DOI: 10.1111/cgf.12912
Eurographics Conference on Visualization (EuroVis) 2016
K.-L. Ma, G. Santucci, and J. van Wijk
(Guest Editors)

Volume 35 (2016), Number 3

Critical Points of Gaussian-Distributed Scalar Fields on Simplicial Grids

T. Liebmann[1] and G. Scheuermann[1]

[1]Department of Computer Science, Leipzig University, Germany

Abstract

Simulations and measurements often result in scalar fields with uncertainty due to errors or output sensitivity estimates. Methods for analyzing topological features of such fields usually are not capable of handling all aspects of the data. They either are not deterministic due to using Monte Carlo approaches, approximate the data with confidence intervals, or miss out on incorporating important properties, such as correlation. In this paper, we focus on the analysis of critical points of Gaussian-distributed scalar fields. We introduce methods to deterministically extract critical points, approximate their probability with high precision, and even capture relations between them resulting in an abstract graph representation. Unlike many other methods, we incorporate all information contained in the data including global correlation. Our work therefore is a first step towards a reliable and complete description of topological features of Gaussian-distributed scalar fields.

1 Introduction

Critical points are important features of scalar fields. They build the foundation for topological analysis of isocontours as they represent events of structural change of level sets with varying isovalue. Because scalar fields often originate from measurements and simulations, uncertainty in the data is the rule rather than the exception. One common source of uncertain scalar fields are ensemble simulations. By applying small perturbations to the input parameters, indicators for output reliability can be derived. To then model these errors into the data, several approaches such as deriving further indicators [PGA13], approximating the value distributions [PH13] [PKXJ12], or directly operating on the finite set of ensemble members [LPK05] exist. A widely used representation for uncertainty is Gaussian distributions as they require only a mean and variance at every point of the domain and provide good approximation in error modelling. For Gaussian-distributed scalar fields, the usual topological definitions are not applicable directly. Even critical points – the basis of topological analysis – become fuzzy. Most methods are only capable of approximating indicators for critical points or miss out on using all important properties of the data. One property often not completely incorporated is local and especially global correlation. By visualizing correlation using clustering [PW12] and glyphs [PW13], Pfaffelmoser and Westermann show that some important features can only be discovered from correlation. Therefore it is advisable to handle this relation between grid points as a major property of the data.

In this paper we provide a deterministic method for extracting critical points that can appear in the realizations of a Gaussian-distributed scalar field defined over a simplicial grid. For this, we normalize the space of all possible realizations and investigate the local neighborhoods of the domain points. This leads to the introduction of *singular patches* as representations of critical points. Singular patches enable us to not only extract probabilities of critical points with high precision, but also build an abstract representation as basis for identifying more complex topological features.

2 Related Work

Uncertainty in data is an important factor when it comes to drawing conclusions and modeling abstractions. As it is a relevant topic in scientific visualization [JS03] [Joh04], many publications deal with extending visualizations and algorithms to handle uncertainty or with developing new methods. Some overviews over existing visualizations using different techniques like texture, color-mapping, animation, and additional geometry can be found in [PWL97] [MRH*05] [PRJ12] [BOL12] [GS06].

The topology of deterministic discrete fields allows for a more abstract analysis and has been studied intensively. Especially for topological features of scalar and vector fields many robust and efficient methods have been developed and improved. Based on Morse theory [Mor34], the Reeb graph [Ree46] is used in many applications to extract critical points, level sets, and their relations. Even though there are several algorithms available that deal with fast and stable computation of the Reeb graph [PSBM07] [ATC*08] and the contour tree [PCM02] [CSA03] [CLLR05] – the Reeb graph of a scalar field over a simply-connected domain – none of them can be applied directly in case of uncertain data.

The influence of uncertainty on level sets has been studied by

Pöthkow and Hege [PH11] by introducing level probabilistic measures for crossing isovalues in cells. They use this to extract uncertain isocontours [PWH11]. Pfaffelmoser et al. [PRW11] extend the measures to also address correlation in the data. Uncertain contour trees have been visualized by Kraus [Kra10] by combining two trees representing the range of variation and therefore reducing the data to confidence intervals. Wu and Zang [WZ13] visualize uncertainty on three levels – data, contours, and topology – combining contour trees of multiple but only a finite amount of realizations.

For uncertain vector fields, Otto et al. [OGHT10] redefine streamline integration to extract the probabilistic topological skeleton. Their computation, however, is very expensive due to heavily relying on sampling and does not incorporate correlation.

Since critical points play an important role in the overall topological structure, our focus lies on their identification, extraction, and relation in uncertain scalar fields. Petz et al. [PPH12] extract critical points in uncertain vector fields using local classification based on a point's neighborhood in the grid. They do this by limiting the distribution to the local neighborhood for each point. By only considering this marginal distribution, they may lose features hidden in global correlation and are not able to capture structural information between critical points.

In uncertain scalar fields, Günther et al. [GST14] identified mandatory critical points – regions, in which critical points of a specific type occur in all realizations. For saddle points, their algorithm is very expensive and not applicable to large datasets. They also do not incorporate correlation and limit uncertainty representation to a finite support of the density function – a confidence region for each point not dealing with the actual type of the distribution. Another approach is given by Mihai and Westermann [MW14] propagating uncertainty from Gaussian distributions in the domain to gradients and the Hessian matrices. This allows for extracting regions containing potential critical points and filter them by confidence. As they limit the distribution to the local neighborhood of a point, only considering the marginal distribution, they lose global correlation. Even though they can extract positional indicators, the structure and relation between points is still unclear.

3 Background

As representations of scalar fields and especially uncertainty in scalar fields differ across publications, this section gives the basic notions used in this paper.

Scalar Fields – We call the mapping $s : D \to \mathbb{R}$ a scalar field over a domain $D \subset \mathbb{R}^d$. To allow for combinatorial analysis, the domain usually is approximated using a finite set of sample points $D_S = \{x_1, \ldots, x_n\} \subset D$. By defining a neighborhood $N : D_S \to \mathcal{P}(D_S)$ with $\mathcal{P}$ denoting the power set, one can span a grid across the approximated region. Furthermore we require the grid to be connected and consists only of simplicial cells in which we assume linear interpolated scalar values.

Uncertain Scalar Fields – In contrast to a deterministic scalar field, an uncertain scalar field U can be treated as a multivariate random variable $U : \Omega \to \mathbb{R}^n$. It maps every element ω of the sample space Ω to a realization $U(\omega) \in \mathbb{R}^n$ for which every component $(U(\omega))_i$ represents the scalar value of one sample point x_i.

