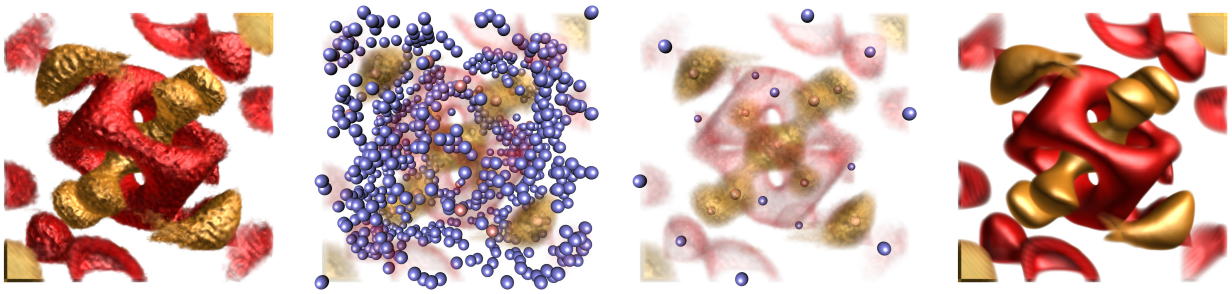
Fast and Memory-Efficient Topological Denoising of 2D and 3D Scalar Fields

David Günther, Alec Jacobson, Jan Reininghaus, Hans-Peter Seidel, Olga Sorkine-Hornung, Tino Weinkauf



(a) Noisy input scalar field. (b) All minima and maxima of the noisy input data. (c) Extrema above a certain noise level (persistence) are selected. (d) Filtered output scalar field contains only the selected extrema.

Figure 1. Our method filters scalar fields with explicit control over the removal and the preservation of minima (blue spheres) and maxima (red spheres). The result is smooth and topologically clean.

Abstract—Data acquisition, numerical inaccuracies, and sampling often introduce noise in measurements and simulations. Removing this noise is often necessary for efficient analysis and visualization of this data, yet many denoising techniques change the minima and maxima of a scalar field. For example, the extrema can appear or disappear, spatially move, and change their value. This can lead to wrong interpretations of the data, e.g., when the maximum temperature over an area is falsely reported being a few degrees cooler because the denoising method is unaware of these features. Recently, a topological denoising technique based on a global energy optimization was proposed, which allows the topology-controlled denoising of 2D scalar fields. While this method preserves the minima and maxima, it is constrained by the size of the data. We extend this work to large 2D data and medium-sized 3D data by introducing a novel domain decomposition approach. It allows processing small patches of the domain independently while still avoiding the introduction of new critical points. Furthermore, we propose an iterative refinement of the solution, which decreases the optimization energy compared to the previous approach and therefore gives smoother results that are closer to the input. We illustrate our technique on synthetic and real-world 2D and 3D data sets that highlight potential applications.

Index Terms—Numerical optimization, topology, scalar fields.

1 Introduction

Noise and sampling artifacts hinder the visual analysis of measurements and simulated data. For example, an isocontour visualization of a noisy scalar field contains a large number of small connected components which make it difficult to see the big picture. As another example, gradient estimation is often negatively affected by noise in a scalar field.

Many methods exist to smooth or denoise scalar fields. Some of them focus on statistical features in the data, e.g., a Gaussian blur or a median filter. Other methods aim to maintain spatial features while smoothing the data, e.g., a bilateral filter preserves edges in an image.

- *David Günther is with Institut Mines-Télécom, Télécom ParisTech, CNRS LTCI, Paris, France. E-mail: gunther@telecom-paristech.fr.*
- *Alec Jacobson is with Columbia University, New York, USA. E-mail: jacobson@cs.columbia.edu.*
- *Jan Reininghaus is with IST Austria, Vienna, Austria. E-mail: jan.reininghaus@ist.ac.at.*
- *Olga Sorkine-Hornung is with ETH Zürich, Zürich, Switzerland. E-mail: sorkine@inf.ethz.ch.*
- *Tino Weinkauf and Hans-Peter Seidel are with Max Planck Institute for Informatics, Saarbrücken, Germany. E-mail: {weinkauf,hpseidel}@mpi-inf.mpg.de.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014; date of publication xx xxx 2014; date of current version xx xxx 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

The method presented in this paper falls into the category of denoising methods that maintain spatial features. In particular, we focus on topological features: the minima and maxima of a scalar field. The input of our method is a scalar field and a subset of its minima and maxima. The output is a smoothed version of the scalar field that contains only the selected minima/maxima, and is otherwise as close as possible to the original data. The values and positions of the selected extrema are preserved, while all other extrema are removed from the data.

Such a filter provides control over the topology, which can be beneficial for subsequent visualization methods. For example, the appearance of connected components in an isocontour visualization is a matter of topology: removing the noise-induced extrema (e.g., identified using persistence [7]) leads to fewer and larger connected components, which provides an isocontour visualization with less clutter.

We extend previous work [15, 24] on topological denoising filters. Our method follows the same computation pipeline, which has two main stages. First, the extrema of a scalar field are extracted and filtered (Section 3). Second, the denoised scalar field is obtained as the solution of a discrete optimization problem – we call this "numerical reconstruction" (Section 4). The improvements over the previous work are due to the following contributions:

- For the extraction and filtering part of the pipeline, we propose a direct coupling of Forman's discrete Morse theory [8] to our numerical optimization scheme, where the output of the former serves directly as the input to the latter. In contrast to [15, 24], this eliminates the need to remesh the domain and solve another optimization problem to get a representative function. (Section 3.2.1)

- We propose a topological simplification scheme that allows user intervention. Compared to previous work, this provides more flexible control over the topology. (Section 3.2.2)
- For the numerical reconstruction, we propose a scheme to iteratively improve the quality of the solution. The result is notably smoother and closer to the original data than in previous work. (Section 4.2)
- For the numerical reconstruction, we propose a novel domain decomposition, which leads to a significant speed-up (6 minutes vs. previously 6.5 hours for 64^3) and a significant reduction of the memory requirements (3 GB vs. previously 10 GB for 64^3). With this contribution, our method achieves, in contrast to previous work, practicable computation times for 3D scalar fields. (Section 4.3)

We extensively evaluate and discuss our method in Section 5, show applications in Section 6, and conclude with a discussion of possible future research directions in Section 7.

2 Related Work

Methods for smoothing scalar data while preserving its salient features are sought after in many domains. In image processing, typical features are intensity edges; methods such as bilateral filtering [22] or non-local means [3] attempt to denoise images without blurring strong edges. In signal and geometry processing, Laplacian smoothing techniques were adapted to interpolate the values at some prescribed points [19]. Such methods cannot guarantee exact preservation of extrema, and new (unwanted) extrema can even emerge in the output [9]. The technique in [9] tracks a given 2D filter, e.g. Laplacian or anisotropic diffusion, and stops it before unwanted changes of the isocontour topology occur.

Scalar field topology is related to the more general notion of vector field topology. An approach to a continuous topology simplification of 2D vector fields is presented by Tricoche et al. [23]. Based on the topological graph structure and certain relevance measures, pairs of critical points are removed by local changes to the vector values at the grid nodes. The method does not guarantee successful removal of all critical points that are scheduled for removal, and smoothness criteria of the result are not addressed. Theisel [20] and Weinkauf et al. [26] construct a 2D or 3D vector field based on a given topological skeleton. Such methods create piecewise-linear vector fields with C^0-continuity and cannot avoid the appearance of additional critical points.

Topology design and control is also of high interest for functions on surfaces. Recently, Tierny et al. [21] proposed an interactive combinatorial algorithm to edit a scalar field on a surface with a user-prescribed topology. This 2D approach uses an iterative heuristic, but shows very good practical performance. However, its extension to 3D is an open problem. Chen et al. [6] use Morse decomposition to edit the topology of vector fields on surfaces. Harmonic functions have been used for constructing Morse functions on surfaces [17]. Topological control is beneficial for binary image segmentation as well [5].

A number of methods exist that exploit the topological simplification of the Morse-Smale complex to simplify 2D scalar fields. Bremer et al. [2] smooth the function in the interior of a Morse-Smale cell after each cancellation step to adhere to the altered topological structure. The resulting scalar field is C^0-continuous between Morse-Smale cells. Alternative approaches for 2D scalar fields are given by Weinkauf et al. [24] and Jacobson et al. [15], where the scalar field is reconstructed from a given subset of the original topology. Both methods employ optimization to construct a 2D scalar field that conforms to the prescribed topology, and is also smooth and close to the input data. It has also been shown in this context that explicit control over the topology of a scalar field has interesting applications for animating characters [14,15] – this requires the topological design of very small 3D scalar fields.

