Vision, Modeling, and Visualization (2015)
D. Bommes, T. Ritschel, and T. Schultz (Eds.)

Interactive GPU-based Visualization of Scalar Data with Gaussian Distributed Uncertainty

S. Schlegel[1], M. Goldau[1] and G. Scheuermann[1]

[1]Image and Signal Processing Group, University of Leipzig, Germany

Abstract

We present a GPU-based approach to visualize samples of normally distributed uncertain, three-dimensional scalar data. Our approach uses a mathematically sound interpolation scheme, i.e., Gaussian process regression. The focus of this work is to demonstrate, that GP-regression can be used for interpolation in practice, despite the high computational costs. The potential of our method is demonstrated by an interactive volume rendering of three-dimensional data, where the gradient estimation is directly computed by the field function without the need of additional sample points of the underlying data. We illustrate our method using three-dimensional data sets of the medical research domain.

Categories and Subject Descriptors (according to ACM CCS):
Mathematics of Computing [G.1.0]: General—Error analysis Mathematics of Computing [G.1.1]: Interpolation—Interpolation formulas Computer Graphics [I.3.3]: Picture/Image Generation—Display algorithms

1. Introduction

In the field of scientific data visualization, data are typically given at a set of sample points on a two- or three-dimensional domain. Interpolation is used to compute values for each arbitrary point within the domain. Often, linear interpolation is used without further consideration of the effect on the resulting continuous data. While this approach is often feasible for densely sampled data, it might be misleading for other samplings. In the case of uncertain data that is Gaussian distributed, linear interpolation does not incorporate the uncertainty and, thus, yields misleading or even wrong results. In prior work [SKS12] we presented an interpolation scheme based on Gaussian process regression to approach the aforementioned problems.

However, the computation of this interpolation is very demanding and therefore the practical relevance is very limited. In this paper, we address this problem and show that it is possible to perform Gaussian process regression in reasonable time by combining different acceleration techniques. As a case study, we perform interactive volume rendering of medical scalar datasets, where we visualize the interpolated mean of the scalar field as well as the behavior of the interpolated variance. We implemented the volume renderer in OpenCL.

Moreover, we introduce explicit interpolation formulas for Gaussian process regression. These formulas are similar to standard interpolation schemes utilizing basis functions. This explicit representation allows us to compute exact derivatives of the uncertain field based on the same sample points. This can be done by differentiating the basis functions of the interpolation scheme instead of the field itself. We will use the derivatives to shade the volume rendering.

2. Related Work

When creating visualizations of scalar data, we should be aware of the fact that uncertainty can reside in the data value or in the position of the data point or in both [PWL96].

To deal with uncertainty, Pöthkow et al. [PH11] presented an uncertain counterpart for isocontours [LC87]. Therefore, they calculate the so called level-crossing probability (LCP). In a given interval, the probability is computed that a certain threshold is crossed within. Therefore, they interpolated the expected values and the roots of the central moments to interpolate the probability density function. In [SKS12] we proposed the method of Kriging to interpolate the mean and the variance in an uncertain Gaussian field. They also applied their method to com-

DOI: 10.2312/vmv.20151257

pute LCP. [AE13] analyzed the effects of uncertainty to linear interpolation and isosurface extraction. The extension of [PH11] to correlated data was done in [PWH11]. To reduce the heavy computation time (mainly caused by Monte Carlo Sampling), two methods called maximum edge crossing probability and linked-pairs to approximate the level-crossing probabilities were introduced in [PPH13]. To overcome the restrictions of predefined probability distributions [PH13] introduced nonparametric models (empirical distributions, histograms, and kernel density estimators) to compute the probability of features in an uncertain field. Based on [PH11], Pfaffelmoser et al. [PRW11] developed an algorithm to compute the so called isosurface first crossing probability. It is an algorithm that incrementally uses a front-to-back volume ray casting to visualize that probability. The rendering is enriched by additionally depicting surfaces of the stochastic distance function (SDF-surfaces). Additional work to compute the gradient of the probability density function of uncertain 3D scalar fields was done in [PMW12]. Kniss et al. [KVUS*05] try to perform classification of medical volume data under uncertainty. They base their transfer function on what they call the decision boundary distance that is 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.

In this paper, we perform volume rendering on the GPU. In [KW03], acceleration methods like early ray termination and empty space skipping are employed. To combine the uncertainty of the data values information into volume rendering, Djurcilov et al. [DKLP01] proposed two methods. In the first method, the uncertainty is incorporated directly into the volume rendering equation, i.e. the transfer function. The other method is to do a post processing on the resulting image, for example by mapping a noise texture on locations with uncertainty. This also the method of choice, we will use in this paper. The texture mapping approach was also used in [RLBS03] to combine isosurface rendering with uncertainty information. In another approach, they used color based on hue, saturation and brightness and give the user the opportunity to map the uncertainty to the given quantities. In the field of Geo-Information science, the technique to display uncertainty using particular color models is used for example in [Hen03]. In [RJ99] volume rendering was used to depict the positional uncertainty of molecules. The key aspect was to blur the molecule position. This concept of blurring was also used in [GR04]. Here the authors used point primitives to disguise the "real" position of a surface. Instead of using blurring or fat surfaces (implemented as multiple iso surfaces or volume rendering), Zehner et al. [ZWK10] chose to add extra geometry to isosurface rendering to depict positional uncertainty. A different approach to display uncertainty in volume rendering was done by Lundström et al. [LLPY07]. They used animation of medical volume data to show a range of images created with different transfer functions used for classification of different types of tissue.

