Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Gaussian Processes for Uncertainty Visualization

Masterarbeit

Leipzig, September 2011 vorgelegt von
Korn, Nico
Studiengang Informatik

Betreuender Hochschullehrer: Gerik Scheuermann, Abteilung für Bild- und Signalverarbeitung

Contents

Chapter 1

Introduction

The goal of visualization is to generate insight. By taking advantage of the visual capabilities of the brain, we can create images or videos that help us find and interpret phenomena that may be hard to understand otherwise. For this reason, good visualizations are becoming ubiquitous in not only medicine, engineering and science in general, but in everyday life too. In recent years we have seen more and more professional visualizations in the media of for example elections, national budgets, open-government data and statistical information in general.

One crucial piece of information however is almost always missing—even in academic visualizations: The uncertainty of the data.

Data is virtually always uncertain in one way or another. If they come from measurement, the sensor measuring it will have limited accuracy or maybe even suffer from outright errors. If the data comes from a simulation, it might depend on some initial values that aren't known or are even random. The different simulation methods may produce different outputs. Lastly, one kind of uncertainty that digital data will almost always suffer from are quantization errors.

Yet, uncertainty information is not routinely included in visualizations. Despite it being a known problem in the visualization community for at least the past two decades.

A common counter example are publications about clinical trials. No two humans are the same, so there is rarely a single truth that holds for all patients. So it is not surprising, that the variance within a group of patients, the uncertainty about how the patient's body will react, plays the key role in determining whether a new treatment is actually significantly better—or if the result of the study could just as well be mere coincidence. For this reason, awareness to proper visualization is

very high in that field.

But still, there are fields in medicine like medical imaging where uncertainty is not explicitly displayed. It is left to the viewer to make an educated guess about the uncertainty of the data, even in situations where it is vital to know the reliability of the visualization to make an informed decision about, for instance, whether a lump might be cancerous and how big it is.

A big issue for uncertainty visualization is, that outside of simple 1D diagrams, there is no established way to do it. The reason for this is likely to be that it is hard to find a method that shows how big the uncertainty is without completely cluttering the display.

Another important question that needs to be solved, is how uncertainty and interpolation interact. Interpolated values are inherently uncertain, because they are heuristically estimated values—not measurements. How much more uncertain are they? How can this effect be modeled?

These and related problems are the main topics of this thesis, which is divided into four parts. First, we will give a survey of the central problems in uncertainty visualization and existing approaches. Chapter 3 introduces the basics of Gaussian processes—a statistical framework that allows for modeling and interpolation of data and, more importantly, its uncertainty! In chapter 4 we will then take a closer look at the practical implementation of Gaussian processes, their sometimes surprising implications and how they can be used for uncertainty visualization. Finally, we will look at possible visualization methods to render the interpolated uncertainty the Gaussian processes produce.

Chapter 2

Visualization of Uncertain Data

There is some amount of uncertainty in almost all kinds of data. It comes in different forms and may be large or small, but it is always important to know about whether there is uncertainty and how large it is. Otherwise, we risk making decisions that rely on data we should not rely upon or doubting data that we think might be uncertain when it is not.

[LLPY07] give a good example for the importance of proper uncertainty visualizaion. A CT seemed to show a stenosis[1], so the patient subsequently underwent surgery. Returning to the CT imagery, it turned out that there was no stenosis at all and the fault was in the visualization. It had been created with a too-low threshold in the transfer function used to classify the tissue as a blood vessel.

To avoid poor decisions like that one or at least inform the viewer about the risks, uncertainty needs to be integrated in visualizations. While it is already commonplace in the display of statistical data, in particular in the field of clinical trials, where misinterpreted results can cost lives, uncertainty visualization in general has become an important issue in visualization only in the last two decades.

In this chapter, we will look at three main topics: Sources of uncertainty, ways of visualizing uncertainty and the modeling of uncertainty.

2.1 Sources of Uncertainty

There are a variety of sources for uncertainty in data. They can be categorized into one of the four categories below[PWL97]. Other popular names for uncertainty are

[1] A decrease in diameter of a blood vessel. It can lead to an infarction of the affected organ.

error and noise, which carry a negative connotation. We will use the more neutral and common *uncertainty*, which is also less prone to confusion.

1. Data acquisition.
2. Algorithms, derivations, transformations.
3. Visualization.
4. Things that cannot be known.[Cou03]

If the subject to be studied is a natural phenomenon, data acquisition usually cannot achieve complete knowledge—there will always be some uncertainty left simply due to our inability to sample the world at infinitely many points with infinite accuracy. There will always be error bars on the data, the data might have missing values—maybe even the position at which the data was acquired is not known exactly. Human error and random natural effects further add to the uncertainty. Lastly, the method of acquisition may already include some processing in order to reduce noise or otherwise improve data quality, thereby hiding some information about the phenomenon in question.

The second kind of uncertainty is uncertainty introduced by the post-processing of the data. For example medical imagery can suffer from a number of artifacts like radial lines or halos that are caused by the image reconstruction algorithm. Side effects of discretization include rounding errors, approximations and numerical problems. A typical example from signal processing occurs when a signal is not sampled frequently enough, so that frequencies higher than the Nyquist frequency show up as low frequencies in the sampled function (so called "aliasing"). Transformation and algorithms generally will change the data in some way, introducing at least rounding errors.

Uncertainty introduced at the above stages can be particularly dangerous, because erroneous data need not look differently from high-quality data. What looks like an artifact might as well be an interesting pattern. A white spot might be a new star, or maybe it was just random radiation disturbing the sensor.

The third kind of uncertainty is the uncertainty that we introduce through our visualization. It is often a necessary step to compress or even leave out information because computer screens have limited resolution and color depth and only produce a two-dimensional image. Also, the visualization will frequently be abstract or approximate in order to convey information that is in the data in a more intuitive way, e.g. using stream lines. [PWL97] emphasize that those and other approximations can lead to undesirable consequences, such as two programs producing different results even when they use the same basic algorithm. They may

have used different interpolation schemes for example, or different ODE[2] solvers.

Things that cannot be known include limits imposed by fundamental laws of nature and logic. Heisenberg's uncertainty for example directly limits how much we can know. A more practically relevant example is the resolution of optical imaging systems that are limited by the wave-properties of light (and other electromagnetic waves) described by the Rayleigh- or Abbe criterion[3]. Other such problems include functions that are not computable, do not have an analytic solution or for which there is probably no practical algorithm because it is in NP or worse.

Luckily, in most scientific fields, people have learned to identify and deal with errors, artifacts, noise and other kinds of uncertainty. Yet, the example from the beginning shows, that even experts can fail to notice, that the visualization they are looking at shows only one of, quite possibly, a range of possible truths.

2.1.1 Positional Uncertainty

It should be kept in mind that the uncertainty can be in the data value or in the position of the data point or both (Further information about these and more can be found in [PWL97]).

The ideal would be to show the two kinds of uncertainty using two orthogonal visual variables or at least not confuse the two *too* much. For example, a vibrating glyph on a 2D-surface should not vibrate along the two directions that define its position but rather only in the direction orthogonal to the two axes. Another example are fat lines in a 2D plot with the x-axis being the position and the y-axis the value. Are they fat because they cover a range of values (spread in the y-direction)—or are they fat because the position of each value is unclear (spread in the x-direction)? Both are possible and it will be hard to tell, which one it is.

It will usually be clear from the context, what kind of uncertainty is being visualized. But if it is not, we must remember to tell the user beforehand.

2.1.2 Bounded vs Unbounded Uncertainty

With uncertainty being a random effect, the normal distribution may immediately spring to mind as the prototypical distribution. However, some kinds of uncertainty

[2]Ordinary Differential Equation

[3]Though lately "super resolution" techniques like near-field microscopy have been developed to work around this theoretical barrier.

are actually bounded. For example because the data values cannot be negative in the first place or the uncertainty itself is uniformly distributed. In the latter case, it is important not to display a mean since it suggests a most likely or expected value, which may not be sensible if the true value can be any value in a certain range, e.g. when the uncertainty is dominated by rounding errors[OM02].

2.2 Visualization of uncertain Data

There are different approaches to uncertainty visualization, that are based on different conceptions for uncertainty or use different visual metaphors.

They can be roughly divided into four categories.

1. Ambiguation
2. Marking
3. Non-Visualization
4. Interactive Techniques

2.2.1 Ambiguation

Having uncertainty implies that there is no single definite answer to some question. So, it is appropriate to give an intentionally ambiguous answer to provide the viewer with at least some information while still making clear that we are uncertain about that answer.

Ambiguation tries to display uncertain information in such a way, that the user will be just as uncertain as to the value of a variable as we are.

Most of the following techniques are based on probabilities. They interpret the data, sometimes implicitly or even unknowingly, as random variables that are no longer single values, but distributions that are spread out to a *range* of possible values—a probability density function.

Mathematically, this amounts to adding some random noise to a true (noiseless) value[4]. This in turn translates to convolving the delta function with the density function of the noise, which yields that same density function centered at the true

[4]Noiseless values have no uncertainty, but they can still be formally treated as random variables—having the Dirac delta function as their probability density function.

value. Keeping this in mind will help understand the theoretical background of the following methods.

Blur

One intuitive way to make dots, lines and other glyphs look uncertain is to blur them, i.e. replace them with a (visualization of a) Gaussian distribution for example, so that they look out of focus. That way, the viewer immediately notices the fact that there is some uncertainty in the data and can even see the underlying density function.

The big disadvantage of this approach is that it is only effective for features that have a clear boundary. Otherwise, the difference will be too hard to see. A simple but effective solution for this problem was developed by [CR00]. They add a grid right on top of the visualization and blur that grid instead of the actual data.

[BWE05] use blurring to extend LIC[5] to integrate uncertainty visualization by using a noise pattern with relatively few bright spots and blurring those perpendicular to the flow direction. [PMG04] directly map their likelihood to brightness which also leads to a blurry representation, although here it is a byproduct of their probabilistic model. Rheingans et al. have used blurring too, to display uncertainty in the position of atoms in molecules[GR02][RJ99].

Unfortunately, the actual implementation of blurring usually brings with it the need to switch to an entirely different visualization method. For example, a point without uncertainty is simply a point, flat circle, sphere or some other primitive that can be rendered easily. If that point is now uncertain and needs to be blurred, we suddenly need to switch to textures or direct volume rendering (DVR) in order to display the distribution of the value. So implementation-wise, adding uncertainty in this fashion is not a simple extension of existing methods but calls for a complete re-implementation using a completely different method that typically comes with a number of problems such as dependency on the resolution of the display and high memory consumption.

Another caveat here is that such a visualization will often include color gradients that come with the usual problems such as the lack of a natural ordering of colors and problems for the color blind. If a gray scale coding is used, contrast and brightness of the screen or printer will influence the visualization —which can make the exact same image look significantly different on different devices.

[5]Line Integral Convolution

Fat Features

A simpler possibility of conveying uncertainty is to make lines or other glyphs "fat" so that their width scales dependent on the uncertainty. From a probabilistic point of view, this corresponds to a uniform distribution of values (in contrast to a Gaussian for blur).

The main advantage of this method is in its clarity. The amount of uncertainty is clearly visible and measurable, in contrast to blurring, where it is much harder to judge how large the standard deviation really is.

On the other hand, this method visually implies that the uncertainty shown is bounded. If that is not the case, i.e. the underlying uncertainty is not uniformly distributed, the method can still be used e.g. to display the area bounded by the mean function $\pm$ one standard deviation. But then, great care must be taken to clearly label the image and perhaps show the areas of $\pm 2SD$ and $\pm 3SD$ too—simply to make the viewer aware that something is going on and thereby forcing them to inform themselves instead of blindly assuming something.

[Ger98] points out that there is a certain risk that a **thick line** becomes an eye-catcher and is mistakenly believed to particularly relevant, which is the opposite of what we want to express with our image. He suggests using **transparency** to avoid this effect.

Fat glyphs can be applied relatively easily to extend a wide array of existing methods to include uncertainty. Lines or points in a graph simply get a variable, possibly anisotropic, diameter. [LFLH07] for example, use an ellipse to show the anisotropic[6] uncertainty of the star positions they are visualizing with their system. Lines in 3D become ribbons or tubes. See [WSF+95] and [LB98] for example.

For heightfields, fat surfaces are used. They can either be implemented as multiple surfaces or direct volume renderings. Their inherent problem is that the surfaces will obscure each other. In particular, the surfaces that symbolize the upper boundary of the uncertainty may be misinterpreted to be the mean or true value. To avoid this, the mean surface can be of different color and high specularity as in [PH11].

Samples

Again using the probabilistic interpretation, the visualization can simply include multiple samples, multiple possible realities. This way, it becomes immediately

[6] Their angular position in the sky is well-known. Their distance is much less certain.

obvious that there is no single true value.

It is also easy to implement, because it is simply more of the same. If the uncertainty is given as a distribution, standard methods usually exist to draw a sample. However, that is not always the case and then it may be a non-trivial task.

