Eurographics Conference on Visualization (EuroVis) 2013
B. Preim, P. Rheingans, and H. Theisel
(Guest Editors)

Volume 32 (2013), Number 3

Progressive High-Quality Response Surfaces for Visually Guided Sensitivity Analysis

I. Demir[1] and R. Westermann[1]

[1]Computer Graphics & Visualization Group, Technische Universität München, Germany

Abstract

In this paper we present a technique which allows us to perform high quality and progressive response surface prediction from multidimensional input samples in an efficient manner. We utilize kriging interpolation to estimate a response surface which minimizes the expectation value and variance of the prediction error. High computational efficiency is achieved by employing parallel matrix and vector operations on the GPU. Our approach differs from previous kriging approaches in that it uses a novel progressive updating scheme for new samples based on blockwise matrix inversion. In this way we can handle very large sample sets to which new samples are continually added. Furthermore, we can monitor the incremental evolution of the surface, providing a means to early terminate the computation when no significant changes have occurred. When the generation of input samples is fast enough, our technique enables steering this generation process interactively to find relevant dependency relations.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

In many areas of science and engineering, multivariate scalar functions $f : \mathbb{R}^n \to \mathbb{R}$ arise, where the values depend on many independent variables. In many such applications the objective is to monitor the sensitivity of a process and the generated results to the choice of parametrization over the input variables, and to finally optimize the result taking into account the observed dependency relations.

Very often multivariate functions are not given analytically, but values are available only at a discrete set of sampling points in the n-dimensional parameter domain. While often the generation of results is a time-consuming preprocess, sometimes the result can be updated at very high rates from a given new parametrization. This later possibility becomes more and more feasible in numerical experiments due to the ever increasing capacities of computing architectures. As already recognized by van Wijk and van Liere [vWvL93], it is particular appealing because it gives rise to a visual navigation in high dimensional parameter space and allows steering towards an optimal solution by interactive variable modifications.

However, the visual analysis of multivariate data in an intuitive way is challenging, since it is virtually impossible for humans to recognize visual representations beyond three dimensions. Thus, multivariate data analysis, in general, requires to reduce the dimensionality of the multidimensional data set and, thus, to enable visualizations using common techniques for data with less than 4 dimensions.

A popular approach for dimensionality reduction is to slice the data or to orthogonally project it onto some two-dimensional subspace of the n-dimensional parameter space, and then to view the resulting value distributions in these subspaces. Since in general the sample positions are sparse in the selected subspaces, dimensionality reduction requires interpolation between samples to compute a continuous representation in the used 2D sampling structures. This gives rise to an effective prediction of the values at unobserved sites and the (non-linear) dependencies between parameters and function values.

Especially if the continuous approximation is visualized as a response surface, which is the graph of the continuous objective function plotted over the two-dimensional subspace, an improved analysis of the relationships between parameters and objective is possible. In particular the possibility to take into account the surface's topography such as extreme points, gradients, or ridges, can greatly enhance the understanding of the dependency relations.

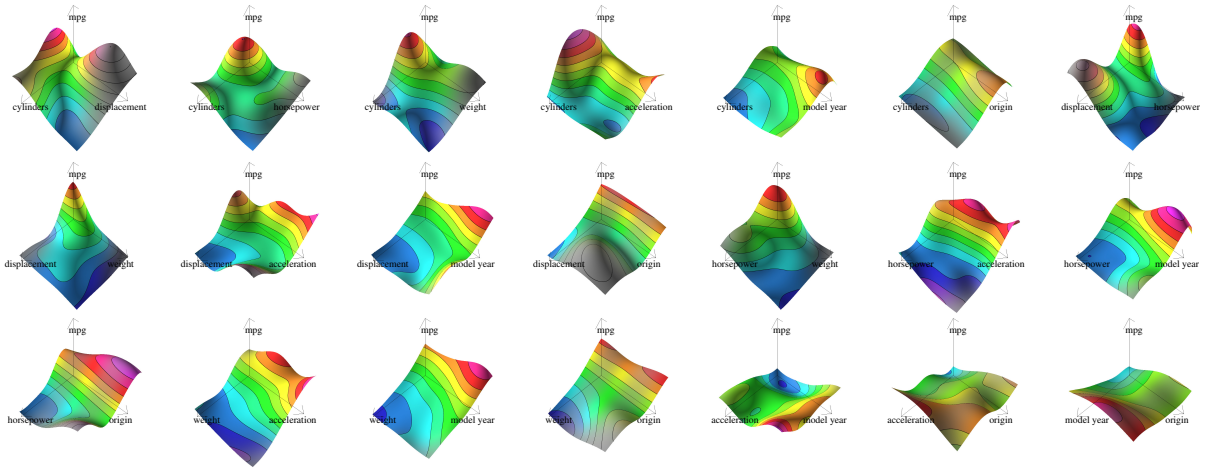

Figure 1: *Visualization of response surfaces for all pairs of parameters of a multivariate scalar function. Incremental update and visualization of the surfaces is performed at less than 5 ms.*

Unfortunately, computing response surfaces with provably optimal fairness from multivariate input samples is computationally expensive, since it requires scattered interpolation at points in the sampling structure over which the surface is formed. For large sets of input samples a time consuming initial preprocess is required to obtain the interpolation weights for surface reconstruction. Even more importantly, when input samples are generated progressively, for instance by user-driven parameter space navigation, every new sample requires to repeat the exhaustive preprocess, prohibiting an in-turn visual analysis of the evolving surface.