3. Foundation

In this section, we want to to employ an interpolation scheme based on weighted sums of basis functions for the interpolation of Gaussian distributed variables. In most cases, we state the formulas without proof. If the reader is interested in the derivation of some of the statements, we refer to the appendix sections A and B.

In general, interpolation defines values at arbitrary points in a given domain using a set of sampled data values. While there are many approaches for deterministic data, random variables cannot be interpolated using standard techniques, see [Bur96]. If these samples are Gaussian-distributed random variables, it is suitable to use the concept of *Gaussian processes*. Interpolating random variables in a Gaussian process is also known as *Kriging* [Kri51] or *Gaussian process regression* [RW06].

A Gaussian process f given on a domain S defines a Gaussian distributed random variable at any position $s \in S$. It is defined by a mean function at every position s and a covariance function between any two positions s and s', e.g., see [AT11]:

$$\begin{aligned} \mu : S \mapsto \mathbb{R} \quad & \mu(s) = \mathbb{E}[f(s)], \\ k : S \times S \to \mathbb{R} \quad & k(s,s') = \mathbb{E}[(f(s)-\mu(s))(f(s')-\mu(s'))], \end{aligned} \tag{1}$$

where the mean function is assumed to be constant. This defines a random variable $X(s)$ at position s with mean $\mu(s)$ and variance $\sigma^2(s) = k(s,s)$.

Basically, using the Kriging approach, one assumes the prior distribution for any unobserved position in the domain and adapts it according to the covariance function and the given samples. Hence, the prior distribution is turned into a posterior distribution. The posterior distribution can be interpreted as the resulting variable conditional to the observations. In contrast, the prior distribution is an assumption of the actual value at the unobserved locations in our domain. Typically, it is chosen as constant over the whole domain. If we do not have any information about a prior distribution for the domain, we can specify the prior mean using the empirical mean of the domain. The prior variance can be interpreted as the variance of the data acquisition method that we used to gather the existing measurements, i.e. the scalar field. If there is no specific information on the prior variance, we can estimate it from the given data. The maximum variance in the data set is an obvious choice for the prior variance, because the variance of the data acquisition method is at least as big as the maximum variance residing in the data set.

Previously we remarked in [SKS12] that, choosing the covariance function is equal to choose an interpolation scheme, since the covariance function models the spatial interdependence of the random variables. Throughout this paper, we

use the squared exponential covariance function. Of course, our method also applies to other covariance functions. The squared exponential is given by

$$k(s,s') = \sigma_p^2 exp(-\frac{1}{2l^2}|s-s'|^2), \quad (\sigma_p^2, l > 0). \tag{2}$$

The parameters σ_p^2 (the aforementioned prior variance) and l (length scale) are hyperparameters. Gaussian processes, as well as the optimization of the hyperparameters, are discussed in detail in [RW06].

Let S be sampled with N Gaussian distributed variables at positions s_i, $i = 1,\ldots,N$, with $X(s_i) = X_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right)$ and the covariance function $k(s,s')$. Then one can calculate the covariances between those sample points and generate the covariance matrix

$$K = \begin{pmatrix} k(s_1,s_1)+\sigma_1^2 & \ldots & k(s_1,s_N) \\ \ldots & \ldots & \ldots \\ k(s_N,s_1) & \ldots & k(s_N,s_N)+\sigma_n^2 \end{pmatrix}, \tag{3}$$

The posterior distribution at position s is then defined as

$$X(s) \sim \mathcal{N}\left(\vec{k}(s)^T K^{-1}\vec{\mu}_i, \quad k(s,s) - \vec{k}(s)^T K^{-1}\vec{k}(s)\right), \tag{4}$$

with $\vec{\mu}_i$ being the vector of the means of the sampled X_i and $\vec{k}(s) = (k(s,s_1),\ldots,k(s,s_N))^T$. It can be shown that the variance of the posterior distribution is minimized, when interpolating $X(s)$ in that way.