Another disadvantage is that, by pure chance, the random samples might be biased, i.e. do not properly represent the whole range of possible values. If for example all samples happen to be far below the mean, the viewer will be misled. So, it is advisable to prevent such extreme cases from happening, for example by drawing triples of samples until a non-degenerate case is produced. In general, showing samples is not a good method if the exact magnitude of uncertainty needs to be visualized. The mean can be hard to see too, which can be a disadvantage depending on whether the distribution has a meaningful expected value or not.

Another problem is the visual clutter that can occur in real-life applications. This happened to [RJ99]. They consequently made the samples semi-transparent so that they add up in places where they are nearly in the same location, but stay transparent in areas where they are not. This helps reduce the clutter a little bit.

Noise

Making uncertain areas noisy is another natural method to show the viewer that the data is uncertain. However, this method is risky as the viewer could mistakenly assume that noisy areas are areas where the data was sampled very accurately, thinking: "Otherwise the resolution/accuracy would have been too low to actually see the noise!" Similarly, smooth areas next to noisy ones might be interpreted to simply be low-resolution and therefore less reliable.

So, simply perturbing the existing visualization is not enough. It must be made obvious that the noise is artificial and meant to symbolize uncertainty. One way this could be done is to disconnect noisy triangles from the rest of the surface instead of simply displacing their vertices.

2.2.2 Marking

The simplest form of uncertainty visualization is to take uncertainty to be just another scalar variable that has to be included in the visualization. It is conceptually easy and allows us to re-use all the existing methods of visualizing scalar data: colormaps, heightmaps, textures, glyphs. A variety of methods can be found in [PWL97].

However, we have to explicitly tell the viewer how to interpret the visualization since it will usually not be intuitively clear that the additional scalar represents uncertainty. Another disadvantage can be that those methods do not necessarily convey a sense of the magnitude of the error. The problem might be avoided by using glyphs, though this can lead to visual clutter. Error bars have been shown to perform even worse compared to other glyphs and color maps[SZB+09].

All in all, it is usually preferable to use a specially designed uncertainty visualization rather than re-using the normal methods. If possible, they can still be used to make the uncertainty visualization redundant. For example, a large glyph can help show the viewer points of unusually high uncertainty. This is particularly useful when a visualization method is used where single data points can easily be overlooked. For example, a single small group of transparent pixels in an otherwise inconspicuous data set may simply go unnoticed! This can and should be avoided, for example by marking such positions with large red glyphs by default (which the user may disable later). [SCB+04] do this using a needle-style glyph consisting of a well visible red ball and a thin vertical line that literally pinpoints the exact location.

2.2.3 Non-Visualization

A third way of visualizing uncertainty information is to simply not display anything at all. It is very intuitive and not hard to implement.

One of the caveats is, that it ultimately leads to some sort of transparency, which raises the question of what is behind that line, surface or texture.

[Hen03] used non-visualization method to soil data using color maps, brightening the colors depending on the uncertainty. This works well when the uncertainty is very high, but when the data is only a little uncertain, and thus the colors only slightly brighter, it is very hard to even notice. This is a big problem, even when using an HSB color model where brightness can easily be decoupled from hue.

A solution for this issue can be to let a checker pattern or wire frame show through instead of a solid white “canvas”.

An interesting application of non-visualization comes from map-making. [LC97] looked into the possibility to only draw boundaries of areas when they are certain about where they actually are. If they are uncertain, the line dividing to areas simply stops. Such lines are called “Twains”. They also look at algorithms to generate maps from uncertain data. Such techniques from the mapping community might be helpful in the visualization of topologies of uncertain flow fields.

A combination of noise and non-visualization is used by [SMI99] to visualize the architecture of ancient buildings. There may be very little data about them, sometimes the ruins being the only data source. Uncertainty is therefore substantial and Strothotte et al. developed a visualization tool that tries to avoid photorealism and instead mimics a line drawing with the strokes becoming thinner, transparent, shaky and less consistent in areas where the uncertainty is high.

For direct volume rendering, transparency is the natural way to implement non-visualization. [DKLP02] experimented with this method (and others) and their results look useful. The biggest advantage of DVR, in terms of making the user aware of uncertainty, is that the user has to define his own transfer function on a scatter plot that has uncertainty as one axis. This way, the user cannot ignore uncertainty, but is *forced* to decide how much uncertainty he is willing to tolerate. With that in mind, it would likely benefit *any* uncertainty visualization —no matter if it is a DVR or a 2D visualization —to have the user control the transfer function in the that way.

2.2.4 Interactive Techniques

One central problem to uncertainty visualization is the fact that it uses up at least one more visual variable. So, extending the visualization to have another dimension, time, can be a means to add uncertainty without interfering with the established way of visualizing.

[Eva97] looked at various methods of interactively displaying uncertainty in GIS data. Apart from showing data and uncertainty side by side, she found flickering to be effective, although somewhat annoying. Flickering is also on of Ware's "preattentive variables"—visual variables, that "pop out"[War04],which makes it particularly suitable to direct the users attention to the presence of uncertainty.

Another use of flickering would be to randomly show alternating images that show different samples. That way the user can see the uncertainty without having the image change all the time from the data view to the uncertainty view.

It is also possible to animate the visualization to oscillate and show all or some range of possible realities. This avoids accidentally picking biased samples (cf. 2.2.1). The speed of such an animation should be variable. It could be fast (like flickering) by default to make the user aware of the uncertainty and then allow the user to adjust the rate to something slower so that he can examine the amplitude of the error. This could be done semi-automatically by having the frequency scale with the zoom factor. [LLPY07] animate medical volume data to show a range of

images created with different transfer functions used for classification of different types of tissue.

The equivalent for glyphs is to vibrate[SCB+04] along the direction of the uncertainty. [PWL97] also describe the possibility of adding and removing glyphs to make the uncertainty even more obvious.

2.3 Modeling Uncertainty

This thesis will deal with Gaussian process regression, a statistics technique that has applications in machine-learning[RW06] and is also known in the GIS scene, there called *Kriging* (e.g. [VRNDW10]) named after the South African mining engineer who developed the method. It can be used to model uncertain data (eg. [HT06]), i.e. a set of Gaussian-distributed random variables. However, it has not been used for general purpose visualization yet and there is not much material for heteroscedastic Gaussian processes, i.e. Gaussian processes with different uncertainty levels at the data points.

There do not seem to be other widespread methods for general purpose uncertainty modeling, indeed not even Kriging seems to be known outside the GIS community. But some approaches for special applications do exist.

For very simple cases, if may be sufficient to simply assume identical distributions, perhaps two-dimensional or more, at every data point, for example a Gaussian/normal, log-normal, exponential, Poisson or some other, and just store the parameters (e.g. (co)variance) at every data point. For more complicated situations, other schemes are needed.

[PMG04] visualize possible surfaces that fit a given point cloud. To do this, they created their own empirical model to estimate the probability of a line going through a particular point and compute that probability for all pixels. That probability is designed in such a way that a low angle between the point and a possible line segment, and a small length of that line segment lead to a high probability. All the probabilities with regard to all line segments is merged into a distance-weighted sum which gives the final probability for the surface to pass through that point. A big advantage of that method is, that distinct alternative routes are possible, which is not usually the case as regression normally seeks to find a single mean function.

[PH11] presented a generalization of isocontours for uncertain fields. They compute the level-crossing probability—the probability for an interval to have a crossing

of the given threshold in it. A simple, but convincing method that is easy to understand and compute.

[KVUS+05] try to perform classification of medical volume data under uncertainty. The model they base their transfer function on is what they call the decision boundary distance that is being computed for every class, which is a maximal log-odds ratio of all the *other* classes. Roughly speaking, it is a measurement of the risk of being wrong to assume that the current class is the correct one.

Chapter 3

Introduction to Gaussian Processes

Uncertainty usually presents itself as a random effect, as all the *non*-random effects hopefully have already been discovered and included in the mathematical model that was used to gather, create or derive the data we are trying to visualize—and therefore are not uncertain anymore.

Thus, treating uncertain values not as single values, but as a distribution about some mean value or a range of possible values is a natural choice. Another advantage of using such a stochastic framework is the large body of existing standard methods to describe random phenomena. A lot of which have been extensively studied on both the theoretical and the empirical side.

The *Gaussian process* is one such method. It provides us with a framework that will allow us to model uncertain, Gaussian-distributed[1] data. This chapter will present the basic ideas behind Gaussian processes and how to use them. A more in-depth treatment of the topic can be found in [RW06].

3.1 A Note on Naming and Typography

From now on, "data" will refer to random variables characterized by a normal distribution, in practice represented by two scalars: mean and variance.

It will later become important to distinguish between matrices, scalars, vectors, positions (which in practice typically are vectors too) and vectors of vectors (lists

[1] alias: normal distributed

of positions).

Matrices will always be uppercase letters. Generic random variables that are of no particular type will be called X (the uppercase letter X). That convention will only be relevant at the beginning of this chapter, although of course all values derived from the random data are again random.

Vectors will be formatted like this: $\vec{x}$—with a small arrow above. Positions, although usually vectors too, will be formatted differently—in boldface: $\mathbf{x}$.

Values (of any type) that do not come from the original data, for example positions at which we interpolate or posterior/interpolated values, will have a subscript asterisk. E.g. Σ_* for the posterior covariance matrix.

3.2 Definition

A random process can be defined simply as a set of random variables $\{X_1, \ldots, X_n\}$. E.g. a series of coin tosses. In the special case of the *Gaussian* process, those random variables all follow a Gaussian distribution. That being the case, they are each characterized by their mean μ and variance, usually denoted as σ^2 or $\mathrm{Var}(X)$.

Since there may be dependencies between each two of those random variables, the covariance $\mathrm{Cov}(X, Y)$ plays the important role of defining that dependency. Note that the variance is the covariance of that variable and itself: $\mathrm{Var}(X) = \mathrm{Cov}(X, X)$.

A slightly different, but more useful way to look at a Gaussian process is to take it to be a single, but multivariate, normal distributed random variable.

$$(X_1, \ldots, X_n) = \vec{X} \sim N(\vec{\mu}, \Sigma)$$

with mean $\vec{\mu}$ (now a vector) and covariance matrix Σ that are defined as follows using the expected value $\mathbb{E}(.)$.

$$\vec{\mu_i} = \mathbb{E}\left(\vec{X_i}\right) \tag{3.1}$$

$$\Sigma_{ij} = \mathbb{E}\left((\vec{X_i} - \vec{\mu_i})(\vec{X_j} - \vec{\mu_j})\right) \tag{3.2}$$

And finally, the probability density function of the normal distribution is defined as

$$f(\vec{x}) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})\right) \tag{3.3}$$

($|\Sigma|$ being the determinant of Σ.)

3.3 Multivariate Distributions as Distributions of Functions

In visualization, the data often consists of a set of points $\mathbf{s}$ in some two or three-dimensional space, perhaps time-dependent, that have some value associated with it, in our case a mean and variance of a random variable. Also, there can usually only be one true value at a certain point. This means we can index the variables that comprise our process more intuitively by their position rather than some arbitrary index.

$$X(\mathbf{s}) \text{ or } X_{\mathbf{s}}$$

Note, however, that $\mathbf{s}$ is still only one of a finite number of positions at which we actually have any data.

The underlying truth that we are trying to model is usually a function that is smooth and defined on a subset of $\mathbb{R}^n$. This leads us to two core problems:

1. How do we refine the process to get a somewhat continuous function? That is, how do we add points to the process at arbitrary positions?

2. How do we get the resulting function to be smooth?

The first problem is the interpolation problem. It has many established solutions ranging from simple methods such as linear interpolation to more sophisticated ones such as splines or even the sinc filter. The Gaussian process framework will lead us to yet another method.

The second problem has a solution too. Unlike the name "*random* process" suggests, the function need not become random noise as we add more points. We can design the covariance structure to respect spatial proximity, forcing close points to be correlated!

3.3.1 Covariance Structure

So far, the covariance structure[2] has been defined by a covariance matrix, effectively a table containing the covariance of each pair of random variables. It is possible to estimate these covariances from the data using equation 3.2 and some estimator for the expected value (e.g. the arithmetic mean).

[2]called that to distinguish it from single covariances

But there is another method to define a covariance structure—the *covariance function*:

$$\mathrm{Cov}(X_{\mathbf{s}}, X_{\mathbf{t}}) = k(\mathbf{s}, \mathbf{t}) \quad \text{with} \quad k : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R} \tag{3.4}$$

Where k is a function that depends *not* on the random variables themselves but on their *positions only*! This simplification is justified intuitively by the idea that data points that are far apart usually should not influence one another. Theoretically, this idea is supported by the fact that the ideal interpolation function $\mathrm{sinc}(x)$ tends to zero as $x \to \pm\infty$.

It is important to note that we are, at least for a moment, deviating from the path usually followed by statistics. Instead of estimating the covariance structure from the data, we boldly *defined* one as we saw fit. *But*, we will later see that it will only act as a *prior*—not as the finished result! However, the covariance function will still have a large influence and we will analyze this influence in chapter 4 and look at alternative covariance functions in 3.6.2.

