A Practical Approach to Morse-Smale Complex Computation: Scalability and Generality

Attila Gyulassy, Peer-Timo Bremer, *Member, IEEE*, Bernd Hamann, *Member, IEEE*, and Valerio Pascucci, *Member, IEEE*

Abstract—The Morse-Smale (MS) complex has proven to be a useful tool in extracting and visualizing features from scalar-valued data. However, efficient computation of the MS complex for large scale data remains a challenging problem. We describe a new algorithm and easily extensible framework for computing MS complexes for large scale data of any dimension where scalar values are given at the vertices of a closure-finite and weak topology (CW) complex, therefore enabling computation on a wide variety of meshes such as regular grids, simplicial meshes, and adaptive multiresolution (AMR) meshes. A new divide-and-conquer strategy allows for memory-efficient computation of the MS complex and simplification on-the-fly to control the size of the output. In addition to being able to handle various data formats, the framework supports implementation-specific optimizations, for example, for regular data. We present the complete characterization of critical point cancellations in all dimensions. This technique enables the topology based analysis of large data on off-the-shelf computers. In particular we demonstrate the first full computation of the MS complex for a 1 billion/1024^3 node grid on a laptop computer with 2Gb memory.

Index Terms—Topology-based analysis, Morse-Smale complex, large scale data.

1 Introduction

Scientific data is becoming increasingly complex, and sophisticated techniques are required for its effective analysis and visualization. Additionally, data size increases accordingly with the size of memory, therefore analysis techniques must also be scalable. Topology-based visualization has become a useful technique in extracting features for a wide range applications, primarily due to its ability to simplify features in a controlled manner. The MS complex is a structure that represents the gradient flow behavior and completely encapsulates the topology of level sets of a scalar function. It has been shown to be effective in identifying, ordering, and selectively removing features. Computing a combinatorially correct MS complex is very challenging, and previous algorithms are memory-intensive and computationally expensive, restricting their use to smaller datasets.

We present a new algorithm for constructing a consistent MS complex: a framework which utilizes a divide-and-conquer strategy for dealing with large scale data in a variety of data formats and of any dimension. The kernel of our algorithm computes the discrete gradient on a *parcel* of the input data, generates an MS complex from the gradient on the parcel, and then merges the MS complexes together across the boundaries of the parcels. We use the discrete formulation of Morse theory as opposed to the continuous formulation for two reasons: it is simpler to implement because there are no special cases when dealing with higher dimensional components of the MS complex; and it makes it possible to fix the gradient flow on the boundary of parcels to enable a stratified approach. The discrete gradient and MS complex are computed independently on each parcel, and only the MS complex and gradient on the boundary of the parcel are necessary to merge parcels. We can control the size of the parcels and size of the MS complex through simplification to obtain a memory-efficient algorithm. We resolve degeneracies in the scalar function, such as flat regions and multi-saddles, in a consistent manner and construct the discrete gradient field and associated MS complex to agree with the scalar flow wherever possible. We characterize cancellation operations for MS complexes of any dimensions. We present the algorithm in a framework that makes it possible to implement multiple data formats by means of simple query functions, and also permits format-specific optimizations, for example, for regular grids. We show that this approach is comparable in performance to the fastest previous algorithm, but applicable to significantly larger data sets.

• *Attila Gyulassy is with UC Davis and Lawrence Livermore National Laboratory, E-mail: aggyulassy@ucdavis.edu.*
• *Peer-Timo Bremer is with Lawrence Livermore National Laboratory, E-mail: ptbremer@acm.org.*
• *Bernd Hamann is with University of California, Davis, E-mail: hamann@cs.ucdavis.edu.*
• *Valerio Pascucci is with University of Utah, E-mail: pascucci@acm.org.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mailto:tvcg@computer.org.

1.1 Related work

Often, features in a scalar field correspond to topological changes in the isosurface during a sweep of the domain. The life-cycle of a topological feature during this sweep is indicated by a pair of critical points, one indicating the creation of the feature and the other the feature's destruction. Topology-aware methods have proven to be effective in controlled simplification of scalar functions and hence in the creation of multiresolution representations. As opposed to geometry simplification using mesh decimation operators like edge contraction [7, 11, 12, 18, 23, 30], which result in unpredictable simplification of topological features, topology-aware methods either monitor changes to the topology [6, 13] or explicitly compute the topological features and perform necessary geometric operations to remove small features.

The Reeb graph [26] traces components of isosurfaces (or contours) as one sweeps through the allowed range of isovalues. In the case of simply connected domains, the Reeb graph has no cycles and is called a *contour tree*. Reeb graphs, contour trees, and their variants have been used successfully to guide the removal of topological features [7, 4, 14, 31, 32, 33, 3]. Of particular note is the approach by Pascucci et al. [24], which shows how the Reeb graph can be constructed in a streaming manner for large datasets. Reeb graphs and contour trees have been used to trace the construction, merging, and destruction of isosurface components. The MS complex, however, is a more complete description, since it also detects genus changes in isosurfaces.

Partitions of surfaces induced by a piecewise-linear function have been studied in different fields, under different names, motivated by the need for an efficient data structure to store surface features. Cayley [5] and Maxwell [22] proposed a subdivision of surfaces using peaks, pits, and saddles along with curves between them. The development of various data structures for representing topographical features was discussed by Rana [25].

The MS complex is a topological data structure that provides an abstract representation of the gradient flow behavior of a scalar field [29, 28]. Edelsbrunner et al. [9] defined the MS complex for piecewise-linear 2-manifolds by considering the PL function as the limit of a series of smooth functions and used this intuition to transfer ideas from the smooth case. They also provided an efficient algorithm to compute the MS complex, restricted to edges of the input triangulation, and to build a hierarchical representation by repeated cancellation of pairs of critical points.

Bremer et al. [2] improved on the algorithm and described a multi-resolution representation of the scalar field. Both algorithms trace paths of steepest ascent and descent beginning at saddle points. These paths constitute boundaries of 2D cells of the MS complex. Cells in the MS complex of a 3D scalar field can be of dimension 0, 1, 2, or 3. Tracing boundaries of the 3D cells while maintaining a combinatorial valid complex is a non-trivial task and a practical implementation of such an algorithm remains a challenge [8]. Nevertheless, the MS complex has been computed for volumetric data and successfully used to identify features through repeated application of atomic cancellation operations [15]. Computation of the complex in this manner requires a preprocessing step that subdivides every voxel by inserting "dummy" critical points, and therefore has a large computational overhead. This approach was improved by using a sweeping plane [16], but data size and computational overhead still proved to be a limiting factor. An algorithm based on region-growing [17] similar to the watershed transform [27, 1] was introduced for simplicial meshes of three dimensions, with a tenfold improvement in efficiency, however, the need to store several fields at each cell of the input, and the requirement to represent the entire output explicitly limits the scalability of this approach. In each of these approaches, the MS complex computed is *consistent*, meaning that the structure of the complex is combinatorially correct. Degenerecies in the data are overcome by consistent combinatorial decisions, resulting in MS complexes that reflect a particular interpretation of the input.