Our goal is now to turn Eq. 4 into a form that is more similar to an interpolation approach. In detail, we want to write $X(s)$ as a sum of basis functions $\phi_i(s)$ multiplied with the given uncertain samples X_i. Additionally to the given distributions X_i, we denote the prior distribution with $X_0 \sim \mathcal{N}(0,\sigma_p^2)$. By defining the basis functions

$$\phi_i(s) = \begin{cases} -1, & \text{if } i = 0 \\ \sum_{j=1}^N (K^{-1})_{ij} k(s_j,s), & \text{otherwise} \end{cases} \tag{5}$$

we can write the interpolation of $X(s)$ as

$$X(s) = \sum_{i=0}^{N} X_i \phi_i(s). \tag{6}$$

with

$$\begin{aligned} X(s) \sim \mathcal{N}\Big(&\mu(s) = \sum_{i=1}^{N} \phi_i(s)\mu_i, \\ &\sigma^2(s) = k(s,s) - \sum_{i=1}^{N} \phi_i(s) k(s_i,s)\Big). \end{aligned} \tag{7}$$

Using Eq. 6, we can calculate the derivative of the moments of $X(s)$ by differentiating the basis functions:

$$\begin{aligned} \frac{\delta\mu(s)}{\delta s^{(n)}} &= \mathbb{E}\left(\sum_{i=1}^{N} \frac{\delta\phi_i(s)}{\delta s^{(n)}} X_i\right), \\ \frac{\delta\sigma^2(s)}{\delta s^{(n)}} &= \sum_{i=1}^{N}\sum_{j=1}^{N}\Big(\frac{\phi_i(s)}{\delta s^{(n)}}\phi_j(s)Cov(X_i,X_j) \\ &\quad + \phi_i(s)\frac{\phi_j(s)}{\delta s^{(n)}}Cov(X_i,X_j)\Big), \end{aligned} \tag{8}$$

see Section A.

Here, $s^{(n)}$ $(n = 1,2,3)$ denotes the n-th dimension of position vector s. We want to use the derivative of the interpolated variable, to enable lighting in direct volume rendering of the scalar field. Using the squared exponential, the derivative is

$$\frac{\delta\phi_i(s)}{\delta s^{(n)}} = \sum_{j=1}^{N} \frac{-1}{2l^2}(2s^{(n)} - 2s_j^{(n)})k(s,s_j)(K^{-1})_{ij}, i > 0. \tag{9}$$

The goal of this section was to show, that the interpolation of uncertain Gaussian distributed variables can be done as high dimensional linear interpolation and that differentiating the basis functions of that interpolation enables us to differentiate the interpolated Gaussian variable.

4. Method

4.1. Cell-Based Computation

The reason why Gaussian process regression performs poorly on many datasets, is the storage and the inversion of the covariance matrix. The first step, to reduce the computational requirements, is the use of many small Gaussian processes (and thus covariance matrices) instead of one large process. For regular sampled datasets, it is feasible to create a small Gaussian process for each grid cell composed of the data points lying in a $3l + d$ radius of the barycenter of the cell. Where l is the length scale of the covariance function and d is the diameter of the cell. This approach is described in more detail in [SKS12]. The result is, that we have to invert one relatively small covariance matrix for every cell instead of one large matrix, which can also be done in parallel for another speed up. From now on, we use the quantity M to describe the (average) size of the cell caches for the rest of the paper. This quantity is not to be confused with the actual number of data points in our data set, which in general is much higher.

4.2. Caching

Caching the much smaller inverted covariance matrices for each cell gives a major speedup compared to the naive approach. Nevertheless, we still have to perform $O(M^2)$ computations to calculate either the mean or the variance. But having employed the concept of cell caches, we can further

reduce complexity for the visualization of the mean. By inserting Eq. 5 into Eq. 7, we get

$$\mu(s) = \sum_{j=1}^{M} \left(\sum_{i=1}^{M} (K)_{ij}^{-1} \mu_i \right) k(s_j, s) \tag{10}$$

The inner sum in Eq. 10 only depends on the data itself and therefore can be precomputed. So for each cell, we store the M sums and are able to interpolate the mean at any position by performing $O(M)$ computations. This cache has to be computed only once for every data set and can be reused whenever it is needed.

Unfortunately, a similar approach is not possible in order to cache the calculation of the variance. To store such a variance cache, we would need $O(NM^2)$ memory, which is not suitable. However, [BFRD13] propose an estimation scheme using a special diagonal matrix D instead of K, which gives an upper bound for the resulting variance. D is calculated as

$$(D)_{jj} = \sum_{i=1}^{M} (K)_{ij} \tag{11}$$

We can store the M diagonal entries of D and calculate the predictive variance as

$$\tilde{\sigma}(s) = \sum_{j=1}^{M} \frac{k(s, s_j)^2}{(D)_{jj}}. \tag{12}$$

This prediction is useful to investigate the behavior of the variance because the authors show that the change of the exact variance is similar to the change of the predictive variance.

4.3. Empty Space Skipping

Another very efficient way to reduce the computation time is to use some kind of empty space skipping. If the values at all positions that define a local Gaussian process fall below a certain threshold with a certain probability, we do not need to compute the inverted covariance matrix, if those values can be considered as noise. Similarly, we do not need to evaluate the interpolated Gaussian variable when performing the volume rendering. This gives a huge speedup especially in the precomputation step.

