Parallel Computation of 2D Morse-Smale Complexes

Nithin Shivashankar, Senthilnathan M, and Vijay Natarajan, *Member, IEEE*

Abstract—The Morse-Smale complex is a useful topological data structure for the analysis and visualization of scalar data. This paper describes an algorithm that processes all mesh elements of the domain in parallel to compute the Morse-Smale complex of large two-dimensional data sets at interactive speeds. We employ a reformulation of the Morse-Smale complex using Forman's Discrete Morse Theory and achieve scalability by computing the discrete gradient using local accesses only. We also introduce a novel approach to merge gradient paths that ensures accurate geometry of the computed complex. We demonstrate that our algorithm performs well on both multicore environments and on massively parallel architectures such as the GPU.

Index Terms—Topology-based methods, discrete Morse theory, large datasets, gradient pairs, multicore, 2D scalar functions.

1 INTRODUCTION

The Morse-Smale (MS) complex of a real-valued function is an abstract representation of its gradient flow behavior. It has been extensively studied both within the computational geometry and the visualization communities. Research within the computational geometry community has resulted in better understanding of the mathematical structure of the complex and has led to efficient algorithms to compute the MS complex for piecewise linear (PL) scalar functions [6], [7]. On the other hand, work within the scientific visualization community has focused on efficient computation of the MS complex in practice [4], [11], [13], [14] and effective application to the analysis and visualization of 2D and 3D scalar fields [12], [17]. Data sizes grow faster than processor speeds resulting in an ever-present demand for better algorithms to process the data. In this paper, we describe a parallel algorithm to compute the MS complex. Our algorithm utilizes the multiple cores available in the CPU and GPU of a typical desktop computer to compute the MS complex of large two-dimensional data, consisting of several hundred million vertices, within a few minutes.

The definition and computation of the MS complex for sampled functions requires gradient / steepest path computation and path tracing, which is inherently serial in nature. We prove two lemmas on gradient flow paths and symbolic perturbation that lead to an algorithm for computing the cells of the MS complex in a few massively parallel steps.

- *Nithin Shivashankar and Senthilnathan M are with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, 560012.*
  E-Mail nithin@csa.iisc.ernet.in and senthil@csa.iisc.ernet.in
- *Vijay Natarajan is with the Department of Computer Science and Automation, and Supercomputer Education and Research Center, Indian Institute of Science, Bangalore, 560012.*
  E-Mail vijayn@csa.iisc.ernet.in

1.1 Related work

Topology-based methods have become very effective for controlled simplification of features in scalar fields. These methods are primarily based on ideas from Morse theory [19], the study of the relationship between critical points of smooth functions and the topology of the domain. The Morse-Smale complex partitions the domain into regions. Each region, defined by a pair of critical points of the scalar function, is covered by gradient flows between the critical point pair. MS complexes were introduced first to study dynamical systems [24], [25]. Edelsbrunner et al. [7] first posed the problem of computing the MS complex for piecewise linear functions defined on two-dimensional manifolds. The function was sampled at vertices of a mesh that represented the domain and linearly interpolated within mesh elements. They interpreted the piecewise linear function as the limit of a series of smooth functions and hence used ideas from Morse theory to classify critical points, follow gradient flows, and compute cells of a quasi MS complex whose bounding arcs are restricted to edges of the input mesh. The combinatorial structure of the quasi MS complex was proved to be identical to that of the MS complex. Henceforth, we refer to the quasi MS complex simply as the MS complex. A similar approach was employed to construct MS complexes of three-dimensional functions [6]. Changes in the topology of isosurfaces of the scalar function during a sweep of the domain correspond to the features of interest. Pairs of critical points represent the creation and destruction of the feature during the sweep. Hence, topological simplification refers to the removal or cancellation of a pair of critical points.

Bremer et al. [4] focused on efficient computation of the MS complex, building a multi-resolution representation of the scalar field via controlled topological simplification, and application of the MS complex to various data analysis and visualization tasks including feature identification, noise removal, and view-dependent simplification. These early approaches were based on tracing the gradient paths from saddle critical points, which produced a boundary representation of cells in

the MS complex. Later approaches [12], [13] especially for three-dimensional functions, were based on repeated cancellations applied on an artificial complex created by including dummy critical points. The cancellations were appropriately scheduled in order to remove the dummy critical points leaving behind the true critical points and cells of the MS complex. The scheduling of the critical point pairs for cancellation plays a crucial role both in determining the quality of the result and the efficiency of the algorithm.

Forman developed discrete Morse theory, an analog of Morse theory used to study cell complexes and discrete functions defined on them [10]. King et al. described a method for computing a discrete function on a mesh given a function sampled at mesh vertices while guaranteeing that the discrete gradient field agrees with the large-scale flow behavior of the input [16]. Reininghaus et al. [21], [22] discuss an application of discrete Morse theory to analyze vector fields. Bauer et al. [1] discuss computing simplified functions on surfaces such that the input function is modified by no more than a threshold δ and all surviving critical point pairs have persistence greater than 2δ. Discrete Morse theory has also been successfully used to compute the MS complex of piecewise linear functions. Early work based on this approach by Cazals et al. [5] and Lewiner et al. [18] demonstrated applications to segmentation, visualization, and mesh compression. More recently, Gyulassy et al. [11] employed this approach for efficient computation of MS complexes of large data that do not fit in main memory. They partition the data into blocks called "parcels", compute gradient flows on the boundary of the parcels, propagate the flows to the interior and compute the MS complex restricted to the parcel. The critical cells created on the boundary are canceled during a subsequent merge step resulting in the MS complex of the union of the parcels. This method scales well for large data. However, the geometry of the MS complex computed using this method is sensitive to the order of cancellations chosen during the merge step.

Robins et al. [23] proposed an algorithm to compute the Morse complex of 2D and 3D grayscale digital images modeled as discrete functions on cubical complexes. While the algorithm computes the Morse complex with provable guarantees on its correctness with respect to the critical cells, it does not guarantee the geometric accuracy of the complex. Further, the algorithm does not scale to large datasets.

In summary, the above mentioned methods are slow because (a) they compute and trace the gradient serially or (b) do not guarantee that they trace the correct geometry of the gradient flow. We address the former shortcoming by designing a massively data parallel algorithm and the latter by ensuring that we reproduce the gradient flows independent of the choice of partition.

1.2 Results

The main result of this paper is a parallel algorithm to compute the MS complex of a two-dimensional scalar function. We partition the domain into sub-domains, compute gradient flows within each sub-domain, and merge the gradient flows while merging the sub-domains. The combinatorial connectivity of the MS complex is computed during the merge step. The geometry of the cells of the MS complex is computed in a subsequent traversal of a history tree that records the merges. The correctness and efficiency of the algorithm is based on two key lemmas that are valid for all dimensions:

- The Order Independent Pairing Lemma, which states that the discrete gradient pairs that define the gradient field can be computed independent of the order in which the cells are processed.
- The Order Independent Cancellation Lemma, which states that the geometry of the gradient flow is computed correctly independent of the order in which the critical point pairs on the sub-domain boundary are canceled.

We discuss novel implementation strategies to ensure that the massive parallelism available in GPUs is fully utilized. We also describe parallel methods to query the 2D MS complex for feature identification and visualization. We demonstrate using synthetic and real-world data that the algorithm is able to compute the MS complex of very large data sets that do not fit in main memory. We also discuss an application of our algorithm to efficient processing and tracking of features in 2D time-varying data.

1.3 Outline

Section 2 presents the necessary background on Morse-Smale complexes and topological simplification. Section 3 presents an overview of our parallel algorithm and Sections 4-5 describe the algorithm in detail. Section 6 discusses implementation details and Section 7 presents experimental results. Section 8 concludes the paper.

2 Background

This section reviews the necessary background on Morse functions and discrete Morse functions required for the algorithm description.

2.1 Morse functions

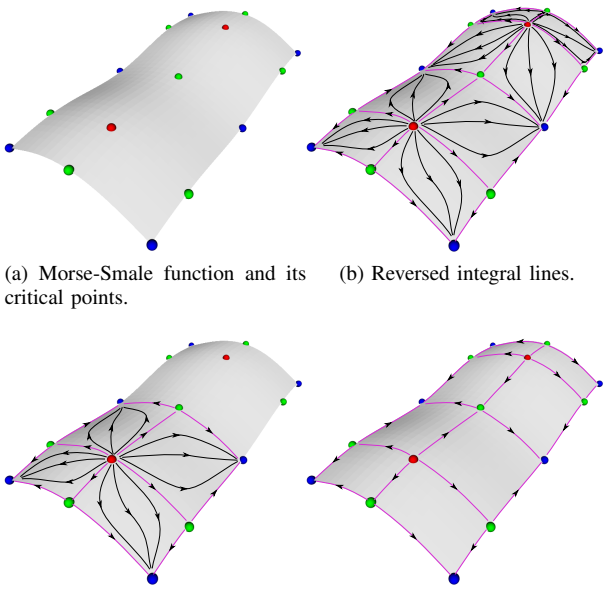
Consider a smooth scalar function $f : \mathbb{R}^n \to \mathbb{R}$. A point $p \in \mathbb{R}^n$ is called a *critical point* with respect to f if the *gradient* of f,

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right),$$

is identically zero at p. A critical point is non-degenerate if the *Hessian* of f, equal to the matrix of second order partial derivatives, is non-singular. We call f a *Morse function* if all of its critical points are non-degenerate.