1.1. Contribution

We present a method which enables an interactive visual exploration of multivariate scalar functions via high-quality response surfaces (see Fig. 1 for an example). The surfaces are computed via kriging interpolation [Kri51], a Gaussian process regression model for inference from scattered samples in multidimensional parameter space. Kriging determines the interpolation weights entirely by the data configuration and the covariance model, and it finds the least squares estimate of a stationary random function which minimizes the variance of the random function increments. The contributions we make to the field of multivariate scalar data interpolation and visualization are as follows:

- **Fast incremental updates of kriging calculations**: To make kriging suitable for an interactive exploration of large sets of multivariate scalar samples, we propose a progressive updating scheme for the kriging interpolation weights. This scheme builds upon incremental matrix inversion [Ban37], and it enables updating the interpolation weights with only minor computational effort when new samples are added. We employ a similar principle as underlying so-called online learning algorithms [Opp98], where only the last example is used for updating a learning network's parameters. For data sets being so large that the construction of a response surface takes unacceptably long, the scheme enables to construct the surface incrementally by considering only one new sample at a time. Since kriging requires exhaustive use of matrix-vector and vector-vector operations to compute the interpolation weights and data estimates, we have implemented the entire scheme on the GPU, including incremental update of the inverse kriging matrix and interpolation.
- **Response Surface Selection**: To enable an interactive visual analysis of multivariate scalar functions, we have embedded the progressive update scheme for kriging interpolation into a slice-based navigation interface similar to scatterplots and HyperSlice [vWvL93]. We propose using parallel coordinates to interactively select the 2D subspaces of the high-dimensional sample space for which response surfaces should be computed and visualized. As we provide immediate visual feedback about the structure of the response surfaces in all subspaces, if new sample values can be generated in turn the user can steer the location of these samples interactively to further refine regions of interest.

The remainder of this paper is organized as follows: Next, we discuss work that is related to ours. In section 3 we introduce the concept underlying kriging and outline its use for scattered data interpolation. Our progressive scheme (including a complexity analysis and GPU aspects) is presented in section 4.2. Then, we discuss approaches for selecting response surfaces via slicing and projection. We finally present results using a real-world data set as well as a detailed per-

formance statistics, and we conclude the paper with a discussion of future extensions of our work.

2. Related Work

Our approach uses common techniques for dimensionality reduction of multivariate data, such as orthogonal projection and slicing [Asi85, FB94, vWvL93]. An overview of techniques in multivariate data visualization can be found in work by Scott [Sco92] and Grinstein et al. [GTC01]. The particular model we use for displaying the reduced data is that of a response surface, i.e., a continuous surface that predicts the objective values from given samples over a selected 2-dimensional domain [MM95, Mon06]. Unlike scatter plots [Cle85], by using response surfaces we aim at displaying continuous regions from which the user can see correlations more easily and estimate values at unmeasured sample positions. A different approach is parallel coordinates [Ins85, Weg90], were the data are mapped to 1D graphs rather than points in a 2D subspace. Parallel coordinates perform a dimensionality reduction and enable findings of the existence of linear dependencies between variables, however, spatial relationships are lost and topographic properties as for response surfaces cannot be determined.

For computing a continuous approximation of a discrete multivariate function on a selected sampling structure we employ techniques for scattered data interpolation [FN91, Wen05]. Radial basis functions (RBFs) are commonly used for scattered data interpolation problems [Buh03], since they are able to interpolate arbitrary constraints in a smooth manner. While compactly supported radial basis functions lead to sparse linear systems and hence can be used to efficiently interpolate large amounts of data samples, they do not provide the same approximation quality as basis functions of global support. Globally supported RBFs, on the other hand, result in dense matrix structures and far less efficient solution methods thereof. Even though improved methods exist, such as multipole methods [CBC*01] and incremental least-squares solutions [BK05], such methods require a considerable amount of preprocessing when new samples arrive.

An alternative class of (statistical) methods for predicting a latent function from a given set of discrete samples are Gaussian processes models [OK78, Nea99]. In such models, a Gaussian process is used to describe the a priori uncertainty about the function based on empirical observations. Such models attempt to describe the dependency of an observation on a corresponding input via a conditional distribution. If the observation (or objective) is a one dimensional scalar, then the distribution is called a regression model. In geostatistics, regression with Gaussian processes to predict a spatial phenomenon at unobserved sites is known as kriging or BLUE (best linear unbiased estimator) interpolation [Kri51, MB62, Mat63].

In data visualization, only few approaches so far have adopted statistical prediction of continuous representations from discrete sets of multi-dimensional samples for sensitivity analysis. Examples include the work by Pieringer et al. [PBK10], where surface plots of multivariate functions have been used to analyze the sensitivity of surrogate models. Torsney-Weir et al. [TWSM*11] employed Gaussian process models for predicting quality metrics in image segmentation based on selected parameter settings. They also employed the concept of response surfaces to display the variation of the used objective function. Berger et al. [BPFG11] have used nearest-neighbor and model-based statistical prediction for deriving objectives at unobserved locations in parameter space. The later approach performs statistical sampling in parameter space to obtain a set of training data from which a response function can be trained. Recently, Schlegel et al. [SKS12] shed light on the interpolation properties of Gaussian process regression (including kriging interpolation) for data interpolation and provided analytical descriptions of the underlying spatial basis functions.

To the best of our knowledge, in none of the previous approaches was the problem addressed to perform continuous statistical predictions efficiently from discrete sets of multi-dimensional samples which are updated progressively. Current approaches usually assume a static set of observations at known locations in parameters space. The possibility to handle progressive updates of the internal representations, for instance when new observations arrive, has not been considered. Thus, we see our approach as an important addition to these works, opening new ways for interactive multivariate data exploration.

3. Kriging Interpolation

Kriging interpolation is a probabilistic method to describe a quantity at unobserved sites from a discrete set of observations at given locations. The principles underlying kriging interpolation were introduced by Krige [Kri51] in the context of geostatistics, and later put into a formal concept by Matheron and Blondel [MB62, Mat63]. Underlying kriging is the notion of a random function, which describes a quantity over a spatial domain as a set of random variables at the given locations. The spatial relations between the given observations is expressed by the covariance structure for all pairs of variables, often represented by the variance of the observed increments over distance, ie. the so called variogram.

