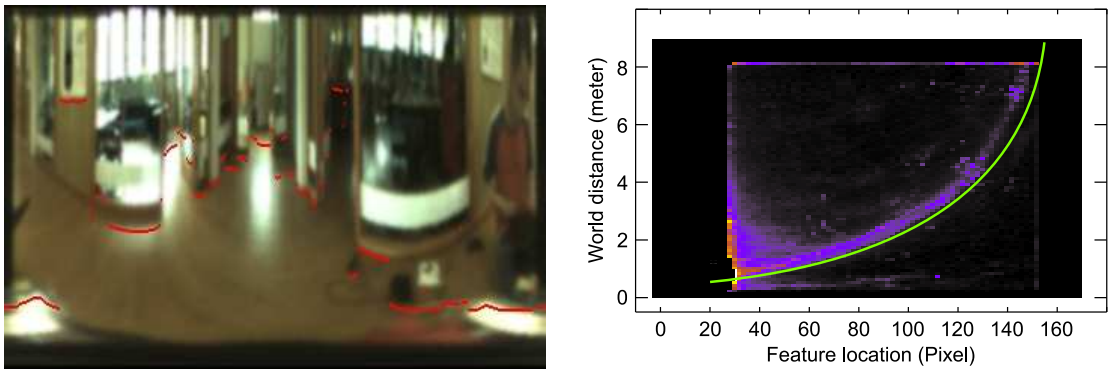


Figure 8.5: Left: Example *Laws5+LMD* feature extracted from one of the Freiburg images. Right: Histogram for *Laws5+LMD* edge features. Each cell in the histogram is indexed by the pixel location of the edge feature (x-axis) and the length of the corresponding laser beam (y-axis). The optimized (parametric) mapping function that is used as a benchmark in our experiments is overlaid in green.

to yield features that lie as close as possible to the laser end points projected onto the omnidirectional image using the acquired training set. For our regression model, we can now construct 4D feature vectors $\mathbf{v}$ consisting of the Canny-based feature, the *Laws5*-based feature, and both features with additional preprocessing using lightmap-damping. Since every one of these individual features captures slightly different aspects of the visual input, the combination of all, in what we call the *Feature-GP*, can be expected to yield more accurate predictions than any single one.

As a benchmark for predicting range information from edge features, we also evaluated the individual features directly. For doing so, we fitted a parametric function to training samples of feature-range pairs. This mapping function computes for each pixel location of an edge feature the length of the corresponding laser beam. The right diagram in Fig. 8.5 shows the feature histogram for the *Laws5+LMD* features from one of our test runs that was used for the optimization. The color of a cell (c_x, c_y) in this diagram encodes the relative amount of feature detections that were extracted at the pixel location c_x (measured from the center of the omnidirectional image) and that have a corresponding laser beam with a length of c_y in the training set. The optimized projection function is overlayed in green.

8.2 Learning Depth from Images

Given a training set of images and corresponding range scans acquired in a setting, we can treat the problem of predicting range in *new* situations as a supervised learning problem. The omnidirectional images can be mapped directly to the laser scans since both measurements can be represented in a common, polar coordinate system. Note that our approach is not restricted to omnidirectional cameras in principle. However, the correspondence be-

tween range measurements and omnidirectional images is a more direct one and the field of view is considerably larger compared to standard perspective optics.

In the spirit of the Gaussian beam processes (GBP) model introduced in Chapter 7, we propose to put a GP prior on the range function, but in contrast, here we use the visual features $\mathbf{v}$ described in the previous section as indices for the range values rather than the bearing angles α. Concretely, we extract for every viewing direction α a vector of visual features $\mathbf{v}$ from an image $\mathbf{c}$ and phrase the problem as learning the range function $f(\mathbf{v}) = y$ that maps the visual input $\mathbf{v}$ to distances y. We learn this function in a supervised manner using a training set $\mathcal{D} = \{\mathbf{v}_i, y_i\}_{i=1}^n$ of observed features $\mathbf{v}_i$ and corresponding laser range measurements y_i. If we place a GP prior on the non-linear function f, i.e., we assume that all range samples y indexed by their corresponding feature vectors $\mathbf{v}$ are jointly Gaussian distributed, we obtain

$$f(\mathbf{v}_*) \quad \sim \quad \mathcal{N}(\mu_*, \sigma_*^2) \tag{8.1}$$

for the noise-free range with

$$\mu_* = \mathbf{k}_{\mathbf{v}_*\mathbf{v}}^\top (\mathbf{K}_{\mathbf{vv}} + \sigma_n \mathbf{I})^{-1} \mathbf{y} \tag{8.2}$$

$$\sigma_*^2 = k(\mathbf{v}_*, \mathbf{v}_*) - \mathbf{k}_{\mathbf{v}_*\mathbf{v}}^\top (\mathbf{K}_{\mathbf{vv}} + \sigma_n \mathbf{I})^{-1} \mathbf{k}_{\mathbf{v}_*\mathbf{v}} \tag{8.3}$$

as defined in Chapter 2 [for the noisy case $y_* = f(\mathbf{v}_*) + \epsilon$]. Here, we have $\mathbf{K}_{\mathbf{vv}} \in \mathbb{R}^{n \times n}$ with $[\mathbf{K}_{\mathbf{vv}}]_{ij} = k(\mathbf{v}_i, \mathbf{v}_j)$, $\mathbf{k}_{\mathbf{v}_*\mathbf{v}} \in \mathbb{R}^n$ with $[\mathbf{k}_{\mathbf{v}_*\mathbf{v}}]_i = k(\mathbf{v}_*, \mathbf{v}_i)$, $\mathbf{y} = (y_1, \ldots, y_n)^\top$, and $\mathbf{I}$ the identity matrix. σ_n denotes the global noise parameter. As covariance function, we apply the squared exponential form

$$k(\mathbf{v}_p, \mathbf{v}_q) = \sigma_f^2 \cdot \exp\left(-\frac{1}{2\ell^2}|\mathbf{v}_p - \mathbf{v}_q|\right). \tag{8.4}$$

A particularly useful property of Gaussian processes for our application is the availability of the predictive uncertainty at every query point. This means that visual features $\mathbf{v}_*$ which lie close to points $\mathbf{v}$ of the training set result in more confident predictions than features which fall into a less densely sampled region of feature space.

So far, our model assumes *independent* range variables y_i and it thus ignores dependencies that arise, for instance, because "neighboring" range variables and visual features are likely to correspond to the same object in the environment. Angular dependencies can be included, for example, (a) by explicitly considering the angle α as an additional index dimension in $\mathbf{v}$ or (b) by applying Gaussian beam processes (GBPs) as an independent post-processing step to the *predicted* range scan. While the first variant would require a

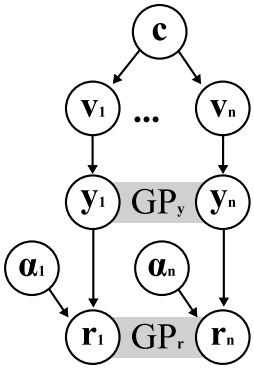


Figure 8.6: Graphical model for predicting ranges **r** from a camera image **c**. The gray bars group sets of variables that are fully connected and that are jointly distributed according to a GP model.

large amount of additional training data—since it effectively couples the visual appearance and the angle of observation, the second alternative is relatively easy to realize and to implement. Figure 8.2 gives a graphical representation of the second approach. The gray bars group sets of variables that are fully connected and jointly distributed according to a GP model. We denote with $\mathcal{GP}_y$ the Gaussian process that maps visual features to ranges and with $\mathcal{GP}_r$ the heteroscedastic GP that is applied as a post-processing step to single, predicted range scans. For $\mathcal{GP}_r$, the task is to learn the mapping $\alpha \mapsto r$ using a training set of *predicted* range values **r**. Since we do not want to constrain the model to learning from the *mean* predictions $\mu_*(\mathbf{x}_i)$ only, we need a way of incorporating the predictive uncertainties $\sigma_*^2(\mathbf{v}_i)$ for the feature-based range predictions y_*. This can be achieved by modifying the matrix $\mathbf{R}$ [see Eq. (6.1) and Eq. (6.2)], which contains the *local* noise rates at the individual index locations, to

$$\tilde{\mathbf{R}} = \mathrm{diag}(\tilde{\mathbf{r}}) \ , \qquad \tilde{r}_i = r_i \cdot \sigma_*^2(\mathbf{v}_i) \ . \tag{8.5}$$

Here, each noise rate r_i that is associated to a training point is adapted according to the confidence $\sigma_*^2(\mathbf{v}_i)$, with which $\mathcal{GP}_y$ has estimated the corresponding range value. Note that this "trick" of gating out training points by artificially increasing their associated variance was also applied in Chapter 5 for deriving a GP mixture model. A more detailed discussion of the approach can be found there and in [Tresp, 2000].

To summarize, in the full model that also deals with angular dependencies in a range scan, which is denoted by the postfix *+GBP* in the experimental evaluation, prediction of a full range scan given one omnidirectional image is performed as follows.

1. Warp the omnidirectional image into a panoramic view.

2. Extract for every pixel column i a vector of visual features $\mathbf{v}_i$.

3. Use $\mathcal{GP}_y$ (termed *Feature-GP*) to make independent range predictions about y_i.

4. Learn a heteroscedastic GBP $\mathcal{GP}_r$ for the set of predicted ranges $\{y_i\}_{i=1}^n$ indexed by their bearing angles α_i and make the final range predictions r_i for the same bearing angles.

As the following experimental evaluation revealed, this additional GBP treatment (post-processing with $\mathcal{GP}_r$) further increases the accuracy of range predictions. The gain, however, is rather small compared to what the GP treatment $\mathcal{GP}_y$ adds to the accuracy achievable with the baseline feature mappings. This might be due to the fact that the extracted features—and the constellation of several feature types even more so—carry information of neighboring pixel strips, such that angular dependencies are incorporated at this early stage already.

8.3 Experimental Results

The experiments presented in this section are designed to evaluate how well the proposed system is able to estimate range from single monocular camera images. We document a series of different experiments: First, we evaluate the accuracy of the estimated range scans using (a) the individual edge features directly, (b) the *PCA-GP*, and (c) the *Feature-GP*, which constitutes our regression model with the four edge-based vision features as input dimensions. Then, we illustrate how these estimates can be used to build grid maps of the environment. We also evaluated whether applying the GBP model, which was introduced in Chapter 7, as a post-processing step to the predicted range scans can further increase the prediction accuracy. The GBP model places a Gaussian process prior on the range function (rather than on the function that maps features to distances) and, thus, also models angular dependencies. We denote these models by *Feature-GP+GBP* and *PCA-GP+GBP*.

The two data sets used for the experiments have been recorded using a mobile robot equipped with a laser scanner, an omnidirectional camera, and odometry sensors at the AIS lab at the University of Freiburg [Fig. 8.3 (left)] and at the German Research Center for Artificial Intelligence (DFKI) in Saarbrücken [Fig. 8.3 (right)]. The two environments have quite different characteristics—especially in the visual aspects. While the environment in Saarbrücken mainly consists of solid, regular structures and a homogeneously colored floor, the lab in Freiburg exhibits many glass panes, an irregular, wooden floor, and challenging lighting conditions.

8.3.1 Accuracy of Range Predictions

We evaluated eight different system configurations, each on both test data sets. Table 8.1 summarizes the average RMSE (root mean squared error) obtained for the individual scenarios. The error is measured as the deviation of the range predictions using the visual input from the corresponding laser ranges recorded by the sensor. The first four configurations, referred to as C1 to C4, apply the optimized mapping functions for the different edge features (see Fig. 8.5). Depending on the data, the features provide estimates with an RMSE of between 1.7 m and 3 m. We then evaluated the configurations C5 and C6 which use the four edge-based features as inputs to a Gaussian process model as described in Sec. 8.2 to learn the mapping from the feature vectors to the distances. The learning algorithm was able to perform range estimation with an RMSE of around 1 m. Note that we measure the prediction error relative to the recorded laser beams rather than to the true geometry of the environment. Thus, we report a conservative error estimate that also includes mismatches due to reflected laser beams contained in the test set. To give a visual impression of the prediction accuracy of the *Feature-GP*, we give a typical laser scan and the mean predictions in the right diagram in Fig. 8.7.

As configuration C7, we evaluated the *PCA-GP* approach that does not require engineered features, but rather works on the low-dimensional representation of the raw visual input computed using the PCA. The resulting six-dimensional feature vector is used as input to the Gaussian process model. With an RMSE of 1.2 m to 1.4 m, the *PCA-GP* outperforms all four engineered features, but is not as accurate as the *Feature-GP*. For configurations C6 and C8, we predicted the ranges per scan using the two different methods and additionally applied the GBP model (see Chapter 7) to incorporate angular dependencies between the predicted beams. This post-processing step yields slight improvements compared to the

Table 8.1: Average errors obtained with the different methods. The root mean squared errors (RMSE) are calculated relative to the mean predictions for the complete test sets.

	RMSE on test set	
Configuration	**Saarbrücken**	**Freiburg**
C1: Laws5	1.70m	2.87m
C2: Laws5+LMD	2.01m	2.08m
C3: Laws3+Canny	1.74m	2.87m
C4: Laws3+Canny+LMD	2.06m	2.59m
C5: Feature-GP	1.04m	1.04m
C6: Feature-GP+GBP	**1.03m**	**0.94m**
C7: PCA-GP	1.24m	1.40m
C8: PCA-GP+GBP	1.22m	1.41m

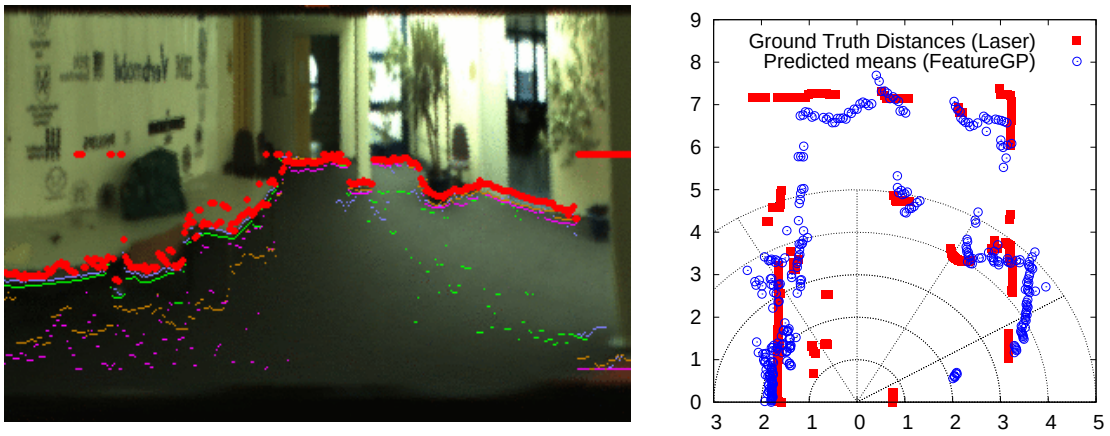


Figure 8.7: Left: Estimated ranges projected back onto the camera image using the feature detectors directly (small dots) and using the *Feature-GP* model (red points). Right: Prediction results and the true laser scan at one of the test locations visualized from a birds-eye view.

original variants C5 and C7.

The left image in Fig. 8.7 depicts the predictions based on the individual vision features and the *Feature-GP*. It can be clearly seen from the image, that the different edge-based features model different parts of the range scan well. The *Feature-GP* fuses these unreliable estimates to achieve high accuracy on the whole scan. The result of the *Feature-GP+GBP* variant for the same situation is given in Fig. 8.1. The right diagram in Fig. 8.7 visualizes a typical prediction result and the corresponding laser scan—which can be regarded here as the ground truth—from a birds-eye view. The evolution of the RMSE for the different methods over time is given in Fig. 8.8. As can be seen from the diagrams, the prediction using the *Feature-GP* model outperforms the other techniques and achieves a near-constant error rate.

In summary, our GP-based technique outperforms the individual, engineered features for range prediction. The smoothed approach (C6) yields the best predictions with a RMSE of around 1 m. Even if requiring no background information one can obtain good results by a combination of PCA for dimensionality reduction and GP learning with an error that is only slightly larger (C8 versus C6).

8.3.2 Application to Mapping

Our approach can be applied to a variety of robotics tasks such as obstacle avoidance, localization, or mapping. To illustrate this, we show how to learn a grid map of the environment from the predictive range distributions. Compared to occupancy grid mapping where one estimates for each cell the probability of being occupied or free, we use the

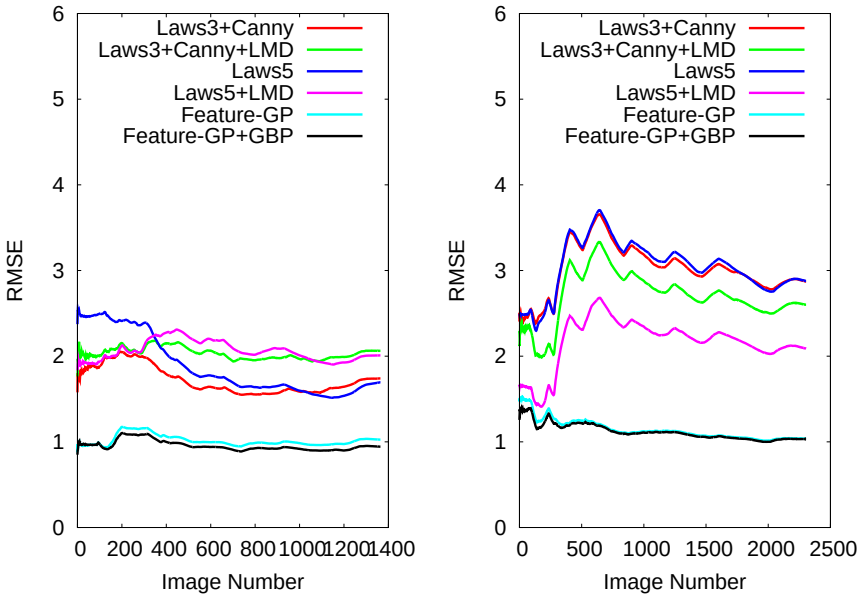


Figure 8.8: The evolution of the root mean squared error (RSME) for the individual images of the Saarbrücken (left) and Freiburg (right) data sets.

so-called *reflection probability maps*. A cell of such a map models the probability that a laser beam passing this cell is reflected or not. Reflection probability maps, which are learned using the so-called *counting model*, have the advantage of requiring no hand-tuned sensor model such as occupancy grid maps (see [Burgard *et al.*, 2007] for further details). The reflection probability m_i of a cell i is given by

$$m_i \quad = \quad \frac{\alpha_i}{\alpha_i + \beta_i} \; , \tag{8.6}$$

where α_i is the number of times an observation hits the cell, i.e., ends in it, and β_i is the number of misses, i.e., the number of times a beam has intercepted a cell without ending in it. Since our GP approach does not estimate a single laser end point, but rather a full (normal) distribution $p(z)$ of possible end points, we have to integrate over this distribution (see Fig. 8.9). More precisely, for each grid cell c_i, we update the cell's reflectance values according to the predictive distribution $p(z)$ according to the following formulas:

$$\alpha_i \quad \leftarrow \quad \alpha_i + \int_{z \in c_i} p(z)\, dz \tag{8.7}$$

$$\beta_j \quad \leftarrow \quad \beta_i + \int_{z > c_i} p(z)\, dz \; . \tag{8.8}$$

Note that for perfectly accurate predictions, the extended update rule is equivalent to the standard formula stated above.

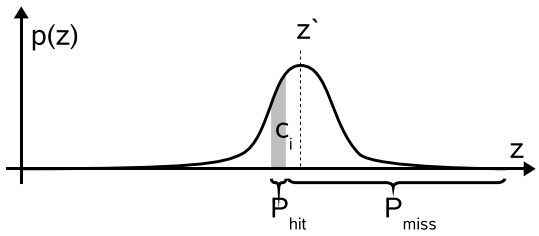


Figure 8.9: The counting model for reflectance grid maps in conjunction with sensor models that yield Gaussian predictive distributions over ranges.

We applied this extended reflection probability mapper to the trajectories and range predictions that resulted from the experiments reported above. Figure 8.10 presents the laser-based maps using a standard mapper (left column) and the extended mapper using the predicted ranges (right column) for both environments (Freiburg on top and Saarbrücken below). In both cases, it is possible to build an accurate map, which is comparable to maps obtained with infrared proximity sensors [Ha and Kim, 2004] or sonars [Thrun *et al.*, 1998].

8.4 Related Work

The problem of recovering geometric properties of a scene from visual measurements is one of the fundamental problems in computer vision and is also frequently addressed in the robotics literature. Stereo camera systems are widely used to estimate the missing depth information that single cameras cannot provide directly. Stereo systems either require a careful calibration to analytically calculate depth using geometric constraints or, as flow[Sinz *et al.*, 2004] demonstrated, can be used in combination with non-linear, supervised learning approaches to recover depth information. Often, sets of features such as SIFT [Lowe, 2004] are extracted from two images and matched against each other. Then, the feature pairs are used to constrain the poses of the two camera locations and/or the point in the scene that corresponds to the image feature. In this spirit, the motion of the camera has been considered by Davision *et al.* [2007] and Strasdat *et al.* [2007]. Sim and Little [2006] present a stereo-vision based approach to the SLAM problem, which also includes the recovery of depth information. Their approach contains both the matching of discrete landmarks as well as dense grid mapping using vision cues.

An active way of sensing depth using a single monocular camera is known as *depth from defocus* [Favaro and Soatto, 2005] or *depth from blur*. Corresponding approaches typically adjust the focal length of the camera and analyze the resulting local changes in image sharpness. Torralba and Oliva [2002] present an approach for estimating the mean depth of full scenes from single images using spectral signatures. While their approach is likely to

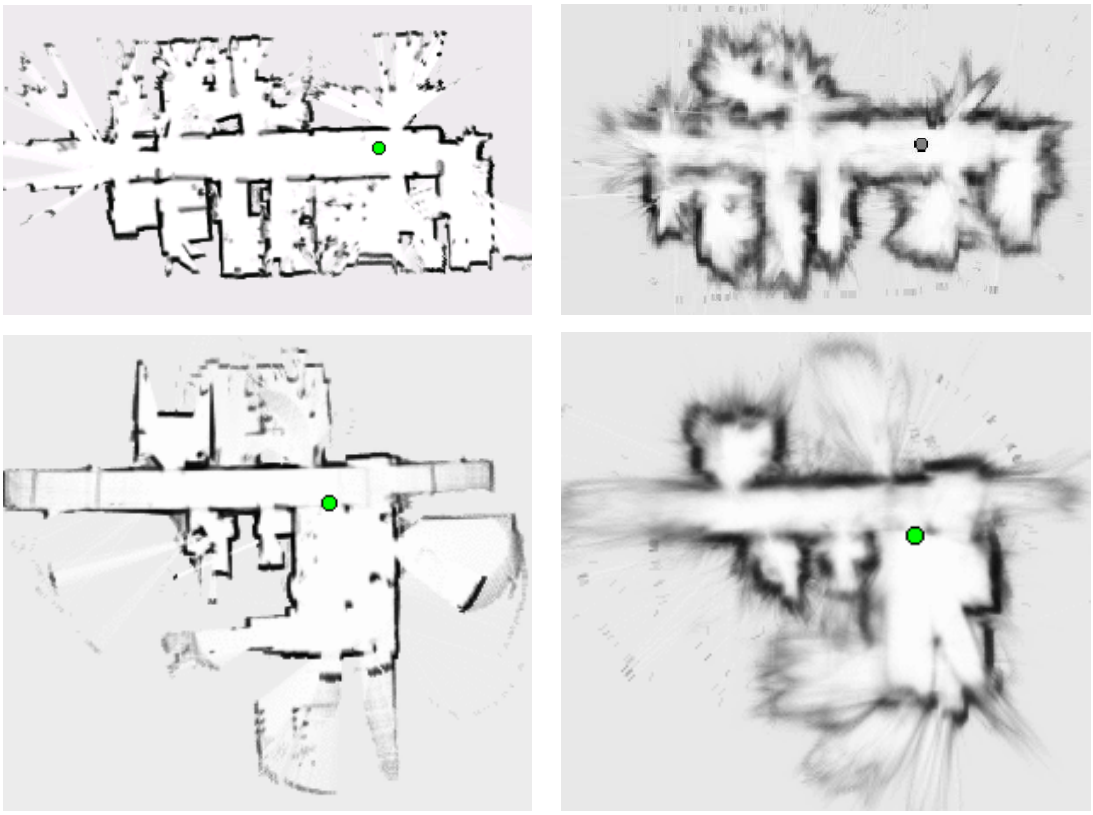


Figure 8.10: Maps of the Freiburg AIS lab (top row) and DFKI Saarbrücken (bottom row) using real laser data (left) and the predictions of the *Feature-GP* (right).

improve a large number of recognition algorithms by providing a rough scale estimate, the spatial resolution of their depth estimates does not appear to be sufficient for the problem studied in this chapter. Dahlkamp *et al.* [2006] learn a mapping from visual input to road traversability in a self-supervised manner.

The problem dealt with in this chapter is closely related to the work of Saxena *et al.* [2007], who utilize Markov random fields (MRFs) for reconstructing dense depth maps from single monocular images. An alternative approach that predicts 2D range scans based on using reinforcement learning techniques has been presented by Michels *et al.* [2005]. Compared to these methods, our Gaussian process formulation provides the predictive uncertainties for the depth estimates directly, which is not straightforward to achieve in an MRF model. Hoiem *et al.* [2007] developed an approach to monocular scene reconstruction based on local features combined with global reasoning. Whereas Han and Zhu [2003] presented a Bayesian method for reconstructing the 3D geometry of wire-like objects, Delage *et al.* [2005] introduced an MRF model on orthogonal plane segments to recover the 3D structure of indoor scenes.

Ewerth *et al.* [2007] extract depth cues from monocular image sequences in order to facilitate image retrieval from video sequences. Their major cue for feature extraction is the motion parallax. Thus, their approach assumes translational camera motion and a rigid scene.

In own previous work [Plagemann *et al.*, 2007b] (see also Chapter 7), we applied Gaussian processes to improve sensor models for laser range finders. Here, the goal is to exchange the highly accurate and reliable laser measurements by noisy and ambiguous vision features.

As mentioned above, one potential application of the approach described in this chapter is to learn occupancy grid maps. This type of maps and an algorithm to update such maps based on ultrasound data has been introduced by Moravec and Elfes [1985]. In the past, different approaches to learn occupancy grid maps from stereo vision have been proposed [Thrun *et al.*, 1998, Sabe *et al.*, 2004]. If the positions of the robot are unknown during the mapping process, the entire task turns into the so-called simultaneous localization and mapping (SLAM) problem. Vision-based techniques have been proposed by Elinas *et al.* [2006] and Davision *et al.* [2007] to solve this problem. In contrast to the mapping approach presented in this chapter, these techniques mostly focus on landmark-based representations.

8.5 Conclusion

We presented a novel approach to predicting range functions from single images recorded with a monocular camera. Our model is based on a Gaussian process model for regression, utilizing edge-based features extracted from the image or, alternatively, using the PCA to find a low-dimensional representation of the visual input in an unsupervised manner. Both models outperform the optimized individual features.

We furthermore showed in experiments with a real robot that the range predictions are accurate enough to feed them into an extended mapping algorithm for predictive range distributions and that the resulting maps are comparable to maps obtained with infrared or sonar sensors.

Part III

Modeling Input-Dependent Smoothness

Chapter 9

Terrain Modeling as a Regression Problem

We formalize terrain modeling as a probabilistic regression problem and derive a solution based on Gaussian processes. As an integral part of our model, we utilize nonstationary covariance functions, which adapt the smoothing behavior locally, balancing smoothing against the preservation of structural features like edges and corners.