The *index* of a critical point is the number of negative eigenvalues of the Hessian matrix. An *integral line* passing through a point p is a one-dimensional curve $l : \mathbb{R} \to \mathbb{R}^n$, where $\frac{\partial}{\partial t} l(t) = \nabla f(l(t)),\ \forall t \in \mathbb{R}$ and $l(0) = p$. In other words, it is a maximal curve in $\mathbb{R}^n$ whose tangent at every point equals the gradient of f at that point. The function f increases along the integral line. The limit points of integral lines, $t \to \pm\infty$, are the critical points of f.

The set of all integral lines that share a common source $p = \lim_{t \to -\infty} l(t)$, together with the point p, is called the *ascending*



(a) Morse-Smale function and its critical points.

(b) Reversed integral lines.

(c) Descending manifold of a maximum.

(d) Combinatorial structure of the MS complex.

Fig. 1: (a) A Morse-Smale function shown as a height field over a two dimensional domain. Critical points are shown in red, green, and blue corresponding to maxima, saddle, and minima, respectively. (b) The reversed integral lines of the function over the surface. (c) The descending manifold of a maximum shown as the closure of the set of reversed integral lines that originate from the critical point. (d) Combinatorial structure of the MS complex where nodes are critical points and connecting integral lines are arcs.

manifold of p and the set of all integral lines that share a common destination $p = \lim_{t\to\infty} l(t)$, together with the point p, is called the *descending manifold* of p. The ascending manifolds of all critical points partition the domain. Similarly the descending manifolds of all critical points also partition $\mathbb{R}^n$. The *Morse-Smale complex* is a partition of $\mathbb{R}^n$ into cells formed by the collection of integral lines that share a common source and a common destination.

The ascending manifold of a critical point of index d is a $(n-d)$-dimensional manifold, where as its descending manifold is an n-dimensional manifold. A Morse function f is called a *Morse-Smale function* if all ascending and descending manifolds of two critical points intersect transversally. Thus, if the index of two critical points differ by one then their ascending / descending manifolds either do not intersect or intersect along a one-dimensional manifold connecting the critical points. The critical points, referred to as *nodes*, along with the 1-manifolds that connect them, referred to as *arcs*, form the 1-skeleton of the MS complex, which is referred to as the combinatorial structure of the complex.

2.2 Simplification

A Morse-Smale function f can be simplified to a smoother function by repeated application of a cancellation operation that removes a pair of critical points connected by an arc in the MS complex. This cancellation corresponds to the removal of the feature represented by the critical point pair. Features are ordered based on the notion of *persistence*, equal to the absolute difference in function value between the two critical points. Persistence measures the importance of a critical point pair [8]. More sophisticated measures of importance based on persistence have also been described in the literature. Since the focus of this paper is on the computation of MS complex and not necessarily on efficient simplification, we restrict our discussion to the persistence measure. The least persistent critical point pair is always connected by an arc in the MS complex [7].

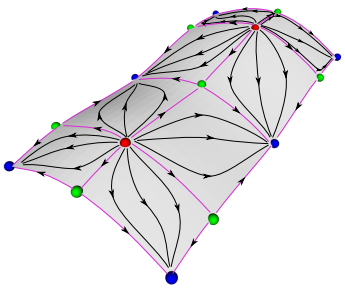
Simplification of a pair of critical points can be achieved by a local smoothing of the function in the neighborhood of the two critical points, more precisely within the ascending / descending manifolds containing the critical points. The cancellation is realized by updating the 1-skeleton of the MS complex. For example, consider the case of a two-dimensional Morse-Smale function after a maximum-saddle cancellation. The 1-skeleton is updated by deleting the two nodes, deleting the arcs incident on the saddle, and re-routing the arcs incident on the maximum to the surviving maximum adjacent to the saddle (see Figure 2). The embedding of a new arc is obtained by extending the old arc along the arc between the maximum and saddle. We allow only those cancellations that can be realized by a local smoothing of the function. This is feasible if the pair of critical points is connected by a single arc. Canceling a pair of critical points that are connected by two distinct arcs in the Morse-Smale complex results in a *strangulation*, which cannot be realized by a local smoothing of the function.

2.3 Piecewise Linear(PL) Functions

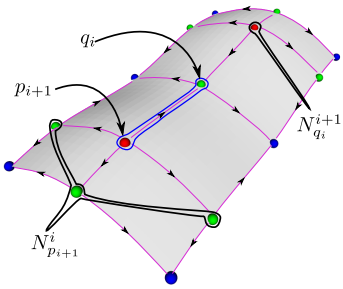
Earlier approaches to compute MS complexes were based on PL extensions of functions sampled at vertices of simplicial complexes [6], [7]. Though we adopt the discrete formulation of MS complexes for our computations, we introduce here some notions of PL function so that we may establish the closeness of our approach to the PL approach. For further reading on the basic notions of algebraic topology, we refer the reader to the classic text books by Munkres [20] and Hatcher [15].

A function f sampled at vertices of a simplicial complex may be extended to form a continuous function that is linear on every cell. The *star* of a vertex v is the set of simplices incident on v. The *link* of a vertex v is the set of faces of cells in the star of v, that are not incident on v. The *lower star* of vertex v is the set of cells in the star where the PL extension assumes values lower than $f(v)$. The *lower link* of a vertex v is the set of faces of cells in the lower star of v, that are not incident on v.

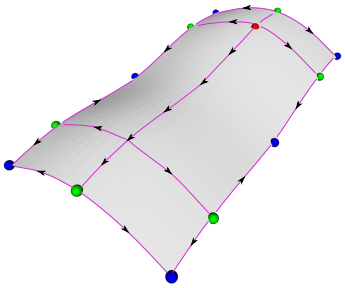
The *Betti numbers* of a cell complex, K, are a useful characterization of the underlying space of a cell complex. They are defined for each $k = 0, 1.., dim(K)$ and denoted by β_k. Intuitively, β_0 counts the number of components of K, β_1 counts the number of tunnels in K and β_2 counts the number of voids of K. The *reduced Betti number*, denoted by $\tilde{\beta}_k$ and



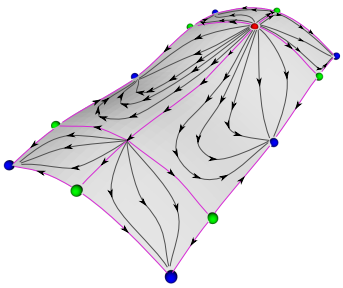
(a) MS complex and reversed integral lines



(b) Canceling a pair of critical points



(c) Combinatorial realization



(d) Geometric realization

Fig. 2: (a) MS complex for a simple height function. (b) Canceling a pair of critical points, q_i, p_{i+1}, of index i, $i+1$ that are connected by a single 1-manifold. (c) Combinatorial realization: connect all index i critical points ($N_{p_{i+1}}^{i}$) that are connected to p_{i+1} except q_i, to index $i+1$ critical points ($N_{q_i}^{i+1}$) that are connected to q_i except p_{i+1}. (d) Geometric realization: compute the union of the descending manifold of p_{i+1} with the descending manifolds of all index $i+1$ critical points connected to q_i. Compute the union the ascending manifold of q_i with the ascending manifolds of all index i critical points connected to p_{i+1}.

defined for $k = -1, 0, ..., dim(K)$, is exactly the same as β_k for all $k = 1, ..., dim(K)$. The zeroth reduced Betti number is $\beta_0 - 1$ if $\beta_0 > 0$ and 0 otherwise. Also $\tilde{\beta}_{-1} = 1$ if $\beta_0 = 0$ and 0 otherwise.

Reduced Betti numbers of the lower link can be used to classify a vertex of a simplicial complex as non-critical (regular) or critical and to further classify critical vertices. A vertex in a 2D simplicial complex is said to be regular if all $\tilde{\beta}_k$'s of the lower link are zero, minimum if $\tilde{\beta}_{-1} = 1$ and $\tilde{\beta}_0 = \tilde{\beta}_1 = 0$, simple saddle if $\tilde{\beta}_0 = 1$ and $\tilde{\beta}_{-1} = \tilde{\beta}_1 = 0$, and maximum if $\tilde{\beta}_1 = 1$ and $\tilde{\beta}_{-1} = \tilde{\beta}_0 = 0$. Critical points with $\tilde{\beta}_0 > 1$ are called multi-saddles.

The weak Morse inequality is a classic result of Morse theory which states that, given a Morse function f defined on a manifold, the number of index k critical points of f is greater than or equal to the k^{th} Betti number [20]. Forman established the analogous result for discrete Morse functions [10].