We focus on Gaussian-distributed data, as it is an established model and used in many publications. A Gaussian-distributed scalar field $U \sim \mathcal{N}_n(\mu, \Sigma)$ is fully described by the mean vector $\mu \in \mathbb{R}^n$ and the covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$. Every sample point x_i has a marginal distribution of $U_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ with σ_i being the standard deviation. For the sake of readability, we often write vectorial elements such as μ and σ as mappings $\mu : D_S \to \mathbb{R}$, mapping a value to each sample point. The covariance matrix encodes not only the individual variance, but also the correlation between every pair of sample points. Every entry can be decomposed into $\Sigma_{i,j} = \rho(x_i, x_j) \cdot \sigma_i \cdot \sigma_j$, where $\rho : D_S \times D_S \to [-1, 1]$ is the Pearson product-moment correlation coefficient. A correlation coefficient of 1 or -1 signals direct/indirect proportionality between two random variables whereas 0 describes no dependency at all. Note that nonlinear dependencies cannot be modelled using this representation. In real data, these correlations can either be part of the dataset, e.g. they arise from modelling the measurement environment or simulation, or they can be estimated using statistical methods. For further details and an example see Section 13.

The interpolation between sample points in Gaussian fields is not trivial. To ensure that critical points of realizations are located on sample points, we assume a linear interpolation as used by most publications. Although Schlegel et al. [SKS12] have shown that interpreting the whole field as a Gaussian process, more accurate assumptions on the distributions of points within cells can be made, this increases the theoretical complexity by a lot and only has a notable effect in poorly sampled regions.

Scalar Field Topology – The topology of scalar fields often is defined in terms of the evolution and relation of level sets. A level set for a function $s : D \to \mathbb{R}$ is $L_s(\alpha) = \{x \in D \mid s(x) = \alpha\}$, namely all points of the domain mapped to certain function value α. We denote a connected component of a level set as a contour. Altering α leads to structural changes in the level sets at certain points of the domain. These points are called critical points and can be divided into three types. At minima/maxima, new contours appear or existing contours vanish whereas saddle points represent merging or splitting contours or a change of their genus numbers.

Focusing on uncertain scalar fields, the aforementioned definitions do not apply directly. With every point theoretically evaluating to every possible scalar value, assumptions about level sets, contours, and singularities become fuzzy. We show, however, in the following sections that by looking at the space of all realizations and transferring the concept of a critical point into this space, we can investigate topological features of Gaussian-distributed fields.

4 Overview

Figure 1 gives an overview over all steps we take to extract critical points, their properties, and structure. The main goal is to compute *singular patches* – a representation of critical points in the context of uncertain Gaussian scalar fields. A patch stands for a specific scalar value configuration in the neighborhood of a specific grid point. As the topological type of a point only depends on the local neighborhood, we first abstract the value difference to all neighbors introducing *neighbor configurations*. To handle uncertainty, we look at the space of all possible realizations and combine this

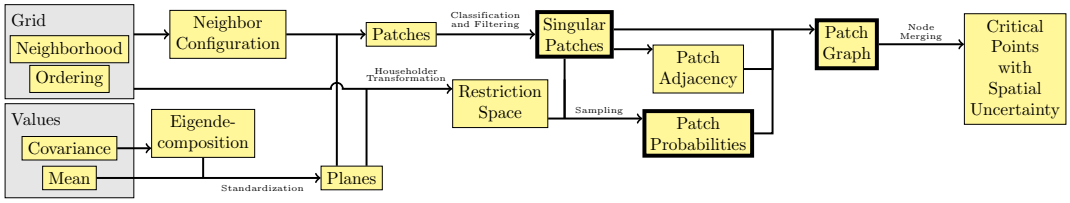


Figure 1: *Overview over our method including intermediate steps and relevant structures. Most important results are marked with thick outlines. Starting with the dataset, our method extracts critical points, their probability, and even relations.*

with the neighbor configurations. This leads to singular patches as segments on hyperplanes in a high-dimensional space. Based on these segments we then approximate probabilities using sampling in low dimensional subspaces and even extract relations between singular patches. This leads to the *patch graph* – an abstract structure to represent all possible critical points and their behaviour in the space of all realizations. Besides describing basic features that can be found in the graph, we emphasize its advantage by providing a merging method that leads to the identification of critical points that change their locations across different realizations.

5 Realization Space

With U being an n-dimensional random variable, every point in $\mathbb{R}^n$ represents exactly one realization of U. In this space, the positive semidefinite covariance matrix Σ describes an ellipsoid expressing the variance in every direction. We first standardize this space by transforming it so that the underlying distribution becomes a standard normal distribution. By diagonalizing Σ into $\Gamma\Lambda\Gamma^T$ using the eigenvalue decomposition, we can rewrite U as

$$U = \mu + \Gamma\Lambda^{\frac{1}{2}}Z. \tag{1}$$

With $Z \sim \mathcal{N}_n(0, I_n)$ being a standard normal distributed random variable. Here, $\Lambda \in \mathbb{R}^{n\times n}$ is the diagonal matrix with eigenvalues $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$. $\Gamma \in \mathbb{R}^{n\times n}$ is the matrix containing the corresponding eigenvectors $\gamma_1, \ldots, \gamma_n \in \mathbb{R}^n$ as columns. All this equation does is first applying the deviations of U by the multiplication with $\Lambda^{\frac{1}{2}}$. Γ then aligns the main directions to the ones described by Σ. Adding μ to translate the mean then completes the transformation.

Since the eigenvalues represent the variances in the main directions, which now are the main axes of the space of Z, we can reduce the dimension by removing axes with eigenvalues below a certain threshold $\theta > 0$. W.l.o.g. let the eigenvalues be sorted in descending order. For the rest of the paper we denote the number of significant eigenvalues as d. For a point $z \in \mathbb{R}^d$, the value of a realization of U at a grid point x can be written as

$$U_z(x) = \mu(x) + \sum_{k=1}^{d} \gamma_k(x)\sqrt{\lambda_k} z_k. \tag{2}$$

This standardization process is well known in many different disciplines (e.g. as *Principal Component Analysis* [Hot33]). For us, it allows to look at uncertain scalar fields in a well-defined space while also reducing the amount of free parameters. The degree of reduction can be controlled by the expressive parameter θ.

For most of the paper, we refer to $\mathbb{R}^d$ as *realization space*. Furthermore, we use U_z as additional dimension leading to the *extended realization space*. In the extended realization space, the function graph of Equation (2) describes a hyperplane for every grid point as it is a linear combination of the components of z. This can be seen more clearly in Figure 2. While Figure 2 (c) shows the realization space, Figure 2 (b) has the aforementioned extension to express scalar values of grid points as hyperplanes.

6 Critical Points in Realization Space

Identifying and classifying critical points on simplicial grids can be done locally. Edelsbrunner et al. [EHP08] use reduced Betti numbers to identify simple critical points. Using the lower link of a grid point, namely all adjacent grid points having a smaller scalar value and all connections between them, based on the number of holes and connected components a classification can be made. In our work, we use a notation based on ordering the local neighborhood and encode the lower and upper links in a neighbor configuration. This allows for a consistent usage in further steps while also reflecting the underlying implementation.

