ALBERT-LUDWIGS-UNIVERSITY
FREIBURG

DEPARTMENT OF
COMPUTER SCIENCE

Autonomous Intelligent Systems Lab

Prof. Dr. Wolfram Burgard

Gaussian Processes for Terrain Modeling

MASTER THESIS

Tobias Johannes Lang

November 2006 – May 2007

ALBERT-LUDWIGS-UNIVERSITY FREIBURG
FACULTY OF APPLIED SCIENCES

Department of Computer Science
Autonomous Intelligent Systems Lab
Prof. Dr. Wolfram Burgard

Gaussian Processes for Terrain Modeling

Master Thesis

Author: Tobias Johannes Lang

Submitted on: May 21, 2007

First Referee: Prof. Dr. Wolfram Burgard
Second Referee: Prof. Dr. Hans Burkhardt
Supervisor: Christian Plagemann

Abstract

Three-dimensional digital terrain models are of fundamental importance in areas such as the geo-sciences and outdoor robotics. Accurate modeling requires the ability to deal with a varying data density and to balance spatial smoothing against the preservation of structural features like discontinuities. The latter is particularly important for robotics applications, as discontinuities that arise, for example, at steps, stairs, or building walls are important features for path planning or terrain segmentation tasks. In this thesis, we present an extension of the well-established Gaussian process regression technique, that utilizes non-stationary covariance functions to locally adapt to the structure of the terrain data. In this way, we achieve strong smoothing in flat areas and along edges while at the same time preserving edges and corners. The derived model yields predictive distributions for terrain elevations at arbitrary locations. This allows us to fill gaps in the data and to perform conservative predictions of terrain elevations in occluded areas. Our model is able to account for all types of occluded areas, i.e., highly non-linear heterogenous terrain structures as well as linear discontinuities. An essential component in our model are local kernels, that capture the terrain properties of individual input locations and enable us to achieve non-stationarity in an interpretable and easy to visualize way. We investigate and evaluate two different learning algorithms to adapt the individual kernel parameters to terrain data: a gradient ascent approach over the data-likelihood and an approach based on local terrain gradients.

Zusammenfassung

Die Verarbeitung und Repräsentation räumlicher Daten hat eine zentrale Bedeutung in Bereichen wie den Geowissenschaften und der Robotik. Digitale Geländemodelle sollten mit variierender Datendichte umgehen können, fehlerbehaftete Daten glätten und dabei gleichzeitig Strukturmerkmale wie Ecken und Kanten erhalten. Letzteres ist besonders wichtig im Falle von Robotikanwendungen, da dort Unstetigkeiten, wie sie beispielsweise bei Stufen, Treppen oder Gebäudewänden auftreten, wichtige Merkmale für die Pfadplanung oder Terrainsegmentierung darstellen. In dieser Arbeit stellen wir eine Erweiterung des Gaußschen Prozessansatzes für Regressionsprobleme vor, welche mittels einer nichtstationären Kovarianzfunktion die lokale Adaptierung an die Terrainstruktur ermöglicht. Dadurch können räumliche Daten in flachen, homogenen Gebieten stark geglättet werden, während Ecken und Kanten erhalten bleiben. Das erlernte Modell bietet prädiktive Verteilungen für Geländehöhen an beliebigen Stellen des Grundraumes. Dadurch wird es möglich, Löcher in den beobachteten Daten zu schließen und in verdeckten Gebieten konservative Vorhersagen für die Terrainhöhen zu treffen. Das vorgestellte Modell kann dabei alle Arten verdeckten Terrains repräsentieren, d.h., sowohl hoch nicht-lineare, heterogene Geländestrukturen als auch lineare Unstetigkeiten und Kanten. Zentraler Bestandteil des Modells ist das Konzept lokaler Kernelstrukturen, die die Terraineigenschaften an einzelnen Stützstellen repräsentieren und die anschauliche Darstellung des nichtstationären Charakters ermöglichen. Zur Adapation der Kernelparameter an die zugrundeliegende Geländestruktur stellen wir zwei alternative Lernverfahren vor: einerseits die unmittelbare Optimierung der Datenlikelihood mittels Gradientenaufstieg und andererseits ein schnelles, an die Bildverarbeitungsliteratur angelehntes Verfahren basierend auf lokalen Geländegradienten. Die entwickelten Verfahren wurden vollständig implementiert und in simulierten und realen Testszenarien evaluiert.

Acknowledgements

Ich danke Professor Wolfram Burgard sowie allen Mitgliedern seines Lehrstuhls für Autonome Intelligente Systeme für ihre Hilfe in vielen kleinen und größeren Dingen und vor allem dafür, dass ich mich während meiner Arbeit im Lab sehr wohlgefühlt habe.

Ich danke Christian Plagemann für die tolle Betreuung meiner Masterarbeit. Es war eine meiner besten Entscheidungen, unter seiner Verantwortung meine Abschlussarbeit zu machen. Ich danke Patrick Pfaff und Rudolph Triebel für die Bereitstellung ihrer Datensätze und ihres Codes sowie für ihre Hilfe bei der Roboter-Hardware. Ich danke Rainer Kümmerle für seine Hilfe mit unserem kleinen Roboter Herbert. Ich danke Kristian Kersting für viele anregende Unterhaltungen. Ich danke Professor Hans Burkhardt dafür, dass er sich als Zweitgutachter zur Verfügung gestellt hat.

Ich danke meinen Eltern für ihre Unterstützung während meines gesamten Studiums, vor allem aber während des letzten Jahres.

I thank Professor Wolfram Burgard as well as all members of his Autonomous Intelligent Systems lab for their help in many situations and for making the lab an enjoyable place to work.

I thank Christian Plagemann for having been an excellent supervisor. It was one of my best decisions to work under his responsibility. I thank Patrick Pfaff and Rudolph Triebel for supplying many data-sets, for their help with robot hardware and for providing their code. I thank Rainer Kümmerle for his help with our small robot Herbert. I thank Kristian Kersting for many inspiring discussions. I thank Professor Hans Burkhardt for agreeing to be the second referee of this thesis.

I thank my parents for their support during my studies, in particular during the last year.

Contents

Chapter 1

Introduction

The modeling of three-dimensional terrain data has been widely studied across different research areas such as the geo-sciences or outdoor robotics. Important applications in outdoor robotics include mobile robots for agriculture, search and rescue, surveillance, and space missions (Figure 1.1). In these domains, accurate and dense models of the three-dimensional structure of the environment enable the robot to estimate the traversability of locations, to plan its path to a goal location, or to localize itself using range sensor measurements. Another application is, for instance, seafloor mapping where terrain models need to be built from high-precision ultrasound measurements. Seafloor maps, as illustrated in Figure 1.2, help to identify areas of erosion on the continental shelf and of geohazards, to locate pathways for movement of sediment and pollutants, to build underwater constructions such as pipelines or to retrieve sunken ships. Building a digital terrain model means to transform a set of sensory inputs, typically a three-dimensional point cloud or raw range sensor readings, to a function which maps two-dimensional pose coordinates to elevation values. While geological applications often operate on a large spatial scale, where local terrain features are not important, autonomous robots greatly rely on distinct structural features like edges or corners to guide navigation, localization, or terrain segmentation. Consider, for example, an autonomous car driving in urban terrain as depicted in Figure 1.3(a). 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. We therefore have two at first glance contradicting requirements for terrain models: 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 path planning, localization and object recognition. Furthermore, uncertainty estimates for predictions need to be incorporated to represent uncertainty on all levels as required in the probabilistic robotics approach [Thrun *et al.*, 2005]. This uncertainty is caused by two independent components, the noise in the data and the uncertainty in the estimation of the target function.

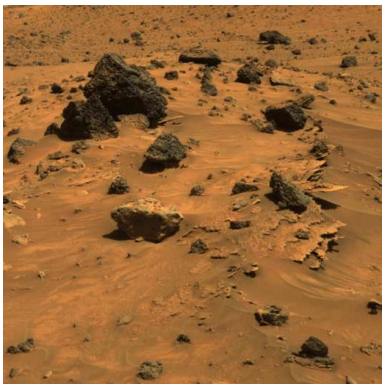


Figure 1.1: Surface of planet Mars. In space missions, robots need to cope with unkown terrains autonomously. (Source: http://www.nasa.gov/mission_pages/mer/)

1.1 The Terrain Modeling Problem

Our goal is to construct terrain models from sensory measurements. Data for building three-dimensional 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, which covers larger areas, as needed, e.g., for unmanned areal vehicles (UAVs) or autonomous cars. After various preprocessing steps, the raw measurements are typically represented as three-dimensional point clouds or are transformed into three-dimensional occupancy grids or elevation maps [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.

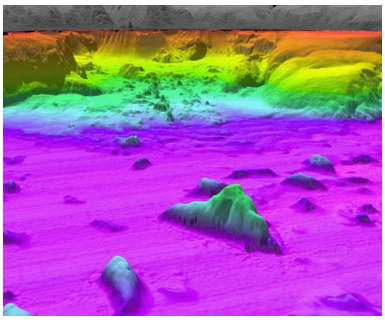
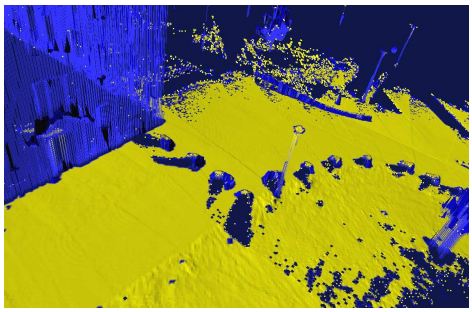


Figure 1.2: Seafloor map of Lake Tahoe in California (Source: http://walrus.wr.usgs.gov/pacmaps/)

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 a hard one 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, the proximity of the sensor location is usually more densely sampled than 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 completing small gaps by prediction, even single missing measurements may lead to the perception of a non-traversable obstacle (a hole in this case) 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 higher risks at these locations. Finally, as a last non-trivial requirement, the model should preserve structural elements like edges and corners as these are important features for various applications including path planning or object recognition.

(a) Partially non-traversable road curb (Source: http://www.pedestrians.org/images/)



(b) Occlusions within a surface map of the campus of Freiburg University

Figure 1.3: 3D terrain models need to be able to represent discontinuities and occlusions.

1.2 Related Work

Many important tasks in outdoor robotics such as planning and localization require data-structures for representing dense terrain data stemming from 3D range measurements. An efficient representation are elevation maps [Bares *et al.*, 1989, Pfaff and Burgard, 2005] which consist of two-dimensional grids in which each cell stores the elevation of the corresponding territory. Elevation maps have been extended to multi-level probabilistic surface (MLS) maps [Triebel *et al.*, 2006] which allow to store multiple surfaces in each cell of the grid. In a typical MLS map, many cells are empty due to occlusions and faulty measurements as visualized in Figure 1.3(b). One of the goals of this thesis was to estimate the elevations for cells where no measurements are available. In this regard, our approach can be viewed as orthogonal which can be used to preprocess spatial data that has to be represented in a MLS map. Most importantly, to achieve precise predictions we need to be able to cope with spatial inhomogeneity within terrains. Most studies using spatial or spatio-temporal data make the assumption that the modeled terrain is homogenous, i.e., its properties are the same over the complete input space. In other words, local characteristics are ignored. As pointed out by Sampson and Guttorp [1992], in most natural scenarios, this assumption is clearly violated: local influences are significant for the covariance structure of the input space which necessitates a heterogeneous model.

A straight-forward approach for filling gaps in three-dimensional models taking local properties into account is presented by Früh *et al.* [2005] who build three-dimensional models of facade meshes of cities from series of 2D scans. Due to faulty observations, e.g. caused by glass surfaces, and occlusions of the desired buildings by foreground objects, the generation of facade models is difficult. Früh *et al.* propose to fill local gaps based on local linear interpolation. They report promising results in several city mapping scenarios. Wellington *et al.* [2005] use multiple Markov random fields which interact through a hidden semi-Markov model to estimate terrain elevations and to enable classification for outdoor navigation. A classical approach for modeling non-stationary and anisotropic terrain properties, i.e., the properties vary with respect to different input locations and along different dimensions of the input space, is warping of the input space. Input locations are non-linearly mapped into a latent space which is characterized by a stationary covariance function. A classic reference for this approach is [Sampson and Guttorp, 1992]. They use two spaces, the G-space which is the geographical input space and the D-space measuring the dispersion of the input space. In order to calculate covariances for two input locations, these locations are mapped from G to D. The spatial structure within the D space is homogenous. Thin-plate splines are used there for estimation. The selection of the type of mapping function as well as using thin-plate splines for interpolation are arbitrary choices. Also, this model fails to account for uncertainty in the predictions. Schmidt and O'Hagan [2003] extended this approach to a fully Bayesian treatment. They represent the mapping between G and D by an unkown function $d(\cdot)$ for which they define a Gaussian process prior. This makes it possible to account for the uncertainty resulting from the mapping. Their model, however, has the drawback that the calculation of the posterior is not straightforward and needs to be done by means of computationally demanding Markov-Chain Monte-Carlo (MCMC) sampling.

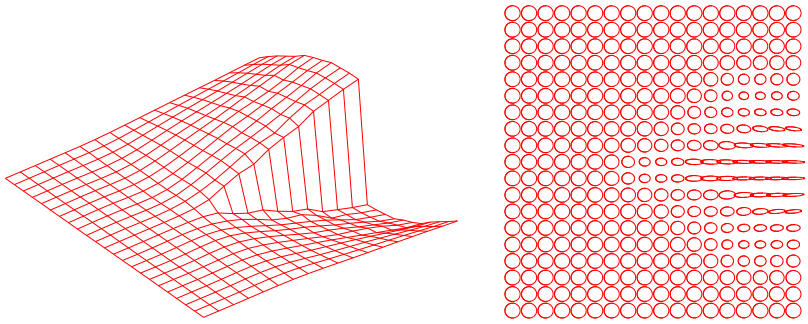
The idea of using Gaussian processes (GPs) for modeling spatial data is a quite old one. GPs first appeared in the field of geo-statistics. The mining engineer Daniel G. Krige explored the distance-weighted average gold grades at the Witwatersrand reef complex in South Africa. He developed a geo-statistic procedure for estimating the distribution of spatial data based on spatial dependencies [Krige, 1951]. His theory was further developed by the French geo-statistician Georges Matheron [Matheron, 1963] who called this procedure "Kriging" in honor of its inventor. GPs generalize the ideas of kriging to a wide range of regression and classification problems. Rasmussen and Williams [2006] as well as MacKay [1998] provide excellent introductions. GPs did not receive a greater popularity in other areas until the 1990s. Since then, GPs have come to the fore in machine learning and are now a well-understood and established technique. For instance, GPs have been applied successfully in different areas including positioning systems using cellular networks [Schwaighofer *et al.*, 2004], learning automatically generated music playlists [Platt *et*

al., 2000], or in bio-genetics [Chu *et al.*, 2005]. Recently, GPs have also become popular in the domain of robotics. For example, Brooks *et al.* [2006] use GPs to derive measurement models for mobile robot localization from sparse and noisy observations taken by a sensor with an unknown geometric model. The usage of GPs allows them to maintain an estimate of the uncertainty of the model over the entire map. Plagemann *et al.* [2007] use GPs for model-based failure detection. They employ GP models to learn poposal distributions for a particle filter which is applied to track the state of the robot and thereby to detect failure states such as collisions with unseen obstacles.

Most works on GP models assume stationary covariance structures over the input space. It is possible in various ways, however, to derive GP models that are able to account for inhomogeneity of the input space. For instance, one can partition the input space into segments and use mixtures of GPs where each GP is responsible for a different segment of the modeled environment. This is akin to the idea of partition models, which explain the data by fitting local distributions to local areas of the input space. For example, Denison *et al.* [2002] partition the input space by means of a Voronoi tessellation which is generalized by Blackwell and Moller [2003] to deformed tessellations. In the context of perception problems, Williams [2006] uses a GP framework in order to extract disparity information from binocular stereo images. He splits up the images into foreground and background images. A latent segmentation function assigns segment probabilities to pixels. Based on the segment, different covariance functions are employed in this *switched* GP which makes it possible to model both smooth regions and discontinuities. Another possibility to explicitely incorporate discontinuities into a GP model based on terrain partitioning is presented by Cornford *et al.* [1999]. They deal with straight discontinuities (*fronts*) in wind fields. They place auxiliary GP models along both sides of the discontinuity. These are then used to derive a GP model that represents the complete wind field by conditioning on the values along the front. The discontinuity takes a parametric form. While its precise location is also learned, the authors note that the "parameters describing the location of the front need to be initialized close to the correct values" (p. 6). In their approach one has to specify a-priori the number of discontinuities as well as their approximate locations. A discontinuity is assumed to split the complete input space.

Tresp [2000] provides a Gaussian process variant of the mixture of experts model of Jacobs *et al.* [1991]. Based completely on the input, a gating network assigns probabilities to different GP expert models such that the model with the most appropriate local characteristics as specified by a bandwidth is selected. Rasmussen and Ghahramani [2002] extend these ideas and present an infinite mixture of experts model. Their model infers the number of components required to capture the data and learns the hyperparameters of its experts. In contrast to Tresp [2000], each GP expert predicts only on the basis of the training data it was assigned which improves runtime and avoids problems in boundary regions. Meeds and Osindero [2006] extend this model to a full generative model over input and output space. In a similar setting, Schwaighofer *et al.* [2005] propose a hierarchical Bayesian framework in a recommendation system scenario. There, hierarchical Bayesian modeling amounts to learning the mean and covariance function of a GP model common to all individual scenarios by means of an Expectation-Maximization-based algorithm. The resulting model is then used in the prediction of the GP models of the individual scenarios.

There are certain scenarios that approaches based on input-space segmentation cannot solve satisfactorily. Consider, for example, the situation depicted in Figure 1.4(a) where a coherent region splits up into different segments. There is no straight-forward way to partition this map and to assign segments to different GP models. Alternative approaches avoid this by employing non-stationary covariance functions. These allow to stick to only one GP model while being able to express different regression characteristics, e.g. smoothness, in different regions of the input space. Higdon *et al.* [1999] introduce a covariance formulation that achieves non-stationarity by assigning an individual kernel to each input location which determines its covariance to other input locations. The covariance structure of the whole GP is thus defined by these individual kernels, i.e., the individual kernel parameters specify the behavior and the smoothness of the GP within the respective local area. Higdon *et al.* apply their method to a small example in toxic waste remediation. Paciorek and Schervish [2004] extend this formulization to a class of non-stationary



(a) Map (b) Kernels

Figure 1.4: A function whose smoothness varies across input space can be modeled by means of a non-stationary covariance function based on individual kernels. The local kernels are visualized by means of ellipses with semi-axis lengths chosen proportionally to the corresponding eigenvalues of the kernel matrices.

covariance functions which provides a general framework to derive non-stationary variants of stationary covariance functions. They report promising results for two- and three-dimensional input spaces. They found that their model readily generalizes to non-Gaussian data. As they use GP priors to model the structure of individual kernels, their model, however, is computationally demanding and only feasible for data-sets with less than 1,000 data-points. In order to derive an efficient model that is applicable to large data-sets possibly in online applications, the most important challenge is thus how to represent and learn the local kernels of the input locations. Stephenson *et al.* [2005] develop a similar formulation of non-stationary covariance functions by spatially evolving the spectral density of a stationary GP model in the frequency domain.

The problem of adapting kernels to local structure has also been studied in the computer vision community. Takeda *et al.* [2006] perform non-parametric 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.

Terrain maps are often very large so that we need to find sparse representations. Guestrin *et al.* [2005] present an algorithm for sensor-placements for monitoring spatial phenomena by means of GP models. They derive a formulization based on mutual information as optimization criterion to choose the best location of sensors such that the chosen sensor placements are most informative about unsensed locations. Thereby the uncertainty of the posterior GP is decreased.

1.3 Contribution and Outline

This thesis presents a novel terrain modeling approach based on an extended Gaussian process formulation. Our model addresses the following requirements, which are particularly important in application domains like outdoor robotics:

1. Elevations need to be predicted at arbitrary locations.

2. Sensory data have to be smoothed to remove noise.

3. Discontinuities need to be preserved.
4. Uncertainty estimates for the predictions are required.
5. Varying data densities have to be dealt with.

While other approaches as discussed in the section on related work accommodate for several of these requirements, our aim in this thesis was to address them in one consistent framework. We build on the well-established Gaussian processes framework, which is a non-parametric Bayesian approach to the regression problem. To deal with the preservation of structural features like edges and corners, we employ non-stationary covariance functions as introduced by Paciorek and Schervish [2004]. Non-stationarity is achieved by introducing local regression kernels that model the local characteristics around an input location. Kernels need to be locally adapted to the underlying structure. Figure 1.4 illustrates the effects of local kernel adaptation. The left diagram of this figure depicts a simulated terrain surface which contains a sharp edge that should not be smoothed over by the model. The right diagram of the same figure depicts the local kernels after adaptation by our approach, visualized by ellipses with semi-axis lengths chosen proportionally to the corresponding eigenvalues of the kernel matrices. As can be seen from the diagram, the kernels adapted to the local structure of the surface, allowing to smooth along the edge as well as within the flat regions, but preventing to smooth perpendicular to the edge. The task of kernel adaptation is a formidable optimization problem as it exhibits a high dimensionality. In this work, we have studied two novel, alternative approaches for solving this optimization problem:

1. A *maximum-likelihood approach* based on a gradient ascent algorithm over the pseudo-likelihood of the observed data.
2. A *terrain gradient approach* that adapts the kernels iteratively according to the gradients in the local elevation structure.

The first approach is more principled from a theoretical point of view, but brings certain practical problems. The second approach works well in practice as we demonstrate on several hard simulated and real-world regression scenarios. Its idea is akin to the solutions to the adaptive image smoothing problem 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 are not designed for dealing with a varying density of data-points or with potential gaps to fill, they proved to be applicable to our kernel adaptation problem.

This thesis is structured as follows. In Chapter 2, we present the technique of Gaussian process regression and discuss its advantages. We illustrate the importance of the covariance function and highlight the idea of kernels. In Chapter 3, we introduce the non-stationary formulation of Gaussian process regression. We present the non-stationary covariance function used in our models and analyze its properties. We discuss the implications of these properties for the terrain modeling problem which has consequences for the adaptation of the local kernels. In Chapter 4, we present our first approach to learning the local kernels which is based on a gradient ascent optimization over the data-likelihood. In various experiments, we visualize the advantages and difficulties of this approach. In Chapter 5, we introduce the alternative approach to fit the kernels based on local terrain gradients. We discuss plenty of experiments on artificial and real data. Finally, we present our conclusions and ideas for future work in Chapter 6.

Chapter 2

Gaussian Process Regression

Regression is at the core of many problems in machine learning. Given a set of n observations $D = (\mathbf{x}_i, y_i)_{i=1}^n$ consisting of inputs $\mathbf{x_i} \in \mathbb{R}^D$ and of corresponding targets $y_i \in \mathbb{R}$, the goal is to recover a function f such that

$$y_i = f(\mathbf{x_i}) + \epsilon_i \quad \text{with} \quad \epsilon_i \sim \mathcal{N}(0, \sigma_n^2),\ \epsilon \text{ i.i.d. } \forall i\ . \tag{2.1}$$

The observed targets y_i are assumed to be affected by additive error terms ϵ_i which are independently and identically normally-distributed. Plenty of techniques for learning such regression functions have been proposed. Gaussian process regression has a long history in the field of geostatistics where the corresponding method is known as *kriging*. Nevertheless, Gaussian processes only recently became important in other areas like machine learning and robotics.