In our approach, we utilize discrete Morse theory as presented by Forman [10]. Lewiner et al. [21] showed how a discrete gradient field can be constructed and used to identify the MS complex, however, this construction requires modification of the input mesh and an explicit representation of gradient paths, restricting the applicability of the method. King et al. [19] presented a method for constructing a discrete gradient field that agrees with the large-scale flow behavior of the data defined at vertices of the input mesh. All of these algorithms for constructing the MS complex have a critical shortcoming: they require processing of the entire dataset and a representation of the complex at the finest level of detail before any simplification can be done. In practice, this imposes limits on both the size, and the complexity of the data that can be handled.

1.2 Contributions

We present the following new contributions:

- a complete characterization of cancellations of critical points of an MS complex of any dimensions in terms of how the cancellation affects the 1-skeleton and cells of the complex;
- a new algorithm for computing the discrete gradient field associated with a function with discrete samples at vertices of a mesh;
- a memory-efficient divide-and-conquer approach for constructing MS complexes, which computes portions of the complex independently and "glues" them back together; and
- a simple framework that supports large data of many formats and of any dimensions.

2 Background

Morse theory has been well-studied in the context of smooth scalar functions. However, scientific data is often presented as a set of discrete samples over a domain, often also involving a volumetric grid or a tetrahedralization, which necessitates an adaptation of the smooth theory. Discrete Morse theory is a parallel theory which is specially designed to operate on meshes.

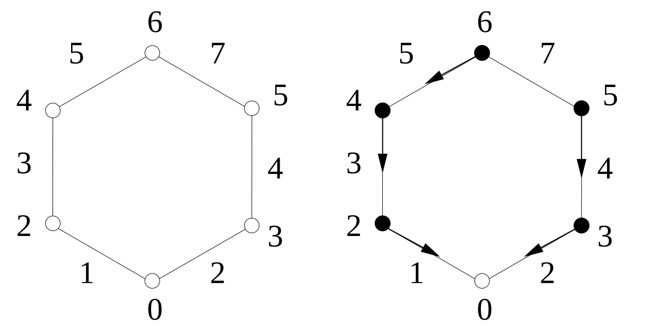


Fig. 1. A discrete Morse function (left) assigned to the simplices of a simple circle. The associated discrete gradient field (right) is a pairing of the vertices and edges. Note that the critical simplices of the discrete Morse function correspond exactly to the unpaired simplices of the discrete gradient field. In both cases, the vertex $f^{-1}(0)$ and the edge $f^{-1}(7)$ are the minimum and maximum respectively.

2.1 Morse Functions and the Morse-Smale Complex

A real-valued smooth map $f : \mathbb{M} \to \mathbb{R}$ defined over a compact d-manifold $\mathbb{M}$ is a *Morse function* if all its critical points are non-degenerate (*i.e.*, the Hessian matrix is non-singular for all critical points) and no two critical points have the same function value. An integral line of f is a maximal path in $\mathbb{M}$ whose tangent vectors agree with the gradient of f at every point of the path. Each integral line has a natural origin and destination at critical points of f where the gradient becomes zero. *Ascending* and *descending* manifolds are obtained as clusters of integral lines having common origin and destination respectively. The *Morse-Smale (MS) complex*, denoted Γ, is a partition of $\mathbb{M}$ into regions clustering integral lines that share common origin and destination. In Morse-Smale functions, the integral lines only connect critical points of different indices.

Each critical point of index n is the origin of a set of integral lines that forms an ascending $d-n$-manifold. Symmetrically, it is the destination of a set of integral lines that forms a descending n-manifold. All ascending and descending manifolds of a Morse-Smale function intersect transversally. Therefore, given two critical points a and b, where the index of a is one less than the index of b, the intersection of the ascending manifold of a and the descending manifold of b is either empty or a 1-manifold. The critical points and these 1-manifolds are called *nodes* and *arcs*. The one-skeleton formed by the nodes and arcs forms the *combinatorial structure* of the MS complex. The combinatorial structure contains much of the semantic information of f, and is useful for simplification and feature identification. The *neighborhood* of a node a of an MS complex Γ is the set of nodes N_a that are connected to a by an arc in Γ.

2.2 Discrete Morse Theory

The discrete Morse theory introduced by Forman [10], is a parallel theory to smooth Morse theory, and shows how to apply principles from smooth theory to the discrete setting. We present some basic definitions from discrete Morse theory. A d-cell is a topological space that is homeomorphic to a d-ball $B^d = \{x \in \mathbb{E}^d : |x| \leq 1\}$. For cells α and β, $\alpha < \beta$ means that α is a *face* of β and β is a *co-face* of α, *i.e.*, the vertices of α are a proper subset of the vertices of β. If $dim(\alpha) = dim(\beta) - 1$, we say α is a *facet* of β, and β is a *co-facet* of α. A cell α has dimension d, and we denote this as $\alpha^{(d)}$.

A *finite CW-complex* is a topological space X such that there exists a finite-nested sequence

$$\emptyset \subset X_0 \subset X_1 \subset \cdots \subset X_n = X$$

such that for each i = 0, 1, 2, ... , n, X_i is the result of attaching a cell to $X_{(i-1)}$. A *regular CW complex* is a finite CW-complex, where any two incident cells ρ and τ with $dim(\tau) = dim(\rho) - 2$, there are exactly two cells σ_1 and σ_2 such that $\tau < \sigma_1 < \rho$ and $\tau < \sigma_2 < \rho$. This requirement imposes restrictions on the attaching map, forcing

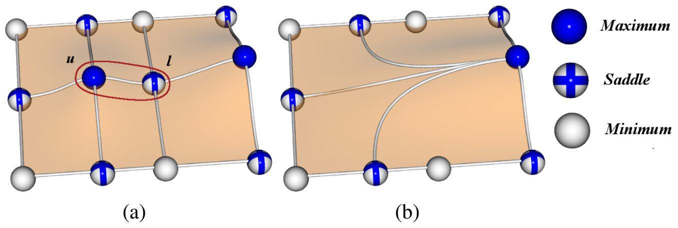


Fig. 2. The circled arc connects a saddle l to a maximum u (a). Cancellation of (l,u) removes all arcs attached to l or u, and creates new arcs from the lower neighbors of u to the upper neighbors of l (b). In the two-dimensional case, this connects all the saddles neighboring u to the maximum neighboring l, in effect, merging l and u with the maximum).

the entire boundary of an attached cell to be glued to the topological space, and restricting the boundary of a d-cell to be homeomorphic to a $(d-1)$-sphere.