6.1 Neighbor Configuration

Since the topological type of a point only depends on the relative value to its neighbors, we first abstract this neighborhood. For this, we use the signum function of the value difference

$$\operatorname{sgn}\big(U_z(x_i) - U_z(x_j)\big) \tag{3}$$

which either evaluates to -1, 0, or 1 depending on whether the value of point x_i is lower, equal, or higher than the value of x_j in a certain realization z. To extend this to all neighbors and allow for a algebraic notation, we first define an ordering on the set of neighbors $N(x)$ for every grid point $x \in D_S$. This is done by the mapping

$$o_x : \{1, \ldots, |N(x)|\} \to N(x) \tag{4}$$

with $\forall i, j \in \{1, \ldots, |N(x)|\} : o_x(i) = o_x(j) \Leftrightarrow i = j$. This allows for retrieving the i-th neighbor of x with $o_x(i)$. How one chooses the ordering can be arbitrary. Usually it is given implicitly through the description of the grid or can be computed easily (e.g. by ordering neighbors in counter clockwise direction in 2-dimensional grids). Using the same ordering across the grid is not necessary and, in fact, not always possible due to different neighborhood sizes.

To express the relation of one point to all its neighbors, we introduce the *neighbor configuration*. We define it as a mapping $c_x : \mathbb{R}^d \to \{-1, 0, 1\}^{|N(x)|}$ with

$$(c_x(z))_i = \operatorname{sgn}(U_z(x) - U_z(o_x(i))). \tag{5}$$

c_x depends on the point $x \in D_S$ as the neighborhood can vary across the grid. It also maps different values to different realizations $z \in \mathbb{R}^d$, due to the actual values at the grid positions being dependent on the realization.

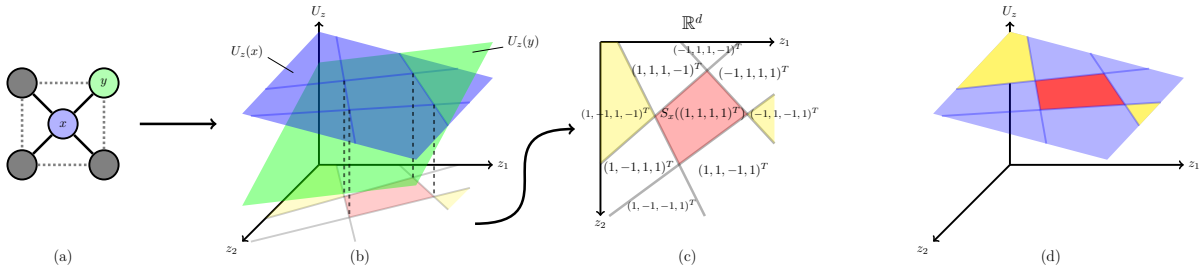


Figure 2: *Different spaces used to define singular patches based on a 2-dimensional realization space. A grid point x and all of its neighbors, e.g. y, are represented as hyperplanes in the extended realization space (b). Here $U_z(x)$ and $U_z(y)$ may have an intersection denoting all realizations with equal value at x and y. Projecting this intersection and all the ones for x's other neighbors into the space spanned by z_i's (the realization space) it decomposes this space (c) into sets of realizations with x having the same neighbor configuration. We call these regions patches. Classifying patches based on the configuration leads to singular patches representing maxima (red), saddles (yellow) and minima (not present in this example). Combining the regions in realization space (realization sets) with the point's plane, one can also think of patches as regions on hyperplanes in the extended realization space (d).*

6.2 Classification

Up to this point, c_x only describes the relation between a grid point's scalar value and the values of all its neighbors. To decide whether this configuration belongs to a certain type of critical point, a classification has to be made.

Identifying a maximum (max) or minimum (min) is straightforward as it is independent from any kind of ordering. For a neighborhood size of $|N(x)| = 4$, e.g. a maximum is clearly indicated by $c_x(z) = (1,1,1,1)^T$, while a minimum corresponds to $c_x(z) = (-1,-1,-1,-1)^T$. For saddle points (sad), however, the ordering is crucial. A simple saddle point represents the split or join of two or more contours. Therefore, at the point's scalar value, two or more contours have to intersect the point. Locally, this can be decided by counting adjacent cells that lead to a contour traversing through the point. E.g., in a 2-dimensional grid with neighbor ordering by angle and a neighborhood size of $|N(x)| = 4$, only $c_x(z) = (1,-1,1,-1)^T$ and $c_x(z) = (-1,1,-1,1)^T$ indicate saddle points. Grid points located on the border of the domain are exceptions to this saddle classification. Here, a saddle point cannot be decided locally as its type depends on the course the contours take across the domain. We handle those cases by introducing a separate type (bsad) to represent potential border saddles that only require one intersecting contour. Including regular points (reg), the topological classification can be formalized with a mapping:

$$t_x : \{-1,1\}^{|N(x)|} \to \{\text{reg}, \text{max}, \text{min}, \text{sad}, \text{bsad}\}. \tag{6}$$

We deliberately exclude cases with $(c_x(z))_i = 0$, namely one or more neighbors having the same scalar value. Depending on the concrete type, those cases can form topological borderline cases with critical points being expanded over multiple grid points. In Morse theory, these borderline cases are called degenerate critical points and are usually avoided by applying small distortions to the affected grid points or by using some further ordering criteria, which is often referred to as *Simulation of Simplicity* [EM90]. Due to the seamless realization space, we are not able to change scalar values of single realizations without affecting fundamental properties of the space. By excluding those cases in all further steps, we avoid those problems and still keep consistency.

6.3 Realization Set

The concept of the neighbor configuration can also be looked at from a different perspective. Instead of describing the relation a point has to its neighbors in a certain realization, one can also ask, which realizations have a specific neighbor configuration. For this, we use the following inequation:

$$r \cdot (U_z(x_i) - U_z(x_j)) > 0. \tag{7}$$

With this, we can find all realizations that share a specific relation between two grid points x_i and x_j only by choosing $r \in \{-1,1\}$ – again excluding cases with $U_z(x_i) = U_z(x_j)$. E.g., all realizations with x_i having a greater value than x_j fulfill Inequation (7) with $r = 1$. Together with a given neighbor configuration $\bar{c} \in \{-1,1\}^{|N(x)|}$ we can extend this to all neighbors of a grid point x by writing

$$\bar{c}_i \cdot (U_z(x) - U_z(o_x(i))) > 0. \tag{8}$$

Furthermore, equation (2) gives us an exact description of the points' values, leading to

$$\bar{c}_i \cdot \sum_{k=1}^{d} \underbrace{\left(\gamma_k(x) - \gamma_k(o_x(i))\right)\sqrt{\lambda_k}}_{(A_x)_{i,k}} z_k > \bar{c}_i \cdot \underbrace{\left(\mu(o_x(i)) - \mu(x)\right)}_{(b_x)_i}. \tag{9}$$

With a matrix $A_x \in \mathbb{R}^{|N(x)| \times d}$ and a vector $b_x \in \mathbb{R}^{|N(x)|}$ with entries defined as indicated by the above formula we can write the whole system of inequalities for a specific neighbor configuration as

$$C_{\bar{c}} \cdot A_x \cdot z \succ C_{\bar{c}} \cdot b_x. \tag{10}$$

Here, $C_{\bar{c}} = \text{diag}(\bar{c}_1, \ldots, \bar{c}_{|N(x)|})$ is a diagonal matrix multiplying the i-th entry of $\bar{c}$ to the i-th row of the system of inequalities and $\succ$ is the component-wise greater relation.