Gaussian processes can be seen as a generalization of locally weighted nearest neighbor regression or splines. A Gaussian process is a stochastic process that generates samples $(X_t)_{t \in T}$ for an arbitrary index set T such that any finite set of samples is normally distributed. Gaussian processes are completely defined by

- their *expected value* $T \to \mathbb{R},\ t \mapsto \mathbb{E}(X_t)$ and
- their *covariance function* $T \times T \to \mathbb{R},\ (t, t') \mapsto \text{cov}(X_t, X_{t'}) = \mathbb{E}((X_t - \mathbb{E}(X_t))(X_{t'} - \mathbb{E}(X_{t'})))$.

Thus, for all indices $t_1, \ldots, t_n \in T$, the multivariate distribution of $(X_{t_1}, \ldots, X_{t_n})$ is given by an n-dimensional normal distribution. Gaussian processes can be used to define a prior probability distribution over functions. Inference based on observations takes place directly in the space of functions. Within the setting of terrain modeling, the indices correspond to two-dimensional input locations $\mathbf{x}_i \in \mathbb{R}^2$ and the corresponding samples to terrain elevations $y_i \in \mathbb{R}$. A set of samples is thus a set of observations $D = (\mathbf{x_i}, y_i)_{i=1}^n$. The regression goal is to learn a function f as presented in Equation (2.1) which yields elevations for arbitrary input locations. Viewing any finite set of samples y_i as being jointly normally distributed, we get the predictive distribution for the observed targets

$$p(y_1, \ldots, y_n \mid \mathbf{x_1}, \ldots, \mathbf{x_n}) \sim \mathcal{N}(\boldsymbol{\mu}, K) \tag{2.2}$$

according to a mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and a covariance matrix K. μ is typically assumed $\mathbf{0}$. K is specified in terms of a covariance function k with a global noise σ_n as

$$K_{ij} = \text{cov}(y_i, y_j) = \text{cov}(f(\mathbf{x_i}), f(\mathbf{x_j})) =: k(\mathbf{x_i}, \mathbf{x_j}) + \sigma_n^2 \delta_{ij}\ , \tag{2.3}$$

where δ_{ij} is the Kronecker delta which is one if $i = j$ and zero otherwise. Note that the covariance function is defined by the input locations $\mathbf{x}_i$ and $\mathbf{x}_j$ and not the target values.

Gaussian process regression offers substantial advantages over other techniques:

- *Non-parametricity*: No specific parametric form of the target function f is assumed. This makes it possible to learn any kind of target function.
- *Bayesian framework*: It is possible to specify a-priori assumptions over the target functions. In a sound and well-founded mathematical framework, evidence from observations can then be combined with these priors to yield a posterior over target functions – instead of yielding one single function as a result.
- *Non-linear regression*: Complex non-linear dependencies can be modeled with straightforward linear algebra.
- *Predictive distributions*: GPs do not only provide predictions of the targets for arbitrary input locations, but also confidence estimates for these predictions which quantify the uncertainty stemming from the model and the noise in the observations.

In the remainder of this chapter, we will describe in Section 2.1 how prediction takes place in the GP framework and present the idea of learning in GPs in Section 2.2. We will discuss why the covariance function is the essential part of a GP model in Section 2.3 and introduce the idea of local kernels which are an important component in our terrain modeling approach in Section 2.4.

2.1 Prediction

Based on our set of observations $D = (\mathbf{x_i}, y_i)_{i=1}^n$, we want to predict the target f^* for an arbitrary input location $\mathbf{x}^*$. As GPs are a non-parametric learning technique, the model structure for prediction is not specified a-priori but is instead determined from the observations D. In contrast, a parametric technique would absorb the information of the training points into a model whose structure has been specified in advance (e.g. linear regression or a neural network). Viewing any finite set of samples as being jointly normally distributed, one can derive the $n+1$-dimensional joint Gaussian distribution for the observations and the test location $p(y_1, \ldots, y_n, f^* | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{x}^*)$ as

$$\begin{bmatrix} \mathbf{y} \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K + \sigma_n^2 I & \mathbf{k} \\ \mathbf{k}^T & k(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix}\right) \tag{2.4}$$

with $K \in \mathbb{R}^{n \times n}$, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, denoting the matrix containing the covariance values of the observations, $\mathbf{k} \in \mathbb{R}^n$, $k_j = k(\mathbf{x}^*, \mathbf{x}_j)$, the vector of the covariances of the test location, the training targets $\mathbf{y} \in \mathbb{R}^n$, and the identity matrix I. Conditioning this joint distribution on the observations yields the one-dimensional normal distribution for the test target defined by

$$f^* \sim \mathcal{N}(\mu^*, v^*) , \tag{2.5}$$

$$\mu^* = E(f^*) = \mathbf{k}^T \left(K + \sigma_n^2 I\right)^{-1} \mathbf{y} , \tag{2.6}$$

$$v^* = V(f^*) = k(\mathbf{x}^*, \mathbf{x}^*) + \sigma_n^2 - \mathbf{k}^T \left(K + \sigma_n^2 I\right)^{-1} \mathbf{k} . \tag{2.7}$$

Deriving the posterior means to restrict the joint prior distribution to contain only those functions which agree with the observations. The central ingredient in this formulation is the covariance function k as it specifies the influence that each training point $(\mathbf{x}_i, y_i)$ has in the prediction of the new target f^*. Thereby, k constrains the space of target functions and thus represents the prior knowledge about the target distribution. k is usually a parametric function. Together with the global noise parameter σ_n, the parameters of k are called the *hyperparameters* $\boldsymbol{\theta}$ of the GP model. The term hyperparameters emphasizes the fact that these are parameters of a non-parametric model making no assumptions about the target model – in contrast to the parameters of a parameteric model whose structure is determined a-priori.

2.2 Learning

Learning in the Gaussian process framework means finding the optimal hyperparameters $\boldsymbol{\theta}$ that determine the parametric covariance function and the noise of the process. A possible optimization criterion is the *marginal data-likelihood* $\mathcal{L}$ which takes the form

$$\mathcal{L}(\boldsymbol{\theta}) = p(\mathbf{y}|X, \boldsymbol{\theta}) = \int p(\mathbf{y}|f, X)p(f|X, \boldsymbol{\theta})df \quad . \tag{2.8}$$

$\mathcal{L}$ is called *marginal* as it integrates over all possible target functions f. Let $\mathbf{f}$ denote the vector of function values of target function f for the training inputs X. In the GP model, we have a Gaussian prior $\mathbf{f}|X \sim \mathcal{N}(\mathbf{0}, K)$ and the likelihood is a factorized Gaussian $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2\, I)$. Thus, one can derive [Rasmussen and Williams, 2006]

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2\, I)^{-1}\mathbf{y} - \frac{1}{2}\log|(K + \sigma_n^2\, I)| - \frac{n}{2}\log 2\pi \quad . \tag{2.9}$$

The three involved terms have natural interpretations. $-\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2\, I)^{-1}\mathbf{y}$ is the only term containing the targets $\mathbf{y}$ and measures the data-fit of the observations. $-\frac{1}{2}\log|(K + \sigma_n^2\, I)|$ is a complexity penalty. $-\frac{n}{2}\log 2\pi$ is a normalizing constant and independent of the hyperparameters. Thus, the data-likelihood trades off data-fit and complexity penalty. The complexity penalty prevents overfitting when using the data-likelihood as optimization criterion. It holds $|K + \sigma_n^2\, I| \geq 0$ as $K + \sigma_n^2\, I$ is a positive-definite matrix. This determinant is large when the values on the main diagonal of the matrix dominate the other values. This is for example the case when the covariance function yields large self-covariances (covariances with respect to the points themselves), i.e., $k(x_i, x_i) \gg k(x_i, x_j)\ \forall i \neq j$. In this case, the GP model overfits the observations.

The optimal hyperparameters $\boldsymbol{\theta}$ can be found by gradient ascent methods that fix the parameters by optimizing the marginal data-likelihood $\mathcal{L}$ of the observed training data-set. Alternatively, the parameters can be integrated over using prior distributions, which results in a fully Bayesian model. This is usually computationally more demanding which might necessitate using Markov-Chain Monte Carlo sampling to approximate intractable integrals (depending on the type of covariance function).

2.3 Covariance Functions

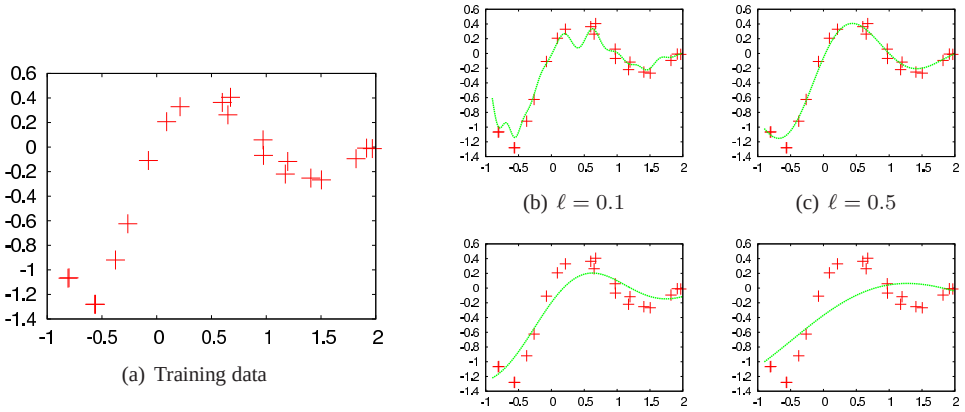
An important component of GP models are the covariance functions. As described above, GPs use the training points directly for predicting a target for a new input location. Visually speaking, the covariance function k determines the influence that each training point has. Within the GP framework, the assumption is made that this dependency is based solely on the inputs and independent of the targets,

$$\text{cov}(y_i, y_j) = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma^2\delta_{ij} \quad . \tag{2.10}$$

The covariance defines the dependencies between targets. Therefore, it represents our prior knowledge about the target distribution. For instance, it incorporates assumptions about the smoothness of the target function. Learning a covariance function means finding a measure for target similarity based on the input locations. The goal is to assign high covariances to points exhibiting the same terrain structure, while giving small covariances to points of different terrain structures. Learning the covariance function corresponds to finding the optimal parameter values for the parametric function k such that this requirement is fulfilled.

2.3.1 Theoretical Background

This subsection describes how the choice of covariance function constitutes the prior within the GP framework. The predictive distribution over target values for an input location $\mathbf{x}$,



(a) Training data

(b) $\ell = 0.1$

(c) $\ell = 0.5$

(d) $\ell = 1.0$

(e) $\ell = 2.0$

Figure 2.1: The lengthscale parameter ℓ of k_{SE} determines the smoothness of the learned function. Given the training data in (a), different functions (b)-(e) are learned depending on ℓ.

$$p(y|\mathbf{x}) = \int p(y|\mathbf{x}, f)p(f)\ df\ , \tag{2.11}$$

necessitates a prior $p(f)$ over target functions. The question is how to specify the space of hypothesis functions constituting the prior. This prior shall impose constraints on the set of admissible target functions, e.g. smoothness assumptions. Within the GP setting, this hypothesis space can be specified by means of the covariance functions as the theory of *Reproducing Kernel Hilbert Spaces* (RKHS) shows. Let H be a Hilbert space of real functions defined on an index set X. The norm is induced by the inner product: $\parallel f \parallel = \sqrt{< f, f >}$. H is called an RKHS with an inner product $< \cdot, \cdot >_H$ if there exists a function $k : X \times X \mapsto \mathbb{R}$ with the following properties [Rasmussen and Williams, 2006]:

- For every $\mathbf{x}$, $k(\mathbf{x}, \mathbf{x}')$ as a function of $\mathbf{x}'$ belongs to H.
- k has the reproducing property $< f(\cdot), k(\cdot, \mathbf{x}) >_H = f(\mathbf{x})$.

The norm $\parallel \cdot \parallel$ encodes complexity assumptions which determine the smoothness of the target functions. There is a direct relation between kernel functions and RKHSs as the Moore-Aronszjan theorem [Aronszajn, 1950] states: For every positive-definite function $k(\cdot, \cdot)$ on $X \times X$, there exists a unique RKHS, and vice versa. A kernel function k is positive-definite if

$$\sum_{i,j=1}^{n} c_i c_j k(\mathbf{x_i}, \mathbf{x_j}) \geq 0, \quad \forall\, n \in \mathbb{N},\ \forall\, c_i, c_j \in \mathbb{R}\ . \tag{2.12}$$

The covariance functions used in GPs are positive-definite and can be understood as kernel functions. Thus, they implicitely define the hypothesis space and therefore form the prior within the GP framework. This shows that a careful choice and adaptation of the covariance function is crucial.

2.3.2 Squared Exponential

A common and convenient choice for the covariance function is the squared exponential,

$$k_{SE}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T \, \Sigma \, (\mathbf{x}_i - \mathbf{x}_j))\right) \, , \tag{2.13}$$

where $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$, the symmetric $d \times d$ matrix Σ encodes the dependencies among the d dimensions and σ_f denotes the amplitude (or signal variance). k_{SE} is a stationary covariance function as it depends only on the difference $|\mathbf{x}_i - \mathbf{x}_j|$. Σ might, for instance, be a diagonal matrix $\Sigma = diag(\boldsymbol{\ell})^{-2}$ where $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_d)$ contains the characteristic length-scales of the individual dimensions,

$$k_{SEdiag}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{k=1}^{d} \frac{(\mathbf{x}_{i,k} - \mathbf{x}_{j,k})^2}{\ell_k^2}\right) \, . \tag{2.14}$$

The parameters of the squared exponential have an intuitive interpretation. The length-scales ℓ_k in Equation (2.14) determine how far one needs to move along a particular axis k in input space for the function values to become uncorrelated. Large values ℓ_k will make the function become almost independent of that input since it assigns similar covariance values along the complete axis. In contrast, small ℓ_k leads to overfitting, as close input locations receive much larger weight than more distant locations. Figure 2.1 illustrates an example of the influence of the lengthscale in a one-dimensional input space. A diagonal Σ makes it possible to describe the influence structure along the individual dimensions of the input space. If one uses a non-diagonal matrix Σ instead, it is possible to rotate the axes and thus describe the covariance structure along oriented dimensions. This yields a more general understanding of Equation 2.13: Σ corresponds to a kernel matrix which is the covariance matrix of a Gaussian kernel. This is further exposed in the next section.

2.4 Kernels

The interesting parameter of k_{SE} is the kernel Σ specifying the covariance structure of an input location. Σ corresponds to the covariance matrix of a normal distribution with mean $\mathbf{0}$. The difference vector $\mathbf{x}_i - \mathbf{x}_j$ is weighted according to this distribution which yields the covariance of the two input locations. Learning a GP model amounts to fitting the covariance function to the observations. In the case of k_{SE}, we need to find a way to adapt the kernel Σ according to some optimization criterion. In the following, we will focus on 2×2 real-valued matrices as used in the terrain modeling problem where we are concerned with two-dimensional input data.

Fitting kernels is based on an adequate parameterization of kernels. Any symmetric, positive semi-definite matrix is a valid kernel as the kernel corresponds to the covariance matrix of a multivariate normal distribution. The spectral theorem says that for a matrix A that is normal[1] there exists a unitarian[2] matrix R such that $A = R \, \Lambda \, R^T$ where Λ is the diagonal matrix the entries of which are the eigenvalues of A. The column vectors of R are the eigenvectors of A and thus are orthonormal. As the kernel Σ of k_{SE} is symmetric it is also normal. Thus, one can represent the kernel as a combination of a rotational matrix R and a diagonal eigenvalue matrix Λ. This gives the parameterization of kernels

$$\Sigma = R \begin{pmatrix} \ell_1^2 & 0 \\ 0 & \ell_2^2 \end{pmatrix} R^{-1} \, . \tag{2.15}$$

where ℓ_i are the square-roots of the eigenvalues of Σ.

[1] A real-valued matrix A is normal if it satisfies $AA^T = A^T A$.
[2] A real-valued matrix A is unitarian if it satisifies $A^T A = I$ where I is the identity matrix.

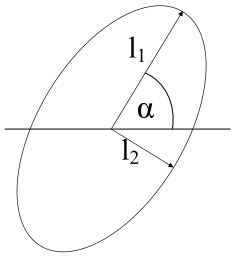


Figure 2.2: In case of a two-dimensional input space, the kernel Σ with lengthscales ℓ_i and orientation α used in k_{SE} can be visualized as ellipse.

There is a one-to-one correspondence between kernel matrices und ellipses which can be exploited for visualization. Kernels are specified by their eigenvalues and their eigenvectors. The eigenvalues determine the lengths along the semi-axes of the ellipse while the eigenvectors specify the orientation. The square-roots of the eigenvalues are the standard deviations of the corresponding normal distribution. This is visualized in Figure 2.2. The ellipse with semi-axis lengths of a standard deviation of one covers about 39.35% of randomly drawn values of the corresponding normal distribution. The ellipse drawn with two standard deviations covers about 86.47% randomly drawn values. The standard deviation along a semi-axis corresponds to the lengthscale hyperparameter ℓ_i in this direction as used in Equation (2.14).

The orientation matrix R and thus the eigenvectors of Σ can be specified by an angle α as

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} . \tag{2.16}$$

This parameterization has the drawback that α has the range $[0, 2\pi)$. Thus, it is cylindrical which is not compatible with the range of a GP. This is particularly a problem in the case of gradient ascent techniques where one assumes that the influence of a single parameter with respect to the resulting kernel is monotonous. To avoid this problem, one can overparameterize the orientation matrix by two parameters u and v as

$$R = \begin{pmatrix} \frac{u}{l_{uv}} & \frac{-v}{l_{uv}} \\ \frac{v}{l_{uv}} & \frac{u}{l_{uv}} \end{pmatrix} . \tag{2.17}$$

where $l_{uv} = \sqrt{u^2 + v^2}$. The range of u and v is $\mathbb{R}$. This leads to entries in R in the range $[-1, 1]$.

To sum up, the kernel Σ which parameterizes k_{SE} is specified by parameter sets $\boldsymbol{\theta} = \{\ell_1, \ell_2, \alpha\}$ or $\boldsymbol{\theta} = \{\ell_1, \ell_2, u, v\}$. In order to learn a suitable covariance function, these parameter sets need to be fitted to the training set D.

Chapter 3

Non-Stationary Gaussian Process Regression

Most existing applications of Gaussian process regression use stationary covariance functions which assume the same covariance structure over the whole input space. In the case of terrain modeling, however, it is one of our most important requirements to take local structure into account. This is necessary to derive a regression function that achieves de-noising in homogenous areas while preserving discontinuities in regions of large structural change. Therefore, we need a model that is able to adapt the covariance structure to local terrain properties. This can be achieved by means of non-stationary covariance functions as described in this chapter. Section 3.1 formalizes the idea of non-stationarity. Section 3.2 presents the non-stationary variant of the squared exponential covariance function which we apply in our terrain models. In Section 3.3, we thoroughly analyze this function to get a deep understanding of the resulting covariances. This is needed to derive adequate terrain models. Section 3.4 describes how prediction takes place in the non-stationary setting. In Section 3.5, we highlight the implications that the chosen covariance function and different ways of prediction have on the terrain modeling problem. Finally, Section 3.6 introduces different learning approaches in the non-stationary framework which are then described in detail in the following chapters.

3.1 Non-Stationarity

Stationary covariance functions k depend only on the difference $\mathbf{d}_{ij} = |\mathbf{x}_i - \mathbf{x}_j|$ of their input values, i.e., $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{d}_{ij})$. By ignoring the absolute values of the inputs, they fail to adapt to a varying smoothness in the target function. Figure 3.1, for instance, depicts a one-dimensional target function whose characteristics change significantly at $x = 0.2$. It is impossible to specify a global lengthscale parameter to fit this function: the left part of the function demands for a large lengthscale while the right part is wiggly which necessitates a small lengthscale. Non-stationary covariance functions take the absolute values or local properties of the input locations into account. This makes it possible to adapt to functions whose smoothness varies with the inputs. For example, depending on the input location we might choose a different lengthscale in the example of Figure 3.1. In the setting of terrain modeling, non-stationary covariance functions enable us to capture local terrain properties. Therefore, we apply this type of covariance function in our framework.

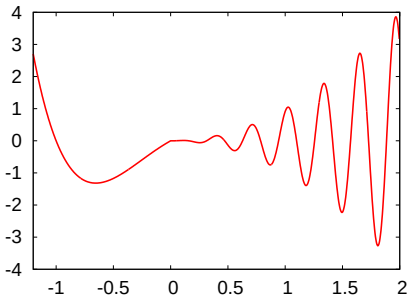


Figure 3.1: Function whose smoothness varies across input space. This necessitates a non-stationary covariance function.

3.2 Non-Stationary Squared Exponential

Paciorek and Schervish [2004] introduced the following family of non-stationary covariance functions. Let Q_{ij} be a quadratic form for the inputs $\mathbf{x}_i$ and $\mathbf{x}_j$ defined by

$$Q_{ij} = (\mathbf{x}_i - \mathbf{x}_j)^T \left(\frac{(\Sigma_i + \Sigma_j)}{2}\right)^{-1} (\mathbf{x}_i - \mathbf{x}_j) \; . \tag{3.1}$$

Then, one can derive a family of non-stationary covariance functions using the following theorem.

Theorem 3.2.1 *Let Q_{ij} be defined as in Equation (3.1). If a stationary correlation function, $R^S(\tau)$, is positive-definite on $\mathbb{R}^D$ for every $D = 1, 2, \ldots$, then*

$$R^{NS}(\mathbf{x}_i, \mathbf{x}_j) = |\Sigma_i|^{\frac{1}{4}} |\Sigma_j|^{\frac{1}{4}} |(\Sigma_i + \Sigma_i)/2|^{-\frac{1}{2}} R^S(\sqrt{Q_{ij}}) \tag{3.2}$$

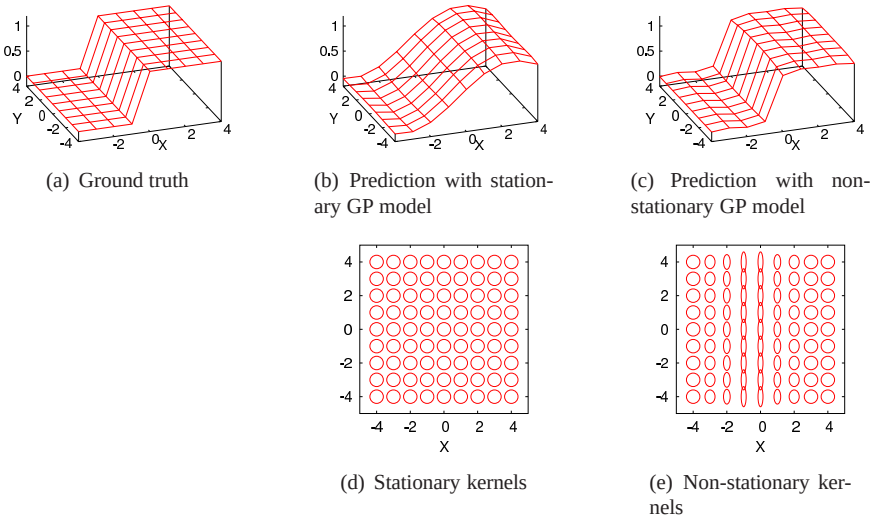
is a non-stationary covariance function, positive-definite on $\mathbb{R}^D$ for every $D = 1, 2, \ldots$.

The central idea of this family of covariance functions is the usage of kernels. Each input location $\mathbf{x}_i$ is assigned its individual kernel Σ_i. This kernel Σ_i captures the local properties of the target function at input location $\mathbf{x}_i$. In areas of wiggly behavior of the target function, for instance, the kernel might incorporate small lengthscales while in smooth areas it might use large lengthscales. By using different kernels at different input locations, it becomes possible to account for varying local function properties. Stephenson *et al.* [2005] derive a similar formulation of non-stationary covariance functions by spatially evolving the spectral density of a stationary GP model in the frequency domain.