In short, kriging methods first perform a structural analysis of the given observation to derive the dependency structure, and then estimate interpolation weights for each given location by solving a least squares minimization of the estimated error variance. A thorough introduction to kriging is given by Cressie [Cre93]. Even though the interpolation properties of kriging are well known, its use outside geostatistics is limited. In our opinion this is mainly because of the inherent computational complexity for calculating the

linear interpolation weights. Solving the kriging equations directly for n observations involves inversion of an $n \times n$ matrix.

To overcome this limitation, acceleration schemes have been proposed, like so called ad-hoc methods [Haa95] using locally adaptive covariance prediction, fixed rank kriging [CJ08] using empirical low-rank variances and covariances estimates, or a GPU implementation which intertwines the calculation of kriging weights and interpolation in a very efficient way [HCL*11]. To the best of our knowledge, possibilities to recompute kriging weights progressively upon the arrival of new samples on the GPU has not been proposed until now.

3.1. Main Principles

The main idea behind kriging is to spatially interpolate the quantity at a point x^* by finding linear weights λ_i of the k collected data points $(x_i, f(x_i)), 1 \leq i \leq k$, yielding the interpolated value

$$\hat{f}(x^*) = \sum_{i=1}^{k} \lambda_i(x^*) f(x_i). \tag{1}$$

In this work we consider what is called ordinary kriging, since, aside from the samples, it only needs a covariance function to accomplish the interpolation. The covariance is a probabilistic measurement which specifies to which degree two random variables change together. More formally, it is defined as

$$Cov(x,y) = \mathrm{E}[(x - \mathrm{E}[x]) \cdot (y - \mathrm{E}[y])],$$

where E denotes the expectation value operator. Intuitively the covariance should increase when two points of the random field are closer together. Assuming that the underlying random field is stationary, ie., expectation value μ and variance σ^2 are constant, and covariance depends only on the distance between two points, we can use a simple model to estimate the unknown covariance. In this work we use the Gaussian model given by

$$Cov(x,y) = \begin{cases} \left(\sigma^2 - n\right) \exp\left(\frac{-\|x-y\|^2}{a \cdot r^2}\right) & x \neq y \\ \sigma^2 & x = y \end{cases},$$

where σ^2 denotes the sill to which the covariance tends if the distance between x and y decreases, and r denotes the range which determines how rapidly the covariance decreases with a larger distance. Note that the sill is also equal to the variance, since the covariance equals the variance if $x = y$. A nugget effect is modeled by n to prevent oscillatory results when sample points with different values lie close together. The value of a allows further adjustment of the impact of the range and is typically set to $a = 1/3$.

3.2. Ordinary Kriging

We now briefly introduce the ordinary kriging method. In order to calculate the linear interpolation weights $\lambda_i(x^*)$ at the interpolation point x^*, let us consider the covariances between all sample points $Cov(x_i, x_j)$, and the covariances between the interpolation point and all sample points $Cov(x_i, x^*)$, where $1 \leq i, j \leq k$. To begin with we try to find weights $\lambda_i(x^*)$ such that

$$Cov(x_i, x^*) = \sum_{j=1}^{k} \lambda_j(x^*) Cov(x_i, x_j)$$

holds for every $i \in \{1, \ldots, k\}$. In other words, the covariance between any sample point x_i and the interpolation point x^* is equal to the linear combination of the covariances between x_i and every sample point x_j weighted by λ_j.

However, one obstacle needs to be overcome, namely the fact that the weights do not necessary sum up to 1, which is generally required for the interpolation value to be unbiased. Therefore we introduce a Lagrange multiplier $\nu(x^*)$ such that

$$C \cdot \lambda(x^*) + \mathbf{1} \cdot \nu(x^*) = c(x^*), \lambda^T(x^*) \cdot \mathbf{1} = 1.$$

By defining the extended matrix and vectors

$$C_+ = \begin{pmatrix} C & \mathbf{1} \\ \mathbf{1}^T & 0 \end{pmatrix}, c_+(x^*) = \begin{pmatrix} c(x^*) \\ 1 \end{pmatrix}, \lambda_+(x^*) = \begin{pmatrix} \lambda(x^*) \\ \nu(x^*) \end{pmatrix}$$

the problem can be rewritten as

$$C_+ \cdot \lambda_+(x^*) = c_+(x^*)$$

yielding the solution

$$\lambda_+(x^*) = C_+^{-1} \cdot c_+(x^*). \tag{2}$$

In addition, we calculate the kriging variance to obtain a measure for the uncertainty in the current sampling

$$\begin{aligned} \sigma_k^2(x^*) &= \sigma^2 - c_+^T(x^*) \cdot C_+^{-1} \cdot c_+(x^*) \\ &= \sigma^2 - c_+^T(x^*) \cdot \lambda_+(x^*). \end{aligned} \tag{3}$$

This uncertainty tells where the density of observations is too low such that no reliable estimate of the response surface is possible. In our current tool the uncertainty is used to guide the user towards regions in the parameter space where additional samples should be retrieved.

4. Progressive GPU Kriging

To perform ordinary kriging interpolation, the inverse of the extended covariance matrix C_+ is required. Since C_+, and thus its inverse, have to be updated every time new samples are added, kriging can be very time- and memory-consuming when implemented naively. To address this problem we propose a method which enables us to update the kriging matrix incrementally, meaning that the previous inverse matrix can be reused and only a small matrix has to be inverted.

This method is based on blockwise matrix inversion proposed by [Ban37].