With this short notation, one can describe regions in realization space with a grid point having a certain neighbor configuration. We will also refer to this as the *realization set* $S_x : \{-1,1\}^{|N(x)|} \to \mathbb{R}^d$ with

$$S_x(\bar{c}) = \{z \in \mathbb{R}^d \mid C_{\bar{c}} \cdot A_x \cdot z \succ C_{\bar{c}} \cdot b_x\} \tag{11}$$

as the set of all realizations with a point $x \in D_S$ having a neighbor configuration $\bar{c}$.

7 Singular Patches

By *patch* we denote a region in the realization space with a grid point having a specific neighbor configuration. A patch p therefore

is defined as a tuple $p := (x_p, c_p)$ with $x_p \in D_S$ being a grid point and $c_p \in \{-1,1\}^{|N(x_p)|}$ describing a neighbor configuration of x_p. We again exclude cases with $(c_p)_i = 0$ as we want to be able to classify all defined patches. It is worth noting because of this, the realization set $S_{x_p}(c_p)$ of a patch always is an open set. As *singular patches* we denote those patches that lead to a grid point being a critical point in the corresponding realizations. Therefore, a patch $p = (x_p, c_p)$ is singular, iff $t_{x_p}(c_p) \neq \text{reg}$.

Figure 2 summarizes the conception of singular patches on the example of a point x with $|N(x)| = 4$. Despite the shown realization space only having dimension 2, relations and structures in the figure directly translate into arbitrary dimensions. Figure 2 (d) shows patches interpreted as regions on hyperplanes. This becomes especially useful when dealing with patch relations in Section 9.

7.1 Patch Filtering

Even though the above definitions lead to a finite amount of singular patches, the actual number can be very large. Our example in Section 13 operates on a 2-dimensional regular grid with every grid point having 6 neighbors. This neighborhood results in $2^6 = 64$ different neighbor configurations of which 34 describe critical points. Therefore, there are $34 \cdot n$ singular patches which have to be handled in further analysis steps. The following sections describe two methods reducing the amount of patches significantly.

Empty Patches – Due to correlation and the elimination of directions with low variance in Section 5, for a patch $p = (x_p, c_p)$, the set of realizations $S_{x_p}(c_p)$ can turn out to be an empty set. As the system of inequalities (10) forms a convex polytope in the d-dimensional space, emptiness can be checked using well established methods from optimization theory. Many efficient and stable algorithms are available with runtimes depending on the number of restrictions and the amount of variables to consider. In our implementation we use the linear and quadratic programming algorithms of the CGAL-library [FGSW15] as it provides good performance and stability.

Region of Interest – To further reduce the amount of patches, we use a confidence region in the realization space to remove patches representing critical points that are very unlikely. Since the standardization resulted in a fully symmetric probability density function, we can use a spherical region around the origin. The radius describes how many realizations still are covered with the remaining patches intersecting the confidence region. To make this more expressive, we choose a threshold $\delta \in [0,1]$ which denotes the percentage of realizations that have to be covered. Based on this threshold, one has to find the radius of the confidence region, which cannot be done in an analytical way. Since we only want to make sure we cover enough realizations, the radius can be overestimated. Therefore, one simple numerical computation per dataset can be used to estimate an upper bound for this radius. To extract only patches with realization sets having a non-empty intersection with the confidence region, we can, again, use optimization methods. For a patch $p = (x_p, c_p)$, by solving the quadratic program

$$z^T z \to \text{Min!}, \quad C_{c_p} \cdot A_{x_p} \cdot z \succ C_{c_p} \cdot b_{x_p}, \tag{12}$$

the minimal distance of the patch's polytope to the origin can be found. By simply comparing it with the radius of the confidence region, we can decide whether to eliminate or keep it. Although we reduce the amount of patches, we keep a consistent and complete coverage of the defined region in realization space. This is important for further steps investigating the structure of singular patches. Also note that this step is not necessary to make our method applicable but reduces computation time of subsequent steps.

7.2 Patch Sampling

Given a patch $p = (x_p, c_p)$, we are interested in extracting the probability for the point x_p taking neighbor configuration c_p. To do so, we have to integrate the probability density function over the polytopic realization set $S_{x_p}(c_p)$. Just like in the previous section, this is, however, not possible in an analytical way. For this reason we approximate the probability using sampling. The naive approach would be to draw k sample points from $\mathcal{N}_d(0, I_d)$ and count the amount of samples k_{in} that are part of the realization set $S_{x_p}(c_p)$. The probability is then approximated by $\frac{k_{\text{in}}}{k}$. The main drawback of this method is the usual high dimension d. Not only does this seem to increase the amount of required sample points tremendously but also makes the inside test for the polytopic region expensive. In the following section, however, both problems get resolved by limiting the sampling method to a low dimensional subspace.

8 Restriction Space

The realization set of a patch $p = (x_p, c_p)$ is a convex region in the d-dimensional space. Sampling this region to retrieve the probability of a patch therefore might seem to require a number of sample points that depends on d. This is, however, not the case as the convex region is bound by only a small number of restrictions – namely the number of neighbors $|N(x_p)|$ of the patch's grid point. As $|N(x_p)|$ usually is much smaller than d, the convex region is unbounded in most of the directions. Sampling the multidimensional Gaussian kernel in an unbounded direction always results in a factor of 1 and therefore can be omitted. To only sample in bounded directions, we extract a subspace – we call it *restriction space* – for every grid point. The bounded directions are the normals of the restrictions. As these normals appear as row vectors in the already computed matrix A_{x_p}, all there is left to do is to define the space spanned by these vectors by extracting a basis. This can be done by orthonormalization. We chose the Householder transformation [Pre07] as it provides high numerical stability. Because it is defined operating on column vectors, we apply it to ${A_{x_p}}^T \in \mathbb{R}^{d \times |N(x_p)|}$ resulting in a so-called QR-decomposition

$${A_{x_p}}^T = Q_{x_p} \cdot R_{x_p}. \tag{13}$$

Here, $Q_{x_p} \in \mathbb{R}^{d \times d}$ is an orthogonal matrix representing a basis transformation and $R_{x_p} \in \mathbb{R}^{d \times |N(x_p)|}$ is an upper triangular matrix containing the restriction normals in the new subspace. With this, the system of inequalities (10) describing the realization set of a patch $p = (x_p, c_p)$ can be rewritten as

$$C_{c_p} \cdot {R_{x_p}}^T \cdot \underbrace{Q_{x_p}^T \cdot z}_{z'} \succ C_{c_p} \cdot b_{x_p}. \tag{14}$$