2.4 Discrete Morse functions

Discrete Morse theory was developed by Forman [10] to study the topology of cell complexes. A d-cell α^d is a topological space homeomorphic to a d-ball $B^d = \{x \in \mathbb{E}^d : |x| \leq 1\}$. For example, a vertex is a 0-cell, an edge between two vertices is a 1-cell, a polygon is a 2-cell, and in general a d-dimensional polytope is a d-cell. We will restrict our attention to cells of the above kind, which can be represented by a set of vertices. A cell α is a *face* of β, denoted $\alpha < \beta$, if α is represented by a subset of vertices of β. The cell β is called a *coface* of α. A face α is called a *facet* of β if $\alpha < \beta$ and $dim(\alpha) + 1 = dim(\beta)$. In this case β is a *cofacet* of α denoted by $\alpha \lessdot \beta$. The set of zero-dimensional faces of a cell α is called the vertex set of α denoted by V_α.

A *cell complex* K is a collection of cells that satisfies two properties: (a) If α belongs to K then so do all faces of α, and (b) If α_1 and α_2 are two cells in K then either they are disjoint or they intersect along a common face. A *regular cell complex* is a cell complex in which, given two incident cells, β^{d+1} and γ^{d-1}, there are exactly two cells α_1^d, α_2^d such that $\gamma^{d-1} < \alpha_1^d, \alpha_2^d < \beta$. In this paper, we consider only finite regular cell complexes. A *filtration* of a cell complex K is a sequence of nested cell complexes $K_0, K_1, \ldots, K_n$, such that K_0 is the empty cell complex, K_n is the cell complex K, and K_i is obtained by attaching one or more cells to K_{i-1} for $i = 1..n$.

Note that a *simplex* is a d-cell which has exactly $d+1$ vertices in its vertex set. A *simplicial cell complex* is a cell complex whose cells are d-dimensional simplices such as vertices, edges, triangles, tetrahedra and so on. A simplicial complex is also a regular cell complex.

Given a regular cell complex K representing the domain, a function $f : K \to \mathbb{R}$ is said to be a *discrete Morse function* if for all $d-$cells $\alpha^d \in K$,

$$|\{\beta^{d+1} \mid \alpha^d < \beta^{d+1} \text{ and } f(\beta) \leq f(\alpha)\}| \leq 1 \text{ and}$$
$$|\{\gamma^{d-1} \mid \gamma^{d-1} < \alpha \text{ and } f(\gamma) \geq f(\alpha)\}| \leq 1.$$

A cell α^d is critical if

$$|\{\beta^{d+1} \mid \alpha^d < \beta^{d+1} \text{ and } f(\beta) \leq f(\alpha)\}| = 0 \text{ and}$$
$$|\{\gamma^{d-1} \mid \gamma^{d-1} < \alpha \text{ and } f(\gamma) \geq f(\alpha)\}| = 0$$

A discrete *vector* is a pairing between two incident cells that differ in dimension by one. A *discrete vector field* on K is a set of discrete vectors such that every cell in K is represented in at most one pair of the field. A *V-path* is a sequence of cells

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

such that α_i^d and α_{i+1}^d are facets of β_i^{d+1} and $(\alpha_i^d, \beta_i^{d+1})$ is a vector, $i = 1..r$. A V-path is called a *gradient path* if it contains no cycles (see Figure 3). A *discrete gradient field* is a discrete vector field that contains no non-trivial closed V-paths. We refer to discrete vectors in a discrete gradient path as *gradient pairs*.

Maximal gradient paths of the discrete Morse function correspond to the notion of integral lines of Morse functions. Ascending / descending manifolds are similarly defined for discrete Morse functions.

2.5 Simulation of simplicity

Simulation of simplicity (SoS) is a programming technique that allows us to cope with degenerate data for many geometric algorithms [9]. In the context of Morse-Smale complexes

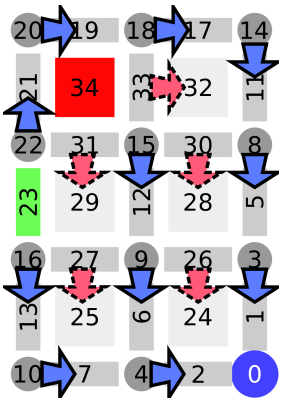


Fig. 3: A discrete Morse function defined over a 2D cell complex. The gradient pairs are shown as arrows oriented towards the higher dimensional cell. Dashed red arrows denote edge-quad pairs and solid blue arrows denote vertex-edge pairs. Critical cells are shown in red (maxima), green (saddle), and blue (minima).

we require that the function be nowhere flat to ensure non-degeneracy. Implementing SoS is simple in this case. We simulate a non-degenerate function using the available order of vertices in the storage device and hence consistently resolve comparisons when function values at two vertices are equal. In the following section we assume that $f(x) \neq f(y)$ for all vertices $x \neq y$. We discuss how to extend this technique to obtain a total order on all input mesh cells in Section 4.

3 Algorithm Overview

In this section, we present an overview of our approach towards the design of a parallel algorithm to compute the MS complex for two-dimensional scalar functions. When the entire dataset fits in memory, the MS complex is computed by a two-step algorithm:

Stage 1. Compute the discrete gradient on the domain.

Stage 2. Compute the combinatorial MS-complex and the geometry of ascending / descending manifolds of critical points.

Section 4 discusses in detail how a discrete gradient field based on the scalar function is computed and how that is used to extract the combinatorial structure of the MS-complex. Specifically, a technique to extend a scalar function sampled at vertices of a regular CW-complex to a totally ordered discrete Morse function is discussed. The definition of this function for a CW-cell relies only on the scalar values of its vertex set. This motivates a massively parallel algorithm to determine discrete gradient pairs (of CW-cells). The extraction of the qualitative structures of the MS-complex using the discrete gradient field and a simple BFS algorithm is then discussed. The BFS algorithm, being serial, does not scale well to massively parallel environments. For this case, we describe an alternate method for traversal which is applicable only to 2D datasets. We corroborate the relevance of the computed MS complex by arguing the closeness of its critical points and gradient pairs to the PL formulation of critical points and gradients.

For large datasets that do not fit in memory, a split and merge strategy is adopted. The gradient computation proceeds without change. The algorithm for large datasets is split into five stages:

Stage 1. Split the domain into sub-domains. Compute the discrete gradient on each sub-domain. Compute the combinatorial MS complex on each sub-domain.

Stage 2. Merge the combinatorial MS-complexes of each of the sub-domains.

Stage 3. Simplify the MS-complex

Stage 4. Traverse the history of merges in reverse order to determine the incidence of geometry of ascending / descending manifolds of critical points that are outside the sub-domain.

Stage 5. Extract the geometry, restricted to the sub-domain, of the ascending / descending manifolds of critical points (that lie possibly outside the sub-domain).

Section 5 discusses the strategy to merge the sub-domain pieces. Essentially, we mark gradient pairs that cross a common boundary as critical. This enables us to identify these pairs as critical points of the MS complex of both sub-domains, perform a merge, and simplify them away. We show that the MS complex is combinatorially and geometrically unaltered by the merge procedure. Furthermore, we show that the order of these cancellations do not alter the resulting MS complex.

Since the number of gradient pairs crossing common boundaries is significant, storing the geometry of the ascending / descending manifolds of these critical points along with the combinatorial connectivity data strains memory requirements. We propose an alternate scheme, whereby we traverse the history of merges in reverse order to infer the geometric contribution of the canceled critical points to surviving critical points. Stage 1 and 5 can proceed in parallel on each sub-domain whilst the other stages merge sub-domains and thus proceed hierarchically.

In both cases of small and large datasets, the algorithm to compute the discrete gradient pairs works for higher dimensional data also. However, for massively parallel environments the subsequent stage of the algorithm, which traverses the gradient field, is restricted to 2D datasets.

4 MS complex algorithm

We now describe our algorithm to compute the MS complex under the assumption that the dataset fits in memory. We first describe a canonical extension of scalar functions sampled at vertices to discrete Morse functions and demonstrate why it is not a suitable extension for computing the MS complex. Next, we introduce a weighted discrete Morse function which satisfies a key property leading to an algorithm that computes gradient pairs in parallel. We discuss how the gradient field defined by the collection of gradient pairs is used to extract the MS complex. Finally, we analyze the computed gradient field and argue for its correctness.