4.4. Optimize Memory Consumption

To analyze the memory consumption, we first have to evaluate some implementation details.

When the precomputation steps are performed, we have stored a value for the mean cache and a value for variance cache for every grid point in the local processes, i.e. for every grid cell. Since a grid point usually is part of multiple local processes, we also store multiple cache values for every grid point. This data is stored in two arrays (one for the mean cache and one for the variance cache) and is sent to the GPU. Furthermore, to evaluate the covariance function, we need to send the position of every grid point to the GPU, which is also stored in an array. To avoid the multiple storage of the same position for several local processes, we send an index array to the GPU which holds indices of the position array. This way, we have to store every position only once on the GPU. To match the entries in the mean cache array, the variance cache array and the index array to the entries of the local process, we also create an array which holds an offset into these three arrays for every cell, i.e. local process.

Now we are able to evaluate the following formulas on the GPU for every sample point of the raycasting at position s in cell c in order to compute the mean, variance and the derivative of the mean:

$$\begin{aligned}
\mu(s) &= \sum_{j=\text{cellOffset}[c]}^{\text{cellOffset}[c]+M} k(s, s[j])\tau[j], \\
\sigma^2(s) &= \sigma_p^2 - \sum_{j=\text{cellOffset}[c]}^{\text{cellOffset}[c]+M} \frac{k(s, s[j])^2}{\upsilon[j]}, \\
\frac{\mu(s)}{\delta s^{(n)}} &= \frac{-1}{l^2} \sum_{j=\text{cellOffset}[c]}^{\text{cellOffset}[c]+M} k(s, s[j])\tau[j](s^{(n)} - s[j]^{(n)})
\end{aligned} \tag{13}$$

with

$$k(s, s[j]) = \sigma_p^2 exp(-\frac{1}{2l^2}|s - \rho[\kappa[j]]|^2) \tag{14}$$

where τ is the array of the mean cache, ρ is the array of positions, κ is the array of position indices and υ is the array of the variance cache. The prior variance σ_p^2 as well as the length scale l are sent to the GPU. The sample position s is already given by the raycasting. We see, that the memory consumption on the GPU mainly depends on the arrays for the mean cache, the variance cache and the indices. Each of them uses memory of amount $O(NM)$.

By using the fact, that we operate on a regular grid, we can compute the position for every grid point of a local process. This makes the storage of the positions and therefore the indices obsolete. We simply store an array containing point offsets. The illustration in Fig. 1 shows, how the point offsets are calculated on a 2D grid. The 1st point of the cell in the center of the local process gets the offset $(0,0,0)$. All other points get an offset corresponding to their position relative to the base point. This array can be reused for every local process since all local processes have the same length scale. In order to compute Eq. 13, we have to determine in which cell the position s lies. The next step is to calculate the global index of the base point of that cell (i.e. at which place in the ordered array of the grid positions lies the base point). The global index of the position we need ($s[j]$), is base_point_index + index_offset. Eventually, we can calculate the position using the computed index and the field dimensions. The formulas for this calculation are presented in detail in appendix section C.

We further reduced memory consumption by using a

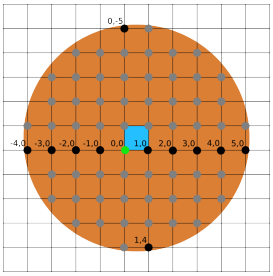
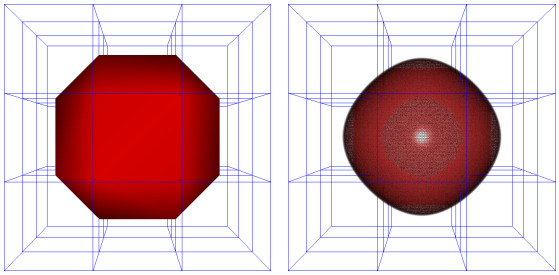


Figure 1: *Illustration of the numbering scheme for the index offsets inside a local process. The blue grid cell is where the local process is defined. The influence radius of the local process is depicted by the orange circle. The green dot represents the base point. The indices for the grey grid points were omitted in order not to clutter the illustration. The 3D case is handled accordingly with index* $(0,0,0)$ *for the base point.*



(a) (b)

Figure 2: *2(a): Isosurface of low sampled sphere function. 2(b): Volume Rendering of 2(a). Additionally, the variance is depicted as noise.*

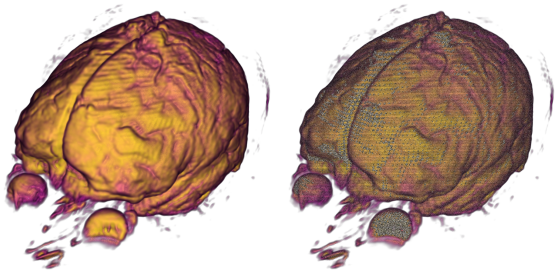
16 bit index to store each the mean cache and the variance cache. Therefore, we quantize the values for the caches and use the index to calculate the cache value. This is in general more accurate than using the half float data type which also needs 16 bit. We pack the two 16 bit values for each cache entry into a 32 bit number. This reduces the global memory access on the GPU when fetching the entries from GPU memory.