If using the squared exponential covariance function k_{SE} for $R^S(\tau)$, one gets the non-stationary squared exponential covariance function,

$$\begin{aligned} k_{NSE}(\mathbf{x}_i, \mathbf{x}_j) = \quad & \sigma_f^2 \quad |\Sigma_i|^{\frac{1}{4}} |\Sigma_j|^{\frac{1}{4}} \left|\tfrac{\Sigma_i + \Sigma_j}{2}\right|^{-\frac{1}{2}} \\ & \exp\left[-(\mathbf{x}_i - \mathbf{x}_j)^T \left(\tfrac{\Sigma_i + \Sigma_j}{2}\right)^{-1} (\mathbf{x}_i - \mathbf{x}_j)\right] \; . \end{aligned} \tag{3.3}$$

Note that this is the generalized form of the stationary squared exponential given in Equation (2.13). If one places the same kernel at all input locations, i.e., $\Sigma_i = \Sigma_j \; \forall i, j$, one gets the stationary squared exponential. Figure (3.2) illustrates the idea of using local kernels. It depicts a small artificial terrain containing a simple step. In the stationary case, we place the same kernels at all input locations. This



(a) Ground truth

(b) Prediction with stationary GP model

(c) Prediction with non-stationary GP model

(d) Stationary kernels

(e) Non-stationary kernels

Figure 3.2: The stationary squared exponential k_{SE} uses the same kernel for all input locations which makes it impossible to adapt to local structures. In this example, this leads to oversmoothing of the step. The non-stationary squared exponential k_{NSE} places different kernels at individual input locations considering the local terrain properties. This makes it possible to preserve the edge while still using large kernels in the flat areas.

has the effect of oversmoothing the step. In the non-stationary case, we use different kernels to represent the discontinuity. These kernels have a large lengthscale in the direction along the step, but a very small one in the direction perpendicular to the step. This makes it possible to preserve the discontinuity. Figure 3.3 illustrates the covariance structure when kernels are used to represent a discontinuity, in this case a diagonal step. The covariance function is supposed to assign high covariances among the points of the same terrain level and small covariances for points that belong to different levels. This is achieved by using the long and thin kernels oriented along the step as seen in the previous example. The right side of the figure depicts the resulting covariances with respect to input location $(2, 0)$. Covariances are high in the area right along the diagonal step which contains points of the upper step-level of which also $(2, 0)$ is part. In contrast, covariances are small with respect to all points of the lower level.

3.3 Analyzing the Non-Stationary Model

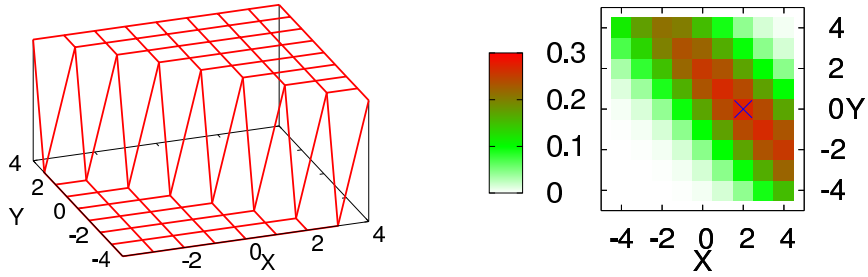
Besides the signal variance σ_f^2, the non-stationary covariance function k_{NSE} introduced in the last section consists of two parts, the prefactor p and the exponential part e:

$$k_{NSE}(x_i, x_j) = \sigma_f^2 \cdot p \cdot e \tag{3.4}$$

$$p = |\Sigma_i|^{\frac{1}{4}} |\Sigma_j|^{\frac{1}{4}} \left| \frac{\Sigma_i + \Sigma_j}{2} \right|^{-\frac{1}{2}} \tag{3.5}$$

$$e = \exp\left[-(x_i - x_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2} \right)^{-1} (x_i - x_j) \right] \tag{3.6}$$

The exponential part e measures the Mahalanobis distance between the two input locations, i.e., the distance between the two locations is weigthed according to a kernel which is the average of the individual



(a) A terrain map containing a discontinuity (a step)

(b) Suitable covariance structure for the location $(2, 0)$ (blue cross) for the problem depicted in (a).

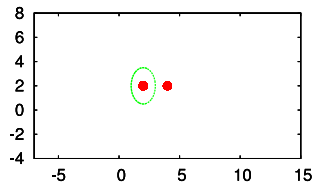
Figure 3.3: The covariance function is supposed to assign high covariances among data-points of the same terrain region.

kernels at both input locations. This kernel gives different weight to the individual dimensions of the difference vector. Thus, the resulting value of e is influenced by *(i)* the Euclidean distance of the two locations, and *(ii)* the averaged kernel. This corresponds to the intuition that the more distant two input locations are, the smaller the resulting value becomes. On the other hand, the larger the averaged kernel is, the less important is the Euclidean distance between both input locations. Averaging has the effect that it does not matter with respect to the exponential part e whether we combine two medium-sized kernels or a large with a small kernel. Also, as expected, two large kernels produce significantly higher values than the combination of a small and a large kernel.

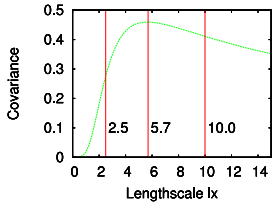
In contrast to the easy to interpret factor e, the prefactor p is harder to understand. In fact, under certain conditions it leads to unexpected effects that are well worth analyzing in detail. This was already noted by Paciorek as a sidemark in his PhD thesis [Paciorek, 2003] noting "this effect seems to be restricted to situations in which the kernel sizes change very quickly, so it may not be material in practice" (p. 28). We, however, experienced that the prefactor has a strong effect on the resulting terrain model. Thus, it is worthwhile to get a deeper understanding of this prefactor. Intuitively, the prefactor is needed to make the covariance function positive-definite. It basically introduces a penalty if the shape of the two involved kernels significantly differs. In the case of equal shape of both kernels, it attains its largest value, $p = 1$. Otherwise, the prefactor decreases towards 0 the more different the two kernels are. The effects of p can be demonstrated best in the one-dimensional case where the kernels Σ are one-dimensional "matrices". Thus, their determinant corresponds to their only entry. Assuming two kernel matrices $A = (a)$ and $B = (b)$, the prefactor takes the form

$$p = (a)^{\frac{1}{4}}(b)^{\frac{1}{4}}(\frac{a+b}{2})^{-\frac{1}{2}} = \left(\frac{\sqrt{ab}}{\frac{1}{2}(a+b)}\right)^{\frac{1}{2}} . \tag{3.7}$$

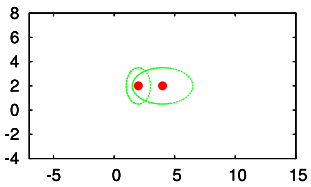
This corresponds to the square-root of the fraction of the geometric mean and the arithmetic mean which is always less than or equal 1. Thus, independent of their size, if a and b are equal this will yield a value of 1. In contrast, the larger the difference between a and b is the smaller is the resulting prefactor. This may lead to the following unexpected effect. Assume we are given two pairs of kernels, $A_1 = (a_1)$ and $B_1 = (b_1)$ as well as $A_2 = (a_2)$ and $B_2 = (b_1)$. The kernels of the first pair have equal size, i.e., $a_1 = b_1$. The kernels of the second pair differ in size, i.e., $a_2 \neq b_2$, but both are significantly larger than the corresponding kernels of the first pair, i.e., $a_2 \gg a_1$ and $b_2 \gg b_1$. In this case, the prefactor will be larger for the first pair (namely 1) than for the second pair where it is smaller than 1. This may lead to the effect that large kernels yield smaller covariance values than small kernels – even if the distance between the respective kernels



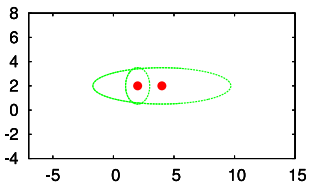
(a) Two kernels. The left one is fixed while we search the optimal right one.



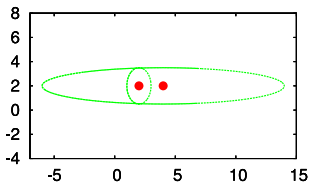
(b) Covariance as function of ℓ_x^{right} of the right kernel



(c) $\ell_x^{right} = 2.5$



(d) $\ell_x^{right} = 5.7$



(e) $\ell_x^{right} = 10.0$

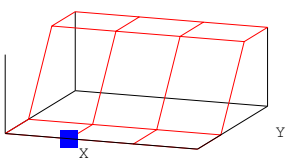
ℓ_x^{right}	Covariance	Prefactor	Exponential
2.5	0.275	0.830	0.332
5.7	0.459	0.583	0.788
10.0	0.411	0.445	0.924

(f) Covariance calculation

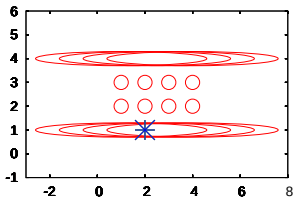
Figure 3.4: The covariance of k_{NSE} is not monotonous in the kernel parameters as exemplified for two kernels. The lengthscales of the left kernel ($\ell_x^{left} = 1.0$, $\ell_y^{left} = 1.5$) and the lengthscale along the y-axis of the right kernel ($\ell_y^{right} = 1.5$) are fixed, while we test different values for ℓ_x^{right}.

is the same. The prefactor cannot simply be omitted since it is needed to achieve positive-definiteness of the covariance matrix. Therefore, one needs to be aware of its behavior when fitting a GP model with the covariance function k_{NSE}. In the following, some exemplary two-dimensional situations are presented which deepen the understanding of the covariance function and thus of our non-stationary model.

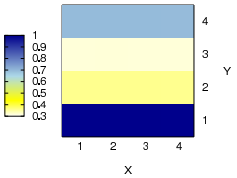
Dependency of the Covariance on the Kernel Size Figure 3.4 illustrates the effects of the different terms p and e of k_{NSE} for a pair of two kernels. While the first kernel is fixed, the lengthscale ℓ_x of the second kernel along the x-axis is scaled in x-direction. Figure 3.4(b) depicts the covariance as a function of this lengthscale. The corresponding curve is not monotonous. It has its maximum at $\ell_x = 5.7$ while it yields smaller values for smaller and larger lengthscales. Figures 3.4(c) - 3.4(e) visualize exemplary kernels for the optimal lengthscale and for a smaller and a larger lengthscale. Note that a larger kernel leads to a smaller covariance value. As the table in Figure 3.4(f) shows, this is due to the prefactor of k_{NSE} as already noted above. The non-monotonicity of the covariance function with respect to a single kernel parameter shows that the learning task for a complete data-set where each observation is assigned its individual kernel may be prone to local extrema – for the covariance function is an essential component in determining the model structure and thus directly influences the data-likelihood. Without additional constraints on the kernels, it is extremely difficult to find a global optimum.



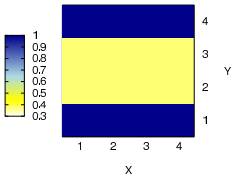
(a) Terrain (blue square at input location (2,1))



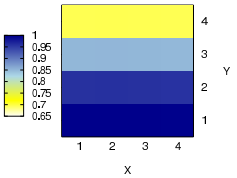
(b) Kernels



(c) Covariances w.r.t. input location (2,1)



(d) Prefactors w.r.t. input location (2,1)



(e) Exponential parts w.r.t. input location (2,1)

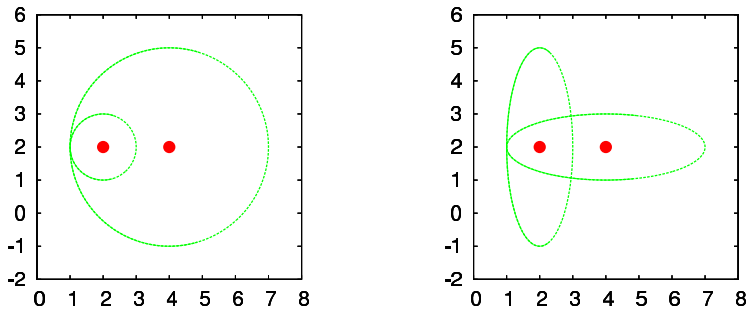
Figure 3.5: Counter-intuitive effect of k_{NSE} due to the prefactor: the covariances of the input location at $x = 2$, $y = 1$ are larger with respect to some locations across the step than to some neighboring points.

Non-monotonous Covariance across a Step Figure 3.5 illustrates the same effect by means of a small simulated terrain example. This elevation map consists of four rows with four data-points each. We set the outer two rows to have long thin kernels oriented along the row. This kernel type is typically used at discontinuities, i.e., when the influence shall only be along the x-axis. The middle two rows employ small kernels which are used in the case of rapidly changing environments. This kernel constellation could well be used at a discontinuity such as the one illustrated in Figure 3.5(a): the kernels close to the discontinuity are small while the outer kernels try to avoid smoothing across the discontinuity. The bottom three diagrams of Figure 3.5 visualize the covariances with respect to the input location (2,1), i.e., $x = 2, y = 1$. Formally, the diagrams illustrate the values for

$$k_{NSE}(\begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix}) \, . \tag{3.8}$$

As expected, the covariances are largest along the same row. Counter to intuition, however, they are also comparably large for the outer row ($y = 4$) which might be part of the other side of the discontinuity. In contrast, the covariances with respect to the input locations of the inner two rows are small. This effect is due to the prefactor p which is one for the two outer rows where the same kernel type is employed. p is significantly smaller for the two inner rows which have a completely different kernel type. In contrast, the exponential parts e follow intuition and yield smaller values the farther a row is away. However, the prefactor dominates the overall covariance values. This effect might lead to oversmoothing of the discontinuity as the covariance across the step is large. The solution to avoid such effects is the usage of smoothly varying kernels. If the kernels of the two inner rows were more similar to those of the outer rows, the covariance structure would be changed such that the upper row would have least covariance to the input location $(2, 1)$ as needed for preserving the discontinuity.

Scaled vs. Orthogonal Kernels An analytic example of how different kernel combinations can lead to the same covariance structure is presented in the following. First, consider the two scaled kernels illustrated



(a) Scaled kernels (b) Orthogonal kernels

Figure 3.6: Scaled and orthogonal kernels may produce the same values for the covariance function.

in Figure 3.6(a),

$$A_1 = \begin{pmatrix} x & 0 \\ 0 & x \end{pmatrix} , \quad A_2 = \begin{pmatrix} \lambda\, x & 0 \\ 0 & \lambda\, x \end{pmatrix} . \tag{3.9}$$

The resulting prefactor is

$$\left|\begin{pmatrix} x & 0 \\ 0 & x \end{pmatrix}\right|^{\frac{1}{4}} \left|\begin{pmatrix} \lambda\, x & 0 \\ 0 & \lambda\, x \end{pmatrix}\right|^{\frac{1}{4}} \left|\begin{pmatrix} \frac{x+\lambda\, x}{2} & 0 \\ 0 & \frac{x+\lambda\, x}{2} \end{pmatrix}\right|^{-\frac{1}{2}} \tag{3.10}$$

$$= (x^2)^{\frac{1}{4}} (\lambda^2 x^2)^{\frac{1}{4}} (\frac{(x+\lambda x)^2}{4})^{-\frac{1}{2}} \tag{3.11}$$

$$= 2\,\frac{x^{\frac{1}{2}} \lambda^{\frac{1}{2}} x^{\frac{1}{2}}}{(x+\lambda x)} = 2\,\frac{\lambda^{\frac{1}{2}}}{(1+\lambda)} . \tag{3.12}$$

Second, consider the two orthogonal kernels visualized in Figure 3.6(b),

$$B_1 = \begin{pmatrix} x & 0 \\ 0 & \lambda\, x \end{pmatrix} , \quad B_2 = \begin{pmatrix} \lambda\, x & 0 \\ 0 & x \end{pmatrix} . \tag{3.13}$$

The resulting prefactor is

$$\left|\begin{pmatrix} x & 0 \\ 0 & \lambda\, x \end{pmatrix}\right|^{\frac{1}{4}} \left|\begin{pmatrix} \lambda\, x & 0 \\ 0 & x \end{pmatrix}\right|^{\frac{1}{4}} \left|\begin{pmatrix} \frac{x+\lambda x}{2} & 0 \\ 0 & \frac{x+\lambda x}{2} \end{pmatrix}\right|^{-\frac{1}{2}}$$

$$= (x^2\lambda)^{\frac{1}{4}} (x^2\lambda)^{\frac{1}{4}} (\frac{(x+\lambda x)^2}{4})^{-\frac{1}{2}} = 2\frac{\lambda^{\frac{1}{2}}}{(1+\lambda)} .$$

Thus, the prefactors are the same in both kernel constellations. Also the exponential part is the same for arbitrary x and λ as the averaged kernels are

$$\frac{1}{2}(A_1 + A_2) = \frac{1}{2}\begin{pmatrix} (1+\lambda)\, x & 0 \\ 0 & (1+\lambda)\, x \end{pmatrix} = \frac{1}{2}(B_1 + B_2) .$$

This example points out the difficulty to find an optimal kernel constellation. In the case of a large observation set this problem becomes even more severe as different kernel combinations may lead to similar data-fits. Therefore, additional constraints are needed to make the optimal solution unique and suitable for generalization.

3.4 Prediction for New Input Locations

Our terrain modeling approach builds on the concept of local kernels Σ. These are used in the calculation of the non-stationary covariance function k_{NSE}. In the adaptation procedure of our model, we learn a local kernel Σ_i for each input location $\mathbf{x}_i$ of the training set D. However, we face a problem when predicting a new input location $\mathbf{x}^*$: there is no kernel Σ^* which is needed to calculate the covariances $k_{NSE}(\mathbf{x}^*, \mathbf{x}_i)$ of $\mathbf{x}^*$ with respect to the training locations $\mathbf{x}_i$. There are different possibilities to cope with this problem.

First, one might renounce on calculating an averaged kernel and use only the kernel Σ_i of the training location instead. This has the disadvantage that the penalty introduced by the prefactor in the covariance function gets lost (cf. section 3.3). This may lead to serious side-effects as this penalty has been effective in calculating the covariance matrix K for the training locations which is also used for prediction.

A *second* possibility is to determine the best stationary kernel for the training set and use this kernel for input locations that lack a kernel. This has the disadvantage that this counteracts the idea of using non-stationary kernels. In particular, this will lead to very low covariance values in terrain areas where the training locations have kernels with shapes strongly adapted to the local properties. The difference in shape with respect to the stationary kernel will decrease the prefactor p. In turn, the predicted target y^* will tend to the mean of the GP as all covariances are rather low.

The *third* and probably best solution appears the usage of a second level learning process over kernel parameters. Based on the kernels of the training locations, this top-level process predicts Σ^* which is used for the prediction of y^*. This has the advantage that it constrains Σ^* to be similar to the training kernels of the local neighborhood. A natural choice for this top-level model is a Gaussian process. A top-level process over kernels, however, introduces the burden of a second optimization problem. Also, a fully Bayesian treatment has to account for the dependencies among the two levels to optimize the overall model. A hierarchical approach that employs two GP levels [Paciorek and Schervish, 2004] is described in Section 3.6. This approach, however, is only feasible for small data-sets and thus not applicable in the domain of terrain modeling. As an approximation, one might therefore simply use a weighted average over the local neighborhood to estimate Σ^*. This mimics the use of an isotropic, stationary GP model as top-level process.

3.5 Implications for Terrain Modeling

When fitting a non-stationary GP model to a regression problem such as a terrain data-set, care has to be taken to avoid overfitting as in any learning problem. As described above, overfitting takes place if the covariance function assigns very high covariance values to points in the close neighborhood and small values to points further apart. In the case of using k_{NSE}, overfitting may be caused by both of its two subfunctions, the prefactor p and the exponential part e: *(i)* Very small kernels lead to small covariance values with respect to almost all points due to e. *(ii)* Kernels with different shapes, no matter which orientation and size, lead to small covariance values as the different shapes decrease p. Even if two kernels are large, but of different shape, this might yield a small covariance due to p. Therefore, overfitting is also possible with large kernels. To avoid overfitting, one thus needs to ensure two requirements:

1. *Kernels should not become too small.* This avoids overfitting by means of the exponential part.
2. *Kernels should vary smoothly across input space.* This avoids overfitting by means of the prefactor.

The second requirement also makes sense from an intuitive point of view. The kernel structure never needs to change rapidly across input space – in contrast to the terrain itself which may exhibit sudden changes. Note that also at a discontinuity the kernels vary smoothly as illustrated in Figure 1.4. On both sides of the discontinuity the same specialized kernel types (thin kernels elongated along the discontinuity) can be applied.

3.6 Adapting Kernels

Learning a GP model means to optimize the hyperparameters of the process according to some optimization criterion such as the marginal data-likelihood given in Equation (2.9). Besides the global noise rate σ_n and the signal variance σ_f, the parameters of the local kernels used by k_{NSE} form our set of hyperparameters. Optimizing these kernel parameters corresponds to adapting the kernels to the local terrain structure. Depending on the parameterization, we have three or four parameters per kernel (cf. Section 2.4). Given n observations, we get $3n + 2$ or $4n + 2$ hyperparameters in total. Obviously, this is a hard high-dimensional optimization problem vulnerable to local optima. Depending on the optimization criterion, the optimization problem is *ill-posed*, i.e., there exists no unique solution. Different kernel combinations may lead to similar data-fits due to the nature of k_{NSE}. In order to make this problem well-posed, we need to introduce additional constraints on the parameter space. As discussed in Section 3.5, one such criterion is to constrain the kernels to vary smoothly across input space. There are different ways to incorporate this requirement. Paciorek and Schervish [2004] who introduced the family of non-stationary covariance functions used in this thesis employ a hierarchical model for this purpose. They introduce a top-level GP model on the kernels as follows. They define a multivariate process for the matrix-valued function $\Sigma(\cdot)$. To model this process, they use an independent univariate process for each kernel parameter. To make the kernels vary smoothly across input space, all single processes are given a GP prior with a common stationary covariance function. Straightforward optimization, e.g., by means of gradient ascent, however, is precluded by the additional GP models for the kernel parameters. Therefore, Paciorek *et al.* employ Markov-Chain Monte-Carlo (MCMC) sampling. Unfortunately, due to the large number of parameters this results in slow computation, "limiting the feasibility of the model to approximately $n < 1000$" (p. 7). In terrain modeling we are dealing with large numbers of training observations. Thus, this hierarchical model is not applicable to our problem setting and we need to explore alternatives for adapting the kernels of the covariance function. This thesis studies two possibilities: a gradient ascent approach over the pseudo data-likelihood introduced in Chapter 4 and an approach based on local terrain gradients presented in Chapter 5. By means of both procedures, it is possible to fulfill the requirements that the kernels vary smoothly across input space and do not become too small.

Chapter 4

Gradient Ascent

Our first approach to learn the optimal local kernels Σ_i of the non-stationary covariance function k_{NSE} is to maximize the marginal data-likelihood restated here for convenience,

$$\mathcal{L} = \log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi \tag{4.1}$$

where we use the shorthand $K_y = K + I\,\sigma_n^2$. A popular optimization technique is gradient ascent. Due to the interdependencies among the many kernel parameters involved in k_{NSE}, however, the gradient of this formula cannot be derived analytically. As an approximation, we follow the common practice to optimize the pseudo-likelihood [Besag, 1977]. We do not view the kernel parameters as random variables in our optimization process, but assume them all as fixed except the one we are optimizing. This is described in Section 4.1 which enables us to find analytic forms for the gradients. In contrast to hybrid approaches that use gradient ascent to derive a proposal distribution for MCMC [Paciorek, 2003], the approach proposed in this chapter is fully based on gradient ascent. Section 4.2 describes our learning procedure. In Section 4.3, we propose a method to incorporate smoothness priors over the kernel parameters. Section 4.4 presents our experimental evaluations and results. In Section 4.5, we discuss this approach.