In blockwise inversion one assumes that the $(k+l)\times(k+l)$ matrix to be inverted is in block form, i.e., it consists of an upper left $k\times k$ and lower right $l\times l$ matrix P and S, and a lower left $l\times k$ and upper right $k\times l$ matrix R and Q. It is assumed that the block P is invertible. Then, the matrix inverse can be computed as

$$\begin{pmatrix} P & Q \\ R & S \end{pmatrix}^{-1} = \begin{pmatrix} W & X \\ Y & Z \end{pmatrix}, \text{ with}$$
$$Z = \left(S - RP^{-1}Q\right)^{-1}, X = -P^{-1}QZ, Y = -ZRP^{-1}$$
$$W = P^{-1} - P^{-1}QY = P^{-1} - XRP^{-1}.$$

In our scenario, since both C and C_+ are symmetric due to the fact that $Cov(x,y) = Cov(y,x)$, we can assume $P = P^T$, $R = Q^T$, $S = S^T$. Thus, we arrive at the following simplified formula

$$\begin{pmatrix} P & Q \\ Q^T & S \end{pmatrix}^{-1} = \begin{pmatrix} W & X \\ X^T & Z \end{pmatrix}, \text{ with}$$
$$Z = \left(S - Q^T P^{-1} Q\right)^{-1}, W = P^{-1} - P^{-1}QX^T. \quad (4)$$

Our method for incrementing the inverse of the extended covariance matrix is now split into two stages: From the given previous sample positions $x_i, 1 \le i \le k$, the previous covariance matrix C and its inverse C^{-1}, and the added sample positions $x'_j, 1 \le j \le l$, the new inverse covariance matrix C'^{-1} is calculated in the following way. Let $P := C$, $(Q)_{ij} := Cov\left(x_i, x'_j\right)$, and $(S)_{j\tilde{j}} := Cov\left(x'_j, x'_{\tilde{j}}\right)$, where $1 \le i \le k$, $1 \le j, \tilde{j} \le l$. Then we have

$$C' = \begin{pmatrix} P & Q \\ Q^T & S \end{pmatrix}$$

which enables us to calculate C'^{-1} by blockwise inversion. Here it is worth noting that it is not necessary to store C since it is not required in order to calculate C'^{-1}. Only C^{-1} has to be stored. Also note that the only matrix which needs be inverted is of dimension $l\times l$, where l equals the number of added samples.

In the second stage we calculate C'^{-1}_+ by a similar procedure. We now set $P := C'$, $Q := \mathbf{1}$, $S := 0$ and obtain

$$C'_+ = \begin{pmatrix} P & Q \\ Q^T & S \end{pmatrix}.$$

By applying the blockwise inversion, we finally obtain C'^{-1}_+. Again, C' is not required in the computation, and only C'^{-1} has to be stored. The matrix to be inverted is of dimension 1×1. Also note that most operations in this process are matrix multiplications which can be efficiently parallelized on the GPU.

4.1. Computational complexity

In the progressive kriging approach, the matrix C'^{-1} is obtained by first calculating $V := P^{-1}Q$, Z, X, and W, and then putting these intermediate results together. The reason for calculating V explicitly is that it is used in subsequent steps. Since it takes $\mathcal{O}(pqr)$ floating point operations to compute the matrix multiplication of one $p\times q$ and one $q\times r$ matrix, $\mathcal{O}(pq)$ operations for adding two $p\times q$ matrices, and $\mathcal{O}\left(p^3\right)$ operations for inverting a $p\times p$ matrix, calculating V, Z, X and W respectively takes $\mathcal{O}\left(k^2 l\right)$, $\mathcal{O}\left(kl^2 + l^3\right)$, $\mathcal{O}\left(kl^2\right)$, and $\mathcal{O}\left(k^2 l\right)$ operations. This gives a total of $\mathcal{O}\left(k^2 l + kl^2 + l^3\right)$ operations for increasing the inverse covariance matrix by l new samples. A similar consideration leads to $\mathcal{O}\left(k^2\right)$ operations for computing C'^{-1}_+.

Let us now consider a state-of-the-art GPU matrix inversion method as proposed by Ezatti et. al. [EQOR11]. Their algorithm takes $\Theta\left(p^3\right)$ operations to invert a matrix of size $p\times p$, i.e. $\Theta\left((k+l)^3\right)$ operations for calculating C'^{-1}_+. Compared to the progressive approach this yields a significantly higher run-time complexity when l is small compared to k, ie. $l = o(k)$. In this case our method has a complexity of $\mathcal{O}\left(k^2\right)$ as opposed to $\Theta\left(k^3\right)$ for the conventional method. In section 6.1 we verify this result in practice.

4.2. CUDA Parallelization

The capabilities of recent GPUs are employed via the CUDA programming language to perform both the calculation of the kriging interpolation weights and the final interpolation of the initial data samples in a highly efficient way. The first step when a new sample arrives and has to be considered in the kriging interpolation is the calculation of the covariance vector with respect to the new sample position. This process is carried out in a straight forward way by computing each element of the covariance vector by one CUDA thread in parallel. In all our examples we used the Gaussian model with nugget effect as the covariance function. Even for large sets of samples, the overall time for performing this step is so small that it does not affect the overall runtime.

From the discussion of (progressive) kriging it becomes clear that all time-critical computations which are required when new samples are added and have to be considered in the interpolation step are matrix multiplications. Matrix-matrix multiplication can be parallelized quite effectively on the GPU by computing each entry in the result matrix in parallel. This concept can be further improved by a technique called tiling, which reduces redundant memory accesses (see Kirk and Hwu [KH10]). In our current implementation we use CUBLAS for virtually all matrix operations, a highly optimized GPU matrix library available in CUDA. One of the great achievements of CUBLAS is that it makes efficient

use of the GPU memory hierarchy, trying to best utilize the fast memory segments which are shared by certain groups of threads.

Updating the kriging matrices C^{-1} and C_+^{-1} is done in a straightforward way by restating Eq. 4 with CUBLAS functions. To calculate the weight vectors $\lambda_+(x^*)$ at the interpolation points all at once with a single call to the CUBLAS library we utilize the fact that we can combine all matrix-vector products of Eq. 2 to a single matrix-matrix product. That is to say we define

$$\hat{C}_+ = \begin{pmatrix} c_+(x_1^*) & c_+(x_2^*) & \cdots \end{pmatrix}$$

as a matrix containing all interpolation points, leading to

$$\hat{\Lambda}_+ = C_+^{-1} \cdot \hat{C}_+$$

where each column of $\hat{\Lambda}_+$ contains the weight vector of the corresponding interpolation point.