Even though z' is also an element of $\mathbb{R}^d$, due to $R_{x_p}^T$ being a lower triangular matrix, only $|N(x_p)|$ components contribute to the system. This shows that we only have to generate a fairly low amount

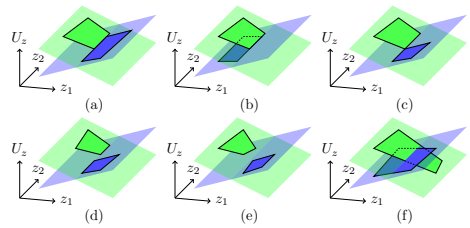


Figure 3: *Different situations of patches in the extended realization space with them either being adjacent (a,b,c) or not (d,e,f). We differentiate between horizontal (a), vertical (b), and higher order (c) adjacency.*

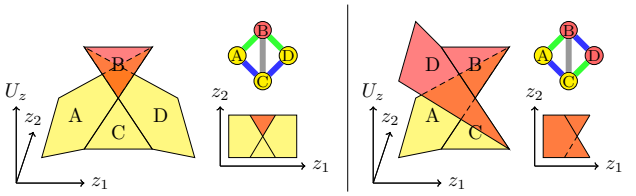


Figure 4: *Exemplary horizontal (a) and vertical (b) folds in a 2-dimensional realization space (bottom right). In extended realization space (left) two patches overlap forming two other patches. Folds appear in the patch graph as special patterns (top right). Maximum patches are colored red while saddle patches are yellow. The edges of the graph indicate vertical (green), horizontal (blue), and order 2 (gray) adjacency.*

of samples in a subspace of dimension $|N(x_p)|$, which gives us a high precision approximation of the patch probabilities. Furthermore, the orthonormalization preserved the standard normal distribution due to the matrix Q_{x_p} being orthogonal, which keeps the sample generation simple. Memory consumption is also low as the triangular matrix R_{x_p} only requires to store at most $\frac{1}{2}|N(x_p)|^2$ values for every grid point.

The other methods used in our pipeline operating on the realization set and therefore in d-dimensional space are the test for emptiness and the application of the thresholding from Section 7.1. As both require solving linear or quadratic programs, their computational cost depends on the underlying dimension. It turns out that these methods can also be applied in the subspace. The check for emptiness is trivial in that sense, as emptiness can only arise from bounded directions which are by definition preserved in the restriction space. As it is an orthogonal transformation preserving distances, also the patch filtering using a threshold radius around the center can be done in the subspace. Here, one only has to take the distance of the subspace in the omitted directions into account, which are given as right sides in the system of restrictions.

The restriction space is a key point in our computation, as it makes all of the used algorithms be only dependent on the neighborhood size and the number of grid points. Although it requires a preprocessing step, computing R_x for all grid points using the Householder transformation, this step is much faster and scalable than operating in the original space.

9 Singular Patch Adjacency

When looking at singular patches as convex regions on planes in the extended realization space, as shown in Figure 2 (d), an interesting observation can be made. Leaving the realization set of a patch over one of its restrictions leads to a change in the neighbor configuration of the corresponding grid point. If this results in the point changing its topological type, the Poincaré-Hopf theorem [Lib12] implies that some other point has to change its type too for the global topology to remain consistent. If the neighbor configuration changes with the point keeping its type, the point has to be a saddle patch, as it is the only type that has multiple configurations. In either case there has to be at least a second patch sharing the exact same edge of the convex region. This we call patch adjacency. In general, we call two patches *adjacent* iff they share a facet and their corresponding grid points also are adjacent in the grid. Figure 3 shows some common situations of two patches of two different grid points over a 2-dimensional realization space. (a), (b), and (c) are examples for the patches being adjacent. In (d) and (e), the intersection is empty resulting in both patches clearly not being adjacent. Case (f), however, despite both patches having a non-empty intersection, they are not adjacent by the definition above. The reason for this is that the corresponding grid points cannot be direct neighbors as otherwise the plane of one point would be a restriction on the plane of the other point. In the following we will describe different forms of patch adjacency and introduce two properties – order and direction – to classify adjacency.

As *order* of adjacency we denote the type of facet the patches share. In a two dimensional space, e.g., order 1 stands for a shared line (Figure 3 (a) and (b)) whereas order 2 represents a point (Figure 3 (c)). In three dimensions, order 1 is a plane, 2 a line, and 3 a point. It can be shown that the order always lies within the range $[1,d]$ with d being the dimension of the realization space. If two adjacent patches are located on the same plane, thus representing different configurations for the same grid point, the order is determined by the number of different digits in the neighbor configuration. If the patches are on different planes, the order is retrieved differently. First, for both patches to be adjacent at all, they have to be bordered by the restriction which comes from the two planes intersecting each other. We call this the *mutual restriction*. For the order to be higher than one, the patches have to share additional restrictions for which they have to lie in different half spaces. These additional restrictions can only arise from additional planes of grid points that are in the neighborhood of both grid points of the patches.

The second property we use for differentiating adjacency is the *direction*. If the patches are located on different planes, the directions states whether both lie on the same or on different sides of the mutual restriction. In case of both being located on the same side, we call it *vertical* adjacency (Figure 3 (b)), as the patches are on top of each other with respect to the scalar value axis U_z. If both lie on different sides, we call it *horizontal* (Figure 3 (a)) as they are side by side. As patches on the same plane cannot overlap each other, the adjacency is always classified as horizontal. The reason for this classification is that different types of adjacencies represent different topological changes when traversing over the corresponding facet in realization space. This becomes more clear in the following section.

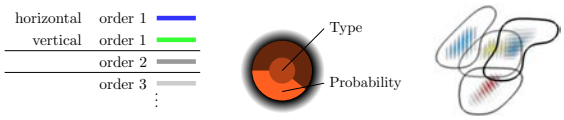


Figure 5: *Edge coloring (a) and node glyph (b) used for visualizing the patch graph. In (c), some regions indicating critical points with spatial uncertainty can be seen. The texture indicates the probability of the points being at a specific location within the region.*

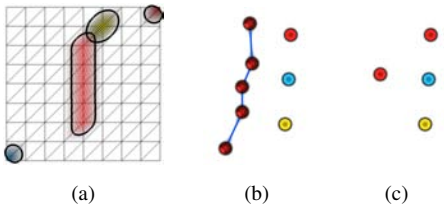


Figure 6: *Dataset containing a maximum with spatial uncertainty (a). Merging the patch graph (a) allows for identifying the varying maximum as an important critical point (b).*

10 Singular Patch Graph

To represent singular patches and their relations given by different types of adjacency, we build a graph structure. For every singular patch, we introduce a node augmented with all patch-specific properties like patch probability and topological type. Adjacency between patches is expressed via edges connecting the corresponding nodes. Edge properties are order and direction of adjacency. In this graph, several interesting structures can be identified. Some of them are described in the following.