4.1 Partial Derivatives

We optimize the likelihood $\mathcal{L}$ by taking the gradient with respect to the *pseudo-likelihood*. This means we assume all parameters fixed except the one for which we are calculating the derivative. The partial derivative of parameter p of kernel Σ_i belonging to input location $\mathbf{x}_i$ takes the form

$$\frac{\partial}{\partial\theta_{ip}}\log p(\mathbf{y}|\mathbf{X},\Theta) = \frac{1}{2}\mathbf{y}^T K^{-1}\frac{\partial K}{\partial\theta_{ip}}K^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}(K^{-1}\frac{\partial K}{\partial\theta_{ip}}) \tag{4.2}$$

$$= \frac{1}{2}\mathrm{tr}((\alpha\alpha^T - K^{-1})\frac{\partial K}{\partial\theta_{ip}}) \quad \text{where} \quad \alpha = K^{-1}\mathbf{y}\ . \tag{4.3}$$

The proof is given in the appendix in Section A.3. To compute (4.3) we need to calculate the derivative of the covariance matrix

$$\frac{\partial K}{\partial \theta_{ip}} = \begin{pmatrix} 0 & \dots & \frac{\partial k(\mathbf{x}_1,\mathbf{x}_i)}{\partial \theta_{ip}} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial k(\mathbf{x}_i,\mathbf{x}_1)}{\partial \theta_{ip}} & \dots & \frac{\partial k(\mathbf{x}_i,\mathbf{x}_i)}{\partial \theta_{ip}} & \dots & \frac{\partial k(\mathbf{x}_i,\mathbf{x}_n)}{\partial \theta_{ip}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \frac{\partial k(\mathbf{x}_n,\mathbf{x}_i)}{\partial \theta_{ip}} & \dots & 0 \end{pmatrix} . \tag{4.4}$$

This matrix has non-zero entries only for the i-th row and column which contain the covariances of input location $\mathbf{x}_i$. Thus, we need to calculate the derivatives of all covariance functions for the input location $\mathbf{x}_i$ since only these incorporate Σ_i and thus θ_{ip}. These derivations require lengthy algebraic manipulations, but can be derived analytically. They are described in the appendix in Section A.4. The resulting functions can be evaluated in constant time.

4.2 Learning Procedure

As described in Section 3.3, the data-likelihood given in Equation (4.1) contains many local extrema since it is not monotonous in the kernel parameters and different kernel combinations may lead to similar data-fits. In order to avoid bad local optima, we need a suitable learning procedure. This problem is akin to the problem of weight optimization in neural networks. In this area, the adaptation procedure *RProp* [Riedmiller and Braun, 1993] is state-of-the-art. The idea of *RProp* is to perform gradient ascent, but to ignore the length of the gradient because it has no general intuitive interpretation that might be used for adaptation. In contrast, learning is based only on the temporal behavior of the sign of the derivative.

We propose to take up these ideas and adapt *RProp* to our needs. Our learning algorithm is given in Algorithm 1. First, we learn the optimal stationary kernel parameters by standard gradient ascent and use them as initialization for the individual kernels Σ_i. In an iterative procedure, we optimize each kernel Σ_i by means of partial derivatives as follows. Each kernel parameter θ_{ip} maintains its individual update-value α_{ip} which completely defines the length of the update step. α_{ip} is adapted according to the direction of the derivative. If the derivative retains its sign in subsequent iterations, α_{ip} is increased by a factor η^+ in order to speed up convergence (*else if*-part in Algorithm 1). If the sign changes (*else*-part), indicating that the last update was too big, α_{ip} is decreased by a factor η^-. Also, the previous step is completely retracted. Because of this backtracking step, the derivative is supposed to change its sign once again. To avoid double punishment of α_{ip}, the backtracking flag β_{ip} indicates that no adaptation of α_{ip} shall take place in the following step (*if*-part). In our experiments, we found $\eta^+ = 1.1$ and $\eta^- = 0.5$ to produce the best results. We set the adaptation maximum $\Delta_{max} = 3.0$ and the minimum $\Delta_{min} = 0.00001$. The time complexity of this algorithm is as follows. Assume we need m iterations until convergence. In one iteration, we adapt all parameters of each kernel. Given n observations, we have $c \cdot n$ adaptations per iteration ($c = 3, 4$ depending on kernel parameterization). A single adaptation requires the calculation of the partial derivative with respect to this parameter. This necessitates the derivation of the covariance matrix given in Equation (4.4) which requires calculating n covariance functions, namely those that incorporate the respective kernel. The derivative of a single covariance function with respect to a specific kernel parameter is calculated in constant time. Thus, we get a total complexity of $O(m \cdot c\, n \cdot n) = O(m\, n^2)$.

Algorithm 1 Gradient Ascent Kernel Adaptation

Input: observation set D, η^+, η^-, Δ_{max}, Δ_{min}
Output: optimal kernels Σ_i

Learn global parameters $\mathbf{\Theta}_{stat}$ for k_{SE}.
Initialize all local kernels Σ_i with $\mathbf{\Theta}_i = \mathbf{\Theta}_{stat}$.
while not converged **do**
  for all Σ_i **do**
    for all $\theta_{ip} \in \mathbf{\Theta}_i$ **do**
      if β_{ip} **then** // Backtracking in iteration before
        $\alpha_{ip}(t) = \alpha_{ip}(t-1)$
        $\Delta\theta_{ip}(t) = \text{sign}(\frac{\partial \mathcal{L}}{\partial \theta_{ip}}(t)) * \alpha_{ip}(t)$
        $\theta_{ip}(t+1) = \theta_{ip}(t) + \Delta\theta_{ip}(t)$
        $\beta_{ip} = \text{false}$
      else if $\frac{\partial \mathcal{L}}{\partial \theta_{ip}}(t-1) * \frac{\partial \mathcal{L}}{\partial \theta_{ip}}(t) > 0$ **then** // Derivative retains sign
        $\alpha_{ip}(t) = \min(\alpha_{ip}(t-1) * \eta^+, \Delta_{max})$ // $\rightarrow$ Increase convergence speed
        $\Delta\theta_{ip}(t) = \text{sign}(\frac{\partial \mathcal{L}}{\partial \theta_{ip}}(t)) * \alpha_{ip}(t)$
        $\theta_{ip}(t+1) = \theta_{ip}(t) + \Delta\theta_{ip}(t)$
      else // Derivative changes sign
        $\alpha_{ip}(t) = \max(\alpha_{ip}(t-1) * \eta^-, \Delta_{min})$ // $\rightarrow$ Decrease convergence speed
        $\theta_{ip}(t+1) = \theta_{ip}(t) - \Delta\theta_{ip}(t-1)$
        $\beta_{ip} = \text{true}$
      end if
    end for
  end for
end while

4.3 Regularization

Taking the gradient with respect to only one single parameter, as proposed in the previous section, has severe practical limitations. As described in the last chapter, the covariance function k_{NSE} is strongly influenced by the shapes of the kernels of the two input locations. Even if these kernels vary only slightly, the covariance might decrease quickly. This has the effect that the gradient can only moderately indicate kernel modifications if it wants to retain a high covariance value with respect to an input location with a kernel of similar shape. On the other side, decreasing the covariance between two input locations can be achieved simply by choosing different kernel shapes. This easily leads to overfitting. As many different kernel combinations may lead to similar data-fits, the optimization problem is ill-posed. To achieve a model with robust generalization performance, we need to impose the constraint that kernels vary smoothly across input space (cf. Section 3.5). One way to achieve this is by introducing a further constraint into the optimization criterion, i.e., the marginal data-likelihood in our case. This is described by the theory of Tikhonov regression [Tikhonov, 1943], also known as ridge regression [Hoerl, 1962], which explains how to derive a well-posed problem by additional assumptions. In the ridge regression formulation, we want to minimize the functional

$$J[f] = \lambda \parallel f \parallel_H^2 + Q(\mathbf{y}, \mathbf{f}) \tag{4.5}$$

where $\mathbf{y}$ are the observed training targets and $\mathbf{f}$ are the values predicted by function f. The first term is the regularizer which encodes the smoothness assumptions of the hypothesis space H. The second term

assesses the quality of the data-fit. This might be, for instance, the squared error which corresponds to the negative log-likelihood of a Gaussian noise model. λ trades off both terms. The regularization method yields a minimizer f^* of this functional. This minimizer can be viewed as the maximum a-posteriori (MAP) function under the posterior distribution over functions of the GP [Rasmussen and Williams, 2006]. In contrast, the GP model represents a posterior distribution over functions (and not only a single estimated function).

Using the regularizer as defined in Equation (4.5) has no advantage over the GP procedures as it yields the same MAP function. It points out, however, how we can impose on our model the constraint that the kernels vary smoothly across input space. We derive a constrained likelihood $\mathcal{L}_\mathcal{C}$ by introducing a regularizer for each kernel parameter type θ_p ($\theta_p \in \{\ell_1, \ell_2, \alpha\}$ or $\{\ell_1, \ell_2, u, v\}$)

$$R_{\theta_p} = \sum_i \sum_j \frac{\parallel \theta_{ip} - \theta_{jp} \parallel}{\parallel x_i - x_j \parallel} \tag{4.6}$$

which we combine with the original likelihood $\mathcal{L}$ to a new maximization criterion

$$\mathcal{L}_\mathcal{C} = \mathcal{L} - \sum_{\theta_p} \lambda_{\theta_p} R_{\theta_p} \ . \tag{4.7}$$

R_{θ_p} quantifies the difference of the kernel parameters θ_p for neighboring input locations. Since this penalty term does not depend on the targets, it is conceptually equivalent to a prior on the hyperparameters, i.e. the kernel parameters. We use the standard norm $\parallel a \parallel = \sqrt{a^2}$. The regularization term enforces the constraint that the closer two input locations are the more similar are their kernels. The coefficient λ_{θ_p} defines the importance each kernel parameter type has. It is difficult to derive this coefficient analytically, so it should be estimated by means of cross-validation. The derivative of the regularization term is given in the appendix in Section A.4.1.

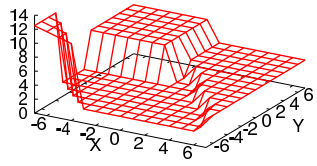
4.4 Experiments

To investigate the modeling capabilities of our gradient ascent approach, we applied it to different synthetic data-sets. In a first experiment, we learned the kernels for a difficult terrain structure. Then, we performed experiments where we adapted lengthscales and orientation separately to assess the general ability of this approach to fit kernel parameters to local properties. We evaluated the following variants of the algorithm:

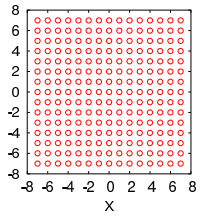
- With and without a regularization term that constrains the kernels to vary smoothly across input space (cf. Section 4.3).
- With and without removal of observation points to assess the generalization capabilities.
- Different approaches for setting kernel matrices on locations where no latent kernel structure has been adapted (e.g., at the gap locations that are not contained in the training set; cf. Section 3.4).

4.4.1 Adapting All Parameters

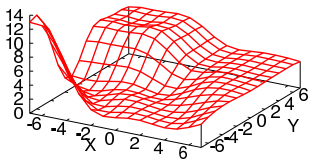
In our first experiment, depicted in Figure 4.1, we applied our learning procedure to a difficult terrain data-set consisting of 225 data-points (Figure 4.1(a)). This set contains uniform regions as well as sharp edges and corners, which are hard to adapt to locally. Note, for example, that the edge 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 approximately in the middle of the diagram. To initialize the kernels at the input locations, we learned the best stationary kernels for this map. Prediction with this stationary GP is



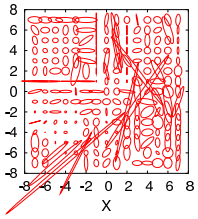
(a) Ground truth and observation



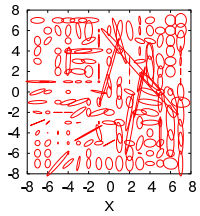
(b) Optimal stationary kernels



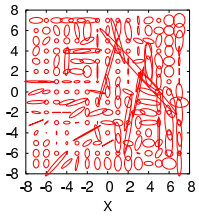
(c) Prediction with stationary GP



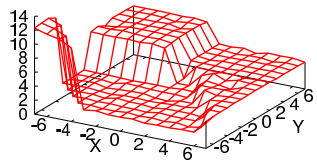
(d) Kernels *without* regularization (0% gaps)



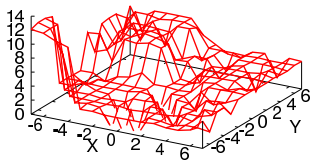
(e) Kernels *without* regularization (15% gaps)



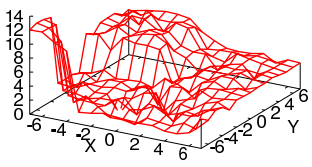
(f) Kernels *without* regularization (15% gaps; kernel-averaging for gaps)



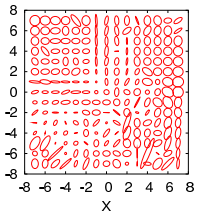
(g) Prediction *without* regularization (0% gaps)



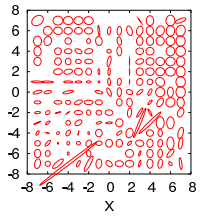
(h) Prediction *without* regularization (15% gaps)



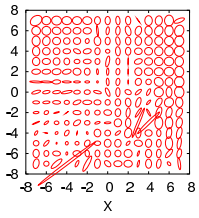
(i) Prediction *without* regularization (15% gaps; kernel-averaging for gaps)



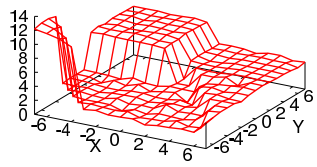
(j) Kernels *with* regularization (0% gaps)



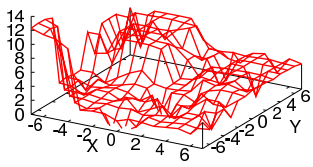
(k) Kernels *with* regularization (15% gaps)



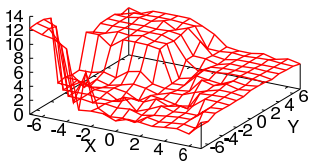
(l) Kernels *with* regularization (15% gaps; kernel-averaging for gaps)



(m) Prediction *with* regularization (0% gaps)



(n) Prediction *with* regularization (15% gaps)



(o) Prediction *with* regularization (15% gaps; kernel-averaging for gaps)

Figure 4.1: Results of gradient ascent adaptation without and with regularization on a difficult artificial map.

poor as it largely oversmoothes the discontinuities in the map. We applied the gradient ascent procedure described in Algorithm 1 to learn individual kernels for the input locations. Convergence was achieved after about 50 iterations.

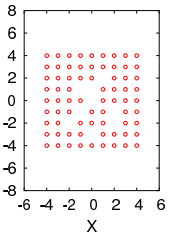
Without regularization term, the learned kernel map looks chaotic: small kernels can be found next to very large kernels and kernel orientations change abruptly. While many of the learned kernels meet expectation, e.g. the long thin kernels at the border of the upper-left plateau, other kernels do not at all correspond to intuition, e.g. the very long, thin kernels at the bottom-left step which are oriented not along, but perpendicular to the step. Nevertheless, the corresponding prediction fits the data very well. This is, however, deceiving as for many parts of the map this is due to severe overfitting. As the kernel shapes vary significantly, the covariances among the input locations get small. This problem becomes evident when we remove data-points in the training set to measure the generalization capabilities of the model. We removed 15% of the data-points and repeated the experiment. The prediction for the gaps is poor as expected. First, we did not assign kernels to the unseen input locations (cf. Section 3.4). In this case, we only make use of the kernel of the other data-point with respect to which we are calculating the covariance. As an alternative, we averaged kernels by means of an isotropic two-dimensional Gaussian with a standard deviation of 2 and set the resulting kernel estimates at the unseen locations which mimics a hierarchical top-model for the kernels. This, however, improved the prediction performance only slightly.

With regularization term, the learned kernel map is much more structured. In most parts of the input space, kernels vary smoothly. In case of 15% of the data-points removed, the prediction quality is still unsatisfactory when one does not assign individual kernels to the unseen locations. If we use neighborhood averages, however, the prediction quality improves. The effect of regularization is clearly visible here. Though still being far from perfect, the resulting predictive map reliably closes the gaps. Close inspection of the filled kernel map reveals that in spite of regularization some kernels still have shapes that differ significantly in comparison to their neighborhood. These outliers might be accounted for by using a quadratic penalty for the regularizer.

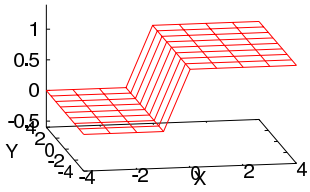
This experiment reveals that finding an optimal kernel map is a hard problem. Due to the many parameters involved, it is prone to local extrema. Nevertheless, the results indicate that our gradient ascent procedure is in principle capable of learning an adequate kernel structure and that regularization is crucial. In order to study this approach in more detail, we simplified the problem-setting to problems where only some of the kernel parameters are adapted.

4.4.2 Adapting Lengthscales I (qualitatively)

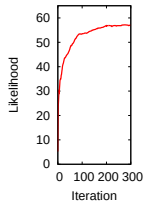
In this experiment, we investigated the capability of our model to reliably adapt the lengthscales of the kernels when their orientations are fixed. Figure 4.2 presents our adaptation results for a small data-set consisting of 81 data-points. 9 points, at least two rows distant from the map border, were randomly removed. The set represents a straight step along the y dimension. A stationary model oversmoothes the step as visualized in Figure 3.2. Our gradient ascent procedure needs to find adequate kernels to preserve the step. *Without regularization* term, a significantly larger data-likelihood is achieved. This is due to the large kernels the model is allowed to produce in this case. Using large kernels results in a smaller complexity penalty which boosts the data-likelihood. Despite of the large kernels, the model is still able to fit the data well by exploiting the prefactor of the covariance function. The model uses two different kernel types to preserve the discontinuity, one for each side of the step. Although both types are large, the covariance across the step is small since the types differ significantly in shape which decreases the prefactor of k_{NSE}. A third completely different kernel type is used along the step. This avoids influence of any of the two step sides which thus corresponds to overfitting. Therefore, the learned model is not able to generalize well and to reliably fill the gaps. In particular, the gaps along the step are filled poorly. These problems are only slightly improved if one places neighborhood average kernels at the unseen locations. In contrast, gradient ascent *with regularization* term is able to predict the discontinuity correctly. The



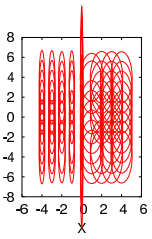
(a) Optimal isotropic kernels



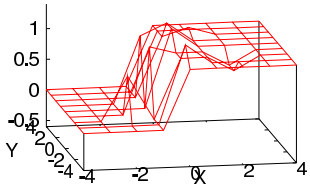
(b) Ground truth



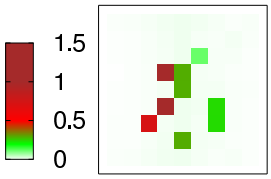
(c) Data likelihood *without* regularization



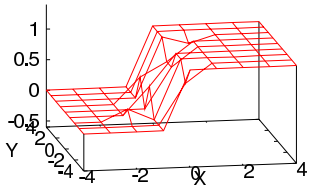
(d) Kernels *without* regularization



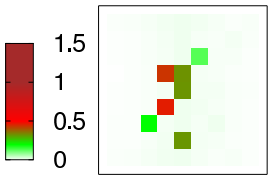
(e) Predicted map *without* regularization



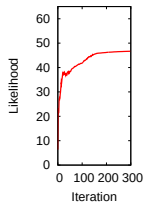
(f) Local errors *without* regularization



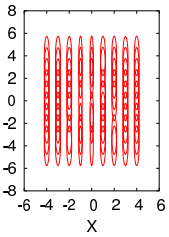
(g) Predicted map *without* regularization and with *kernel-averaging*



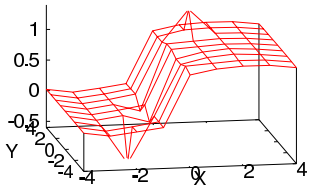
(h) Local errors *without* regularization and with *kernel-averaging*



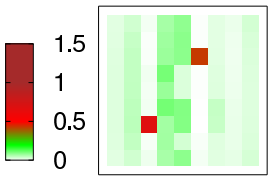
(i) Data likelihood *with* regularization



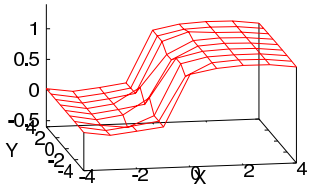
(j) Kernels *with* regularization



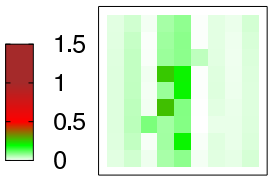
(k) Predicted map *with* regularization



(l) Local errors *with* regularization

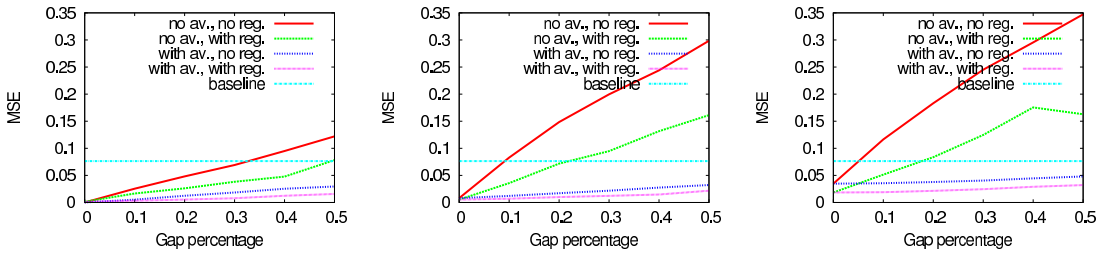


(m) Predicted map *with* regularization and with *kernel-averaging*



(n) Local errors *with* regularization and with *kernel-averaging*

Figure 4.2: Results of gradient ascent adaptation after convergence on artificial terrain with 10% of the data-points removed. Only kernel lengthscales are adapted. For kernel-averaging, the missing kernels of the unseen locations were calculated as an average over the local neighborhood based on an isotropic Gaussian with a standard deviation of 2.



(a) Noise $\sigma = 0.0$ (b) Noise $\sigma = 0.1$ (c) Noise $\sigma = 0.2$

Figure 4.3: Performance of gradient ascent in terms of MSE on map depicted in Figure 4.2(b) with respect to different noise levels and fractions of missing observations (20 runs per configuration).

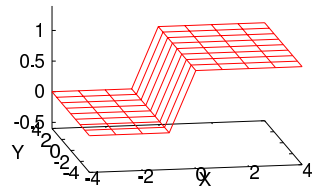
regularization effect can be observed easily: the kernels vary smoothly in input space and are similar. As the discontinuity needs to be modeled by thin, elongated kernels in the center structure, this model is not able to improve on the penalty term of the likelihood by extending the kernel size. This has the desired effect that overfitting is avoided. While all other unseen locations are almost perfectly predicted, two outliers can be observed. If one places averaged kernels at the unseen locations, this problem disappears. This has the minor disadvantage, however, that the discontinuity becomes less sharply represented at some other unseen locations.