We will use the following covariance function:

$$k_{\mathrm{SE}}(\mathbf{s}, \mathbf{t}) = \sigma_f^2 \exp\left(-\frac{|\mathbf{s} - \mathbf{t}|^2}{2l^2}\right) \tag{3.5}$$

which is a Gaussian function but is called the *squared exponential* to avoid confusion. It effectively only depends on the distance between the two points. $\sigma_f^2 \in \mathbb{R}$ and $l \in \mathbb{R}$ are hyperparameters that will become important later.

The use of a covariance function simplifies computation and, more importantly, enables us to compute a covariance between any two points no matter if we have data about them! In the next section, this will serve us to compute a mean and variance at any point we want.

3.4 Gaussian Process Regression

Gaussian processes can be used for interpolation. However, we will find that it is in a number of ways unlike what computer science traditional calls interpolation. Being a statistical method, what we will actually be doing is *regression*, which will get us a *model* to fit our data.

Gaussian process regression is done in two steps and is conceptually based on the process being a multivariate Gaussian distribution. First, a prior covariance structure and mean of the points in question are formally added to the distribution, thereby tying the new points to the already existing ones via the covariance between

them. Second, the conditional distribution is being computed, which gets us a posterior mean and covariance for the new points. Roughly speaking, we are telling the process that the new variables are correlated to the existing data points. Then, we provide the values at those data points and, by correlation, the process "learns" about the value at the new points.

Augmenting the covariance matrix Σ with the covariances of the new points $\vec{X_*}$ is easy. We simply compute them using the covariance function presented above[3] and get

$$\Sigma' = \begin{pmatrix} K(\vec{\mathbf{x}}, \vec{\mathbf{x}}) & K(\vec{\mathbf{x}}, \vec{\mathbf{x}_*}) \\ K(\vec{\mathbf{x}_*}, \vec{\mathbf{x}}) & K(\vec{\mathbf{x}_*}, \vec{\mathbf{x}_*}) \end{pmatrix} \tag{3.6}$$

The mean is assumed to be zero. This condition can easily be met by subtracting the empirical mean value from at all data points and later adding it again to the final result. More sophisticated schemes will be discussed in 4.3.4.

The conditional distribution then has the following posterior mean and covariance matrix:

$$\vec{\mu_*} = K(\vec{\mathbf{x}_*}, \vec{\mathbf{x}}) K(\vec{\mathbf{x}}, \vec{\mathbf{x}})^{-1} \vec{\mu} \tag{3.7}$$

$$\Sigma_* = K(\vec{\mathbf{x}_*}, \vec{\mathbf{x}_*}) - K(\vec{\mathbf{x}_*}, \vec{\mathbf{x}}) K(\vec{\mathbf{x}}, \vec{\mathbf{x}})^{-1} K(\vec{\mathbf{x}}, \vec{\mathbf{x}_*}) \tag{3.8}$$

$\vec{\mu_*}$ contains the posterior mean at the new points and Σ_* the covariance between them. Recall that the trace of Σ_* contains the variances of the new points.

This type of regression is what is known as *Kriging* in GIS community.

We can now interpolate the process at any position we like. For this reason, the Gaussian process is sometimes denoted as

$$\mathcal{GP}\left(\mu_*(\mathbf{s}), k_{\mathrm{SE}}(\mathbf{s}, \mathbf{t})\right) \tag{3.9}$$

where the posterior mean $\mu_*(\mathbf{s})$ is called the *mean function*. Similarly, we will call the variance encoded in the posterior covariance matrix Σ_* the *variance function*.

An example for a Gaussian process interpolated in this fashion can be seen in Figure 3.1. The variances from this simple process already give us important information about the uncertainty in our data, in particular about how big a mistake we potentially make when interpolating values between two data points!

That example is for scalar values in 1D, but Gaussian processes can easily be applied to any higher-dimensional space as long as there is a covariance function. Such processes are sometimes called random *fields*.

[3] $K(\vec{\mathbf{a}}, \vec{\mathbf{b}})_{ij} = k_{\mathrm{SE}}(\vec{a_i}, \vec{b_j})$. So, K can be thought of as a batch-processing version of k that returns a covariance sub-matrix

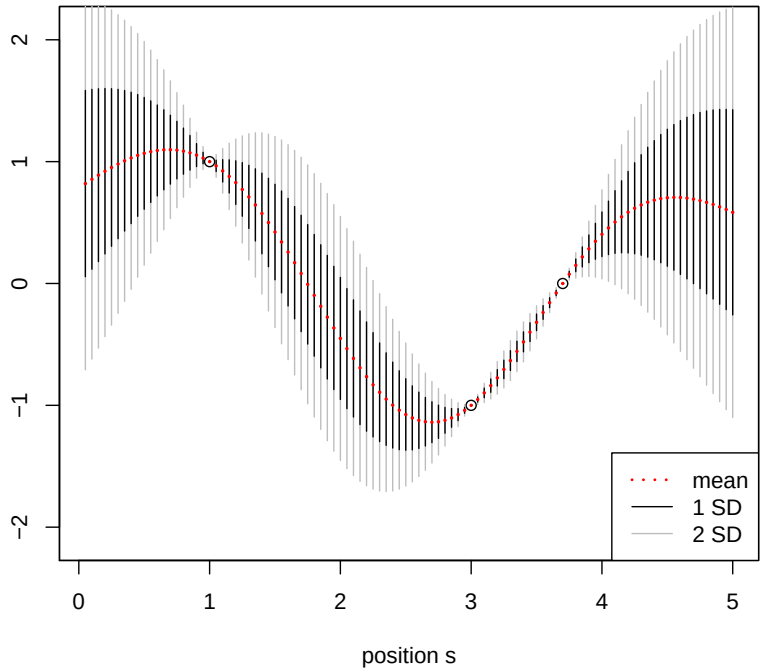

Figure 3.1: A Gaussian process based on three data points, interpolating the rest. Each red dot is the mean of an interpolated point. The black line extends to ± one standard deviation, the gray one to ± two standard deviations. (The images in this chapter were created using a custom R script.)

3.4.1 Relationship to Convolution

Setting $W = K(\vec{\mathbf{x}}_*, \vec{\mathbf{x}})K(\vec{\mathbf{x}}, \vec{\mathbf{x}})^{-1}$, the equation for the mean is reduced to

$$\vec{\mu_*} = W\vec{\mu} \tag{3.10}$$

and it becomes obvious that it is really only a simple linear mapping, a weighted sum, and thus a kind of discrete convolution with the so-called *equivalent kernels* for each interpolated value $\vec{\mu}_i$ in the corresponding row of W.

Knowing this, the ideal covariance structure (in some sense) would be one that produces sinc as the equivalent kernel or other good low-pass filters like the Gaussian/binomial filter. However, this also implies that there is a risk of generating the a filter that is too narrow-band and leaves us with a process that is unable to properly model the data if the data contains higher-frequency features. On the other hand, the fact that the equivalent kernel has finite support also means that there will inevitably be high frequencies in the mean function (although hopefully highly attenuated).

There are some analytical results for the structure and properties of equivalent kernels[4], but we will not discuss them here because they require more theoretical background and mainly deal with the mean function itself, which, although relevant, is not the main focus here.

The primary motivation for using the, relatively costly, Gaussian process interpolation is indeed not that we get $\vec{\mu_*}$—there already are established interpolation methods for that. The more important result here is that we get interpolated *variances* based on a reliable statistical framework.

3.5 Noisy Data

Until now we have dealt with noise free, zero-variance processes[5]. While those already provide us with useful information about the uncertainty that exists in the space between two data points, what we really want is the process to incorporate our knowledge about the uncertainty at the data points.

[4] See [RW06] or [AT10].

[5] Confusingly, the prior variances, more exactly the diagonal entries of the prior covariance matrix, are actually not zero at all. To the contrary, the squared exponential has its maximum there, as the two positions coincide: $k_{\mathrm{SE}}(\mathbf{s}, \mathbf{s}) = \sigma_f^2 > 0$. However, if you work out the regression, you will see that this variance will end up being subtracted from itself, yielding zero. Gaussian process regression is a bit odd there.

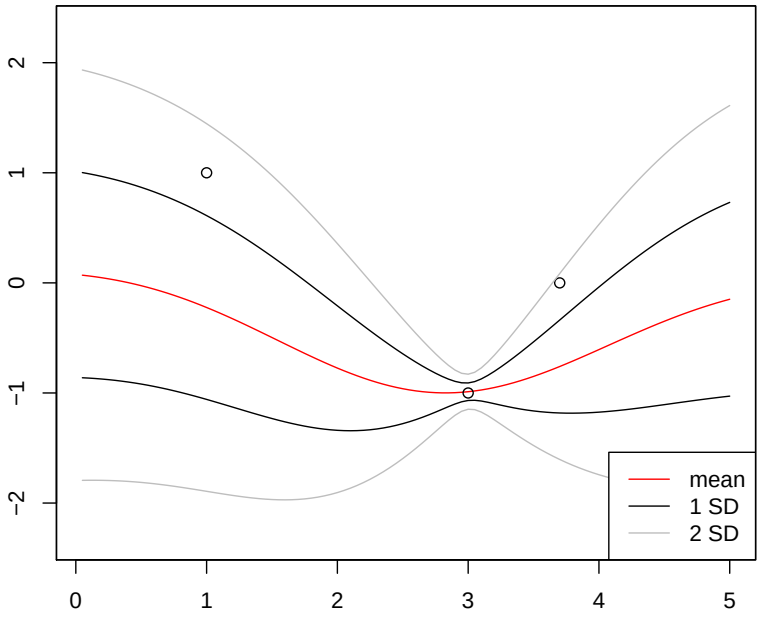

Figure 3.2: The Gaussian process from Fig. 3.1, this time with noisy data with standard deviations 2, 0.08, 1). Note: In this and the following plots, mean function and variances will be displayed as continuous curves (a large number of posterior values connected by line segments) rather than dots or lines, for the sake of a clearer visualization.

This is done by adding the desired variances at the corresponding points to the prior covariance matrix, that is, to the diagonal:

$$\Sigma' = \begin{pmatrix} K(\vec{\mathbf{x}}, \vec{\mathbf{x}}) + \mathrm{diag}(\vec{\sigma^2}) & K(\vec{\mathbf{x}}, \vec{\mathbf{x}_*}) \\ K(\vec{\mathbf{x}_*}, \vec{\mathbf{x}}) & K(\vec{\mathbf{x}_*}, \vec{\mathbf{x}_*}) \end{pmatrix} \tag{3.11}$$

where $\vec{\sigma^2}_i$ contains the desired variance of the i-th data-point and $\mathrm{diag}(\vec{\cdot})$ is the diagonal matrix containing the elements of the given vector as its diagonal entries.

3.5.1 Differences to traditional Interpolation

As can be seen in Fig. 3.2, the noisy interpolation naturally differs a lot from the noiseless version. What is more interesting, is that it also differs from what we traditionally expect from an interpolation in a number of ways.

First of all, the mean function does not run through the given points. The reason for this is, that by adding the data variances to the process, we "admitted" that our data is imperfect. This does not only imply that there exists some error defined by the variance but also that the given values usually are not the same as the true values which we try to model by our mean function[6]!

Another effect can be seen at the left and right ends of the plot. There, the curves adjust to the prior variance and prior mean of the process. The first is encoded in the covariance function as σ_f^2 which is one of the two hyperparameters we will deal with in section 4.3.1. The prior mean is zero, since the Gaussian process as we have presented it here assumes zero-mean. If there are too many missing values in the data, the effect will show up there as well. This and other side-effects of the prior will be treated in more detail in section 4.3.1.

3.6 Hyperparameters and Models

The parameters σ_f^2 and l of the covariance function, called *hyperparameters*, influence the process in multiple ways.

Recall that squared exponential we are using as the covariance function is

$$k_{\mathrm{SE}}(\mathbf{s}, \mathbf{t}) = \sigma_f^2 \exp\left(-\frac{|\mathbf{s}-\mathbf{t}|^2}{2l^2}\right) \tag{3.5}$$

Because this kernel is symmetric, we can use a simpler version here that only depends on the distance between the two points.

$$k_{\mathrm{SE}}(\Delta_{\mathbf{st}}) = \sigma_f^2 \exp\left(-\frac{\Delta_{\mathbf{st}}^2}{2l^2}\right) \tag{3.12}$$

Although it is not normalized to have

$$\int_{-\infty}^{\infty} k_{\mathrm{SE}}\, d\Delta_{\mathbf{st}} = 1$$

and therefore is not a Gaussian *distribution*, it is still a Gaussian function and the hyperparameter l can be thought of as a standard deviation in the sense that 68% of the function's mass is concentrated in the interval $[-l, l]$, 95% in $[-2l, 2l]$ and

[6]For this reason it might be better to call the mean function "expected function" instead, as it is only loosely coupled with the mean value of the distributions of the data-points.

so on. For this reason, l is called the *characteristic length scale* that determines the radius in which a data-point significantly influences the process.