Let K be a regular complex that is a mesh representation of $\mathbb{M}$. A function $f : K \to \mathbb{R}$ that assigns scalar values to every cell of K is a *discrete Morse function* if for every $\alpha^{(d)} \in K$, its number of co-faces $|\{\beta^{(d+1)} > \alpha | f(\beta) \le f(\alpha)\}| \le 1$, and its number of faces $|\{\gamma^{(d-1)} < \alpha | f(\gamma) \ge f(\alpha)\}| \le 1$. A cell $\alpha^{(d)}$ is critical if its number of co-faces $|\{\beta^{(d+1)} > \alpha | f(\beta) \le f(\alpha)\}| = 0$ and its number of faces $|\{\gamma^{(d-1)} < \alpha | f(\gamma) \ge f(\alpha)\}| = 0$. Figure 1 illustrates these configurations.

A *vector* in the discrete sense is a pair of cells $\{\alpha^{(d)} < \beta^{(d+1)}\}$, where we say that an arrow points from $\alpha^{(d)}$ to $\beta^{(d+1)}$. Intuitively, this simulates a direction of flow. A *discrete vector field* V on K is a collection of pairs $\{\alpha^{(d)} < \beta^{(d+1)}\}$ of cells of K such that each cell is in at most one pair of V.

Given a discrete vector field V on K, a V*-path* is a sequence of cells

$$\alpha_0^{(d)}, \beta_0^{(d+1)}, \alpha_1^{(d)}, \beta_1^{(d+1)}, \alpha_2^{(d)}, \ldots, \beta_r^{(d+1)}, \alpha_{r+1}^{(d)}$$

such that for each i = 0,..., r, the pair $\{\alpha^{(d)} < \beta^{(d+1)}\} \in V$, and $\{\beta_i^{(d+1)} > \alpha_{i+1}^{(d)} \neq \alpha_i^{(d)}\}$. A V-path is the discrete equivalent of a streamline in a smooth vector field. A discrete vector field in which all V-paths are monotonic and do not contain any loops is a *discrete gradient field*. We will use V-paths to compute the MS complex of a discrete gradient field.

2.3 Persistence-based Simplification

A function f is simplified by repeated cancellation of pairs of critical points. The local change in the MS complex indicates the smoothening of the gradient vector field and hence of the function f. The ordering of critical point pairs is defined by *persistence*, which quantifies the importance of the topological feature associated with a pair. The *persistence* of a critical point pair is the absolute difference in value of f between the two points. We use the ordering given by persistence to reduce the number of critical points and hence remove topological features from f.

A cancellation operation is *valid* (*i.e.*, it can be realized by a local perturbation of the gradient vector field) on a pair of critical points if and only if there is exactly one arc connecting them in the complex. Therefore, the indices of the two critical points must differ by one. Also, any critical point pair that is connected by multiple arcs represents a configuration known as a *strangulation* or a *pouch*, for which there is no direct perturbation of the gradient that removes the critical point pair.

We characterize the cancellation operation for MS complexes of any dimensions in terms of a change in the combinatorial structure of the complex. The geometric change in the manifolds of critical points can be derived from this combinatorial change.

Cancellation: Let Γ be an MS complex for a scalar function defined on a closed d-manifold $\mathbb{M}$. Let l and u be the lower and upper nodes of an arc a in Γ, with index i and $i+1$ respectively. Let A_l be the set of arcs that have l as one end point, A_u the set of arcs that have u as one end point, N_l the set of nodes in the neighborhood of l, and N_u the set of nodes in the neighborhood of u.

The *combinatorial cancellation* of (l,u) changes the combinatorial structure of the MS complex and is characterized as follows:

1. Create a new arc connecting every critical point of index $i+1$ in N_l to every critical point of index i in N_u, and add them to Γ.
2. All arcs in A_l, A_u are removed from the complex, and l and u are also removed from the complex.

This operation changes the 1-skeleton of the MS complex, however, it also represents a change in the embedding. This change can be derived from the combinatorial cancellation, and one simple way to maintain a valid embedding is to characterize it as a merging of the manifolds of the nodes involved in the cancellation. We call the change in the embedding the *geometric realization* of the cancellation and it is characterized as follows:

1. For every node of index $i+1$ in N_l, merge its descending manifold with the descending manifold of u.
2. For every node of index i in N_u, merge its ascending manifold with the ascending manifold of l.

The cells of Γ after the cancellation are only different where there are new intersections of the changed ascending and descending manifolds. These intersections are represented by the new arcs in the combinatorial structure of the MS complex. Although there are potentially $|N_l| \times |N_u|$ new arcs and cells created in the complex, the number of nodes in the complex is reduced by two, and eventually all the new arcs are also removed in saddle-extremum cancellations. Figure 2 illustrates this cancellation operation in the case of two-dimensional complexes.

3 Algorithm

Overview: Our algorithm for computing the MS complex relies on a divide-and-conquer approach. Figure 3 shows an overview of the algorithm. The divide-and-conquer approach divides the dataset into parcels where the discrete gradient and MS complex are computed locally on the boundary and in the interior. The boundary flow is fixed such that any flow passing through the boundary must pass through critical points restricted to the boundary. When two parcels are merged, the gradient and MS complex on the new interior is updated. In particular, the steps in this process are the following: the dataset is split into parcels, and cells are classified as interior, boundary, or exterior; the discrete gradient is computed on the boundary and interior of each parcel, and then an MS complex is computed on the interior of each parcel; the parcels are glued back together, merging the MS complexes; and finally, the artifacts introduced by the merging are removed by cancellation of the ε-persistence pairs of the MS complex. We describe each step in detail in the following.

3.1 Splitting and Classifying

A dataset may be given in any number of formats. It is partitioned into *parcels* during a pre-processing stage. Each parcel is a set of cells with certain attributes, which will be discussed in section 4.1. Logically, a parcel P is a collection of spatially coherent cells which form a regular CW complex, and is a subset of the input complex, $P \subseteq K$. The complex P has scalar values defined at its vertices.

The cells of a parcel P are categorized as interior, exterior, or boundary according to the following conditions:

1. A cell in P with all its faces and co-faces also in P is considered *interior* to P.
2. A cell with one or more of its faces in $K-P$ is considered *exterior* to P.
3. A cell with a co-face in $K-P$ or exterior to P is considered *boundary*.