The modeling of terrain surfaces has been studied widely across different research areas like the geosciences or robotics. Important applications in the latter case include mobile robotics for agriculture, planetary exploration, search and rescue, or surveillance. In these domains, accurate and dense models of the three-dimensional environment enable the robot to estimate the cost of traversability of locations, to plan its path to a goal location, or to localize itself using sensor measurements. Building a digital terrain model involves transforming a set of sensory inputs, typically a 3D point cloud or raw range sensor readings, to a function mapping 2D location coordinates to elevation values. While geological applications often operate on a larger spatial scale, in which *local* terrain features can be neglected, autonomous robots greatly rely on distinct structural features like edges or corners to guide navigation, localization, or terrain segmentation. We therefore have two, at the first glance contradicting requirements for terrain models in robotics: First, raw sensory data needs to be smoothed in order to remove noise and to be able to perform elevation predictions at all locations and, second, discontinuities need to be preserved as they are important features for important tasks. Consider, for example, an autonomous

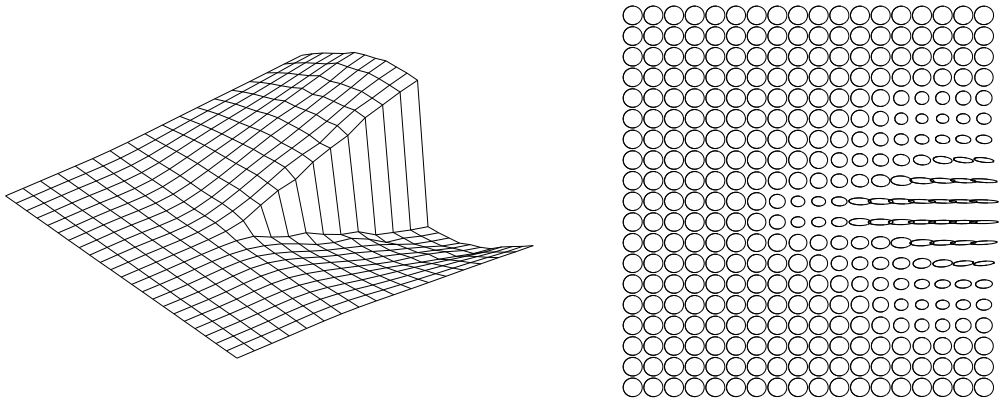


Figure 9.1: Left: A hard, synthetic regression problem. The continuous parts should be smoothed without removing the strong edge feature. Right: Our approach achieves this by adapting local kernels to the terrain data.

car driving in urban terrain. The street itself should be reconstructed as a smooth surface to enable the path planning algorithm to find a smooth trajectory while the step to the sidewalk should be as sharp as possible to robustly identify it as a non-traversable obstacle.

In this chapter, we present a novel terrain modeling approach based on an extended Gaussian process formulation. Our model uses nonstationary covariance functions as proposed by Paciorek and Schervish [2004] to allow for local adaptation of the regression kernels to the underlying structure. This adaptation is achieved by iteratively fitting the local kernels to the characteristics of the data using local gradient features and the local marginal data likelihood (see the right diagram in Fig. 9.1 for an illustration). Indeed, this idea is akin to adaptive image smoothing studied in computer vision, where the task is to achieve de-noising of an image without reducing the contrast of edges and corners [Takeda *et al.*, 2006, Middendorf and Nagel, 2002]. Although these approaches from the computer vision literature are not specifically designed for dealing with a varying density of data points or with potential gaps to fill, they nevertheless served as an inspiration for our kernel adaptation approach.

The chapter is structured as follows. In Sec. 9.1, we formalize the terrain modeling problem using Gaussian processes and introduce our approach to nonstationary adaptive regression in Sec. 9.3. Section 9.4 then presents our experimental results on real and simulated terrain data sets.

9.1 Digital Terrain Modeling

Data for building 3D models of an environment can be acquired from various sources. In robotics, laser range finders are popular sensors as they provide precise, high-frequency

measurements at a high spatial resolution. Other sensors include on-board cameras, which are chosen because of their low weight and costs, or satellite imagery. The latter covers larger areas and is used for, e.g., guiding unmanned areal vehicles (UAVs) or autonomous cars. After various preprocessing steps, the raw measurements are typically represented as 3D point clouds or are transformed into a 3D occupancy grid or elevation map [Bares *et al.*, 1989]. In this work, we introduce a technique for constructing continuous, probabilistic elevation map models from data points, that yield predictive distributions for terrain elevations at arbitrary input locations.

The terrain modeling problem can be formalized as follows. Given a set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of n location samples $\mathbf{x}_i \in \mathbb{R}^2$ and the corresponding terrain elevations $y_i \in \mathbb{R}$, the task is to build a model for $p(y_*|\mathbf{x}_*, \mathcal{D})$, i.e., the predictive distribution of elevations y_* at new input locations $\mathbf{x}_*$. This modeling task is hard for several reasons. First, sensor measurements are inherently affected by noise, which an intelligent model should be able to reduce. Second, the distribution of available data points is typically far from uniform. For example, in proximity to the sensor location there are usually more samples than in areas farther away. Third, small gaps in the data should be filled with high confidence while more sparsely sampled locations should result in higher predictive uncertainties. To illustrate the last point, consider an autonomous vehicle navigating in off road terrain. Without filling small gaps, even single missing measurements may lead to the perception of an un-traversable obstacle and consequently the planned path might differ significantly from the optimal one. On the other hand, the system should be aware of the increased uncertainty when filling larger gaps to avoid overconfidence at these locations. As a last non-trivial requirement, the model should preserve structural elements like edges and corners as they are important features for various applications including path planning or object recognition.

In the following, we propose a model that accommodates for all of the above-mentioned requirements using Gaussian process regression. In order to deal with the preservation of structural features like edges and corners, we employ nonstationary covariance functions as introduced by Paciorek and Schervish [2004] and present a novel approach to local kernel adaptation based on gradient features and the local marginal data likelihood.

9.2 Nonstationary Gaussian Process Regression

The standard Gaussian process model for regression as introduced in Sec. 2.2 already accounts for three of the requirements discussed in the previous section, namely de-noising, dealing with non-uniform data densities, and providing predictive uncertainties. Proper adaptation to local structure, however, cannot be achieved by using *stationary* covariance functions, such as defined in Eq. (2.3) or in Eq. (2.4). These depend only on the *differences*

between input locations and, thus, apply the same smoothness prior at every location of the input space. In practice, this significantly weakens important features like edges or corners. The left diagram of Fig. 9.1 depicts a synthetic data set which contains homogenous regions which should be smoothed, but also a sharp edge that has to be preserved. Our model, which is detailed in the next section, addresses this problem by adapting a nonstationary covariance function to the local terrain properties. Our approach is able to solve situations such as the one depicted in Fig. 9.1, in which a continuous surface is "cut" by a sharp edge from its center to the boundary. Note that such situations can only sub-optimally be dealt with by approaches based on segmenting the terrain into continuous sub-models as they would inevitably introduce unjustified discontinuities at those segment boundaries passing through the flat parts of the function.

A powerful model for building *nonstationary* covariance functions from arbitrary stationary ones has been proposed by Paciorek and Schervish [Paciorek and Schervish, 2004]. For the Gaussian kernel, their nonstationary covariance function takes the simple form

$$k_{NS}(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{\Sigma}_i|^{\frac{1}{4}} |\mathbf{\Sigma}_j|^{\frac{1}{4}} \left| \frac{\mathbf{\Sigma}_i + \mathbf{\Sigma}_j}{2} \right|^{-\frac{1}{2}} \cdot \exp\left[-(\mathbf{x}_i - \mathbf{x}_j)^\top \left(\frac{\mathbf{\Sigma}_i + \mathbf{\Sigma}_j}{2} \right)^{-1} (\mathbf{x}_i - \mathbf{x}_j) \right], \quad (9.1)$$

where each input location $\mathbf{x}_i$ is assigned an individual Gaussian kernel matrix $\mathbf{\Sigma}_i$ and the covariance between two targets y_i and y_j is calculated by averaging between the two individual kernels at the input locations $\mathbf{x}_i$ and $\mathbf{x}_j$. In this way, the local characteristics at both locations influence the modeled covariance of the corresponding target values. In this model, each kernel matrix $\mathbf{\Sigma}_i$ is internally represented by its eigenvectors and eigenvalues. Paciorek and Schervish build a hierarchical model by placing additional Gaussian process priors on these kernel parameters and solve the integration using Markov Chain Monte Carlo (MCMC) sampling. While the model presented in [Paciorek and Schervish, 2004] provides a flexible and general framework, it is, as also noted by the authors, computationally demanding and clearly not feasible for the real world terrain data sets that we are aiming for in this work. As a consequence, we propose to model the kernel matrices in Eq. (10.1) as independent random variables that are initialized with the learned kernel of the corresponding stationary model and then adapted iteratively to the local structure of the given terrain data. Concretely, we assign to every input location $\mathbf{x}_i$ from the training set $\mathcal{D}$ a local kernel matrix $\mathbf{\Sigma}_i$, which in turn is represented by one orientation parameter and two scale parameters for the length of the axes. Given these parameters, the evaluation of Eq. (10.1) is straightforward. In the following section, we will discuss in detail, how the kernel matrices $\mathbf{\Sigma}_i$ can be adapted to the local structure of the terrain.

9.3 Local Kernel Adaptation

The problem of adapting smoothing kernels to local structure has been studied intensively in the computer vision community. It is therefore not surprising that, although image processing algorithms are typically restricted to dense and uniformly distributed data, we can use findings from that field as an inspiration for our terrain adaptation task. Indeed, Middendorf and Nagel [2002] present a technique for iterative kernel adaptation in the context of optical flow estimation in image sequences. Their approach builds on the concept of the so-called gray-value structure tensor (GST), which captures the local structure of an image or image sequence by building the locally weighted outer product of gray-value gradients in the neighborhood of the given image location. Analogously, we define the *elevation structure tensor* (EST) for a given location $\mathbf{x}_i$ as

$$\mathrm{EST}(\mathbf{x}_i) := \overline{\nabla f (\nabla f)^\top}(\mathbf{x}_i) \;, \tag{9.2}$$

where $f(\mathbf{x})$ denotes the terrain elevation at a location $\mathbf{x}$ and $\overline{\cdot}$ stands for the operator that builds a locally weighted average of its argument according to the kernel $\mathbf{\Sigma}_i$. For two-dimensional $\mathbf{x} = (x_1, x_2)^\top$, Eq. (9.2) calculates the locally weighted average of the outer product of $\nabla f(\mathbf{x}) = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2})^\top$. This local elevation derivative can be estimated directly from the raw elevation samples in the neighborhood of the given input location $\mathbf{x}_i$. We cope with the noise stemming from the raw data by averaging over the terrain gradients in the local neighborhood.

Equation (9.2) yields a tensor, representable as a 2×2 real-valued matrix, which describes how the terrain elevation changes in the local neighborhood of location $\mathbf{x}_i$. To get an intuition, what $\mathrm{EST}(\mathbf{x}_i)$ encodes and how this can guide the adaptation of the local kernel $\mathbf{\Sigma}_i$, consider the following situations. Let λ_1 and λ_2 denote the eigenvalues of $\mathrm{EST}(\mathbf{x}_i)$ and β be the orientation angle of the first eigenvector. If $\mathbf{x}_i$ is located in a flat part of the terrain, the elevation gradients ∇f are small in the neighborhood of $\mathbf{x}_i$. This results in two equally small eigenvalues of $\mathrm{EST}(\mathbf{x}_i)$. In contrast, if $\mathbf{x}_i$ was located in an ascending part of the terrain, the first eigenvalue of $\mathrm{EST}(\mathbf{x}_i)$ would be clearly greater than the second one and the orientation β would point towards the strongest ascent.

Intuitively and as discussed in more detail by Middendorf and Nagel [2002], the kernel $\mathbf{\Sigma}_i$ describing the extent of the local environment of $\mathbf{x}_i$ should be set to the inverse of $\mathrm{EST}(\mathbf{x}_i)$. In this way, flat areas are populated by large, isotropic kernels, while sharp edges have long, thin kernels oriented according to the edge orientation. Corner structures, having strong elevation gradients in all dimensions, result in relatively small local kernels. To prevent unrealistically large kernels, Middendorf and Nagel describe how this inversion

can be bounded to yield kernels, whose standard deviations lie between given values σ_{min} and σ_{max}. Based on their findings, we give three concrete local adaptation rules that have been compared in our experimental evaluation. To simplify notation, we introduce $\overline{\lambda_k} = \lambda_k/(\lambda_1 + \lambda_2)$, $k = 1, 2$, and the re-parameterization

$$\mathbf{\Sigma}_i = \mathbf{R}^{-\top} \begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \mathbf{R}^{-1} \ , \tag{9.3}$$

where α_1 and α_2 scale in orthogonal directions and $\mathbf{R}$ is a rotation matrix specified by the orientation angle θ.

1. *Direct Inverse Adaptation*: $\mathbf{\Sigma}_i = \mathrm{EST}(\mathbf{x}_i)^{-1}$

2. *Bounded Linear Adaptation*:

$$\alpha_k = \overline{\lambda_k}\ \sigma_{min}^2 + (1 - \overline{\lambda_k})\ \sigma_{max}^2 \quad , k = 1, 2$$

3. *Bounded Inverse Adaptation*:

$$\alpha_k = \frac{\sigma_{max}^2 \sigma_{min}^2}{\overline{\lambda_k}\ \sigma_{max}^2 + (1 - \overline{\lambda_k})\ \sigma_{min}^2} \quad , k = 1, 2$$

The two *bounded* adaptation procedures prevent unrealistically small and large kernels. The *Bounded Inverse* strongly favors the larger eigenvalue dimension and produces more pronounced kernels (larger difference between semi-axes) while *Bounded Linear* tends to produce more balanced and larger kernels. This is why *Bounded Linear* performs better in the presence of sparse data as it is less vulnerable to overfitting. For the experiments reported on below, the bounds σ_{min} and σ_{max} were optimized empirically on independent test data.

So far, we have described how to perform one local adaptation step for an arbitrary kernel $\mathbf{\Sigma}_i$. As the complete learning and adaptation procedure, which is summarized in Alg. 2, we propose to assign to each input location $\mathbf{x}_i$ of the training set $\mathcal{D}$ a kernel matrix $\mathbf{\Sigma}_i$, which is initialized with a global parameter vector $\mathbf{\Theta}$, that in turn has been learned using standard GP learning with the corresponding stationary covariance function. The local kernels are then iteratively adapted to the elevation structure of the given terrain data set until their parameters have converged. To quickly adapt the kernels at locations where the regression error is high (relative to the given training data set), we propose to make the adaptation speed for each $\mathbf{\Sigma}_i$ dependent on the local data fit $\mathrm{df}(\mathbf{x}_i)$, which is the normalized observation likelihood of the corresponding y_i from the training set relative to the current

predictive distribution, and the kernel complexity approximated as $c_i = 1/|\boldsymbol{\Sigma}_i|$. Both quantities are used to form a learning rate parameter calculated by means of a modified sigmoid function, $\eta_i = \text{sigmoid}(-\text{df}(\mathbf{x}_i) \cdot c_i; \boldsymbol{\delta})$, where the additional parameters $\boldsymbol{\delta}$ are determined empirically. Intuitively, we get a high adaptation speed when the data fit relative to the kernel size is small. Algorithm 2 summarizes the adaptation procedure.

9.4 Experimental Results

The goals of the experimental evaluation presented in this section are (a) to show that our terrain modeling approach is indeed applicable to real data sets, (b) that our model is able to remove noise while at the same time preserving important structural features, and (c) that our model yields more accurate and robust elevation predictions at sparsely sampled input locations than an alternative approach to this problem.

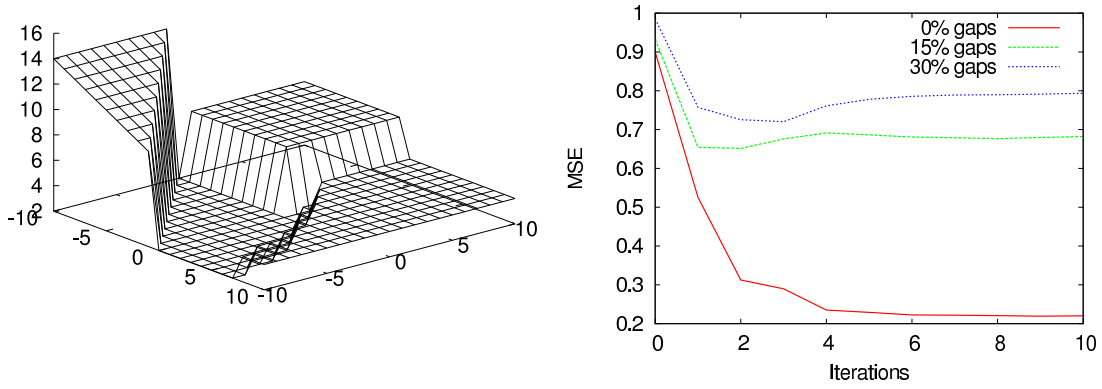
As an evaluation metric, we use the mean squared error $\text{MSE}(\mathcal{X}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - y_i^*)^2$ of predictions y_i^* relative to ground truth elevations y_i on a set of input locations $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^m$.

9.4.1 Evaluation on Artificial Terrain Data

The first set of experiments was designed to quantify the benefits of local kernel adaptation and to compare the different adaptation rules. As a test scenario, we took the artificial terrain data set depicted in Fig. 9.2 consisting of 441 data points containing uniform regions as well as sharp edges and corners, which are hard to adapt to locally. Note that the edge between the lowest and the second lowest plateau has a curvature and that three different height levels can be found in the local neighborhood of the corner in the middle of the diagram. We set $\sigma_{min} = 0.001$ and $\sigma_{max} = 5.0$ for the bounded adaptation rules. To generate training data sets for the different experiments reported on here, we added white noise of a varying standard deviation σ to the true terrain elevations and randomly removed a portion of the samples to be able to assess the model's predictive abilities.

Algorithmus 2 Local Kernel Adaptation

```
Learn global parameters Θ for the stationary squared exponential covariance function.
Initialize all local kernels Σ_i with Θ.
while not converged do
  for all Σ_i do
    Estimate the local learning rate η_i
    Estimate EST(x_i) according to Σ_i
    Σ_i* ← ADAPT(EST(x_i))
    Σ_i ← η_i Σ_i* + (1 − η_i) Σ_i
  end for
end while
```



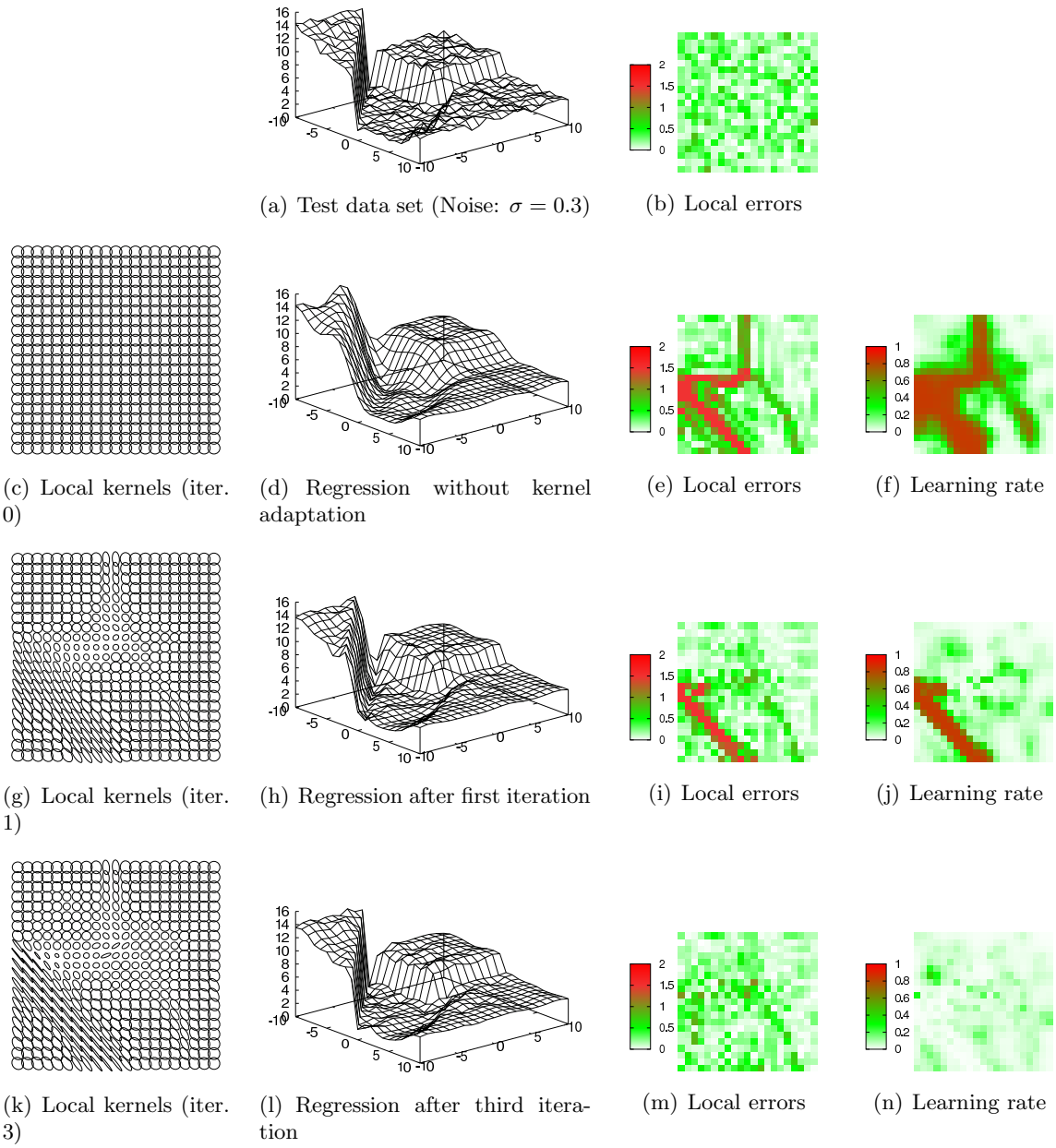
(a) Terrain (b) Convergence

Figure 9.2: (a): An artificial terrain data set used in the experimental evaluation, that exhibits several local features that are hard to adapt to. Test data sets are generated by adding white noise and randomly removing a portion of the data points. (b): The mean squared error (MSE) of predicted elevations converges with an increasing number of adaptation steps. Iteration 0 gives the MSE for the learned standard GP. Values are averaged over ten independent runs.

Figure 9.3 visualizes a complete adaptation process for the case of a data set generated using a noise rate of $\sigma = 0.3$. On average, a single iteration per run took 44 seconds on this data set using a PC with a 2.8 GHz CPU and 2 GB of RAM. Figures 9.3(c)-9.3(f) show the results of standard GP regression which places the same kernels at all input locations. While this leads to good smoothing performance in homogeneous regions, the discontinuities within the map are also smoothed as can be seen from the absolute errors in the third column. Consequently, those locations get assigned a high learning rate—plotted in the right column—used for local kernel adaption.

The first adaptation step leads to the results depicted in Figures 9.3(g)-9.3(j). It is clearly visible, that the steps and corners are now better represented by the regression model. This has been achieved by adapting the kernels to the local structure, see the first column of this row. Note, how the kernel sizes and orientations reflect the corresponding terrain properties. Kernels are oriented along discontinuities and are small in areas of strongly varying elevation. In contrast, they have been kept relatively large in homogeneous regions. After three iterations, the regression model has adapted to the discontinuities accurately while still de-noising the homogeneous regions [Fig. 9.3(k)-9.3(n)]. Note, that after this iteration, the local learning rates have all settled at low values.

Figure 9.2 gives the convergence behavior of our approach using the *Bounded Linear* adaptation rule in terms of the mean squared prediction error for different amounts of points removed from the noisy data set. After around 6 iterations, the errors have settled close to their final value.



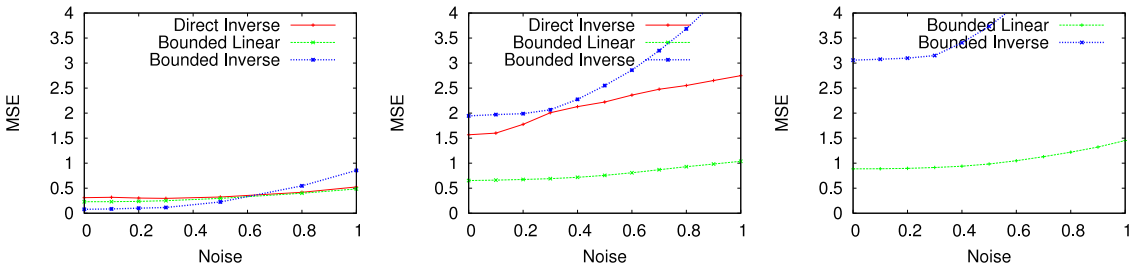
(a) Test data set (Noise: $\sigma = 0.3$) (b) Local errors

(c) Local kernels (iter. 0) (d) Regression without kernel adaptation (e) Local errors (f) Learning rate

(g) Local kernels (iter. 1) (h) Regression after first iteration (i) Local errors (j) Learning rate

(k) Local kernels (iter. 3) (l) Regression after third iteration (m) Local errors (n) Learning rate

Figure 9.3: The local kernel adaptation process on an artificial terrain data set: the original data set, depicted in Figure 9.2, exhibits several local features that are hard to adapt to. The test data set (a) was generated by adding white noise, resulting in the errors shown in (b). The second row of diagrams gives information about the initialization state of our adaptation process, i.e., the results of standard GP learning and regression. The following two rows depict the results of our approach after the first and after the third adaptation iteration, respectively. In the first column of this figure, we visualize the kernel dimensions and orientations after the corresponding iteration. The second column depicts the predicted means of the regression. The third column gives the absolute errors to the known ground truth elevations and the right-most column gives the resulting learning rates η_i for the next adaptation step resulting from the estimated data likelihoods.



(a) All data points given.

(b) 15% of the data points removed.

(c) 30% of the data points removed.

Figure 9.4: Prediction accuracy for the scenario depicted in Figure 9.3 with (a) all data points available, (b) 15% of the data points randomly removed and (c) 30% randomly removed. Each figure plots the mean squared error of elevation predictions for a varying level of added white noise. The values are averaged over 10 independent runs per configuration. (In the case of (c), the error of *Direct Inverse* was always greater than 4.0).

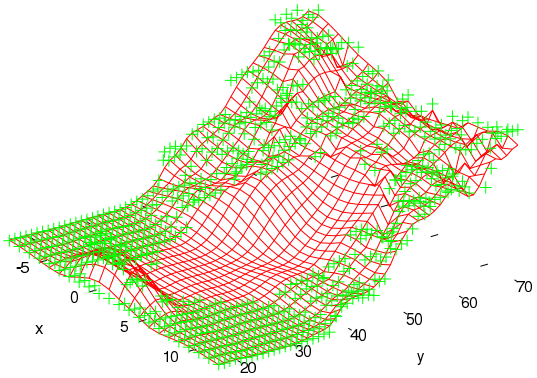
In a different set of experiments, we investigated the prediction performance of our approach for all three adaptation rules presented in Sec. 9.3. For this experiment, we added white noise of a varying noise level to the artificial terrain given in Fig. 9.2. The diagrams in Fig. 9.4 give the results for different amounts of points removed from the noisy data set. When no points are removed from the test set, the *Bounded Inverse* adaptation rule performs best for small noise values. For large noise values, *Bounded Linear* and *Direct Inverse* achieve better results. In the case of 15% and 30% data points removed, *Direct Inverse* and *Bounded Inverse* are not competitive. In contrast, *Bounded Linear* still achieves very good results for all noise levels. This is due to the fact that *Bounded Linear* adaptation is least "aggressive" when fitting to the data points and it is thus able to span larger gaps.