The work presented in this paper is an extension of [15,24] to larger data sets. The extension is non-trivial, since directly applying the methods of [15,24] to 3D data leads to impractical performance, including excessive memory use and computation times in the order of several hours (6.5 hours for 64^3). Our solution is a novel domain decomposition approach. While standard decomposition techniques may introduce spurious extrema, we propose an iterative processing of the decomposed blocks which communicates the information in-between blocks. This results in a smooth output, avoids spurious extrema, and guarantees that only the user-selected extrema are present. This divide and conquer approach leads to significantly faster computation times in the order of a few minutes (6 minutes for 64^3), and also opens the door to a parallel and distributed computation — decreasing the computation times even further. Additionally, we provide an iterative scheme to drastically improve the quality of the solution, and a direct coupling between the topology extraction and the numerical optimization.

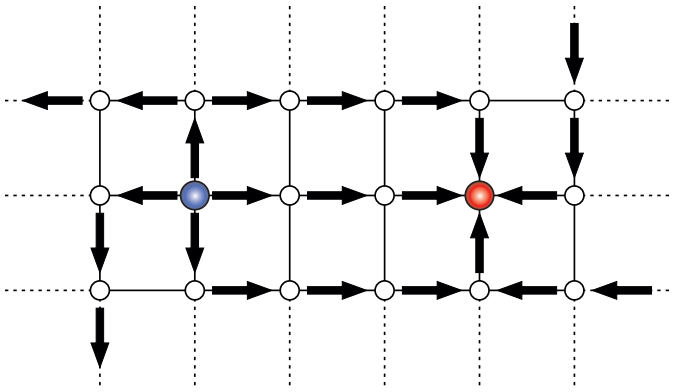


Figure 2. The directed acyclic graph $\mathscr{G}$ encodes the monotonicity on the input grid with respect to the selected extrema $K_{min} \cup K_{max}$. It is shown here using arrows pointing into the direction of a larger neighbor. All neighbors of a minimum (blue) are larger than the minimum itself. For a maximum (red), all neighbors are smaller. All other vertices (white) are described as non-extremal points by having at least one larger and one smaller neighbor.

3 Extrema Extraction and Filtering

This section discusses the extraction and filtering of extrema in a 2D or 3D scalar field. In Section 4, we will feed the filtered extrema to the numerical reconstruction, which generates a smooth scalar field containing only these extrema and being otherwise as close as possible to the input scalar field.

Let G be a uniform grid in $\mathbb{R}^{2,3}$ with vertices $\mathbf{v}_i \in V$ and let $\widehat{s}$ be a scalar field defined on that grid. Let $\widehat{K}_{min} \subset V$ and $\widehat{K}_{max} \subset V$ denote the minima and maxima of $\widehat{s}$, respectively.[1] Our goal is now to extract these extrema and allow the user to select some of them, i.e., we have a subset of minima $K_{min} \subseteq \widehat{K}_{min}$ and a subset of maxima $K_{max} \subseteq \widehat{K}_{max}$. We often refer to $K_{min} \cup K_{max}$ as *selected extrema*.

The second input to the numerical reconstruction is the *monotonicity graph* $\mathscr{G} = (V, \mathscr{E})$. It is a connected, directed acyclic graph on the input grid G, where V denotes the vertices of G and $\mathscr{E}$ is a set of directed edges between neighboring vertices. Figure 2 gives a 2D illustration. The edge-set $\mathscr{E}$ includes all edges emanating from K_{max}, all edges incident on K_{min}, and at least one edge pointing in and one edge pointing out of all other vertices. Loosely speaking, $\mathscr{G}$ represents gradient information — it points into the direction of a larger neighbor. It describes that all neighbors of a minimum/maximum are larger/smaller than the extremum itself. Most importantly, it describes that any other vertex cannot be an extremum, since it has at least one incoming and one outgoing edge.

In general, a given set of extrema can be represented by many possible graphs $\mathscr{G}$. We will propose different schemes for constructing valid monotonicity graphs in the following. A comparison of different monotonicity graphs is given in Section 5.3.

3.1 Approaches without Morse Theory

The extraction and filtering methods in this section are easy to implement but the resulting monotonicity graph is *unaware* of the input scalar

[1] Note that a trilinearly interpolated function attains its minima and maxima always at the vertices of the grid. Hence, it is reasonable to define the extrema as a subset of the vertices.

field. Consequently, a subsequent numerical reconstruction requires usually more iterations to converge compared to the methods from Section 3.2.

3.1.1 Non-Topological Extraction and Filtering

A simple algorithm for extracting the extrema of a scalar field is this: Visit each vertex $\mathbf{v}_i$. If all of its neighbors are larger/smaller, then $\mathbf{v}_i$ is a minimum/maximum. One may now use any selection criteria to define a set of selected extrema.

Now we construct a cheap-to-compute, yet unsmooth and data-unaware scalar field s_a that contains only the selected extrema. This corresponds to the *representative function* in [15]. We solve the following Laplace problem:

$$Ls_a = 0 \tag{1}$$

$$\text{s.t. } s_a(\mathbf{v}_i) = 0 \qquad \forall \mathbf{v}_i \in K_{min} \tag{2}$$

$$s_a(\mathbf{v}_i) = 1 \qquad \forall \mathbf{v}_i \in K_{max} \tag{3}$$

where L denotes the finite-difference discretization of the Laplace operator on regular grids. The result s_a is a harmonic function, for which the maximum principle of discrete harmonic functions guarantees that, when choosing the Dirichlet boundary conditions as above, the locations in K_{min} and K_{max} become the only minima and maxima, respectively.

As proposed in [15], a monotonicity graph is built from s_a by constraining all edges around an extremum as well as the two edges around every other vertex corresponding to the steepest ascent and descent.

3.1.2 Persistence Pairs

Persistence [7] provides an alternative way for selecting the extrema of a scalar field. The algorithm tracks the topological changes in the evolution of the sublevel sets in a scalar field, amongst which we find the minima and maxima of the scalar field. Most importantly, persistence provides an "importance" for each extremum. Noise-induced extrema have a low persistence while dominant ones have a high persistence. This is very useful for filtering extrema. Fast algorithms [12] and open-source implementations[2] [1] for computing persistence are available.

A monotonicity graph can be computed from the selected extrema using a representative function as described above.

3.2 Approaches based on Morse Theory

The extraction and filtering methods in this section are more involved. In contrast to above, the monotonicity graph is initially constructed from the input scalar field and then carefully modified in combination with the extrema filtering. Hence, the monotonicity graph is *data-aware* and a subsequent numerical reconstruction usually requires less iterations to converge compared to the methods from Section 3.1.

3.2.1 Classic Topological Simplification

In the following, we recapitulate the main idea of [15, 24] before we present an algorithmic extension yielding a more direct and simpler optimization strategy in which a representative function is not needed.

In [15, 24], the Morse-Smale complex of the input scalar field is computed based on discrete Morse theory [8]. The complex consists of the critical points (minima, maxima, saddles) and the separatrices connecting the critical points. Filtering is done by means of topological simplification in which extremum-saddle pairs are repeatedly removed from the Morse-Smale complex (subject to certain conditions). In [15, 24], the simplified Morse-Smale complex is used to construct a data-aware representative function using an algorithm that requires an explict remeshing of the domain and solving optimization problems in a pre-processing step.

In this work, we propose a simpler approach which provides a *direct coupling* between Forman's discrete Morse theory and our numerical reconstruction presented in Section 4.

We exploit the fact that discrete Morse theory allows us to encode the Morse-Smale complex in a so-called *discrete gradient field*. It encodes the monotonicity of the scalar field — very much like our monotonicity graph $\mathscr{G}$. Furthermore, the topological simplification of the Morse-Smale complex can be done by working directly on the discrete gradient. For more details and schematic illustrations regarding the simplification process in discrete Morse theory, we refer to [11]. The only caveat is that the discrete gradient is defined on the cell complex of the domain, whereas we require a vertex-based representation of the monotonicity graph $\mathscr{G}$. We circumvent this issue by doing all topological computations, including the discrete gradient and the Morse-Smale complex, on an auxiliary grid with a halved resolution. More precisely, the resolution for each dimension is $1+(N-1)/2$, where N refers to the resolution of the input grid in that dimension.[3] The cell complex of the auxiliary grid has a 1:1 correspondence to the vertices of the input grid, i.e., the discrete gradient can be directly used as the monotonicity graph. A representative function is *not* required.