4.4.3 Adapting Lengthscales II (quantitatively)

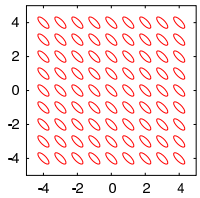
We repeated the previous experiment on the small simulated terrain, which represents a simple step and is depicted in Figure 4.2(b), for different combinations of noise distortion and fractions of missing data. We applied our gradient ascent procedure to learn the lengthscales of the kernels while we left the angles fixed. In particular, we explored noise levels of $\sigma = 0.0, 0.1, 0.2$. As the step has an elevation of 1, it does not make sense to consider larger noise values. The fractions of removed points were 0%, 10%, 20%, 30%, 40%, and 50%. Each combination was repeated 20 times with different random seeds. As evaluation metric, we used the mean squared error

$$\text{MSE}(\mathcal{X}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - y_i^*)^2 \tag{4.8}$$

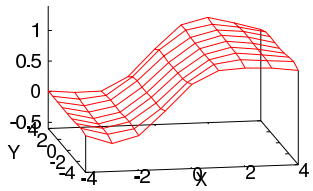
of predicted elevations y_i^* relative to ground truth elevations y_i on the set of input locations $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{81}$. We used the regularization weights $\lambda_x = 0.5$ and $\lambda_y = 10.0$ which we found to produce best results. Kernel-averaging for the unseen locations was done by means of an isotropic Gaussian with a standard deviation of 2. Figure 4.3 presents our results. In all combinations of noise and gap fraction, the gradient ascent adaptation clearly performs better in case of involving a regularization term. Prediction also significantly improves when we place averaged kernels estimated from the local neighborhood at the unseen input locations (instead of using only the other kernel when calculating k_{NSE} with respect to an observation). The baseline corresponds to a simple hyperplane-fit. Using gradient ascent with kernel-averaging always leads to a better performance than the baseline. This proves the capability of our non-stationary GP model to adequatly adapt to and preserve discontinuities.



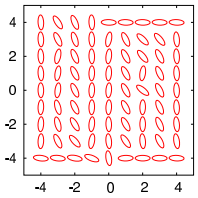
(a) Ground truth and observation



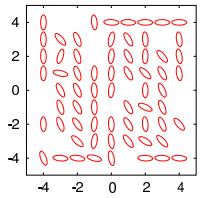
(b) Initial kernels



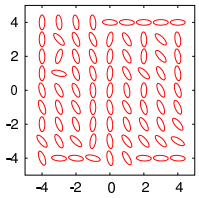
(c) Prediction with initial kernels



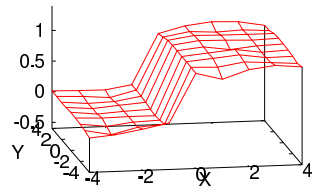
(d) Kernels *without* regularization (0% gaps)



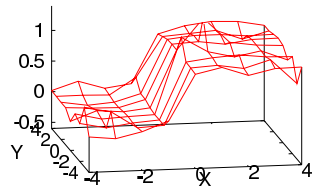
(e) Kernels *without* regularization (15% gaps)



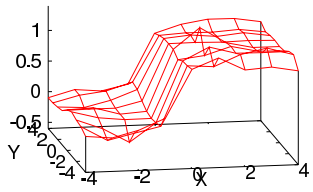
(f) Kernels *without* regularization (15% gaps; kernel-averaging for gaps)



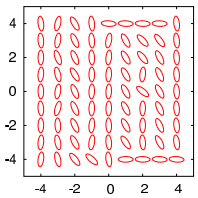
(g) Prediction *without* regularization (0% gaps)



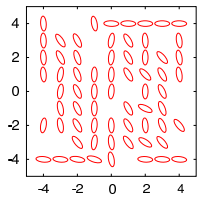
(h) Prediction *without* regularization (15% gaps)



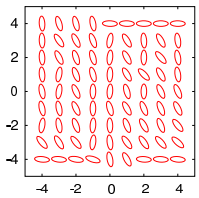
(i) Prediction *without* regularization (15% gaps; kernel-averaging for gaps)



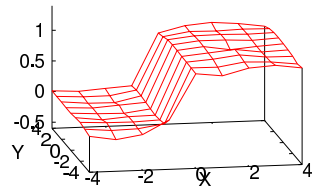
(j) Kernels *with* regularization (0% gaps)



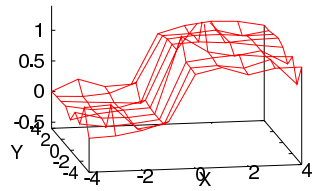
(k) Kernels *with* regularization (15% gaps)



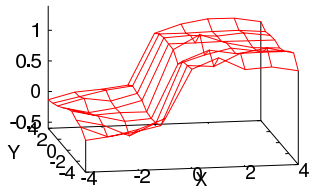
(l) Kernels *with* regularization (15% gaps; kernel-averaging for gaps)



(m) Prediction *with* regularization (0% gaps)



(n) Prediction *with* regularization (15% gaps)



(o) Prediction *with* regularization (15% gaps; kernel-averaging for gaps)

Figure 4.4: Results of adapting kernel orientations by means of gradient ascent

4.4.4 Adapting Orientation

In this experiment, which is presented in Figure 4.4, we investigated learning the optimal kernel orientation by gradient ascent. While keeping the lengthscales of the kernels fixed, their angles were adapted to the terrain structure. Again, we used the terrain with the simple step. We applied the same kernel shape for the whole input space: a thin and elongated kernel initialized at an orientation of 45° with respect to the discontinuity. The optimal orientation is 0° so that the kernels are oriented along the step. In the case of 0% gaps, including a regularization term does not show a benefit. While not all kernels get optimally rotated, the important kernels next to the step receive the correct orientation which leads to a good data-fit. Thus, the learning procedures stops adapting the kernels. Some kernels at the map border, however, get rotated wrongly even when using a regularization term. This seems to be an artifact of their special location. Also if 15% of the data-points are removed, using a regularization term yields only a slight improvement. The same holds for kernel-averaging for unseen locations which leads only to a minor prediction improvement. The discontinuity is well represented which substantiates the claim that gradient ascent optimization is a promising procedure also for learning kernel orientation.

4.5 Discussion of Experimental Results

Gradient ascent optimization is a promising way to learn the local kernels for a non-stationary GP model. It is prone, however, to get trapped by one of the many local optima. We were able to show on a simulated terrain that gradient ascent is able to learn almost optimal lengthscales and angles in separate adaptation procedures. This can be seen as a proof of concept for our approach in a limited problem setting. In our experiments, we did not focus on optimization of the parameters of the learning procedure itself. In order to make this approach work in more complex, real terrains, the learning procedure based on gradient ascent needs more sophistication to avoid local optima. A first step in this direction is the use of a regularization term which constrains kernel parameters to vary smoothly across input space. We have qualitatively and quantitatively illustrated the benefit of such a regularization term. Furthermore, we found that in the prediction for unseen areas it is most useful to estimate the corresponding kernels based on the kernels of the local neighborhood. Already a simple averaging over the neighborhood results in major improvements in prediction quality.

Chapter 5

Terrain Gradient Adaptation

The gradient ascent model presented in the previous chapter provides a flexible and general framework for learning the kernels used by k_{NSE}. It is, however, prone to the many local optima of our terrain modeling problem. As an alternative approach to fitting the non-stationary GP model, we propose to learn the optimal kernels by means of adaptation to the local terrain gradient structure. We model the kernel matrices as independent random variables that are initialized with the learned kernel of the corresponding stationary model and then iteratively adapted to the local structure of the given terrain data. This approach is inspired by work in the computer vision community where the problem of adapting smoothing kernels to local structure has been well studied. 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. Section 5.1 introduces the concept of elevation structure tensors. These tensors guide the adaptation of the local kernels as described in Section 5.2. Section 5.3 presents the complete learning algorithm based on terrain gradients. Section 5.4 describes our experiments. In Section 5.5, we discuss the experimental results.

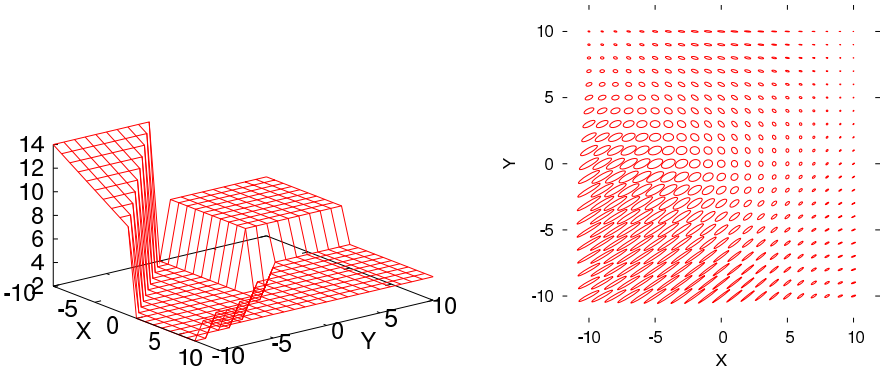
5.1 Elevation Structure Tensors

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 grey-value structure tensor (GST), which captures the local structure of an image or image sequence by building the locally weighted outer product of grey-value gradients in the neighborhood of the given image location. Analogously to their work, we define the *elevation structure tensor* (EST) for a given location $\mathbf{x}_i$ as

$$EST(\mathbf{x}_i) := \overline{\nabla y (\nabla y)^T}(\mathbf{x}_i) \;, \tag{5.1}$$

where $y(\mathbf{x})$ 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 Σ_i. For two-dimensional $\mathbf{x}_i$, Equation (5.1) calculates the locally weighted average of the outer product of $\nabla y = (\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2})^T$. This local elevation gradient can be estimated directly from the raw elevation samples in the neighborhood of the given input location $\mathbf{x}_i$.

Equation (5.1) yields a tensor, representable as a 2×2 real-valued matrix, which describes how the terrain elevation changes in the local neighborhood of location $\mathbf{x}_i$. The first eigenvector of this matrix points into the direction of the strongest ascent. The corresponding eigenvalue measures this ascent, while the second eigenvalue measures the ascent in the perpendicular direction. Thus, small eigenvalues denote flat areas while large eigenvalues are typical for regions of steep ascent, e.g. at discontinuities.



(a) Artificial data-set (b) Tensors

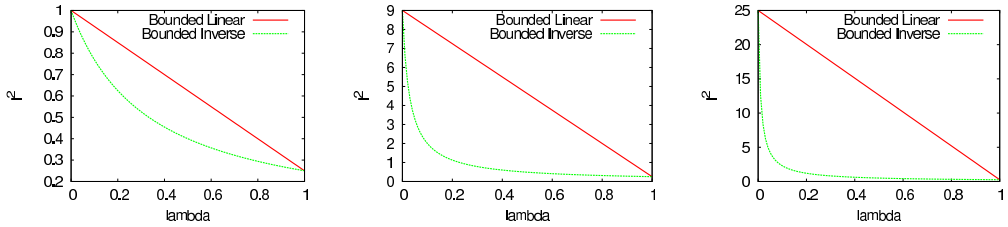
Figure 5.1: Elevation structure tensors (EST) capture local terrain properties.

To get an intuition, how this can guide the adaptation of the local kernel Σ_i, consider the following situations. Let λ_1 and λ_2 denote the eigenvalues of $EST(\mathbf{x}_i)$ 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 ∇y are small in the neighborhood of $\mathbf{x}_i$. This results in two equally small eigenvalues of $EST(\mathbf{x}_i)$. In contrast, if $\mathbf{x}_i$ was located in an ascending part of the terrain, the first eigenvalue of $EST(\mathbf{x}_i)$ would be clearly greater than the second one and the orientation β would point towards the strongest ascent. Figure 5.1 illustrates the tensors of an artificial map. The most prominent structural change is the steep step in the bottom-left corner which is reflected by the long thin tensors oriented perpendicular to this step. Smaller tensors of similar shape dominate at the other smaller steps. In contrast, the homogenous terrain areas in the upper right and upper left corner are represented by very small tensors without a significant elongation.

5.2 Adapting Kernels

As discussed by Middendorf and Nagel [2002], the kernel Σ_i describing the extent of the local environment of $\mathbf{x}_i$ should be set to the inverse of $EST(\mathbf{x}_i)$. This corresponds to intuition: In regions of low structural change such as flat areas, the eigenvalues of the tensors are small. These regions have homogenous local properties, so the covariance is large also with respect to distant input locations. Thus, the kernels should be large. In contrast, the eigenvalues of tensor eigenvectors pointing into directions of rapid structural change are large. Along these directions, covariances should quickly decrease. Therefore, the corresponding kernel eigenvalues should be small. Since the kernel eigenvalues are adapted according to the eigenvector directions of the tensors, the kernel orientation is set corresponding to the orientation of the tensor. By means of this adaptation strategy, flat areas are populated by large, isotropic kernels, while sharp edges have long, thin kernels oriented along the edge directions. 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. Let λ_1 and λ_2 denote the eigenvalues of the EST and α its orientation. To simplify notation, we introduce $\overline{\lambda_k} = \lambda_k/(\lambda_1 + \lambda_2)$, $k = 1, 2$, and use the kernel parameterization of Equation (2.15) restated



(a) $\sigma_{max} = 1.0$ (b) $\sigma_{max} = 3.0$ (c) $\sigma_{max} = 5.0$

Figure 5.2: Curves for adapting ℓ^2 based on the normalized eigenvalue $\overline{\lambda}$ ($\sigma_{min} = 0.5$)

here,

$$\Sigma = R \begin{pmatrix} \ell_1^2 & 0 \\ 0 & \ell_2^2 \end{pmatrix} R^{-1} , \tag{5.2}$$

where the lengthscales ℓ_1 and ℓ_2 scale in orthogonal directions and R is a rotation matrix specified by the orientation angle α.

1. *Direct Inverse* adaptation:

$$\Sigma_i \quad = \quad EST(\mathbf{x}_i)^{-1}$$

2. *Bounded Linear* adaptation:

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

3. *Bounded Inverse* adaptation:

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

Figure 5.2 presents curves for ℓ_k^2 for different σ_{min} and σ_{max}. The *Direct Inverse* adaptation procedure has the disadvantage that the resulting ℓ_k are unbounded. If the EST is very small, the resulting kernel can become unrealistically large which disrupts the complete adaptation procedure. The *Bounded Linear* adaptation procedure leads to rather balanced kernels without a strong bias for one dimension as the resulting eigenvalues are of similar order. In contrast, *Bounded Inverse* strongly favors the smaller eigenvalue. This has the effect that it is more suitable to closely fit the data, but is more prone to overfitting, while *Bounded Linear* copes better with sparse data and gaps.

5.3 Learning Procedure

So far, we have described how to perform one local adaptation step for an arbitrary kernel Σ_i. As the complete learning and adaptation procedure, which is also summarized in Algorithm 2, we propose to assign to each input location $\mathbf{x}_i$ of the training set $\mathcal{D}$ a kernel matrix Σ_i which is initialized with a global parameter vector $\mathbf{\Theta}$. $\mathbf{\Theta}$ has been learned using standard GP learning with the stationary squared exponential covariance function k_{SE}. The local kernels are then iteratively adapted to the elevation structure of the given terrain data-set until their parameters have converged. The local weights in the tensor computation

of the average $\overline{\cdot}$ are chosen according to the current estimate of Σ_i. In order to speed up the adaptation, we introduce a learning rate η_i to make the adaptation speed for each Σ_i depend on the local data-fit, given by

$$\mathrm{df}(\mathbf{x}_i) = \frac{p(y_i|\mathbf{x}_i)}{\max_y p(y|\mathbf{x}_i)} \,. \tag{5.3}$$

This data-fit quantifies the regression error and corresponds to the normalized observation likelihood, i.e., the likelihood of the observation y_i divided by the maximum of the predictive distribution for $\mathbf{x}_i$. We also take the kernel complexity into account to avoid overfitting which we approximate by $c_i = 1/|\Sigma_i| = 1/(l_1^2 l_2^2)$ (see Appendix section A.1). Both components are used to form a learning rate parameter calculated by means of a modified sigmoid function, $\eta_i = \mathrm{sigmoid}(-\mathrm{df}(\mathbf{x}_i) \cdot c_i; \boldsymbol{\delta})$, where the additional parameters $\boldsymbol{\delta}$ are tuned empirically. η_i defines how quickly the current kernel estimate is adapted to the local structure. Intuitively, we get a high adaptation speed when the data-fit relative to the kernel size is small. Algorithm 2 summarizes the adaptation procedure.

Algorithm 2 Local Kernel Adaptation Based on Terrain Gradients

Input: observation set D, σ_{min}, σ_{max}
Output: optimal kernels Σ_i

Learn global parameters $\boldsymbol{\Theta}_{stat}$ for k_{SE}.
Initialize all local kernels Σ_i with $\boldsymbol{\Theta}_i = \boldsymbol{\Theta}_{stat}$.
while not converged **do**
  for all Σ_i **do**
    Estimate the local learning rate η_i.
    Estimate $\mathrm{EST}(\mathbf{x}_i)$ according to Σ_i.
    $\Sigma_i^* \leftarrow \mathrm{ADAPT}(\mathrm{EST}(\mathbf{x}_i))$
    $\Sigma_i \leftarrow \eta_i \Sigma_i^* + (1 - \eta_i)\Sigma_i$
  end for
end while

5.4 Experiments

To evaluate the terrain gradient adaptation approach, we performed several experiments on synthetic and real data. First, we tested our approach on a small, though difficult artificial terrain data-set. Then, we investigated two real-world scenarios to prove the capability of our approach to predict terrain elevations in occluded areas. In a final experiment, we applied our approach to a large real-world environment. For the real-world data-sets, we acquired sets of 3D scans of a scene using a mobile robot equipped with a laser range finder and a pan-tilt unit. To evaluate our predictions, we use the mean squared error (MSE) as given in Equation (4.8).

5.4.1 Artificial Terrain Data

The first set of experiments was designed to illustrate the terrain gradient adaptation approach, to quantify the benefits of local kernel adaptation, and to compare the three different adaptation rules. As a test scenario, we took the artificial terrain data-set depicted in Figure 5.3 consisting of 441 data-points, which contains uniform regions as well as sharp edges and corners, which are hard to adapt to locally. Note, for

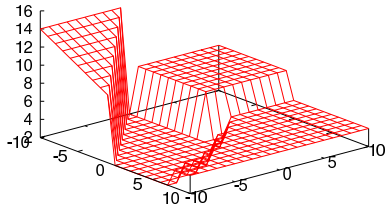


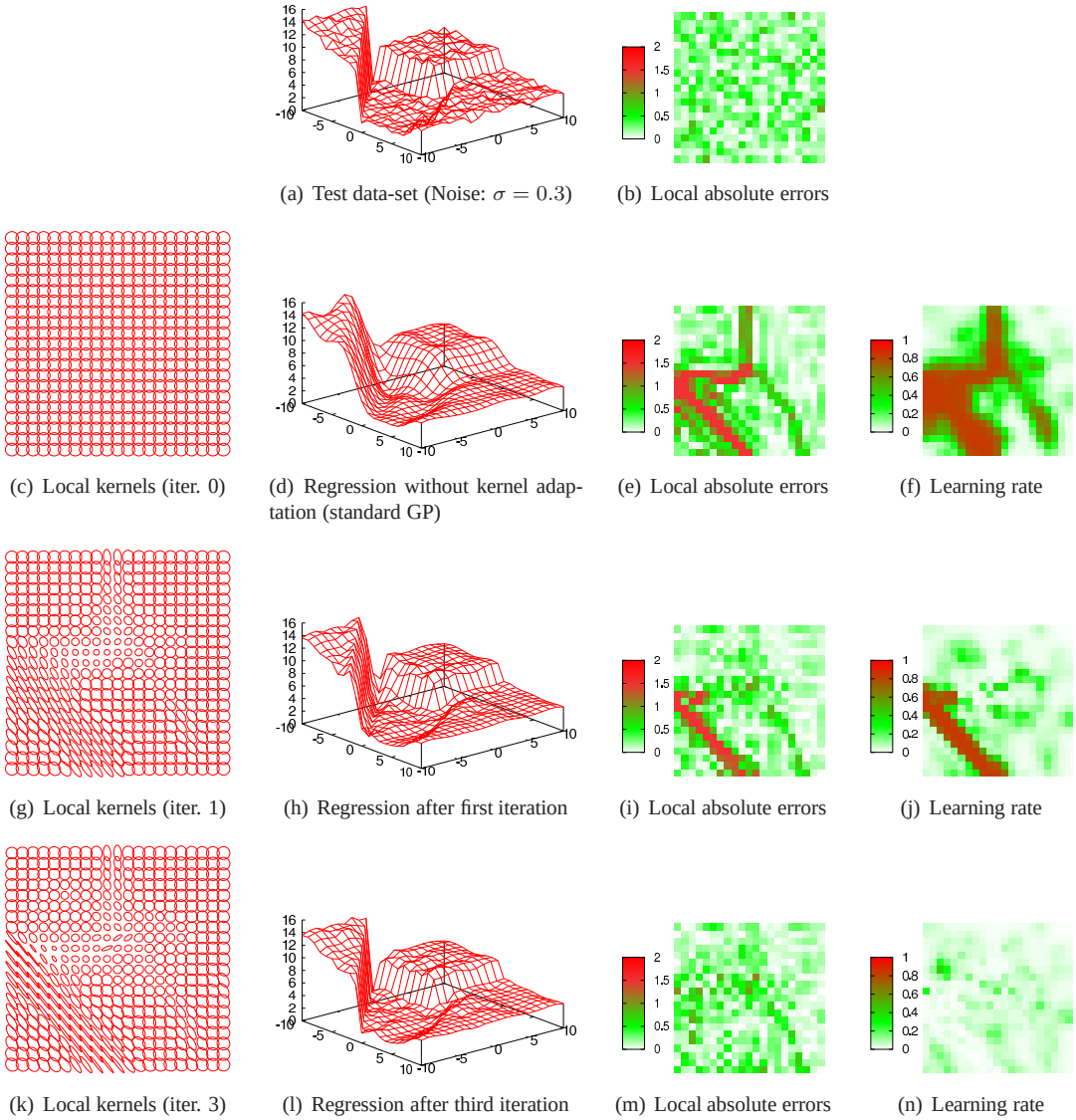
Figure 5.3: An artificial scenario used in the experimental evaluation, that exhibits several local features that are hard to adapt to. Test data-sets are generated by adding white noise and then randomly removing a portion of the data-points.

example, that the edge between the lowest and the second lowest plateau has a curvature and that three different height levels are present in the vicinity of the corner approximately in the middle of the diagram. 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.

Figure 5.4 visualizes a complete adaptation process using the *Bounded Inverse* adaptation procedure for a data-set generated with a noise rate of $\sigma = 0.3$. Figures 5.4(c)-5.4(e) give 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 also quantified by the absolute errors in the third column. Consequently, those locations get assigned a high learning rate, see right column, for the first local kernel adaption step.

The first adaptation step leads to the results depicted in Figures 5.4(g)-5.4(i). 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 (Figures 5.4(k)-5.4(m)). After this iteration, the local learning rates, see Figure 5.4(n), have all settled to low values. It is important to note that the local error distribution is non-uniform over the terrain as can be seen from Figure 5.4(m). In homogenous areas such as the upper right plateau, denoising has been successful so errors are small there. Along the discontinuities already minor prediction errors may lead to high error measures. Thus, local errors are larger there although they have drastically been reduced in comparison to the standard GP model.

In a second experiment, we investigated the prediction performance of our approach for all three adaptation rules presented in Section 5.2. We added white noise of a varying noise level to the artificial terrain depicted in Figure 5.3. The diagrams in Figure 5.5 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% of the data-points removed, *Direct Inverse* and *Bounded Inverse* are not competitive. In contrast, *Bounded Linear* still achieves good results for all noise levels. Overall, *Bounded Linear* produces reliable predictions for all tested noise rates and data densities and we therefore employed this adaptation rule in all subsequent experiments. Figure 5.6 depicts 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 at most 6 iterations, the errors have settled close to their final value. Concerning the computational efficiency, a single iteration per run takes in average about 44 seconds on this data-set on a PC with a 2.8 GHz processor and a 2 GB CPU.