Birth/Death of Branches – The first observation is that edges representing vertical adjacency of order 1 appear mostly between a saddle patch and an extremum patch. Traversing over the facet connecting the patches in realization space therefore leads to the birth/death of an extremum while also creating/erasing a saddle. This is expected, as for every extremum there has to be a saddle it is connected to to maintain a consistent topological situation. Note that in the context of contour trees, such adjacencies can also be interpreted as birth/death of branches.

Points with Spatial Uncertainty – Edges representing horizontal adjacency usually appear between singular patches of the same type. Like before, traversal over the corresponding facet in realization space leads to changes in the topology of the resulting scalar field. For patches belonging to the same grid point this simply means that the neighbor configuration changes without changing the type of the grid point. In case of patches on different planes, thus different grid points, this situation represents a critical point moving to a different grid point in the scalar field. The extraction of these critical points with spatial uncertainty can be quite relevant, as they allow for tracking critical points over different realizations.

Folds – Besides the basic connections between nodes representing local changes of topology, more complex structures can be extracted from the graph. One that can be found very frequently are folds. Two typical patch configurations and their corresponding nodes and edges in the patch graph are shown in Figure 4. In both cases, the patches A and D overlap each other forming the patches B and C. Again we can distinguish between horizontal (a) and vertical (b) folds based on the relation of the overlapping patches. The theory, exact structure, and possible side effects of folds require further research and are therefore not discussed in detail.

11 Node Merging

Although the focus of this paper lies on introducing and computing the singular patch graph, we want to emphasize its benefit by extracting points with spatial uncertainty. We do this by merging nodes that represent the same critical point having different locations in the domain. For two nodes to be merging candidates, they have to have the same topological type and must be connected by horizontal adjacency. By redirecting edges previously connected to either one of the nodes to the merged one, we can repeat merging until no more candidates are present. This process, however, does not preserve the property of merged patches not overlapping in realization space. The reason for this is the reconnection of horizontal adjacency edges not ensuring disjoint realization sets. Therefore we explicitly forbid merging nodes that have overlapping realization sets by using the emptiness check for polytopes.

Limiting the merging to non-overlapping patches is a strength and a weakness at the same time. On the one hand it ensures that at most one critical point of the merged set is present in any realization and therefore allows for summing up the probabilities from Section 7.2. On the other hand it introduces ambiguity and an expensive check for emptiness. In case of a horizontal fold, e.g., as shown in Figure 4, while patch C can either be merged onto A or D, A and D themselves cannot be merged together due to them overlapping. Despite these limitations it still allows for detecting critical points that otherwise would have been missed due to lack of prominence.

12 Visualization

For visualization of the patch graph we use two methods. Although the graph itself can be quite complex and therefore not suitable for direct visualization, we draw it to get first insights into its structure. Using a force-directed layout we can show a large amount of nodes only by specifying edge lengths. Edge properties can be expressed using not only these lengths but thickness and color. Furthermore, node properties, like patch probability and the topological type, are represented using glyphs. In Figure 5 the edge coloring (a) and node glyphs (b) are shown. As colors we use red for maxima, blue for minima, yellow for saddles, and green for border saddles.

The second visualization bridges the gap between the abstract singular patch graph and the uncertain scalar field by operating on the underlying grid. The main purpose is to communicate the results of merging critical points with spatial uncertainty and the computed probabilities. As can be seen in Figure 5 (c), we surround the region of a critical point by a thin border with transparency indicating the accumulated probability of all patches involved. To indicate the type as well as the probability distribution within such a region, we fill it with a stripe texture having a specific color. The stripes help distinguish overlapping regions. This method also allows for providing additional context information e.g. in the form of a map or color mapped mean field in the background.

#grid points	800	3600	5400	7200	9000	10800	12600	14400
patches	0:00:15	0:00:25	0:00:35	0:00:38	0:00:43	0:00:51	0:00:56	0:00:58
adjacency	0:45:50	1:19:23	1:29:48	1:26:02	1:26:37	1:33:52	1:39:31	1:38:13
merging	0:44:11	1:43:16	2:37:24	2:53:18	3:23:54	4:23:25	5:00:32	5:22:34

Table 1: *Computation times (h:mm:ss) of the main steps in our pipeline applied to different grid sizes.*

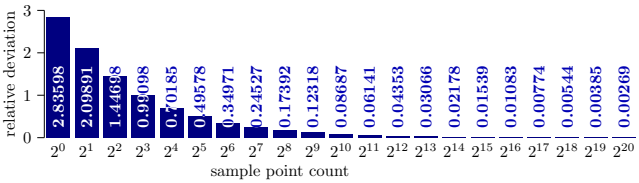


Figure 7: *An estimate on the relative sampling error by averaging the deviation at different sample counts.*

13 Examples

We applied our methods to two datasets. To show the basic concept of critical points with spatial uncertainty, we first look at a synthetic dataset. It also makes the advantage of our method compared to existing approaches clear. As second dataset we chose a temperature forecast, which represents a typical real-world example.

13.1 Synthetic dataset

We generated a synthetic dataset to show a typical situation in which existing methods fail to detect a significant feature that our method is able to find. Based on a 2-dimensional regular 9×9 grid, we chose the covariance matrix and mean vector to create a specific topological situation. First, there are three critical points which are in every realization. There are also five points in the center of the grid of which exactly one is a maximum in any realization. Figure 6 (b) shows the patch graph which got simplified by our merging in (c). (a) shows the critical point regions after the merging step. The graph clearly shows what critical points can occur, what their probabilities are and how they are related to each other. While most of the points form single components due to them being part of every realization, the five points in the middle of the grid correctly get identified. The horizontal adjacencies between them allow for easy and overlap-free merging, resulting in the whole region getting identified as one critical point with spatial uncertainty. As the individual probabilities of the points are small and could be even made smaller by extending the grid, methods treating them separately will miss them. E.g., Günther et al. [GST14] would not identify these points as they clearly are not mandatory. Also, because Mihai and Westermann [MW14] filter single points by their confidence, the individual points can fall below the threshold while the combined one lies above it.