Thus, *Bounded Linear* produces reliable predictions for all tested noise rates and data densities. This finding was supported by experiments on other real data sets not presented here.

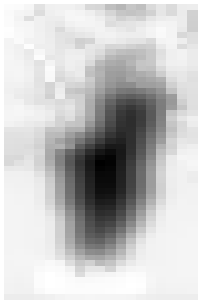
9.4.2 Evaluation on Real Terrain Data

In order to demonstrate the usefulness of our approach on real data sets, we acquired a set of 3D scans of a scene using a mobile robot equipped with a laser range finder, see Figure 9.5(a). We compared our prediction results to an approach from the robotics literature [Früh *et al.*, 2005], which has been applied successfully to the problem of 3D mapping urban areas. We employed the *Bounded Linear* adaptation procedure for our learning algorithm where we set $\sigma_{min} = 0.25$ and $\sigma_{max} = 4.0$. Figure 9.5 gives the results of this experiment. An obstacle, in this case a person, is placed in front of the robot and

(a) The first test scenario



(b) Observations (points) and predicted means (lines)



(c) Predictive uncertainties (white: zero)

Figure 9.5: (a): A real-world scenario, where a person blocks the robot's view on an inhomogeneous and sloped terrain. (b): The raw data points as well as the predicted means of our adapted nonstationary regression model. (c): Our model also yields the predictive uncertainties for the predicted elevations (bright: certain, dark: uncertain).

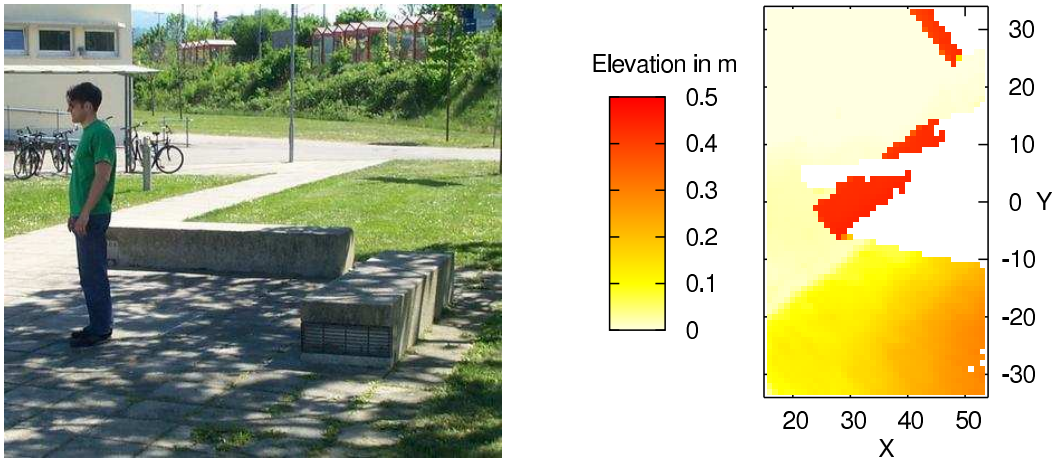
thus occludes the sloped terrain behind.

We evaluated our approach for the situation depicted in the figure as well as for three similar ones and compared its prediction accuracy to the approach of Früh *et al.* [2005], who perform horizontal linear interpolation orthogonally to the robot's view. These scenarios used are actually rather easy ones for [Früh *et al.*, 2005], as the large gaps can all be filled orthogonally to the robot's view, which is not the case in general. To estimate the kernels at unseen locations, we built a weighted average over the local neighborhood with an isotropic two-dimensional Gaussian with a standard deviation of 3 which we had found to produce the best results. Table 9.1 gives the results. In all four cases, our approach achieved higher prediction accuracies, reducing the errors by 30% to 70%. Figure 9.5(b) depicts the predictions of our approach in one of the situations. In contrast to Früh *et al.*, our model is able to also give the predictive uncertainties. These variances are largest in the center of the occluded area as can be seen in Figure 9.5(c).

In a second real-world experiment illustrated in Fig. 9.7, we investigated the ability of our

Table 9.1: Prediction performance in terms of MSE relative to a second, not occluded scan.

Scenario	Linear Interp. [Früh *et al.*, 2005]	Adapted GP	Improvement
1 (Fig. 9.5)	0.116	0.060	48.3%
2	0.058	0.040	31.0%
3	0.074	0.023	69.9%
4	0.079	0.038	51.9%



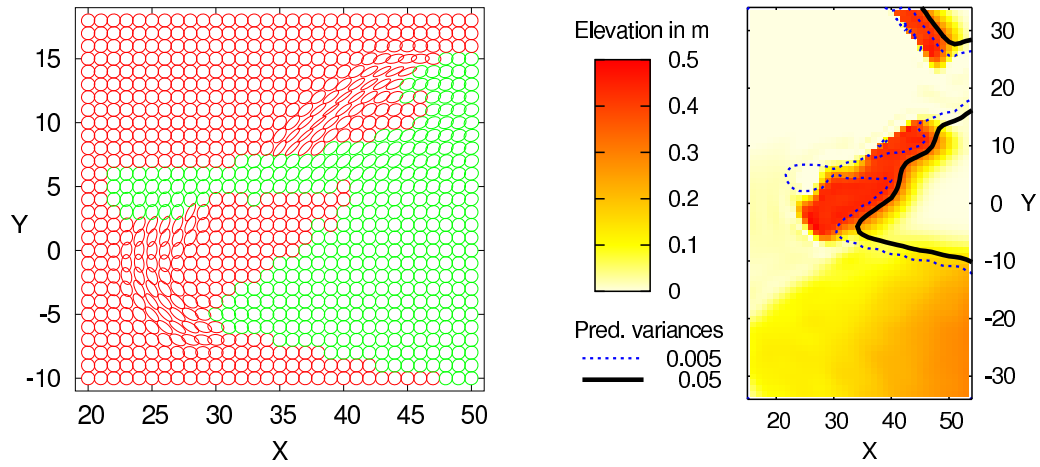
(a) The second test scenario

(b) Observed elevations

Figure 9.6: A real-world scenario where a person blocks the robot's view on a stone block, i.e., a sharp linear discontinuity (a). Figure (b) visualizes the kernels that have adapted to the observed block edges illustrated in (c). Figure (d) illustrates the predicted terrain elevations and two contour lines for two different predictive uncertainty thresholds.

terrain model approach to preserve and predict sharp discontinuities in real terrain data. We positioned the robot in front of a rectangular stone block such that the straight edges of the block run diagonally to the robot's line of view. A person stood in between the robot and the block, thereby occluding parts of the block and of the area in front of it. This scenario is depicted in Fig. 9.6(a). The task is to recover the linear structure of the discontinuity and fill the occluded area consistent with the surrounding terrain elevation levels. The adaptation procedure converged already after two iterations. The learned kernel structure, illustrated in Fig. 9.7(a), enables the model to represent the stone blocks correctly as can be seen from the predicted elevations visualized in Fig. 9.7(b). This figure also illustrates the uncertainties in these predictions, corresponding to the variances of the predictive distributions, by means of two contour lines. This indicates that a mobile robot would be relatively certain about the block structure within in the gap region although not having observed it directly. On the other hand, it would also be aware that it cannot rely upon the model predictions in the occluded areas beyond the blocks: There are no observations within a reasonable distance and, thus, the predictive variances are large.

To show that our approach is applicable to large, real-world problems, we have tested it on a large dataset recorded at the University of Freiburg campus The raw terrain data was preprocessed, scan-corrected, and then represented in a multi-level surface map with a cell size of 10 cm $\times$ 10 cm. The scanned area spans approximately 299 by 147 meters. For simplicity, we only considered the lowest data points per location, i.e. we removed



(a) Adapted kernels at observed (red) and occluded locations (green)

(b) Predicted elevations

Figure 9.7: A real-world scenario where a person blocks the robot's view on a stone block, i.e., a sharp linear discontinuity (a). Figure (b) visualizes the kernels that have adapted to the observed block edges illustrated in (c). Figure (d) illustrates the predicted terrain elevations and two contour lines for two different predictive uncertainty thresholds.

overhanging structures like tree tops or ceilings. The resulting test set consists of 531 920 data points. To speed up computations, we split this map into 542 overlapping sub-maps. This is possible without loss of accuracy as we can assume compact support for the local kernels involved in our calculations (as the kernel sizes in our model are bounded). We randomly removed 20% of the data points per sub-map. A full run over the complete data set took about 50 hours. Note that the computational complexity can be reduced substantially by exploiting the sparsity of our model (due to the bounded kernels) and by introducing additional sparsity using approximative methods. This will be addressed in the following chapters. Table 9.2 gives the results of this experiment for the different adaptation rules. *Bounded Linear* and *Bounded Inverse* outperform the *Standard GP* model where kernels are not adapted, while *Direct Inverse* is not competitive. Together with the results of the other experiments, this leads to the conclusion that *Bounded Linear* is an adequate choice for synthetic and real-world scenarios.

Table 9.2: Prediction performance on a large campus environment.

Adaptation procedure	**MSE**
Standard GP	0.071
Direct Inverse	0.103
Bounded Linear	0.062
Bounded Inverse	0.059

9.5 Related Work

A broad overview of methods used for modeling terrain data is given by Hugentobler [2004]. Elevation maps have been used as an efficient data structure for representing dense terrain data [Bares *et al.*, 1989, Pfaff and Burgard, 2005] and have been extended later to multi-level probabilistic surface maps [Triebel *et al.*, 2006]. Früh *et al.* [2005] present an approach to filling local gaps in 3D models based on local linear interpolation. As their approach has yielded promising results in city mapping applications, we compare its modeling accuracy to our approach in Sec. 9.4.

Classical approaches for dealing with nonstationarity include input-space warping [Sampson and Guttorp, 1992, Schmidt and O'Hagan, 2003] and hierarchical modeling using local kernels [Paciorek and Schervish, 2004]. The latter approach provides the general framework for this work. Discontinuities in wind fields have been dealt with by Cornford *et al.* [1999]. They place auxiliary GPs along the edge on both sides of the discontinuity. These are then used to learn GPs representing the process on either side of the discontinuity. In contrast to our work, they assume a parameterized segmentation of the input space, which appears to be disadvantageous in situations such as depicted in Fig. 9.1 and on real-world terrain data sets. The problem of adapting to local structure has also been studied in the computer vision community. Takeda *et al.* [2006] perform nonparametric kernel regression on images. They adapt kernels according to observed image intensities. Their adaptation rule is thus based on a nonlinear combination of both spatial and intensity distance of all data points in the local neighborhood. Based on singular value decompositions of intensity gradient matrices, they determine kernel modifications. Middendorf and Nagel [2002] propose an alternative kernel adaptation algorithm. They use estimates of gray value structure tensors to adapt smoothing kernels to gray value images.

9.6 Conclusion

In this chapter, we proposed an adaptive terrain modeling approach that balances smoothing against the preservation of structural features. Our method uses Gaussian processes with nonstationary covariance functions to locally adapt to the structure of the terrain data. In experiments on synthetic and real data, we demonstrated that our adaptation procedure produces reliable predictions in the presence of noise and is able to fill gaps of different sizes. Compared to a state-of-the-art approach from the robotics literature, we achieve a prediction error reduced by approximately 30% to 70%.

Chapter 10

Terrain Modeling for Legged Robot Locomotion

We make our adaptive terrain model applicable to large data sets and discuss foothold selection and path planning for legged robots. The system was implemented and tested with a quadruped robot equipped with a light-weight laser scanner on the tasks of autonomously sensing a rough terrain surface, planning a path, and executing it online.

Legged robots have a number of advantages over traditional wheeled systems, such as their ability to move in rough and unstructured terrain or to step over obstacles. Without accurate knowledge of the terrain, however, these advantages cannot be realized as motion planning requires means of estimating the stability of configurations and the traversability of locations. Acquiring and representing models of rough terrain is a challenging task. First of all, terrains are defined over a continuous space such that the space of all models has in principle infinitely many dimensions. Discretizing this continuous space either results in models of enormous size or in a loss of information that in turn may lead to the selection of statically unstable and kinematically infeasible configurations of the robot. Furthermore, we must rely on the robot's noisy sensors to gather information about the world, which requires statistical inference in the continuous and high-dimensional space of the model. Finally and most importantly, we wish to be able to deal with a varying data density, to balance the reduction of noise against the preservation of discontinuities, and to be able to make sensible predictions about unseen parts of the terrain.

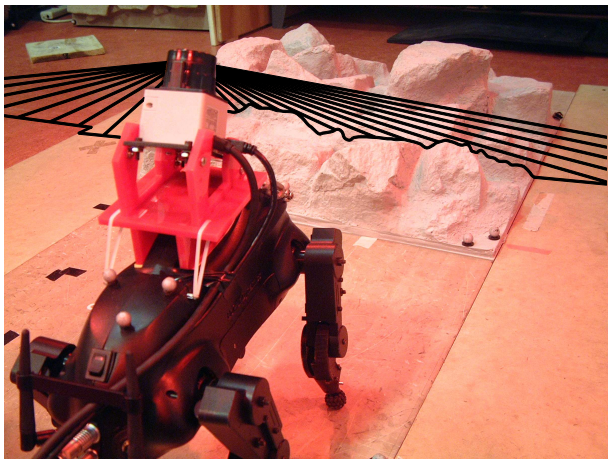


Figure 10.1: Our quadruped robot is equipped with a laser sensor in order to observe its local surrounding. By bending its leg joints, it is able to acquire dense 3D scans of the environment.

In this chapter, we extend the probabilistic, nonstationary terrain modeling approach introduced in Chapter 9 towards efficient learning and inference using a decomposition of the model into smaller, overlapping sub-models. As a result, our model is substantially more efficient and accurate than those based on standard Gaussian processes and also outperforms grid-based approaches to elevation mapping in several aspects. As a second contribution of this chapter, we apply the terrain model to the task of terrain mapping with a quadruped robot, specifically the Boston Dynamics LittleDog (see Fig. 10.1). The robot faces the tasks of actively sensing the rough terrain, building a model, planning a path, and executing it autonomously. It is geometrically impossible for the robot to directly sense every point in the terrain from a single pose since the sensor has a limited field of view and the terrain self-occludes in multiple locations. The ability to predict terrain elevations at unseen locations, however, is highly beneficial for planning. Although it is difficult to express *a priori* how the local structure can predict the unobserved terrain heights, we demonstrate how this relationship can be learned from experience. Finally, we also show how our locally adaptive GP model allows us to select safe foothold locations and to plan a path to a goal location. The terrain model we present was implemented and tested with a real robot. Figure 10.2 shows screen-shots of our controller application in some example situations, i.e. during scan acquisition, terrain adaptation, and path planning.

We proceed as follows. First, we restate terrain modeling as a nonparametric regression problem and discuss its extension to large datasets. Then, we show how to generate and execute locomotion plans for a quadruped robot in Sec. 10.2 and present our empirical evaluation on a real quadruped robot equipped with a laser range finder in Sec. 10.3 and compare our model to popular grid-based alternatives.

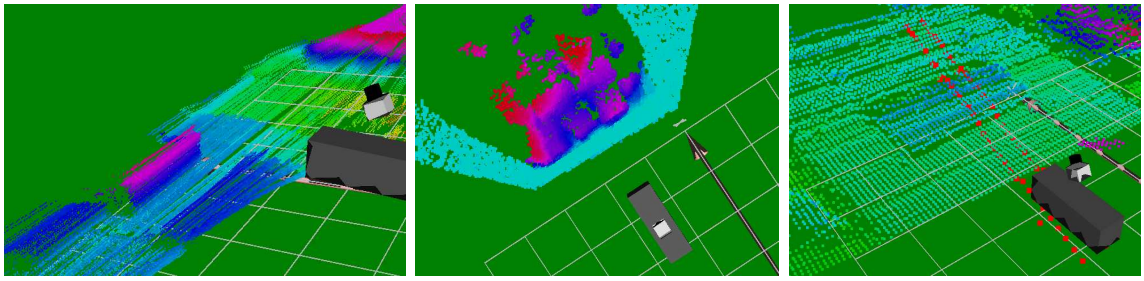


Figure 10.2: Online visualization using the controller application of our implemented system in three different scenarios. Left: The robot bends its legs to acquire a 3D scan. Middle: It learns a probabilistic terrain regression model. Right: It plans a path to a goal location. The terrain elevations are color coded, ranging from green/blue (lowest) to pink/red (highest).

10.1 Large Nonstationary Models

Traversable surfaces can be characterized by a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}, f(\mathbf{x}) = y$, where $\mathbf{x}$ indexes a location in the 2D plane and y denotes the corresponding terrain elevation in the direction opposing the gravity vector. Elevation grids are a popular way of representing such functions and learning them from a set of elevation samples $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$. In the grid formulation, the space of locations $\mathbf{x}$ is discretized, such that each grid cell is assigned a constant elevation value [Pfaff and Burgard, 2005] or a parametric function $p(y)$ is fitted to the distribution of its elevation values [Triebel *et al.*, 2006]. As described in the previous chapter, we take a different approach by placing a Gaussian process prior on the elevation function f and by applying a nonstationary covariance function that is adapted to the data locally. Recalling Eq. (10.1) from the previous chapter, it takes the form

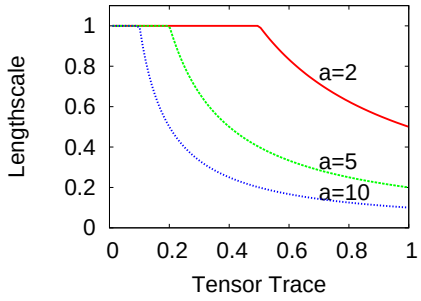
$$k_{NS}(\mathbf{x}_i, \mathbf{x}_j) = |\boldsymbol{\Sigma}_i|^{\frac{1}{4}} |\boldsymbol{\Sigma}_j|^{\frac{1}{4}} \left| \frac{\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j}{2} \right|^{-\frac{1}{2}} \cdot \exp \left[-(\mathbf{x}_i - \mathbf{x}_j)^\top \left(\frac{\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j}{2} \right)^{-1} (\mathbf{x}_i - \mathbf{x}_j) \right] \quad (10.1)$$

As in the previous chapter, we adapt the local kernels $\boldsymbol{\Sigma}_i$ using terrain gradient information. For an input location $\mathbf{x}_i$, we estimate the gradient

$$\nabla f(\mathbf{x}_i) = \left(\frac{\partial f}{\partial([\mathbf{x}_i]_1)}, \frac{\partial f}{\partial([\mathbf{x}_i]_2)} \right)^\top$$

from elevation observations in the local neighborhood. We then calculate the trace of the elevation structure tensor (EST)

$$T(\mathbf{x}_i) = \operatorname{trace} \left(\overline{\nabla f (\nabla f)^\top}(\mathbf{x}_i) \right) \quad (10.2)$$



Missing data	WA	HGP
15%	0.437	0.104
30%	0.531	0.168
50%	0.735	0.406

Figure 10.3: Left: The parametric function $\ell(\mathbf{x}_i)$ maps terrain slope to length-scales. Right table: Placing a *hyper GP* (HGP) over the latent length-scales reduces the mean squared error (MSE) of predictions w.r.t. to *weighted averaging* (WA) for different levels of missing data.

to yield a single scalar representation of the terrain's slope. Here, $\overline{\cdot}$ denotes the locally weighted averaging operator. Differing from the approach presented in the previous chapter, we consider isotropic local kernels only, i.e. $\mathbf{\Sigma}_i = \ell(\mathbf{x}_i)^2 \cdot \mathbf{I}$, which have the local length-scale $\ell(\mathbf{x}_i) \in \mathbb{R}$ as free parameter. As a second difference, we do not iteratively adapt to the local structure, but rather learn a mapping function for this task offline that yields for a given $T(\mathbf{x}_i)$ the best local length-scale $\ell(\mathbf{x}_i)$. Our mapping function takes the form

$$\ell(\mathbf{x}_i) = \begin{cases} a \cdot T(\mathbf{x}_i)^{-1} & \text{if } a \cdot T(\mathbf{x}_i)^{-1} < \ell_{max} \\ \ell_{max} & \text{else} \end{cases} \tag{10.3}$$

in order to yield short length-scales in high variance terrain and long length-scales in flat parts. $\ell(\mathbf{x}_i)$ is bounded by ℓ_{max} to prevent length-scales from going to infinity in large, flat regions. The scale parameter a is learned in parallel to the search for the GP's hyperparameters. To give an intuition, we visualize $\ell(\mathbf{x}_i)$ in Fig. 10.3 (left) for different parameter settings.

To be able to make elevation predictions at arbitrary locations, we need to evaluate the covariance function at arbitrary locations and, thus, need to have local kernels $\mathbf{\Sigma}$ available at any point in the input space—not just at the locations $\mathbf{x}_i$ of the training set. We are only able to calculate gradients and kernels directly where we have sufficient elevation observations in the local neighborhood. Whereas in the previous chapter, we used weighted averaging to calculate kernels in regions with few or no observations, instead, we propose to put another GP prior on the local kernel's parameters. We call this a *hyper GP*, since its predictions are used as hyperparameters for the GP that models the elevations.

The hyper GP represents the function $\mathbf{x} \mapsto \ell(\mathbf{x})$ for the length-scales of the main GP. As the length-scales have to be positive, we transform its training data into logarithmic

space. The hyper GP itself uses an isotropic length-scale l_h, which can be treated as an additional hyperparameter of the model. At inference time, we calculate the most likely length-scale given by the mean prediction of the hyper GP and use the resulting kernels for elevation predictions of the elevation GP. The improvement in elevation prediction on the benchmark dataset of Sec. 9.4 (see Fig. 9.2) using the hyper GP (HGP) with respect to the weighted averaging (WA) approach taken in the previous chapter is shown in the table in Fig. 10.3. We give the results for three different fractions of points removed from the training set.

Model Tiling

The cubic time complexity of standard GPs makes a direct application to large data sets impossible. In terrain modeling tasks, the training data grows as new parts of the terrain are explored. With only a few laser scans, the training points are in tens of thousands and regular GPs would spent days on fitting the regression function.

We propose to use an ensemble of overlapping GPs, where every sub-model is assigned to a specific region in the input space. For most of the covariance functions relevant in practice, the covariance between two points decreases drastically with their distance. The length-scale parameter of the covariance function is simply a factor scaling this decay, which does not influence the asymptotic behavior. Consequently, the decrease in covariance is asymptotically the same for our nonstationary covariance function, where length-scales vary smoothly and are ultimately bounded by a maximum value. If a point lies sufficiently far from a region in input space, it has virtually no influence on the regression result within this region. Thus, we propose to split the input space into rectangular segments and to assign an individual GP model to each of the segments. This sub-model is then provided with observations from within its segment only. This idea is similar to the approximation that considers the full dependencies between observations if they belong to the same segment, but applies approximations for longer distances [Snelson and Ghahramani, 2007]. To avoid problems at segment boundaries, we arrange the segments in an overlapping fashion and only use the center parts of the segments for the final prediction. In this way, we overcome the problem that predictions close to segment borders have unreasonably high variance [Snelson and Ghahramani, 2007].

Concretely, for a prediction at input location $\mathbf{x}$, we first determine the GP segment which we consider most likely to have the best approximation for $\mathbf{x}$, i.e. the segment which has the center with the shortest Euclidian distance to $\mathbf{x}$. Given this hard assignment to segments, the resulting function is no longer continuous on the whole domain in general. We have, however, not observed notable problems in practice, since already small overlaps

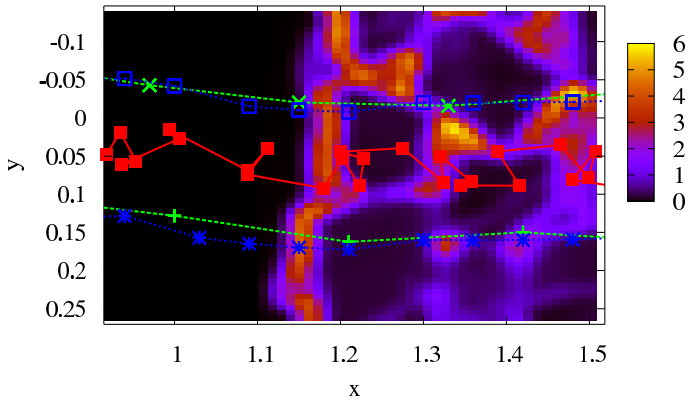


Figure 10.4: Parts of a plan generated by our algorithm including the underlying cost function, which depends on the terrain gradient and the uncertainty about elevation predictions estimated by the Gaussian process model. The red line (filled boxes) depicts the trajectory of the center of body, the other lines (stars and empty boxes) visualize feet motions. The cost function is color coded ranging from black (little costs) to yellow/light-gray (high costs). Axis dimensions are given in meters.

between segments result in similar regression results at the boundary. In order to evaluate the gain in runtime performance, let us assume that

a) every segment contains at most a fraction c of all training data n, and

b) segments overlap by a fraction v of their inputs.

Every segment then uniquely covers $cn{-}cnv = cn(1{-}v)$ of the training data, which makes it necessary to use $\frac{n}{cn(1-v)} = [c(1{-}v)]^{-1}$ segments to cover the whole input space. The original GP training time of $O(n^3)$ can then be expressed as $O((c(1{-}v))^{-1}{\cdot}(cn)^3) = O(c^2(1{-}v){\cdot}n^3)$. If we keep v constant, and scale c anti-proportionally to n—which corresponds to keeping the segment size constant—the training time becomes linear in n.

In the experiments documented in Sec. 10.3, we specifically evaluated the benefits of our model tiling strategy as well as the advantages that the locally adapted GP has w.r.t. the standard GP model and grid-based approaches. Before doing so, we describe how to generate useful trajectories for a quadruped robot using the predictive elevation model introduced in this section.

10.2 Plan Generation

We can use the learned terrain models to plan a path for the robot that is collision-free and statically stable. Our overall planning approach is an adaptation of the conventional probabilistic roadmap algorithm [Kavraki *et al.*, 1996]. Our simplified model of the quadruped robot is a body with four two-link legs and point-shaped feet. The planning algorithm is

a search for motions between static stances, always moving just a single leg at a time and maintaining static stability over uneven terrain. We first randomly sample a set of potential footholds across the terrain, which are used to generate a graph of potential *stances*, that is, statically stable and kinematically feasible positions of the robot. Graph search is then used to find a sequence of stances from the start to the goal; the sequence of stances can then be converted to series of planned joint angles.

10.2.1 Sampling Footholds

Let us assume that the planning problem is to find a motion plan that is essentially a futtock (midline) motion across the uneven terrain from the start position to the goal. This assumption will allow us to simplify the sampling to examining potential footholds around the straight line to the goal, selecting footholds $\boldsymbol{\phi} = (x, y, z)$ according to some regular discretization around the line of intended motion, e.g., see the right diagram of Fig. 10.2 for the simplest case of equidistantly sampled footholds around a straight line. We do this without loss of generality; we can easily support more complex scenarios by choosing different sampling strategies. The sampling problem is outside the scope of this chapter but has been discussed, for example, in [Hsu *et al.*, 2005].

Each sampled foothold is evaluated with respect to a cost function and rejected completely if the expected cost is above some threshold. The cost function used in this work consists of the terrain gradient (i.e., slope) and the uncertainty in the terrain model (i.e., the Gaussian process predictive uncertainty) at the sampled foothold $\boldsymbol{\phi}$.