The only exception where we do not make use of CUBLAS is for computing the kriging variance (see Eq. 3) as we need to calculate a dot product at each interpolation point. Since CUBLAS does not provide a routine which allows parallelized computation of many dot products at once we implemented a CUDA method for this task. It is based on the most efficient parallel reduction method presented by Harris [Har07].

5. Visualizing Response Surfaces

Let $X \subseteq \mathbb{R}^n, n \geq 2$ denote a discrete multidimensional data set with a given scalar value assigned to each element via the mapping $f : X \to \mathbb{R}$. We call every tuple $(x, f(x))$ a sample, where $x \in X$ indicates the position and $f(x)$ the value of the sample. Our goal is to visualize this data set by a set of surfaces such that each surface represents the scalar values over two distinct dimensions. For each selected parameter pair we define a two-dimensional planar sampling grid and interpolate the scalar values at the sample points in this grid via kriging. These values are then used to form a surface. Each surface is rendered as a triangle mesh on the GPU, using colors in HSL space to indicate differences in value and uncertainty. The hue of a surface point is determined by its altitude, ie. the scalar value at the corresponding grid point, while its saturation is set inversely proportional to the uncertainty. Additionally we draw contour lines to provide the user a better classification of the surface points.

5.1. Response Surface Selection

For selecting the response surfaces which should be visualized, we utilize the HyperSlice method. Given a user-selected center point, for each pair of parameter space axes a 2D slice parallel to the plane spanned by the two axes and going through the selected point is defined. For every slice a separate response surface is computed.

HyperSlice allows the user to interactively change the center point and thereby steer through the whole data set. In the original work this was performed by pressing a mouse button while the cursor points to a slice, and then dragging the center point according to this slice. In contrast, we employ an approach which makes use of parallel coordinates to indicate the sample density in the region in which the center point is actually positioned. In this way, the center point can be adjusted towards those parameter intervals which are already well resolved, or, for instance in a visual steering application, those intervals which are poorly resolved can be prioritized for point selection. The proposed selection procedure is demonstrated in Fig. 2, where each sample is represented by a polyline which has its vertices aligned at the parallel coordinate axes. To support the user in choosing the position of the center point, every axis is color-coded from blue to red according to decreasing sample density.

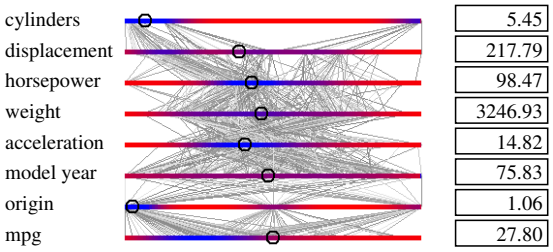

Figure 2: *Utilizing parallel coordinates to adjust the center point (indicated by circles) in HyperSlice. Parallel coordinates are color coded to show the sample density on each parameter axis.*

Besides visualizing the response surfaces of all selected slices in a structure similar to a scatterplot matrix (see Fig. 1 for such a visualization according to the selected center point in Fig. 2), the use of parallel coordinates for center point selection gives rise to an alternative visualization strategy: The user changes the coordinate of the center point along one selected parameter axis interactively, while keeping all other coordinates fix. Thus, the response surfaces for every pair of axes involving the selected axis remain unchanged, and only for every other pair a change of the surface is triggered.

This is demonstrated in Fig. 3, where we analyze data describing the fuel efficiency of automobiles in miles per gallon (mpg) [Aut93]. It contains a total of 398 samples where each sample corresponds to one specific car, comprising the values mpg, cylinders, displacement, horsepower, weight, acceleration, model year, car name, and origin, a discrete value representing different states (note here that the use of kriging for nominal data like origin is for demonstration purposes only, and that in general any interpolation of nominal data requires some data-specific rational). Figs. 1 and 2 show visualizations of this data. By choosing three center points with distinct values for horsepower and weight, one obtains

three response surfaces displaying the fuel efficiency according to origin and model year as shown in Fig. 3a,b,c.

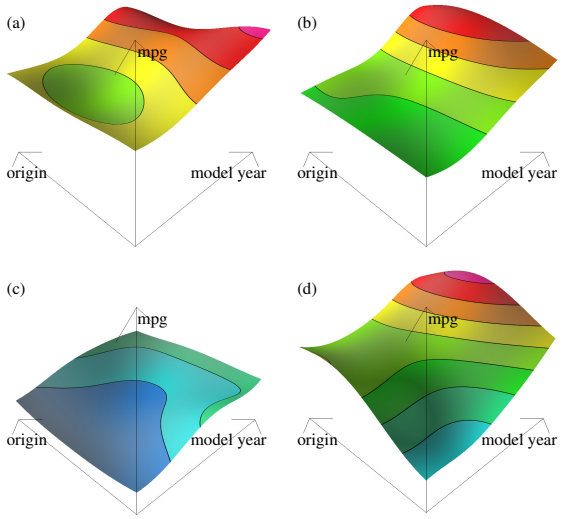

Figure 3: *HyperSlice is used to display the fuel efficiency (mpg) according to origin and model year for three different tupels of horsepower and weight: (a)* $(50, 1900)$*, (b)* $(80, 2500)$*, (c)* $(150, 3600)$*. Note that there is virtually no impact of origin. (d) All samples are projected to the selected subspace and smoothly interpolated. Now one can see that cars from one origin tend to have low fuel efficiency, meaning that they generally differ in other attributes like horsepower or weight.*

In Fig. 3d, all automobile samples are projected to the subspace represented by the same axes used in HyperSlice before, and a response surface is computed from the projected samples. The advantage of projection over HyperSlice is that all samples are visualized simultaneously, giving an immediate overview of the whole data set without requiring the user to adjust a focal point. The corresponding response surface shows that cars from one origin generally have low fuel efficiency. But for now we do not know if this is really caused by the origin or just by the fact that cars from other countries tend to differ in other attributes, for instance they might generally have less horsepower. This difficulty arises because the distances between sample positions in the high-dimensional sample space get lost due to the projection. This leads to the false impression that some distant samples lie close together.