All in all, we managed to reduce the memory footprint by a factor greater three. For example, the memory usage for the aneurysm dataset in Sec. 5 drops from 1590 MB to 490 MB.

5. Results

5.1. Synthetic Dataset

The first dataset we show, is the volume rendering of a sphere function $f(s) = \sqrt{s^{(0)}s^{(0)} + s^{(1)}s^{(1)} + s^{(2)}s^{(2)}}$ sampled on a $3\times3\times3$ regular grid centered on position $(0,0,0)^T$.



(a) (b)

Figure 3: *Shaded volume rendering of the mean of the MRI dataset (Fig. 3(a)). The normals for the lighting were calculated on the GPU using the derivative of the basis functions. In Fig. 3(b), we depicted the interpolated variance approximation on the rendered surfaces using noise.*

We used a constant variance of 5 to model the uncertainty. To demonstrate the reconstruction properties and the smoothness of the derivatives, we did a volume rendering of the isosurface given in Fig. 2(a).

Fig. 2 shows the result of the volume rendering. As covariance function, we used the squared exponential with a length scale of 91% of the cell length. The prior variance was computed as the maximum variance in the dataset, i.e. 5. We interpolated the scalar values as the means of the sampled Gaussian distribution and calculated the gradient of those means to enable pixel precise Phong shading using Phong lighting on the volume rendering. Fig. 2(b) shows the result of the interpolation. Dithered noise (see [PWL96]) was used to visualize the behavior of the variance. This reveals, that the variance (and hence the uncertainty) in the interpolated image is low at the grid points and high within the grid cells, see [SKS12].

5.2. Medical Datasets

We will visualize a brain MRI dataset (T2-weighted images). The dataset consists of seven consecutively recorded images of the same person. We extracted the empirical mean and the empirical variance out of that image sequence. The data is given on a regular $160\times200\times160$ grid with a spatial resolution of 1 mm. The chosen covariance function has a length scale of 0.7 mm and the prior distribution is calculated as in Section 5.1. To generate the image in Fig. 3(a), we incorporated lighting into the volume rendering by calculating the derivative of the mean directly on the GPU. We also incorporated the variance approximation into the volume rendering of the interpolated mean (Fig. 3(b)). The sources of the variances are mainly motion artifacts, especially in places that fade into water-filled areas.

Another example for the volume rendering of a medical dataset is shown in Fig. 4. We depicted the pressure of a

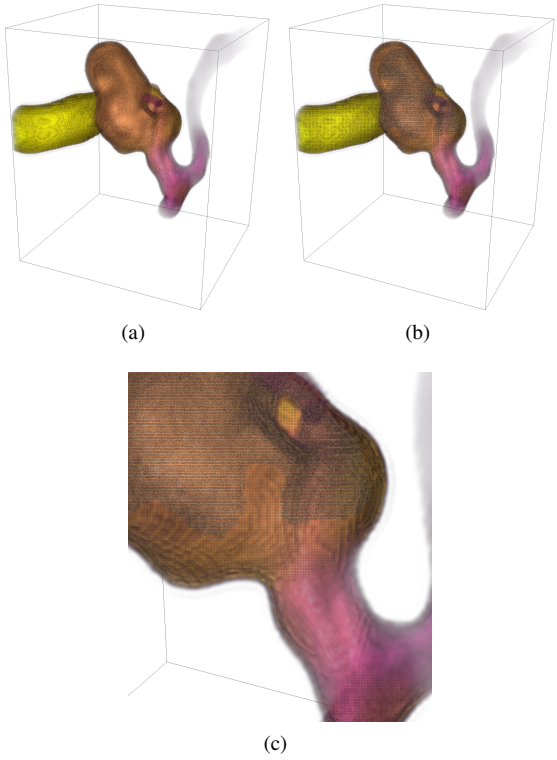


Figure 4: *Volume rendering of the pressure field of an aneurysm dataset. Fig. 4(a) shows a shaded volume rendering of the interpolated mean field. On Fig. 4(b), we additionally depicted the variance as noise, which can be seen in more detail in Fig 4(c).*

Table 1: *Rendering speed of the two presented datasets at a resolution of* 1170×740*. The minimum fps was measured at full screen coverage, whereas the maximum fps were measured at roughly 30% screen coverage. The precomputation of the caches took roughly 10 minutes on our test system (Intel I7 980 @3.3GHz).*

	Geforce Titan	Geforce GTX 660TI
Brain	3.7 fps - 8.4 fps	2.5 fps - 7.3 fps
Aneurysm	6.3 fps - 16.1 fps	4.2 fps - 12.1 fps

blood flow velocity field in an aneurysm dataset. The uncertainty in the vector field originates from an ensemble of simulation results forced by 9 different parameter configurations. We resampled the original field on a regular 180×148×163 grid and performed the interpolation using a squared exponential covariance function with a length scale of the width of one grid cell.