10.2.2 Stance Graphs

Given a set of potential footholds, we next generate feasible stances of the robot. A *stance* is an assignment of each foot i to a foothold, $\boldsymbol{\phi}_i$, such that it is kinematically feasible for the robot to place its feet at each of the four footholds and remain statically stable. Note that determining whether a stance is feasible or not is not directly computable from a set of foot positions because the feet do not provide a unique description of the robot's pose. The robot has 18 degrees of freedom: 6 DOF for the center of the body $(x, y, z, roll, pitch, yaw)$ and three joints $(hip_\alpha, hip_\beta, knee)$ in each leg. Under the assumption that the positions of the feet are fixed, the feet constitute 12 constraints, leaving 6 unconstrained degrees of freedom, corresponding to the position of the center of body. A stance s_i is therefore an assignment of feet to footholds $\boldsymbol{\phi}_{1..4}$ and a selection of a center of body position $\boldsymbol{\xi}$.

Given an assignment of the center of the body position for a set of foot positions, the known kinematics can be used to recover the joint angles of the legs and determine if the pose is consistent with the dimensions of the leg links and the limits on the joint angles.

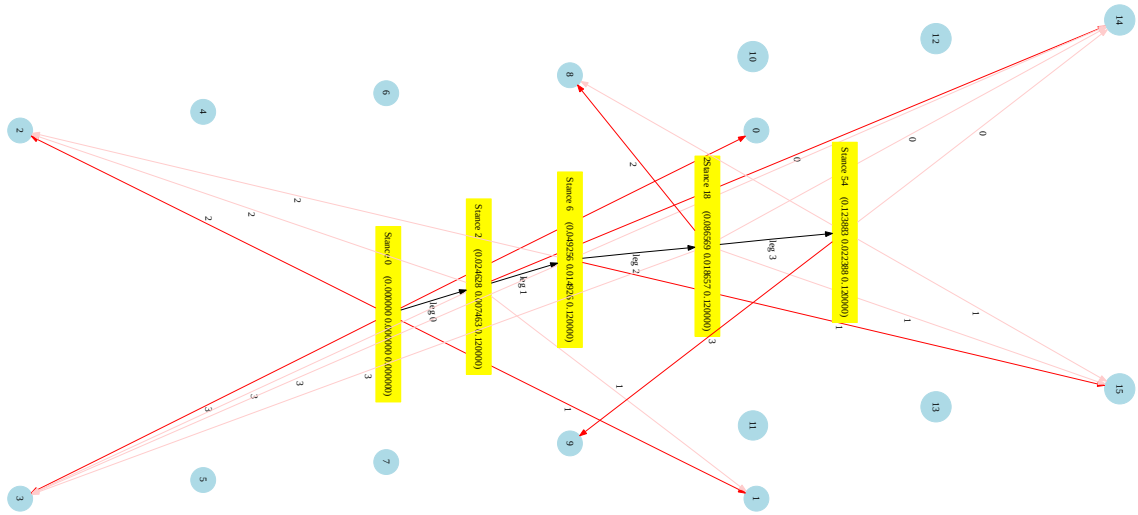


Figure 10.5: An example stance graph generated from the set of foothold locations. Each stance node (yellow) is connected to four foothold nodes (blue). Each two connected stance nodes have three stance legs in common.

Given knowledge of the joint angles and that the stance is kinematically feasible, the center of mass can then be determined; if the projection of the center of mass onto the ground plane lies outside the support polygon (the convex hull of the four feet on the ground plane), then the stance is not statically stable and the robot will fall.

In assigning the position of the center of body for a given set of foot positions, we would ideally choose a center of body that provides static stability. Unfortunately, no closed form solution exists for finding a feasible and stable center of body, and the problem is in general non-convex. We therefore use a heuristic search strategy around the centroid of the support polygon. If none of the sampled centers of body provide a kinematically feasible and stable solution to the robot position given the foot positions, then the foot positions are rejected as an infeasible stance.

The feasible stances constitute nodes in a stance graph, to which we then add edges between pairs of stances s_i and s_j when a feasible motion exists to transform the robot from the start stance s_i (foot positions and center of body) to the end stance s_j. This problem is also underdetermined, in that an arbitrarily complex motion may be required to move from one stance to another. We therefore simplify this problem by considering only a fixed set of motion patterns consisting of (I.) a stance phase, during which the dog shifts its center of body to remain stable while stepping, and (II.) a foot swing phase during which a foot is moved from one foothold to another. In Fig. 10.5, we give an example stance graph built from 16 potential footholds. Figure 10.4 shows a longer example graph in less detail with the cost function underlayed.

Algorithmus 3 The Planning Process.

Require: Terrain model, start stance s_0 and goal $\vec{x}_g$.
  Sample footholds $\boldsymbol{\phi}$ using the terrain model
  Initialize $Q \leftarrow s_0$
  while Q is not empty **do**
    $s \leftarrow$ pop Q
    for all ϕ **do**
      $s' \leftarrow s$
      Update of position of foot to move, $\phi(\psi(s')) \leftarrow \phi$ in s'
      Update foot to move, $\psi(s') \leftarrow \psi(s') + 1 \mod 4$
      Search for new center of body position $\xi(s')$
      if $||\xi(s') - \vec{x}_g|| < \epsilon$ **then**
        return Parents $[s']$.
      end if
      if $\xi(s')$ exists **then**
        Set parent, $\pi[s'] = s$
        Push $Q \leftarrow s'$
      end if
    end for
  end while
  return *nil*

Once the stance graph has been built, we use Dijkstra's algorithm [Dijkstra, 1959] to find the shortest feasible sequence of stances from the start stance to a goal stance that gives a center of body position with some ϵ of the desired goal position, in practice, combining the search process with the stance graph generation. Additionally, we add a gait-order constraint, so that the plan must consist of a well-formed gait in which foot i is followed by foot $i+1 \mod 4$. By augmenting each stance variable with an additional foot-ordering variable ψ, this gait-ordering constraint dramatically improves the planning speed. Finally, we also use a hash table to prune the search, such that if two different routes are found to the same stance node s, then the search along the longer path is terminated. The full planning algorithm is given in Alg. 3. The hash table is omitted for space reasons.

10.3 Experimental Results

The goal of the experimental evaluation is to demonstrate the usefulness of the predictive terrain model introduced in Sec. 10.1 for path planning in real environments and to show that our adapted Gaussian process model is more accurate than conceptually simpler approaches such as standard GPs or elevation grid maps. We additionally analyzed the benefits of our tiling approach introduced in Sec. 10.1 with respect to runtime and accuracy performance.

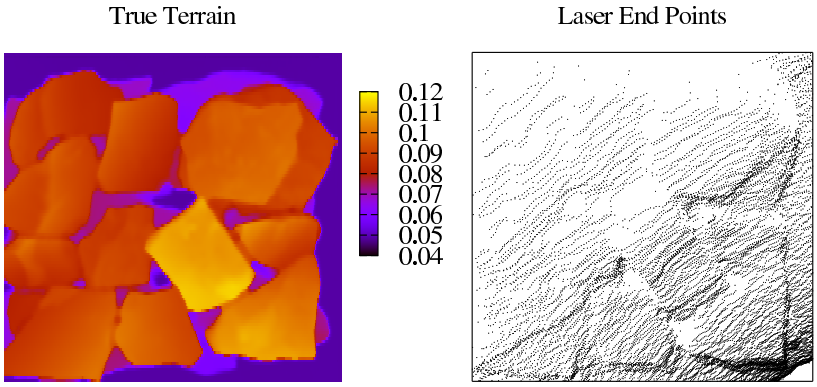


Figure 10.6: A rocky, real terrain of dimensions 0.6m × 0.6m with a maximum elevation of approximately 7cm (color-coded as yellow) above ground level. We give the true elevation values (left) and the set of laser end points (right) recorded by the robot standing on the bottom right corner w.r.t. the diagram and performing a tilt motion.

10.3.1 Adapting to Local Terrain Structure

In a set of simulated experiments, we first compared the performance of standard GP regression using the squared exponential (SE) covariance function against our nonstationary covariance function with local length-scale adaption. Our evaluation terrain was an excerpt from a ground truth model of the rocky terrain depicted in the left diagram of Fig. 10.6. Using this model, we simulated 2 500 laser observations from a single viewpoint. We uniformly selected 4 350 points from the true terrain for evaluation. We then conducted Monte-Carlo search in the parameter space of the covariance functions and on the parameters of the adaption procedure. In a preliminary run over 34 000 configurations, we determined general ranges for the parameters and in a secondary search, we evaluated 10 000 configurations in the predetermined ranges. In Fig. 10.7, we give a scatter plot of the best results plotting the achieved mean squared prediction errors (MSE) against the negative log predictive likelihood (NLPD) per parameter sample. It can be seen that the nonstationary covariance function is able to achieve both better MSE w.r.t. the ground truth and also a better NLPD—which also takes the predicted variances into account.

Figure 10.8 shows a cut through the achieved regression surfaces of the stationary (left panel) and nonstationary (right panel) models. It can clearly be seen that the stationary GP model (left diagram) needs to select an extremely small length-scale parameter in order to represent the sharp edges. It is thus not able to smooth flatter and more sparsely sampled regions. In contrast, the nonstationary covariance function (right diagram) decreases the length-scale only at the sharp "cliffs" and where no gradients are observed. It is thus able to account for the highly varying parts at the first ascent and for the higher uncertainties in the occluded parts, while still smoothing in the flat regions.

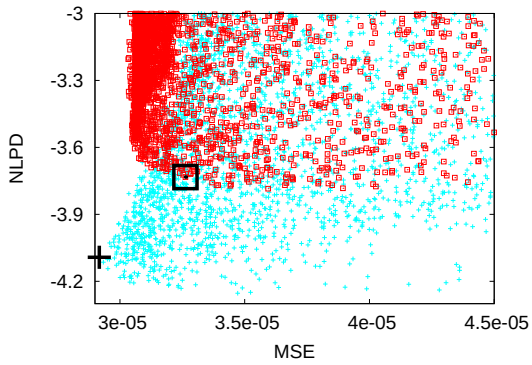


Figure 10.7: The Monte-Carlo parameter searches for the stationary GP (red boxes) and for the nonstationary GP (bright crosses). The selected optima, which minimize the MSE as well as the NLPD error, are marked by a large black box and by a large black cross, respectively.

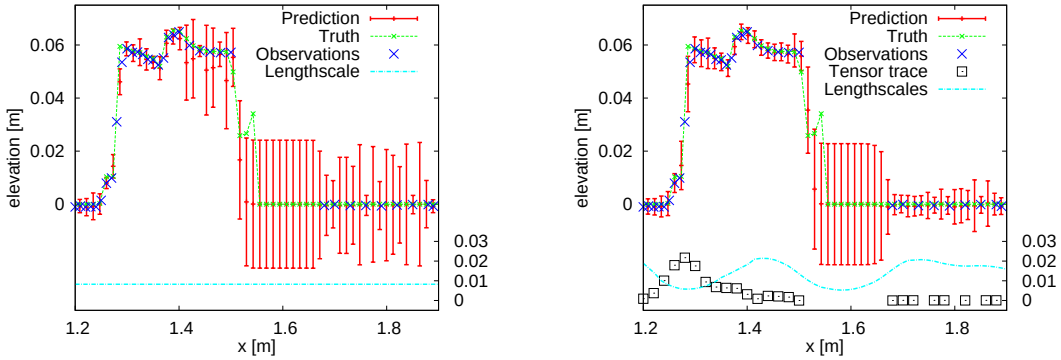


Figure 10.8: Regression models learned using the standard GP model (left) and our locally adaptive approach (right). Here, we visualize a slice plane of the terrain surface (y is fixed, x varies on the horizontal axis in meters, terrain elevation on the vertical axis). We also give the adapted length-scales below the curves using a second vertical axis).

10.3.2 Splitting the Terrain into Overlapping Sub-Models

We evaluated the benefits of segmenting the input space in overlapping tiles. To do so, we applied different tile sizes to the terrain model analyzed in the previous experiment. We measured the prediction accuracy in the innermost 0.0025 m^2 of a tile while linearly increasing the tile area, and thereby also the amount of training data of the associated GP. The upper left diagram of Fig. 10.9 shows that with an increasing tile size, both MSE and NLPD quickly decrease and almost converge as the area reaches 0.003 m^2. The runtime, however, continues to grow cubically with the segment size beyond this point. The remaining three diagrams in Fig. 10.9 give a visual impression of the effects that different tile sizes have on the regression function in its center. Increasing the tile size from 0.03 m^2 (lower left) to 0.056 m^2 (lower right) does not lead to a notable improvement.

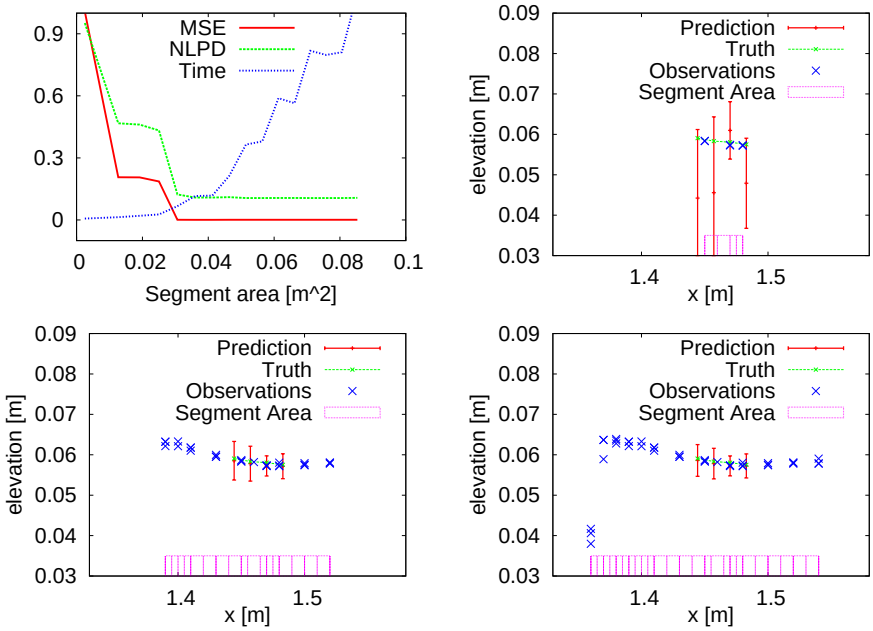


Figure 10.9: Mean squared error (MSE), negative log predictive likelihood (NLPD) of the inner $0.0025m^2$ of the segment and training time (relative plot, scale omitted), for different segment areas (left). Values are linearly scaled to fit in plot. Plots of prediction for segment sizes $0.0025m^2$, $0.03m^2$, and $0.056m^2$.

The runtime requirements on a standard PC for learning from approximately 100 000 points using our `C++` implemenation are in the order of 1.5 sec. with overlapping stationary GPs and 3 sec. with overlapping nonstationary GPs.

10.3.3 Mapping Accuracy on Real Terrain

We evaluated our terrain model with a real quadruped robot in a situation similar to the one depicted in Fig. 10.1. The robot used was a quadruped *LittleDog* developed by Boston Dynamics. We have equipped the dog with a Hokuyo URG laser scanner. A high-resolution motion capture system, the Vicon MX, yields estimates of the robot pose using measurements from reflective markers attached to the robot's body. The laser sensor is mounted on top of the robot in a 25° angle facing towards the ground, such that (a) terrain elevation measurements can be acquired while executing plans and (b) 3D range scans can be recorded by executing a tilt motion using the front and rear legs. The evaluation in this section concentrates on the question of how accurately the elevation structure of the terrain board (approximately 0.6 m $\times$ 0.6 m) can be recovered from such a single 3D scan.

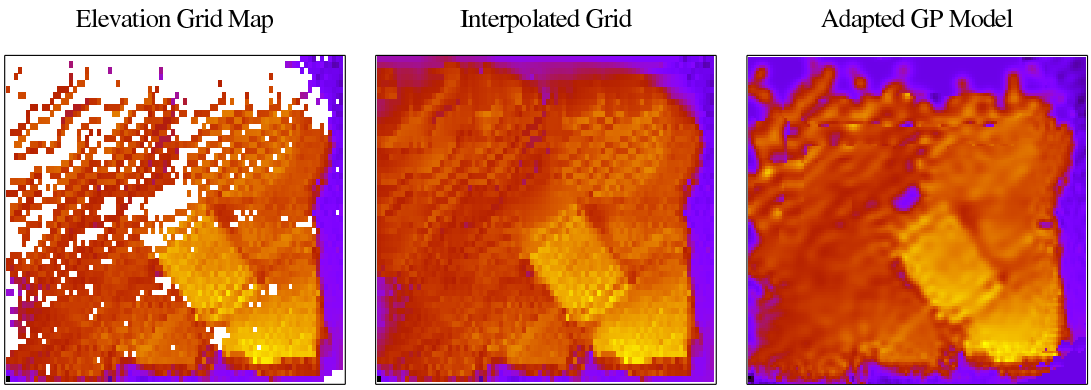


Figure 10.10: Mapping results using the elevation grid model (left), its bilinear completion (center), and the mean predictions of our adapted GP model (right).

Calibration

Note that our system is not performing simultaneous localization and mapping (SLAM) and, thus, we assume accurate knowledge of the pose of the laser. The specifics of the Vicon motion capture system used to track the robot's position required us to automatically infer the time offsets between the laser data and pose measurements, and to calculate the 6 DOF transformation between a known position of the dog and the actual laser sensor position at the same time. In order to compute both these calibration quantities, we recorded a 3D range scan of three orthogonal boards placed in front of the robot (side lengths approximately 1 m) and optimized for the 7 parameters of the transformation in a sampling-based fashion similar to simulated annealing. Here, random parameter samples are evaluated using a predefined objective function. Additionally, a *temperature* variable defines how close the newly sampled values shall be from the current optimum. As the objective function, we chose the sum of squared distances of laser end points to the board surfaces. By decreasing the temperature level gradually over time, accurate calibration parameters are typically obtained within $300 - 800$ iterations.

Mapping Rough Terrain

We evaluated our adapted GP approach using scans of a rocky terrain surface acquired by the quadruped robot against a known ground-truth model of the terrain acquired using a high-accuracy metrology system. The left diagram in Fig. 10.6 depicts the true elevation structure of this terrain (see the caption for details). The right diagram shows the raw set of laser endpoints that were acquired by the robot when it executed a tilt motion. It can be clearly seen from the uneven distribution of points that parts of the terrain are not sampled densely due to occlusions and a larger distance to the sensor location (which was

towards the bottom right w.r.t. the diagram). An elevation grid map can be built from this training set of surface points by discretizing the x-y space and by fitting 1D Gaussians to the elevation samples falling into the respective grid cells. The result of this operation is depicted in the left diagram in Fig. 10.10.

A standard way of filling gaps in grid maps without altering the known cells is called *bilinear interpolation*, that is, the extension of linear interpolation to bivariate functions. The result of such an operation applied to an incomplete elevation grid map is depicted in the middle diagram in Fig. 10.10 and the results obtained with our adapted GP approach are depicted in the rightmost diagram. Here, we plot the mean predictions for terrain elevations. The right diagram in Fig. 10.8 shows a vertical cut through this surface. This visualization shows that our our model yields tight uncertainty bounds around the observed points and that its predictions are consistent with the true elevation structure at un-observed locations.

We compared the prediction errors of our adapted GP model to the baseline models *Elevation Grid* and *Interpolated Elevation Grid*. In Fig. 10.11, we give the squared error of elevation predictions averaged over 10 000 samples drawn randomly from the terrain. The error-bars give the standard deviations of the individual sample sets. To assess the influence of grid resolution for the two grid-based models, we tested six different numbers of cells per grid dimension (x-axis). Since the standard elevation grid does not make predictions in occluded or less densely sampled areas, its performance was evaluated on its occupied cells only—which makes the comparison a rather conservative one. It can be seen from the diagram that our adapted GP model is as good as the elevation grid at optimal resolution, even though its performance measure additionally includes predictions of un-observed elevations.

10.3.4 Path Planning using Real Terrain Data

In order to evaluate the practical usefulness of our approach, we tested our terrain models in combination with the trajectory planner described in Sec. 10.2. The experimental setup was to sample 1 000 random starting locations in front of the rought part of the terrain and to pick corresponding goal locations behind it. For each of these location pairs and each of the three alternative terrain models, the planner generated a set of footholds and searched for the best path towards the goal location. We then evaluated (a) the maximal path length that could be constructed given the kinematic constraints of the robot, and (b) the errors of the elevation predictions at the selected foothold locations. An example plan and the corresponding cost function that was computed from the underlying terrain model are depicted in Fig. 10.4

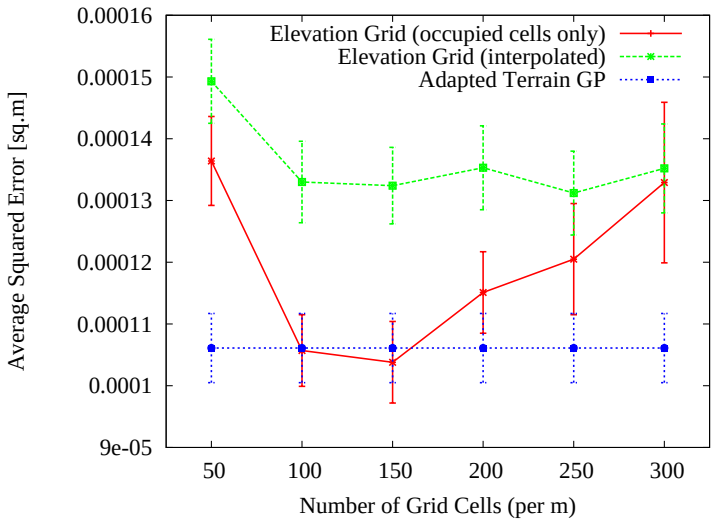


Figure 10.11: Errors of elevation predictions by the different models depending on the number of grid cells used. The elevation grid (without interpolation) was evaluated on occupied cells only.

Figure 10.12 summarizes our results. It can be seen from the left diagram that it was always possible to plan the maximal path of 2 m using the interpolated grid and the adapted GP model. Using the standard grid, however, the plans never exceeded a length of 1.6 m, which is not surprising given the large number of unknown cells which prohibit foot placements. As can be seen from the the mean squared error values in the right diagram, the adapted GP model predicts the true terrain elevations better at the chosen foothold locations than the interpolated grid model, which means that there is a lower risk of failure when executing these plans.

10.4 Related Work

Terrain modeling and map building are central tasks within robotics. We follow the approach presented in the previous chapter and model the terrain using a GP with a nonstationary covariance function originally proposed by Paciorek and Schervish [2004]. If not specifically addressed, the nonparametric nature of this approach causes computational problems for large terrains, due to an unfavorable N^3 scaling for training, where N is the number of observations. To overcome this problem, we propose to use an ensemble of GPs, where every GP is assigned to a specific region, an idea akin to GP mixture models such as [Williams, 2006]. Rasmussen and Ghahramani [2002] extend the ideas of Tresp [2000] and present an "infinite mixture of experts model" where the individual experts are different Gaussian process models. Cornford *et al.* [1999] model straight discontinuities in wind fields by placing auxiliary GPs along the edge on both sides of the discontinuity. These

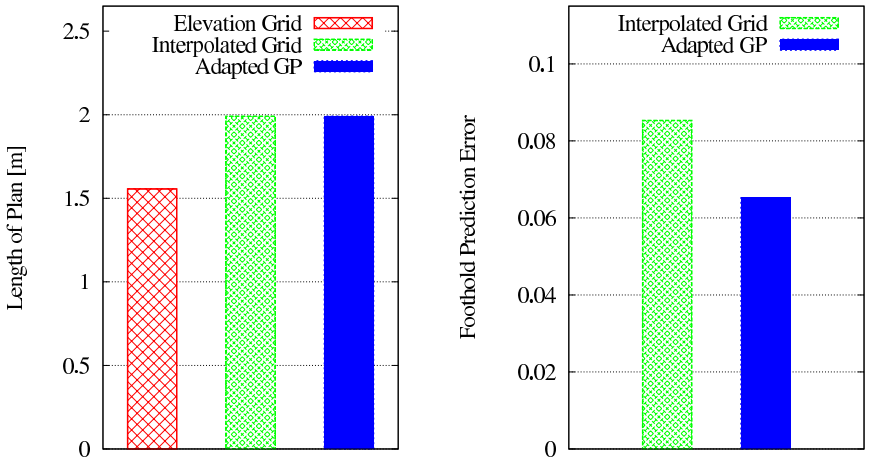


Figure 10.12: Evaluation of 1 000 plans generated using the different terrain models. The left bar plot shows the maximal length of generated plans and the right plot gives the mean squared errors (scaled by 10^{-3}) of elevation predictions at the planned footholds.

are then used to learn GPs representing the process on either side of the discontinuity. A broad overview of methods used for modeling terrain data was given by Hugentobler [2004]. Elevation maps in particular have been used as an efficient data structure for representing dense terrain data [Bares *et al.*, 1989, Pfaff and Burgard, 2005] and were later extended to multi-level probabilistic surface maps [Triebel *et al.*, 2006]. Früh and Zakhor [2002] present an approach to filling local gaps in 3D models based on local linear interpolation. Compared to these approaches, Gaussian process (GP) models [Rasmussen and Williams, 2006b] have the advantage of not assuming a fixed discretization of the space and of additionally providing predictive uncertainties.

10.5 Conclusion

In this chapter, we considered the terrain mapping problem for a legged robot equipped with a laser range finder. We demonstrated the use of a nonparametric Bayesian regression approach, Gaussian processes, for reliably modeling rough terrain. Our extended model balances smoothing against the preservation of structural features and is capable of accurately predicting elevations in the presence of noise even at unobserved locations. These features allow us to plan foot trajectories of a quadruped robot to reach a goal location. We showed in experiments with data acquired using a real robot that this nonparametric terrain modeling approach was able to infer the terrain more reliably, leading to better planning than grid-based terrain models.

Chapter 11

Locally Adaptive Gaussian Process Regression

We derive a nonstationary Gaussian process model from first principles using point estimates of local smoothness. Our approach allows to employ efficient gradient-based optimization techniques for learning the parameters of a latent smoothness process and of the observed process jointly.

Many regression problems that arise in practice, including terrain modeling as discussed in the previous two chapters, exhibit a nonstationary structure, that is, the function underlying the data changes its smoothness characteristics with the index location. Modeling this input-dependent smoothness has the potential of better explaining the observed data and also of better generalizing to unseen locations. In the past, several approaches for specifying nonstationary GP models haven been proposed in the literature [Sampson and Guttorp, 1992, Schmidt and O'Hagan, 2003]. A particularly promising approach, which was also adopted in Chapters 9 and 10, was presented by Paciorek and Schervish [2004], who proposed to explicitly model the input-depending smoothness using additional, latent GPs. This approach

(a) provides the user with a continuous latent space of local kernels,

(b) allows the user to analyze the estimated length-scale function yielding insights into the problem domain, and

(c) it fully stays in the GP framework so that methods for speeding up GP inference and fitting can be adapted.

Paciorek and Schervish provide a flexible and general framework based on approximation by MCMC integration, which unfortunately, as also noted by the authors, is computationally demanding for large data sets and which is thus not feasible in the real world situations that are typically encountered in robotics and engineering tasks. In this chapter, we present a different approximation that does not integrate over all latent variables but uses the predicted mean values only. We call these "point estimates of local smoothness". Specifically, we parameterize the nonstationary covariances using a second GP with m latent length-scales. Assuming $m \ll n$, where n is the number of training points, this results in a nonstationary GP regression method with only little overhead compared to standard GPs. More importantly, using point estimates naturally leads to gradient-based techniques for efficiently learning the parameters of both processes jointly.