3.2.2 Extrema Cancellation with User Control

The order of a classic topological simplification is usually determined by an importance measure for critical points such as persistence, i.e., the method is completely automated.

In the following, we introduce an algorithm for removing a user-selected set of extrema K_u from the Morse-Smale complex and the corresponding discrete gradient $\mathbf{g}$. This algorithm works in 2D and 3D, and can be run standalone or after a classic topological simplification as described above.

Background. Let p denote a path[4] in a discrete gradient $\mathbf{g}$. If two critical points $\mathbf{a}$ and $\mathbf{b}$ are connected by one and only one path p, then we call p a cancellation path. As given by Forman [8], reversing the flow direction along p creates a new discrete gradient $\mathbf{g}'$ where $\mathbf{a}$ and $\mathbf{b}$ are no longer critical.

Algorithm. First, we create a priority queue into which we insert saddles as follows. For each saddle $\mathbf{s}$, find the extremum $\mathbf{e} \in K_u$ to which $\mathbf{s}$ is connected by a cancellation path and to which it has the smallest height difference in the input scalar field $|\widehat{s}(\mathbf{s}) - \widehat{s}(\mathbf{e})|$. If $\mathbf{e}$ exists, push $\mathbf{s}$ into the priority queue according to the value of the height difference.

We process the queue as follows. Pop the top element $\mathbf{s}$ off the queue. Again, find the extremum $\mathbf{e} \in K_u$ with the smallest height difference that is connected to $\mathbf{s}$ by a cancellation path p. If such a cancellation path does not exist anymore, then ignore the saddle and proceed with the queue. Otherwise, compare the new height difference to the height difference of the *next* saddle in the queue. If it is larger, then reinsert $\mathbf{s}$ into the priority queue with the new value. If it is smaller, then do the actual cancellation: reverse the flow in $\mathbf{g}$ along p and remove the extremum from K_u. Proceed with the queue.

With respect to the number of critical points, the time complexity of this algorithm is cubic in the worst case, but linear in practice. The memory complexity is always linear. The monotonicity graph is obtained from the simplified $\mathbf{g}$ as with classic topological simplification.

4 Numerical Reconstruction

Equipped with a set of selected extrema and a corresponding monotonicity graph, we come now to the main part of our method: the reconstruction of a smooth scalar field containing only the selected extrema and following the prescribed monotonicity. This works for both 2D and 3D scalar fields.

We begin with a recapitulation of the optimization problem introduced in [15]. Given the mathematical formulation, we propose an iterative reconstruction scheme which converges to a smooth scalar field and minimizes the distance to the input field. In the last part of this section, we present our new domain decomposition approach allowing an efficient solving of the optimization problem.

[2] DIPHA: `http://dipha.googlecode.com/`

[3] Note that this procedure neglects extrema whose Morse cell is smaller than a 1-ring in the input grid. This is not an issue in our target applications, since we want to remove small features anyway. An alternative is to double the resolution of the input grid and supersample the data first.

[4] This concept is similar to an integral curve in a smooth gradient field.

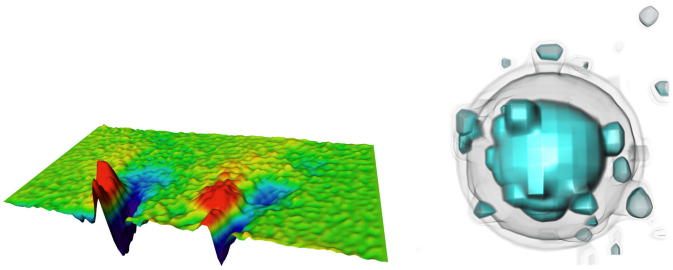


Figure 3. Explanatory data sets. (left) 2D vorticity data set from [15, 24] shown as terrain. (right) 3D spherical function distorted by very strong salt & pepper noise.

For explanations, we use a 3D data set with a spherical function distorted by salt & pepper noise (Figure 3, right). The range of the noise *exceeds* the range of the spherical function by a factor of two. This is a demanding scenario where a simple smoothing filter is not able to remove the noise, but our topology-based approach is able to do so. Filtering is straightforward: we select one single maximum in the center of the data set (results are shown in Figure 7a). In 2D, we use the same vorticity data set that has been used in [15, 24] (Figure 3, left). Extrema are filtered here like in the previous work using a persistence threshold of 18.6 (results are shown in Figure 5).

4.1 Problem Setup and Modeling

Let G be a uniform grid in $\mathbb{R}^{2,3}$ with vertices $\mathbf{v}_i \in V$ and let $\widehat{s}$ be the original scalar field defined on that grid. Let $K_{min} \cup K_{max} \subset V$ be the selected extrema of $\widehat{s}$. Our goal is to find a new scalar field s with the following properties:

I: The selected extrema appear as extrema in s of the same type.

II: s has no other extrema.

III: s interpolates the original scalar values at the selected extrema.

IV: s approximates the original scalar field at all other vertices.

V: s minimizes a smoothness energy.

We assume that the set of selected extrema does not contradict the Morse inequalities. For example, a non-constant function on G must have at least one minimum. We also assume that at least one selected minimum is smaller than all selected maxima and vice-versa.

Many functions exist that fulfill the topological requirements I & II, i.e., they have exactly the same set of extrema. This is still true when requiring specific values for these extrema (requirement III). However, the requirements IV & V provide a way of measuring the suitability of such functions by means of an energy

$$E(s) = w_D E_D(s) + E_L(s) = w_D \sum \|s(\mathbf{v}_i) - \widehat{s}(\mathbf{v}_i)\|^2 \; + \; \|Ls\|^2 \quad (4)$$

with a vertex-wise least-squares energy for the data term E_D (requirement IV), a Laplacian energy E_L for the smoothness term (requirement V), and the factor w_D for balancing their weight. In contrast to previous works [15, 24], which use finite elements on irregular triangle/tetrahedral meshes, we work on a regular grid. Hence, we employ a finite-difference discretization of the bi-Laplacian operator $L^\intercal L$.

The requirements I-V translate to the "ideal" discrete optimization problem [15] as follows:

$$\text{IV \& V:} \quad \arg\min_s \; E(s) \quad (5)$$

subject to the constraints

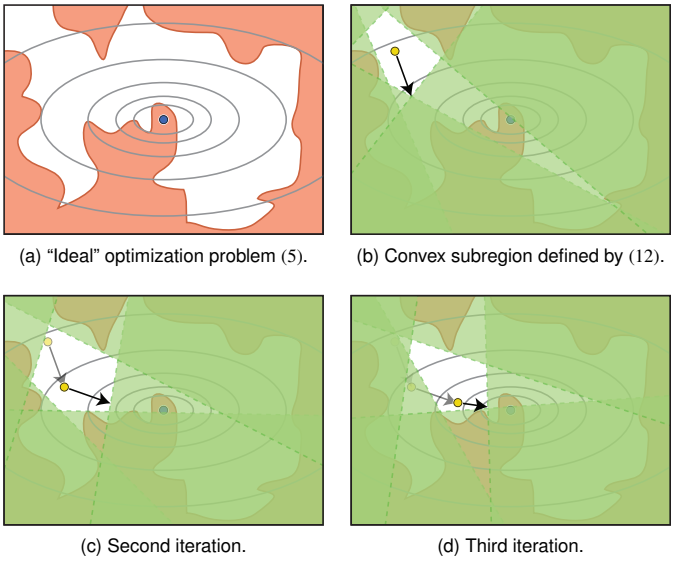
$$\text{I:} \quad s(\mathbf{v}_j) > s(\mathbf{v}_i) \qquad \forall \mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i), \forall \mathbf{v}_i \in K_{min} \quad (6)$$

$$\text{I:} \quad s(\mathbf{v}_j) < s(\mathbf{v}_i) \qquad \forall \mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i), \forall \mathbf{v}_i \in K_{max} \quad (7)$$

$$\text{II:} \quad s(\mathbf{v}_i) > \min_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i)} s(\mathbf{v}_j) \qquad \forall \mathbf{v}_i \notin K_{min} \cup K_{max} \quad (8)$$

$$\text{II:} \quad s(\mathbf{v}_i) < \max_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i)} s(\mathbf{v}_j) \qquad \forall \mathbf{v}_i \notin K_{min} \cup K_{max} \quad (9)$$

$$\text{III:} \quad s(\mathbf{v}_i) = \widehat{s}(\mathbf{v}_i) \qquad \forall \mathbf{v}_i \in K_{min} \cup K_{max} \quad (10)$$



(a) "Ideal" optimization problem (5). (b) Convex subregion defined by (12). (c) Second iteration. (d) Third iteration.