13.2 Weather forecast

The second dataset is temperature data from the European Center for Medium-Range Weather Forecasts (ECMWF). Not only do they provide measured temperature fields but also uncertain fields representing forecasts resulting from ensemble simulations. While the uncertain fields are given by a field for mean and one for standard deviation, the correlation matrix is missing. However, this matrix can be generated using different techniques. A common approach is to compute correlation based on an analytic function, e.g. the

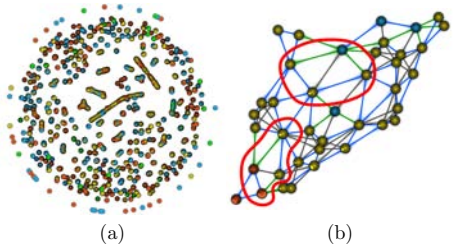


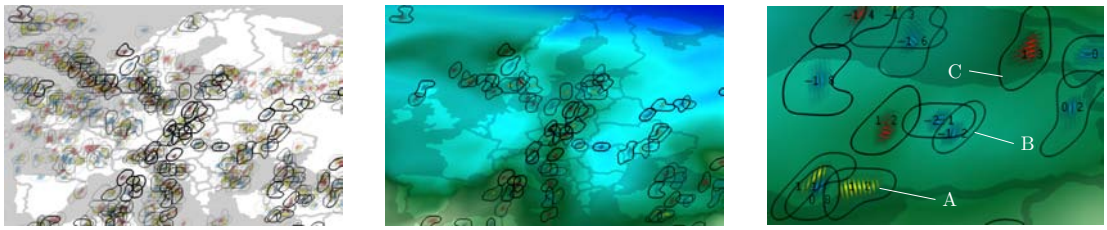
Figure 8: *The patch graph (a) of an uncertain temperature forecast with 2 eigenvalues chosen. Zooming to a connected component (b) allows for investigating patch adjacency and even higher level structures like folds (red).*

squared exponential [Pac03]. Since we have access to measured temperature data, we use this instead to generate the correlation matrix. For this, we use 500 measures, which were acquired over a period of more than 8 months. These measures are then used to extract the correlation between every pair of points. The basic assumption is that by using a large amount of fields, the extracted correlation only depends on geographic factors. Combining the correlation matrix with the standard deviation of the forecast to get a complete covariance matrix, we have all data to apply our method.

We use a 240 hour forecast above the region of Europe as uncertain field. The triangular grid consists of 14 400 points with a neighborhood size of 6 for interior points. Even though the uncompressed covariance matrix is as big as 1.6GB, using the SLEPc [HRV05] library we are able to quickly compute the most significant eigenvalues and eigenvectors. We only use the first 18 eigenvalues as they are enough to describe 95% of the data's variance. Our algorithm extracts 69 354 singular patches intersecting a threshold region with $\delta = 0.95$ (see Section 7.1). The adjacency computation results in 1 681 477 connections between patches. Merging nodes to extract critical points with spatial uncertainty as described in section 11 leads to a reduction to 22 819 critical points.

Computation Times – We did our computations on an Intel Xeon CPU E5-2630 with 32 virtual cores running at 2.4GHz. In Table 1, the computation times of the major steps in our pipeline are shown. For a better overview, we did multiple runs on subsets of the domain with different grid point counts. One can see that the merging algorithm takes the majority of the time as it requires a lot of emptiness checks. Also, while the parallelization of the computation of patches and patch adjacency is straightforward, the merging has a non-trivial parallelization, which also contributes to high computation times. As the methods mainly depend on the number of patches and patch adjacencies and therefore the output size, there is a non-linear behaviour regarding the number of grid points.

Sampling Precision – As the computation of the probabilities for all critical points relies on sampling, there always is an error involved. To get an idea about the magnitude of this error, we did multiple sampling runs with different sample counts. For every sample count and every patch, we computed the mean and standard deviation across 50 runs. We then averaged the relative standard deviation of all patches with a mean probability of at least 0.05. Figure 7 shows the results of this computation and therefore gives an estimate of the relative error introduced during sampling. One can see that the error decreases fairly quickly and even sample point counts of only 10 000 result in an average relative deviation of only 3%.



(a) (b) (c)

Figure 9: *Regions of merged critical points in a Gaussian-distributed scalar field of temperature data. In (a), an overview over all critical points is given. In (b), only points which are present in at least 50% of all realizations with a color mapped mean field in the background are shown. Zooming in gives a clearer look at the probability distributions inside the merged regions (c). Here, e.g., the critical points A and C have a higher spatial uncertainty than B as the texture is spread out more.*

Patch Graph – Even with the reduced amount of 22 819 nodes, the patch graph is too complex to look into manually. The reason for this is it being a structure representing 18-dimensional features embedded in a 2-dimensional visualization. For demonstration purposes and to get an idea about the structure of the patch graph, we show the patch graph resulting from using only the first 2 eigenvalues in Figure 8 (a). In (b), a typical connected component is shown. Besides the different forms of adjacency, folds can be identified (red border) as they are always represented by specific combinations of nodes and edges. We plan on studying the properties of the patch graph in lower dimensions to then use the insights gained to algorithmically operate on more complex graphs in the future.

Critical Points – With every merged node representing multiple patches that can belong to multiple grid points, we can draw them as regions as described in Section 12. The resulting image can be seen in Figure 9 (a). More probable points can be identified by a more prominent border. Alternatively, one can filter out critical points with low probability as shown in Figure 9 (b). Using only borders and transparent fillings allows for visualizing context information. In both (a) and (b) we added country borders to provide a geographic reference. Furthermore, (b) has the color coded mean field as background which gives an idea of the actual scalar values. A closeup look can be seen in Figure 9 (c). The probability distribution inside the regions is important to get a better understanding about the spatial uncertainty of the points. While the texture of (B) only is very local, the stripes of (A) and (C) are spread out more indicating a higher likelihood to be at different locations. Just like in our synthetic example, points like (A) and (C) might be missed by methods looking only at mandatory critical points or at local probability estimates. Only by deterministically extracting all possible points, merging, and accumulating the probability we are able to detect those features. Our sampling method not only gives us probability information but also allows for computing other measures. The numbers shown in Figure 9 (c) denote the mean scalar value the critical point has across all realizations it is present in. These values could be interesting, e.g., when it comes to topological simplification.

14 Limitations, and Future Work

Even though the theoretical complexity of the time-critical parts in our computation pipeline only depends on the number of grid points and the neighborhood sizes, there still are performance issues. The main reason for this is the large amount of linear and quadratic programs that have to get solved to check for patch emptiness and adjacency. In future work we want to improve our algorithms and replace important steps by combinatorial counterparts using insights gained from the patch graph. We also would like to get rid of the two parameters – namely the number of significant eigenvalues and the threshold radius – thus the need to compute the eigen decomposition at all. As our method depends on the neighborhood sizes it does not scale well with increasing dimension of the underlying domain. Despite our method being applicable in arbitrary dimensions, the fast increasing number of neighbors currently does not allow to use it with domain dimensions higher than two due to long computation times.

An even more important goal is to dive deeper into the structural properties of the patch graph. While our node merging is able to extract critical points that otherwise would be hard to identify, it requires explicit overlap tests and depends on the order of patch merging. Investigating higher-level features like folds can lead to simplification, abstract representation, and an overall better understanding of topology of uncertain scalar fields.