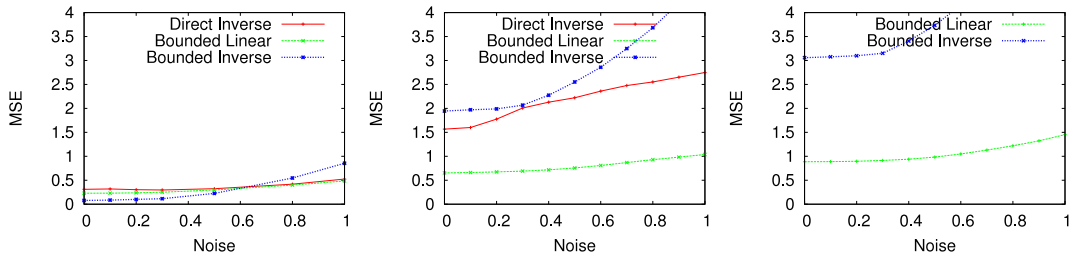
(a) Test data-set (Noise: $\sigma = 0.3$) (b) Local absolute errors

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

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

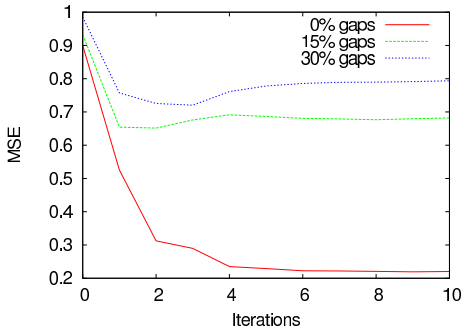
(k) Local kernels (iter. 3) (l) Regression after third iteration (m) Local absolute errors (n) Learning rate

Figure 5.4: The local kernel adaptation process on an artificial terrain data-set: the original data-set, depicted in Figure 5.3, exhibits several local features that are hard to adapt to. The test data-set (a) was generated by adding white noise, resulting in the absolute errors shown in (b). The second row of diagrams depicts 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 in 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 local learning rates η_i for the next adaptation step.



(a) All data-points given. (b) 15% of the data-points removed. (c) 30% of the data-points removed.

Figure 5.5: Prediction accuracy for the scenario depicted in Figure 5.3 with (a) all data-points available, (b) 15% of the data-points removed randomly, and (c) 30% removed randomly. 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).



(a)

Iteration	0	1	2	3	4	5
Gaps 0%	0.900	0.525	0.313	0.290	0.235	0.229
Gaps 15%	0.933	0.654	0.651	0.676	0.691	0.686
Gaps 30%	0.985	0.757	0.726	0.720	0.761	0.778

(b)

Figure 5.6: The mean squared error (MSE) of predicted elevations for the scenario depicted in Figure 5.3 converges with an increasing number of adaptation steps. Iteration 0 gives the MSE for the learned standard GP. Values are averages over ten independent runs.

5.4.2 Real Scenario I: Occluded Stone Block

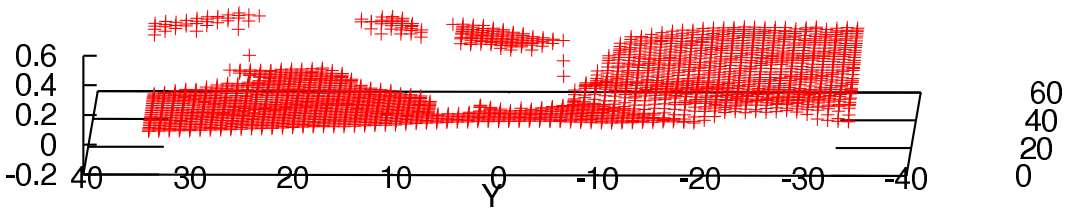
In this experiment, we investigated the ability of our terrain model approach to preserve and predict sharp discontinuities in real terrain data. We positioned the robot in front of a rectangular stone block such that the straight edges of the block run diagonally to the robot's line of view. A person stood inbetween the robot and the block, thereby occluding parts of the block and of the area in front of it. This scenario is depicted in Figure 5.7(a). The task is to recover the linear structure of the discontinuity and fill the occluded area consistent with the surrounding terrain elevation levels.

To simplify the calculations and to enhance the visibility of the results, we manually removed the person from the data-set in this scenario. Although this is a simplification giving an advantage to our model, it is not unrealistic: it corresponds to a pre-classification of the scence where data-points that differ significantly in height are assigned different class label. A more advanced algorithm could also directly recognize the human-being as such and deduce that it is a dynamic object which should not be used in the adaptation of the terrain model. To derive a ground truth map, we repeated the laser scan without the person. Figure 5.8(a) depicts the resulting elevation levels of the ground truth while Figure 5.8(b) illustrates the observed elevations within the occluded terrain. The observed data-set consists of 2,219 data-points.

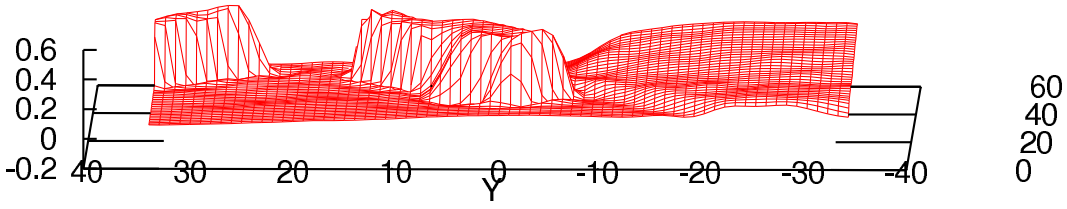
We applied our learning algorithm with the *Bounded Linear* adaptation procedure where we set $\sigma_{min} = 0.25$ and $\sigma_{max} = 4.0$. The adaptation procedure converged already after two iterations. One iteration took about 22 minutes on a standard PC of which only 7 seconds are needed for adapting the kernels. The remaining time is used to calculate the new covariance matrix and the likelihoods of the training points. The resulting kernels are visualized in Figure 5.8(e). To estimate the kernels of the 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. The adapted kernels reflect the terrain structure very well. They are oriented along the discontinuities of the stone blocks. Note in particular the curve structure at the bottom-left of the main block. In contrast, the initial isotropic kernels have been kept in the homogenous areas, i.e., on the ground terrain and on top of the blocks. They are also used in the slowly ascending area next to the main stone block. Note that although the kernels in the occluded area of the strong discontinuity are rather isotropic, the covariance structure at these locations has a significant orientation as it is constructed by averaging over the neighboring kernels (cf. Equation (3.3) for the non-stationary covariance function k_{NSE}). The learned kernel structure enables the model to correctly adapt to the stone blocks as can be seen from the predicted elevations visualized in Figure 5.8(c) and the predicted map in Figure 5.7(c). In particular, the area occluded by the person is correctly filled: due to the long, diagonally oriented kernels, the discontinuities of the stone blocks are predicted sharply while the tops of the blocks and the areas in front of them are correctly estimated as flat terrain.

The uncertainties of these predictions, corresponding to the variances of the predictive distributions, are illustrated in Figure 5.8(d). They are large in the areas behind the stone blocks where no observations are available. The corresponding predictions tend to the mean of the GP model which is 0, i.e., the height of the ground terrain. In contrast, the predictive variances are almost zero at the observed input locations. Most importantly, they are also small at those occluded locations that are in reasonable distance of some observations. This holds in particular for the area occluded by the person. This is also visualized in Figure 5.8(c) where the contour lines for predictive variances of 0.005 and 0.05 are given. Within the area occluded by the person, the model is certain about its prediction in a range of ± 14cm (this range covers 95.5% of the density mass of the predictive distribution). This is an impressive result given the large discontinuity in this area. A mobile robot would thus be relatively certain about the block structure within the gap although not having observed it directly. In contrast, it would be aware that it cannot rely upon its terrain model in the occluded areas beyond the blocks: there are no observations within a reasonable distance and thus, the predictive variances are large.

(a) Photo of the scenario

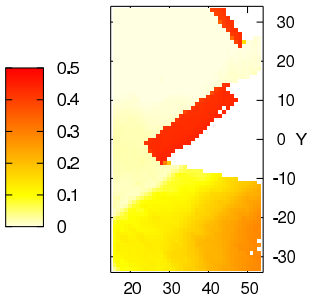


(b) Raw observations

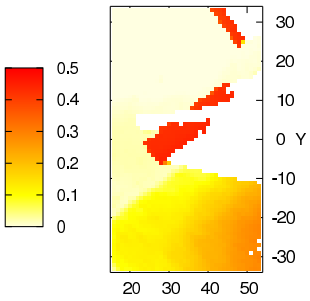


(c) Prediction

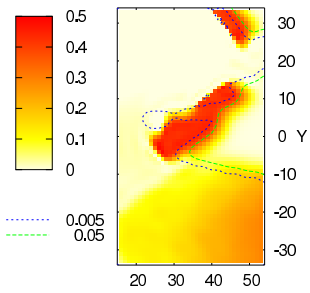
Figure 5.7: A real-world scenario where a person blocks the robot's view on a stone block. The edges of the stone run diagonally to the robot's line of view. Figure (c) depicts the prediction of the terrain model which was learned from the observations visualized in (b). The sharp edges of the stone are preserved and the stone structure is correctly retrieved in the occluded area. This is due to the adapted kernels as visualized in Figure 5.8.



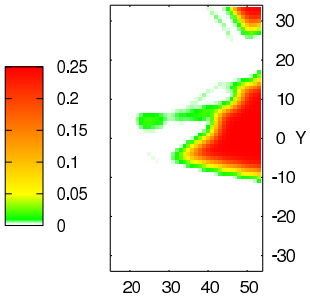
(a) Ground truth elevations



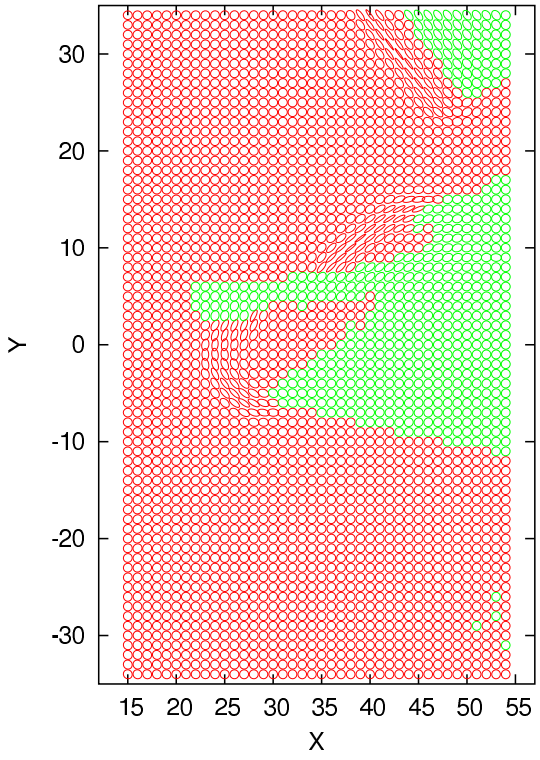
(b) Observed elevations



(c) Predicted elevations



(d) Predictive uncertainties



(e) Kernels

Figure 5.8: Results for the occluded stone block scenario presented in Figure 5.7. Figures (a)-(c) visualize the elevations of the ground-truth (obtained from a second scan without the obstacle), of the observations, and of the prediction. Figure (c) also contains two contour lines for the predictive variances given in (d). Figure (e) depicts the kernels which have been adapted along the block edges.

Scenario	Linear Interpolation	Our model	Improvement
1	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%

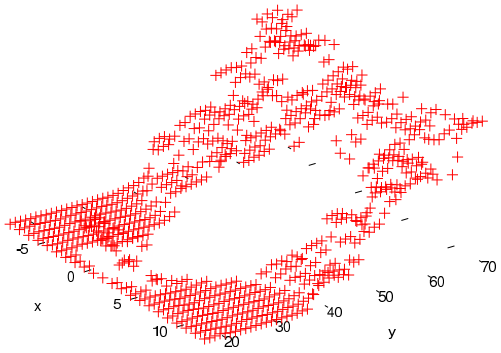
Table 5.1: Prediction performance for occluded hill structures as illustrated in Figure 5.9 in terms of MSE relative to a second, not occluded scan.

5.4.3 Real Scenario II: Inhomogenous Hill Structure

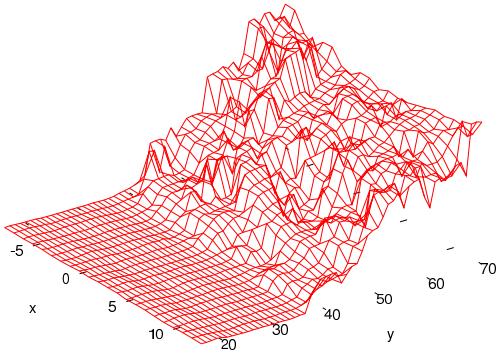
The previous experiment demonstrated the usefulness of our approach for preserving and estimating sharp linear discontinuities. To prove that our approach is also able to predict non-linear structures in unseen areas, we investigated a second occlusion scenario. Again, a person stood in front of the robot and shadowed some parts of the terrain. The occluded area contains varying structures, flat areas as well as an ascending slope of a small hill. A photo of this scenario is given in Figure 5.9(a). We applied our learning algorithm with the *Bounded Linear* adaptation procedure where we set $\sigma_{min} = 0.25$ and $\sigma_{max} = 4.0$. Figure 5.9 illustrates the results of this experiment.

We compared our prediction results to an approach from the robotics literature [Früh *et al.*, 2005] that has been applied successfully to the problem of three-dimensional mapping of urban areas. Früh *et al.* perform horizontal linear interpolation orthogonally to the robot's view. We evaluated our approach on the situation depicted in the figure as well as three similar ones. Note that the scenarios used are actually rather easy ones for Früh *et al.*, as the large gaps can all be filled orthogonally to the robot's view, which is not the case in general. Table 5.1 gives the obtained results. In all four cases, our approach achieved higher prediction accuracies, reducing the errors by 30% to 70%. Figure 5.9(d) depicts the predictions of our approach in one of the situations. In contrast to Früh *et al.*, our model is able to also estimate its predictive uncertainties. These are largest in the center of the occluded area as can be seen in Figure 5.9(e). Figure 5.9(f) depicts the predictive distribution for terrain elevations along a cut through the map. The estimated means are close to the true terrain structures while the corresponding variances indicate that the regression model is aware of its predictive uncertainties.

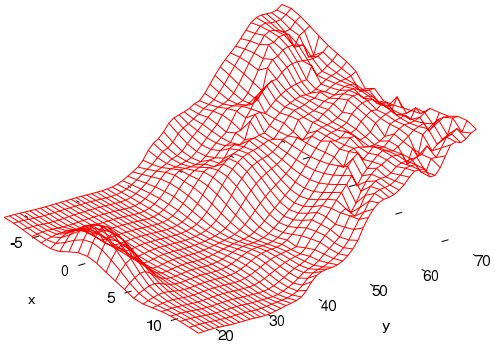
(a) Photo of the scenario



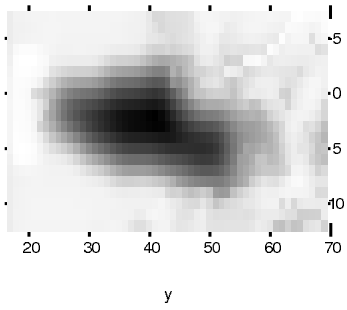
(b) Raw observations



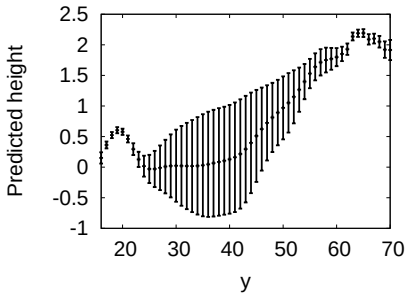
(c) Ground truth



(d) Prediction



(e) Predictive uncertainties (white: zero predictive variance)



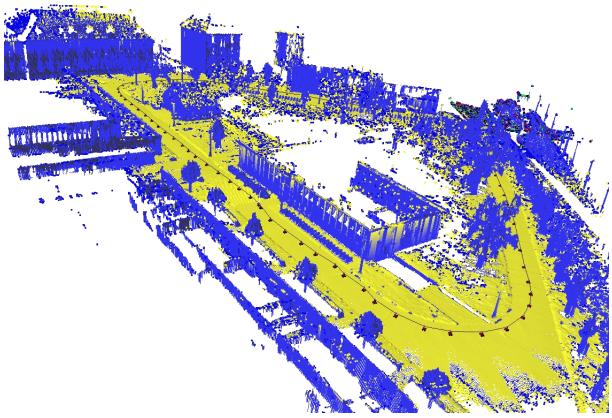
(f) Predictive distribution for $x = 3$

Figure 5.9: A real-world scenario, where a person blocks the robot's view on an inhomogeneous and sloped terrain (a). Figure (b) depicts the raw data-points. Figure (d) gives the predicted means of our adapted non-stationary regression model. Figure (c) depicts the true terrain elevations as aquired by recording a second 3D scan without the person in the scene. Importantly, our model also yields the predictive uncertainties for the predicted elevations as depicted in Figure (e). Figure (f) visualizes the full predictive distribution as a cut through the terrain at $x = 3m$.

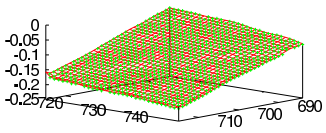
5.4.4 Real Scenario III: Large Campus Environment

To evaluate our approach on a large-scale environment, we applied it to a real-world data-set recorded on the campus site of the University of Freiburg. Figure 5.10(a) illustrates this scenario. The data is represented by means of a multi-level surface map with a cell size of 10cm $\times$ 10cm. The scanned area spans approximately 299 by 147 meters. For simplicity, we only considered the lowest data-points per location, i.e., we removed overhanging structures like tree tops or ceilings. Our resulting data-set consisted of 531,920 data-points. To speed up computations, we split this map into 542 overlapping submaps such that at least 80% of the cells of the resulting submaps are occupied. 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 about 20% of the data-points per submap. A full run over the complete data-set took about 50 hours.

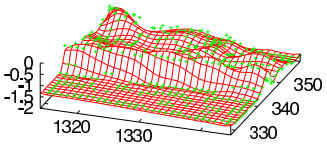
We compared the prediction performance of the three different adaptation rules with respect to a standard stationary GP model where the local kernels are not adapted. The table in Figure 5.10(h) gives the results of this experiment. The *Bounded Linear* and *Bounded Inverse* adaptation rules clearly outperform the standard GP model, while *Direct Inverse* is not competitive. Figures 5.10(b)-5.10(g) illustrate the model predictions and kernels learned by means of *Bounded Linear* for three submaps. Figure 5.10(b) depicts a submap whose terrain structure is simple and easy to model. Kernels do not need to be adapted as the data-likelihoods of the training points are sufficiently high using the initial stationary GP. The terrain of the submap visualized in Figure 5.10(c) is of medium complexity. It contains a long wall next to a flat area which is reflected by the elongated kernels. Figure 5.10(d) presents a challenging terrain with a complex structure. This submap has been the one with the worst prediction performance which is also due to the comparatively large elevation differences (which have been scaled in the figure due to space constraints) which punish already minor prediction errors. This variety of submaps shows that our model is able to cope with real terrain types of varying complexity. As the comparison to the stationary GP model shows, our approach gains its predictive power from the adapted local kernels.



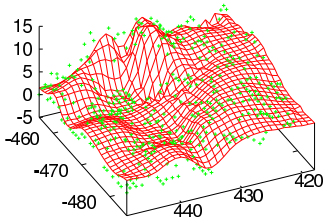
(a) Large campus map of University of Freiburg (Figure courtesy of Patrick Pfaff and Rudolph Triebel)



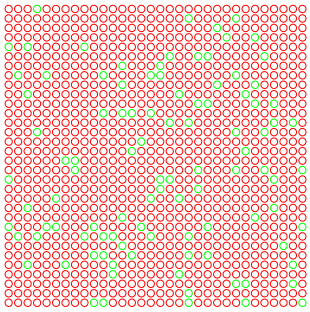
(b) Submap containing simple structure only (MSE = 0.000001)



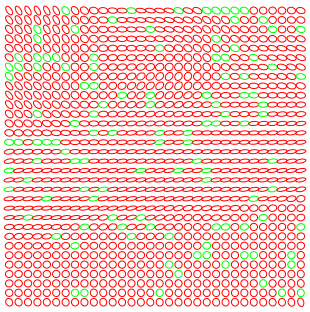
(c) Submap containing structure of medium complexity (MSE = 0.0103)



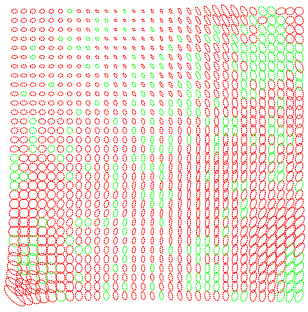
(d) Submap containing difficult structure (MSE = 3.1685)



(e) Adapted kernels of submap containing simple structure only



(f) Adapted kernels of submap containing structure of medium complexity



(g) Adapted kernels of submap containing difficult structure

Adaptation procedure	MSE
No adaptation	0.071
Direct Inverse	0.103
Bounded Linear	0.062
Bounded Inverse	0.059

(h) Prediction performance

Figure 5.10: Experiment on a large campus environment. We compare our three adaptation procedures with a standard GP. Figures (b)-(g) illustrate the predictions and learned kernels for three submaps of varying terrain structure complexity. In (b)-(d), the green points depict the observations while the red lines illustrate the prediction. In (e)-(f), the learned kernels for the observed locations are given in red, while the estimated kernels for the unseen locations are drawn in green.

5.5 Discussion of Experimental Results

Adaptation based on terrain gradients is an effective way of learning the local kernels of our non-stationary terrain model. We use learning rates based on the local data-fits and kernel complexities to speed up computations. Various experiments on synthetic and real data demonstrate that this iterative procedure converges. We have shown on a difficult artificial terrain data-set that our model successfully balances smoothing against the preservation of local structure such as sharp discontinuities. In several real-world scenarios, we have shown that our model is able to recover occluded terrains of different structures, sharp linear discontinuities as well as inhomogenous non-linear terrain structure. In an experiment on a large, heterogenous real terrain, we proved the applicability of our approach to complex large-scale data-sets. More work needs to be done with respect to the computational complexity of our approach. While our kernel adaptation procedure is comparibly efficient, the standard GP calculations (building and inverting the covariance matrix based on the new kernels) have cubic time complexity. Established techniques like sparse GP models and approximative algorithms should be able to significantly reduce the computation times.

Chapter 6

Conclusions

In this thesis, we have proposed an adaptive terrain modeling approach that balances smoothing against the preservation of discontinuities. The latter is particularly important in the area of outdoor robotics where, for example, steps, stairs, or building walls provide important features for path planning or terrain segmentation tasks. Our model uses Gaussian process regression with non-stationary covariance functions to locally adapt to the structure of the terrain data. This makes it possible to account for discontinuities such as edges and corners while at the same time being able to achieve strong smoothing in flat areas and along edges. The learned model yields predictive height distributions for arbitrary locations of the terrain. This enables us to fill gaps in the data stemming from occlusions and faulty observations as well as to assign uncertainty estimates to our predictions. Our model is able to account for all types of occluded terrains, highly non-linear inhomogenous terrain structures as well as sharp linear discontinuities.