5.3. Discussion

The main advantages of Gaussian processes for the interpolation of uncertain data are the mathematical sound basics, the reconstruction properties of low sampled datasets (see Sec. 5.1), and the variance minimization. In the majority of cases, we get a better approximation for the samples (in terms of less variance) than the original field provided. On top of that, it provides a closed analytical form for the derivative of mean and variance at every position in our domain.

However, the practical use of the proposed method heavily depends on the dataset and what the user is really interested in. This type of uncertainty interpolation is, regardless of the presented acceleration techniques, computational very expensive on many 3D datasets. In some cases, it may be sufficient to assume that the scalar field is already sampled at a rate that we can actually use basic trilinear interpolation of the samples and the variances without incorporating an error of unacceptable amount. However, we showed that we can visualize the interpolated samples with interactive frame rates using the GPU and a precomputed cache. So we can study the development of the sample values and the effect of the variance on them at arbitrary positions inside the data. Unfortunately, we cannot cache the computation of the variance in the same way as we can do with the mean. Nevertheless, we are able to compute an upper bound for the variance, which behaves like the variance itself. In most cases this is sufficient in order to see, at which positions we have a high or low amount of uncertainty.

6. Conclusion and Future Work

We have shown that the generation of images can be accomplished in reasonable time whenever the underlying data model requires the interpolation of Gaussian distributed data. Despite the high computational effort to interpolate uncertainty data using Gaussian process regression, we managed to provide reasonable frame rates using an intelligent pre-computation step. The volume rendering was enhanced by a shading step. To compute the necessary gradient, the employed interpolation scheme is used to directly calculate derivatives based on the basis functions instead of estimating numerical derivatives using the field. This analytical approach avoids typical artifacts in the volume rendering originating from approaches such as finite differences. In addition, no further sampling of the field is needed.

As we can conclude from Table 1, the differences regarding the rendering speed between the two GPUs are not significant. This is because the global memory access (i.e. the precomputed cache) is a huge bottleneck. Therefore, the goal of our future work will be to reduce fetches from global memory during rendering for example by compressing the data. This will significantly increase the rendering speed. Furthermore, since the computation of the cache is highly

parallelizable, we plan to move the cache computation to the GPU to reduce the precomputation time.

Appendix A: Interpolation and Derivative of Gaussian Variables

The mean of the linear interpolated Gaussian variable

$$X(s) = \sum_{i=0}^{N} X_i \phi_i(s), \quad \phi_0 = -1, X_0 : \text{prior distribution} \tag{15}$$

is defined as

$$\mu(s) = \sum_{i=1}^{N} \phi_i(s)\mu_i. \tag{16}$$

and the variance of $X(s)$ is

$$\begin{aligned}
&Var\big(X(s)\big) = \sigma^2(s) \\
&= \sum_{i=0}^{N}\sum_{j=0}^{N} \phi_i(s)\phi_j(s)Cov(X_i,X_j) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} \phi_i(s)\phi_j(s)Cov(X_i,X_j) + \sum_{i=0}^{0}\sum_{j=1}^{N} \phi_i(s)\phi_j(s)Cov(X_i,X_j) \\
&+ \sum_{i=1}^{N}\sum_{j=0}^{0} \phi_i(s)\phi_j(s)Cov(X_i,X_j) + \sum_{i=0}^{0}\sum_{j=0}^{0} \phi_i(s)\phi_j(s)Cov(X_i,X_j) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} \phi_i(s)\phi_j(s)Cov(X_i,X_j) - \sum_{j=1}^{N} \phi_j(s)Cov(X_0,X_j) \\
&- \sum_{i=1}^{N} \phi_i(s)Cov(X_i,X_0) + Cov(X_0,X_0) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} \phi_i(s)\phi_j(s)Cov(X_i,X_j) - 2\sum_{i=1}^{N} \phi_i(s)Cov(X_0,X_i) \\
&+ Cov(X_0,X_0) \\
&= \sum_{i,j=1}^{N} \phi_i(s)\phi_j(s)k(s_i,s_j) - 2\sum_{i=1}^{N} \phi_i(s)k(s_i,s) + k(s,s).
\end{aligned} \tag{17}$$

It can be shown that the derivative of the basis functions leads to the derivative of the variance and the mean of $X(s)$. We denote the derivative of $\phi_i(s)$ with $\phi_i'(s)$. The derivative of the mean with respect to s is straight forward:

$$\begin{aligned}
\frac{\mu(s)}{\delta s^{(n)}} &= \frac{\sum_{i=0}^{N} \mu_i\phi_i(s)}{\delta s^{(n)}} = \sum_{i=0}^{N} \mu_i \frac{\phi_i(s)}{\delta s^{(n)}} \\
&= \sum_{i=1}^{N} \mu_i \frac{\phi_i(s)}{\delta s^{(n)}} = \mathbb{E}\left(\sum_{i=1}^{N} \frac{\phi_i(s)}{\delta s^{(n)}} X_i\right),
\end{aligned} \tag{18}$$

because $\frac{\phi_0(s)}{\delta s^{(n)}} = 0$.