As a second contribution of this chapter, we address runtime efficiency for large datasets using an approach inspired by the *mixtures of local experts* (MoE) approach [Tresp, 2000, Rasmussen and Ghahramani, 2002]. We decompose the full nonstationary GP model into multiple *local* ones, each responsible only for a subset of the data. A *gating* function is used to assign responsibilities to certain GPs, defining a mapping from input space to model associations. In contrast to other approaches that model nonstationarity using mixtures of *stationary* experts [Tresp, 2000], we do not require the expert-boundaries to coincide with the locations of changing smoothness by considering mixtures of *nonstationary* experts that share their latent length-scales in the overlapping regions.

We present experiments carried out on synthetic and real-world data sets from challenging application domains such as robotics and embedded systems showing the relevance and feasibility of our approach. More specifically, our nonstationary GP approach significantly outperforms standard GPs in terms of prediction accuracy, while also being significantly faster than the MCMC-based approach by Paciorek and Schervish [2004]. In additional experiments we evaluate the benefits of sparsification (or explicitly: "model tiling") that were outlined above and compare our approach to *locally weighted projection regression* (LWPR), a state-of-the-art approach to non-linear regression developed by Vijayakumar *et al.* [2005a].

We regard these empirical results as a contribution on its own as they tighten the link between advanced regression techniques based on GPs and application domains such as robotics and embedded systems. To the best of our knowledge, it is the first time that nonstationary GPs have been learned in a principled way in these challenging domains.

The chapter is organized as follows. We introduce our approach to nonstationary GP regression using point estimates of local smoothness in Sec. 11.1 and show how to learn the hyperparameters using gradient-based methods. In Sec. 11.2, we then discuss the mixture

of nonstationary experts approach. Before discussing related work, we demonstrate the feasibility and relevance of our approach in an extensive set of experiments in Sec. 11.3.

11.1 Modeling Variable Smoothness

Recalling Sec. 2.2 and the previous two chapters, we aim at learning a model for $p(y_*|\mathbf{x}_*,\mathcal{D})$ given a set of observed training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. We place a Gaussian process prior on the dependency of y on $\mathbf{x}$ yielding the predictive distribution $y_* \sim \mathcal{N}(\mu_*, \sigma_*^2)$ with parameters

$$\mu_* = \mathbf{k}_{\mathbf{x}_*,\mathbf{x}}^T(\mathbf{K}_{\mathbf{xx}} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} \tag{11.1}$$

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_{\mathbf{x}_*,\mathbf{x}}^T(\mathbf{K}_{\mathbf{xx}} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_{\mathbf{x}_*,\mathbf{x}} + \sigma_n^2 \tag{11.2}$$

and we assume the nonstationary covariance function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \, |\mathbf{\Sigma}_i|^{\frac{1}{4}} \, |\mathbf{\Sigma}_j|^{\frac{1}{4}} \left| \frac{\mathbf{\Sigma}_i + \mathbf{\Sigma}_j}{2} \right|^{-\frac{1}{2}} \cdot \exp\left[-\mathbf{d}_{ij}^T \left(\frac{\mathbf{\Sigma}_i + \mathbf{\Sigma}_j}{2} \right)^{-1} \mathbf{d}_{ij} \right] \tag{11.3}$$

of Paciorek and Schervish [2004] for defining the covariance structure of the process—i.e., the quantities $k(\cdot,\cdot)$, $\mathbf{k}$, and $\mathbf{K}$ in Eq. (11.1) and (11.2). See also the summary of notation in Sec. 1.4.

The intuition behind the nonstationary covariance function (11.3) is that each input location $\mathbf{x}_i$ is assigned a local Gaussian kernel matrix $\mathbf{\Sigma}_i$ and the covariance between two targets y_i and y_j is calculated by averaging between the two local kernels at the input locations $\mathbf{x}_i$ and $\mathbf{x}_j$. In this way, the local characteristics at both locations influence the modeled covariance of the corresponding target values. For the sake of simplicity, we consider the isotropic case only in this chapter. The general case can be treated in the same way.

In the isotropic case, in which the eigenvectors of the local kernels are aligned to the coordinate axes and their eigenvalues are equal, the matrices $\mathbf{\Sigma}_i$ simplify to $\ell_i^2 \cdot \mathbf{I}_n$ with a real-valued length-scale parameter ℓ_i. In the one-dimensional case, Eq. (11.3) then simplifies to

$$k(x_i, x_j) = \sigma_f^2 \cdot (\ell_i^2)^{\frac{1}{4}} \cdot (\ell_j^2)^{\frac{1}{4}} \cdot \left(\frac{1}{2}\ell_i^2 + \frac{1}{2}\ell_j^2 \right)^{-\frac{1}{2}} \cdot \exp\left[-\frac{(x_i - x_j)^2}{\frac{1}{2}\ell_i^2 + \frac{1}{2}\ell_j^2} \right] . \tag{11.4}$$

We do not specify a functional form for the length-scale $\ell(x)$ at location x but place a GP prior over them. More precisely, an independent GP is used to model the log-

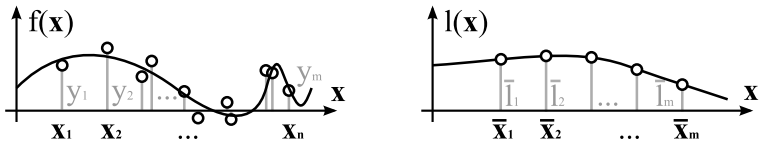


Figure 11.1: Placing a GP prior over the latent length-scales for nonstationary GP regression. An observed Gaussian process $\mathcal{GP}_y$ is sketched on the left and the latent $\mathcal{GP}_\ell$ governing the local length-scales is shown on the right. In the situation sketched here, the function is smoother for smaller index values $\mathbf{x}$ and less smooth for larger ones.

arithms $\log(\ell(x))$ of the length-scales to avoid negative values. This process, denoted as $\mathcal{GP}_\ell$, is governed by a different covariance function specified by the hyperparameters $\boldsymbol{\theta}_\ell = \langle \overline{\sigma}_f, \overline{\sigma}_\ell, \overline{\sigma}_n \rangle$. Additionally, we have to maintain the set of m support values $\overline{\boldsymbol{\ell}}$ as part of the model as depicted in Fig. 11.1. Summarizing the parameters introduced so far, we have $\boldsymbol{\theta} = \langle \boldsymbol{\theta}_y, \boldsymbol{\theta}_\ell, \overline{\boldsymbol{\ell}} \rangle = \langle \sigma_f, \sigma_n, \overline{\sigma}_f, \overline{\sigma}_\ell, \overline{\sigma}_n, \overline{\boldsymbol{\ell}} \rangle$.

For *making predictions* we have to integrate over the latent length-scales to get the predictive distribution

$$p(y_*|\mathbf{x}_*, \mathcal{D}, \boldsymbol{\theta}) = \int\int p(y_*|\mathbf{x}_*, \mathcal{D}, \exp(\ell_*), \exp(\boldsymbol{\ell}), \boldsymbol{\theta}_y) \cdot p(\ell_*, \boldsymbol{\ell}|\mathbf{x}_*, \mathcal{D}, \overline{\boldsymbol{\ell}}, \overline{\mathcal{X}}, \boldsymbol{\theta}_\ell)\, d\boldsymbol{\ell}\, d\ell_* \quad (11.5)$$

of a regressand y_* at location $\mathbf{x}_*$ given a dataset $\mathcal{D}$ and hyperparameters $\boldsymbol{\theta}$. Note that $\mathcal{GP}_\ell$ is defined over the log length-scales to ensure that they are non-negative. Because the marginalization in Eq. (11.5) is intractable analytically, Paciorek and Schervish [2004] apply MCMC to approximate it. We propose instead, to consider $p(y_*|\mathbf{x}_*, \mathcal{D}, \boldsymbol{\theta}) \approx p(y_*|\mathbf{x}_*, \mathcal{D}, \exp(\hat{\ell}_*), \exp(\hat{\boldsymbol{\ell}}), \boldsymbol{\theta}_y)$ where $(\hat{\ell}_*, \hat{\boldsymbol{\ell}})$ are the most-likely values of the length-scale process at $\mathbf{x}_*$ and at the index locations of $\mathcal{D}$, that is, the mean predictions of $\mathcal{GP}_\ell$. Since the length-scales are independent latent variables in the combined regression model, making predictions amounts to making two standard GP predictions using Eq. (11.1) and (11.2), one using $\mathcal{GP}_\ell$ to get $(\hat{\ell}_*, \hat{\boldsymbol{\ell}})$ and one using $\mathcal{GP}_y$ with $(\hat{\ell}_*, \hat{\boldsymbol{\ell}})$ treated as fixed parameters. Experimental evaluation showed that our approximation leads to the same model accuracay on benchmark data sets as the MCMC-based one. As a notable difference, however, our approximation yields a closed-form solution for prediction and density estimation as well as for their derivatives.

11.1.1 Learning the Hyperparameters

So far, we have described our model assuming that we have the joint hyperparameters $\boldsymbol{\theta}$ of the overall process. In practice, we are unlikely to have these parameters a priori and instead we wish to estimate them from observed data.

Assume a given set of n observations $\mathbf{y}$ at locations $\{\mathbf{x}_i\}_{i=1}^n$, represented by a (row-wise

stacked) matrix $\mathbf{X}$, $[\mathbf{X}]_{ij} = [\mathbf{x}_i]_j$. We seek to find the hyperparameters that maximize the likelihood of observing $\mathbf{y}$ at $\mathbf{X}$, i.e., we seek to maximize $p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{X}, \boldsymbol{\ell}, \boldsymbol{\theta}_y) \cdot p(\boldsymbol{\ell}|\mathbf{X}, \overline{\boldsymbol{\ell}}, \overline{\mathbf{X}}, \boldsymbol{\theta}_\ell)\, d\boldsymbol{\ell}$. As for making predictions, such a marginalization is intractable and we instead maximize

$$L(\boldsymbol{\theta}, \overline{\boldsymbol{\ell}}) = \underbrace{\log p(\mathbf{y} \mid \mathbf{X}, \exp(\hat{\boldsymbol{\ell}}), \boldsymbol{\theta}_y)}_{\mathcal{GP}_y} + \underbrace{\log p(\hat{\boldsymbol{\ell}}|\mathbf{X}, \overline{\boldsymbol{\ell}}, \overline{\mathbf{X}}, \boldsymbol{\theta}_\ell)}_{\mathcal{GP}_\ell} + \underbrace{\text{const.}}_{\boldsymbol{\theta} \text{ prior}}, \tag{11.6}$$

where $\hat{\boldsymbol{\ell}}$ are the mean predictions of $\mathcal{GP}_\ell$. The optimum of this objective function w.r.t. to the hyperparameters $\boldsymbol{\theta}$ and $\overline{\boldsymbol{\ell}}$ can be found using gradient-based optimization. In our experiments, we optimized $\overline{\sigma}_f$, $\overline{\sigma}_n$, and $\overline{\sigma}_\ell$ of the latent length-scale process in an outer cross-validation loop on an independent test set and assumed $\partial L(\boldsymbol{\theta})/\partial \bullet = 0$, $\bullet \in \{\overline{\sigma}_f, \overline{\sigma}_n, \overline{\sigma}_\ell\}$, within the (inner) gradient-based optimization runs. The locations $\overline{\mathbf{X}}$ of the latent kernel width variables were sampled uniformly on the bounding rectangle given by $\mathbf{X}$.

In the following, we will detail the objective function and its gradient with respect to the hyperparameter.

11.1.2 The Objective Function

We maximize the marginal likelihood Eq. (11.6) of the data with respect to the joint hyperparameters as well as the support values $\overline{\boldsymbol{\ell}}$ of the length-scale process. The first term in this equation is the standard objective function for Gaussian processes

$$\log p(\mathbf{y}|\mathbf{X}, \exp(\hat{\boldsymbol{\ell}}), \boldsymbol{\theta}_y) = -\frac{1}{2}\mathbf{y}^T(\mathbf{K}_{\mathbf{x},\mathbf{x}} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_{\mathbf{x},\mathbf{x}} + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log(2\pi)\ ,$$

where $|\mathbf{M}|$ denotes the determinant of a matrix $\mathbf{M}$ and $\mathbf{K}_{\mathbf{x},\mathbf{x}}$ stands for the noise-free nonstationary covariance matrix for the training locations $\mathbf{X}$ detailed below. Our point estimate approach considers the most likely latent length-scales $\hat{\boldsymbol{\ell}}$, i.e., the mean predictions of $\mathcal{GP}_\ell$ at locations $\mathbf{X}$. Thus, the second term of Eq. (11.6) has the form

$$\log p(\hat{\boldsymbol{\ell}}|\mathbf{X}, \overline{\boldsymbol{\ell}}, \overline{\mathbf{X}}, \boldsymbol{\theta}_\ell) = -\frac{1}{2}\log|\mathbf{K}_{\overline{\mathbf{x}},\overline{\mathbf{x}}} + \overline{\sigma}_n^2\mathbf{I}| - \frac{n}{2}\log(2\pi)\ .$$

Putting both together, we get the objective function

$$L(\boldsymbol{\theta}, \overline{\boldsymbol{\ell}}) = c_1 + c_2 \cdot \left[\mathbf{y}^T\mathbf{A}^{-1}\mathbf{y} + \log|\mathbf{A}| + \log|\mathbf{B}|\right]\ , \tag{11.7}$$

where c_1 and c_2 are real-valued constants, and $\mathbf{A} := \mathbf{K}_{\mathbf{x},\mathbf{x}} + \sigma_n^2\mathbf{I}$ and $\mathbf{B} := \mathbf{K}_{\overline{\mathbf{x}},\overline{\mathbf{x}}} + \overline{\sigma}_n^2\mathbf{I}$ are covariance matrices. The noise-free part of the nonstationary covariance matrix $\mathbf{K}_{\mathbf{x},\mathbf{x}}$ is calculated according to Eq. (11.3). As mentioned above, we consider the isotropic case

only for the sake of simplicity. We express Eq. (11.4) for the case of multivariate inputs $\mathbf{x}_i$ using the compact matrix-vector notation suggested in [Brooks, 2005]. Recalling that $\hat{\boldsymbol{\ell}}$ represents the local length-scale means at the training locations $\mathbf{X}$, we get

$$\mathbf{K}_{\mathbf{x},\mathbf{x}} = \sigma_f^2 \cdot \mathbf{P_r}^{\frac{1}{4}} \circ \mathbf{P_c}^{\frac{1}{4}} \circ (1/2)^{-\frac{1}{2}} \, \mathbf{P_s}^{-\frac{1}{2}} \circ \mathbf{E} \tag{11.8}$$

with

$$\begin{aligned} &\mathbf{P_r} = \mathbf{p} \cdot \mathbf{1}_n^T \,, \quad && \mathbf{P_c} = \mathbf{1}_n^T \cdot \mathbf{p}^T \,, && \mathbf{p} = \hat{\boldsymbol{\ell}}^T \hat{\boldsymbol{\ell}} \,, \\ &\mathbf{P_s} = \mathbf{P_r} + \mathbf{P_c} \,, && \mathbf{E} = \exp[-s(\mathbf{X}) \div \mathbf{P_s}] \,, && \hat{\boldsymbol{\ell}} = \exp\left[\mathbf{K}_{\mathbf{x},\overline{\mathbf{x}}}^T \left[\mathbf{K}_{\overline{\mathbf{x}},\overline{\mathbf{x}}} + \overline{\sigma}_n^2 \mathbf{I}\right]^{-1} \overline{\boldsymbol{\ell}}\right] . \end{aligned}$$

Note that $\mathbf{p} \in \mathbb{R}^n$ and, thus, $\mathbf{P_r}$ and $\mathbf{P_c}$ are matrices built using the outer vector product. Here, $s(\mathbf{X})$ calculates the $n \times n$ matrix of squared distances between the input vectors $\mathbf{x}$ contained in $\mathbf{X}$. The symbols $\circ$ and $\div$ denote element-wise multiplication and division respectively and matrix exponentiation $\mathbf{M}^\alpha$ is also defined element-wise for $\alpha \neq -1$. In the same notation, the covariance function for the latent length-scale process $\mathcal{GP}_\ell$ becomes (in the stationary squared exponential form)

$$\mathbf{K}_{\overline{\mathbf{x}},\overline{\mathbf{x}}} = \overline{\sigma}_f{}^2 \cdot \exp\left[-\frac{1}{2} s(\overline{\sigma_\ell}^{-2} \overline{\mathbf{X}})\right]$$

and, analogously, for making predictions within $\mathcal{GP}_\ell$

$$\mathbf{K}_{\mathbf{x},\overline{\mathbf{x}}} = \overline{\sigma}_f{}^2 \cdot \exp\left[-\frac{1}{2} s(\overline{\sigma_\ell}^{-2} \mathbf{X}, \overline{\sigma_\ell}^{-2} \overline{\mathbf{X}})\right] \,.$$

11.1.3 The Gradient

Using standard results from matrix calculus, the partial derivative of the objective Eq. (11.7) w.r.t. an element $\bullet$ of $\boldsymbol{\theta}$ turns out to be

$$\frac{\partial L(\boldsymbol{\theta}, \overline{\boldsymbol{\ell}})}{\partial \bullet} = -\mathbf{y}^T \mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \bullet} \mathbf{A}^{-1} \mathbf{y} + \mathrm{tr}(\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \bullet}) + \mathrm{tr}(\mathbf{B}^{-1} \frac{\partial \mathbf{B}}{\partial \bullet}) \,, \tag{11.9}$$

where $\mathrm{tr}(\mathbf{M})$ is the trace of a matrix $\mathbf{M}$. For the two hyperparameters of $\mathcal{GP}_y$ we get the straight-forward results

$$\frac{\partial \mathbf{A}}{\partial \sigma_n} = 2\sigma_n \mathbf{I} \,, \qquad \frac{\partial \mathbf{B}}{\partial \sigma_n} = 0 \,, \qquad \frac{\partial \mathbf{A}}{\partial \sigma_f} = 2\sigma_f \mathbf{K}_{\mathbf{x},\mathbf{x}} \,, \qquad \frac{\partial \mathbf{B}}{\partial \sigma_f} = 0 \,.$$

The case $\bullet = \overline{\boldsymbol{\ell}}$ yields $(\partial \mathbf{B}/\partial \overline{\boldsymbol{\ell}}) = 0$ and $(\partial \mathbf{A})/(\partial \overline{\boldsymbol{\ell}}) = (\partial \mathbf{K}_{\mathbf{x},\mathbf{x}})/(\partial \overline{\boldsymbol{\ell}}) =$

$$\sigma_f^2 \ (1/2)^{-\frac{1}{2}} \cdot \left[\left(\frac{\partial(\mathbf{P_r}^{\frac{1}{4}})}{\partial \overline{\boldsymbol{\ell}}} \circ \mathbf{P_c}^{\frac{1}{4}} \circ \mathbf{P_s}^{-\frac{1}{2}} \circ \mathbf{E} \right) + \left(\mathbf{P_r}^{\frac{1}{4}} \circ \frac{\partial(\mathbf{P_c}^{\frac{1}{4}})}{\partial \overline{\boldsymbol{\ell}}} \circ \mathbf{P_s}^{-\frac{1}{2}} \circ \mathbf{E} \right) + \left(\mathbf{P_r}^{\frac{1}{4}} \circ \mathbf{P_c}^{\frac{1}{4}} \circ \frac{\partial(\mathbf{P_s}^{-\frac{1}{2}})}{\partial \overline{\boldsymbol{\ell}}} \circ \mathbf{E} \right) + \left(\mathbf{P_r}^{\frac{1}{4}} \circ \mathbf{P_c}^{\frac{1}{4}} \circ \mathbf{P_s}^{-\frac{1}{2}} \circ \frac{\partial(\mathbf{E})}{\partial \overline{\boldsymbol{\ell}}} \right) \right] .$$

The remaining simplifications can be achieved by substitution with the definitions given after Eq. (11.8) and by applying general rules for differentiation such as the chain rule

$$\frac{\partial f(g(\mathbf{X}))}{\partial \mathbf{x}} = \frac{\partial (f(\mathbf{U}) \colon)}{\partial \mathbf{U}} \cdot \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \quad \Big|_{\mathbf{U}=g(\mathbf{X})}$$

where $\mathbf{X}\colon$ denotes the vectorization of a matrix by stacking its columns, e.g., as applied to a term containing element-wise division

$$\frac{\partial(\mathbf{A} \div \mathbf{B})}{\partial \mathbf{x}} = \mathbf{A} \circ \frac{\partial \ \mathrm{inv}(\mathbf{U}) \colon}{\partial \mathbf{U} \colon} \cdot \frac{\partial \mathbf{B} \colon}{\partial \mathbf{x}} \quad \Big|_{\mathbf{U}=\mathbf{B}}$$

for a matrix $\mathbf{A}$ that does not depend on $\mathbf{x}$. Substituting the resulting partial derivatives in Eq. (11.9) yields the gradient $\partial L(\boldsymbol{\theta})/\partial \boldsymbol{\theta}$, which can be used in gradient-based optimization techniques, such as Møller's [1993] scaled conjugate gradients (SCG), to jointly optimize the hyperparameters of $\mathcal{GP}_y$ and $\mathcal{GP}_\ell$.

11.2 Learning Large Datasets

A significant practical limitation of the standard Gaussian process model is its $\mathcal{O}(n^3)$ runtime complexity, where n is the number of training points. This is due to the inversion of the covariance matrix during learning and prediction (in the latter case only if predictive variances are required). Since the covariance functions typically employed in practice (e.g., from the Matérn class) and also their nonstationary extensions such as Eq. (11.3) model local influences, the local experts approach is a sensible choice that scales linearly in n under some mild model assumptions.

11.2.1 Model Tiling

We propose to approximate the nonstationary model outlined above using a mixture of nonstationary GP "experts". Our local models are distributed uniformly over the space allowing for fast indexing. Their individual responses mix within overlapping regions with neighboring models according to an analytic gating function. More concretely, we consider

M Gaussian processes $\mathcal{GP}^{[j]}, j = 1 \dots M$, and set

$$p(y_* \mid \mathbf{x}_*, \mathcal{D}) = \sum_{j=1}^{M} g^{[j]}(\mathbf{x}_*)\; p^{[j]}(y_* \mid \mathbf{x}_*, \mathcal{D}), \quad g^{[j]} = \frac{w^{[j]}}{\sum_{j=1}^{M} w^{[j]}(\mathbf{x}_*)}\;, \tag{11.10}$$

where $p^{[j]}(y_* \mid \mathbf{x}_*, \mathcal{D})$ is the posterior of a target y_* at location $\mathbf{x}_*$ according to the j'th expert model $\mathcal{GP}^{[j]}$. For brevity, we omit the hyperparameters $\boldsymbol{\theta}$ here. Making the locality of the expert models explicit, we define *support regions* $S^{[j]}$ plus surrounding *border regions* $B^{[j]}$ respectively and set $w^{[j]}$ for the individual experts to

$$w^{[j]}(\mathbf{x}_*) = \begin{cases} 1\;, & \text{for } \mathbf{x}_* \in S^{[j]} \\ \omega(d; l^{[j]})\;, & \text{for } \mathbf{x}_* \in B^{[j]}, d = \min_{\mathbf{p} \in S^{[j]}}(\|\mathbf{x}_* - \mathbf{p}\|) \\ 0\;, & \text{for } \mathbf{x}_* \notin S^{[j]} \cup B^{[j]}\;. \end{cases} \tag{11.11}$$

Here, we model the "blurring" $\omega(d; l)$ of the border region $B^{[j]}$ inspired by the truncated covariance of Storkey [1999] on $[0, \infty)$ as

$$\omega(d; l) = \begin{cases} [(2\pi - d/l)(1 + 1/2\cos(d/l)) + 3/2\sin(d/l)]/(3\pi)\;, & \text{for } d < 2\pi l, \\ 0\;, & \text{otherwise}\;. \end{cases}$$

This function approximates $n(d) \propto \mathcal{N}(d; 0, \sigma)$ scaled to $n(0) = 1$ with $\sigma \approx 5/3\, l$. We maximize the extent of expert j on $S^{[j]} \cup B^{[j]}$ by setting the width parameter l to $\max_{\mathbf{x} \in B^{[j]}} \min_{\mathbf{p} \in S^{[j]}}(\|\mathbf{x} - \mathbf{p}\|)/2\pi$. Note, that this is straightforward to calculate, e.g., for rectangular or circular regions $S^{[j]}$ and $B^{[j]}$. Also note, that $w^{[j]}(\mathbf{x})$ is continuous, differentiable, and non-zero only on a bounded region. Our experience with this formulation and with alternatives for defining the locality of models within the tiling setup of Eq. (11.10) confirms the statement of Cleveland and Loader [1996] about local regression: If the gating function for local models is (a) continuous, (b) non-zero only on a bounded region, and (c) ensures enough overlap between the mixed models, the actual choice of gating function is not too critical.

11.2.2 Learning Nonstationary Tiled Models

We now apply model tiling as described above and the MAP approximation for nonstationary GPs as described in the previous section to achieve a tractable approximation of Eq. (11.5) even for large datasets. Specifically, we consider tiled approximations for both, $\mathcal{GP}_y$ and $\mathcal{GP}_\ell$. Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, one could in principle compute the gradient of Eq. (11.6) w.r.t. the hyperparameters $\boldsymbol{\theta} = \langle \boldsymbol{\theta}_y, \boldsymbol{\theta}_{\boldsymbol{\ell}}, \overline{\boldsymbol{\ell}} \rangle$ of the covariance

functions including the latent length-scales $\overline{\boldsymbol{\ell}}$:

$$\frac{\partial L(\boldsymbol{\theta}, \overline{\boldsymbol{\ell}})}{\partial \bullet} = \sum_{i=1}^{n} \frac{\partial}{\partial \bullet} \log \sum_{i=1}^{M} g^{[j]}(\mathbf{x}_i)\ p(y_i \mid \mathbf{X}, \exp(\hat{\boldsymbol{\ell}}), \boldsymbol{\theta}_y) + \sum_{i=1}^{n} \frac{\partial}{\partial \bullet} \log \sum_{i=1}^{M} g^{[j]}(\mathbf{x}_i)\ p(\hat{\boldsymbol{\ell}} \mid \mathbf{X}, \overline{\boldsymbol{\ell}}, \overline{\mathbf{X}}, \boldsymbol{\theta}_\ell)$$

Applying the chain rule to the logarithms of the two sums would provide us with the desired partial derivatives. This, however, is quite complicated and tedious. Therefore, we instead maximize the (simpler to differentiate) lower bound of the objective function by recalling Jensen's inequality [Jebara and Pentland, 2000], i.e., $f(\mathrm{E}(x)) \geq \mathrm{E}(f(x))$ for concave f, which allows us to move the sums out of the logarithms,

$$\frac{\partial \tilde{L}(\boldsymbol{\theta}, \overline{\boldsymbol{\ell}})}{\partial \bullet} = \sum_{i=1}^{n} \sum_{i=1}^{M} g^{[j]}(\mathbf{x}_i)\ \frac{\partial}{\partial \bullet} \log p(y_i \mid \mathbf{X}, \exp(\hat{\boldsymbol{\ell}}), \boldsymbol{\theta}_y) + \cdots \ . \tag{11.12}$$

Experiments have shown that optimization based on scaled conjugate gradients (SCG) using this lower bound works well in practice. Starting from Eq. (11.12), one can now calculate the gradients using standard linear algebra and the definitions in Sec. 11.1.3. Note that this involves differentiation of the nonstationary covariance function [Eq. (11.3)], which is more efficient to implement if done in matrix notation directly (e.g., avoiding the explicit summation over data points in Eq. (11.12)).