We achieve non-stationarity in our model by assigning local kernels to input locations. These kernels capture the local terrain properties. The covariance between two elevation variables is calculated by averaging the two individual kernels involved. In this way, the local characteristics at both locations influence the covariance of the corresponding target elevations. The main task in learning an adequate terrain model consists of adapting these kernels to the local terrain properties. We have shown in a detailed analysis of our non-stationary covariance function that different kernel combinations may lead to similar data-fits. Without additional constraints, the problem of finding optimal kernels is clearly ill-posed due to the many local optima caused by the non-stationary covariance function. We have argued that it is sensible to require the kernel parameters to vary smoothly across input space. Using a full hierarchical model to achieve this by placing additional GP priors on the kernels is only feasible for small data-sets. As terrain models typically contain huge numbers of data-points, we have introduced two different learning algorithms. First, we have investigated a gradient ascent approach over the data-likelihood. We adopt the kernel parameters in a fashion similar to the *RProp* algorithm used in neural networks. We have demonstrated that inserting a regularization term into the optimization criterion is a suitable means to achieve smoothness of the kernels across input space. This improves the generalization performance of the learned model which makes it possible to better fill gaps. We have shown that this approach is applicable to limited problem settings and yields acceptable results there. If applied to real data, however, it is prone to be trapped by one of the many local optima. To overcome this problem, more work needs to be done with respect to the learning procedure, in particular the way the kernel parameters are modified according to their gradients. In our second approach to learning the local kernels, we adapt the kernel parameters based on the local gradients in the terrain structure. We employ adaptation rules that have proven successful in problem domains studied by the computer vision community. In experiments on synthetic and real data, we have demonstrated that this learning method produces accurate and reliable predictions in the presence of noise and is able to fill gaps stemming from occlusions. The time complexity of our approach is dominated by the standard GP

model calculations, in particular the inversion of the covariance matrix, while the adaptation of the kernels requires only a small fraction of the overall runtime.

6.1 Outlook

A main objective of future work is to strengthen the connection between our terrain modeling approach and the concept of multi-level surface maps. In particular, it should be investigated how predictions of our model can be best incorporated into such representations. In this work, we have focused on single-leveled elevation maps only. In natural scenarios, however, we have to deal with several levels, for example in the case of bridges and trees. The difficulty is how to assign individual terrain models to the different elevation levels found in the multi-level surface map and how to exploit the relations among the individual models. Furthermore, for real-world applications it is important to explore how large environments can be processed most efficiently while keeping prediction quality high. In our work, we split up the map of a large campus environment into several submaps to be able to compute terrain models in reasonable time. It would be interesting to investigate the optimal submap size that balances model learning speed with the quality of the resulting predictions. Also, other ways to improve the efficiency of our learning algorithms are worth exploring, for example by using approximations for the linear algebra involved and by the use of sparse GPs. This might significantly speed-up the process of rebuilding the GP model with the new kernels, i.e., calculating and inverting the covariance matrices, which is the most expensive part in our formulation. We have used preprocessed surface maps as data-structures for our terrain models. It might be advantageous to work directly on the sensory inputs, typically point clouds, and thereafter build a surface map based on the learned terrain model. Another requirement is how to cope with dynamic objects and new observations. One should investigate how the kernels of new training points are most efficiently adapted in an incremental, online fashion. One could restrict oneself only to the local neighborhood which makes it possible to apply a full hierarchical model with MCMC.

Although the terrain gradient approach produces good results, it would be desirable to have an analytical derivation for optimal kernels based solely on data-likelihoods and model complexity. We have proposed a gradient ascent technique which yields good results in small artificial data-sets. In order to be able to apply this approach in real-world scenarios, this learning technique needs to become more robust against local optima. There are many possibilities to improve our proposed gradient ascent procedure, in particular in the way the parameter values are modified and in further exploring the idea of using a regularization term to achieve smoothness of the kernels across input space. Alternatively, one could continue the work on Paciorek's hierarchical model which uses MCMC to learn the local kernels. In particular, its efficiency needs to be improved in order to become applicable to larger data-sets. Similarly, variational methods for covariance adaptation appear promising. The learned kernel structure is an interesting intermediate representation level by itself. As we retrieve terrain properties in terms of the kernels, its application to terrain segmentation and classification is promising. Another interesting problem is how to represent this structure more efficiently which would lead to a dense model of the underlying terrain. For example, in a flat area all input locations share approximately the same kernel type and could thus be represented sparsely by a single kernel. Using a sparse hierarchical model in this way might lead to more effective ways to estimate the kernels of unseen locations.

Various applications of our model are worth exploring. Many types of autonomous agents need to build terrain models from their sensor measurements in order plan their pathes. Autonomous cars, for instance, collect sensory data while driving. This leads to serious gaps in the reconstructed map which need to be filled. In the RoboCup Rescue scenario, autonomous robots need to deal with varying terrain structures and to reliably recognize the forms of obstacles in order to find the best paths and behaviors to reach their goal. There is increasing research in using helicopters and blimps for landscape mapping. Based on stereo recordings, our model might provide reliable estimates of the scanned area. A further promising application

is seebed mapping where coping with varying data densities and the preservation of discontinuties are key requirements. Furthermore, as our terrain gradient approach uses ideas from computer vision, it might be inspiring to evaluate it on typical test cases in computer vision and to compare it with the algorithms of this community.

Appendix A

Mathematical Background

A.1 Kernel Matrices

A.1.1 Standard Parameterization

Using the standard parameterization of the rotational matrix with rotation angle α, a kernel matrix takes the form

$$\Sigma = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} \ell_1^2 & 0 \\ 0 & \ell_2^2 \end{pmatrix} \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \tag{A-1}$$

$$= \begin{pmatrix} (\cos\alpha)^2\ell_1^2 + (\sin\alpha)^2\ell_2^2 & \cos\alpha\sin\alpha\ell_1^2 - \sin\alpha\ell_2^2\cos\alpha \\ \sin\alpha\cos\alpha\ell_1^2 - \cos\alpha\ell_2^2\sin\alpha & (\sin\alpha)^2\ell_1^2 + (\cos\alpha)^2\ell_2^2 \end{pmatrix} . \tag{A-2}$$

Its determinant is

$$\begin{aligned} |\Sigma| &= (((\cos\alpha)^2\ell_1^2 + (\sin\alpha)^2\ell_2^2) \cdot ((\sin\alpha)^2\ell_1^2 + (\cos\alpha)^2\ell_2^2)) \\ &\quad -((\sin\alpha\cos\alpha\ell_1^2 - \cos\alpha\ell_2^2\sin\alpha) \cdot (\cos\alpha\sin\alpha\ell_1^2 - \sin\alpha\ell_2^2\cos\alpha)) \\ &= (\cos\alpha)^2(\sin\alpha)^2\ell_1^4 + (\cos\alpha)^4\ell_1^2\ell_2^2 \\ &\quad +(\sin\alpha)^4\ell_1^2\ell_2^2 + (\cos\alpha)^2(\sin\alpha)^2\ell_2^4 \\ &\quad -(\sin\alpha)^2(\cos\alpha)^2\ell_1^4 + (\sin\alpha)^2(\cos\alpha)^2\ell_1^2\ell_2^2 \\ &\quad +(\sin\alpha)^2(\cos\alpha)^2\ell_1^2\ell_2^2 - (\sin\alpha)^2(\cos\alpha)^2\ell_2^4 \\ &= ((\sin\alpha)^4 + (\cos\alpha)^4 + 2(\sin\alpha)^2(\cos\alpha)^2)\ell_1^2\ell_2^2 \\ &= ((\sin\alpha)^2 + (\cos\alpha)^2)^2\ell_1^2\ell_2^2 \\ &= \ell_1^2\ell_2^2 . \end{aligned} \tag{A-3, A-4}$$

A.1.2 Overparameterization

Let $l_{uv} = \sqrt{u^2 + v^2}$. When using an overparameterized rotational matrix, a kernel matrix has the form

$$\Sigma = \begin{pmatrix} \frac{u}{l_{uv}} & \frac{-v}{l_{uv}} \\ \frac{v}{l_{uv}} & \frac{u}{l_{uv}} \end{pmatrix} \begin{pmatrix} \ell_1^2 & 0 \\ 0 & \ell_2^2 \end{pmatrix} \begin{pmatrix} \frac{u}{l_{uv}} & \frac{v}{l_{uv}} \\ \frac{-v}{l_{uv}} & \frac{u}{l_{uv}} \end{pmatrix} \tag{A-5}$$

$$= \begin{pmatrix} \left(\frac{u}{l_{uv}}\right)^2\ell_1^2 + \left(\frac{v}{l_{uv}}\right)^2\ell_2^2 & \frac{u}{l_{uv}}\ell_1^2\frac{v}{l_{uv}} - \frac{u}{l_{uv}}\ell_2^2\frac{v}{l_{uv}} \\ \frac{u}{l_{uv}}\ell_1^2\frac{v}{l_{uv}} - \frac{u}{l_{uv}}\ell_2^2\frac{v}{l_{uv}} & \left(\frac{v}{l_{uv}}\right)^2\ell_1^2 + \left(\frac{u}{l_{uv}}\right)^2\ell_2^2 \end{pmatrix} . \tag{A-6}$$

Let $\bar{u} := \frac{u}{l_{uv}}$, $\bar{v} := \frac{v}{l_{uv}}$. Then, its determinant is

$$\begin{aligned} |\Sigma| &= (\bar{u}^2\ell_1^2 + \bar{v}^2\ell_2^2)(\bar{v}^2\ell_1^2 + \bar{u}^2\ell_2^2) - (\bar{u}\bar{v}\ell_1^2 - \bar{u}\bar{v}\ell_2^2)(\bar{u}\bar{v}\ell_1^2 - \bar{u}\bar{v}\ell_2^2) \\ &= \bar{u}^2\bar{v}^2\ell_1^4 + \bar{u}^4\ell_1^2\ell_2^2 + \bar{v}^4\ell_1^2\ell_2^2 + \bar{u}^2\bar{v}^2\ell_2^4 - \bar{u}^2\bar{v}^2\ell_1^4 + 2\bar{u}^2\bar{v}^2\ell_1^2\ell_2^2 - \bar{u}^2\bar{v}^2\ell_2^4 \\ &= \ell_1^2\ell_2^2(\bar{u}^4 + \bar{v}^4 + 2\bar{u}^2\bar{v}^2) = \ell_1^2\ell_2^2(\bar{u}^2 + \bar{v}^2)^2 \ . \end{aligned}$$

As we have

$$\bar{u}^2 + \bar{v}^2 = \frac{u^2}{l_{uv}^2} + \frac{v^2}{l_{uv}^2} = \frac{u^2 + v^2}{u^2 + v^2} = 1 \ ,$$

we get

$$|\Sigma| = \ell_1^2\ell_2^2 \ . \tag{A-7}$$

A.2 Identities

We present some basic identities which are used in the proofs of the following sections [Petersen and Pedersen, 2006]. A denotes a $n \times n$-matrix, whose entries are functions $A_{ij} = a_{ij}(x)$. $\mathbf{v}$ is a constant vector of size n. Let $|A|$ denote the determinant of A.

$$|rA| = r^n|A| \tag{A-8}$$

$$|A^{-1}| = |A|^{-1} \tag{A-9}$$

$$|A| = |A^T| \tag{A-10}$$

$$\frac{\partial A^{-1}}{\partial x} = -A^{-1}\frac{\partial A}{\partial x}A^{-1} \tag{A-11}$$

$$\frac{\partial |A|^{-\frac{1}{2}}}{\partial x} = -\frac{1}{2}|A|^{-\frac{3}{2}}\frac{\partial |A|}{\partial x} \tag{A-12}$$

$$\frac{\partial(\mathbf{v}^T A\mathbf{v})}{\partial x} = \mathbf{v}^T\frac{\partial A}{\partial x}\mathbf{v} \tag{A-13}$$

Proof of Equation (A-13):

$$\begin{aligned} \frac{\partial(\mathbf{v}^T A\mathbf{v})}{\partial x} &= \frac{\partial}{\partial x}\sum_{i=1}^{n}\sum_{j=1}^{n} v_i a_{ij}(x) v_j = \frac{\partial}{\partial x}\sum_{i=1}^{n}\sum_{j=1}^{n} v_i v_j a_{ij}(x) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} v_i v_j \frac{\partial}{\partial x} a_{ij}(x) = \mathbf{v}^T\frac{\partial A}{\partial x}\mathbf{v} \end{aligned}$$

A.3 Marginal Data-Likelihood

We prove Equation (4.3) which yields the derivative of the marginal data-likelihood.

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|X,\theta) &= \frac{1}{2}\mathbf{y}^T K^{-1}\frac{\partial K}{\partial\theta_j}K^{-1}\mathbf{y} - \frac{1}{2}\,\mathrm{tr}(K^{-1}\frac{\partial K}{\partial\theta_j}) && \text{(A-14)}\\
&= \frac{1}{2}\,\mathrm{tr}(K^{-1}\mathbf{y}\mathbf{y}^T K^{-T}\frac{\partial K}{\partial\theta_j}) - \frac{1}{2}\,\mathrm{tr}(K^{-1}\frac{\partial K}{\partial\theta_j})\\
&= \frac{1}{2}\,\mathrm{tr}(K^{-1}\mathbf{y}\mathbf{y}^T K^{-T}\frac{\partial K}{\partial\theta_j} - K^{-1}\frac{\partial K}{\partial\theta_j})\\
&= \frac{1}{2}\,\mathrm{tr}((\boldsymbol{\alpha}\boldsymbol{\alpha}^T - K^{-1})\frac{\partial K}{\partial\theta_j}) && \text{(A-15)}\\
&\quad \text{where } \boldsymbol{\alpha} = K^{-1}\mathbf{y}
\end{aligned}$$

This proof uses the identity $\mathbf{v}^T M\mathbf{v} = \mathrm{tr}(\mathbf{v}\mathbf{v}^T M)$, where $\mathbf{v} = K^{-1}\mathbf{y}$ and $M = \frac{\partial K}{\partial\theta_j}$. This equality holds as on the one side, we have:

$$\begin{aligned}
\mathbf{v}^T M\mathbf{v} &= \begin{pmatrix} v_1 & \dots & v_n \end{pmatrix}\begin{pmatrix} m_{11} & \dots & m_{1n}\\ \vdots & & \vdots\\ m_{n1} & \dots & m_{nn}\end{pmatrix}\begin{pmatrix} v_1\\ \vdots\\ v_n\end{pmatrix}\\
&= \begin{pmatrix} v_1m_{11}+\dots+v_nm_{n1} & \dots & v_1m_{1n}+\dots+v_nm_{nn}\end{pmatrix}\begin{pmatrix} v_1\\ \vdots\\ v_n\end{pmatrix}\\
&= (v_1m_{11}+\dots+v_nm_{n1})v_1+\dots+(v_1m_{1n}+\dots+v_nm_{nn})v_n
\end{aligned}$$

On the other side, we have:

$$\begin{aligned}
\mathbf{v}\mathbf{v}^T M &= \begin{pmatrix} v_1\\ \vdots\\ v_n\end{pmatrix}\begin{pmatrix} v_1 & \dots & v_n\end{pmatrix}\begin{pmatrix} m_{11} & \dots & m_{1n}\\ \vdots & & \vdots\\ m_{n1} & \dots & m_{nn}\end{pmatrix}\\
&= \begin{pmatrix} v_1^2 & v_1v_2 & \dots & v_1v_n\\ v_2v_1 & \vdots & \vdots & \vdots\\ v_nv_1 & \dots & \dots & v_n^2\end{pmatrix}\begin{pmatrix} m_{11} & \dots & m_{1n}\\ \vdots & & \vdots\\ m_{n1} & \dots & m_{nn}\end{pmatrix}\\
&= \begin{pmatrix} v_1^2m_{11}+v_1v_2m_{21}+\dots+v_1v_nm_{n1} & \dots & \dots\\ \vdots & \vdots & \vdots\\ \dots & \dots & v_nv_1m_{1n}+v_nv_2m_{2n}+\dots+v_n^2m_{nn}\end{pmatrix}\\
&= \begin{pmatrix} v_1(v_1m_{11}+v_2m_{21}+\dots+v_nm_{n1}) & \dots & \dots\\ \vdots & \vdots & \vdots\\ \dots & \dots & v_n(v_1m_{1n}+v_2m_{2n}+\dots+v_nm_{nn})\end{pmatrix}
\end{aligned}$$

Thus, we get

$$\mathrm{tr}(\mathbf{v}\mathbf{v}^T M) = (v_1m_{11}+v_2m_{21}+\dots+v_nm_{n1})v_1+\dots+(v_1m_{1n}+v_2m_{2n}+\dots+v_nm_{nn})v_n\ .$$

A.4 Derivative of the Covariance Function

To calculate the derivative of the covariance matrix K, we need to compute the derivatives of the covariance functions which are parameterized over the input locations. In other words, for input locations $\mathbf{x}_i$ and $\mathbf{x}_j$ we calculate the derivative of k_{NSE} with respect to the p-th parameter of the kernel Σ_i belonging to input location $\mathbf{x}_i$ (p thus denotes a lengthscale or the angle), namely

$$\left(\frac{\partial K}{\partial \theta_{ip}}\right)_{ij} = \frac{\partial k_{NSE}(\mathbf{x}_j, \mathbf{x}_j)}{\partial \theta_{ip}} \; . \tag{A-16}$$

To repeat, the covariance function is

$$\begin{aligned} k_{NSE}(\mathbf{x}_i, \mathbf{x}_j) &= |\Sigma_i|^{\frac{1}{4}} |\Sigma_j|^{\frac{1}{4}} \left|\frac{\Sigma_i + \Sigma_j}{2}\right|^{-\frac{1}{2}} \\ &\quad \exp\left[-(\mathbf{x}_i - \mathbf{x}_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} (\mathbf{x}_i - \mathbf{x}_j)\right] . \end{aligned} \tag{A-17}$$

To simplify notation, we define the variables

$$\begin{aligned} a_i &:= |\Sigma_i|^{\frac{1}{4}} |\Sigma_j|^{\frac{1}{4}} \; , \\ b_i &:= \left|\frac{\Sigma_i + \Sigma_j}{2}\right|^{-\frac{1}{2}} \; , \text{ and} \\ c_i &:= \exp\left[-(\mathbf{x}_i - \mathbf{x}_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} (\mathbf{x}_i - \mathbf{x}_j)\right] \; , \end{aligned}$$

so that we get

$$k_{NSE}(\mathbf{x}_i, \mathbf{x}_j) = a_i \cdot b_i \cdot c_i \; .$$

By means of the product rule, we get the derivative of the covariance function

$$\frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta_{ip}} = \frac{\partial (a_i \cdot b_i \cdot c_i)}{\partial \theta_{ip}} \tag{A-18}$$

$$= \frac{\partial a_i}{\partial \theta_{ip}} b_i c_i + a_i \frac{\partial b_i}{\partial \theta_{ip}} c_i + a_i b_i \frac{\partial c_i}{\partial \theta_{ip}} \; . \tag{A-19}$$

The derivative of a_i is

$$\frac{\partial a_i}{\partial \theta_{ip}} = \frac{\partial (|\Sigma_i|^{\frac{1}{4}} |\Sigma_j|^{\frac{1}{4}})}{\partial \theta_{ip}} = |\Sigma_j|^{\frac{1}{4}} \frac{\partial (\ell_{i1}^2 \ell_{i2}^2)^{\frac{1}{4}}}{\partial \theta_{ip}} = |\Sigma_j|^{\frac{1}{4}} \frac{\partial (\ell_{i1}^{\frac{1}{2}} \ell_{i2}^{\frac{1}{2}})}{\partial \theta_{ip}} \tag{A-20}$$

which is 0 for the angle parameters and for the lengthscales it is

$$|\Sigma_j|^{\frac{1}{4}} \frac{\partial (\ell_{i1}^{\frac{1}{2}} \ell_{i2}^{\frac{1}{2}})}{\partial \ell_{i1}} = |\Sigma_j|^{\frac{1}{4}} \sqrt{\ell_{i2}} \frac{1}{2} \frac{1}{\sqrt{\ell_{i1}}} \tag{A-21}$$

$$|\Sigma_j|^{\frac{1}{4}} \frac{\partial (\ell_{i1}^{\frac{1}{2}} \ell_{i2}^{\frac{1}{2}})}{\partial \ell_{i2}} = |\Sigma_j|^{\frac{1}{4}} \sqrt{\ell_{i1}} \frac{1}{2} \frac{1}{\sqrt{\ell_{i2}}} \; . \tag{A-22}$$

The derivative of b_i is

$$\begin{aligned}
\frac{\partial b_i}{\partial \theta_{ip}} &= \frac{\partial \left|\frac{\Sigma_i+\Sigma_j}{2}\right|^{-\frac{1}{2}}}{\partial \theta_{ip}} = \frac{\partial((\frac{1}{2}^2)\,|\Sigma_i+\Sigma_j|)^{-\frac{1}{2}}}{\partial \theta_{ip}} && \text{(A-23)}\\
&= -2\cdot\frac{1}{2}|\Sigma_i+\Sigma_j|^{-\frac{3}{2}}\frac{\partial\,|\Sigma_i+\Sigma_j|}{\partial \theta_{ip}}\\
&= -|\Sigma_i+\Sigma_j|^{-\frac{3}{2}}|\Sigma_i+\Sigma_j|\ \mathrm{tr}((\Sigma_i+\Sigma_j)^{-1}\frac{\partial\,(\Sigma_i+\Sigma_j)}{\partial \theta_{ip}})\\
&= -|\Sigma_i+\Sigma_j|^{-\frac{1}{2}}\ \mathrm{tr}((\Sigma_i+\Sigma_j)^{-1}\frac{\partial\,(\Sigma_i+\Sigma_j)}{\partial \theta_{ip}}) && \text{(A-24)}
\end{aligned}$$

and the derivative of c_i is

$$\frac{\partial c_i}{\partial \theta_{ip}} = \frac{\partial \exp\left[-(\mathbf{x}_i-\mathbf{x}_j)^T\left(\frac{\Sigma_i+\Sigma_j}{2}\right)^{-1}(\mathbf{x}_i-\mathbf{x}_j)\right]}{\partial \theta_{ip}} \quad \text{(A-25)}$$

$$= \exp\left[-(\mathbf{x}_i-\mathbf{x}_j)^T\left(\frac{\Sigma_i+\Sigma_j}{2}\right)^{-1}(\mathbf{x}_i-\mathbf{x}_j)\right]\frac{\partial\left[-(\mathbf{x}_i-\mathbf{x}_j)^T\left(\frac{\Sigma_i+\Sigma_j}{2}\right)^{-1}(\mathbf{x}_i-\mathbf{x}_j)\right]}{\partial \theta_{ip}} \quad \text{(A-26)}$$

where

$$\begin{aligned}
\frac{\partial\left[-(\mathbf{x}_i-\mathbf{x}_j)^T\left(\frac{\Sigma_i+\Sigma_j}{2}\right)^{-1}(\mathbf{x}_i-\mathbf{x}_j)\right]}{\partial \theta_{ip}} &= -(\mathbf{x}_i-\mathbf{x}_j)^T\frac{\partial\left(\frac{\Sigma_i+\Sigma_j}{2}\right)^{-1}}{\partial \theta_{ip}}(\mathbf{x}_i-\mathbf{x}_j) && \text{(A-27)}\\
&= -(\mathbf{x}_i-\mathbf{x}_j)^T\cdot 2\cdot\frac{\partial\,(\Sigma_i+\Sigma_j)^{-1}}{\partial \theta_{ip}}(\mathbf{x}_i-\mathbf{x}_j)\\
&= 2(\mathbf{x}_i-\mathbf{x}_j)^T(-(\Sigma_i+\Sigma_j)^{-1}\frac{\partial\,(\Sigma_i+\Sigma_j)}{\partial \theta_{ip}}(\Sigma_i+\Sigma_j)^{-1})(\mathbf{x}_i-\mathbf{x}_j)\ . && \text{(A-28)}
\end{aligned}$$

Thus, in order to calculate the derivatives of b_i and c_i, we need to calculate the derivative of the sum of two kernels

$$\frac{\partial\,(\Sigma_i+\Sigma_j)}{\partial \theta_{ip}}\ .$$

The derivation of this term depends on the parameterization of the kernel, either by using the standard rotational matrix with angle α or by using the overparameterized rotational matrix with parameters u and v.