4.1 Discrete function

Given a regular cell complex K with vertex set V and a scalar function $f : V \to \mathbb{R}$, a canonical extension of f to a discrete Morse function, $F_d : K \to \mathbb{R}$, is defined recursively on a cell α as $F_d(\alpha) = \max_{\sigma<\alpha} F_d(\sigma) + \varepsilon$, where $\varepsilon > 0$ is an infinitesimally small real value [11]. Extending the function f in this manner results in all cells becoming critical with

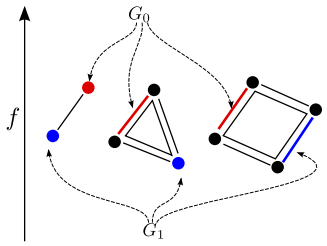


Fig. 4: The weighted discrete function is defined recursively as a weighted sum of the function value at faces G_0 and G_1. Faces G_0 (in red) and G_1 (in blue) for an edge, triangle and quad cells are shown. The function value at vertices increases along the vertical axis.

respect to the discrete Morse function F_d. This implies that each cell in the input is essentially a cell of the MS complex. Further, newly introduced critical cells that are incident on each other can be canceled using an infinitesimally small persistence threshold ε to create an ε-persistent MS complex. The motivation for extending the function f to F_d is that the MS complex can be computed via repeated cancellations of ε-persistent pairs. The collection of ε-persistent critical point pairs are viewed as a pairing of incident cells or discrete gradient pairs. The pairs are represented by arrows from the lower dimensional cell to the higher dimensional cell indicating descent. These pairings constitute a discrete gradient field. Gyulassy et al. [11], [13] compute the MS complex via a sequence of cancellations of the ε-persistent critical point pairs. However, this approach does not necessarily compute paths of steepest descent. Consider the case when two cells, β_1, β_2, share a common facet α such that $F_d(\beta_1)$ and $F_d(\beta_2)$ are written as

$$F_d(\beta_1) = F_d(\alpha) + \varepsilon$$
$$F_d(\beta_2) = F_d(\alpha) + \varepsilon.$$

Either one of β_1 or β_2 can be paired with α. For both pairs, the difference in value of F_d is equal to ε. The tie is broken arbitrarily in this case.

4.2 Weighted discrete function

We now describe a method to extend a given real valued function (f) on the vertex set (V) of a given mesh (K) to a function (F_w) that is defined on all cells of the mesh. We show that this function is a discrete Morse function and that it imposes a total order on the cells. Since the algorithm for computing the MS complex requires only the order between cells, we describe a symbolic comparator that does not explicitly compute the function value. We assume that the input vertices are totally ordered based on the input function specified at the mesh vertices.

4.2.1 Definition of F_w

We define a weighted discrete function F_w on a d-dimensional cell α^d recursively as

$$F_w(\alpha^d) = F_w(G_0(\alpha^d)) + \varepsilon^d \times F_w(G_1(\alpha^d)),$$

where ε is an infinitesimally small positive real number,

$$G_0(\alpha^d) = \underset{\gamma < \alpha^d}{arg\,max} F_w(\gamma), \text{ and}$$

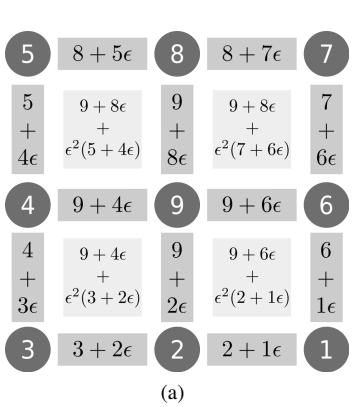


Fig. 5: (a) A scalar function, f, defined on the vertices is recusively extended to a discrete Morse function F_w. The value of F_w is shown for each cell. (b) Gradient pairs determined by algorithm AssignGradient. (c) The combinatorial MS complex computed using a BFS traversal on the gradient field.

$$G_1(\alpha^d) = \underset{\gamma < \alpha^d, V_\gamma \cap V_{G_0(\alpha^d)} = \phi}{arg\,max} F_w(\gamma).$$

$V_{G_0(\alpha)}$ is the vertex set of $G_0(\alpha^d)$, and arg max denotes the value of the argument γ that maximizes the function. Similar to F_d, F_w is also equal to f at mesh vertices. The weighted version of the discrete function ensures that when two cells share a common face whose function value is the maximum among both face sets, then the tie is broken using the second maximum face whose vertex sets are disjoint from the above common face. See Figure 4 for the definition of the weighted discrete function for some common cell types. Figure 5a shows the expansion of F_w for a function sampled on a 2D grid.

$G_0(\alpha^d)$ is necessarily a $d-1$ cell. This is because F_w of any $d-1$ face of α^d is greater than all faces incident on the $d-1$ cell. Thus, the $d-1$ cell that maximizes F_w will have higher function value than all faces of α^d. Also $G_1(\alpha^d)$ must exist for all cells with $d > 0$. Theoretically, we require $F_w(G_1(\alpha^d))$ to be strictly positive to ensure that its value at cofacets is greater than at the facet. This assumption is valid if we rescale the range of f to $[0+\delta, 1]$, $\delta \in (0,1)$. In practice we obtain the order on the cells via a symbolic comparison and do not need to explicitly compute F_w.

4.2.2 F_w is well defined and totally ordering

For F_w to be well defined we require G_0, G_1 to be unique. The cells $G_0(\alpha^d)$ and $G_1(\alpha^d)$ for a given α^d are unique if F_w induces a total order on all cells of dimension less than d. This suggests an inductive proof, which we outline below.

Since $f(x) \neq f(y)$ for all $x \neq y$, F_w is well defined and induces a total order on all zero-dimensional cells. Now, assume that F_w induces a total order on all cells of dimension less than d. We will show that we can order two cells α_1^d, α_2^d or $\alpha_1^d, \alpha_2^{d'}$ where $d' < d$ and $\alpha_1 \neq \alpha_2$.

Let $G_0(\alpha_1^d) = \gamma_1$ and $G_1(\alpha_2^d) = \gamma_2$. We have $F_w(\alpha_1^d) = F_w(\gamma_1) + \varepsilon^d \times F_w(G_1(\alpha_1^d))$ and $F_w(\alpha_2^d) = F_w(\gamma_2) + \varepsilon^d \times F_w(G_1(\alpha_2^d))$. Assume that $\gamma_1 \neq \gamma_2$. Cells γ_1 and γ_2 have dimension less than d and can therefore be ordered. We

choose ε to be arbitrarily small, so that the comparison of $F_w(\alpha_1^d)$ and $F(\alpha_2^d)$ is dominated by the comparison of $F(\gamma_1)$ and $F(\gamma_2)$.i.e. $F_w(\gamma_1) < F_w(\gamma_2) \Rightarrow F_w(\alpha_1) < F_w(\alpha_2)$ and $F_w(\gamma_1) > F_w(\gamma_2) \Rightarrow F_w(\alpha_1) > F_w(\alpha_2)$.

If $\gamma_1 = \gamma_2$ then the second term induces an order on α_1 and α_2. Note that if $\alpha_1^d \neq \alpha_2^d$ then $G_1(\alpha_1^d) \neq G_1(\alpha_2^d)$ because K is a cell complex. This is because if two d-dimensional cells intersect they do so along a single common cell whose dimension is less than d. The cells α_1^d and $\alpha_2^{d'}$ can be ordered using a similar argument. Thus, the weighted discrete function F_w is well defined and induces a total order.

4.2.3 Symbolic comparison of cells

We essentially require only an ordering of cells in K and not the explicit values of F_w. We now describe a method to establish this order using comparisons.

The value of $F_w(\alpha)$ is equal to the weighted sum of F_w at a subset of the vertices of α. Further, since $G_0(\alpha)$ and $G_1(\alpha)$ are also vertex disjoint, no vertex appears more than once in the above sum. Replace the coefficient of $F_w(G_1(\alpha))$ to $\varepsilon^{T(G_1(\alpha))}$, where $T(\alpha)$ is equal to the number of terms in the weighted sum of $F_w(G_1(\alpha))$. It can be easily verified that F_w remains well defined and induces the same total order. This is because $T(\alpha^d) \geq d$. The function $F_w(\alpha)$ is therefore expressed as a weighted combination of the function values at a subset of V_α. Thus, the ordering of cells follows from a lexicographical ordering where a cell α is represented as an ordered list of vertices, S_α, which is a subset of the vertex set, V_α.

In the case of simplicial complexes, this ordering is equal to the sorted order of all vertices. For quad cells in a rectilinear mesh, it is the ordered vertices of the edge with highest value of F_w followed by the ordered vertices of the edge disjoint from the first edge. This ordering of vertices is a specific permutation of the vertex set. Similarly the ordering for a three dimensional cube mesh is a specific permutation of the vertex set. Note that it is not necessary for the set S_α to contain all vertices of V_α. For example, the size of S_α will remain four for a hexagon cell in a hexagonal tessellation of the two dimensional plane whereas α contains six vertices.

4.3 Computing gradient pairs

We now outline our algorithm that computes gradient pairs using the comparator based weighted discrete function defined above. We prove that the pairs found by the algorithm are unique and independent of the order in which the cells are considered, thus providing scope for parallelizing the algorithm.