Figure 4. Illustration of the non-linear optimization problem (5) and our scheme to iteratively decrease the energy of the solution by repeated convexification of the feasible region using the linear constraints (12). See the text for a detailed description.

where we denote the grid neighbors of vertex $\mathbf{v}_i$ with the set $\mathcal{N}(\mathbf{v}_i)$. The constraints (8) & (9) are non-linear inequality constraints. They are non differentiable, and they describe a generally non-convex feasible region. Solving with such constraints directly leads to impracticable running times.

Instead, we use the monotonicity graph $\mathcal{G} = (V, \mathcal{E})$ to conservatively linearize these constraints. It enforces that all neighbors of a minimum/maximum are larger/smaller than the extremum itself. Any other vertex cannot become an extremum since it has at least one incoming and one outgoing edge. It allows us to *convexify* the feasible region [15]:

$$\text{IV \& V:} \quad \arg\min_s \; E(s) \quad (11)$$

subject to the constraints

$$\text{I \& II:} \quad s(\mathbf{v}_i) > s(\mathbf{v}_j) \qquad \forall (\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{E} \quad (12)$$

$$\text{III:} \quad s(\mathbf{v}_i) = \widehat{s}(\mathbf{v}_i) \qquad \forall \mathbf{v}_i \in K_{min} \cup K_{max} \quad (13)$$

In other words, the linear constraints (12) define a convex subspace of the larger feasible region described by the topological requirements I & II. The resulting optimization problem is a quadratic program which can be efficiently optimized via conversion to a conic program (see [15] for details). We solve it using the software package MOSEK [16].

4.2 Iterative Convexification

In the following, we explore the relationship between the "ideal", non-linear optimization problem (5) and the convex, linear optimization problem (11). Based on our observations, we propose a scheme to iteratively decrease the energy of the solution while remaining feasible.

Figure 4a is a 2D illustration of the high-dimensional, non-linear optimization problem (5). The energy is symbolized by concentric, elliptic isolines with a global energy minimum depicted by the blue dot. The "ideal" constraints (6)-(10) partition the domain into a *feasible region* where they are fulfilled (white), and an *infeasible region* where they are not fulfilled (red).[5]

Figure 4b illustrates the linear optimization problem (11). The linear inequalities (12) define overlapping halfspaces (shown with green overlays). They form a convex subset of the feasible region, visible

[5] Note that the depictions in Figure 4 are artistic interpretations. In general, the feasible region is nearly impossible to chart. It is generally not convex and perhaps even high-genus.

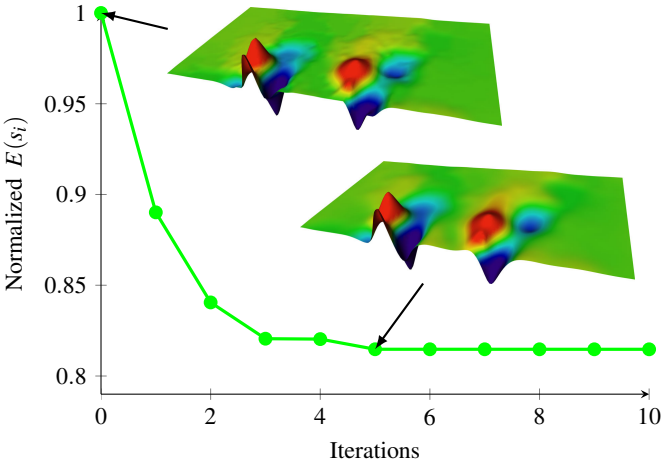


Figure 5. We can significantly decrease the energy of the solution by iteratively convexifying the feasible region. The upper inset corresponds to the result from [15]. We can significantly improve this result with our iterative scheme as shown in the lower inset. Compare also to the input data set from Figure 3.

as the white region not covered by the green overlays. Note that the unique minimum of the energy *within this convex subset* can always be found using quadratic or conic programming [18].

Previous methods [15, 24] stop here. The reconstructed scalar field s contains only the selected extrema but the energy $E(s)$ may still be high. We propose the following scheme to further decrease the energy:

1. We construct a new monotonicity graph $\mathscr{G}_{k+1}$ from the previous optimization result s_k as follows:
   - Around the selected extrema, it contains the same edges as $\mathscr{G}_k$.
   - For every other vertex $\mathbf{v}_i$, we define two directed edges $(\mathbf{v}_i^{\min}, \mathbf{v}_i)$ and $(\mathbf{v}_i, \mathbf{v}_i^{\max})$ where $\mathbf{v}_i^{\min}, \mathbf{v}_i^{\max}$ refer to the smallest and largest neighbor of $\mathbf{v}_i$ in s_k.
2. We solve the optimization problem (11) using the new monotonicity graph $\mathscr{G}_{k+1}$, which defines another convex subset of the feasible region.

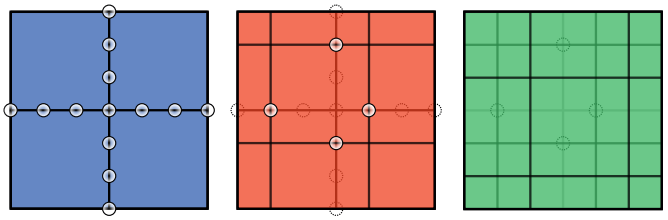
We may repeat this process until convergence, which is guaranteed because the energy of our iterative solutions $E(s_k)$ is monotonically decreasing. This can be seen as follows. Since the constraints are constructed according to s_k, we know that the feasible subset they describe is nonempty: it contains at least s_k. Optimizing (11) for the next solution s_{k+1} guarantees that $E(s_{k+1}) \leq E(s_k)$, since s_{k+1} is the unique global minimum in the feasible subset. Figures 4c-d show this.

However, we are not guaranteed to find the feasible global minimum or even a feasible local minimum. Our heuristic, in general, will be an approximation of the complex, non-convex feasible region. Still, this moving convex window will always reduce the energy or keep it on the same level, but never increase it. In fact, we observed for all our examples enormous energy reductions in the first few iterations, with diminishing returns afterwards. This makes it superior to running just one convexification and solving one optimization. Figure 5 plots the energy reductions and compares the result from [15] without iteration to our new result with iteration. A similar plot is shown for a 3D data set in Figure 7b.

Our process of iteratively redefining the convex feasible set and solving QPs is related to Sequential Quadratic Programming (SQP). However, generic SQP assumes all nonlinear inequality constraints are at least twice continuously differentiable [18].

4.3 Domain Decomposition

We propose an approach to a domain decomposition of the optimization problem defined in equation (11). It leads to substantially faster computation times and requires significantly less memory – even when executed on a single thread. This approach is the key to handling 3D scalar fields with a practicable performance. It also speeds up the optimization in the 2D case compared to the state of the art. Furthermore, the domain decomposition can be used to parallelize and even distribute the computations, which leads to even faster computation times.



(a) Initial decomposition into blocks of maximal size $n \times n$. The values at the inner boundaries are fixed.

(b) Shift of the decomposition by $n/3$, which frees most formerly fixed vertices, except for the few highlighted.

(c) After a final shift of the decomposition by $n/3$, all vertices have been free at least once.

Figure 6. Scheme of the repeated domain decomposition for a 2D scalar field. In 3D, we need four decompositions shifted by $n/4$.

Consider a domain decomposition of the underlying grid G into blocks of maximal size n^3 (or n^2 in 2D). Such a decomposition creates a conflict with the monotonicity constraints (12). These constraints form chains along which the function is monotonically increasing. The decomposition interrupts these chains, which means that a naive independent optimization of each block may introduce unwanted extrema. It is a classic dependency problem: when optimizing a block B, we need to know the values of the neighboring blocks at the boundary in order to correctly enforce the monotonicity constraints (12). On the other hand, the neighboring blocks need the same input from the block B as well. Introducing "thick" boundaries between blocks is not a solution since they still interrupt the monotonicity chains.