When HyperSlice was used in Fig. 3a,b,c, the center point was placed at three different positions, with vastly different values of horsepower and weight. In contrast to projection, HyperSlice shows that the value of origin does not have a significant impact on the fuel efficiency even if entirely different values for horsepower and weight are chosen, which generally have the greatest influence. For other slices not shown here we obtain similar views. We can thus conclude that the origin influences certain other attributes, for instance horsepower, which on their part affect the fuel efficiency. This example also demonstrates the difference between the surface projection and the HyperSlice method. Especially in an interactive session, where the user moves the center point coordinate along the selected parameter axis, the corresponding changes of the response surfaces can effectively reveal more subtle dependencies.

From the example we can conclude, that it might be a good practice to first use projection and then HyperSlice to visualize a multi-dimensional data set. The projection method gives the user a general overview of the whole data set, even when only a small but representative subset of all samples is considered. For instance, such a subset can be generated by randomly selecting samples from the initial set. Relevant features can then be further analyzed using HyperSlice, by incrementally adding those samples which are close to the currently selected center point in all but two variables. Whenever the user changes the center point, new samples will always be selected according to this new position. Due to the specific selection strategy of the next samples to be considered, even for very large data sets a rather small subset of the given samples might already suffice for an accurate analysis.

5.2. Slice-based Interpolation

Our next step is to apply kriging interpolation to HyperSlice and projection. Since all slices visualized in HyperSlice share the same space, the method requires only one kriging matrix containing the covariances of the actual multidimensional sample positions. This matrix needs to be updated only when new samples are added, because it does not depend on the position of the currently selected slices. As the inverse kriging matrix is required to perform the interpolation, it has to be updated as well if the kriging matrix was changed. For each interpolation point, ie. the grid points of the sampling structures used to represent the selected slices, the covariance vector is calculated and used to determine the interpolation weights of every other point via Eq. 2. These weights are finally used to compute the interpolation values via Eq. 1.

The interpolation step can be fully parallelized since every interpolation can be computed independently of each other. In order to calculate the covariance vectors at the interpolation points we need to embed the 2D sampling structures into the multidimensional sample space. This is performed by setting the coordinates of the interpolation points along the two spanning parameter axes according to their relative position in the respective 2D subspace, and filling the remaining coordinates with the values of the center point. This also implies that each time the center point is moved or the resolution of the sampling grid is changed the interpolation process has to be repeated.

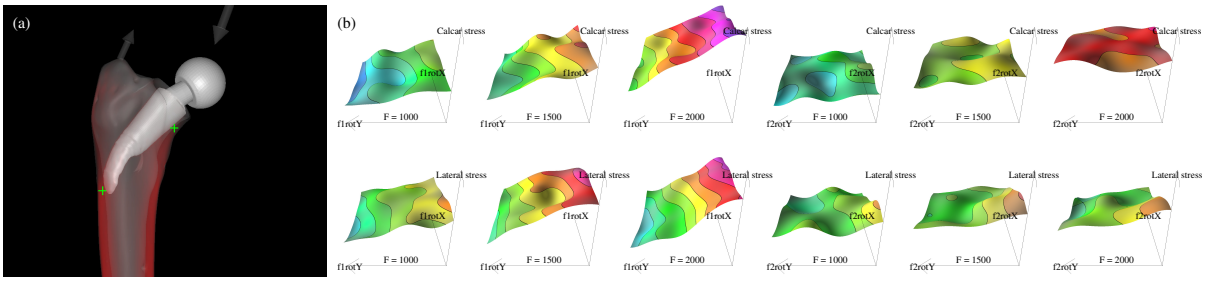

Figure 4: *(a) Simulation setting, including arrow glyphs indicating the simulated forces. (b) Response surfaces for different force magnitudes depending on force direction.*

The projection method requires a procedure that differs mainly in the computation of the covariances. Since in this case every sample is projected to a 2D subspace, we have to maintain a separate kriging matrix and its inverse. For the kriging matrix we use a covariance function that takes only those components of each sample position into account that correspond to the respective subspace. The same holds for the computation of the covariances at the interpolation points. Unlike embedding the grid points into the multidimensional space we project the sample positions to the selected subspace and then calculate the covariance between the projected sample positions and the grid points.

6. Results

In addition to the data we have used so far for demonstrating the potential of response surfaces for sensitivity analysis, we have used our approach in a computational steering environment for analyzing material stresses depending on exerting forces. The underlying application is implant planning for hip joint replacements, where in a pre-operative planning phase a surgeon tries to find the patient-specific optimal implant shape, size and position [DGBW08]. The steering tool is designed in such a way that external forces can be issued at the surface of an implant which has been inserted into the bone, and the resulting stresses in the bone interior can be simulated and visualized. To realistically simulate stresses which occur during walking, we consider an additional force of constant magnitude which is applied by the muscles. Fig. 4a shows the simulation setting, including arrow glyphs indicating the simulated forces.

The simulation system computes the internal stresses, ie., the scalar von Mises stress norm, in less than 200 milliseconds once the external forces have been specified. The volume rendering in Fig. 4a shows the simulated scalar field. In our particular scenario we have analyzed the sensitivity to the inserted forces of the stresses at two critical points P_1 and P_2 in the calcar region and the lateral femoral wall, marked by green crosses in Fig. 4a. Therefore, we have integrated the response surface approach into the simulation tool, enabling the user to interactively change the points at which the forces are acting and the force magnitude via a navigation tool similar to the one shown in Fig. 2. We specify a force direction in polar coordinates of two virtual spheres enclosing the regions of contact, always acting towards a sphere's center, and compute the intersection points between the force vectors and the implant and bone. Overall, we have 5 free parameters (2 angles for each point and a force magnitude), and we obtain stress values for the points P_1 and P_2.