Algorithm 1 ASSIGNGRADIENT (Cell complex K)

```
1: for all α ∈ K do
2:     P_α = {β | α ≺ β and α = G_0(β)}
3:     if P_α ≠ φ then
4:         β = Min_F(P_α)
5:         pair_cells (α,β)
```

In the above algorithm, α denotes a cell in the complex K, and β is a cofacet of α, denoted by $\alpha \prec \beta$. The set P_α is the collection of cofacets, β, of α such that $\alpha = G_0(\beta)$. In other words, P_α is the set of cofacets of α where α is the facet with the maximum value of F_w. Figure 5b shows the gradient field determined by the algorithm ASSIGNGRADIENT for the function in Figure 5a.

ORDER INDEPENDENT PAIRING LEMMA. *The pairing determined by the algorithm* ASSIGNGRADIENT *is independent of the order in which cells are processed. In particular, if a cell α pairs with its cofacet β then β will not pair with any of its cofacets.*

Proof: Inconsistencies occur if the algorithm determines two or more pairs for the same cell. A cell present in two pairings can be of the nature (α,β),(α,β') or (α',β), (α,β) or (γ,α), (α,β) where $\gamma \prec \alpha \prec \beta$.

This first conflict is trivially not possible because for a cell α we determine a unique pair from a set of candidate facets. In the second case, if β were to be paired with two different facets, α and α', then $\beta \in P_\alpha, P_{\alpha'}$. But, from the definition of P_α we know that $G_0(\beta)$ is unique and equal to either α or α'. Therefore, β must either belong to P_α or to $P_{\alpha'}$ but not both. So, β is paired either with α or with α'.

To prove that the third conflict does not arise, we show that if α pairs with one of its cofacets β, then α is not the lowest pairable cofacet of any of its facets *i.e.* $\beta = Min_F(P_\alpha)$ implies $\alpha \neq Min_F(P_\gamma)$ for all $\gamma \prec \alpha$. This will imply that if α paired with β, then it is not paired with any other cell γ. Consider a facet γ of α, $\gamma \prec \alpha$. If $\alpha \notin P_\gamma$ then there is nothing to prove because the algorithm will not pair γ with α. Now assume $\alpha \in P_\gamma$. For a regular cell complex, if γ is a face of a cell β such that $dim(\gamma) = dim(\beta) - 2$, then there exists exactly two cells σ_1, σ_2 such that $\gamma \prec \sigma_1 \prec \beta$ and $\gamma \prec \sigma_2 \prec \beta$. Without loss of generality, we relabel σ_1, σ_2 as α, α'. Since (α,β) form a pair and not (α',β), we have $F_w(\alpha') < F_w(\alpha)$. Hence, it is sufficient to show that $\alpha' \in P_\gamma$.

Assume that $\alpha' \notin P_\gamma$. There exists $\gamma' \neq \gamma \in K$ such that $\gamma' \prec \alpha'$ and $\alpha' \in P_{\gamma'}$. This implies $F_w(\gamma') > F_w(\gamma)$. Since $F_w(\alpha) = F_w(\gamma) + \varepsilon$ and $F_w(\alpha') = F_w(\gamma') + \varepsilon$ we have $F_w(\alpha) < F_w(\alpha')$. This is a contradiction. Hence, we have $\alpha' \in P_\gamma$ and $F_w(\alpha') < F_w(\alpha)$. So, if (α,β) is a pair then $\alpha \neq Min_F(P_\gamma)$ for any $\gamma < \alpha$, which implies that there is no such pair (γ,α). □

4.4 Computing the MS complex

Once the discrete gradient field is computed, the descending / ascending manifolds and the combinatorial MS complex are extracted as a collection of gradient paths. The descending manifold of a critical point is equal to the closure of all gradient paths that originate from that critical point. This is computed using a breadth first traversal of gradient pairs beginning from the critical point. The ascending manifold is the closure of the set of gradient paths that terminate at a given critical point. This is computed using a breadth first traversal of reversed gradient pairs beginning from the critical point. A combinatorial connection between any two critical cells is established if there is a gradient path that connects them. Figure 5c shows the combinatorial MS complex extracted from the gradient field shown in Figure 5b. For multicore

environments, multiple BFS traversals from critical points are launched. The number of parallel BFS traversals launches usually depends on the number of cores.

Since the BFS algorithm is essentially a serial algorithm we adopt a different strategy in the case of massively parallel environments. This strategy is applicable for 2D discrete gradient fields. 2D discrete gradient paths that originate at maxima split but do not merge and discrete gradient paths that terminate at minima merge but do not split [5]. Thus, every gradient pair on a path from a maximum is immediately preceded by a unique source which is either another gradient pair or the maximum. Similarly, every pair on a path to a minimum is succeeded by a unique destination which is either another gradient pair or the minimum. The traversal is now posed as an iterative search for the source/destination extremum of every gradient pair. For completeness, maxima are their own source and minima are their own destination. Each work item (thread) is mapped to iteratively determine the eventual source/destination of a gradient pair. At every iteration the source of gradient pairs that are on gradient paths originating from a unique maximum is updated to the source of its source. Similarly, the destination of gradient pairs that are on gradient paths terminating at a unique minimum is updated to the destination of its destination. The iterations stop when all pairs find their unique source or destination. For a path of length n, the first iteration updates each node's source to the gradient pair at a distance two. The next iteration updates it to the gradient pair at a distance four. Thus, the process terminates in $log_2(n)$ steps. Though the worst case asymptotic complexity of this traversal is $n\log_2(n)$, in practice we observe that traversal requires $\log_2(n)$ time due to the parallelization. The combinatorial MS complex is computed by querying the source/destination of gradient paths that originate/terminate at facets/cofacets of saddles. The geometry of extrema is available as a disjoint set of trees rooted at them. However, the geometry of saddles is not directly available and is extracted by serial BFS traversals.

4.5 Analysis and Correctness

In this section, we argue for the correctness of the MS complex computed by our algorithm. Specifically we show that the computed critical points and gradient pairs are close to those of the PL function.

4.5.1 *Closeness of critical cells to PL critical points*

The weighted discrete function $F_w(\alpha^d)$ is defined recursively. In order to obtain a simple expression, we introduce $G_0^i(\alpha^d)$ that allow us to unravel the definition of $F_w(\alpha^d)$ up to i levels of recursion. Let $G_0^i(\alpha^d)$ denote the G_0 function applied i (≥ 0) times on a cell α^d. For example, $G_0^2(\alpha^d) = G_0(G_0(\alpha^d)), G_0^0(\alpha^d) = \alpha^d$. Define ε-lower star of the vertex v as the set of cells σ^d such that $v = G_0^d(\sigma)$:

$$\varepsilon LST(v) = \{\sigma^d \in K \mid v = G_0^d(\sigma^d)\}$$

We note that if K is a simplicial complex, v is a vertex and the function is a PL extension of samples at the vertices, then the ε-lower star of v is exactly the lower star of v (See Figure 6a).

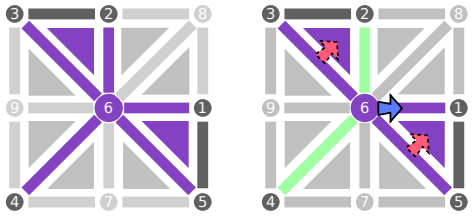


Fig. 6: (a)εLST (cells in purple) and εLLK (cells in dark gray) of a vertex, for a function sampled at the vertices. Other cells are shown in light gray. (b) Gradient vector pairs and critical cells determined by the algorithm AssignGradient.

Similarly, define ε-lower link (εLLK) of a vertex v to be the set of faces of cells in $\varepsilon LST(v)$ that are not incident on v. We show that any gradient algorithm that pairs cells within the ε-lower star of a vertex v must retain at least $\tilde{\beta}_{k-1}$ index k critical cells in $\varepsilon LST(v)$, where $\tilde{\beta}_k$ is the reduced Betti number of $\varepsilon LLK(v)$.

We first claim that the filtration K induced by attaching cells in increasing order of F_w is a valid. Furthermore, we claim that the cells in $\varepsilon LST(v)$ are ordered contiguously by F_w. The first part of the claim is true because faces of a cell always have function value lower than that of the cell (by definition of F_w) and therefore appear before the cell in the ordering. For the second part, consider any $\gamma^d \notin \varepsilon LST(v)$. We can express $F_w(\gamma^d)$ as

$$F_w(\gamma^d) = F_w(G_0^d(\gamma^d)) + \sum_{i=1}^{d} \varepsilon^i \times F_w(G_1(G_0^d(\gamma^d)))$$

by successively rewriting the leading term. Since $\gamma^d \notin \varepsilon LST(v)$ we have that $G_0^d(\gamma^d) \neq v$. By writing the expression for F_w for all cells $\sigma^{d'} \in \varepsilon LST(v)$ in the above form, the comparison of γ^d and $\sigma^{d'}$ will be dominated by the comparison of cells $G_0^d(\gamma^d)$ and $G_0^{d'}(\sigma^{d'})$. Hence γ^d would precede or succeed all cells of $\varepsilon LST(v)$.