The choice of the length scale l has a very strong influence on the interpolated function (cf. fig. 3.3). The smaller it is, the quicker the covariance between two points decreases with their distance and the more freely the process can vary. This enables the process to fit the data points better, but there is a high risk of overfitting, i.e the model degenerating into a curve that fits perfectly with the data we already know, but fails to produce a meaningful estimate for areas where we have no data.

In addition to the increased flexibility of the curve, the function will fall back to the prior sooner as the influence of the data points does not reach as far anymore. This is another symptom that may indicate overfitting.

Higher values on the other hand make the function smoother but sacrifice accuracy. This in turn suggests that the overall variance of the process must be pretty high. Otherwise, the large amount of what are now basically outliers would not be plausible or likely[7]. In this case, we probably have underfitting, i.e. our model will present us with new estimates, even at places where we have no data, but already fails to respect the data we already have.

Having said all that, we will need to evaluate the different models that come with different hyperparameters.

3.6.1 Choosing the Hyperparameters

In an ideal world, we would simply take the values for the hyperparameters from the mathematical/physical theory that our data is based on. For example, if we already know that the values are scattered around some mean and will only exceed a certain value with a certain known probability, we can create a Gaussian distribution that fits with this information and use its variance for σ_f^2 (if the data is Gaussian).

In most cases, however, we will not have such information—in particular not about the length scale l—so we might want to estimate it from the data and pick the "best" model.

There are different methods for evaluating a given model. Cross-validation for example trains the model with only a subset of the data, and then checks how well it models the rest, that it previously did not know about. The idea behind this

[7] "likely" in both the statistical and the common sense of the word.

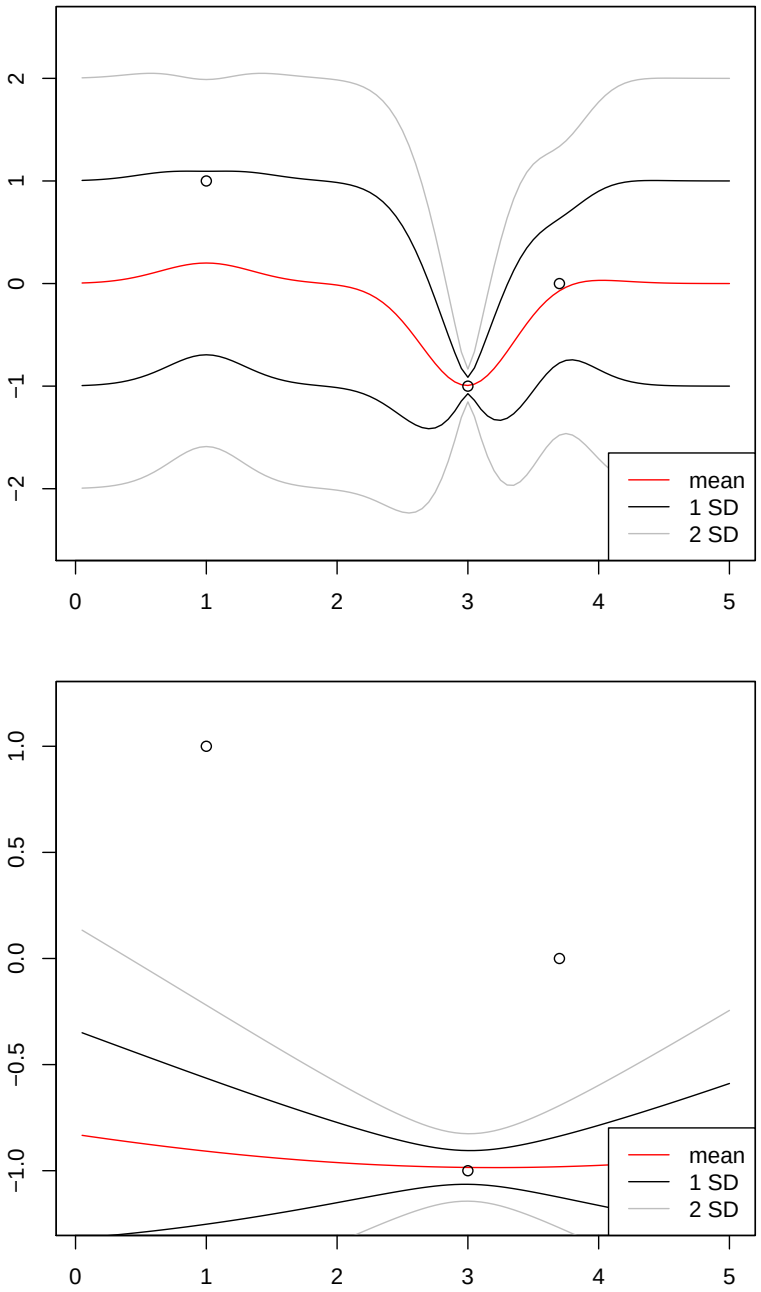

Figure 3.3: The same process with different values for the characteristic length. $l = 0.25$ (top) and $l = 4$ (bottom).

approach is that good models should not merely mimic the given data, but also be able to provide a sensible prediction.

Here, we will take a Bayesian approach by optimizing the *marginal likelihood* to evaluate our process. The question that this answers is: Assuming our model represents the truth. What is the likelihood to observe our data?[8]

The log marginal likelihood of the Gaussian process is given by evaluating the probability density function of the process (remember that the it is a multivariate normal distribution) at the data points[RW06]:

$$\begin{aligned}\log p(\vec{y}|X) = &-\frac{1}{2}\left(\vec{y}^T(K(\vec{\mathbf{x}},\vec{\mathbf{x}}) + \mathrm{diag}(\vec{\sigma^2}))^{-1}\vec{y}\right) \\ &-\frac{1}{2}\log|K(\vec{\mathbf{x}},\vec{\mathbf{x}}) + \mathrm{diag}(\vec{\sigma^2})| \\ &-\frac{n}{2}\log 2\pi\end{aligned} \tag{3.13}$$

n being the number of data points and $|.|$ the determinant.

The goal now is to find hyperparameters that create the process that makes the data look the most likely. Finding that maximum is simply an optimization problem (albeit a computationally costly one as each evaluation involves a matrix inversion) that can be solved using the usual numerical methods.

The derivatives of the log marginal likelihood are given by

$$\frac{\partial}{\partial\theta}\log p(\vec{y}|X) = \frac{1}{2}\,\mathrm{tr}\left((\vec{\alpha}\vec{\alpha}^T - K(\vec{\mathbf{x}},\vec{\mathbf{x}})^{-1})\frac{\partial K(\vec{\mathbf{x}},\vec{\mathbf{x}})}{\partial\theta}\right) \tag{3.14}$$

$$\text{where}\quad \vec{\alpha} = K(\vec{\mathbf{x}},\vec{\mathbf{x}})^{-1}\vec{y}$$

allowing us to use gradient descent methods to perform the optimization.

Note, however, that there may be more than one local maximum[RW06]. Also, in some degenerate cases, in particular for large σ_f^2 there need not be a maximum.

Optimizing the properties of the process in this fashion may look a bit like wishful thinking, but it turns out that the marginal likelihood does punish both over- and

[8]From a mathematical stance, the likelihood is a probability. *But* if we were to call it that, then this question would be backward and indeed, if you do not subscribe to the Bayesian school of thought, you may very well claim that the question is not even valid or that the answer is always "100%" since we *observed* that data—it *already happened*! So the probability ca not be anything else than 1! For this reason, there is a clear distinction between "probability" and "likelihood".

underfitting. The length scale l for example cannot be too large because that would obviously worsen the fit. On the other hand, if l gets very small the covariance between two neighboring points drops to almost zero. Declared to be uncorrelated in this way, they will not benefit from the added evidence that otherwise had come from having multiple samples at nearly the same location.

The prior variance σ_f^2 will be discussed in detail in 4.3.1.

3.6.2 Other Covariance Functions

We have so far used the standard choice for the covariance function—the squared exponential kernel. It has some desirable properties like being infinitely differentiable and therefore very smooth.

Some phenomena are not smooth though. In fact, those areas were the data is not smooth, are often of particular interest. Consider singularities in a vector field for example or edges in an image.

For this reason, it can be worth looking at other covariance functions. In principle, we can use whatever function seems fitting to model our data, as long as it is positive semidefinite.

To avoid having to select a function ourselves, we could take a set of suitable functions that not only have parameters for their radius, but also for their shape, ideally directly for the of the equivalent kernels they produce. Then we could again use a most-likely approach to find the best parameters. Another possibility would be to estimate the covariance from the data and then find a kernel that resembles the empirical covariances. If there is no kernel that matches all the data points, the points should perhaps not be in the same process.

The topic is discussed in detail in [RW06, Ch. 4, 5.4] including a list of covariance functions and an example of a periodic covariance function and analytical results about the equivalent kernels that the different covariance functions define.

One example for the effects of an alternative covariance function is given in 3.4. The alternative function, a polynomial, makes the mean function match the data, coming from a triangular function, better than the original process based on the squared exponential. The latter tending to produce more rounded corners and overshoots at the critical points of the function.

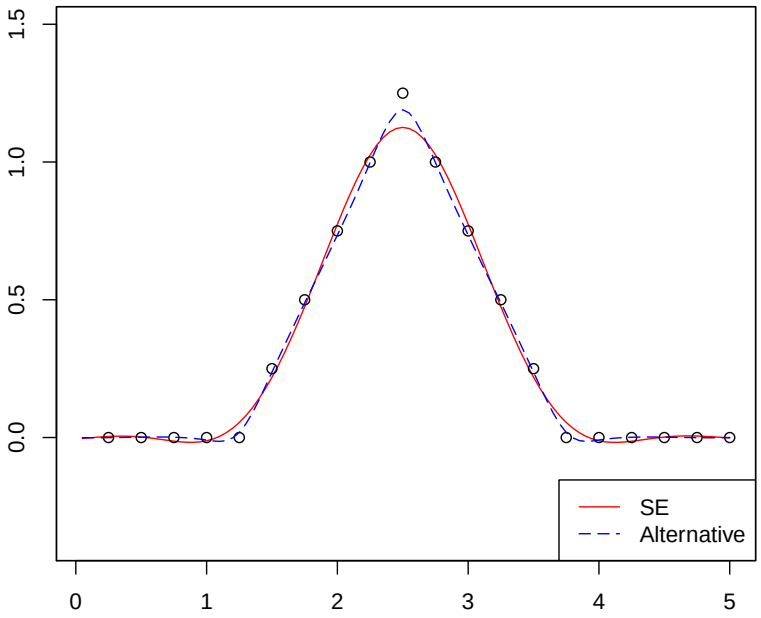

Figure 3.4: The same process computed with two different covariance functions. Red: Squared exponential. Blue/dashed: A polynomial kernel with compact support.

Chapter 4

Modeling uncertain Scalar Fields

4.1 Modeling Scalar Data

Modeling scalar data is the simplest application for Gaussian processes. In this context, the model is defined mainly by the covariance matrix, because the mean is zero anyway (see 4.3.4). That matrix in turn depends on the covariance function.

We choose to interpolate at only a one position $\mathbf{x}_*$ at a time, so the posterior covariance becomes

$$\Sigma' = \begin{pmatrix} K(\vec{\mathbf{x}}, \vec{\mathbf{x}}) + \text{diag}(\vec{\sigma^2}) & K(\vec{\mathbf{x}}, \mathbf{x}_*) \\ K(\mathbf{x}_*, \vec{\mathbf{x}}) & k(\mathbf{x}_*, \mathbf{x}_*) \end{pmatrix} \tag{4.1}$$

The central equations for actually evaluating/interpolating the process at that position are

$$\vec{\mu_*} = K(\mathbf{x}_*, \vec{\mathbf{x}}) \left(K(\vec{\mathbf{x}}, \vec{\mathbf{x}}) + \text{diag}(\vec{\sigma^2}) \right)^{-1} \vec{\mu} \tag{4.2}$$

$$\Sigma_* = k(\mathbf{x}_*, \mathbf{x}_*) - K(\mathbf{x}_*, \vec{\mathbf{x}}) \left(K(\vec{\mathbf{x}}, \vec{\mathbf{x}}) + \text{diag}(\vec{\sigma^2}) \right)^{-1} K(\vec{\mathbf{x}}, \mathbf{x}_*) \tag{4.3}$$

The posterior *mean* is important too, but for uncertainty visualization it is not our primary focus. The posterior covariance can be simplified to

$$\Sigma_* = \sigma_f^2 - K(\mathbf{x}_*, \vec{\mathbf{x}}) \left(K(\vec{\mathbf{x}}, \vec{\mathbf{x}}) + \text{diag}(\vec{\sigma^2}) \right)^{-1} K(\mathbf{x}_*, \vec{\mathbf{x}})^T \tag{4.4}$$

Note that Σ_* is now a 1×1 matrix containing only σ_*^2 - the posterior variance at $\mathbf{x}_*$[1].