Upon simulating the stresses, the simulation module exports the values at P_1 and P_2, and these values are then considered in the computation of new response surfaces. Fig. 4b shows the response surface visualization using HyperSlice after approximately 7000 different parameter combinations have been performed. In an interactive session the user would now start refining the parameter setting in local extremum regions to further analyze the stress sensitivity, or the response surfaces for a different implant position would be compared. As we will show next, by using a conventional method for response surface construction from all given samples, an interactive parameter space navigation would be impossible.

6.1. Performance Analysis

The performance analysis is structured in two parts. First we measure the performance of incrementally constructing the covariance matrices when new samples are added, and then we measure the time to perform the kriging interpolation at different grid resolutions. All measurements were performed with HyperSlice on an NVIDIA GeForce GTX 580 graphics card.

Fig. 5 shows the times in milliseconds for computing the inverse of a covariance matrix, when the inverse for a given number of samples is given and one new sample arrives. In our tests we have considered samples of different dimensions $d \in \{4,6,8,10\}$, and we have made distinguishable the respective graphs by different colors. Note that for a particular d we compute $\binom{d}{2}$ surfaces and, therefore, the same number of covariance matrices are required. These are

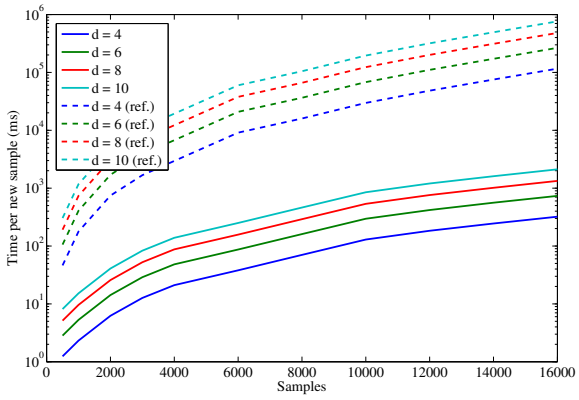

Figure 5: *Computation times in ms for inverting the covariance matrix when samples are added incrementally. Times for different numbers of dimensions d are considered and compared with a direct non-incremental approach as reference.*

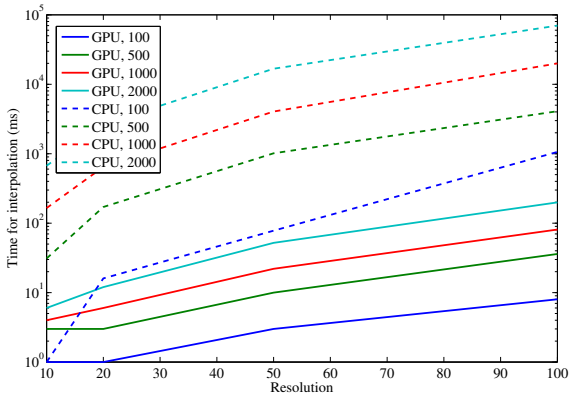

Figure 6: *GPU/CPU Computation times for kriging interpolation at different resolutions and for different numbers of samples.*

computed and inverted in parallel on the GPU. We compare the run-times to those being achieved with a state-of-the-art GPU matrix inversion method as proposed by Ezatti et. al. [EQOR11]. Here for each new sample the whole kriging matrix has to be inverted at once. The timings are shown in Fig. 5 as dashed lines. By using a logarithmic scale one can observe that the progressive approach is by several magnitudes faster than the conventional GPU approach.

Before the kriging interpolation can be performed, the inverse kriging matrices have to be computed using the inverse covariance matrices. This step needs to be carried out only once before the interpolation takes place, and its time consumption is negligible compared to that of the other parts.

The performance of the final interpolation of data estimates at the points of the discrete sampling grids has been measured for different data sets comprising 100, 500, 1000 and 2000 samples, and at different grid resolutions of 10×10, 20×20, 50×50 and 100×100. The timing statistics is shown in Figure 6. We compared the times of our GPU-based implementation and its CPU counterpart, computed on an Intel Xeon X5675 CPU at 3.07 GHz. It can be clearly seen that the GPU method is significantly faster and allows moving the center point nearly in real-time even for a large number of samples. Since the resolution can be changed at any time without affecting the kriging matrix, it is possible to use a lower resolution grid while adding new samples or moving the center point, and switching to a higher resolution grid for analyzing the surfaces in more detail.

7. Conclusion

In this paper we presented a novel progressive approach for visualizing multidimensional data via response surfaces. For computing high-quality response surfaces we employed kriging interpolation on the GPU. In contrast to previous approaches, our method enables progressive updates of the surfaces at interactive rates. In this way, we do not rely on any fixed sample set for which the surfaces are computed in a preprocess, but we can handle newly arriving samples—possibly retrieved based on user navigation—at very high speeds. We have shown that our method can be integrated effectively into existing techniques for multidimensional data visualization such as HyperSlices and projection.

In the future we will investigate the following aspects in more detail: Firstly, we will consider integrating GPU-kriging as proposed in Huang et al. [HCL*11] to accelerate the interpolation step. Even though this step is not the bottleneck in our current implementation, it might become so when interpolating on higher resolution sampling structures or in 3D. Secondly, and most importantly, we will use our tool to investigate whether the response surfaces' topography can guide the user towards specific features in the multivariate scalar functions. Even though it was observed by Tory et al. [TSD09] that colored point plots in a 2D domain can be remembered easier by the viewer than 2.5D visualizations such as response surfaces, the surface representation can encode local features and their relative positions and elevations in a more effective way. Local fluctuations of the surface, especially in regions where the sample set is sparse, can be easier quantified. In the future we will in particular analyze functions for which the locations of extreme points are known, and for which we then examine the occurrence of response surfaces in the vicinity of such points in order to derive shape-based feature indicators. Ideally we would find specific occurrences of response functions which guide towards locally maximum or minimum locations, even though the samples at which the extreme values occur have not yet been generated.

Acknowledgment

This work was supported by the European Union under the ERC Advanced Grant 291372—SaferVis—Uncertainty Visualization for Reliable Data Discovery.