11.2.3 Discussion

Approximation: In the extreme case of $B^{[j]} = \emptyset$ and $S^{[i]} \cap S^{[j]} = \emptyset, \forall i, j$, the gating function becomes binary and the space is assigned to the experts in a Voronoi-diagram-like manner. In the other extreme, if $S^{[i]} = S^{[j]}, \forall i, j$, there is an infinite overlap between all models and Eq. (11.10) simplifies to $p(\mathbf{y} \mid \mathbf{x}) = \sum_{i=1}^{M} 1/M\ p^{[i]}(\mathbf{y} \mid \mathbf{x})$ and all expert models become equal, $p^{[i]} \propto p^{[j]}, \forall i, j$. In this case, since every expert model $\mathcal{GP}_y^{[i]}$ is a full nonstationary GP, the tiled approximation Eq. (11.10) is a perfect one. The suitability of Eq. (11.10) as an approximation of a single, large nonstationary model in the case of small, local, overlapping models $\mathcal{GP}_y^{[i]}$ becomes apparent when one considers the *equivalent kernel* [Sollich and Williams, 2004] for Gaussian processes. Under this view, the GP mean and variance predictions, Eq. (11.1) and (11.2), are expressed as dot products $\mu(\mathbf{x}_*) = \mathbf{h}(\mathbf{x}_*)^T \mathbf{y}$ of the vector of target values $\mathbf{y}$ and weight vectors $\mathbf{h}(\mathbf{x}_*)$. Here, the weight function $\mathbf{h}(\mathbf{x}_*)$, aka the equivalent kernel (EK), depends both on the query location $\mathbf{x}_*$ and on the covariance function k. Due to the inversion of the covariance matrix in Eq. (11.1), the EK is not straightforward to calculate even for the simple case of the stationary squared

exponential k_{se}. Sollich and Williams [2004] derive an approximation for this case, which shows that the EK is strongly localized and, thus, disregarding target values y that lie far away from the query point $\mathbf{x}_*$ introduces a minimal error for $\mathbf{h}(\mathbf{x}_*)^T\mathbf{y}$. This observation is indeed the basis for other of Gaussian process approximations, such as [Shen *et al.*, 2005]. Numerical experiments revealed that the nonstationary k_{nse} as defined in Eq. (11.3) is stronger or equally localized than the stationary squared exponential components it is composed of. Thus, we can safely apply the tiled approximation outlined above to both the latent length-scale process $\mathcal{GP}_\ell$ as well as to the nonstationary $\mathcal{GP}_y$ discussed in the previous section. For ease of implementation, we use the same tiling and indexing for both processes—but this is not necessary in general.

Time Complexity: To guarantee linear complexity in the unit lengths of covered input space, we apply an additional *binning* operation using a fine-grained grid as a pre-processing step. As argued in [Shen *et al.*, 2005], the weighted influence of data points (in the weight-space view on Gaussian process regression, see above) is approximately equal on local regions that are small compared to the local bandwidth of the covariance function. Thus, we can safely pool observations that fall into a grid cell to one *pseudo-observation*, which is assigned an increased "weight" in the model by dividing the corresponding diagonal entry of the identity matrix $\mathbf{I}$ of $(\mathbf{K}_{\mathbf{x},\mathbf{x}} + \sigma_n\mathbf{I})^{-1}$ by the number of pooled observations (thereby reducing the "noise" of the pseudo-observation). The binning operation ensures that the number of training points per local model is bounded by a constant b, yielding linear complexity $O(n)$ for adding points to the preprocessing grid and $O(M \cdot b^3)$ for learning all local models (not involving the complexity of learning the hyperparameters). Similar to Krause *et al.* [2008], one could derive explicit bounds of the GP approximation error introduced by this binning operation. In the context of a sensor-placement task, they show that the error of choosing the next optimal input location in terms of maximal information gain is bounded if one constraints the search to a grid approximation of the continuous input space.

11.3 Experimental Results

The goal of our experimental evaluation was to investigate to which extent the point estimate approach to nonstationary GP regression is able to handle input-dependent smoothness and to quantify the gains relative to the stationary model. Specifically, we designed several experiments to investigate whether the approach can solve standard regression problems discussed in the literature. We also applied it to two hard and relevant regression problems from embedded systems and robotics. On the two standard test sets, we demonstrate that the prediction accuracy of our approach is comparable to the one

achieved by the MCMC-based method by Paciorek and Schervish [2004], which, compared to our algorithm, is substantially more demanding regarding the computational resources. We will first evaluate the dense nonstationary GP model in the next sections and then discuss our results for the tiled approximation in Sec. 11.3.4.

We have implemented and evaluated our approach in `Matlab`. Using the compact matrix notation for all derivations, the core algorithm is implemented in less than 150 lines of code and, more importantly, advanced optimization strategies like sparse matrix approximations or parallelization can be realized with virtually no additional implementation efforts. As optimization procedure, we applied Møller's scaled conjugate gradient (SCG) [Møller, 1993] approach. In all our experiments, the SCG converged after at most 20 iterations. To evaluate the performance of our nonstationary regression technique quantitatively, we ran 30 to 50 independent test runs for each of the following test cases. Each run consisted of (a) randomly selecting or generating training data, (b) fitting the nonstationary model, and (c) evaluating the predictive distribution of the learned model at independent test locations. The latter was done either using the known ground truth function values or by assessing the likelihood of independent observations in the cases in which the ground truth was not known (e.g., for the RFID and terrain mapping experiments).

In all test scenarios, we evaluate the accuracy of the mean predictions and also the fit of the whole predictive distribution using the *standardized mean squared error* (sMSE) and the *negative log predictive density* (NLPD):

$$\text{sMSE} = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \mu_i^*)^2}{\mathbb{V}(y)}\,, \qquad \text{NLPD} = \frac{1}{n}\sum_{i=1}^{n} -\log p(y_i|\mathbf{x}_i)\ .$$

Here, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ denotes the test data set, $p(\cdot|\mathbf{x}_i)$ stands for the predictive distribution at location $\mathbf{x}_i$, and $\mu_i^* := \mathbb{E}[p(\cdot|\mathbf{x}_i)]$ denotes the predicted mean. Statistical significance was assessed using two-sample t-tests with 95% confidence intervals.

All experiments were conducted using `Matlab` on a Linux desktop PC with a single 2 GHz CPU. The typical runtime for fitting the full nonstationary model to 100 training points was in the order of 50 seconds. The runtime requirements of the MCMC-based approach [Paciorek and Schervish, 2004] which does not employ any gradient information were reported to be in the order of hours for a C-implementation on standard hardware in the year 2004. In the following, we term our nonstationary approach as LA-GP (Locally Adaptive GP), the standard model employing the isotropic, squared exponential covariance function as STD-GP and Paciorek and Schervish's MCMC-based approach as NS-GP (Nonstationary GP).

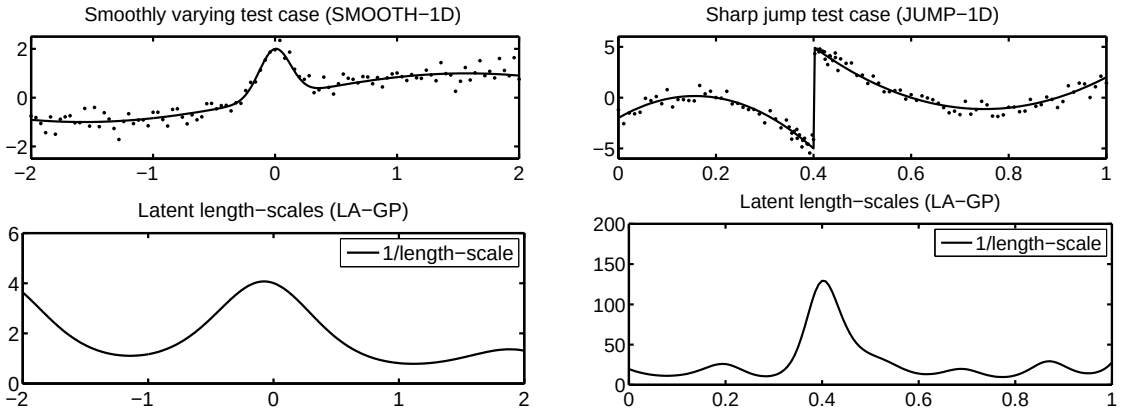


Figure 11.2: Two standard nonstationary test cases SMOOTH-1D (top left) and JUMP-1D (top right) that were used for evaluation purposes in lower two plots give the inverse latent length-scales as optimized by our approach. Higher values in these plots indicate a larger local frequency.

11.3.1 Simulation Results in 1D and 2D

First, we verified that our approach accurately solves standard regression problems described in the literature. To this aim, we considered the two simulated functions shown in Fig. 11.2. Both functions were also used for evaluation purposes by Dimatteo *et al.* [2001] and in [Paciorek and Schervish, 2004]. In the remainder, these test scenarios will be referred to as SMOOTH-1D and JUMP-1D. Whereas SMOOTH-1D is a smoothly varying function with a substantial "bump" close to 0, JUMP-1D has a sharp jump at 0.4. For SMOOTH-1D, we sampled 101 training points and 400 test points from the interval $(-2, 2)$. In the case of JUMP-1D, we sampled 111 training points and 111 for testing from $(0, 1)$. Table 11.1 gives the results for these experiments (averaged over 50 independent runs). Additionally, this table contains results for a two-dimensional simulated function NONSTAT-2D, which is described further below in this sub-section.

Table 11.1: Quantitative evaluation of the proposed nonstationary approach (LA-GP) and the standard Gaussian process (STD-GP) as well as the MCMC-based approach of Paciorek and Shervish (NS-GP). We compare the prediction accuracies using the negative log predictive density (NLPD) and the standardized mean squared errors (sMSE), see text. Results marked by • differ significantly ($\alpha = 0.05$) from the others in their category.

	NLPD		sMSE		
Test Scenario	**LA-GP**	**STD-GP**	**LA-GP**	**STD-GP**	**NS-GP**
SMOOTH-1D	-1.100	-1.026 (•)	0.0156	0.021 (•)	0.015
JUMP-1D	-0.375	-0.440 (•)	0.0268	0.123 (•)	0.026
NONSTAT-2D	-3.405	-3.315 (•)	0.0429	0.0572 (•)	-

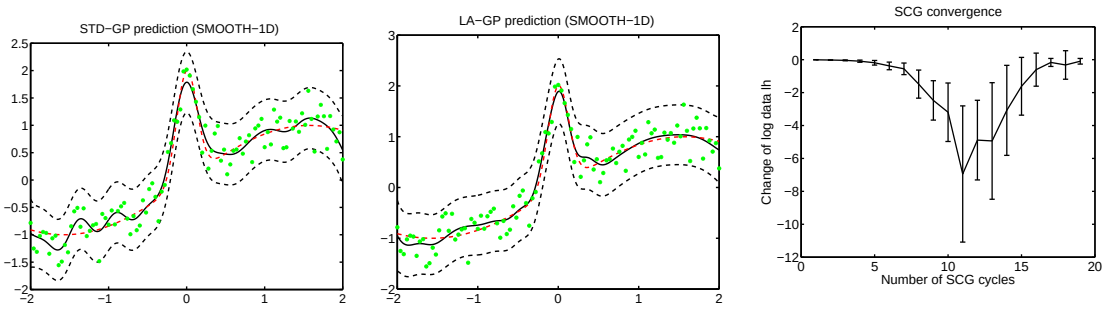


Figure 11.3: Typical regression results in the SMOOTH-1D test scenario for the STD-GP model (left) and LA-GP (middle). The right diagram gives the statistics for changes of the objective function per SCG optimization cycle (in log data liklihood).

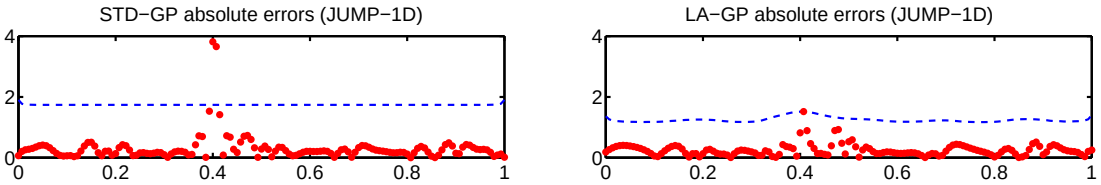


Figure 11.4: Absolute distances of the test points from the predicted means in one run of the JUMP-1D scenario using the STD-GP model (left) and LA-GP (right). The model confidence bounds (2 standard deviations of the predictive distribution) are given by dashed lines.

The results can be summarized as follows: With respect to the sMSE, the accuracy of our approach is comparable to the MCMC-based method of Paciorek and Schervish. Note that values given here were taken from their publication [Paciorek and Schervish, 2004]. Both approaches significantly (α=0.05) outperform standard GPs. Our approach also provides a significantly better performance compared to standard GPs with respect to the NLPD. For a visual comparison of the regression results, consider the diagrams on the left in Fig. 11.3. Whereas the standard GP (left plot) – having a constant length-scale for the whole domain – cannot adapt to all local properties well, our LA-GP accurately fits the bump and also the smoother parts (center plot). It should be noted that LA-GP tends to assign higher frequencies to the border regions of the training set, since there is less constraining data there compared to the center regions (see also the lower left plot in Fig. 11.2).

The right diagram in Fig. 11.3 provides statistics about the individual gains during the SCG cycles for 50 independent test runs. As can be seen from this plot, after about 20 cycles the objective function, which corresponds to the negative log data likelihood, does not change notably any more. Figure 11.4 compares the confidence bounds of the different regression models to the actual prediction errors made. It can be seen that the LA-GP model has more accurate bounds. It should be noted that the predictive variance of the STD-GP model depends only on the local data density and not on the target values and, thus, it is close to constant in the non-border regions.

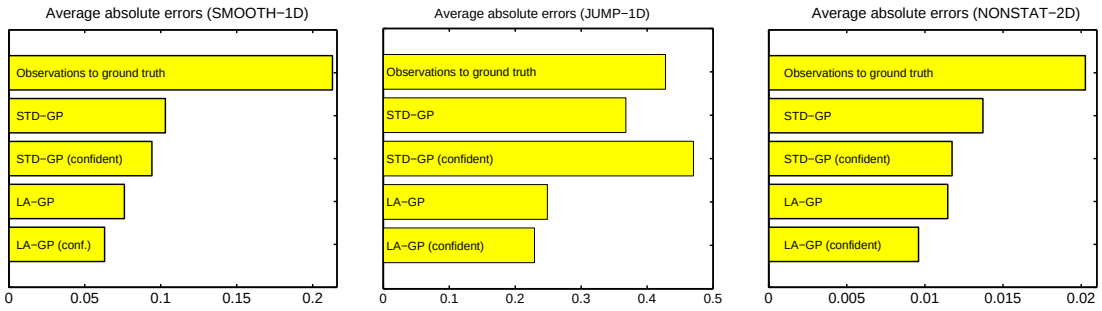


Figure 11.5: Absolute average errors of the mean predictions in the SMOOTH-1D test scenario (left), JUMP-1D (middle), and NONSTAT-2D (right). We give the absolute distances of the simulated observations to the true function values, the overall average errors for the different models, and the average errors of the 50% most confidently predicted means, respectively.

We give the absolute average errors of the mean predictions in the different test cases in Fig. 11.5. To highlight the more accurate confidence bounds of the LA-GP model, we also give the statistics for the 50% most confident predictions.

In addition to the two one-dimensional standard test cases, we evaluated the performance of our approach on a bivariate function (NONSTAT-2D). In particular, we simulated observations $y(x_1, x_2) \sim f(x_1, x_2) + \mathcal{N}(0, 0.025)$ using the noise-free bivariate function $f(x_1, x_2) = 1/10 \cdot (\sin(x_1\, b(x_1, x_2) + \sin(x_2\, b(x_1, x_2))$ and the underlying bandwidth function $b(x_1, x_2) = \pi\ (2x_1 + 0.5x_2 + 1)$. This function and typical observations are depicted in the left diagram in Fig. 11.6. During training, we sampled $11 \cdot 11 = 121$ points from a uniform distribution over $[-0.5, 1] \times [-0.5, 1]$ and corresponding simulated observations (the latter were drawn independently for each run). For testing, we uniformly sampled $31 \cdot 31 = 961$ points from $[-0.5, 1] \times [-0.5, 1]$ including their true function values. A typical example of the resulting optimized length-scales are visualized in the upper right contour plot in Fig. 11.6. It can be seen that larger length-scales (which correspond to stronger smoothing) are assigned to the flat part of the surface around $(-0.5, 0)^T$ and smaller ones towards $(1, 1)^T$.

The quantitative results in terms of NLPD and sMSE for 30 independent test runs are given in Tab. 11.1. The absolute errors of the mean predictions are given in the right chart in Fig. 11.5. The two lower plots in Fig. 11.6 give a visual impression about the accuracy of the two regression models. We show the *local* NLPD loss at equidistantly sampled test locations overlayed by contour plots of the predictive uncertainties. Note that the LA-GP model assigns higher confidence to the flat part of the function, which – given the uniform sampling of training points – can be reconstructed more accurately than the higher-frequency parts.

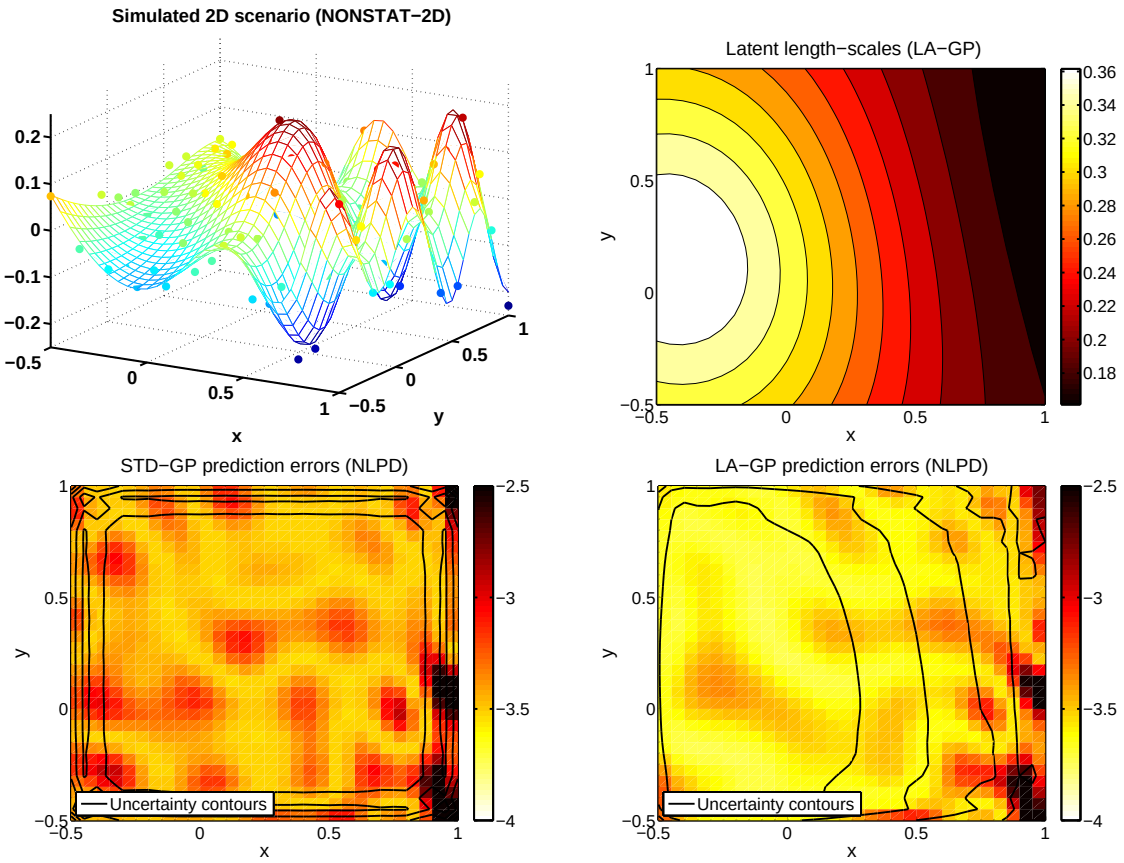


Figure 11.6: The true function and noisy observations in the NONSTAT-2D test case (top left). Note the spatially varying oscillation frequency. The top right plot depicts the contours of the latent length-scales as estimated by our LA-GP model. In the two lower diagrams, we give the individual prediction errors (NLPD) of the Standard GP model (bottom left) and LA-GP (bottom right). The predictive uncertainty of the models is visualized using overlayed contours.

11.3.2 Modeling RFID Signal Strength

We have applied our nonstationary regression approach to the problem of learning the signal strength distribution of RFID (Radio Frequency Identification) tags. For this experiment, 21 794 (log) signal strength measurements have been recorded in a test setup at the University of Freiburg using a static antenna and a mobile, externally localized RFID tag. For efficiency reasons, only the left half-space of the antenna was sampled with real measurements and then mirrored along the respective axis. We randomly sampled 121 training points for learning the regression models and 500 different ones for evaluation. Note that although larger training sets lead to better models, we learn from this comparably small number of observations to achieve faster evaluation runs.

Table 11.2 gives the quantitative comparison to the standard GP model (STD-GP). As

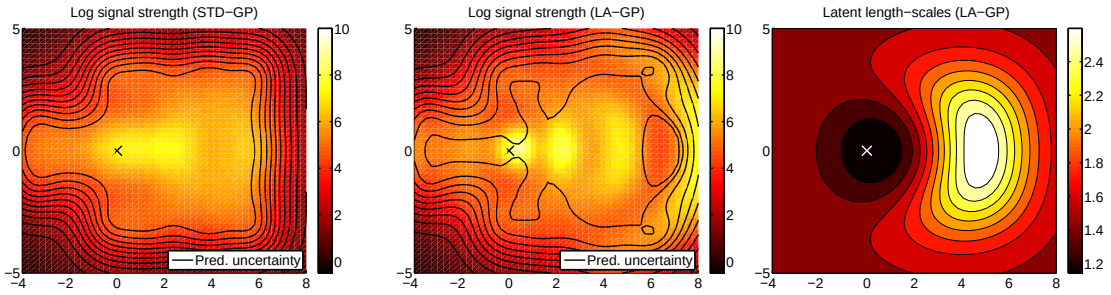


Figure 11.7: Predicted mean log signal strengths of RFID tags using the standard GP (left) and the locally adapted GP (middle). The sensor location (0,0) is marked by a cross and the predictive uncertainties of the models are visualized by overlayed contours. The right plot visualizes the adapted latent length-scales of the LA-GP model. Coordinates are given in meters.

can be seen from the results, the standard model is outperformed by our nonstationary extension both in terms of sMSE and NLPD. Figure 11.7 shows predicted mean log signal strengths of the two models as color maps overlayed with contour plots of the corresponding predictive uncertainties. We also visualize the contours of the latent length-scales modeling higher frequencies in the proximity of the sensor location and lower ones at approx. $(4, 2)$.

11.3.3 Laser-based Terrain Mapping

We also applied our model to the particularly hard robotics problem of learning probabilistic terrain models from laser range measurements. In a joint project with the Massachusetts Institute of Technology, we have equipped a quadruped robot with a Hokuyo URG laser range sensor (see the left picture in Fig. 11.8). The robot was programmed to perform a 'pushup' motion sequence in order to acquire a 3D scan of the local environment. For evaluation, we selected a $20 \times 20cm$ part of a rough terrain (with a maximum height of around 9 cm) including its front edge (see the right plot in Fig. 11.8). 4 282 laser end points of the 3D scan fall into this area.

We have trained the standard GP model and our nonstationary variant on 80 randomly selected training points from a noise-free simulation of the real terrain (TERSIM-2D) and evaluated the prediction accuracy for 500 test points (30 runs). We repeated the same procedure on the real data (TERREAL-2D) and evaluated the prediction accuracy for

Table 11.2: Quantitative results for the RFID-2D experiment. Results marked by • differ significantly ($\alpha = 0.05$) from the others in their category.

	NLPD		sMSE	
Test Scenario	**LA-GP**	**STD-GP**	**LA-GP**	**STD-GP**
RFID-2D	-0.0101 (•)	0.1475	0.3352 (•)	0.4602

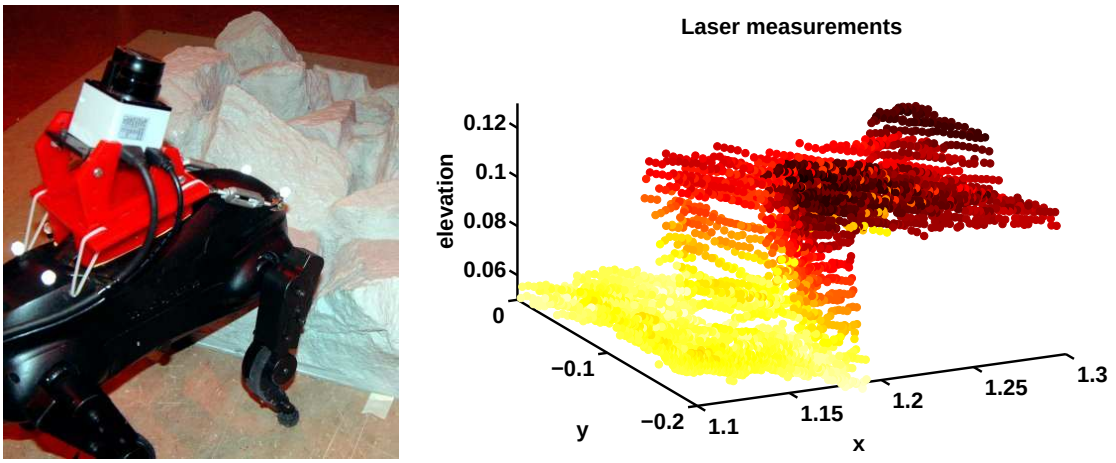


Figure 11.8: Left: A quadruped robot equipped with a laser sensor acquires elevation measurements of a rough terrain surface. Right: Part of the scanned terrain that was used for evaluation.

other, independently selected test points from the real scan. Thus, the latter evaluation quantifies how well the models are able to predict other samples from the same distribution while the former gives the prediction errors relative to a known ground truth function. Table 11.3 gives the quantitative results for these two experiments. The right colormap in Fig. 11.8 depicts the optimized length-scales of the LA-GP model. It can be seen that the flat part of the terrain is assigned larger local kernels compared to the rougher parts.

11.3.4 Tiled Nonstationary Models

For evaluating the tiled approximation of the proposed model, the objectives are twofold: (a) to investigate the gain in speed of the tiling approach and (b) to quantify a possible loss in prediction accuracy. To this aim, we compare the following regression methods: STD-GP is the standard GP implementation that was also used in the previous evaluation. Hyperparameters are learned by maximizing the marginal log data likelihood via conju-

Table 11.3: Quantitative results for the simulated (TERSIM-2D) and the real (TERREAL-2D) terrain mapping experiment. Results marked by • differ significantly ($\alpha = 0.05$) from the others in their category.