[1] In spite of it being a scalar, we opted to keep denoting it as Σ_* to avoid introducing yet another lower-case sigma.

4.2 Basic Algorithm

Now that we have properly modeled the process, the implementation of Gaussian process interpolation is straightforward. We simply evaluate equation 4.4 at our point of interest $\mathbf{x}_*$.

We do this with as many points we need. In our case, that would mean calling that function at every texel (or pixel, voxel, vertex, ... depending on the visualization used).

4.2.1 Optimizations and Run Time

One central problem with this algorithm is the fact that we have to invert the matrix

$$K(\vec{\mathbf{x}}, \vec{\mathbf{x}}) + \mathrm{diag}(\vec{\sigma^2})$$

which has a complexity of about $O(n^3)$, where n is the number of data points[2]. Unfortunately, n is going to be large—usually at least several million, e.g. the vertices of a 2000×1000 grid. So, computing this matrix is not practical.

Limiting the Process by Covariance Function Cutoff

One way to significantly reduce the complexity is to use a large number of smaller processes, which only take a limited number of data points into account. So, if we want to compute the variance at some point $\mathbf{x}_*$, we only consider data points, that are within a certain radius around $\mathbf{x}_*$. That radius should depend on the covariance function, as the covariance function is the measure of how much points influence other points.

We simply pick the radius in such a way that 99% of the mass of the squared exponential are within the radius. This is easy since we are using the squared exponential as the covariance function, which is a Gaussian function. Therefore, the characteristic length scale l is a kind of standard deviation and therefore, 99% of the mass is concentrated in $[-3l, 3l]$[3]. So, $3l$ is our radius. Optimizing the hyperparameters as described in 3.6.1, that radius will typically be relatively small. Alternatively, we could use $[-2l, 2l]$ for instance, if we consider 95% mass to be sufficient.

[2] Algorithms with better complexity exist, but currently are not faster in terms of real-world execution time measured in seconds.[CLRS01]

[3] The three-sigma rule.

The complexity of that algorithm is now $O(pn_r^3 + pn_r^2)$. The time it takes to invert the matrix plus the time for multiplication of the row/column vectors $K(\vec{\mathbf{x}_*}, \vec{\mathbf{x}})$. Both happen p times (the number of points at which we interpolate).

Cell-based Computation

We can cut run time some more by only computing one single Gaussian process for a whole cell[4]. The idea behind this is that the process will depend on nearly the same data points for almost all of the points within a cell, so we can use the same Gaussian process for all of them. In particular, we can re-use the matrix $\left(K(\vec{\mathbf{x}}, \vec{\mathbf{x}}) + \text{diag}(\vec{\sigma^2})\right)^{-1}$ instead of computing it over and over again for each interpolation.

This way, the complexity will be reduced to $O(\mathbf{c}n_r^3 + pn_r^2)$ where c is the number of cells. While theoretically, this does not change the overall complexity, in practice the $O(n_r^3)$ matrix inversion will be the most time consuming operation, so this optimization means quite a lot. This also means that increasing p is *comparatively* cheap.

However we lose some time too, because the radius now has to be the radius plus the maximum diameter of the cell to make sure we get the desired coverage everywhere in the cell. So $r = 3l + d_{\text{cell}}$. This leads to more points being included in the process. One thing that one can try is to use smaller cells, so the effect of the additional d_{cell} is smaller. However, doubling the number of cells also roughly doubles the run time, as we have more individual processes, so this is only useful if the amount of points per process decreases by at least 20% ($\sqrt[3]{0.5} \approx 0.79$).

There are some more standard techniques that can be used to speed things up like limiting ourselves to a region of interest or using the GPU to help with faster matrix operations.

Note that except for the optimization, there need not be any grid, not even topology[5] for the algorithm to work!

[4] A cell of the grid the data lives on or a grid that we create specifically for the purpose of optimization

[5] Neighborhood information

4.2.2 Full Algorithm

The following algorithm computes the variance for every pixel in a texture. Extending it to use a different visualization method is easy in most cases.

Data: field: $\mathbb{R}^2 \mapsto \mathbb{R}$: A scalar field
Result: texture: Texture with color-coded variance

```
1 l,σ_f^2 = optimizeHyperParameters();
2 radius = 3l; // cell diameter will be added later
3 foreach pixel (x,y) in texture do
4 |   Position x_* = coordinates(x,y);
5 |   σ_*^2 = varianceAt( x_*);
6 |   texture.colorCodeAtPos( σ_*^2, (x, y) );
7 end
```

Algorithm 1: Main Algorithm

The creation of the Gaussian process and the optimizations described above are performed in the following function. Computing the variance is simply done by evaluating equation 4.4.

Input: $\mathbf{x}_* \in \mathbb{R}^2$: A position
Output: $\sigma_*^2 \in \mathbb{R}$: the posterior variance at $\mathbf{x}_*$

```
1 cell = findCell(x_*);
2 if cellIsInCache(cell) then
3 |   gp = cache[cell];
4 else
5 |   search_radius = radius + cellDiameter(cell);
6 |   Positions x⃗ = surroundingPositions( x_*, search_radius);
7 |   gp = GaussianProcess( x⃗, l, σ_f^2 )
8 |   cache[cell] = gp;
9 end
10 return gp.evaluateVariance(x_*); // see equation 4.4
```

Algorithm 2: varianceAt()

4.3 Priors and Interaction

4.3.1 Prior Variance

It turns out, that the prior variance σ_f^2 has a very strong influence on the posterior variance. Describing this influence exactly is difficult (though formally, the description is in chapter 3 of course).

Looking only at the positions of the original data points and assuming very little covariance with the other data points, i.e. assuming that they are sufficiently far apart, the posterior variance behaves like a weighted average of the prior variance of the process and the variance of that data point.

To show this, let our prior covariance sub-matrix for the data-points have zero covariance[6]:

$$K(\vec{\mathbf{x}}, \vec{\mathbf{x}}) + \mathrm{diag}(\vec{\sigma^2}) = \begin{pmatrix} \sigma_f^2 + \sigma_d^2 & 0 & 0 \\ 0 & \sigma_f^2 + y & 0 \\ 0 & 0 & \sigma_f^2 + z \end{pmatrix} \tag{4.5}$$

Then, performing the interpolation at the first data-point itself, we get:

$$\Sigma_* = \sigma_f^2 - \begin{pmatrix} \sigma_f^2, & 0, & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sigma_d^2+\sigma_f^2} & 0 & 0 \\ 0 & \frac{1}{y+\sigma_f^2} & 0 \\ 0 & 0 & \frac{1}{z+\sigma_f^2} \end{pmatrix} \begin{pmatrix} \sigma_f^2 \\ 0 \\ 0 \end{pmatrix} \tag{4.6}$$

$$\Sigma_* = \sigma_f^2 - \frac{{\sigma_f^2}^2}{\sigma_d^2 + \sigma_f^2} \tag{4.7}$$

$$\Sigma_* = \frac{\sigma_d^2 \sigma_f^2}{\sigma_d^2 + \sigma_f^2} \tag{4.8}$$

To show that this posterior variance indeed behaves like a weighted average of the prior variance of the process and the variance of that data point, we can rewrite

[6]For clarity of notation, we named the prior variance of the data as follows $\vec{\sigma^2} = (\sigma_d^2, y, z)$.

it in a more intuitive way:

$$w_f = \frac{1}{2}\left(1 - \frac{\sigma_f^2}{\sigma_f^2 + \sigma_d^2}\right) \tag{4.9}$$

$$w_d = \frac{1}{2}\left(1 - \frac{\sigma_d^2}{\sigma_f^2 + \sigma_d^2}\right) \tag{4.10}$$

$$\Sigma_* = w_f\sigma_f^2 + w_d\sigma_d^2 \tag{4.11}$$

In that average, the variance-values are weighted by their certainty, which is measured by $1 - \sigma$ and a normalization factor. Note that the variances are weighted using themselves as weights!

These weights can be rewritten as follows. Note that the subscript in the numerator changes:

$$w_f = \frac{1}{2}\frac{\sigma_d^2}{\sigma_f^2 + \sigma_d^2} \tag{4.12}$$

$$w_d = \frac{1}{2}\frac{\sigma_f^2}{\sigma_f^2 + \sigma_d^2} \tag{4.13}$$

So the complete equation becomes:

$$\Sigma_* = \frac{\sigma_f^2\sigma_d^2 + \sigma_d^2\sigma_f^2}{2(\sigma_f^2 + \sigma_d^2)} \tag{4.14}$$

Which can be read as: A variance-value gets more weight, the larger the uncertainty (i.e. variance) of the *other* variance-value is. The sum in the denominator being the sum of the weights/uncertainties. (Compare this form with eq. 4.8.)

The interesting thing here is that this is a form that corresponds to the variance of a weighted average of two Gaussian distributed variables that have variances σ_d^2 and σ_f^2:

$$X = \sqrt{w_f}X_{\sigma_d^2} + \sqrt{w_d}X_{\sigma_f^2} \tag{4.15}$$

Having those sophisticated factors may seem unintuitive, but if we set the uncertainties, which are the unnormalized weights, to be both the same, we get:

$$\sigma_f^2 := \sigma_d^2 \tag{4.16}$$

$$w_f = w_d = \frac{1}{2}\frac{\sigma_d^2}{2\sigma_d^2} \tag{4.17}$$

$$= \frac{1}{2^2} \tag{4.18}$$

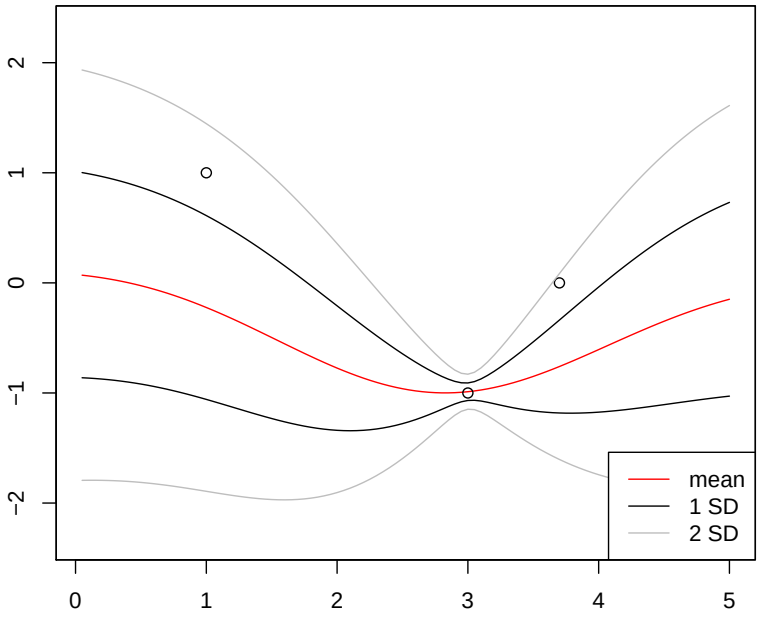

Figure 4.1: The prior $\sigma_f = 1$ clearly influences the shape of the posterior standard deviation, which is nearly constant except for the area around the second data point, which has a higher influence due to its much lower variance.

Data-point position	1	3	3.7
Data prior $\vec{\sigma_d}$	2.000	0.0800	1.000
Posterior $\vec{\sigma_*}$	0.835	0.0796	0.422

$$X = \frac{1}{2}X_{\sigma_f^2} + \frac{1}{2}X_{\sigma_f^2} \tag{4.19}$$

and indeed arrive at the arithmetic mean of two independent Gaussian distributed random variables.

4.3.2 Selectively suppressing the Prior Variance

Knowing how data and prior are combined, estimating the prior from the data itself effectively amounts to treating the data (disguised as a prior) as if it were an additional independent source of information. This might be justifiable if the process uses a lot of data points that all behave similarly. In that case, the single data point only confirms what all the other points around it are already suggesting. It is just one piece of information among millions.

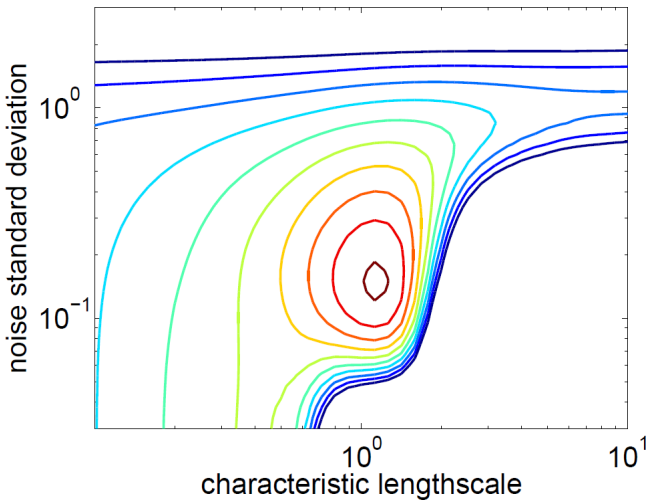

Figure 4.2: The marginal likelihood depends on the two hyper parameters. Note that there is no locally optimal length-scale l for small values σ_f^2. (Iso contour plot taken from [RW06])

However, this would still only be circumstantial evidence. Also, it would not be justified if the process considers very few data points. In that case, we would grossly underestimate the variance. Consider for example a process with only one data point. The process will in that case turn out a much lower variance than we actually have at the data point, which is unacceptable.