The boundary cells form a closed manifold of dimension less than d for any input mesh P, where the highest dimension of any cell in P is d. The steps of partitioning the data, and classifying interior, exterior, or boundary cells are illustrated in Figure 3 (a) and (b).

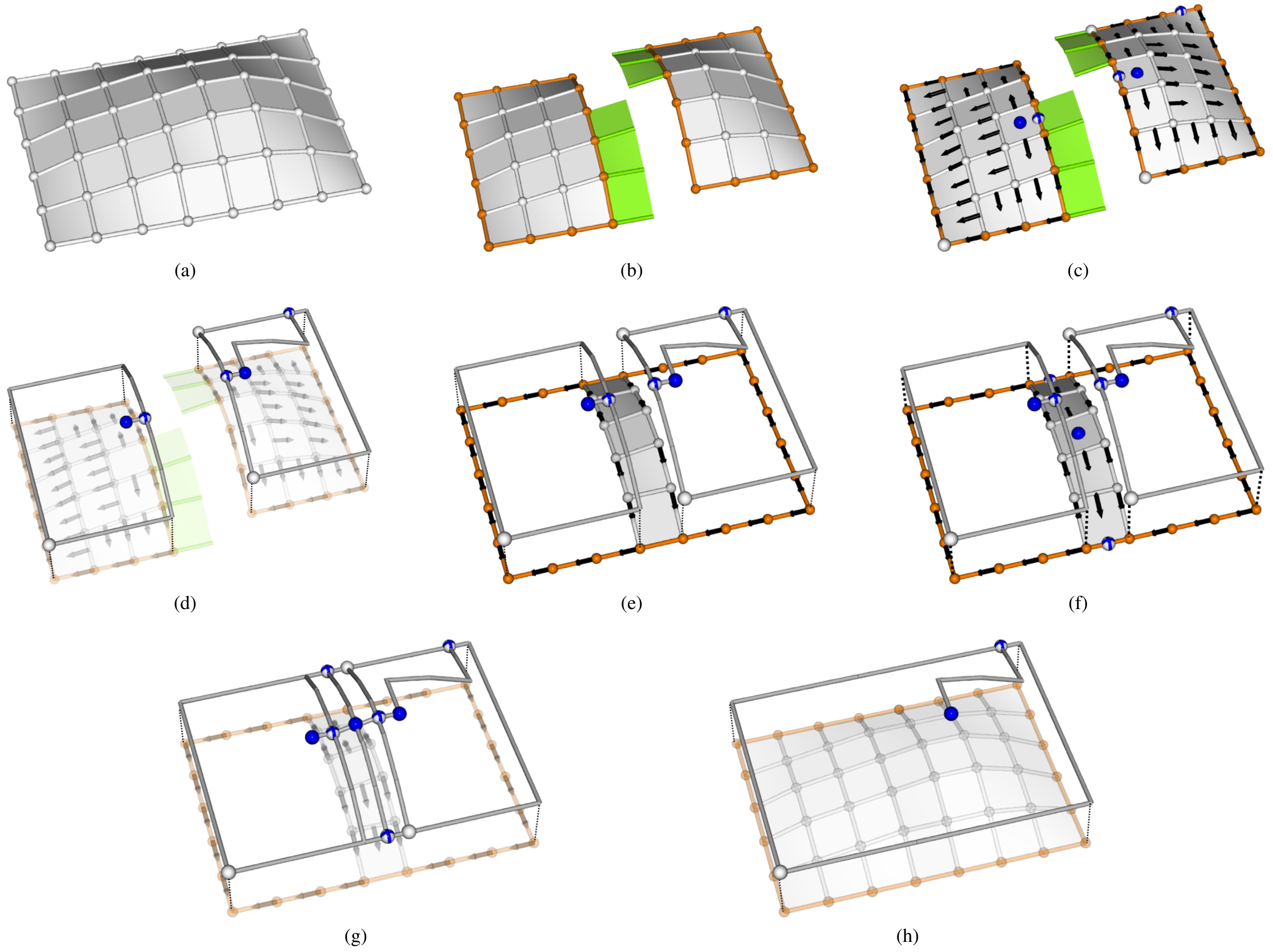


Fig. 3. An overview of the algorithm. The dataset (a) is broken up (b) into parcels, and cells are classified as interior (grey), exterior (green), or boundary (orange). The discrete gradient is found on the boundary and interior of each parcel (c) and the MS complex is computed (d). The parcels are merged, reclassifying boundary and exterior cells (e), and the discrete gradient is computed on the newly introduced boundary and interior cells (f). The complex is computed in these cells to complete the merging process (g). Simplification of ε-persistence pairs removes artifacts that have resulted from merging (h).

3.2 Computing the Discrete Gradient

Given a regular CW-complex K with scalar values defined at the vertices and cells of dimension d and lower, we compute the discrete gradient by assigning gradient arrows in a greedy manner in ordered sweeps over cells of K of increasing dimension, done by this algorithm:

```
 1: for $i \in [0, .., d]$ do
 2:   Iter = $K$ →sortedCellIterator($i$)
 3:   while Iter→hasNext() do
 4:     cellID = Iter→Item()
 5:     if ! $K$ →isMarked(cellID) then
 6:       if $K$ →hasPairableCoFacet(cellID) then
 7:         pairID = $K$ →lowestPairableCoFacet(cellID)
 8:         $K$ →pair(cellID,pairID )
 9:         $K$ →mark(cellID); $K$ →mark(pairID)
10:       else
11:         $K$ →setCritical(cellID)
12:         $K$ →mark(cellID)
13:       end if
14:     end if
15:   end while
16: end for
```

This algorithm iterates through all cells in increasing order of dimension and function value, assigning gradient pairs in a greedy manner. The *sortedCellIterator*(i) iterates through the i-cells of K in order of increasing function value. The lowest cell of dimension i that has not been paired or set as critical is *cellID*, and its co-facets are searched for a possible pair. The function *hasPairableCoFacet*$(cellID)$ returns true if there is a co-facet with exactly one facet that is not marked. If there are multiple such co-facets *lowestPairableCoFacet*$(cellID)$ will return the lowest one. While any pairable co-facet could be paired with the cell, we select the one in the direction of steepest descent to represent gradient flow of the function. When *pair*$(cellID, pairID)$ forms a gradient arrow, the cell is set as the head and the co-facet is marked as the tail.

The discrete vector field that is produced is a discrete gradient field, since each cell is paired exactly once, and no loops can be created in V-paths, since each V-path has a critical cell as a source that cannot be further paired. The flow across cells in a discrete Morse function does not necessarily correspond to scalar flow. However, we assign function values to all cells in a manner that allows gradient arrows to agree with the scalar flow. The particular sorting order of the cells of a dimension given by sorted cell iterator (line 2) and the lowest facet selected for pairing (line 7) determine the shape of the discrete gradient flow. If the sorting is done from lowest to highest, and the facet selected is in the direction of steepest descent, the discrete gradient flow generated will mostly agree with the scalar gradient.