Next we observe that algorithm AssignGradient pairs cells within the εLST of a vertex v, *i.e.* if (α^d, σ^{d+1}) is a pair then both α^d and σ^{d+1} belong to the ε-lower star of some vertex v and no other vertex v'. This follows from the definition of εLST and G_0^i. Thus the same pairs are determined for a given ε-lower star attached to a given ε-lower link regardless of other cells in the cell complex.

Consider the hypothetical situation where a vertex v' precedes v in the filtration such that $\varepsilon LST(v')$ is a duplicate of the $\varepsilon LST(v)$ attached to $\varepsilon LLK(v)$. We will relate the reduced Betti numbers of $\varepsilon LLK(v)$ to the increase in the Betti numbers of the complex after attaching v and its ε-lower star (See Figure 7). Let $K_{v'}$ denote the cell complex obtained after attaching v' and its ε-lower star. Since the gradient pairs are determined within the $\varepsilon LST(v)$, they are not affected by gradient pairing in the rest of the complex. Assume that the gradient field is optimal in the sense that the number of critical points of index k (n_k) is exactly the same as the k^{th} Betti number (β_k). In this scenario the net effect of attaching $\varepsilon LST(v)$ is the creation of $\tilde{\beta}_k$ $(k+1)$ cycles. For example if $\tilde{\beta}_{-1}(\varepsilon LLK(v)) = 1$, attaching $\varepsilon LST(v)$ would create a new component. In other words it increases

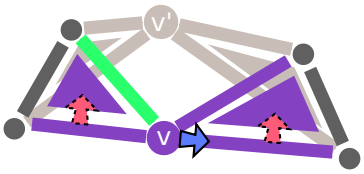


Fig. 7: A cell complex where a duplicate of $\varepsilon LST(v)$ precedes v in the filtration and is attached to $\varepsilon LLK(v)$.

the β_0 of $K_{v'}$ by one. Similarly if $\tilde{\beta}_0(\varepsilon LLK(v)) = c$, attaching $\varepsilon LST(v)$ would increase β_1 of $K_{v'}$ by c. If $\tilde{\beta}_1(\varepsilon LLK(v)) = 1$, attaching $\varepsilon LST(v)$ would increase β_1 of $K_{v'}$ by 1. Thus attaching $\varepsilon LST(v)$ causes an increase in β_k of $K_{v'}$ by $\tilde{\beta}_{k-1}$. Since the gradient field was optimal before $\varepsilon LST(v)$ was attached, n_k should increase by at least $\tilde{\beta}_{k-1}$ to satisfy the weak Morse inequality ($n_k \geq \beta_k$). Since the only new cells were that of $\varepsilon LST(v)$, the new critical points must be present within the $\varepsilon LST(v)$.

This result shows that PL critical points are approximated by a critical cell incident on the PL critical vertex. Furthermore multi-saddles are also approximated with the appropriate number of critical cells.

4.5.2 Steepest descent

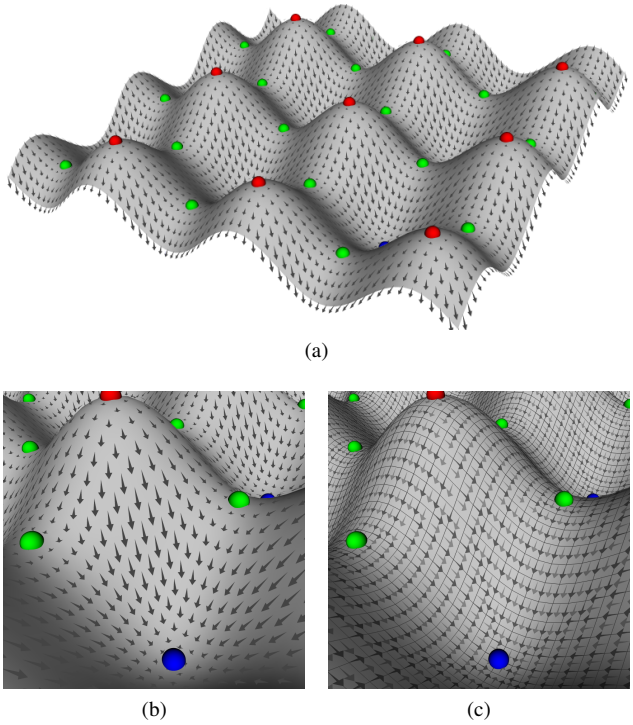
Consider a PL function defined on a simplicial complex whose function value at vertices is known. The gradient pairing algorithm will attempt to pair a cell α^d with a cell σ^{d+1}, where σ^{d+1} is a simplex formed by adding a vertex to V_α and the new vertex has function value lesser than all vertices in V_α. For every point on α^d, the gradient of the PL interpolant is oriented towards the new vertex. Hence the gradient lines originating from the interior of α^d, are oriented towards the interior of σ^{d+1}. Because of the discontinuity of gradients of PL interpolants on cells that are shared, the gradient algorithm will pair the $d+1$-cell attached to α^d with minimum function value. This will be the $d+1$-cell attached to α^d with minimum function value on the vertex not present in V_α, therefore maximizing the magnitude of the gradient. Hence the gradient vector pairing agrees with the maximal PL gradient on a simplicial complex.

In the case of two dimensional rectilinear grids using a bilinear interpolant it is seen that the same argument applies except for the case when the quad contains a face saddle. In this case we see that gradient at the mid point of the maximal edge has steepest descent gradient towards the quad element.

Figure 8 shows the comparison of the continuous gradient of the analytic function $sin(x)+sin(y)$ evaluated at the vertices of the two dimensional rectilinear grid, with the discrete gradient computed on the grid using the gradient algorithm. The discrete gradient pair arrow are aligned along edges for vertex-edge pairs and orthogonal to edges for edge-quad pairs. In both cases, they agree with the gradients computed for the analytic function at mesh vertices.

5 Out-of-core algorithm

We now discuss the computation of the MS complex of 2D scalar functions with a focus on large datasets that do not fit entirely in memory. The computation is done in five stages (see



(a) (b) (c)

Fig. 8: (a) Gradient field of the function $sin(x) + sin(y)$ evaluated at mesh vertices. (b) Close up view of the gradient field. (c) Discrete gradient vectors for function sampled at vertices.

Figure 9). The data is first hierarchically partitioned into sub-domains blocks. The partitioning stops when the sub-domains are small enough to fit in memory.

5.1 Gradient and MS Complex on sub-domains

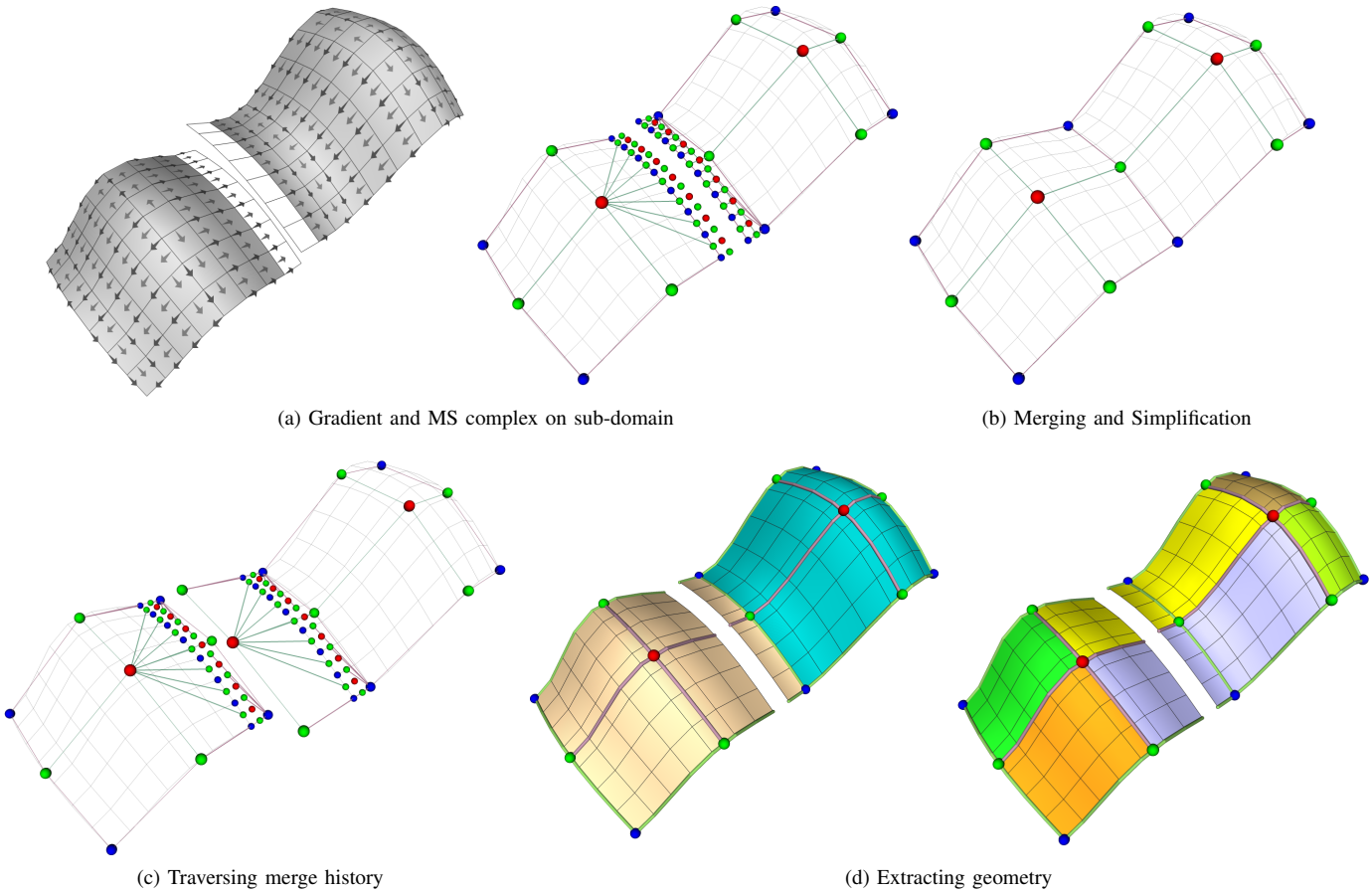
The computation of the gradient proceeds as outlined in the previous section. To obtain a equivalent gradient field on a subdomin, the gradient algorithm needs only a cell's cofacets and their facets in the domain. The cell complex of the sub-domain is extended to include the set of cells that are incident on the shared boundary of sub-domains and gradient is computed only on the initial sub-domain cell complex (see Figure 9a). Thus, we obtain identical pairings for cells along the shared boundary when we process all sub-domains that share the boundary cell.