For this reason, it is sometimes necessary to suppress the influence of the prior variance on the posterior variance.

We must take great care though not to lose the benefits of Gaussian processregression such as the successful modeling of the interpolation error, because the prior variance also influences the covariances and thereby the correlation between two points via the covariance function. It is also involved in the optimization of the length scale l and a prior variance that is too high can lead to the optimization problem having no solution (c.f. fig. 4.2). So, changing the prior variance will always have far reaching consequences.

A simple solution would be to post-process the result and smoothly multiply the variance function with a second function, designed to change the variance function to assume the desired variances at the data points. That is, a function that smoothly interpolates $\vec{\sigma_d}/\vec{\sigma_*}$. The main two drawbacks of doing this are the need to choose an interpolation method and the fact that we would be leaving the Gaussian process framework.

Using the results from the previous section, we could also try to suppress the prior by setting σ_f^2 to a vastly larger value than σ_d^2, so it would have almost no weight. More exactly:

$$\lim_{\sigma_f^2 \to \infty} \frac{\sigma_d^2 \sigma_f^2}{\sigma_d^2 + \sigma_f^2} = \sigma_d^2$$

But this large prior variance would allow the curves to be extremely smooth too, up to the point where the mean becomes nearly constant, dramatically underfitting our data. In practice, this also means that in order to get a meaningful result, the length scale l should not be estimated/optimized[7] using an arbitrary large σ_f^2.

Prior-corrected Data Variances

Equation 4.8 can be used to show:

$$\Sigma_* = \frac{\sigma_d^2 \sigma_f^2}{\sigma_d^2 + \sigma_f^2} \tag{4.21}$$

$$\sigma_f^2 \Sigma_* = \sigma_d^2 \left(\sigma_f^2 - \Sigma_*\right) \tag{4.20}$$

$$\frac{\sigma_f^2 \Sigma_*}{\sigma_f^2 - \Sigma_*} = \sigma_d^2 \tag{4.21}$$

Suppressing the prior means that we want $\Sigma_* = \sigma_d^2$. Replacing Σ_* accordingly, we can compute what we call prior-corrected data variances $\sigma_d^{2'}$, which will have the desired variances as their posterior.

$$\frac{\sigma_d^2 \sigma_f^2}{\sigma_f^2 - \sigma_d^2} = \sigma_d^{2'} \tag{4.22}$$

Replacing all data variances with their prior-corrected counterpart is easy and fast. It provides an elegant and clean solution for the strong-prior problem, suppressing its influence without disturbing the covariance structure[8] of the process in any way.

The only caveat is that $\Sigma_* > \sigma_f^2$ must hold. But this is a sensible assumption and we can easily make sure that it is the case.

[7] see 3.6.1

[8] i.e. the hyperparameters in the covariance function

An alternative Method

Another method to make the process have the desired variance is described in [GWB] and [KPPB07]. It has a deeper mathematical underpinning and is more complex to compute even though the solution is an approximation. They replace the prior for the point at which the process is evaluated with a noisy version, so their version of equation 4.4 becomes:

$$\Sigma_* = \sigma_f^2 + \sigma_+^2 - K(\vec{\mathbf{x}_*}, \vec{\mathbf{x}}) \left(K(\vec{\mathbf{x}}, \vec{\mathbf{x}}) + \mathrm{diag}(\vec{\sigma^2}) \right)^{-1} K(\vec{\mathbf{x}_*}, \vec{\mathbf{x}})^T \tag{4.23}$$

Their goal is to find values for all the σ_+^2 such as to make the formula evaluate to the variance at the data-points σ_d^2, which is not trivial.

Also, note that the prior σ_+^2 over the variance is even stronger because it is added directly to the process, so they have to take care not to overwrite too much of what the process itself produces.

4.3.3 Interaction between Data-Points

A result similar to that for the interaction of prior and data variance can be derived for the interaction between data-points.

Again looking at the simple case, our prior covariance sub-matrices now are

$$K(\vec{\mathbf{x}}, \vec{\mathbf{x}}) + \mathrm{diag}(\vec{\sigma^2}) = \begin{pmatrix} \sigma_f^2 + \sigma_x^2 & \sigma_f^2 & 0 \\ \sigma_f^2 & \sigma_f^2 + \sigma_y^2 & 0 \\ 0 & 0 & \sigma_f^2 + z \end{pmatrix} \tag{4.24}$$

$$K(\mathbf{x}_*, \vec{\mathbf{x}}) = \begin{pmatrix} \sigma_f^2, & \sigma_f^2, & 0 \end{pmatrix} \tag{4.25}$$

Working it all out, we arrive at:

$$\Sigma_* = \frac{\sigma_f^2 \sigma_x^2 \sigma_y^2}{\sigma_x^2 \sigma_y^2 + \sigma_f^2 \sigma_y^2 + \sigma_f^2 \sigma_x^2} \tag{4.26}$$

Which again can be decomposed into a kind of weighted average similar to 4.14:

$$\Sigma_* = \frac{\sigma_f^2 \sigma_x^2 \sigma_y^2 + \sigma_f^2 \sigma_y^2 \sigma_x^2 + \sigma_x^2 \sigma_y^2 \sigma_f^2}{3(\sigma_f^2 \sigma_x^2 + \sigma_f^2 \sigma_y^2 + \sigma_x^2 \sigma_y^2)} \tag{4.27}$$

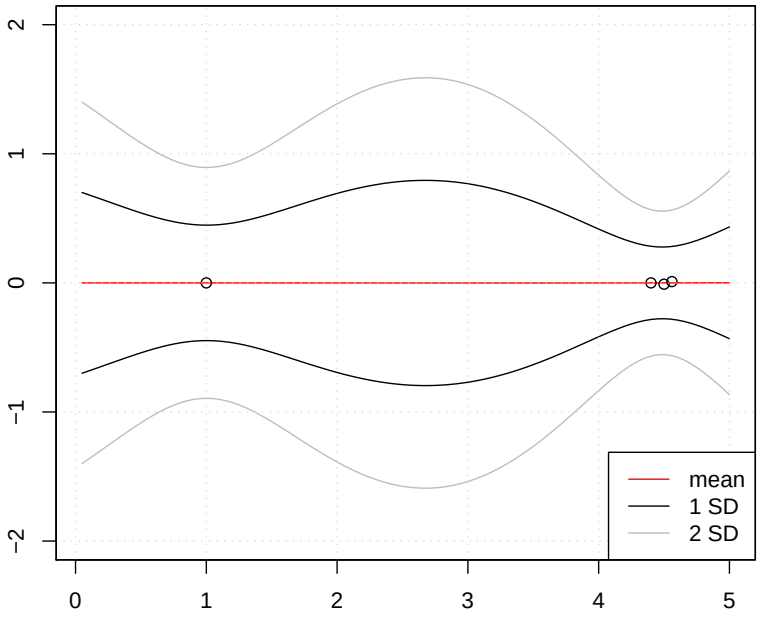

Figure 4.3: The uncertainty decreases where multiple data-points are available, similar to the Law of large numbers.

and by taking the same (still arbitrary) point of view as we did for equation 4.18, we again get the variance of the average of, this time of course three Gaussian random variables:

$$\Sigma_* = \frac{1\sigma_y^2 + 1\sigma_x^2 + 1\sigma_f^2}{3(1+1+1)} \tag{4.28}$$

$$\Sigma_* = \frac{\sigma_y^2 + \sigma_x^2 + \sigma_f^2}{3^2} \tag{4.29}$$

Interaction across Distance

The core of the Gaussian process is the covariance. It defines the interaction between the data points, thereby producing a smooth process. One important side effect of the distance-dependent covariance function is, that it also models the behavior of the process between two data points.

It should be obvious, that the certainty about how the underlying reality looks should be greatest at those places, where we actually have data, i.e. at the data-

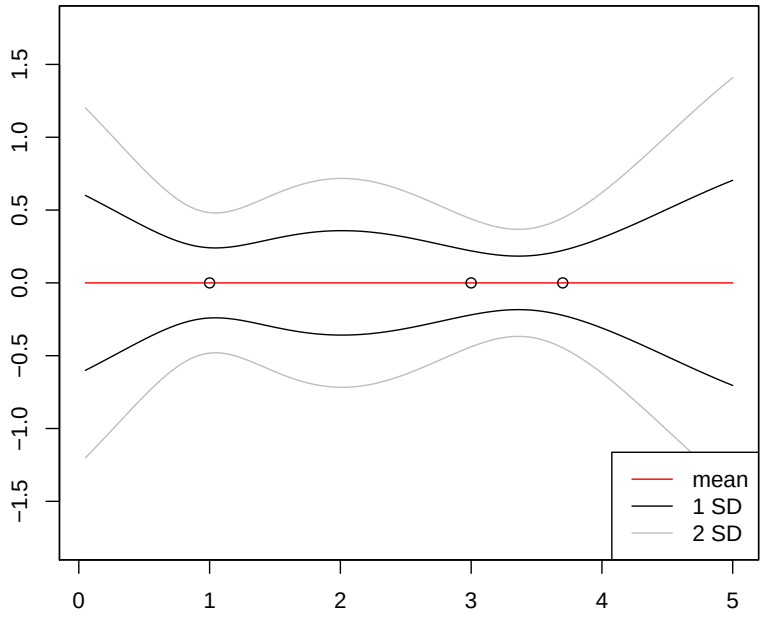

Figure 4.4: The uncertainty may increase (left pair) as well as decrease (right pair) between two data-points of the same process.

points. Put the other way: The farther we stray from our data points, the greater the uncertainty becomes.

What is perhaps not so obvious, is that the uncertainty can also *decrease* between two data points! The intuitive reason for this is that, if two data-points are close enough, they start to behave like two samples from the same position/distribution, making the estimate in that area actually more accurate than elsewhere, because it is based on *two* data-points instead of only one!

The technical explanation is, that the sum of two squared exponentials can exceed the maximum covariance (which is σ_f^2) that could be achieved by a single isolated data-point. This effect may be weaker or stronger for other covariance functions.

In summary, the interaction between data, prior and more data exhibits a behavior similar to that, that one would expect when estimating a mean value from a multiple measurements by computing a weighted average of the involved Gaussian random variables.

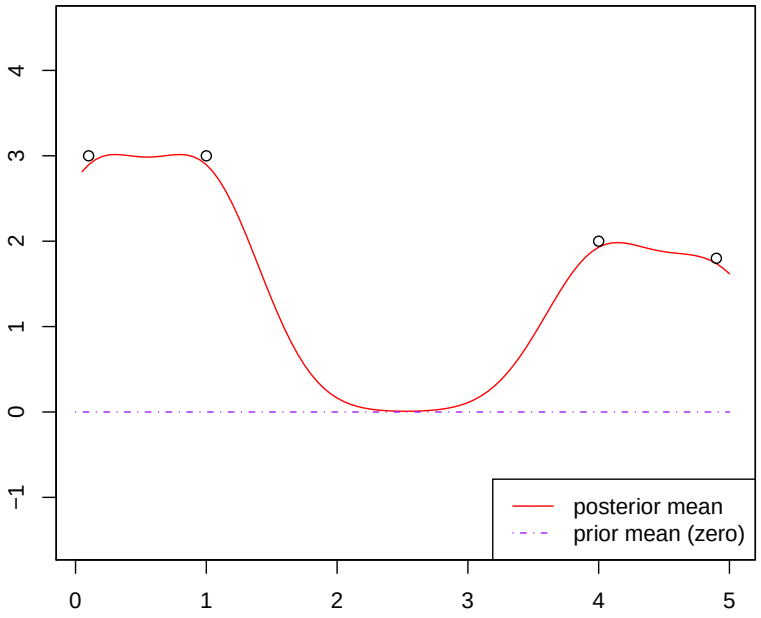

Figure 4.5: Where there is a hole in the data, the mean (red) drops to the prior (i.e. zero, purple). This is not merely a problem of a low length scale l, it might be perfectly valid for all the other data points, but still fail for places where the data is missing.

4.3.4 Prior Mean

If there are areas with missing values in the data, the prior will come into play. This is a desired effect, because it shows us how the variance rises, potentially up to σ_f^2, between two data-points.

However, we do not want the mean to suddenly drop to some prior value that is not justified. Usually, this will not happen, because we made our process have zero variance. Sometimes however, if the data is not at least roughly constant, subtracting a single constant mean is not satisfying.