References

[Asi85] ASIMOV D.: The grand tour: a tool for viewing multidimensional data. *SIAM J. Sci. Stat. Comput. 6*, 1 (Jan. 1985), 128–143. 3

[Aut93] Auto mpg data set. http://archive.ics.uci.edu/ml/datasets/Auto+MPG, 1993. UCI Machine Learning Repository. 6

[Ban37] BANACHIEWICZ T.: Zur Berechnung der Determinanten, wie auch der Inversen, und zur darauf basierten Auflösung der Systeme linearer Gleichungen. *Acta Astronomica, Serie C*, 3 (1937), 41–67. 2, 5

[BK05] BOTSCH M., KOBBELT L.: Real-time shape editing using radial basis functions. In *Computer Graphics Forum* (2005), pp. 611–621. 3

[BPFG11] BERGER W., PIRINGER H., FILZMOSER P., GRÖLLER E.: Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Published in Computer Graphics Forum 30*, 3 (2011), pp. 911 – 920. 3

[Buh03] BUHMANN M.: *Radial basis functions theory and implementations*. Cambridge University Press, Cambridge New York, 2003. 3

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 67–76. 3

[CJ08] CRESSIE N., JOHANNESSON G.: Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society: Statistical Methodology 70*, 1 (2008), 209–226. 4

[Cle85] CLEVELAND W. S.: *The elements of graphing data*. Wadsworth Publ. Co., Belmont, CA, USA, 1985. 3

[Cre93] CRESSIE N.: *Statistics for Spatial Data*. Wiley Series in Probability and Statistics. Wiley-Interscience, Jan. 1993. 3

[DGBW08] DICK C., GEORGII J., BURGKART R., WESTERMANN R.: Computational steering for patient-specific implant planning in orthopedics. In *Proceedings of the First Eurographics conference on Visual Computing for Biomedicine* (2008), pp. 83–92. 8

[EQOR11] EZZATTI P., QUINTANA-ORTÍ E., REMON A.: High performance matrix inversion on a multi-core platform with several gpus. In *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on* (2011), pp. 87 –93. 5, 9

[FB94] FURNAS G. W., BUJA A.: Prosection views: Dimensional inference through sections and projections. *Journal of Computational and Graphical Statistics 3* (1994), 323–385. 3

[FN91] FRANKE R., NIELSON G. M.: Scattered data interpolation and applications: A tutorial and survey. In *Geometric Modelling: Methods and Their Applications*, Hagen H., Roller D., (Eds.). Springer, 1991, pp. 131–160. 3

[GTC01] GRINSTEIN G., TRUTSCHL M., CVEK U.: High-dimensional visualizations. In *Workshop on Visual Data Mining* (2001), 7th Conf. on Knowledge Discovery and Data Mining (KDD), pp. 77–87. 3

[Haa95] HAAS T. C.: Local prediction of a spatio temporal process with an application to wet sulfate deposition. *American statistical Association 90*, 432 (1995), 1189–1199. 4

[Har07] HARRIS M.: Optimizing parallel reduction in cuda, 2007. NVIDIA Developer Technology. 6

[HCL*11] HUANG L., CHEN K., LAI Y., CHANG P., SONG S.: Geological visualization system with gpu-based interpolation. *AGU Fall Meeting Abstracts* (Dec. 2011), B1600. 4, 9

[Ins85] INSELBERG A.: The plane with parallel coordinates. *The Visual Computer 1*, 2 (Aug. 1985), 69–91. 3

[KH10] KIRK D., HWU W.-M.: *Programming massively parallel processors : a hands-on approach*. Morgan Kaufmann Publishers, 2010. 5

[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 (Dec. 1951), 119–139. 2, 3

[Mat63] MATHERON G.: Principles of geostatistics. *Economic Geology 58*, 8 (1963), 1246–1266. 3

[MB62] MATHERON G., BLONDEL F.: Traité de géostatistique appliquée, Tome I. *Memoires du Bureau de Recherches Geologiques et Minieres 14* (1962). 3

[MM95] MYERS R. H., MONTGOMERY D. C.: *Process Improvement with Steepest Ascent, The Analysis of Response Surfaces, Experimental Designs for Fitting Response Surfaces*, 1st ed. John Wiley & Sons, Inc., New York, NY, USA, 1995, pp. 183–351. 3

[Mon06] MONTGOMERY D. C.: *Response surface method and designs*. John Wiley & Sons, New Jersey, 2006. 3

[Nea99] NEAL R. M.: Regression and classification using Gaussian process priors (with discussion). *Bayesian Statistics 6* (1999), 475–501. 3

[OK78] O'HAGAN A., KINGMAN J. F. C.: Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society. Series B (Methodological) 40*, 1 (1978), 1–42. 3

[Opp98] OPPER M.: On-line learning in neural networks. 1998, ch. A Bayesian approach to on-line learning, pp. 363–378. 2

[PBK10] PIRINGER H., BERGER W., KRASSER J.: Hypermoval: Interactive visual validation of regression models for real-time simulation. *Computer Graphics Forum 29*, 3 (2010), 983–992. 3

[Sco92] SCOTT D. W.: *Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*, 1 ed. Wiley, Sept. 1992. 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* (2012), 2305–2314. 3

[TSD09] TORY M., SWINDELLS C., DREEZER R.: Comparing dot and landscape spatializations for visual memory differences. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1033–1040. 9

[TWSM*11] TORSNEY-WEIR T., SAAD A., MÖLLER T., HEGE H.-C., WEBER B., VERBAVATZ J.-M.: Tuner: principled parameter finding for image segmentation algorithms using visual response surface exploration. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1892–1901. 3

[vWvL93] VAN WIJK J. J., VAN LIERE R.: Hyperslice: visualization of scalar functions of many variables. In *Proceedings of the 4th conference on Visualization '93* (1993), pp. 119–125. 1, 2, 3

[Weg90] WEGMAN E. J.: Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association 85* (1990), 664–675. 3

[Wen05] WENDLAND H.: *Scattered data approximation*. Cambridge University Press, Cambridge, UK New York, 2005. 3