Standard parameterization

Using the standard parameterization (cf. section A.1.1), the sum of two individual kernel matrices is (where we use the shorthands $s_k = \sin\alpha_k$ and $c_k = \cos\alpha_k$ for $k = 1, 2$)

$$\Sigma_i + \Sigma_j = \begin{pmatrix} c_i^2\ell_{i1}^2 + s_i^2\ell_{i2}^2 & c_i s_i \ell_{i1}^2 - s_i \ell_{i2}^2 c_i \\ s_i c_i \ell_{i1}^2 - c_i \ell_{i2}^2 s_i & s_i^2\ell_{i1}^2 + c_i^2\ell_{i2}^2 \end{pmatrix} \tag{A-29}$$

$$+ \begin{pmatrix} c_j^2\ell_{j1}^2 + s_j^2\ell_{j2}^2 & c_j s_j \ell_{j1}^2 - s_j \ell_{j2}^2 c_j \\ s_j c_j \ell_{j1}^2 - c_j \ell_{j2}^2 s_j & s_j^2\ell_{j1}^2 + c_j^2\ell_{j2}^2 \end{pmatrix}$$

$$= \begin{pmatrix} c_i^2\ell_{i1}^2 + s_i^2\ell_{i2}^2 + c_j^2\ell_{j1}^2 + s_j^2\ell_{j2}^2 & c_i s_i \ell_{i1}^2 - s_i c_i \ell_{i2}^2 + c_j s_j \ell_{j1}^2 - s_j c_j \ell_{j2}^2 \\ s_i c_i \ell_{i1}^2 - c_i s_i \ell_{i2}^2 + s_j c_j \ell_{j1}^2 - c_j s_j \ell_{j2}^2 & s_i^2\ell_{i1}^2 + c_i^2\ell_{i2}^2 + s_j^2\ell_{j1}^2 + c_j^2\ell_{j2}^2 \end{pmatrix} \tag{A-30}$$

Now, we can calulate the derivative of the sum of two kernel matrices with respect to the different kernel parameters θ_{ip}. For $\theta_{ip} = l_{i1}$, the resulting matrix is

$$\frac{\partial(\Sigma_i + \Sigma_j)}{\partial \ell_{i1}} = \begin{pmatrix} 2 \cdot c_i^2 \ell_{i1} & 2 \cdot c_i s_i \ell_{i1} \\ 2 \cdot s_i c_i \ell_{i1} & 2 \cdot s_i^2 \ell_{i1} \end{pmatrix} , \tag{A-31}$$

while for $\theta_{ip} = l_{i2}$ we get

$$\frac{\partial(\Sigma_i + \Sigma_j)}{\partial \ell_{i2}} = \begin{pmatrix} 2s_i^2 \ell_{i2} & -2 s_i c_i \ell_{i2} \\ -2 c_i s_i \ell_{i2} & 2 c_i^2 \ell_{i2} \end{pmatrix} . \tag{A-32}$$

For $\theta_{ip} = \alpha_i$ the matrix entries are

$$\frac{\partial(c_i^2\ell_{i1}^2 + s_i^2\ell_{i2}^2 + c_j^2\ell_{j1}^2 + s_j^2\ell_{j2}^2)}{\partial \alpha_i} = 2 \cdot c_i(-s_i)\ell_{i1}^2 + 2 \cdot s_i c_i \ell_{i2}^2 = 2 s_i c_i(-\ell_{i1}^2 + \ell_{i2}^2) , \tag{A-33}$$

$$\frac{\partial(c_i s_i \ell_{i1}^2 - s_i c_i \ell_{i2}^2 + c_j s_j \ell_{j1}^2 - s_j c_j \ell_{j2}^2)}{\partial \alpha} \tag{A-34}$$

$$= \ell_{i1}^2(c_i c_i - s_i s_i) - \ell_{i2}^2(c_i c_i - s_i s_i) = (\ell_{i1}^2 - \ell_{i2}^2)(c_i^2 - s_i^2) , \quad \text{and} \tag{A-35}$$

$$\frac{\partial(s_i^2\ell_{i1}^2 + c_i^2\ell_{i2}^2 + s_j^2\ell_{j1}^2 + c_j^2\ell_{j2}^2)}{\partial \alpha} \tag{A-36}$$

$$= 2 \cdot s_i c_i \ell_{i1}^2 + 2 \cdot c_i(-s_i)\ell_{i2}^2 = 2 s_i c_i(\ell_{i1}^2 - \ell_{i2}^2) , \tag{A-37}$$

so that we get

$$\frac{\partial(\Sigma_i + \Sigma_j)}{\partial \alpha_i} = \begin{pmatrix} 2 \sin\alpha_i \cos\alpha_i(-\ell_{i1}^2 + \ell_{i2}^2) & (\ell_{i1}^2 - \ell_{i2}^2)(\cos^2\alpha_i - \sin^2\alpha_i) \\ (\ell_{i1}^2 - \ell_{i2}^2)(\cos^2\alpha_i - \sin^2\alpha_i) & 2 \sin\alpha_i \cos\alpha_i(\ell_{i1}^2 - \ell_{i2}^2) \end{pmatrix} . \tag{A-38}$$

Overparameterization

When using an overparameterized rotational matrix (cf. section A.1.2), the sum of two kernels takes a different form. Let $l_{u_k v_k} = \sqrt{u_k^2 + v_k^2}$, $\overline{u_k} = \frac{u_k}{l_{u_k v_k}}$ and $\overline{v_k} = \frac{v_k}{l_{u_k v_k}}$.

$$\Sigma_i + \Sigma_j = \begin{pmatrix} \left(\frac{u_i}{l_{u_i v_i}}\right)^2 \ell_{i1}^2 + \left(\frac{v_i}{l_{u_i v_i}}\right)^2 \ell_{i2}^2 & \frac{u_i}{l_{u_i v_i}}\ell_{i1}^2\frac{v_i}{l_{u_i v_i}} - \frac{u_i}{l_{u_i v_i}}\ell_{i2}^2\frac{v_i}{l_{u_i v_i}} \\ \frac{u_i}{l_{u_i v_i}}\ell_{i1}^2\frac{v_i}{l_{u_i v_i}} - \frac{u_i}{l_{u_i v_i}}\ell_{i2}^2\frac{v_i}{l_{u_i v_i}} & \left(\frac{v_i}{l_{u_i v_i}}\right)^2 \ell_{i1}^2 + \left(\frac{u_i}{l_{u_i v_i}}\right)^2 \ell_{i2}^2 \end{pmatrix} \tag{A-39}$$

$$+ \begin{pmatrix} \left(\frac{u_j}{l_{u_j v_j}}\right)^2 \ell_{j1}^2 + \left(\frac{v_j}{l_{u_j v_j}}\right)^2 \ell_{j2}^2 & \frac{u_j}{l_{u_j v_j}}\ell_{j1}^2\frac{v_j}{l_{u_j v_j}} - \frac{u_j}{l_{u_j v_j}}\ell_{j2}^2\frac{b}{l_{u_j v_j}} \\ \frac{u_j}{l_{u_j v_j}}\ell_{j1}^2\frac{v_j}{l_{u_j v_j}} - \frac{u_j}{l_{u_j v_j}}\ell_{j2}^2\frac{v_j}{l_{u_j v_j}} & \left(\frac{v_j}{l_{u_j v_j}}\right)^2 \ell_{j1}^2 + \left(\frac{u_j}{l_{u_j v_j}}\right)^2 \ell_{j2}^2 \end{pmatrix}$$

$$= \begin{pmatrix} \overline{u_i}^2\ell_{i1}^2 + \overline{v_i}^2\ell_{i2}^2 + \overline{u_j}^2\ell_{j1}^2 + \overline{v_j}^2\ell_{j2}^2 & \overline{u_i}\ell_{i1}^2\overline{v_i} - \overline{u_i}\ell_{i2}^2\overline{v_i} + \overline{u_j}\ell_{j1}^2\overline{v_j} - \overline{u_j}\ell_{j2}^2\overline{v_j} \\ \overline{u_i}\ell_{i1}^2\overline{v_i} - \overline{u_i}\ell_{i2}^2\overline{v_i} + \overline{u_j}\ell_{j1}^2\overline{v_j} - \overline{u_j}\ell_{j2}^2\overline{v_j} & \overline{v_i}^2\ell_{i1}^2 + \overline{u_i}^2\ell_{i2}^2 + \overline{v_j}^2\ell_{j1}^2 + \overline{u_j}^2\ell_{j2}^2 \end{pmatrix} \tag{A-40}$$

Now, we can calulate the derivative of the sum of two kernel matrices with respect to the different kernel parameters θ_{ip}. For $\theta_{ip} = l_{i1}$, the resulting matrix is

$$\frac{\partial(\Sigma_i + \Sigma_j)}{\partial \ell_{i1}} = \begin{pmatrix} 2 \cdot \overline{u_i}^2 \ell_{i1} & 2 \cdot \overline{u_i v_i} \ell_{i1} \\ 2 \cdot \overline{u_i v_i} \ell_{i1} & 2 \cdot \overline{v_i}^2 \ell_{i1} \end{pmatrix}, \tag{A-41}$$

while for $\theta_{ip} = l_{i2}$, we get

$$\frac{\partial(\Sigma_i + \Sigma_j)}{\partial \ell_{i2}} = \begin{pmatrix} 2 \cdot \overline{v_i}^2 \ell_{i2} & -2 \cdot \overline{u_i v_i} \ell_{i2} \\ -2 \cdot \overline{u_i v_i} \ell_{i2} & 2 \cdot \overline{u_i}^2 \ell_{i2} \end{pmatrix}. \tag{A-42}$$

For $\theta_{ip} = u_i$, the derivation is more complicated:

$$\frac{\partial(\Sigma_i + \Sigma_j)}{\partial u_i} = \begin{pmatrix} l_1^2 \frac{\partial \overline{u_i}^2}{\partial u_i} + l_2^2 \frac{\partial \overline{v_i}^2}{\partial u_i} & l_1^2 \frac{\partial \overline{u_i v_i}}{\partial u_i} - l_2^2 \frac{\partial \overline{u_i v_i}}{\partial u_i} \\ l_1^2 \frac{\partial \overline{u_i v_i}}{\partial u_i} - l_2^2 \frac{\partial \overline{u_i v_i}}{\partial u_i} & l_1^2 \frac{\partial \overline{v_i}^2}{\partial u_i} + l_2^2 \frac{\partial \overline{u_i}^2}{\partial u_i} \end{pmatrix} \tag{A-43}$$

To compute this matrix, we derive the following derivatives:

$$\frac{\partial \overline{u_i}}{\partial u_i} = \frac{\partial \frac{u_i}{\sqrt{u_i^2+v_i^2}}}{\partial u_i} = \frac{\frac{\partial u_i}{\partial u_i}\sqrt{u_i^2+v_i^2} - u_i \frac{\partial \sqrt{u_i^2+v_i^2}}{\partial u_i}}{u_i^2+v_i^2} \tag{A-44}$$

$$= \frac{\sqrt{u_i^2+v_i^2} - u_i \frac{1}{2}(u_i^2+v_i^2)^{-\frac{1}{2}} 2u_i}{u_i^2+v_i^2} = \frac{\sqrt{u_i^2+v_i^2} - u_i^2(u_i^2+v_i^2)^{-\frac{1}{2}}}{u_i^2+v_i^2} \tag{A-45}$$

$$\frac{\partial \overline{v_i}}{\partial u_i} = \frac{\partial \frac{v_i}{\sqrt{u_i^2+v_i^2}}}{\partial u_i} = v_i \frac{\partial (u_i^2+v_i^2)^{-\frac{1}{2}}}{\partial u_i} \tag{A-46}$$

$$= v_i(-\frac{1}{2})(u_i^2+v_i^2)^{-\frac{3}{2}} 2u_i = -v_i u_i (u_i^2+v_i^2)^{-\frac{3}{2}} \tag{A-47}$$

$$\frac{\partial \overline{u_i}^2}{\partial u_i} = 2\overline{u_i}\frac{\partial \overline{u_i}}{\partial u_i} \tag{A-48}$$

$$\frac{\partial \overline{v_i}^2}{\partial u_i} = 2\overline{v_i}\frac{\partial \overline{v_i}}{\partial u_i} \tag{A-49}$$

$$\frac{\partial \overline{u_i v_i}}{\partial u_i} = \overline{u_i}\frac{\partial \overline{v_i}}{\partial u_i} + \overline{v_i}\frac{\partial \overline{u_i}}{\partial u_i} \tag{A-50}$$

For $\theta_{ip} = v_i$, the derivation is analogous:

$$\frac{\partial(\Sigma_i + \Sigma_j)}{\partial v_i} = \begin{pmatrix} \ell_{i1}^2 \frac{\partial \overline{u_i}^2}{\partial v_i} + \ell_{i2}^2 \frac{\partial \overline{v_i}^2}{\partial v_i} & \ell_{i1}^2 \frac{\partial \overline{u_i v_i}}{\partial v_i} - \ell_{i2}^2 \frac{\partial \overline{u_i v_i}}{\partial v_i} \\ \ell_{i1}^2 \frac{\partial \overline{u_i v_i}}{\partial v_i} - \ell_{i2}^2 \frac{\partial \overline{u_i v_i}}{\partial v_i} & \ell_{i1}^2 \frac{\partial \overline{v_i}^2}{\partial v_i} + \ell_{i2}^2 \frac{\partial \overline{u_i}^2}{\partial v_i} \end{pmatrix} \tag{A-51}$$

To compute this matrix, we derive the following derivatives:

$$\frac{\partial \overline{v_i}}{\partial v_i} = \frac{\partial \frac{v_i}{\sqrt{u_i^2+v_i^2}}}{\partial v_i} = \frac{\frac{\partial v_i}{\partial v_i}\sqrt{u_i^2+v_i^2} - v_i \frac{\partial \sqrt{u_i^2+v_i^2}}{\partial v_i}}{u_i^2+v_i^2} \tag{A-52}$$

$$= \frac{\sqrt{u_i^2+v_i^2} - v_i \frac{1}{2}(u_i^2+v_i^2)^{-\frac{1}{2}} 2v_i}{u_i^2+v_i^2} = \frac{\sqrt{u_i^2+v_i^2} - v_i^2(u_i^2+v_i^2)^{-\frac{1}{2}}}{u_i^2+v_i^2} \tag{A-53}$$

$$\frac{\partial \overline{u_i}}{\partial v_i} = \frac{\partial \frac{u}{\sqrt{u_i^2+v_i^2}}}{\partial v_i} = u_i \frac{\partial (u_i^2+v_i^2)^{-\frac{1}{2}}}{\partial v_i} \tag{A-54}$$

$$= u_i(-\frac{1}{2})(u_i^2+v_i^2)^{-\frac{3}{2}}2v_i = -u_i v_i (u_i^2+v_i^2)^{-\frac{3}{2}} \tag{A-55}$$

$$\frac{\partial \overline{v_i}^2}{\partial v_i} = 2\overline{v_i}\frac{\partial \overline{v_i}}{\partial v_i} \tag{A-56}$$

$$\frac{\partial \overline{u_i}^2}{\partial v_i} = 2\overline{u_i}\frac{\partial \overline{u_i}}{\partial v_i} \tag{A-57}$$

$$\frac{\partial \overline{u_i v_i}}{\partial v_i} = \overline{u_i}\frac{\partial \overline{v_i}}{\partial v_i} + \overline{v_i}\frac{\partial \overline{u_i}}{\partial v_i} \tag{A-58}$$

A.4.1 Derivative of the Regularization Term

We want to determine the derivative of the regularization term

$$\frac{\partial}{\partial \theta_{ip}} \lambda_{\theta_p} \sum_k \sum_l \frac{\parallel \theta_{kp} - \theta_{lp} \parallel}{\parallel x_k - x_l \parallel} . \tag{A-59}$$

We have

$$\frac{\partial \parallel \theta_{ip} - \theta_{lp} \parallel}{\partial \theta_{ip}} = \frac{\partial \sqrt{(\theta_{ip}-\theta_{lp})^2}}{\partial \theta_{ip}} \tag{A-60}$$

$$= \frac{1}{2}\frac{1}{\sqrt{(\theta_{ip}-\theta_{lp})^2}} \cdot 2 \cdot (\theta_{ip}-\theta_{lp})$$

$$= \frac{(\theta_{ip}-\theta_{lp})}{\sqrt{(\theta_{ip}-\theta_{lp})^2}} \tag{A-61}$$

and

$$\frac{\partial \parallel \theta_{kp} - \theta_{ip} \parallel}{\partial \theta_{ip}} = \frac{\partial \sqrt{(\theta_{kp}-\theta_{ip})^2}}{\partial \theta_{ip}} \tag{A-62}$$

$$= -\frac{1}{2}\frac{1}{\sqrt{(\theta_{kp}-\theta_{ip})^2}} \cdot 2 \cdot (\theta_{kp}-\theta_{ip})$$

$$= \frac{(\theta_{ip}-\theta_{kp})}{\sqrt{(\theta_{ip}-\theta_{kp})^2}} \tag{A-63}$$

so that in cases $i = k$ and $i = l$ the derivative is zero and otherwise we have $\frac{\partial \|\theta_{ip}-\theta_{kp}\|}{\partial \theta_{ip}} = \frac{\partial \|\theta_{kp}-\theta_{ip}\|}{\partial \theta_{ip}}$. Therefore, the derivative has the form

$$\frac{\partial}{\partial \theta_{ip}} \lambda_{\theta_p} \sum_k \sum_l \frac{\parallel \theta_{kp} - \theta_{lp} \parallel}{\parallel x_k - x_l \parallel} = 2\lambda_\theta \sum_l \frac{\partial}{\partial \theta_i} \frac{\parallel \theta_{ip} - \theta_{lp} \parallel}{\parallel x_i - x_l \parallel} \tag{A-64}$$

$$= 2\lambda_\theta \sum_l \frac{1}{\parallel x_i - x_l \parallel} \frac{(\theta_{ip}-\theta_{lp})}{\sqrt{(\theta_{ip}-\theta_{lp})^2}} . \tag{A-65}$$

Bibliography

[Aronszajn, 1950] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68, 1950.

[Bares *et al.*, 1989] John Bares, Martial Hebert, Takeo Kanade, Eric Krotkov, Tom Mitchell, Reid Simmons, and William Red L. Whittaker. Ambler: An autonomous rover for planetary exploration. *IEEE Computer*, 22(6):18–26, June 1989.

[Besag, 1977] Julian Besag. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrika*, 64(3):616–618, 1977.

[Blackwell and Moller, 2003] P. G. Blackwell and J. Moller. Bayesian inference for deformed tessellation models. *Advances in Applied Probability*, 35:4–26, 2003.

[Brooks *et al.*, 2006] A. Brooks, A. Makarenko, and B. Upcroft. Gaussian process models for sensor-centric robot localisation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2006.

[Chu *et al.*, 2005] W. Chu, Z. Ghahramani, F. Falciani, and D. Wild. Biomarker discovery in microarray gene expression data with Gaussian processes. *Bioinformatics*, 21(16):3385–3393, 2005.

[Cornford *et al.*, 1999] D. Cornford, I. Nabney, and C. Williams. Adding constrained discontinuities to Gaussian process models of wind fields. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, volume 11. MIT Press, Cambridge, MA, 1999.

[Denison *et al.*, 2002] D. G. T. Denison, N. M. Adams, C. C. Holmes, and D. J. Hand. Bayesian partition modelling. *Comput. Stat. Data Anal.*, 38(4):475–485, 2002.

[Früh *et al.*, 2005] Christian Früh, Siddharth Jain, and Avideh Zakhor. Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *International Journal of Computer Vision*, 61(2):159–184, 2005.

[Guestrin *et al.*, 2005] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in Gaussian processes. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 265–272, New York, NY, USA, 2005. ACM Press.

[Higdon *et al.*, 1999] D. Higdon, J. Swall, and J. Kern. Non-stationary spatial modeling. *Bayesian Statistics*, 6:761–768, 1999.

[Hoerl, 1962] A.E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58:54–59, 1962.

[Jacobs *et al.*, 1991] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[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.

[MacKay, 1998] D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, NATO ASI Series, pages 133–166. Kluwer Academic Press, 1998.

[Matheron, 1963] G. Matheron. Principles of geostatistics. *Economic Geology*, 58(8):Principles of geostatistics, 1963.

[Meeds and Osindero, 2006] E. Meeds and S. Osindero. An alternative infinite mixture of Gaussian process experts. In Y. Weiss, B. Schoelkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA, 2006.

[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.

[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.

[Paciorek, 2003] C. Paciorek. *Nonstationary Gaussian Processes for Regression and Spatial Modelling*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2003.

[Petersen and Pedersen, 2006] K. B. Petersen and M. S. Pedersen. The matrix cookbook, 2006. Version 2005-10-03.

[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.

[Plagemann *et al.*, 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.

[Platt *et al.*, 2000] J. Platt, C. Burges, S. Swenson, C. Weare, and A. Zheng. Learning a Gaussian process prior for automatically generating music playlists. In S. Solla, T. Leen, and K.-R. Mueller, editors, *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, volume 12. MIT Press, Cambridge, MA, 2000.

[Rasmussen and Ghahramani, 2002] C. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In T. Diettrich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge, MA, 2002.

[Rasmussen and Williams, 2006] C. E. Rasmussen and C. K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, Massachusetts, 2006.

[Riedmiller and Braun, 1993] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE Intl. Conf. on Neural Networks*, pages 586–591, San Francisco, CA, 1993.

[Sampson and Guttorp, 1992] Paul D. Sampson and Peter Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87(417):108–119, 1992.

[Schmidt and O'Hagan, 2003] Alexandra M. Schmidt and Anthony O'Hagan. Bayesian inference for non-stationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society, Series B*, 65:745–758, 2003.

[Schwaighofer *et al.*, 2004] A. Schwaighofer, M. Grigoras, V. Tresp, and C. Hoffmann. A Gaussian process positioning system for cellular networks. In S. Thrun, L. Saul, and B. Schoelkopf, editors, *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, volume 16. MIT Press, Cambridge, MA, 2004.

[Schwaighofer *et al.*, 2005] A. Schwaighofer, V. Tresp, and K. Yu. Learning Gaussian process kernels via hierarchical bayes. In L. Saul, Y. Weiss, and L. Bottou, editors, *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, volume 17. MIT Press, Cambridge, MA, 2005.

[Stephenson *et al.*, 2005] J. Stephenson, C. Holmes, K. Gallagher, and A. Pintore. A statistical technique for modelling non-stationary spatial processes. In O. Leuangthong and V.C. Deutsch, editors, *7th International Geostatics Congress*, pages 125–134. 2005.

[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.*, 2005] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[Tikhonov, 1943] A.N. Tikhonov. On the stability of inverse problems. *Dokl. Akad. Nauk SSSR*, 5:195–198, 1943.

[Tresp, 2000] V. Tresp. Mixtures of Gaussian processes. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 654–660, 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.

[Wellington *et al.*, 2005] Carl Wellington, Aaron Courville, and Anthony Stentz. Interacting markov random fields for simultaneous terrain modeling and obstacle detection. In *Proceedings of Robotics Science and Systems*, pages 1–8, June 2005.

[Williams, 2006] O. Williams. A switched Gaussian process for estimating disparity and segmentation in binocular stereo. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2006.

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung verwendet wurde.

Tobias Johannes Lang
Freiburg, den 21. Mai 2007