Augmented Function The sorting of all the cells of a particular dimension requires function values to be assigned to every cell. We use the *augmented function F*, where each cell has a function value slightly larger than the highest value of its faces:

$$F(\alpha) = MAX\{\sigma : \sigma < \alpha\} + \varepsilon$$

In this manner, every cell is a critical cell in the discrete sense, and the formation of pairs performed by the algorithmic kernel corresponds to an ε-persistence cancellation of adjacent critical cells. We use symbolic perturbation to resolve the sorting order of two cells with the same function value.

Gradient on a Parcel Although the gradient computation kernel operates on any regular CW-complex, when computing the discrete gradient of a parcel we impose some restrictions. The boundary cells of a parcel represent the interface where flow can pass from one parcel to the next when the parcels are merged later on. To keep this interface as simple as possible, we first compute the discrete gradient on the boundary of a parcel and then on the interior. This re-ordering ensures that no boundary cell will be paired with an interior cell. The motivation behind this reordering is to restrict the number of places where flow can enter/exit the parcel from/to another parcel to the critical points on the boundary. This is an important property that will make merging parcels possible in subsequent steps of the algorithm. The computation of a gradient is a step in the algorithm shown in Figure 3(c), where the gradient is first computed on each parcel, and then the gradient is computed in the interior.

3.3 The MS Complex on a Parcel

The MS complex of a discrete gradient vector field is uniquely determined by the gradient paths. To compute an MS complex from the discrete gradient vector field produced by the algorithm kernel, we find the critical cells, and compute the ascending and descending manifolds by following the gradient paths. All the cells of a path whose origin is a critical cell α belong to the ascending manifold of α. Symmetrically, all the cells of a path whose destination is a critical cell β belong to the descending manifold of β. These paths are computed using a depth-first search through the discrete gradient field. The cells of the MS complex are attained as the intersection of ascending and descending manifolds. The nodes and arcs forming the combinatorial structure of the complex are the critical cells and the gradient paths connecting the critical cells.

Computing the ascending and descending manifolds requires a complete traversal of the gradient paths. Our algorithm computes all cells of the MS complex. However, many cases arise where analysis of the data only requires the combinatorial structure (1-skeleton) of the MS complex. In this case, we can perform a more efficient computation by only tracing ascending manifolds. Nevertheless, computing the ascending manifolds of all minima requires a complete traversal of the entire gradient field. If we are only interested in the combinatorial structure of the complex, this traversal is not necessary, and we compute 1-saddle-minima connections by tracing gradient paths downwards from the 1-saddles. It is guaranteed that there are exactly two paths that terminate at a minimum for each 1-saddle, and the paths cannot split. This makes the computation of 1-saddle minimum connections efficient.

In general, ascending and descending manifolds can merge in a discrete gradient field. We maintain the MS complex by simulating a separation between ascending manifolds and descending manifolds. The result of computing the MS complex on a parcel is shown in Figure 3(d). Note that one major advantage of using discrete Morse theory is that special rules for identifying manifolds of different dimensions are not required, as was the case in [17].

3.4 Merging Parcels

Merging two parcels is a three-step process that involves gluing the two meshes together and updating and classification of cells, computing the discrete gradient on the new boundary and interior, and finally merging the two MS complexes. Two meshes are glued together by updating the interior, boundary, or exterior classification on its cells. The interior cells remain interior, while the boundary cells can become interior cells or remain boundary, and the exterior cells can become interior or boundary or remain exterior. The same rules apply that were presented in section 3.1 for determining this classification. Figure 3 (e) shows the new classification of cells after this first step in the merging process.

We repeat the algorithmic kernel for finding the discrete gradient field on the merged parcels. First the gradient is computed for the cells that became boundary in the first step of the merging process, and then the gradient is computed for the cells that became interior in the first step of the merging process. Figure 3 (f) shows the discrete gradient computed on the new boundary and in the interior cells.

Finally, the MS complexes of each parcel are merged. Due to the way we first computed the discrete gradient on the boundary and then in the interior in section 3.3, flow can only enter or leave a parcel through its boundary critical points. Therefore, we extend the MS complex in each parcel by tracing gradient paths from all the newly classified interior and boundary critical cells. The only possible new connections in the merged MS complex are between newly classified interior critical cells. This fact makes it possible to remove the discrete gradient on the interior of each parcel from memory prior to the merging process, allowing for the memory-efficiency of the divide-and-conquer approach. Figure 3 (g) shows how the complexes are merged by connecting them with critical cells in the new boundary and in the interior.

Artifact Removal The merging of two parcels results in an MS complex on the new parcel with “extra” nodes and arcs where the old boundaries were. These extra nodes and arcs have low persistence, and are removed by canceling all ε-persistence pairs in the MS complex of a parcel. Figure 3 (h) shows how the MS complex in the interior is cleaned up by canceling low-persistence arcs.

4 A Framework for Generality

The algorithm described in the previous section relies on queries that are supported by a wide range of data formats. The internal representation of a parcel can vary based on the needs or optimizations possible for any particular data format. For example, for regular data, the connectivity and classification of cells is attainable directly from their indices and the extents of the parcel, therefore the queries can be resolved in an efficient manner. A more general data format, such as an AMR grid, or simplicial complex, may require a more elaborate storage mechanism. The data format handles queries regarding characteristics of the cells in a parcel as well as the connectivity of the cells within that parcel.

4.1 Queries on the Parcel

To compute the discrete gradient, certain queries must be implemented for the data structure. Given a unique identifier *id* for a cell, our implementation supports these functions:

- dimension(*id*): returns the dimension of the cell
- isPaired(*id*): returns true if the cell has been paired
- isPairable(*id*): returns true if the cell has one unpaired facet
- greaterThan(id_1, id_2): returns true if id_1 has higher function value than id_2

A parcel must also be able to provide iterators that access cells and their neighbors, which are implemented by these functions:

- *d*-cellIterator(): returns an iterator over all *d*-dimensional cells in the parcel in sorted order
- boundary-*d*-cellIterator(): returns an iterator over all *d*-dimensional cells on the boundary of a parcel in sorted order

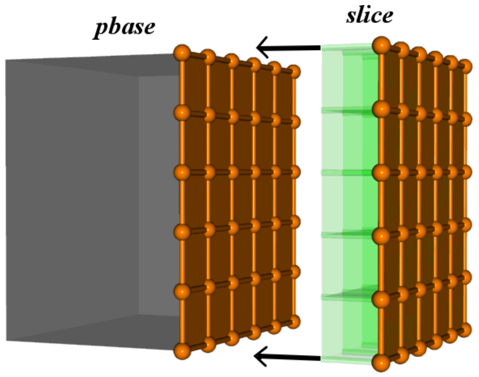


Fig. 4. A new slice is created to be merged with *pbase*. The orange cells mark boundary, the green mark exterior, and the grey area indicates the processed interior of *pbase*.