	NLPD		sMSE	
Test Scenario	**LA-GP**	**STD-GP**	**LA-GP**	**STD-GP**
TERSIM-2D	-4.261 (•)	-4.198	0.127	0.126
TERREAL-2D	-3.652	-3.626	0.441 (•)	0.475

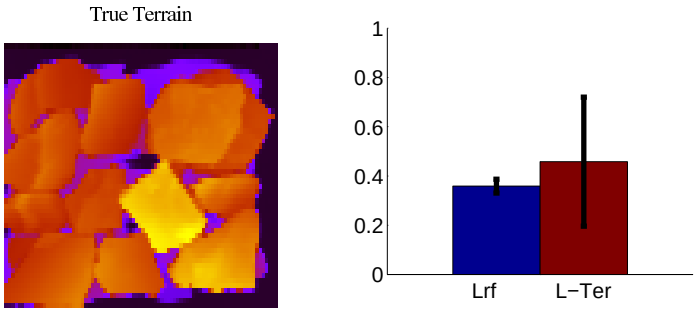


Figure 11.9: Left: Terrain model **L-Ter**. The color encodes the elevation. (Right) Average run time reductions on **L-Ter** and **Lrf**, i.e., the ratio of the run time of our tiled, nonstationary GP and the run time of a global nonstationary GP. The smaller the value the higher the speed-up. The black bar shows one standard deviation.

gate gradient. LA-GP in this section denotes our mixture-of-nonstationary-GPs model. Here, we first use cross-validation to fix the hyperparameters of the length-scale process, and then adapt the GP to the training data by maximizing the marginal log data likelihood using a scaled conjugate gradient algorithm. To compare run time and predictive performance with another state-of-the-art non-linear regression method, we also considered *locally weighted projection regression*, LWPR [Vijayakumar *et al.*, 2005a] setting the parameters as suggested in [Klanke and Vijayakumar, 2008].

We compare the methods on the following synthetic and real-world data sets in the context of mobile robotics:

Sine-Bumps is an enlarged version of the SMOOTH-1D data set used in Sec. 11.3.1. It is the sum of the sine function $y = \sin(x)$ and several "bumps" caused by an additive exponential term $2 \cdot exp(-30 \cdot (x - c)^2)$. In the interval $x = [-10, 10]$ we uniformly sampled 5 bump centers c. We take 500 evenly-spaced data points as the training set and 1000 data points as the test set. Then, we perform 30 independent runs in which we add Gaussian white-noise with a standard deviation of $\sigma = 0.3$ to the y-values of the training set. We evaluate the regression functions using the "noise-free" test set. Where cross-validated (CV) learning of fixed parameters was necessary, we produce a CV-training and CV-test with the same data densities but only a fifth of the size, using the same function over $x = [-2, 2]$ and placing a bump at $c = 0$. We add the same noise $\sigma = 0.3$ to the training set and measure performance on the test set.

L-Ter steams from a real-world terrain model shown in Fig. 11.9 (left) that maps x,y coordinates to elevations. We cut 10 slices at different fixed y coordinates and learn the mapping from x to z. Every slice has about 180 measurements with heights between approximately 4cm and 12cm in an x-range of 1m, which we use as test set. For the training set, we add Gaussian white-noise with standard deviation $\sigma = 0.3$cm

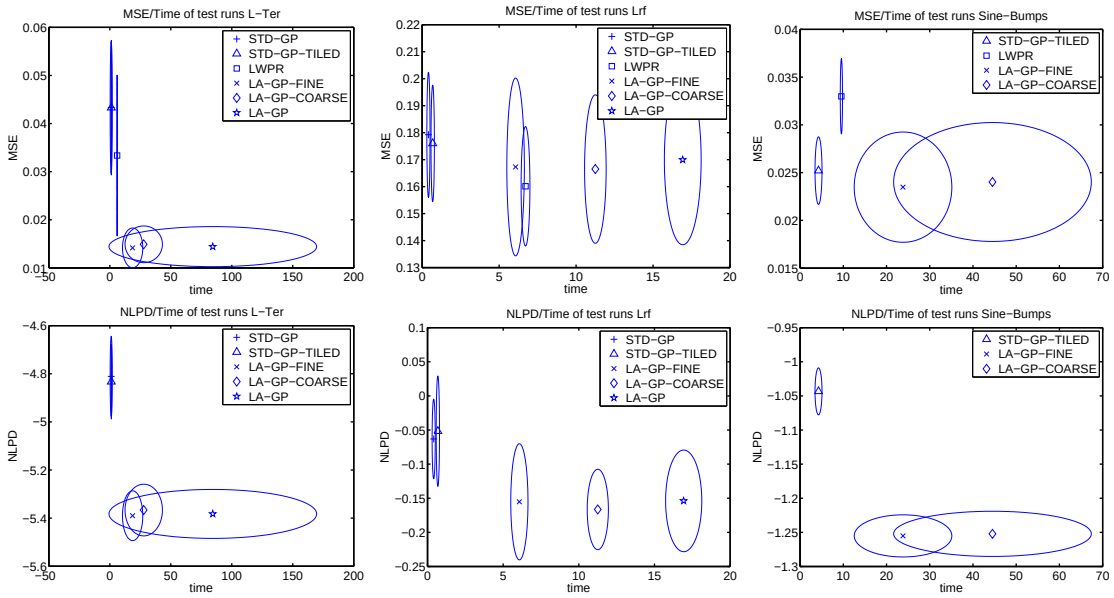


Figure 11.10: Time-performance trade-offs for the different data sets (from left to right: **L-Ter**, **Lrf**, and **Sine-Bumps**). The upper row shows CPU time (x axis) vs. sMSE (on y axis) whereas the lower row CPU time (x axis) vs. NLPD (on y axis). In both cases, the ellipses indicate the average values and the standard deviations in both dimensions.

to the test set. As cross-validation set, we use an arbitrary slice and produce training and test data in the same fashion.

Lrf consists of three laser range finder (LRF) readings taken in an indoor environment. The task is to map an angle ϕ to a range r. The LRF has an opening angle of 180 degrees, in which it measures 60 ranges at evenly spaced angles. We produce 30 training and test sets by randomly selecting half of the data as training and half of the data as test set. We generate a CV set in the same way.

The results are summarized in Fig. 11.9 (right) and Fig. 11.10. Figure 11.9 (right) clearly shows that our tiling approach is significantly faster than the corresponding global nonstationary GP approach. This is also confirmed by the individual time-performance trade-offs as shown in Fig. 11.10: Tiled nonstationary GPs (LA-GP-FINE respective LA-GP-COARSE) are significantly faster (t-test, $p = 0.05$) than global ones (coincides with LA-GP) and show comparable predictive performance; tiled nonstationary GPs show significantly better performance than STD-GP to the expense of a higher run time—as expected. LWPR and the two LA-GP models are similar in speed, while LA-GP significantly outperforms LWPR in every scenario except **Lrf**. Note that **Lrf** was the scenario that exhibited the least nonstationary behavior and, hence, also the standard GP is competitive here.

To investigate the scaling of our approach, we conducted an experiment on **Sine-Bumps**

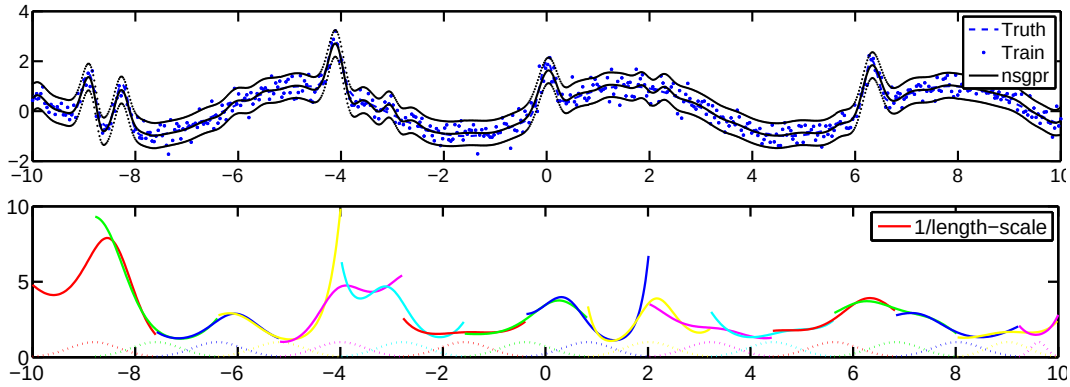


Figure 11.11: Results on large-scale **Sine-Bumps**. The above plot shows the training data and the predictive means and variance of the estimated tiled, nonstationary GP model on the central twentieth of the dataset. The bottom plot shows the corresponding tiled, inverse latent length-scales as optimized by our approach. We use a different color for every segment. The weights assigned by the gating function are shown by the dotted line at the bottom. Note that the "outliers" at the boundaries of each tile are basically ignored.

enlarged by factor 20 (10 000 training examples). We found that both the nonstationary and stationary segmented model achieve equal MSE and NLPD values, while their training time scales only linearly. This is especially remarkable for the nonstationary model, in which the number of hyperparameters also grows by factor 20 to 10 002, but the necessary SCG iterations remained constant.

11.4 Related Work

In previous work [Lang *et al.*, 2007], we modeled 3D terrain data using nonstationary GPs by also following the approach of Paciorek and Schervish [2004] (see also the previous two chapters). There, adaptation to local smoothness, was achieved by "external" tuning of the covariance function using the elevation structure tensor (EST) to measure the local smoothness. Another approach to modeling nonstationarity is to use ensembles of GPs, where every GP is assigned to a specific region, an idea akin to GP mixture models such as presented by Williams [2006]. Cornford *et al.* [1999] model straight discontinuities in wind fields by placing auxiliary GPs along the edge on both sides of the discontinuity. They are then used to learn GPs representing the process on either side of the discontinuity.

Apart from Paciorek and Schervish's [2004] approach (see also the references in there) of directly modeling the covariance function using additional latent GPs, several other approaches for specifying nonstationary GP models can be found in the literature. For instance, Sampson and Guttorp [1992] map a nonstationary spatial process (not based on GPs) into a latent space, in which the problem becomes approximately stationary. Schmidt

and O'Hagan [2003] followed this idea and used GPs to implement the mapping. Similar in spirit, Pfingsten *et al.* [2006] proposed to augment the input space by an additional latent input to tear apart regions of the input space that are separated by abrupt changes of the function values. However, all GP approaches proposed so far followed a Markov chain Monte Carlo approach to inference and learning. Instead, we present a novel maximum-a-posterior treatment of Paciorek and Schervish's approach that fully stays in the GP framework, explicitly models the covariance function, provides continuous estimates of the local kernels, and that naturally allows for gradient-based joint optimization of its parameters.

With our model tiling approach introduced in Sec. 11.2.1, we intended to close the gap between two common approaches to modeling nonstationary GPs: Directly modeling the covariance function using additional latent GPs and mixtures of local experts [Tresp, 2000, Rasmussen and Ghahramani, 2002, Williams, 2006], in which the global model is decomposed into multiple smaller ones, each responsible only for a subset of the data.

11.5 Conclusion

We showed that GP regression with nonstationary covariance functions can be realized efficiently using point estimates of the latent local smoothness. The experimental results have shown that the resulting locally adaptive GPs perform significantly better than standard GPs and that they have the potential to solve hard learning problems from robotics and embedded systems.

Regarding the tiled approximation, the experimental results show that the model produces prediction results that are more sensible than stationary GPs and comparable with locally weighted regression. Compared to its global nonstationary counter-part, it has significantly reduced time requirements while still achieving competitive results.

Discussion and Outlook

Chapter 12

Discussion and Outlook

State-of-the-art robotic systems are required to perform complex functions, which often cannot be realized without employing advanced tools for learning and probabilistic inference. We showed in this thesis that the Gaussian process approach to function regression can be used to solve a variety of relevant robot learning problems. Driven by concrete real-world problems, we developed several generally-applicable extensions of the Gaussian process model, which proved helpful also in a number of non-robotics learning tasks. In the following, we summarize our results in the subareas of robot learning that are covered by this work. For each area, we discuss what has been achieved and outline directions for future research.

Sensor Modeling and Perception Interpreting sensor measurements is one of the fundamental problems studied in robotics, since knowledge about the world provides the basis for any decision-making in robotic systems. Although state-of-the art sensors are able to sense in a wide range of modalities at a high precision, the central question remains how to model a given sensor for a specific task. Here, the accurate handling of observation noise and of complex realistic environments is of special importance.

In Chapter 7, we introduced a novel sensor model for laser range finders that is based on the idea of reasoning in the space of continuous range functions rather than in the discrete domain of beam measurement vectors. The proposed model provides superior robustness compared to state-of-the-art sensor models for range sensors for the tasks of mobile robot localization and tracking. As a second contribution of this chapter, we discussed in depth the problem of overly peaked likelihood functions in conjunction with approximative estimation techniques, such as the particle filter. We derived a practical solution to this problem and showed its benefits in a simulation study as well as for localization a real mobile robot. Directions for future work include modeling of nonstationary covariances

along the lines of Chapter 11 or mixtures (Chapter 5) to increase the accuracy of the model further without weakening its ability to generalize well already from few observations.

Chapter 8 described a novel approach for learning range functions from monocular visual input. Our model utilizes edge-based features extracted from the image and applies the PCA to find a low-dimensional representation of the visual input in an unsupervised manner. In experiments with a real robot, we achieved an accuracy in range predictions comparable to that of dedicated range sensors based on infrared or sonar signals. The number of potential applications of this method is large since, in principle, any robotics-related task involving range functions (such as place classification, mapping, or active exploration) could be approached in the described way. Future research should involve evaluating alternative techniques for dimensionality reduction, especially those considering the target values to be predicted (like supervised PCA [Yu *et al.*, 2006] or LDA [McLachlan *et al.*, 1992, Fisher, 1936]) or others that are directly integrated into the GP framework.

Additional perception-related topics were addressed in conjunction with other topics. For instance, radio frequency identification (RFID) sensors were modeled to test our approach to nonstationary GP regression in Chapter 11. The experimental results suggest that the nonstationary model is well suited to learn and represent distributions of signal strength measurements. In future work, such a model could be used to estimate the locations of RFID tags in an environment or, given their locations, to estimate the location of the sensor. As another perception-related topic, visual marker patterns were used in Chapter 4 to learn the relationship between a robot's actions and changes to its body configuration. It was shown that our Gaussian processes-based approach to body-scheme learning is capable of estimating the configuration of a manipulator arm with an error that is lower than the observation noise.

State Estimation and Self-Modeling In Chapter 3, it was shown that the efficiency of sampling in dynamic Bayesian networks can be improved greatly by employing learned proposal distributions. It was described how continuous as well as discrete state variables can be treated in a consistent manner using GPs for regression and classification, respectively. The approach was applied to the problem of online collision detection for mobile robots. Experiments with a real robot demonstrated that the developed system is able to track the state of the robot more reliably through collision events than an optimized version of a standard particle filter with uninformed proposals. Future work in this area could consider learning sampling models for arbitrary dynamic Bayesian networks, addressing feature discovery, multi-class classification, and the question of dependencies between failures. For a limited set of such discrete failure modes, the combination of the proposed system with the look-ahead particle filter [de Freitas *et al.*, 2003] may be beneficial.

A novel approach to body-scheme learning and life-long adaptation for a robotic manipulation system was developed in Chapter 4. The central idea was to continuously learn a large set of local kinematic models using Gaussian processes and to search for the best arrangement of these models for representing the full system. In experiments carried out with a real robot and in simulation, it was demonstrated that the system was able to deal with missing and noisy observations, operating in full 3D space, and being able to perform relevant tasks such as prediction, control, and online adaptation after failures. Challenging topics for further investigation include developing an active exploration strategy, learning from marker-less observations, point-like features, or range observations, and learning under the presence of never-observed body parts.

State estimation, specifically mobile robot localization, was also a central topic in Chapter 7, in which we developed the novel range sensor model. The main contribution of the chapter in this regard was a novel view on regularization for particle filtering-based state estimation. It was discussed that highly-peaked observation likelihood functions should be integrated over in local environments around the state hypotheses to decrease the variance of the filter estimates. Applied to mobile robot localization with laser range finders, this regularization procedure causes the individual laser beams—which are typically assumed to be independent—to be statistically dependent. Experiments showed that this dependency can efficiently be dealt with using Gaussian process regression over the range functions, requiring only about 5% of the training data that direct estimation of a joint Gaussian distribution requires.

Map Building Building a representation of the environment from sensor measurements is a heavily researched field within robotics. Two particularly hard map learning problems have been considered in this thesis. In Chapter 5, a novel approach to gas distribution mapping based on sparse Gaussian process mixture models was developed. The model is able to account for the typical bimodal nature of gas distributions and it can be learned efficiently by employing an online sparsification strategy. The method was implemented and tested using gas sensors mounted on a real robot. As a distinct advantage compared to state-of-the-art methods in gas distribution modeling, the proposed approach also estimates the predictive uncertainties in a consistent manner. In future research, it should be considered to evaluate nonstationary kernels as a possible alternative to the mixture model (cmp. Chapter 11). Another topic could involve modeling the diffusion in high concentration areas by smoothing the gating function over time.

Two models for terrain modeling based on Gaussian process regression were introduced in Chapters 9 and 10. As the central concept, both approaches employ nonstationary covariance functions adapted to the local terrain structure by using the elevation structure tensor

(EST). The EST was introduced as a new concept, which is related to tensor representations established in the computer vision community, that capture the local characteristics of the terrain. Compared to a state-of-the-art approach from the robotics literature the new approach achieves a reduction of the prediction error by approximately 30% to 70%. The second terrain mapping approach (Chapter 10) addressed terrain modeling for legged robots. The developed and implemented system enables a legged robot equipped with a laser range finder to autonomously sense the terrain in front of it, and to build a probabilistic elevation model that (a) represents the densely sampled structure accurately and (b) is able to make sensible predictions about unseen parts. As experiments with a real robot showed, these features allow a quadruped robot to plan foot trajectories and to reach a goal location more robustly than when using grid-based terrain models.

In future work, the model could be used to improve decision-theoretic path planning by offering the ability to accurately simulate potential future observations and to temporarily include these into the map in order to assess the value of observatory actions. Furthermore, future work could consider a reinforcement learning variant of foot trajectory planning along the lines of Neumann *et al.* [2007] using specific reward functions to learn obstacle avoidance or stable and energy-efficient movements.

Advanced Methods for Non-Linear Regression Additionally to progress in the above-mentioned subfields of robot learning, the following extensions of Gaussian process regression have been developed and evaluated. Chapter 6 has shown that effective Gaussian process regression with input-dependent noise can be fully implemented using standard GP techniques without resorting to time-consuming MCMC approximations. In experimental tests, our approach produced estimates that are significantly better than standard GPs and competitive with other regression approaches that handle input-dependent noise. Directions for future work include studying online learning, classification, and applications within other learning tasks such as reinforcement learning. Furthermore, it would be interesting to investigate "almost sure convergence" along the lines of [Bottou and Bengio, 1995] and to understand it from a variational Bayes perspective. In the spirit of Chapter 11 and as outlined in Section 6.2, the possibility of fixing the local noise rates to the mean predictions of the noise GP should be investigated in order to achieve a tractable model for *joint* gradient-based optimization of hyperparameters and latent noise rates.

In Chapter 11, a model for GP regression with nonstationary covariance functions was developed that can be learned efficiently using point estimates of the latent local smoothness. The ability to adapt to local smoothness can be regarded as one of the major preconditions for scalability, since especially large data sets cannot be assumed to exhibit the same characteristics at all locations. Our model has been shown to clearly outperform station-

ary GP regression in terms of accuracy and MCMC-based nonstationary GPs in terms of runtime requirements. Following this approach, there are several interesting directions for future work. First, the idea of jointly optimizing the parameters of the latent and the observed process should be applied to GP regression with input-dependent noise. In robotic applications, one is likely to encounter both, input-dependent noise and variable smoothness. Hence, the joint treatment of both should be addressed. Another direction is the extension of our approach to the pseudo-input setting introduced by Snelson and Ghahramani [2006b], so that the locations of the length-scale support values are learned from data, too. Finally, one should investigate multi-task learning, for example, along the lines of Yu *et al.* [2007] to generalize across different types of terrains.

Sparse approximations for Gaussian process models were discussed on several occasions (see Chapters 5, 6, and 11). These are essential for reducing the high runtime and space requirements of standard GPs to a tractable level. By making only mild assumptions about the functions to be learned, the tiled nonstationary GP approximation introduced in Chapter 11 was shown to scale linearly in the units of input space as well as in the number of data points. This is a remarkable result, especially since the model accuracy remains nearly unchanged. Nevertheless, in future work, the computational efficiency of this model could be increased further using, e.g., a kd-tree to access relevant local models quickly. Furthermore, the ability of the approach to deal with higher dimensional input spaces should be evaluated and online applications should be considered to demonstrate its practical usefulness under such hard constraints.

12.1 Concluding Remarks

To summarize, it has been shown that Gaussian process regression is well-suited for solving learning problems that arise in robotics research and development. In conjunction with the introduced extensions for learning input-dependent noise and smoothness, the model is able to achieve high prediction accuracies on complex, realistic data sets. On the other hand, it is possible to realize these benefits in a time and space efficient way by employing sparse approximation techniques.

We believe that Gaussian processes have a high potential also beyond the topics addressed in this work, especially in conjunction with approaches to dimensionality reduction, feature discovery, or for non-vectorial data like graphs.

Bibliography

[Arulampalam *et al.*, 2002] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[Bares *et al.*, 1989] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. Whittaker. Ambler — an autonomous rover for planetary exploration. *Computer*, 22(6):18–26, June 1989.

[Benazera *et al.*, 2004] E. Benazera, R. Dearden, and S. Narasimhan. Combining particle filters and consistency-based approaches. In *15th International Workshop on Principles of Diagnosis*, Carcassonne, France, 2004.

[Biller and Fahrmeir, 1997] C. Biller and L. Fahrmeir. Bayesian spline-type smoothing in generalized regression models. *Computational Statistics*, 12(2), 1997.

[Bongard and Lipson, 2007] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. In *Proc. of the Nat. Academy of Science*, volume 104, page 9943, 2007.

[Bongard *et al.*, 2006a] J. Bongard, V. Zykov, and H. Lipson. Automated synthesis of body schema using multiple sensor modalities. In *Proc. of the Int. Conf. on the Simulation and Synthesis of Living Systems (ALIFEX)*, 2006.

[Bongard *et al.*, 2006b] J. Bongard, V. Zykov, and H. Lipson. Resilient machines through continuous self-modeling. *Science*, 314.(5802):1118–1121, 2006.

[Bottou and Bengio, 1995] L. Bottou and Y. Bengio. Convergence properties of the kmeans algorithm. In *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1995.

[Brooks, 2005] M. Brooks. The matrix reference manual. `http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html`, 2005. [Online; accessed 2008-07-01].

[Burgard *et al.*, 2007] W. Burgard, C. Stachniss, and D. Haehnel. *Autonomous Navigation in Dynamic Environments*, volume 35 of *STAR Springer tracts in advanced robotics*, chapter Mobile Robot Map Learning from Range Data in Dynamic Environments. Springer Verlag, 2007.

[Canny, 1986] F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 679–714, 1986.

[Cawley *et al.*, 2006] G.C. Cawley, N.L.C. Talbot, and O. Chapelle. Estimating Predictive Variances with Kernel Ridge Regression. In *Machine Learning Challenges*, pages 56–77, 2006.

[Chan *et al.*, 2006] D. Chan, R. Kohn, D. Nott, and C. Kirby. Locally-adaptive semiparametric estimation of the mean and variance functions in regression models. *Journal of Computational and Graphical Statistics*, 2006.

[Choset *et al.*, 2005] H. Choset, K.M. Lynch, S. Hutchinson, G.A. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, June 2005.

[Cleveland and Devlin, 1988] W. S. Cleveland and S. J. Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610, 1988.

[Cleveland and Loader, 1996] W.S. Cleveland and C. Loader. Smoothing by local regression: Principles and methods. *Statistical Theory and Computational Aspects of Smoothing*, pages 10–49, 1996.

[Cocora *et al.*, 2006a] A. Cocora, K. Kersting, C. Plagemann, W. Burgard, and L. De Raedt. Learning relational navigation policies. *KI - Künstliche Intelligenz, Themenheft Lernen und Selbstorganisation von Verhalten*, 3:12–18, 2006.

[Cocora *et al.*, 2006b] A. Cocora, K. Kersting, C. Plagemann, W. Burgard, and L. De Raedt. Learning relational navigation policies. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[Cornford *et al.*, 1999] D. Cornford, I. Nabney, and C. Williams. Adding constrained discontinuities to gaussian process models of wind fields. In *Advances in Neural Information Processing Systems 11*. MIT Press, Cambridge, MA, 1999.

[Craig, 1989] J.J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[Dahlkamp *et al.*, 2006] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G.R. Bradski. Self-supervised monocular road detection in desert terrain. In *Proc. of Robotics: Science and Systems (RSS)*, 2006.

[Davies, 1997] E. R. Davies. Laws texture energy in texture. In *Machine Vision: Theory, Algorithms, Practicalities*. Acedemic Press, 1997.

[Davis, 2004] T.A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):165–195, 2004.

[Davision *et al.*, 2007] A. Davision, I. Reid, N. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(6), 2007.

[de Freitas *et al.*, 2003] N. de Freitas, R. Dearden, F. Hutter, R. Morales-Menendez, J. Mutch, and D. Poole. Diagnosis by a waiter and a mars explorer. In *Invited paper for Proceedings of the IEEE, special issue on sequential state estimation*, 2003.

[Dearden and Clancy, 2002] R. Dearden and D. Clancy. Particle filters for real-time fault detection in planetary rovers. In *Proceedings of the Thirteenth International Workshop on Principles of Diagnosis*, pages 1–6, 2002.

[Dearden and Demiris, 2005] A. Dearden and Y. Demiris. Learning forward models for robots. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1440–1445, Edinburgh, Scotland, 2005.

[Delage *et al.*, 2005] E. Delage, H. Lee, and A.Y. Ng. Automatic single-image 3d reconstructions of indoor manhattan world scenes. In *Proceedings of the 12th International Symposium of Robotics Research (ISRR)*, 2005.

[Dijkstra, 1959] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[Dimatteo *et al.*, 2001] I. Dimatteo, C.R. Genovese, and R.E. Kass. Bayesian curve-fitting with free-knot splines. *Biometrika*, 88(4):1055–1071, Dec. 2001.

[Doucet *et al.*, 2001] A. Doucet, N. de Freitas, and N. Gordan, editors. *Sequential Monte-Carlo Methods in Practice*. Springer Verlag, 2001.

[Doucet, 1998] A Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Departement of Engeneering, University of Cambridge, 1998.

[Driessen and Boers, 2004] J.N. Driessen and Y. Boers. An efficient particle filter for non-linear jump markov systems. In *IEEE Sem. Target Tracking: Algorithms and Applications, Sussex, UK*, 2004.

[DustBot, 2008] DustBot. DustBot - Networked and Cooperating Robots for Urban Hygiene. `http://www.dustbot.org`, 2008.

[Ecker and Heltshe, 1994] M.D. Ecker and J.F. Heltshe. Geostatistical estimates of scallop abundance. In *Case Studies in Biometry*, pages 107–124. John Wiley and Sons, 1994.

[Edakunni *et al.*, 2007] N.U. Edakunni, S. Schaal, and S. Vijayakumar. Kernel carpentry for online regression using randomly varying coefficient model. In Manuela M. Veloso, editor, *IJCAI*, pages 762–767, 2007.

[Elinas *et al.*, 2006] P. Elinas, R. Sim, and J.J. Little. σSLAM: Stereo vision SLAM using the rao-blackwellised particle filter and a novel mixture proposal distribution. In *Proc. of ICRA*, 2006.

[Ewerth *et al.*, 2007] R. Ewerth, M. Schwalb, and B. Freisleben. Using depth features to retrieve monocular video shots. In *Proceedings of the 6th ACM international conference on Image and video retrieval (CIVR)*, pages 210–217, New York, NY, USA, 2007. ACM.