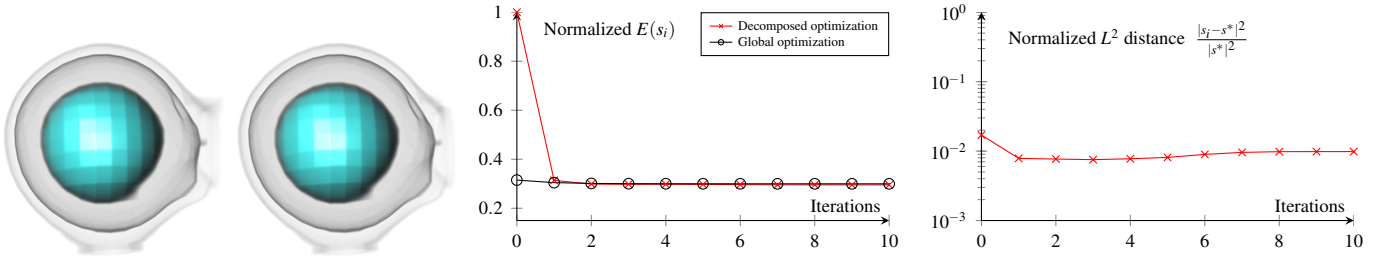
Our solution is twofold by (i) fixing appropriate values at the block boundaries, and (ii) shifting the block boundaries in a repeated process to ensure that every vertex is at least once subject to the optimization.

First, we decouple each block from the rest of the domain by fixing the values at its boundary using a cheap-to-compute approximation s_a of the solution, which conforms to the monotonicity constraints (12), but may exhibit a high energy, i.e., it may not be smooth or close to the input data. We only require s_a to be consistent with (12). We propose two versions:

- Based on a given monotonicity graph $\mathscr{G}$, we minimize the data energy E_D subject to the constraints (12) and (13). This is significantly faster and more memory-efficient than the actual optimization problem (11) since E_D exhibits maximal sparsity.
- We construct s_a using the method described in Section 3.1.1, where we solve a discrete Laplacian system with the selected extrema as Dirichlet boundary. From that we can easily derive a monotonicity graph $\mathscr{G}$, see Section 3.1.1 for details.

In either case, we have a feasible approximation s_a and a corresponding monotonicity graph $\mathscr{G}$. Each block is now decoupled from the rest of the domain by fixing its boundary to the values of s_a, while the inner vertices of the block are free. The blocks can now be optimized independently. Figure 6a illustrates this for a 2D example.

Every vertex needs to be subjected to the optimization, otherwise the resulting scalar field is not smooth at the fixed vertices. For a 3D data set, we *shift* the domain decomposition by a fourth of the block size into all three directions. For a 2D data set, we shift it by a third of the block size, as illustrated in Figure 6b. This gives us different blocks and most (but not all) of the former boundary vertices are now free. The new boundary vertices are fixed to the values from the previous optimization run. Now, we perform a second optimization for each new block independently. In 3D, we need to shift the decomposition two more times to make sure that every vertex is free during at least one of the four optimization runs. In 2D, we only need to shift twice and run three optimizations in total, see Figure 6c.



(a) The converged results of the global (left) and the decomposed (right) optimization are visually identical.

(b) Energy levels of the iterations of the global and the decomposed optimization.

(c) Normalized L^2 distance between the converged result s^* of the global optimization and different iterations s_i of the decomposed optimization.

Figure 7. Comparison of the decomposed optimization to the global optimization. The low normalized L^2 distance (around 10^{-2}) proves that the results are very similar, which can also be observed in the snapshots. The high energy in the first iteration of the decomposition scheme is due to the rough approximation s_a, but this is quickly rectified in the following iteration, where both energies match up. For all practical purposes, both schemes converged with the second iteration.

Our *decomposed optimization* scheme benefits from the behavior described in Section 4.2, namely that repeated optimizations decrease the energy. Furthermore, one may apply the iterative convexification as an outer loop to the decomposed optimization to decrease the energy even further.

Rationale. Theoretically, quadratic programming complexity scales superlinearly with respect to the number of variables, thus solving the individual quadratic programs for each block will be asymptotically faster than solving the global system – even when executed sequentially. Practically, the observed computation times for the global optimization show a quadratic behavior with respect to the number of variables (see Figure 8b in the next section), which makes the domain decomposition scheme even more beneficial.

The decomposition into individual blocks can be interpreted as new constraints restricting the convex feasible set introduced in Section 4.2 even further. Applying the iterative convexification of Section 4.2 will reduce the energy (4) until convergence. However, it is not guaranteed that the global and the decomposed optimization end in the same local energetic minimum because of the highly non-linear character of the underlying energy landscape. In practice, however, the solutions of the global and decomposed optimization converge empirically to the same scalar field, and we see enormous benefits both in terms of final energy and performance (see next section).

Block Size. A strategy for choosing the block size n has to take into account that, generally speaking, smaller n lead to faster computation times and lower memory consumption, while larger n lead to lower energies of the solution. Based on the performance figures discussed in the next section, it is straightforward to decide for a block size on a given machine.

Number of Blocks. Let the grid G be of size N^3 and the blocks of size n^3. Assume that $d = N/n$ is an integer. It is easy to see that the initial domain decomposition consists of d^3 blocks. Shifting the domain decomposition introduces $3d^2 + 3d + 1$ additional blocks. Example: a 64^3 grid is initially decomposed into 64 blocks of size 16^3, i.e., $d = 4$. Shifting the decomposition creates 61 additional (smaller) blocks, making for a total of 125 blocks.

5 Evaluation and Discussion

In the following, we evaluate and discuss our method. We start with an evaluation of our main contribution, followed by a demonstration of the whole method using a simple data set, which serves to highlight the characteristics of the method. We end the section with a discussion of the role of the monotonicity graph.

Unless stated otherwise, all results have been computed on a laptop with an Intel Xeon E31225 (3.1GHz) CPU and 16 GB RAM.

5.1 Evaluation of the Domain Decomposition

In the following, we evaluate the decomposed optimization scheme regarding its performance and its convergence to the global optimization.

	Block size	256^2 GB	256^2 minutes	512^2 GB	512^2 minutes	1024^2 GB	1024^2 minutes
global		0.5	1.5	1.7	11.9	7.4	120
domain decomposition	64^2	0.5	1.9	0.5	6.6	0.5	24.3
	128^2	0.5	3.3	0.5	9.9	0.5	33.9

Table 1. Measured performances of the global and the domain decomposition approach for 2D data. The latter can benefit even further from parallel execution. Computation times measured using a single thread.

	Block size	64^3 GB	64^3 minutes	128^3 GB	128^3 minutes
global		10	390	n/a	n/a
domain decomposition	8^3	0.4	36	0.4	251
	16^3	0.5	35	0.5	223
	24^3	0.6	168	0.6	921

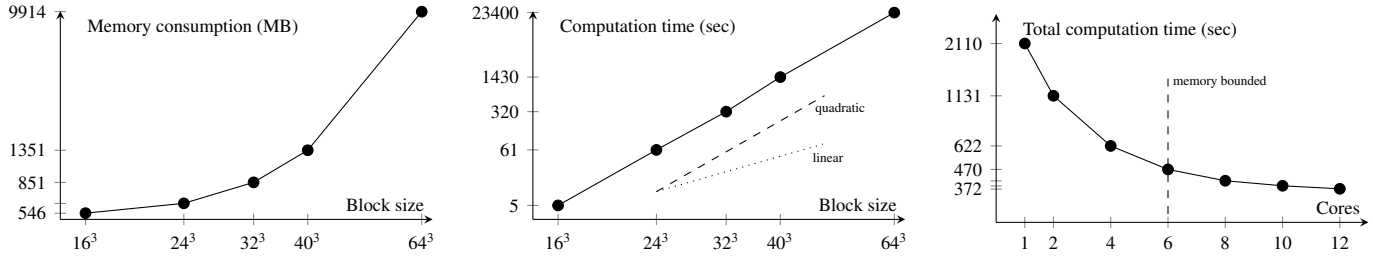
Table 2. Measured performances of the global and the domain decomposition approach for 3D data. The latter can benefit even further from parallel execution. Computation times measured using a single thread.

Convergence. Figure 7 compares the results of both schemes against each other for the spherical salt & pepper data set (compare to Figure 3). First, it has to be noted that the strong salt & pepper distortions could successfully be removed. Most importantly, the global scheme and the decomposition scheme yield visually identical results, which is supported by the very low normalized L^2 distance. We observed the same convergence for other data sets, too.