- neighborIterator(*id*): returns an iterator over the facets and co-facets of a cell

Finally, a parcel must also be able to change the state of some of its cells, which is enabled by the following functions:

- markCritical(*id*): marks the particular cell as a critical cell
- markAndPair(id_1, id_2): marks both as paired, and sets the pair of each to the other

To compute the MS complex from the discrete gradient field, the following queries must be supported:

- criticalPointIterator(): returns an iterator over the critical points in a parcel
- getPair(*id*): returns the identifier of the cell that *id* is paired with

Finally, the queries necessary for visualization of the complex require that the geometric information of a cell can be recovered from its identifier:

- getGeometry(*id*): returns an array of vertices that are the 0-dimensional faces of *id*

4.2 Flow of Control

The basic steps of the algorithm discussed in section 3 are used to compute and merge the MS complex on parcels, however, the order in which computation and merging take place are regulated by the *data manager*. The data manager is a data-format-specific module that organizes the flow of computation for maximal efficiency. In fact, each dataset may have its own data manager to optimize for any structural properties of the data format or feature queries in the function.

The details of dividing the data into parcels, ordering the computation of the gradient and MS complex on parcels, ordering the merging of parcels, and any on-the-fly simplification of the MS complex are left to the implementation. The interface with the algorithm is defined by the following functions:

1. computeGradAndComplex(Parcel p): computes the gradient and MS complex on boundary of a parcel; the results are stored in the state of p. Additionally, the parcel is prepared for merging, by removing the interior cells from memory.
2. merge(Parcel $p1$, Parcel $p2$): combines $p1$ and $p2$ into $p1$, internally updating interior, boundary, and exterior classifications, computing the discrete gradient on the new boundary and interior cells, merging the MS complexes, and performing an ε-persistence simplification. After merging, $p2$ is empty.
3. simplify(Parcel p, Filters f): performs simplification of the MS complex on p according to filters defined in f.

A parcel may only be merged with another if the gradient and MS complex have been computed on its boundary and in its interior, or if it composed entirely of exterior cells. It is the responsibility of the data manager to create parcels and load the data.

Case Study: Slices on a Grid The implicit structure of data defined on a regular grid (with inherent indices i, j, and k) can be exploited to achieve an efficient implementation. For example, the dimension, geometric location, and neighbors of a cell can be derived from its index in the cases of simple Cartesian or uniformly spaced rectilinear grids. Also, rectangular parcels can be defined as lower and upper extents in each dimension. In this case, the extents determine whether a cell is interior, exterior, or boundary. Merging the meshes of two aligned parcels can be accomplished by simply modifying the extents. All these properties make it possible to devise highly efficient implementations of the queries on the parcel and its cells.

In the following, we consider the simple case of a dataset defined on a rectilinear grid with uniform spacing, aligned with the three coordinate axes. The simplest possible data manager creates a parcel for every slice along one axis of the data. For example, in a 3D uniform rectilinear axis-aligned grid of size $X \times Y \times Z$, the data manager creates an $X \times Y$ parcel for every z value. Although there are many ways to order the creation and merging of parcels, the simplest way is to accumulate them in a growing parcel along one axis. In this case the data manager would operate as follows:

```
1: Parcel pbase = empty
2: for $z \in [0,..,Z-1]$ do
3:     Parcel slice = createXYSlice($z$)
4:     computeGradAndComplex(slice)
5:     merge(pbase, slice)
6:     simplify(pbase, filters)
7: end for
```

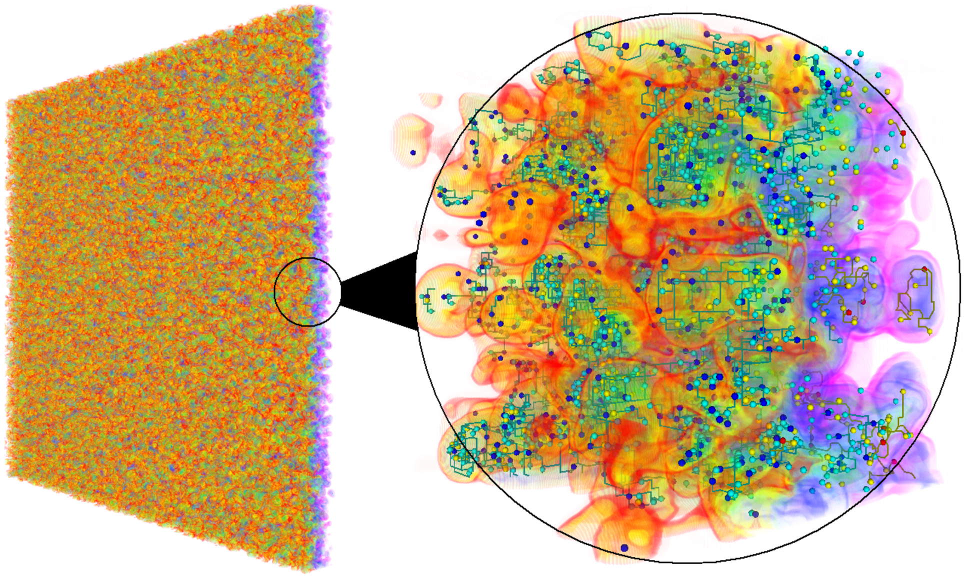
Here the createXYSlice(z) reads the data from the input file corresponding the the XY slice at z and initializes state variables in the slice. The slice that is created is an array of size $X \times Y \times 8$, required for storing the eight cells for every vertex of the data. The slices are merged with *pbase*, until *pbase* contains the complex of the entire mesh. Figure 4 shows the addition of a slice onto *pbase*. The simplification step controls the size of the complex as each slice is added on.