In that case, more sophisticated methods should be used to model the data. Spline curves may be an option, or even a second Gaussian process! Polylines should be avoided, because their lack of smoothness would show through in the final result in the form of visible angles appearing for no reason.

When using the cell-based approach described in section 4.2.1, the mean is less of a problem, because the normalization to zero-mean is done for each cell, so dramatic

outliers or a wide range of values within one Gaussian process are improbable.

4.3.5 Optimizing the Hyperparameters

The optimization of the hyperparameters is described in 3.6.1. Since it involves not only one, but a number of matrix inversions and other operations, it is even more costly and therefore advisable to do only once, or subdivide the data into a handful meaningful partitions. Unless there are harsh discontinuities in the data that make a re-evaluation of l and σ_f^2 necessary, it should still work. When dealing with discontinuities, the squared exponential is an unsuitable choice for the covariance function anyway.

When the cell-based approach is used, it can be argued that the length value will not change that much anyway. How to choose σ_f^2 on the other hand is not clear. Should it be optimized using the whole dataset or only the data that was used for the current process? It remains an open problem.

4.4 Sampling Gaussian Processes

Being multivariate Gaussian distributions, taking a random sample function[9] from a Gaussian process is simply taking a sample from the distribution

$$\mathcal{N}\left(\vec{\mu_*}, \Sigma_*\right) \tag{4.30}$$

We want to sample the whole process, i.e. a lot of positions at once. If we were to sample point by point, each point would be independent from the previous one—ignoring the covariance of the process! To prevent this, we need the whole covariance matrix Σ_*, not just the variance that we would get by evaluating the process point by point. So, we have to use equation 4.3 with a vector of positions instead of the single position $\mathbf{x}_*$.

Standard methods for sampling a multivariate Gaussian distribution are well-known. One is:

$$\vec{s} = \vec{\mu_*} + \text{cholesky}(\Sigma_*)\,\vec{n} \tag{4.31}$$

where $\text{cholesky}(\Sigma_*)$ is the Cholesky decomposition of the matrix and $\vec{n}$ a vector of compatible length having standard normal[10] distributed scalars as their components.

[9] To be more precise: A number of points large enough to resemble a function.

[10] $\mu = 0, \sigma^2 = 1$

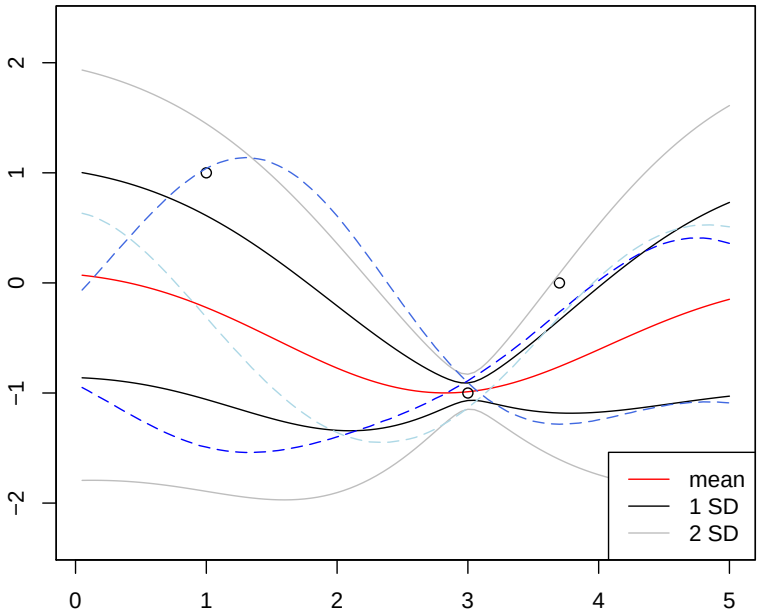

Figure 4.6: The blue dashed functions show three random samples drawn from the Gaussian process

Chapter 5

Visualizations

5.1 Gaussian processes on Real-World Data

We tested Gaussian process regression using global climate data. The data is scalar on a regular grid. For simplicity, we precomputed mean and variance from a time series consisting of 1460 time steps. The regression and visualization were implemented in the visualization system *FAnToM*[1].

For the prior variance σ_f^2, the maximum variance of the data points taken into account by the process was used. The length scale l was optimized using a smaller prior variance σ_f^2 to avoid degenerate cases where the optimization problem has no solution.

A color map of a posterior variance can be seen in fig 5.1. Its local maximums, including critical points are no longer bound to the original data points (indicated by the grid), but can now be found within cells too, where the variance rises due to the distance from the data points. Without Gaussian process regression, these features would be missing from the visualization entirely!

Execution time is typically within the 15min range using generic matrix inversion methods, not specifically optimized toward the use with Gaussian processes. Note that this step has to be done only once for each dataset. The actual visualization can simply load a previously interpolated field.

[1] FAnToM 1, rev. 10786.

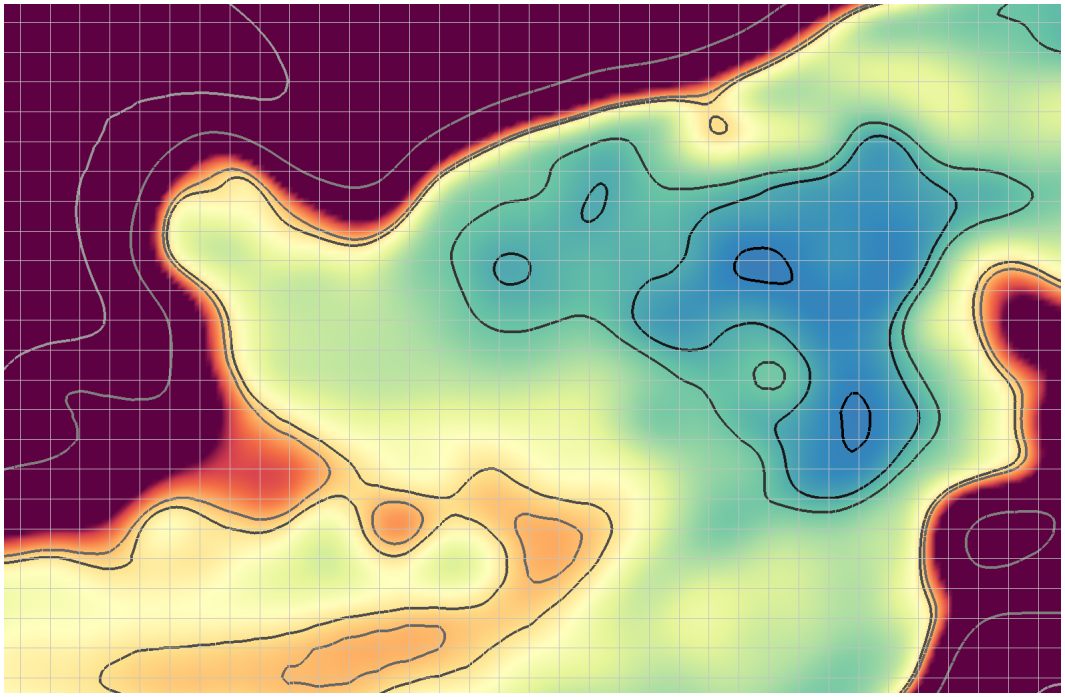

Figure 5.1: Detail of a color map with isocontours showing the posterior variance of surface temperatures in the northern Atlantic ocean. The variance field varies across the map, largely independent from the original grid.

5.2 Large scale Visualization

While color maps and isolines are useful for looking at a region of the posterior variance in detail, they do not immediately convey a sense of scale, so for the final visualization, we had to use other methods. For visualization of the data sets as a whole, we chose a fat-surface-type approach because we felt it to beneficial to be able to make out a clear surface and being able to visually estimate the amount of uncertainty.

Volume renderings (DVR) were considered, but not implemented because they would have occluded the mean surface in the volume's center too much. The mean may have a high uncertainty attached, but it is still our best estimate and should therefore be visible. Another problem with DVR is, that it would probably have been unclear whether the mean is in the center of the volume or not. In our case, it is indeed usually in the center because it follows a normal distribution, but this would not be the case when the process is used for log-normal distributions. Also, the average viewer might be unaware of the underlying probability density function, e.g. if they do not know that the uncertainty data comes from a Gaussian model.

The first visualization method we tested consists of a mean surface and surfaces marking the distance of two standard deviations (SD) from the mean surface. Those *SD surfaces* are almost opaque and color-coded using a black-blue-magenta-yellow-white gradient with the colors uniformly distributed over the full range of variance. The color coding is necessary, because the total height is influenced by both the variance and the mean surface. Therefore, height-coding only would be usually be misleading.

One major problem here is that, much like in DVRs, the colors seamlessly blend into one another, making it hard to tell the two surfaces apart, except when looking at them from a very shallow angle. One possibility to improve the visualization is to give the mean a shiny finish and the SD surfaces a more dull appearance. But even then, the visualization is less successful due to self-occlusion, creating distracting patches of higher opacity. A convincing visualization is only possible by constantly moving the view to get a three-dimensional impression.

The second alternative was using a *grid* instead of a solid surface for the SD surfaces. It looks unusual, but suffers from neither the occlusion problem nor the blending problem because the grid is easily discernible from the underlying surface and the lines are thin, allowing a good three-dimensional look at the mean. The visible parallax helps to judge the position of the grid over the mean surface too.

It also leaves enough space to add further visualization aids such as the black

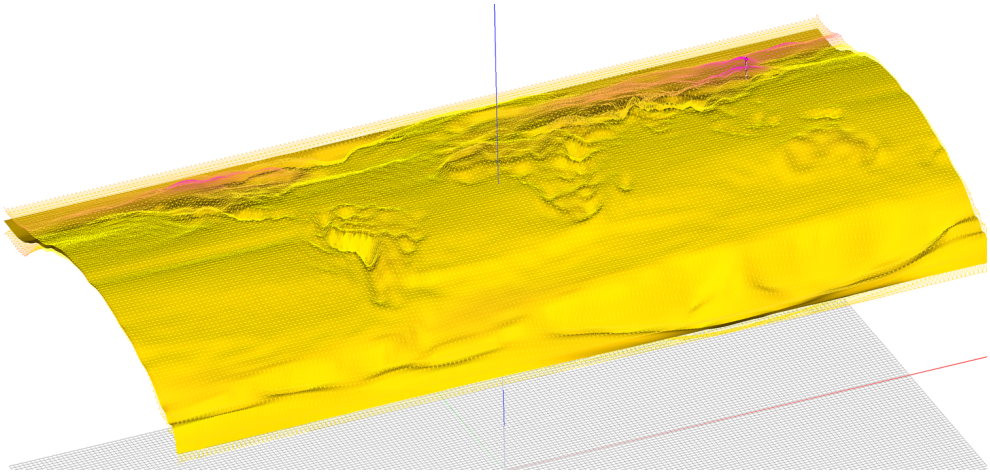

Figure 5.2: Overview of an uncertain scalar field using a grid instead of a solid surface for the +2 standard deviations surface.

and white scale pictured in fig. 5.3. It pinpoints the position with the highest uncertainty and gives an impression of scale. It is designed to always be divided into five sections that are each five "units" long. "Units" meaning a power of ten. Depending on the standard deviation at that point, that power is chosen such that the distance of the SD surface is between one and five sections long. This is done to ensure that there are not to many or too few segments in one place, avoiding visual clutter. The sphere marks the intersection with the SD surface (which is usually not at the end of the scale).

The total height of the SD surface vertices depends on the mean, which can vary wildly, making it hard to reliably spot areas of large uncertainty. One way to avoid this problem is to color-code the grid, as we have done. This makes uncertain areas pop out. On the small scale however, markers like the one described above are vitally important, because the number of colors that can be told apart is very limited. The marker in fig. 5.3 marks the highest uncertainty in the grid. Had we relied in color and height coding only, we would have intuitively placed it a couple of cells next to where it actually is.

To visualize the uncertainty in-between data points, the visualization has to be higher resolution than the original grid. In our experience, Gaussian processs tend to produce variance functions that either increase *or* decrease between data points—not both. So, the refinement can be very conservative, with only a handful of additional samples inside the cells of the grid.