Note how the two different computation schemes start their iterations with very different energies. This is due to the approximation s_a that we require for the decomposed optimization. Here, we used the version where we minimize the data energy E_D as described earlier. While s_a fulfills all monotonicity constraints, it is not smooth, which leads to the very high energy in the first decomposed iteration. However, the global and decomposed optimizations converged to visually the same result with the second iteration. We observed similar behavior in all our experiments. The convergence evaluation for the 2D vorticity data set can be found in the supplemental material.

Performance. Figures 8a-b reveal why directly applying the method of [15] to 3D data sets is impracticable. Here we see for the global optimization that the memory consumption increases exponentially with the number of free vertices, and the computation time increases quadratically (note that both plots have log-log axes). For these plots, we computed a 64^3 data set globally: one iteration computes for 6.5 hours requiring almost 10 GB of main memory.

The decomposition approach is significantly faster and more memory efficient. One iteration for the same 64^3 data set computes for 6 minutes with a block size of 16^3 on 6 threads, thereby requiring 3 GB of main



(a) Memory consumption for different block sizes. The curve indicates an exponential memory consumption if the block size increases.

(b) Computation times for different block sizes. The dotted and dashed lines show reference slopes for linear and quadratic runtime behavior, respectively.

(c) Parallel performance on a shared-memory machine for a 64^3 data set with blocks of size 16^3.

Figure 8. Memory consumption and computation times for different block sizes. The rightmost plot reveals that the parallel algorithm can be memory-bound on shared-memory machines (indicated by the dashed line).

memory. This includes already the four required shifts of the domain decomposition.

Interestingly, we found that the performance of the used optimization package MOSEK does not noticeably depend on the content of the data set, but only on its size.[6] This is good news, since it allows us to predict the performance for a given data set and block size rather accurately from the measurements shown in Figures 8a-b. Assume we want to compute the decomposed optimization for a 64^3 data set with a block size of 16^3. As discussed above, this makes 439 blocks including all shifts (smaller blocks will appear then, but we use the higher estimate here to be on the safe side). Each 16^3 block computes for 5 seconds on our hardware. This makes for a runtime prediction of 2195 seconds, which matches the measured time of 2110 seconds quite nicely (see Table 2 and Figure 8c). Note that these numbers are for computing on a single thread. How this scales to several threads is discussed below.

Side-by-side overviews of *measured* performance figures for the global and decomposed optimization are given in Tables 1-2 for 2D/3D data sets. Note that the decomposed optimization provides significant speedups over the global approach for 3D data sets and large 2D data sets. Furthermore, we can speedup the decomposed optimization even further by optimizing the blocks in parallel.

Scalability. Since we completely decoupled the blocks in our domain decomposition, the decomposed optimization can scale *linearly* with the number of parallel threads or processes. Since there is no need for communication between processes, it can easily run on clusters, where inter-process communication is usually difficult to implement, and achieve optimal scalability.

An interesting question, however, is how it scales on a shared-memory machine where parallel threads compete for access to the memory, which is usually the case on standard workstations. Figure 8c shows the scaling behavior on a machine with 96 GB main memory and 2 Intel XEON X5650 (2.66 GHz) with 6 cores each. As the plot indicates, the memory is fast enough to answer to two threads (speedup: 1.9), but we start to become memory-bound around 4-6 threads preventing an optimal usage of the available cores.

5.2 Data Weight, Single Cancellation, and Noise

In the following, we use simple experiments with the data set from Figure 9a to evaluate our method and provide an intuition about its specific characteristics.

The first experiment answers the question of what happens if all extrema of the scalar field are selected. We extracted the extrema from Figure 9a, kept all of them, and ran the numerical reconstruction. Figure 9c shows the results for two different data weights. As expected, the topology of the smoothed versions coincides with the original data: the extrema are at the same location and have the same value, and there are no additional extrema. This scenario nicely shows the influence of the data weight w_D from Equation (4): lower values emphasize the Laplacian energy E_L, higher values bring out the data term E_D.

[6] The performance of MOSEK is mainly determined by the quadratic coefficient matrix of (4). This matrix does not depend on the data itself, but just on the vertex connectivity in the grid, which is the same for every uniform grid.

The second experiment in Figure 9b shows what happens when we remove a single extremum. Note the targeted change in the middle of the domain where a single minimum has been removed. In all other aspects the data coincides with the right image of Figure 9c, since we use the same data weight. We can see from this that surgical operations are possible with our method: removing an extremum only has an effect in its immediate neighborhood, i.e., its Morse cell. Other parts of the domain are unaffected, since the extrema and the monotonicity graph are still the same there.

The third experiment demonstrates the utility of our approach when denoising data. We added 10% white noise to the data set, which created over 2000 additional minima and maxima. This is shown in the left image of Figure 9d. Persistence is a powerful topological tool to assess the importance of critical points. The critical points with the largest persistence are shown as large spheres. We instructed our method to keep only those while smoothing the data. The result is shown in the right image of Figure 9d. How to work with persistence-based denoising will be explained in more detail in Section 6 using a real data set.

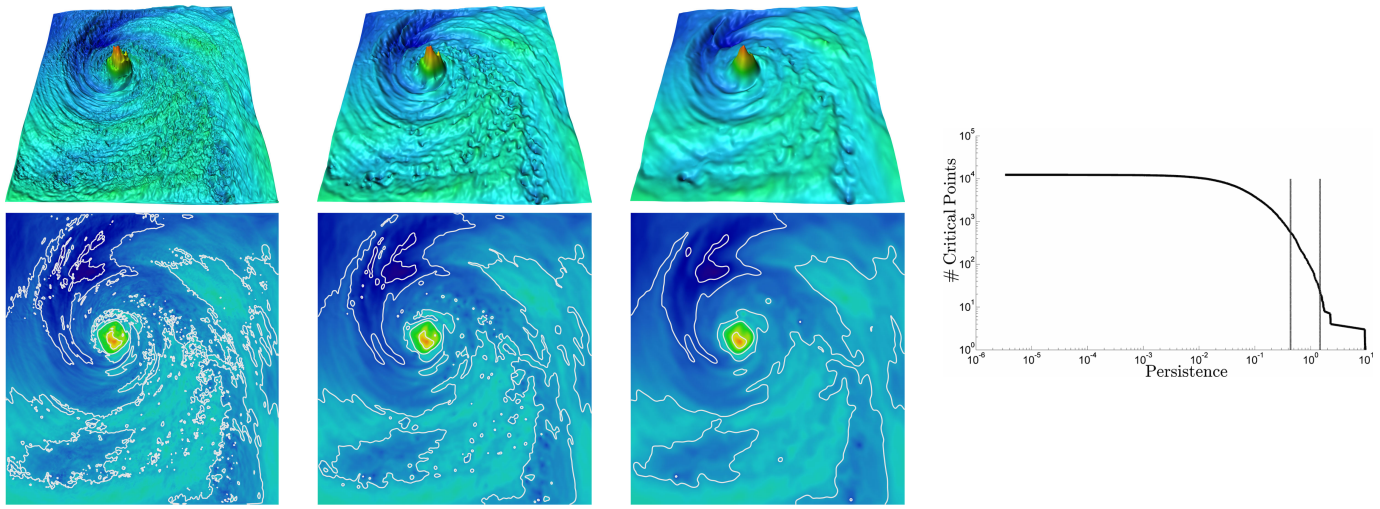
5.3 Comparison of Different Monotonicity Graphs

We presented two classes of methods for extracting and filtering extrema in Sections 3.1 and 3.2. They have different properties regarding their implementational effort and their computation time, as discussed previously. The convergence times for the subsequent reconstruction is shown in Table 3. Summarized, the approaches without Morse theory (Section 3.1) are easier to implement, but they create a data-unaware monotonicity graph that generally leads to more iterations until convergence. Approaches based on Morse theory (Section 3.2) are more involved, but the subsequent optimization benefits from the data-aware monotonicity graph by generally converging with fewer iterations.

The most important observation is that the final reconstruction results are *very similar* as proven by their normalized L^2 distances, see the rightmost column in Table 3. This agrees with the results of a larger experiment that we provide in the supplemental material. Our findings and our experience suggest that the choice for a specific monotonicity graph construction method can be guided by implementational effort and computation time – the iterative convexification seems to be able to make up for unfavorable start conditions.