5 Results

We provide results for data given on a uniform rectilinear grid, using the slicing data manager. The results were generated on an off-the-shelf 2.21GHz AMD Athlon with 2.0Gb memory, the same hardware configuration used in [17]. We compare the performance of our algorithm to the previously fastest approach [17] in Table 1. Our run times are similar, however, while the approach in [17] has a dataset size limit of $256 \times 256 \times 256$, we also have produce results for one timestep of a simulation of a Raleigh-Taylor instability, which has a resolution of $1152 \times 1152 \times 1000$, shown in Figure 5. In Laney et al. [20], a parameter was used to select a 2-manifold level set, and the two-dimensional MS complex used on the height map to identify bubbles in the turbulent mixing layer. A similar kind of analysis is possible using the full complex we computed of the three-dimensional data without the need for parameter selection, nevertheless, performing this kind of in-depth analysis is beyond the scope of this paper. Our run times are larger for the small data set sizes because we simplify ε-persistence arcs on the fly, incorporating some of the post-processing into the construction time. The results of [17] do not include this extra time needed for simplification, and the complex extracted by that method must be simplified to remove noise and attain an MS complex comparable to the result of our approach. Nevertheless, the optimizations made possible by our general framework make the run times similar. The size of the memory footprint was controlled by the on-the-fly simplification, with the overhead for storing parcels being 42Mb, and the size of the complex kept under 1.3Gb.

6 Analysis of the Algorithm

Run Time Analysis The run time analysis of the complete algorithm is heavily dependent on the particular implementation of parcels and the data manager. The run time analysis of the particular steps of the algorithm can be performed using certain reasonable assumptions, such as access to the faces and co-faces of a cell requiring



Total Run Time	23h 15m 22s
∇ + MS-complex on parcel	2h 9m 45s
merging parcels	2h 38m 52s
5% simplification	18h 12m 41s
# cancellations	51,004,765
# parcels	1,000
# merge operations	1,000
# remaining critical points	957,560
# remaining arcs	6,320,506

Fig. 5. A single timestep of a dataset of a simulated Raleigh-Taylor instability simulating the mixing of two fluids. This timestep has a resolution of $1152 \times 1152 \times 1000$ and is an early timestep of the simulation. The data is noisy, therefore we perform a 5% persistence simplification to remove "excess features." We compute the complex for the entire dataset, and the inset shows a small subsection of the data with selected nodes and arcs of the complex. Minima and maxima (blue and red spheres) and their saddle connections trace out the bubble structure in the data. The maxima represent isolated pockets of high-density fluid that have crossed the boundary between the two fluids. The structural complexity is overwhelming, but our prototype allows interactive exploration and visualization, and selective inclusion/omission of user-specified components of the MS complex.

constant-time processing. The discrete gradient computation on a parcel with n cells uses a sorted ordering provided by the parcel, which is of complexity $O(nlogn)$. The computation of the MS complex on a parcel performs a depth-first search from each critical cell which will cover its entire ascending manifold. Since the ascending manifolds can merge, if there are $O(n)$ critical cells, in the worst case, this step can require $O(n^2)$ time. However, in practice, the number of critical points can be modeled as a constant k, and tracing the ascending and descending manifolds requires $O(n)$ time. Merging two parcels with m cells on the interface is accomplished in $O(mlogm)$ time, as the gradient computatation again requires $O(nlogn)$, and the merging of the complexes requires $O(m)$ time. The cancellation of a pair of nodes where the number of neighbors of each is bounded by some value i requires at most $O(i^2)$ time. Therefore, removing the artifacts introduced in a merge operation requires $O(k\, i^2)$.

We analyze the run time for the particular implementation used for generating the results, where slices are attached to a growing base for regular data. Let n be the total size of the data. For each of $n^{1/3}$ slices, a slice is read from the data, and cells are created and initialized in a traversal of the slice taking $n^{2/3}$ time. The gradient and complex are computed on the slice taking $n^{2/3}logn^{2/3} + n^{2/3}$ time. Each slice is then merged in $n^{2/3}logn^{2/3} + n^{2/3}$ time, and simplified in $i\, k^2$ time for a total run time of $O(nlogn) + n^{1/3} i\, k^2$. We do not remove the constant final term, since in the worst case this can lead to a total number of n^2 operations.

The memory requirements of our method are determined mainly by

Data set	Size	(a)	(b)
Neghip	$64 \times 64 \times 64$	8s	7s
Hydrogen	$128 \times 128 \times 128$	47s	27s
Aneurism	$256 \times 256 \times 256$	5m 1s	3m 51s
Instability	$1152 \times 1152 \times 1000$	23h 15m 22s	∞

Table 1. An MS complex is computed for well-known datasets. We compare the run time of our algorithm (a) to the fastest previously published algorithm presented in [17] (b).

two parts: the overhead required for storing the gradient on a parcel, and the storage required for the computed MS complex. Once the MS complex has been computed on a parcel, the interior cells can be removed from memory. In fact, a parcel, with its boundary gradient, external cells, and MS complex, only needs to be kept in memory during a merge operation. Let a parcel P have n interior cells and m boundary cells. During computation of the discrete gradient field, the total footprint of P is $(n+m) \times |\alpha| + |K| + |\Gamma|$, where $|\alpha|$ is the size of a single cell, $|K|$ is the memory overhead of the data structures storing the CW complex, and $|\Gamma|$ is the size of the MS complex computed on the parcel. During the merging of two parcels P_1 and P_2, the total amount of memory required is $(m_1 + m_2) \times |\alpha| + |K_1| + |K_2| + |\Gamma_1| + |\Gamma_2|$. The MS complexes Γ_1 and Γ_2 can be simplified independently prior to the merging operation to reduce their sizes. For regular data, the mesh connectivity is defined implicitly, therefore $K = 0$. In the particular implementation we used for generating the results, given a dataset of size $x \times y \times z$, each parcel is a slice of the data requiring $x \times y \times 8 \times |\alpha|$ space, and only two parcels (the base and the new slice) were kept in memory.

Implications of Divide-and-Conquer The discrete gradient is first computed on the boundary of a parcel and then in the interior, and the restriction of the flow on the boundary potentially creates different MS complexes for the same data if the data is divided in different ways. In simulation of simplicity, order-dependence determines the structures identified whenever degeneracies are encountered, such as flat regions and multi-saddles. The augmented function and the flexible ordering of the pairing of cells allow us to pick a particular ordering such that the flow can be fixed first on the boundary, then on the interior of a parcel, while maintaining consistency. In practice, a subset of the cells of the $d-1$-dimensional MS complex restricted to the boundary of a parcel form the intersection of the cells of the d-dimensional MS complex with the boundary. As a result, after merging parcels and simplifying the artifacts introduced in the process, the choices made in dividing the data result in only slight geometrical differences in the computed MS complexes. Most significantly, however, the complexes extracted are consistent, which means that they repre-