15 Conclusion

In this paper we presented a novel method to look at topological features of Gaussian-distributed scalar fields. Given the mean field and covariance matrix, we introduced singular patches as analogy for critical points taking into account uncertainty and correlation in the data. These patches can be used to not only extract critical points and compute their probability with high precision but also to get further insights into the structure and relation between them. By defining patch adjacency we can build the patch graph – an abstract structure containing much information about topological structures, such as critical points changing their location, birth and death of contour tree branches, and folds. Even though understanding the patch graph to its fullest requires further research, we emphasize the value by a simplification algorithm to extract and visualize some of the critical points with spatial uncertainty.

16 Acknowledgements

First, we want to thank the Deutsche Forschungsgemeinschaft for funding the project SCHE 663/11-1. We also would like to thank Dr. Christian Heine for proofreading and giving valuable criticism and advice that improved the quality and structure of the paper significantly.

References

[ATC*08] AU O. K.-C., TAI C.-L., CHU H.-K., COHEN-OR D., LEE T.-Y.: Skeleton extraction by mesh contraction. In *ACM Transactions on Graphics* (2008), vol. 27, p. 44. 1

[BOL12] BRODLIE K., OSORIO R. A., LOPES A.: A review of uncertainty in data visualization. In *Expanding the Frontiers of Visual Analytics and Visualization*. Springer, 2012, pp. 81–109. 1

[CLLR05] CHIANG Y.-J., LENZ T., LU X., ROTE G.: Simple and optimal output-sensitive construction of contour trees using monotone paths. *Computational Geometry 30*, 2 (2005), 165–195. 1

[CSA03] CARR H., SNOEYINK J., AXEN U.: Computing contour trees in all dimensions. *Computational Geometry 24*, 2 (2003), 75–94. 1

[EHP08] EDELSBRUNNER H., HARER J., PATEL A. K.: Reeb spaces of piecewise linear mappings. In *Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry* (2008), ACM, pp. 242–250. 3

[EM90] EDELSBRUNNER H., MÜCKE E. P.: Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics 9*, 1 (1990), 66–104. 4

[FGSW15] FISCHER K., GÄRTNER B., SCHÖNHERR S., WESSENDORP F.: Linear and quadratic programming solver. In *CGAL User and Reference Manual*, 4.7 ed. CGAL Editorial Board, 2015. 5

[GS06] GRIETHE H., SCHUMANN H.: The visualization of uncertain data: Methods and problems. In *SimVis* (2006), pp. 143–156. 1

[GST14] GÜNTHER D., SALMON J., TIERNY J.: Mandatory critical points of 2D uncertain scalar fields. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 31–40. 2, 8

[Hot33] HOTELLING H.: Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology 24* (1933), 417–441. 3

[HRV05] HERNANDEZ V., ROMAN J. E., VIDAL V.: SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Software 31*, 3 (2005), 351–362. 8

[Joh04] JOHNSON C.: Top scientific visualization research problems. *IEEE Computer Graphics and Applications 24*, 4 (2004), 13–17. 1

[JS03] JOHNSON C. R., SANDERSON A. R.: A next step: Visualizing errors and uncertainty. *IEEE Computer Graphics and Applications 23*, 5 (2003), 6–10. 1

[Kra10] KRAUS M.: Visualization of uncertain contour trees. In *IMAGAPP/IVAPP* (2010), Citeseer, pp. 132–139. 2

[Lib12] LIBGOBER J.: The Euler characteristic, Poincaré-Hopf theorem and applications. *Uchicago math REU* (2012), 17. 6

[LPK05] LOVE A. L., PANG A., KAO D. L.: Visualizing spatial multivalue data. *IEEE Computer Graphics and Applications 25*, 3 (2005), 69–79. 1

[Mor34] MORSE M.: *The calculus of variations in the large*, vol. 18. American Mathematical Soc., 1934. 1

[MRH*05] MACEACHREN A. M., ROBINSON A., HOPPER S., GARDNER S., MURRAY R., GAHEGAN M., HETZLER E.: Visualizing geospatial information uncertainty: What we know and what we need to know. *Cartography and Geographic Information Science 32*, 3 (2005), 139–160. 1

[MW14] MIHAI M., WESTERMANNR R.: Visualizing the stability of critical points in uncertain scalar fields. *Computers & Graphics 41* (2014), 13–25. 2, 8

[OGHT10] OTTO M., GERMER T., HEGE H.-C., THEISEL H.: Uncertain 2D vector field topology. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 347–356. 2

[Pac03] PACIOREK C. J.: *Nonstationary Gaussian processes for regression and spatial modelling*. PhD thesis, Citeseer, 2003. 8

[PCM02] PASCUCCI V., COLE-MCLAUGHLIN K.: Efficient computation of the topology of level sets. In *Proceedings of the Conference on Visualization '02* (2002), IEEE Computer Society, pp. 187–194. 1

[PGA13] POTTE K., GERBER S., ANDERSON E. W.: Visualization of uncertainty without a mean. *IEEE Computer Graphics and Applications 33*, 1 (2013), 75–79. 1

[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. 2

[PH13] PÖTHKOW K., HEGE H.-C.: Nonparametric models for uncertainty visualization. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 131–140. 1

[PKXJ12] POTTER K., KIRBY M., XIU D., JOHNSON C. R.: Interactive visualization of probability and cumulative density functions. *International journal for uncertainty quantification 2*, 4 (2012), 397–412. 1

[PPH12] PETZ C., PÖTHKOW K., HEGE H.-C.: Probabilistic local features in uncertain vector fields with spatial correlation. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 1045–1054. 2

[Pre07] PRESS W. H.: *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007. 5

[PRJ12] POTTER K., ROSEN P., JOHNSON C. R.: From quantification to visualization: A taxonomy of uncertainty visualization approaches. In *Uncertainty Quantification in Scientific Computing*. Springer, 2012, pp. 226–249. 1

[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

[PSBM07] PASCUCCI V., SCORZELLI G., BREMER P.-T., MASCARENHAS A.: Robust on-line computation of reeb graphs: simplicity and speed. In *ACM Transactions on Graphics* (2007), vol. 26, p. 58. 1

[PW12] PFAFFELMOSER T., WESTERMANN R.: Visualization of global correlation structures in uncertain 2d scalar fields. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 1025–1034. 1

[PW13] PFAFFELMOSER T., WESTERMANN R.: Correlation visualization for structural uncertainty analysis. *International Journal for Uncertainty Quantification 3*, 2 (2013), 171–186. 1

[PWH11] PÖTHKOW K., WEBER B., HEGE H.-C.: Probabilistic marching cubes. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 931–940. 2

[PWL97] PANG A. T., WITTENBRINK C. M., LODHA S. K.: Approaches to uncertainty visualization. *The Visual Computer 13*, 8 (1997), 370–390. 1

[Ree46] REEB G.: Sur les points singuliers d'une forme de Pfaff complétement intégrable ou d'une fonction numérique. *CR Acad. Sci. Paris 222* (1946), 847–849. 1

[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. 2

[WZ13] WU K., ZHANG S.: A contour tree based visualization for exploring data with uncertainty. *International Journal for Uncertainty Quantification 3*, 3 (2013), 203–223. 2