Furthermore, the derivative of the variance with respect to position s is given as

$$\begin{aligned}
&\frac{\delta Var\big(\sum_{i=0}^{N} X_i\phi_i(s^{(n)})\big)}{\delta s} \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\phi_i(s)}{\delta s^{(n)}}\phi_j(s)Cov(X_i,X_j) + \phi_i(s)\frac{\phi_j(s)}{\delta s^{(n)}}Cov(X_i,X_j) \\
&- 2\sum_{i=1}^{N} \frac{\phi_i(s)}{\delta s^{(n)}}Cov(X_i,X_0) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\phi_i(s)}{\delta s^{(n)}}\phi_j(s)Cov(X_i,X_j) + \phi_i(s)\frac{\phi_j(s)}{\delta s^{(n)}}Cov(X_i,X_j),
\end{aligned} \tag{19}$$

because $\frac{\phi_0(s)}{\delta s^{(n)}} = 0$ and $Cov(X_i,X_0) = 0, i > 0$.

Appendix B: Basis Functions

The basis functions are chosen in a way that the interpolated variance is minimized. Therefore, we have to take the derivative of the variance with respect to the basis functions and set it to zero.

$$\begin{aligned}
&\sum_{j=1}^{N} \phi_j(s)k(s_i,s_j) - k(s_i,s) = 0, \quad \forall i = 1\ldots N \\
\Leftrightarrow &\sum_{j=1}^{N} \phi_j(s)k(s_i,s_j) = k(s_i,s), \quad \forall i = 1\ldots N
\end{aligned} \tag{20}$$

This equation induces a linear system given whose solutions are the basis functions in Eq.5. If we insert the minimization criterion $\sum_{j=1}^{N} \phi_j(s)k(s_i,s_j) = k(s_i,s)$ into the equation for the variance (Eq. 17), we get

$$\sigma^2(s) = k(s,s) - \sum_{i=1}^{N} \phi_i(s)k(s_i,s) \tag{21}$$

as interpolated minimized variance.

Appendix C: Calculating Positions using Index Offsets

To compute the formulas (see Eq. 13) on the GPU, we need the position $s[j]$ corresponding to the j$^{\text{th}}$ value of the cell's local process. Given the bounding box bb of the grid, the cell dimensions dim, the extents ex of the grid in every dimension and the ray sample position s, we can determine the cell index by

$$\begin{aligned}
\text{cellID} = &\left\lfloor \frac{p0_x}{dim[0]} \right\rfloor + (ex[0]-1)\left\lfloor \frac{p0_y}{dim[1]} \right\rfloor \\
&+ (ex[0]-1)(ex[1]-1)\left\lfloor \frac{p0_z}{dim[2]} \right\rfloor,
\end{aligned} \tag{22}$$

where $p0$ is the point $(s - bb.min)$. The index of the cell's base point is

$$i_b = \text{cellID} + \left\lfloor \frac{\text{cellID}}{(ex[0]-1)} \right\rfloor + \left\lfloor \frac{\text{cellID}}{(ex[0]-1)(ex[1]-1)} \right\rfloor ex[1]. \tag{23}$$

The index of the base point in every direction is

$$\begin{aligned} i_z &= \left\lfloor \frac{i_b}{ex[0]ex[1]} \right\rfloor \\ i_y &= \left\lfloor \frac{i_b - i_z ex[0]ex[1]}{ex[0]} \right\rfloor \\ i_x &= i_b - i_z ex[0]ex[1] - i_y ex[0]. \end{aligned} \tag{24}$$

Using these three indices, we can compute the indices of sj in every direction:

$$\begin{aligned} sj_z &= i_z + offset[j]_z \\ sj_y &= i_y + offset[j]_y \\ sj_x &= i_x + offset[j]_x, \end{aligned} \tag{25}$$

where $offset[j]$ is the jth entry (corresponding to sj) in the offset array, which was computed in Sec. 4.4. Finally, the position of sj can be computed as

$$sj = bb.min + (sj_x dim[0], sj_y dim[1], sj_z dim[2]). \tag{26}$$

References

[AE13] ATHAWALE T., ENTEZARI A.: Uncertainty quantification in linear interpolation for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec 2013), 2723–2732. doi:10.1109/TVCG.2013.208. 2

[AT11] ADLER R., TAYLOR J.: *Topological complexity of smooth random functions: École d'Été de Probabilités de Saint-Flour XXXIX - 2009*. Lecture Notes in Mathematics. Springer, 2011. 2