A third, experimental approach we tried was using a cloud of particles pointing outward along the normal of the surface to create an effect similar to that of

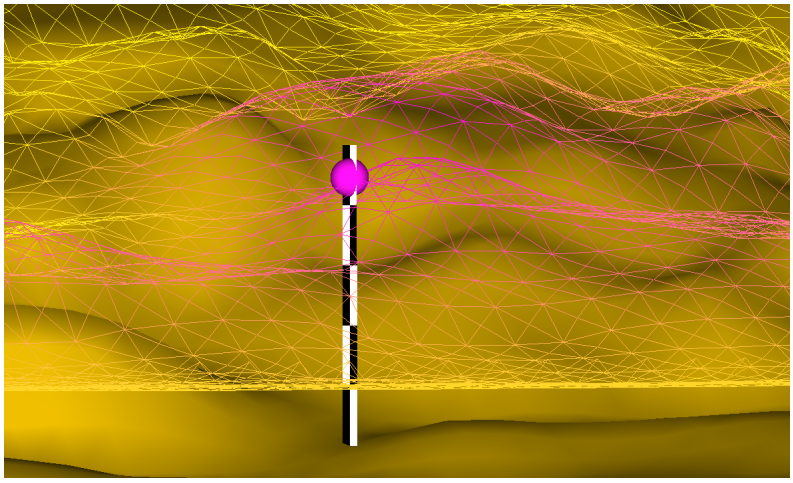

Figure 5.3: A 5×5 units long marker at the point of highest uncertainty. The fact that it is at a place that we would not intuitively have guessed, underlines the need for these kinds of markers.

illustrative visualizations. It is a compromise of having the mean shine through and still be able to see the shape and color of the SD surface. However, different needs call for different visualizations and there is probably no single correct answer to how fat surfaces ought to be implemented. What will undoubtedly be helpful is interactivity, putting different visualizations side by side and be able to explore and manipulate the visualization in real time.

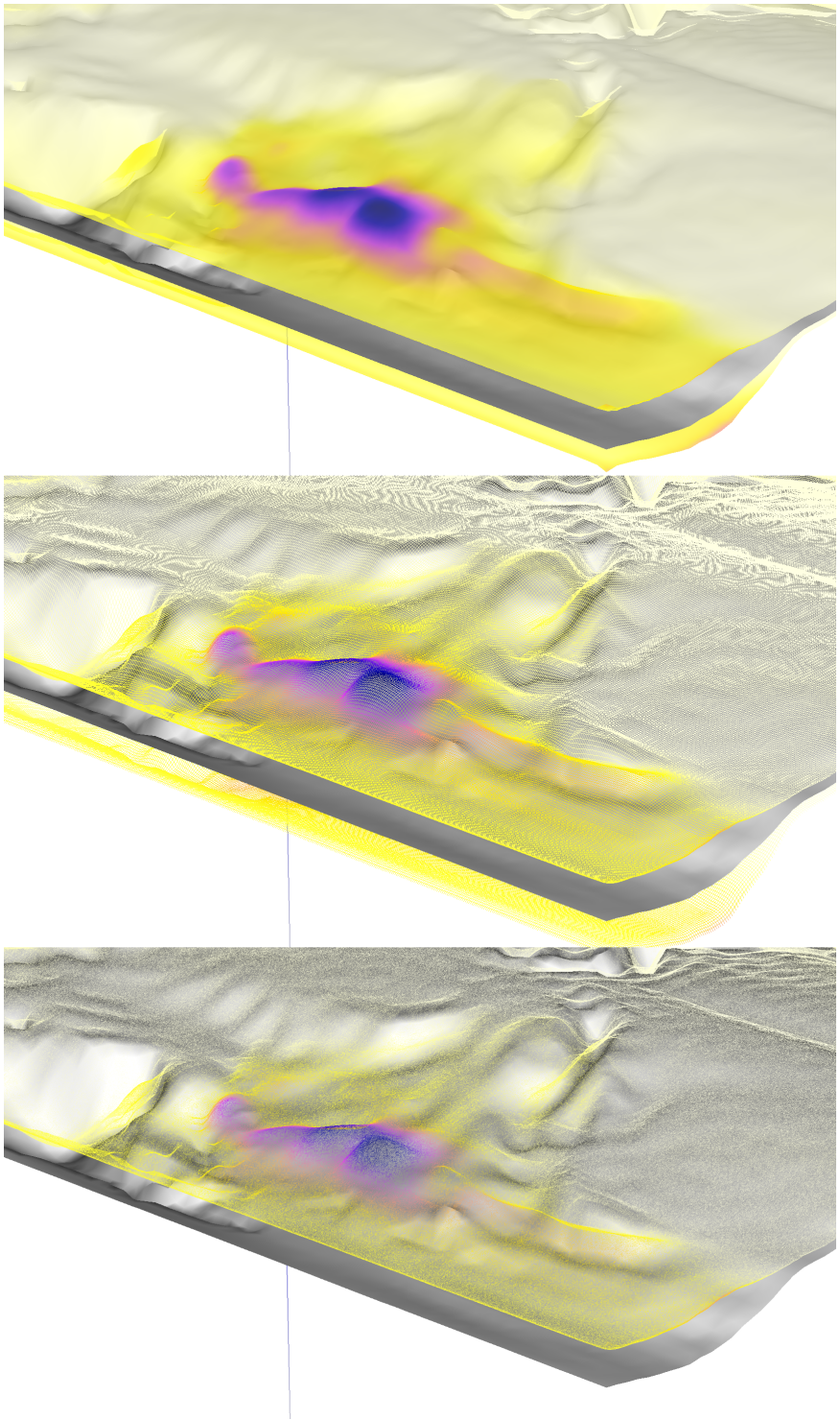

Figure 5.4: Comparing three large scale visualization methods. Top: 80% opaque surface. Center: Grid (of the posterior variance field). Bottom: Particle Cloud.

Chapter 6

Discussion

6.1 Summary

In this thesis, we introduced Gaussian processes, a statistical framework that allows for the smooth interpolation of data with heteroscedastic uncertainty through regression. Its theoretical background makes it a convincing method to analyze uncertain data and create a model of the underlying phenomenon and, most importantly, the uncertainty at and in-between the data points. For this reason, it is already popular in the GIS community where it is known as *Kriging* but has applications in machine learning too.

In contrast to traditional interpolation methods, Gaussian processes do not merely create a surface that runs through the data points, but respect the uncertainty in them. This way, noise, errors or outliers in the data do not disturb the model inappropriately. Most importantly, the model shows the variance in the interpolated values, which can be higher but also lower than that of its neighboring data points, providing us with a lot more insight into the quality of our data and how it influences our uncertainty! This enables us to use uncertainty information in algorithms that need to interpolate between data points, which includes almost all visualization algorithms.

We also took a closer look at a number of peculiarities of Gaussian process regression, such as the interaction between data and data, and data and prior. Gaussian processes turned out to produce a model that is a weighted sum of data, prior and other data, weighted by their certainty. This allows us to combine data from multiple different sources without the need of preprocessing. We also presented a simple way to remove the influence of the prior variance if needed.

Gaussian processes are not only conceptually simple to understand but also relatively easy to implement. To make them practical for real-world problem sizes though, we gave two ways to optimize the regression to limit the computational load incurred by the need to invert matrices. One being limiting the number of Gaussian processes and therefore matrix inversions to one per cell, the other being cutting off the covariance function.

Furthermore, Gaussian processes work without a grid! The data can be a point cloud with no topology or neighborhood information whatsoever as long as there is a meaningful covariance function.

6.2 Future Work

The look of the variance curves suggests that there might be an analytic solution in terms of polynomials. Indeed there seem to be solutions for the equivalent kernel, but to our knowledge, there is none for the variance. An analytic form would also speed up computation and allow for dynamic LOD. It might also give us more insight into the relationship of data and uncertainty and their relationship to traditional interpolation methods.

Another next step would be to adapt the method for use with other models that exhibit non-gaussian noise. Some phenomena, while not normal-distributed themselves, are *based* on Gaussian random variables. In the case of MRI data for instance, the magnitude to be visualized is Rice distributed, which is the distribution of the length of a vector of two Gaussian variables. In this case, it might be possible to acquire data about those underlying variables, perform Gaussian process regression on them and use the results to compute the final distribution. Other phenomena may have noise distributions that are entirely different still.

A last important question is how Gaussian processes behave when applied to the components of vectors using a covariance function that respects possible correlations between them. If a covariance structure was found that properly modeled the relationships between the components of a vector, Gaussian processs may become a useful tool for vector data too, providing valuable insight into the behavior of uncertainty in vector fields!

Bibliography

[AT10] Robert J. Adler and J. E. Taylor. Topological complexity of smooth random functions. 2010.

[BWE05] R. P. Botchen, D. Weiskopf, and T. Ertl. Texture-based visualization of uncertainty in flow fields. pages 647–654, October 2005.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd revised edition edition, September 2001.

[Cou03] Helen Couclelis. The certainty of uncertainty: Gis and the limits of geographic knowledge. *Transactions in GIS*, 7(2):165–175, 2003.

[CR00] Andrej Cedilnik and Penny Rheingans. Procedural annotation of uncertain information. In *Proceedings of the conference on Visualization '00*, VIS '00, pages 77–83, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[DKLP02] Suzana Djurcilov, Kwansik Kim, Pierre Lermusiaux, and Alex Pang. Visualizing scalar volumetric data with uncertainty. *Computers & Graphics*, 26(2):239–248, April 2002.

[Eva97] B. J. Evans. Dynamic display of spatial data-reliability: does it benefit the map user? *Computers & Geosciences*, 23(4):409–422, 1997.

[Ger98] Nahum Gershon. Visualization of an imperfect world. *IEEE Computer Graphics and Applications*, 18:43–45, 1998.

[GR02] Gevorg Grigoryan and Penny Rheingans. Probabilistic surfaces: point based primitives to show surface uncertainty. In *Proceedings of the conference on Visualization '02*, VIS '02, pages 147–154, Washington, DC, USA, 2002. IEEE Computer Society.

[GWB] P. W. Goldberg, C. K. I. Williams, and C. M. Bishop. Regression with input-dependent noise: A gaussian process treatment.

[Hen03] T. Hengl. Visualisation of uncertainty using the hsi colour model: computations with colours. In *Proceedings of the 7th International Conference on GeoComputation*, pages 8–17, 2003.

[HT06] Tomislav Hengl and Norair Toomanian. Maps are not what they seem: representing uncertainty in soil-property maps. In M. Caetano and M. Painho, editors, *7th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences*, pages 805+, 2006.

[KPPB07] Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most likely heteroscedastic gaussian process regression. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 393–400, New York, NY, USA, 2007. ACM.

[KVUS+05] J. M. Kniss, R. Van Uitert, A. Stephens, G. S. Li, T. Tasdizen, and C. Hansen. Statistically quantitative volume visualization. pages 287–294, October 2005.

[LB98] Adriano Lopes and Ken Brodlie. Accuracy in 3d particle tracing. In *In Hans-Christian Hege and Konrad Poltheir, editors, Mathematical Visualization*, pages 329–341, 1998.

[LC97] Kim E. Lowell and Pav Casault. Outside-in, inside-out: Two methods of generating spatial certainty maps. In *University of Otago*, pages 26–29, 1997.

[LFLH07] Hongwei Li, Chi-Wing Fu, Yinggang Li, and A. J. Hanson. Visualizing large-scale uncertainty in astrophysical data. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1640–1647, November 2007.

[LLPY07] Claes Lundstrom, Patric Ljung, Anders Persson, and Anders Ynnerman. Uncertainty visualization in medical volume rendering using probabilistic animation. *IEEE Transactions on Visualization and Computer Graphics*, 13:1648–1655, 2007.

[OM02] C. Olston and J. D. Mackinlay. Visualizing data with bounded uncertainty. *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pages 37–40, 2002.

[PH11] Kai Pothkow and Hans-Christian Hege. Positional uncertainty of isocontours: Condition analysis and probabilistic measures. *Visualization and Computer Graphics, IEEE Transactions on*, 17(10):1393–1406, October 2011.

[PMG04] M. Pauly, N. Mitra, and L. Guibas. Uncertainty and variability in point cloud surface data, 2004.

[PWL97] Alex T. Pang, Craig M. Wittenbrink, and Suresh K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, November 1997.

[RJ99] Penny Rheingans and Shrikant Joshi. Visualization of molecules with positional uncertainty. In *Data Visualization '99, Proceedings of the Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization*, volume 299, pages 299–306, 1999.

[RW06] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[SCB+04] Greg S. Schmidt, Sue L. Chen, Aaron N. Bryden, Mark A. Livingston, Lawrence J. Rosenblum, and Bryan R. Osborn. Multidimensional visual representations for underwater environmental uncertainty. *IEEE Computer Graphics and Applications*, 24:56–65, 2004.

[SMI99] Thomas Strothotte, Maic Masuch, and Tobias Isenberg. Visualizing knowledge about virtual reconstructions of ancient architecture. *Computer Graphics International Conference*, 0:36–43, 1999.

[SZB+09] Jibonananda Sanyal, Song Zhang, Gargi Bhattacharya, Phil Amburn, and Robert Moorhead. A user study to compare four uncertainty visualization methods for 1d and 2d datasets. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1209–1218, November 2009.

[VRNDW10] S. Vasudevan, F. Ramos, E. Nettleton, and H. Durrant-Whyte. Heteroscedastic gaussian processes for data fusion in large scale terrain modeling. pages 3452–3459, May 2010.

[War04] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[WSF+95] Craig Wittenbrink, Elijah Saxon, Jeff J. Furman, Alex Pang, and Suresh Lodha. Glyphs for visualizing uncertainty in environmental

vector fields. In *IEEE Transactions on Visualization and Computer Graphics*, volume 2410, pages 266–279, 1995.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

Leipzig, den 29. September 2011