Data set	Monoton. graph	Energy $E(s)$ first iter.	Energy $E(s)$ last iter.	# iters	Time in sec	L^2 dist.
3D spherical salt & pepper	harmonic	179.4	121.2	5	26.1	0.002
	topology	137.4	121.4	3	15.5	
2D vorticity	harmonic	48993	45521	38	264	0.05
	topology	63919	45479	17	117	

Table 3. Results for running the optimization using different monotonicity graphs, where "harmonic" refers to the solution of the Poisson problem (1), and "topology" to the approach using discrete Morse theory.



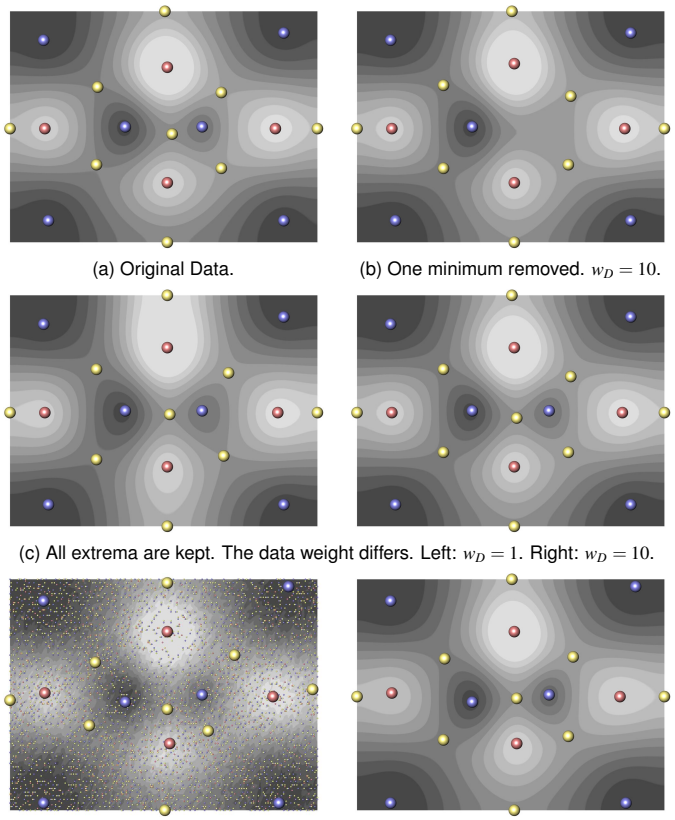
(a) Original data. 14492 extrema.

(b) Result for $P = 0.4$. 279 extrema.

(c) Result for $P = 1.5$. 12 extrema.

(d) Critical points over persistence P. The two vertical lines give the persistence values for (b) and (c).

Figure 10. Temperature in the Hurricane Isabel data set (slice $z = 20$). Using persistence-based filtering, we create a hierarchy of scalar fields: with increasing persistence P, our method creates increasingly smoother versions of the data. Note how the isolines (white) become less cluttered.



(a) Original Data.

(b) One minimum removed. $w_D = 10$.

(c) All extrema are kept. The data weight differs. Left: $w_D = 1$. Right: $w_D = 10$.

(d) Over 2000 noise-induced extrema are filtered using persistence. $w_D = 10$.

Figure 9. Three experiments reveal the characteristics of our method.

6 Applications

Hurricane Isabel. Topological structures are often filtered by their persistence [7,10,13,25]. Loosely speaking, the persistence P measures the prominence of a critical point in the data. The plot in Figure 10d shows this for a slice of the temperature of the Hurricane Isabel data set. The number of critical points drops drastically around $P = 10^0$: only few critical points have higher persistence values while most of them have lower persistence. Using our numerical reconstruction and persistence-based filtering, we can create a hierarchy of increasingly smoother scalar fields. Here, we show this using two persistence thresholds in Figures 10b-c. Compare this to the original data in Figure 10a.

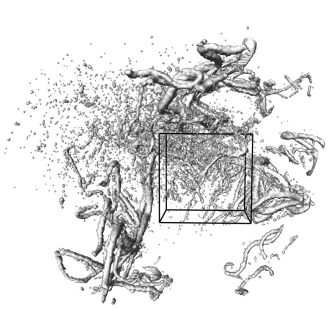
Another interesting observation in Figure 10 can be made regarding the white isolines. They represent the five different isovalues $\{0,2,4,6,10\}$. They are highly distorted for the original, unfiltered data. This is due to the large amount of small-scale extrema. In fact, this data set contains 14492 extrema. The denoised data in Figures 10b-c contains only the most persistent 279 and 12 extrema, respectively. It is known (e.g., [4]) that the number of extrema influences the number of connected components of an isoline visualization. Hence, the filtered data sets show less cluttered isolines. Note also that our denoising method preserves the value range in the data set, which makes the isolines directly comparable. Most importantly, the maximum temperature in all three versions of the data set is the same: 13.1 °C.

Teaser. We use persistence-based filtering also in Figure 1 to identify noise-induced minima and maxima in a 3D data set and remove them using our method. The volume rendering of the input scalar field (Figure 1a) reveals the high level of noise, which creates a large number of local minima and maxima (Figure 1b). The majority of these extrema have a low feature strength, i.e., their persistence is rather low. In Figure 1c, we only keep extrema with a persistence of at least 85% of the data range (9 maxima and 19 minima). The result of our reconstruction is a smooth and topologically clean scalar field (Figure 1d).

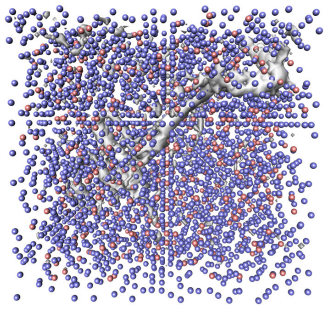
Aneurism. The aneurism data set in Figures 11-12 contains very thin blood vessels obstructed by a high noise level. Smoothing such a challenging data set with a simple Gaussian blur inadvertently interrupts the blood vessels (Figure 12), or better to say, an isosurface showing a blood vessel disintegrates into several connected components when smoothing without topological control.

Our method provides such control as shown in Figure 11. In this example, we zoomed on one of the thinner vessels. Figure 11b shows the large number of local extrema. We know from topology that every local minimum gives rise to a component of an isosurface. This guides our strategy for filtering this data set: we keep one sole minimum which is connected to the vessel (Figure 11c). This enforces a single-component isosurface as shown in Figure 11d. Note how the general appearance of the vessel is preserved. This shows the useful ability of our method to enforce topological constraints while denoising: the noise has been removed and the thin vessel structure has been preserved.

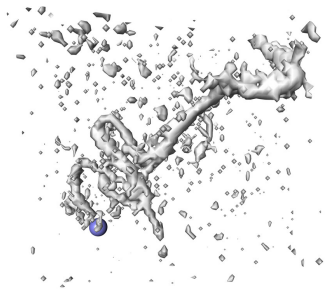
Lymphatic Capillary Network. To demonstrate the potential of our method for the task of image segmentation, we consider a biological data set. Figure 13a depicts a 2D image of a lymphatic capillary network. Here, biologists are interested in segmenting the tubular structures of the lymphatic vessels to reveal their patterns and connectivity.



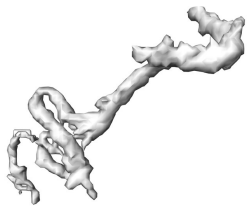
(a) Overview with zoom-in region.



(b) All minima and maxima.

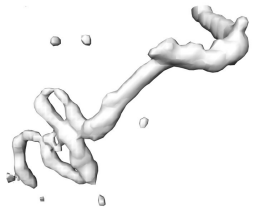


(c) One sole minimum selected to remain after the optimization.

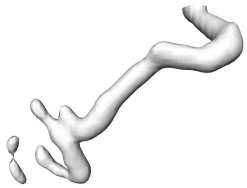


(d) The filtered data set reveals the thin vessel without obstructing noise.

Figure 11. Aneurism data set. Denoising with explicit control over the topology is useful in scenarios like this, where a thin blood vessel structure can be preserved including its delicate appearance, while the noise has been removed.



(a) Subtle Gaussian smoothing cannot remove all noise.



(b) Aggressive Gaussian smoothing interrupts the blood vessel.

Figure 12. Aneurism data set smoothed using a Gaussian filter. Compare to our method in Figure 11.