[BFRD13] BODESHEIM P., FREYTAG A., RODNER E., DENZLER J.: *An Efficient Approximation for Gaussian Process Regression*. Tech. rep., Technical Report TR-FSU-INF-CV-2013-01, Computer Vision Group, Friedrich Schiller University Jena, Germany, January 2013. Technical Report TR-FSU-INF-CV-2013-01. 4

[Bur96] BURSAL F. H.: On interpolating between probability distributions. *Applied Mathematics and Computation 77*, 2-3 (1996), 213 – 244. doi:10.1016/S0096-3003(95)00216-2. 2

[DKLP01] DJURCILOV S., KIM K., LERMUSIAUX P. F. J., PANG A.: Volume rendering data with uncertainty information. In *Data Visualization 2001: Proceedings of the Joint Eurographics - IEEE TVCG Symposium on Visualization* (2001), pp. 243–252. 2

[GR04] GRIGORYAN G., RHEINGANS P.: Point-based probabilistic surfaces to show surface uncertainty. *IEEE Transactions on Visualization and Computer Graphics 10*, 5 (2004), 564–573. 2

[Hen03] HENGL T.: Visualisation of uncertainty using the hsi colour model: Computations with colours. *Proceedings of the 7th International Conference on GeoComputation* (2003), 1–12. 2

[Kri51] KRIGE D. G.: A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical Metallurgical and Mining Society of South Africa 52*, 6 (1951), 119–139. 2

[KVUS*05] KNISS J. M., VAN UITERT R., STEPHENS A., LI G. S., TASDIZEN T., HANSEN C.: Statistically quantitative volume visualization. 287–294. doi:10.1109/VISUAL.2005.1532807. 2

[KW03] KRUGER J., WESTERMANN R.: Acceleration techniques for gpu-based volume rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (Washington, DC, USA, 2003), VIS '03, IEEE Computer Society, pp. 38–. doi:10.1109/VIS.2003.10001. 2

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *COMPUTER GRAPHICS 21*, 4 (1987), 163–169. 1

[LLPY07] LUNDSTRÖM C., LJUNG P., PERSSON A., YNNERMAN A.: Uncertainty visualization in medical volume rendering using probabilistic animation. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov./Dec. 2007), 1648–1655. 2

[PH11] PÖTHKOW K., HEGE H.-C.: Positional uncertainty of isocontours: Condition analysis and probabilistic measures. *IEEE Transactions on Visualization and Computer Graphics 17*, 10 (2011), 1393 – 1406. doi:10.1109/TVCG.2010.247. 1, 2

[PH13] PÖTHKOW K., HEGE H.-C.: Nonparametric models for uncertainty visualization. *Computer Graphics Forum 32*, 3 (2013), 131 – 140. doi:10.1111/cgf.12100target. 2

[PMW12] PFAFFELMOSER T., MIHAI M., WESTERMANN R.: *Probability Distributions for Gradient Orientations in Uncertain 3D Scalar Fields*. Tech. rep., Technische Universität München, 2012. 2

[PPH13] PÖTHKOW K., PETZ C., HEGE H.-C.: Approximate level-crossing probabilities for interactive visualization of uncertain isocontours. *International Journal for Uncertainty Quantification 3*, 2 (2013), 101 – 117. doi:10.1615/Int.J.UncertaintyQuantification.2012003958. 2

[PRW11] PFAFFELMOSER T., REITINGER M., WESTERMANN R.: Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 951–960. 2

[PWH11] PÖTHKOW K., WEBER B., HEGE H.-C.: Probabilistic marching cubes. *Computer Graphics Forum 30*, 3 (2011), 931 – 940. doi:10.1111/j.1467-8659.2011.01942.x. 2

[PWL96] PANG A. T., WITTENBRINK C. M., LODH S. K.: Approaches to uncertainty visualization. *The Visual Computer 13* (1996), 370–390. 1, 5

[RJ99] RHEINGANS P., JOSHI S.: Visualization of molecules with positional uncertainty. In *Data Visualization âĂŹ99, Proceedings of the Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization* (1999), Springer-Verlag, pp. 299–306. 2

[RLBS03] RHODES P. J., LARAMEE R. S., BERGERON R. D., SPARR T. M.: Uncertainty visualization methods in isosurface volume rendering. In *Eurographics 2003, Short Papers* (2003), pp. 83–88. 2

[RW06] RASMUSSEN C. E., WILLIAMS C.: *Gaussian Processes for Machine Learning*. MIT Press, 2006. 2, 3

[SKS12] SCHLEGEL S., KORN N., SCHEUERMANN G.: On the interpolation of data with normally distributed uncertainty for visualization. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2305–2314. doi:10.1109/TVCG.2012.249. 1, 2, 3, 5

[ZWK10] ZEHNER B., WATANABE N., KOLDITZ O.: Visualization of gridded scalar data with uncertainty in geosciences. *Computers and Geosciences 36*, 10 (2010), 1268–1275. 2