[Favaro and Soatto, 2005] P. Favaro and S. Soatto. A geometric approach to shape from defocus. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):406–417, 2005.

[Fiala, 2004] M. Fiala. Artag, a fiducial marker system using digital techniques. Technical report, National Research Council Canada, 2004.

[Fisher, 1936] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[Fox *et al.*, 1999] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[Fox, 2001] D. Fox. Kld-sampling: Adaptive particle filters. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*. MIT Press, 2001.

[Früh and Zakhor, 2002] C. Früh and A. Zakhor. Data processing algorithms for generating textured 3d building façade meshes from laser scans and camera images. In *3DPVT*, pages 834–849, 2002.

[Früh *et al.*, 2005] C. Früh, S. Jain, and A. Zakhor. Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *Int. Journal of Computer Vision*, 61(2):159–184, 2005.

[Goldberg *et al.*, 1998] P.W. Goldberg, C.K.I. Williams, and C.M. Bishop. Regression with input-dependent noise: A gaussian process treatment. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, volume 10, 1998.

[Gordon *et al.*, 1993] N.J. Gordon, D.J. Salmond, and Smith A.F.M. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *Proceedings of IEE Part F*, 1993.

[Grimes *et al.*, 2006] D. Grimes, R. Chalodhorn, and R. Rao. Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, 2006.

[Grzonka *et al.*, 2007] S. Grzonka, C. Plagemann, G. Grisetti, and W. Burgard. Look-ahead proposals for robust grid-based slam. In *Proc. of the International Conference on Field and Service Robotics (FSR)*, Chamonix, France, July 2007.

[Grzonka *et al.*, 2008] S. Grzonka, C. Plagemann, G. Grisetti, and W. Burgard. Look-ahead proposals for robust grid-based slam with rao-blackwellized particle filters. *International Journal of Robotics Research (IJRR)*, 2008.

[Gutierrez-Osuna *et al.*, 1998] R. Gutierrez-Osuna, J.A. Janet, and R.C. Luo. Modeling of ultrasonic range sensors for localization of autonomous mobile robots. *IEEE Transactions on Industrial Electronics*, 45(4):654–662, August 1998.

[Ha and Kim, 2004] Y.S. Ha and H.H. Kim. Environmental map building for a mobile robot using infrared range-finder sensors. *Advanced Robotics*, 18(4):437–450, 2004.

[Han and Zhu, 2003] F. Han and S.-C. Zhu. Bayesian reconstruction of 3d shapes and scenes from a single image. In *IEEE Intern. Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis (HLK*, page 12, Washington, DC, USA, 2003.

[Hastie *et al.*, 2001] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani. *The elements of statistical learning*. Springer New York, 2001.

[Hayes *et al.*, 2002] A.T. Hayes, A. Martinoli, and R.M. Goodman. Distributed Odor Source Localization. *IEEE Sensors Journal, Special Issue on Electronic Nose Technologies*, 2(3):260–273, 2002.

[Haykin, 1998] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1998.

[Hoiem *et al.*, 2007] D. Hoiem, A.A. Efros, and M. Herbert. Recovering surface layout from an image. *IJCV*, 75(1), October 2007.

[Holmes and Spence, 2004] N.P. Holmes and C. Spence. The body schema and the multisensory representation(s) of peripersonal space. *Cognitive processes*, 5(2):94–105, 2004.

[Hsu *et al.*, 2005] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 3885–3891, 2005.

[Hugentobler, 2004] M. Hugentobler. *Terrain Modelling with Triangle Based Free-Form Surfaces*. PhD thesis, University of Zurich, 2004.

[Ishida *et al.*, 1998] H. Ishida, T. Nakamoto, and T. Moriizumi. Remote Sensing of Gas/Odor Source Location and Concentration Distribution Using Mobile System. *Sensors and Actuators B*, 49:52–57, 1998.

[Jebara and Pentland, 2000] T. Jebara and A. Pentland. On reversing jensen's inequality. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 231–237. MIT Press, 2000.

[Kavraki *et al.*, 1996] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*, 12(4), 1996.

[Kersting *et al.*, 2007a] K. Kersting, C. Plagemann, A. Cocora, W. Burgard, and L. De Raedt. Learning to transfer optimal navigation policies. *Advanced Robotics. Special Issue on Imitative Robots*, 21(9), September 2007.

[Kersting *et al.*, 2007b] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic gaussian process regression. In *International Conference on Machine Learning (ICML)*, Corvallis, Oregon, USA, March 2007.

[Khan *et al.*, 2004] Z. Khan, T.R. Balch, and F. Dellaert. An mcmc-based particle filter for tracking multiple interacting targets. In *ECCV (4)*, pages 279–290, 2004.

[Klanke and Vijayakumar, 2008] S. Klanke and S. Vijayakumar. A library for locally weighted projection regression. `http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/lwpr_doc.pdf`, 2008. [Online; accessed 5-June-2008].

[Koenig and Howard, 2004] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. technical report. Technical report, USC Center for Robotics and Embedded Systems, CRES-04-002, 2004.

[Kolmogoroff, 1941] A. Kolmogoroff. Interpolation und extrapolation von stationÃ¤ren zufÃ¤lligen folgen.(russian. german. *Bull. Acad. Sci. URSS, Ser. Math.*, 5:3–14, 1941.

[Kolter and Ng, 2007] J. Kolter and A. Ng. Learning omnidirectional path following using dimensionality reduction. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[Konolige and Chou, 1999] K. Konolige and K. Chou. Markov localization using correlation. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 1999.

[Krause *et al.*, 2008] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, February 2008.

[Kretzschmar *et al.*, 2008] H. Kretzschmar, C. Stachniss, C. Plagemann, and W. Burgard. Estimating landmark locations from geo-referenced photographs. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

[Krige, 1951] D.G. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, 52(6):119–139, 1951.

[Krysander, 2003] M. Krysander. *Design and Analysis of Diagnostic Systems Utilizing Structural Methods*. PhD thesis, Linköping University, Sweden, 2003.

[Kuss, 2006] M. Kuss. *Gaussian process models for robust regression, classification, and reinforcement learning*. PhD thesis, Technische Universität Darmstadt, Germany, 2006.

[Kwok and Fox, 2004] C. Kwok and D. Fox. Map-based multiple model tracking of a moving object. In *RoboCup 2004: Robot Soccer World Cup VIII*, pages 18–33, 2004.

[Kwok *et al.*, 2002] C. Kwok, D. Fox, and M. Meila. Real-time particle filters. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 1057–1064, 2002.

[Kwok *et al.*, 2003] C. Kwok, D. Fox, and M. Meila. Adaptive real-time particle filters for robot localization. In *Proc. of the IEEE International Conference on Robotics & Automation*, 2003.

[Lamon *et al.*, 2006] P. Lamon, C. Stachniss, R. Triebel, P. Pfaff, C. Plagemann, G. Grisetti, S. Kolsky, W. Burgard, and R. Siegwart. Mapping with an autonomous car. In *In IEEE/RSJ IROS 2006 Workshop: Safe Navigation in Open and Dynamic Environments*, Beijing, China, 2006.

[Lang *et al.*, 2007] T. Lang, C. Plagemann, and W. Burgard. Adaptive non-stationary kernel regression for terrain modeling. In *Robotics: Science and Systems (RSS)*, Atlanta, Georgia, USA, June 2007.

[Le *et al.*, 2005] Q.V. Le, A.J. Smola, and S. Canu. Heteroscedastic gaussian process regression. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 489–496, New York, NY, USA, 2005. ACM Press.

[Lehn-Schiøler *et al.*, 2004] T. Lehn-Schiøler, D. Erdogmus, and J. C. Principe. Parzen particle filters. In *ICASSP*, volume 5, pages 781–784, May 2004.

[Leveson, 1995] N. Leveson. *Safeware : System Safety and Computers*. Addison-Wesley Pub Co., Reading, Mass., 1995.

[Lilienthal and Duckett, 2004] A. Lilienthal and T. Duckett. Building Gas Concentration Gridmaps with a Mobile Robot. *Robotics and Autonomous Systems*, 48(1):3–16, 2004.

[Lilienthal *et al.*, 2006] A. Lilienthal, A. Loutfi, and T. Duckett. Airborne Chemical Sensing with Mobile Robots. *Sensors*, 6:1616–1678, 2006.

[Liu and West, 2001] J. Liu and M. West. Combined parameter and state estimation in simulation-based filtering. *Sequential Monte Carlo Methods in Practice*, 2001.

[Lowe, 2004] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[Luber *et al.*, 2008] M. Luber, K. Arras, C. Plagemann, and W. Burgard. Tracking and classification of dynamic objects: An unsupervised learning approach. In *Robotics: Science and Systems (RSS)*, Zurich, Switzerland, June 2008.

[MacKay, 2003] D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Pres, September 2003.

[Maravita and Iriki, 2004] A. Maravita and A. Iriki. Tools for the body (schema). *Trends in Cognitive Sciences*, 8(2):79–86, 2004.

[Matheron, 1973] G. Matheron. The intrinsic random functions and their applications. *Advances in Applied Probability*, 5:439–468, 1973.

[McLachlan *et al.*, 1992] G. J. McLachlan, J. Wiley, and W. InterScience. *Discriminant analysis and statistical pattern recognition*. Wiley New York, 1992.

[Meeds and Osindero, 2006] E. Meeds and S. Osindero. An alternative infinite mixture of gaussian process experts. In *Advances in Neural Information Processing Systems*, 2006.

[Meltzoff and Moore, 1997] A.N. Meltzoff and M.K. Moore. Explaining facial imitation: A theoretical model. *Early Development and Parenting*, 6:179–192, 1997.

[Meyer-Delius *et al.*, 2007] C. Meyer-Delius, C. Plagemann, G. von Wichert, W. Feiten, G. Lawitzky, and W. Burgard. A probabilistic relational model for characterizing situations in dynamic multi-agent systems. In *Proc. of the 31st Annual Conference of the German Classification Society on Data Analysis, Machine Learning, and Applications (GfKl)*, Freiburg, Germany, 2007.

[Michels *et al.*, 2005] J. Michels, A. Saxena, and A.Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *ICML*, pages 593–600, 2005.

[Middendorf and Nagel, 2002] M. Middendorf and H. Nagel. Empirically convergent adaptive estimation of grayvalue structure tensors. In *Proceedings of the 24th DAGM Symposium on Pattern Recognition*, pages 66–74, London, UK, 2002. Springer-Verlag.

[Mitchell, 1997] T. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[Møller, 1993] M. Møller. A Scaled Conjugate Gradient Algoritm for Fast Supervised Learning. *Neural Networks*, 6:525–533, 1993.

[Morasso and Sanguineti, 1995] P. Morasso and V. Sanguineti. Self-organizing body-schema for motor planning. *Journal of Motor Behavior*, 26:131–148, 1995.

[Moravec and Elfes, 1985] H.P. Moravec and A.E. Elfes. High resolution maps from wide angle sonar. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 116–121, 1985.

[Murphy, 2002] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, UC Berkeley, 2002.

[Musso *et al.*, 2001] C. Musso, N. Oudjane, and Le Gland. F. Improving regularized particle filters. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice, New York*, Statistics for Engineering and Information Science, chapter 12, pages 247–271. Springer?Verlag, 2001.

[Nabeshima *et al.*, 2005] C. Nabeshima, M. Lungarella, and Y. Kuniyoshi. Timing-based model of body schema adaptation and its role in perception and tool use: A robot case study. In *Proc. of the IEEE International Conference on Development and Learning (ICDL2005)*, pages 7–12, 2005.

[Natale, 2004] L. Natale. *Linking Action to Perception in a Humanoid Robot: A Developmental Approach to Grasping.* PhD thesis, Univerity of Genoa, Italy, May 2004.

[Neal, 1996] R.M. Neal. *Bayesian Learning for Neural Networks.* Springer, 1996.

[Neal, 1997] R.M. Neal. Monte carlo implementation of gaussian process models for bayesian regression and classification. Technical report, Dept. of Computer Science, University of Toronto., 1997.

[Neal, 1999] R.M. Neal. Regression and classification using gaussian process priors. *Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting, June 6-10, 1998*, 1999.

[Neumann *et al.*, 2007] G. Neumann, M. Pfeiffer, and Maass W. Efficient continuous-time reinforcement learning with adaptive state graphs. In *Proc. of the European Conference on Machine Learning (ECML)*, 2007.

[Ng *et al.*, 2005] B. Ng, A. Pfeffer, and R. Dearden. Continuous time particle filtering. In *Proceedings of the 19th IJCAI, Edinburgh*, 2005.

[Nott, 1996] D. Nott. Semiparametric estimation of mean and variance functions for non-gaussian data. *Computational Statistics*, 21:603–620, 1996.

[O'Hagan, 1978] A. O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society B*, 40(1), 1978.

[Opsomer *et al.*, 1997] J. Opsomer, D. Ruppert, M. Wand, U. Holst, and O. Hossjer. Kriging with nonparametric variance function estimation. Technical report, Iowa St. Univ., 1997.

[Orr, 1996] M. J. L. Orr. Introduction to radial basis function networks. *Center for Cognitive Science, University of Edinburgh, Scotland*, 1996.

[Paciorek and Schervish, 2004] C. Paciorek and M. Schervish. Nonstationary covariance functions for Gaussian process regression. In S. Thrun, L. Saul, and B. Schoelkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[Petrovskaya *et al.*, 2006] A. Petrovskaya, O. Khatib, S. Thrun, and A.Y. Ng. Bayesian estimation for autonomous object manipulation based on tactile sensors. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Orlando, Florida, 2006.

[Pfaff and Burgard, 2005] P. Pfaff and W. Burgard. An efficient extension of elevation maps for outdoor terrain mapping. In *Proc. of the Int. Conf. on Field and Service Robotics (FSR)*, pages 165–176, Port Douglas, QLD, Australia, 2005.

[Pfaff *et al.*, 2006] P. Pfaff, W. Burgard, and D. Fox. Robust monte-carlo localization using adaptive likelihood models. In *European Robotics Symposium*, Palermo, Italy, 2006.

[Pfaff *et al.*, 2007] P. Pfaff, C. Plagemann, and W. Burgard. Improved likelihood models for probabilistic localization based on range scans. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[Pfaff *et al.*, 2008a] P. Pfaff, C. Plagemann, and W. Burgard. Gaussian mixture models for probabilistic localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

[Pfaff *et al.*, 2008b] P. Pfaff, C. Stachniss, C. Plagemann, and W. Burgard. Efficiently learning high-dimensional observation models for monte-carlo localization using gaussian mixtures. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

[Pfingsten *et al.*, 2006] T. Pfingsten, M. Kuss, and C.E. Rasmussen. Nonstationary gaussian process regression using a latent extension of the input space. In *Extended Abstract in Proc. of ISBA Eighth World Meeting on Bayesian Statistics*, Valencia, Spain, 2006.

[Plagemann and Burgard, 2005] C. Plagemann and W. Burgard. Sequential parameter estimation for fault diagnosis in mobile robots using particle filters. In *Autonome Mobile Systeme 2005 (AMS)*, pages 197–202. Springer, 2005.

[Plagemann *et al.*, 2005] C. Plagemann, T. Müller, and W. Burgard. Vision-based 3d object localization using probabilistic models of appearance. In *Pattern Recognition, 27th DAGM Symposium, Vienna, Austria*, 2005.

[Plagemann *et al.*, 2006] C. Plagemann, C. Stachniss, and W. Burgard. Efficient failure detection for mobile robots using mixed-abstraction particle filters. In *European Robotics Symposium 2006*, 2006.

[Plagemann *et al.*, 2007a] C. Plagemann, D. Fox, and W. Burgard. Efficient failure detection on mobile robots using particle filters with gaussian process proposals. In *Proc. of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007.

[Plagemann *et al.*, 2007b] C. Plagemann, K. Kersting, P. Pfaff, and W. Burgard. Gaussian beam processes: A nonparametric bayesian measurement model for range finders. In *Robotics: Science and Systems (RSS)*, Atlanta, Georgia, USA, June 2007.

[Plagemann *et al.*, 2008a] C. Plagemann, F. Endres, J. Hess, C. Stachniss, and W. Burgard. Monocular range sensing: A non-parametric learning approach. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

[Plagemann *et al.*, 2008b] C. Plagemann, K. Kersting, and W. Burgard. Non-stationary gaussian process regression using point estimates of local smoothness. In *Proc. of the European Conference on Machine Learning (ECML)*, Antwerp, Belgium, 2008.

[Plagemann *et al.*, 2008c] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard. Learning predictive terrain models for legged robot locomotion. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.

[Purnamadjaja and Russell, 2005] A.H. Purnamadjaja and R.A. Russell. Congregation Behaviour in a Robot Swarm Using Pheromone Communication. In *Proc. of the Australian Conf. on Robotics and Automation*, 2005.

[Pyk *et al.*, 2006] P. Pyk *et al.* An Artificial Moth: Chemical Source Localization Using a Robot Based Neurona l Model of Moth Optomotor Anemotactic Search. *Autonomous Robots*, 20:197–213, 2006.

[Rasmussen and Ghahramani, 2002] C.E. Rasmussen and Z. Ghahramani. Infinite mixtures of gaussian process experts. In *Advances in Neural Information Processing Systems 14*, 2002.

[Rasmussen and Williams, 2006a] C.E. Rasmussen and C. Williams. GPML Matlab Code. `http://www.gaussianprocess.org/gpml/code/`, 2006. [Online; accessed 10-July-2008].

[Rasmussen and Williams, 2006b] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, Massachusetts, 2006.

[Rasmussen, 1996] C.E. Rasmussen. *Evaluation Of Gaussian Processes And Other Methods For Non-Linear Regression*. PhD thesis, Graduate Department of Computer Science, University of Toronto, 1996.

[Rasmussen, 2006] C.E. Rasmussen. Minimize. `http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize`, 2006. [Online; accessed 10-July-2008].

[Reiser *et al.*, 2008] U. Reiser, C. Mies, and C. Plagemann. Verteilte Software-Entwicklung in der Robotik – ein Integrations- und Testframework. In *Robotik*, Munich, Germany, 2008. In German.

[Riedmiller and Braun, 1992] M. Riedmiller and H. Braun. Rprop: A fast adaptive learning algorithm. *Proc. of the Int. Symposium on Computer and Information Science VII*, 1992.

[Roberts and Webster, 2002] P.J.W. Roberts and D.R. Webster. Turbulent Diffusion. In H. Shen, A. Cheng, K.-H. Wang, M.H. Teng, and C. Liu, editors, *Environmental Fluid Mechanics - Theories and Application*. ASCE Press, Reston, Virginia, 2002.

[Rosales and Gan, 2004] E.M. Rosales and Q. Gan. Forward and inverse kinematics models for a 5-dof pioneer 2 robot arm. Technical Report CSM-413, University of Essex, UK, 2004.

[Rottmann *et al.*, 2007] A. Rottmann, C. Plagemann, P. Hilgers, and W. Burgard. Autonomous blimp control using model-free reinforcement learning in a continuous state and action space. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[Roy and Thrun, 1999] N. Roy and S. Thrun. Online self-calibration for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[Rumelhart and McClelland, 1986] D.E. Rumelhart and J.L. McClelland. *Parallel distributed processing.* MIT Press, 1986.

[Sabe *et al.*, 2004] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara. Obstacle avoidance and path planning for humanoid robots using stereo vision. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, New Orleans, LA, USA, 2004.

[Sampson and Guttorp, 1992] P.D. Sampson and P. Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87(417):108–119, 1992.

[Saxena *et al.*, 2007] A. Saxena, S.H. Chung, and A.Y. Ng. 3-d depth reconstruction from a single still image. *Intern. Journal of Computer Vision (IJCV)*, 2007.

[Schiele and Crowley, 1994] B. Schiele and J. Crowley. A comparison of position estimation techniques using occupancy grids. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1628–1634, 1994.

[Schmidt and O'Hagan, 2003] A.M. Schmidt and A. O'Hagan. Bayesian inference for nonstationary spatial covariance structure via spatial d eformations. *Journal of the Royal Statistical Society, Series B*, 65:745–758, 2003.

[Schoelkopf *et al.*, 2000] B. Schoelkopf, A.J. Smola, R.C. Williamson, and P.L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

[Schoenberg, 1964] I.J. Schoenberg. Spline functions and the problem of graduation. *Proceedings of the National Academy of Sciences*, 52(4):947–950, 1964.

[Shen *et al.*, 2005] Y. Shen, A.Y. Ng, and M. Seeger. Fast gaussian process regression using KD-trees. In *NIPS*, 2005.

[Sigrist, 1994] M. Sigrist. *Air Monitoring by Spectroscopic Techniques*, volume 197 of *Chemical Analysis Series*. John Wiley and Sons, 1994.

[Sim and Little, 2006] R. Sim and J.J. Little. Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2082–2089, 2006.

[Sinz *et al.*, 2004] F. Sinz, J. Quinonero-Candela, G. Bakir, C. Rasmussen, and M. Franz. Learning depth from stereo. In *26th DAGM Symposium*, 2004.

[Smola and Bartlett, 2000] A.J. Smola and P.L. Bartlett. Sparse greedy gaussian process regression. In *NIPS*, pages 619–625, 2000.

[Snelson and Ghahramani, 2006a] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1259–1266, 2006.

[Snelson and Ghahramani, 2006b] E. Snelson and Z. Ghahramani. Variable noise and dimensionality reduction for sparse gaussian processes. In *Uncertainty in Artifical Intelligence*, 2006.

[Snelson and Ghahramani, 2007] E. Snelson and Z. Ghahramani, editors. *Local and global sparse Gaussian process approximations*, volume 11, 2007.

[Snelson *et al.*, 2003] E. Snelson, C.E. Rasmussen, and Z. Ghahramani, editors. *Warped Gaussian Processes*, 2003.

[Sollich and Williams, 2004] P. Sollich and C.K.I. Williams. Using the equivalent kernel to understand gaussian process regression. In *NIPS*, 2004.

[Stachniss *et al.*, 2005] C. Stachniss, D. Hähnel, W. Burgard, and G. Grisetti. On actively closing loops in grid-based FastSLAM. *Advanced Robotics*, 19(10):1059–1080, 2005.

[Stachniss *et al.*, 2008] C. Stachniss, C. Plagemann, A. Lilienthal, and W. Burgard. Gas distribution modeling using sparse gaussian process mixture models. In *Robotics: Science and Systems (RSS)*, Zurich, Switzerland, June 2008.

[Storkey, 1999] A.J. Storkey. Truncated covariance matrices and toeplitz methods in gaussian processes. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 55–60, 1999.

[Strasdat *et al.*, 2007] H. Strasdat, C. Stachniss, M. Bennewitz, and W. Burgard. Visual bearing-only simultaneous localization and mapping with improved feature matching. In *Fachgespräche Autonome Mobile Systeme (AMS)*, 2007.

[Sturm *et al.*, 2008a] J. Sturm, C. Plagemann, and W. Burgard. Adaptive body scheme models for robust robotic manipulation. In *Robotics: Science and Systems (RSS)*, Zurich, Switzerland, June 2008.

[Sturm *et al.*, 2008b] J. Sturm, C. Plagemann, and W. Burgard. Body scheme learning and life-long adaptation for robotic manipulation. In *Robot Manipulation: Intelligence in Human Environments (at RSS 2008)*, Zurich, Switzerland, June 2008.

[Sturm *et al.*, 2008c] J. Sturm, C. Plagemann, and W. Burgard. Unsupervised body scheme learning through self-perception. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

[Swaminathan and Grossberg, 2002] G. Swaminathan and S. Grossberg. Laminar cortical mechanisms for the perception of slanted and curved 3-D surfaces and their 2-D pictorical projections. *J. Vis.*, 2(7):79–79, 11 2002.

[Takeda *et al.*, 2006] H. Takeda, S. Farsiu, and P. Milanfar. Kernel regression for image processing and reconstruction. *IEEE Trans. on Image Processing*, 2006. To appear.

[Thrun *et al.*, 1998] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998.

[Thrun *et al.*, 2000] S. Thrun, D. Fox, W Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1–2):99–141, 2000.

[Thrun *et al.*, 2001] S. Thrun, J. Langford, and V. Verma. Risk sensitive particle filters. In *NIPS*, 2001.

[Thrun *et al.*, 2005] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[Thrun, 2001] S. Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5):335–363, 2001.

[Ting *et al.*, 2006] J. Ting, M. Mistry, J. Peters, S. Schaal, and J. Nakanishi. A bayesian approach to nonlinear parameter identification for rigid body dynamics. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[Torralba and Oliva, 2002] A. Torralba and A. Oliva. Depth estimation from image structure. *IEEE Transactions on Pattern Analysis and Machine Learning*, 2002.

[Tresp, 2000] V. Tresp. Mixtures of gaussian processes. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2000.

[Triebel *et al.*, 2006] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[Verma *et al.*, 2003] V. Verma, S. Thrun, and R. Simmons. Variable resolution particle filter. In *Proc of IJCAI*, 2003.

[Vijayakumar *et al.*, 2005a] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634, 2005.

[Vijayakumar *et al.*, 2005b] S. Vijayakumar, A. D'Souza, and S. Schaal. Lwpr: A scalable method for incremental online learning in high dimensions. *Edinburgh: Press of University of Edinburgh*, 2005.

[Wahba, 1990] G. Wahba. Spline models for observational data. *CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia: Society for Industrial and Applied Mathematics, 1990*, 1990.

[Wandel *et al.*, 2003] M. Wandel, A. Lilienthal, T. Duckett, U. Weimar, and A. Zell. Gas distribution in unventilated indoor environments inspected by a mobile robot. In *Proc. of the Int. Conf. on Advanced Robotics (ICAR)*, pages 507–512, 2003.

[Ware and Lad, 2003] R. Ware and F. Lad. Approximating the distribution for sums of products of normal variables. Technical Report UCDMS 2003/15, University of Canterbury, England, 2003.

[Werbos, 1994] P. J. Werbos. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting.* Wiley-Interscience, 1994.

[Wiener, 1964] N. Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series.* The MIT Press, 1964.

[Williams and Rasmussen, 1995] C.K.I. Williams and C.E. Rasmussen. Gaussian processes for regression. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 514–520. MIT Press, 1995.

[Williams, 1996] P.M. Williams. Using neural networks to model conditional multivariate densities. *Neural Computation*, 8:843–854, 1996.

[Williams, 2006] O. Williams. A switched Gaussian process for estimating disparity and segmentation in binocular stereo. In *Neural Info. Proc. Systems (NIPS)*, 2006.

[Witkin and Baraff, 1997] A. Witkin and D. Baraff. An introduction to physically based modeling. In *SIGGRAPH'97 Course Notes*, 1997.

[Yoshikawa *et al.*, 2004a] Y. Yoshikawa, K. Hosoda, and M. Asada. Binding tactile and visual sensations via unique association by cross-anchoring between double-touching and self-occlusion. In *Proc. of the International Workshop on Epigenetic Robotics*, pages 135–138, Genoa, Italy, 2004.

[Yoshikawa *et al.*, 2004b] Y. Yoshikawa, Y. Tsuji, K. Hosoda, and M. Asada. Is it my body? – body extraction from uninterpreted sensory data based on the invariance of multiple sensory attributes –. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004.

[Yu *et al.*, 2006] S. Yu, K. Yu, V. Tresp, H. P. Kriegel, and M. Wu. Supervised probabilistic principal component analysis. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 464–473, 2006.

[Yu *et al.*, 2007] K. Yu, V. Tresp, and A. Schwaighofer. Learning gaussian processes from multiple tasks. In *ICML 07*, 2007.

[Yuan and Wahba, 2004] M. Yuan and G. Wahba. Doubly penalized likelihood estimator in heteroscedastic regression. *Statistics and Probability Letter*, 69(1):11–20, 2004.