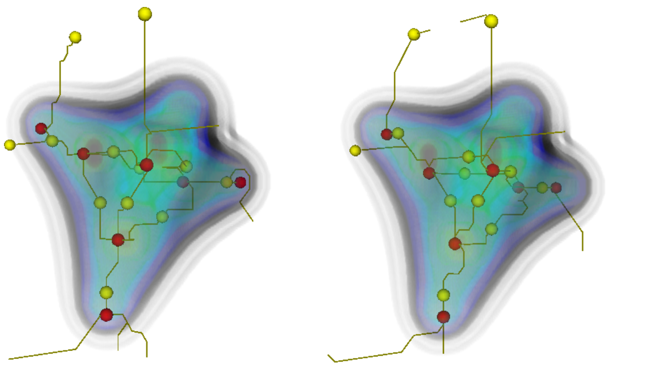


Fig. 6. The critical points of the tetrahedrane are identified using slicing across the z axis (left) and across the x axis (right). The locations of the critical points vary up to a cell, as do the shape of the arcs connecting them. However, the fundamental structure that is found by both methods is the same. Note that the MS complexes found with each slicing direction are consistent with one another, and both are consistent with the MS complex found for the same dataset in [15].

sent a Morse function arbitrarily close to the function defined by the scalar values at vertices. Figure 6 shows the difference when slicing the same data across the z axis and across the x axis. The discrete gradient produced is always a valid gradient field with monotonically descending V-paths.

7 Conclusions

The algorithm we presented in this paper is robust and efficient, and the framework is general and works for various data formats of any dimension. Our divide-and-conquer strategy allows for memory-efficient computation of the MS complex and simplification on the fly to control the size of the output. We sacrifice some time efficiency to gain relatively more storage efficiency and scalability. Furthermore, the operations on each parcel are independent and can be computed in parallel, and our future work will be directed at a parallel implementation. The algorithm works for data of any dimension, and we will investigate using slicing across the time axis in 3D + time data to track features over time. The majority of time in computing the MS complex is spent on simplification. Future work will involve finding a more efficient representation of the MS complex to further accelerate the cancellation process, and also finding an order of cancellations which leads to highly efficient execution.

Acknowledgements

Attila Gyulassy was supported by the Lawrence Scholar Program (LSP). In addition, this research was supported in part by the National Science Foundation, under grant CCF-0702817. We would like to thank the members of the Center for Applied Scientific Computing (CASC), at LLNL, and the members of the Visualization and Computer Graphics Research Group of the Institute for Data Analysis and Visualization (IDAV), at UC Davis. This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

[1] S. Beucher. Watershed, heirarchical segmentation and waterfall algorithm. In J. Serra and P. Soille, editors, *Mathematical Morphology and its Applications to Image Processing*, pages 69–76, 1994.
[2] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385–396, 2004.
[3] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *Symposium on Discrete Algorithms*, pages 918–926, 2000.
[4] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *Proc. IEEE Conf. Visualization*, pages 497–504, 2004.
[5] A. Cayley. On contour and slope lines. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, XVII:264–268, 1859.
[6] Y.-J. Chiang and X. Lu. Progressive simplification of tetrahedral meshes preserving all isosurface topologies. *Computer Graphics Forum*, 22(3):493–504, 2003.
[7] P. Cignoni, D. Constanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of tetrahedral meshes with accurate error evaluation. In *Proc. IEEE Conf. Visualization*, pages 85–92, 2000.
[8] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. In *Proc. 19th Ann. Sympos. Comput. Geom.*, pages 361–370, 2003.
[9] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry*, 30(1):87–107, 2003.
[10] R. Forman. A user's guide to discrete morse theory, 2001.
[11] M. Garland and P. S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proc. IEEE Conf. Visualization*, pages 263–269, 1998.
[12] M. Garland and Y. Zhou. Quadric-based simplification in any dimension. *ACM Transactions on Graphics*, 24(2):209–239, 2005.
[13] T. Gerstner and R. Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In *Proc. IEEE Conf. Visualization*, pages 259–266, 2000.
[14] I. Guskov and Z. Wood. Topological noise removal. In *Proc. Graphics Interface*, pages 19–26, 2001.
[15] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. Topology-based simplification for feature extraction from 3d scalar fields. In *Proc. IEEE Conf. Visualization*, pages 535–542, 2005.
[16] A. Gyulassy, V. Natarajan, V. Pascucci, P. T. Bremer, and B. Hamann. A topological approach to simplification of three-dimensional scalar fields. *IEEE Transactions on Visualization and Computer Graphics (special issue IEEE Visualization 2005)*, pages 474–484, 2006.
[17] A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann. Efficient computation of morse-smale complexes for three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1440–1447, 2007.
[18] H. Hoppe. Progressive meshes. In *Proc. SIGGRAPH*, pages 99–108, 1996.
[19] H. King, K. Knudson, and N. Mramor. Generating discrete morse functions from point data. *Experimental Mathematics*, 14(4):435–444, 2005.
[20] D. Laney, A. Mascarenhas, and P. Miller. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1053–1060, 2006. Member-P. -T. Bremer and Member-V. Pascucci.
[21] T. Lewiner, H. Lopes, and G. Tavares. Applications of forman's discrete morse theory to topology visualization and mesh compression. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):499–508, 2004.
[22] J. C. Maxwell. On hills and dales. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, XL:421–427, 1870.
[23] V. Natarajan and H. Edelsbrunner. Simplification of three-dimensional density maps. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):587–597, 2004.
[24] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of reeb graphs: simplicity and speed. *ACM Trans. Graph.*, 26(3):58, 2007.
[25] S. Rana. *Topological Data Structures for Surfaces: An Introduction to Geographical Information Science.* Wiley, 2004.
[26] G. Reeb. Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus de L'Académie ses Séances, Paris*, 222:847–849, 1946.
[27] J. Roerdink and A. Meijster. The watershed transform: Definitions, algorithms and parallelization techniques, 1999.
[28] S. Smale. Generalized Poincaré's conjecture in dimensions greater than four. *Ann. of Math.*, 74:391–406, 1961.
[29] S. Smale. On gradient dynamical systems. *Ann. of Math.*, 74:199–206, 1961.
[30] O. G. Staadt and M. H. Gross. Progressive tetrahedralizations. In *Proc. IEEE Conf. Visualization*, pages 397–402, 1998.
[31] S. Takahashi, G. M. Nielson, Y. Takeshima, and I. Fujishiro. Topological volume skeletonization using adaptive tetrahedralization. In *Proc. Geometric Modeling and Processing*, pages 227–236, 2004.
[32] S. Takahashi, Y. Takeshima, and I. Fujishiro. Topological volume skeletonization and its application to transfer function design. *Graphical Models*, 66(1):24–49, 2004.
[33] Z. Wood, H. Hoppe, M. Desbrun, and P. Schröder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics*, 23(2):190–208, 2004.