To facilitate merging we mark all gradient pairs that cross a shared boundary as critical (see Figure 9a). We establish the validity of this step in the following section.

5.2 Merging sub-domain MS complexes

Next, we merge the sub-domains in a bottom up fashion by identifying boundary critical point pairs and canceling them when they enter the interior of the union. The cancellation repeatedly merges the MS complex across the sub-domains till we obtain the MS complex of the input function.

We first establish the equivalence of gradient paths and the paths computed by a sequence of cancellations. A consequence of this result is that we can process the sub-domains in parallel



(a) Gradient and MS complex on sub-domain

(b) Merging and Simplification

(c) Traversing merge history

(d) Extracting geometry

Fig. 9: MS complex for large domains is computed in five stages. Data is first split into sub-domains. (a) Gradient is computed on sub-domains. Unpaired cells and gradient pairs incident on shared boundary are marked critical. Combinatorial MS complex on each sub-domain is computed. (b) The combinatorial MS complex of the domain is computed by identifying and canceling gradient pairs incident on the shared boundary. (c) The history of merge cancellations is traversed to reveal the incidence of critical cells across sub-domains. This information is used to trace the geometry of the cells of the MS complex. (d) For each sub-domain, the geometry of the descending and ascending manifold of an incident critical cell restricted to the sub-domain is extracted.

and later merge them to obtain the MS complex while ensuring combinatorial and geometric equivalence.

ORDER INDEPENDENT CANCELLATION LEMMA. *Let $p, \alpha_0, \sigma_0, \ldots, \alpha_i, \sigma_i, \ldots, \alpha_k, \sigma_k, q$ denote a gradient path between two critical points p and q. This gradient path is faithfully traced independent of the scheduled order of boundary critical point pair cancellations.*

Proof: In the above gradient path, canceling pair α_i, σ_i results in establishing the connectivity between $\sigma_{i-1}, \alpha_{i+1}$. Iterating forward, we see that cancellation of any pair along the gradient path successively establishes connectivity between the preceding and succeeding surviving critical point. Eventually the critical points p, q are connected by an arc. Thus combinatorially, this is equivalent to the MS complex obtained without by tracing a path directly from p or q without any intermediate step of creating boundary critical points. The same argument extends to prove the resulting geometric equivalence. □

As a consequence of the above lemma, we can schedule cancellations of boundary critical point pairs in any order. Gyulassy et al. [11] also employ a divide and conquer approach to compute the MS complex. However, they partition the domain into "parcels" that do not share common boundary. The merge step, therefore, has to process new cells and may introduce new critical points. Hence, they are not able to ensure the geometric equivalence of the MS complex. Our partitioning scheme is the central reason for the Order Independent Cancellation Lemma to be true.

5.3 History Tree

One of the implications of declaring all boundary cells and their outgoing / incoming pairs as critical is the creation of a large number of critical cells. Since the merge operation involves cancellation of critical points, the ascending and descending manifolds need to be computed and merged. However the number of cells that are present in the ascending / descending manifold of a critical point is $O(n)$, where n is the number of cells in the cell complex. This leads to a

large memory foot print of intermediate complexes.

The artificial critical points represent regions through which flow enters / leaves a sub-domain. Therefore, recording the combinatorial connectivity to a surviving critical point at the boundary is sufficient to compute the ascending/descending manifold restricted to the sub-domain. We record this information during the merge step and are therefore able to compute the 1-skeleton of the MS complex with a small memory footprint. The recorded combinatorial connectivity between boundary critical points is used later to extract the geometry of the gradient paths. We now describe how we traverse the history of cancellations to compute the geometry of the arcs.

Consider a series of k cancellations to determine the combinatorial connection between two critical points p^i and q^{i-1}. The series of canceled critical point pairs is equal to the gradient path connecting the two critical points:

$$p, \ldots, \alpha_{k-1}^{i-1}, \sigma_{k-1}^{i}, \ldots, \alpha_{k}^{i-1}, \sigma_{k}^{i}, \ldots, \alpha_{k-2}^{i-1}, \sigma_{k-2}^{i}, \ldots, q$$

Consider the final cancellation that determines the connection between p and q. Before cancellation, p is contained in ascending connections of α_k and q is contained in the descending connections of σ_k. Before the cancellation of the $(k-1)^{th}$ pair, α_k^{i-1} is connected to σ_{k-1}^{i}. By retaining this information, after the k^{th} cancellation we can infer that σ_{k-1} is connected to all surviving critical points in the descending connections of α_k's pair. Extending this further to previous cancellations, we see that if we traverse the critical point pairs in reverse order of their cancellations, we can infer the entire geometry of the gradient path. This is accomplished by traversing the history tree, which records all merges, in a top-down manner. At the leaf of the history tree, we obtain the combinatorial connections from the BFS traversal within the sub-domain.

5.4 Geometry extraction

The history tree traversal returns the points of entry and exit of all critical cells that have gradient entering or leaving the sub-domain. Thus the geometry of the descending/ascending manifold of a critical cell restricted to the sub-domain can be computed by tracking the gradient from the cells of entry/exit that are on shared boundaries. If the critical cell is contained in the sub-domain then the geometry is computed as indicated in the first stage.

6 Implementation

In this section we briefly outline our experimental setup for the various stages of our algorithm. We proceed with two setups, one leveraging multicore architectures and another targeted at massively parallel architectures, namely the GPU. We use the OpenCL framework for programming the GPU. We report the implementation and results for rectilinear two-dimensional grids.

6.1 Gradient Pairs and MS complex on sub-domains

Since we work with grid domains, we use the centroids of cells to represent them. We scale the cell identifiers by two so that they are integral values. Therefore cell information is maintained as two-dimensional buffers with the same size as that of the domain. This simplifies queries for facets / cofacets, which can now be computed using arithmetic operations with boundary conditions.

For the gradient information we require one buffer to store the pair of the cell and a flag buffer to store whether the cell is critical or paired or both in the case of boundary critical pairs. While working with the GPU for the gradient assignment we need to mirror these buffers in both the CPU and GPU . This is required only for geometry extraction in the final stage of our algorithm. The counting and collection of critical points from the flag buffers is posed as the parallel prefix sum problem [2], [3]. The prefix scan implementations have asymptotic complexity of $O(nlog_2(n))$ but in practice we observe that traversal requires $log_2(n)$ time due to the parallelization.

For the CPU implementation, the standard BFS algorithm is used considering cells as nodes and edges between cells if they are pairs or if they are adjacent on a gradient path. For the GPU implementation, we adopt the iterative source/destination search described in Section 4.4. Since there is an issue of concurrent updates, we use two buffers to store the source/destination information. Each iteration reads the source/destination information from one buffer and updates it to the second buffer. In the next iteration the roles of the buffers are swapped. A global boolean is initialized to $false$ and set to $true$ if a cell updates its buffer.