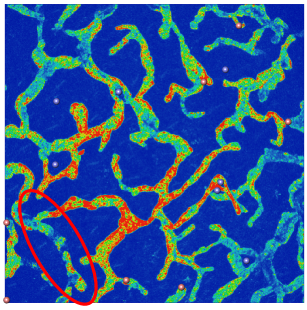
Unfortunately, the imaging process introduces noise and other artifacts causing discontinuities in the data. This can be observed in the lower left corner where a vessel is interrupted due to poor acquisition.

Naively thresholding the original raw data fails to connect this region of the vessel and introduces many speckles of noise throughout the segmentation (see Figure 13b). Applying simple Gaussian smoothing before thresholding removes most of the speckles (Figure 13c), yet the lymphatic vessel remains interrupted. This is because smoothing is only a local operation and cannot enforce such topological constraints.

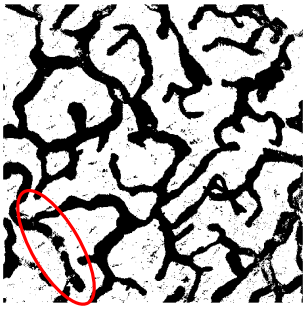
In contrast, our method allows the biologist to select connected components of the background (minima) and foreground (maxima). The boundary of the domain is set to the global minimum to reduce the number of minima that have to be placed. After applying our method, a threshold segmentation is forced to have the desired connectivity. Figure 13d shows this. Note that also all speckles are removed. This shows that topological denoising has utility beyond mere smoothing.

7 Conclusions and Future Work

We presented a topological denoising method that has significantly faster computation times and lower memory consumption than previous methods due to our novel domain decomposition approach. For the first time, this class of algorithms achieves practicable computation

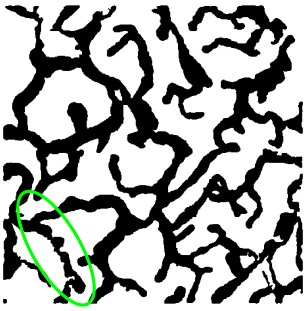


(a) Original data with selected extrema.



(b) Threshold segmentation of original.

(c) Threshold segmentation after applying a Gaussian filter.



(d) Threshold segmentation after applying our method.

Figure 13. Segmentation of a lymphatic capillary network with topological control. Our method denoises the data and enforces topological constraints such that a subsequent threshold segmentation reveals the connectivity of the network as desired by the biologists.

times for 3D scalar fields and large 2D data sets. Furthermore, we proposed an iterative scheme to drastically lower the energy of the solution by repeated convexification of the non-linear, non-convex feasible region. Regarding the extraction and filtering of extrema, we proposed a simple coupling between topological algorithms and the numerical reconstruction as well as a topological simplification scheme that allows user intervention. This leads to a versatile and powerful denoising method which we demonstrated using several examples.

Note that our contributions, in particular the domain decomposition and the iterative convexification, do not depend on a particular choice of the energy. In fact, our chosen energy (4) can be replaced by other formulations. This could be of interest in order to accommodate application-specific requirements. Our proposed method can also be extended to unstructured grids. As discussed in Section 3.2.1, the optimization requires an auxiliary grid, in which each of its vertices corresponds to a cell of the cell complex. Since this is difficult to achieve when we coarsen an unstructured grid, an explicit refinement would be necessary introducing a significant memory overhead.

Another interesting topic for future research is whether a domain decomposition approach can be developed that has a "natural" decoupling of its blocks. In particular, one could consider a topology-based decomposition of the domain, i.e., into Morse cells or Morse-Smale cells. However, no guarantees can be made about the size of those cells. One may end up with highly unbalanced block sizes, which can lead to poor performance. Also, it may be difficult to shift such a decomposition such that all vertices are free at least once.

A multi-resolution approach to the numerical optimization could lower the computation times even further. However, the main challenge is to represent all selected extrema at all resolution levels.

Acknowledgments

This research is supported and partially funded by the RTRA Digiteo *unTopoVis* project (2012-063D), the *TOPOSYS* project (FP7-ICT-318493-STREP), the ERC grant *iModel* (StG-2012-306877), the SNF award (200021_137879), the Intel Doctoral Fellowship, and MPC-VCC. We thank Michael Sixt and Kari Vaahtomeri for the biological data set.

REFERENCES

[1] U. Bauer, M. Kerber, and J. Reininghaus. Clear and compress: Computing persistent homology in chunks. In *Topological Methods in Data Analysis and Visualization III*, pages 103–117. Springer International Publishing, 2014. 3

[2] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE TVCG*, 10(4):385 – 396, 2004. 2

[3] A. Buades, B. Coll, and J. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation (SIAM interdisciplinary journal)*, 4(2):490–530, 2005. 2

[4] H. Carr and J. Snoeyink. Path seeds and flexible isosurfaces - using topology for exploratory visualization. In *Data Visualization 2003. Proc. VisSym 03*, pages 49–58, 2003. 8

[5] C. Chen, D. Freedman, and C. H. Lampert. Enforcing topological constraints in random field image segmentation. In *Proc. CVPR*, pages 2089–2096, 2011. 2

[6] G. Chen, K. Mischaikow, R. S. Laramee, P. Pilarczyk, and E. Zhang. Vector field editing and periodic orbit extraction using Morse decomposition. *IEEE TVCG*, 13(4):769–785, 2007. 2

[7] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28(4):511 – 533, 2002. 1, 3, 8

[8] R. Forman. Morse theory for cell-complexes. *Advances in Mathematics*, 134(1):90–145, 1998. 1, 3

[9] Y. I. Gingold and D. Zorin. Controlled-topology filtering. In *Proc. ACM SPM*, pages 53–61, 2006. 2

[10] D. Günther, H.-P. Seidel, and T. Weinkauf. Extraction of dominant extremal structures in volumetric data using separatrix persistence. *Computer Graphics Forum*, 31(8):2554–2566, December 2012. 8

[11] D. Günther. *Topological analysis of discrete scalar data*. PhD thesis, Saarland University, 2012. 3

[12] D. Günther, J. Reininghaus, H. Wagner, and I. Hotz. Efficient computation of 3D Morse-Smale complexes and persistent homology using discrete Morse theory. *The Visual Computer*, 28:959–969, 2012. 3

[13] A. Gyulassy. *Combinatorial construction of Morse-Smale complexes for data analysis and visualization*. PhD thesis, University of California, Davis, 2008. 8

[14] A. Jacobson, I. Baran, J. Popović, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics (ACM SIGGRAPH)*, 30(4):78:1–78:8, 2011. 2

[15] A. Jacobson, T. Weinkauf, and O. Sorkine. Smooth shape-aware functions with controlled extrema. *Computer Graphics Forum (Proc. SGP)*, 31(5):1577–1586, July 2012. 1, 2, 3, 4, 5, 6

[16] The MOSEK optimization software, http://www.mosek.com/. 4

[17] X. Ni, M. Garland, and J. C. Hart. Fair Morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 23(3):613–622, 2004. 2

[18] J. Nocedal and S. Wright. *Numerical Optimization, Second Edition*. Springer Series in Operations Research. Springer, 2006. 5

[19] G. Taubin. A signal processing approach to fair surface design. In *Proc. ACM SIGGRAPH*, pages 351–358, 1995. 2

[20] H. Theisel. Designing 2D vector fields of arbitrary topology. *Computer Graphics Forum*, 21(3):595–604, 2002. 2

[21] J. Tierny and V. Pascucci. Generalized topological simplification of scalar fields on surfaces. *IEEE Transactions on Visualization and Computer Graphics (Proc. VisWeek)*, 18(12):2005–2013, 2012. 2

[22] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proc. ICCV*, pages 839–846, 1998. 2

[23] X. Tricoche, G. Scheuermann, and H. Hagen. Continuous topology simplification of planar vector fields. In *Proc. Visualization*, pages 159 – 166, 2001. 2

[24] T. Weinkauf, Y. Gingold, and O. Sorkine. Topology-based smoothing of 2D scalar fields with C^1-continuity. *Computer Graphics Forum (Proc. EuroVis)*, 29(3):1221–1230, June 2010. 1, 2, 3, 4, 5

[25] T. Weinkauf and D. Günther. Separatrix Persistence: Extraction of salient edges on surfaces using topological methods. *Computer Graphics Forum (Proc. SGP '09)*, 28(5):1519–1528, July 2009. 8

[26] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Topological construction and visualization of higher order 3D vector fields. *Computer Graphics Forum (Proc. Eurographics 2004)*, 23(3):469–478, 2004. 2