6.2 Merging

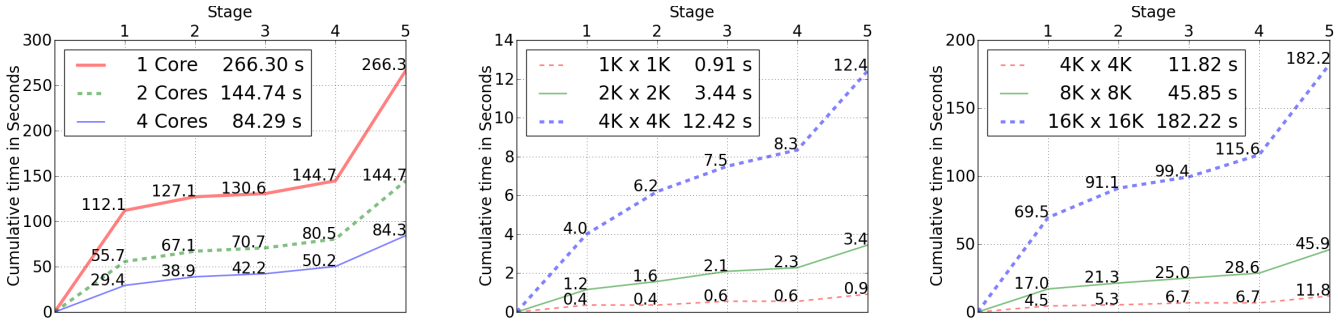
To enable stream processing of sub-domains we recursively divide the domain along a single axis. The desired level of subdivision is adjusted to accommodate the largest possible sub-domain within memory (GPU memory in the case of the GPU implementation). The recursive subdivision leads to a hierarchical structure with 2^d sub-domains, where d is the depth of the recursion, and $2^d - 1$ intermediate nodes that represent the hierarchy. Merging of intermediate nodes in each level can be done in parallel.

6.3 Simplification

We perform a persistence based simplification of the final MS complex. The simplification affects the MS complex computed for each sub-domain. The MS complex of a sub-domain is updated by identifying surviving critical points, deactivating them, and introducing new critical points that may have become incident on the sub-domain. Since simplification by persistence does not require any geometry computation, we simplify before we traverse the history tree and push the results down the tree.

6.4 History tree

The history tree that records the merges is traversed to compute the incidence of surviving critical points on sub-domain boundary. Because of the hierarchical decomposition, the traversal can be done in parallel for all nodes within a level.

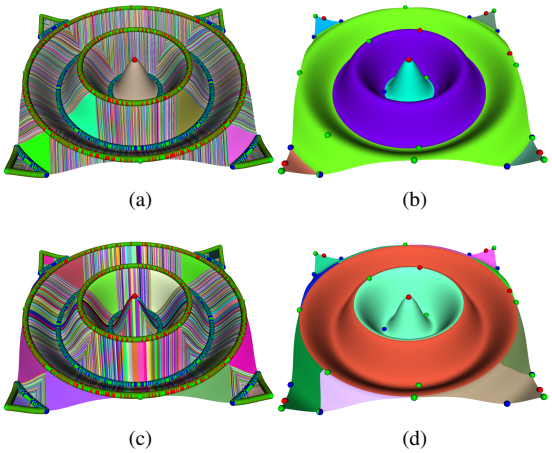


(a) CPU out-of-core (b) GPU in-core (c) GPU out-of-core

Fig. 10: Time required to compute the MS complex for the `wgauss` dataset cumulated over the five stages of the algorithm. (a) The 8192×8192 data does not fit in CPU memory. (b) Data fits in CPU but not GPU memory. (c) Data fits neither in CPU nor GPU memory.

6.5 Fast Geometry Queries

Once we know the combinatorial structure of the MS complex at the boundary of a sub-domain, the computation of descending and ascending manifolds is essentially a traversal of gradient paths from these entry and exit points along with the paths that originate from or terminate at the critical point. In our implementation, we track only the surviving saddle points, because maxima partition the diverging gradient flows and minima partition the converging gradient flows. In our experiments we recompute the gradients because we found that the disk latency involved in storing the gradient and retrieving them later is costlier. This is because, recomputing the gradient requires only a single read of the function values at the vertices.



(a) (b) (c) (d)

Fig. 11: (a),(c) The full resolution descending and ascending Morse complex for the `wgauss` dataset for a grid size of 1024×1024. (b),(d) The simplified descending and ascending Morse complex simplified upto 10%. As expected the descending manifolds partition the domain into regions that correspond to peaks and the ascending manifold partition the domain to regions that correspond to valleys.

7 Experimental Results

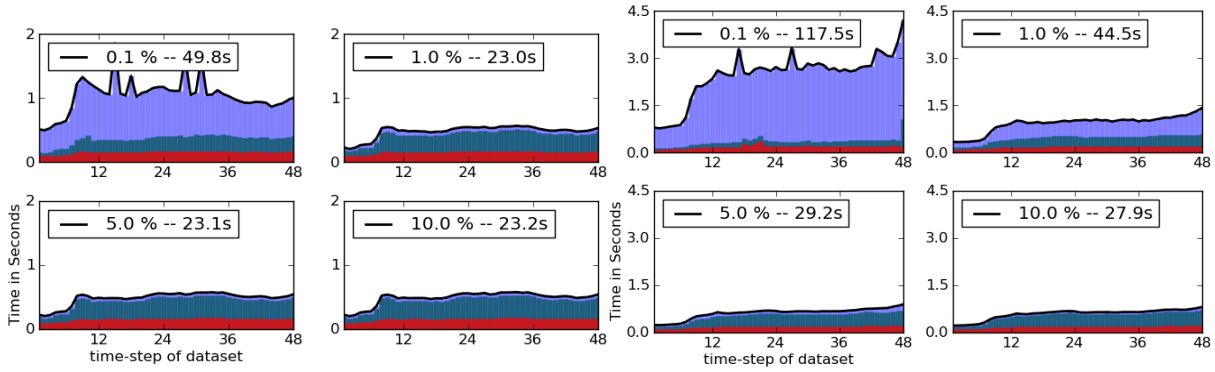
We now present results of our experiments on both synthetic and the hurricane Isabel data set from the Vis 2004 contest [26]. All experiments were performed on a workstation with two Intel Xeon quad core processors, 8GB RAM, and nVidia GeForce 260 GTX graphics card which has 196 cores and 896MB RAM. The first synthetic data set `sine` is a sinusoidal function sampled over a rectilinear grid. The second synthetic data set `wgauss` is a 2D Gaussian distribution centered at the origin and weighted by a radially decreasing sinusoidal curve. The `wgauss` dataset contains large number of critical points and degenerate regions which help to stress test our algorithm. We study the performance and scalability of our algorithm using these two synthetic data sets.

Figure 10a shows the speed up obtained for `wgauss` sampled on an 8192×8192 grid for varying number of processors using the CPU implementation. Time is cumulated over the five stages of the algorithm. The data is processed out-of-CPU-core (not all data is present in CPU memory) to conserve memory. The graphs indicate near linear scaling with the number of cores. We observed a similar execution profile for the sinusoidal dataset with 16384×16384 data points. The MS complex was was computed in 3 minutes and 6 seconds.

To study the scalability of the algorithm with input sizes we conducted experiments with the `wgauss` dataset computed on various grid sizes. Figure 10b shows results from the GPU execution for the `wgauss` for varying grid sizes and the corresponding speed up. Here the data is resident in the CPU memory.

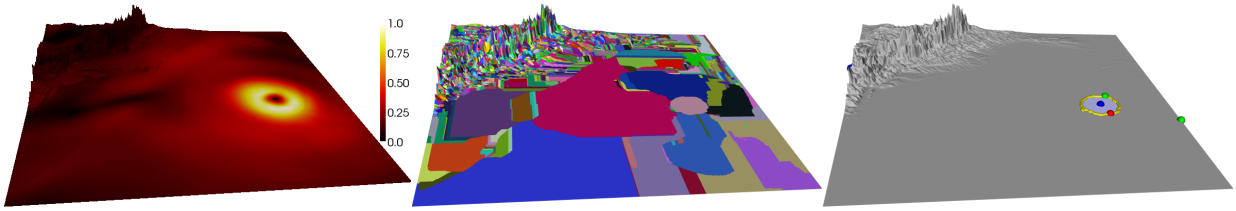
Figure 10c shows results from an out-of-CPU-core execution on `wgauss` using the GPU for varying domain sizes. The size of the sub-domains is restricted to contain 1 million points. Figure 11 shows the full resolution and simplified descending and ascending Morse complex of the `wgauss` dataset with a grid size of 1024×1024.

Hurricane Isabel was a strong hurricane that struck the west Atlantic region in September 2003. We consider a simulation of this event [26]. The domain is a 3D rectilinear grid of size $500 \times 500 \times 100$ available over 48 time steps. We extract a 500×500 grid representing the land/sea surface to study the pressure (Pf), temperature (TCf) and magnitude of wind



(a) Pressure (b) Wind speed

Fig. 12: Time taken for computing the MS complex for all time-steps for simplification thresholds of 0.1%, 1%, 5% and 10% for (a) Pressure and (b) Magnitude of wind velocity fields. Time taken for stages one, three and five are shown in the breakup along y-axis. Stages two and four are not present since the data for each field of each time-step fits in GPU memory. Time taken for geometry extraction in stage five reduces drastically if the MS complex is simplified.



(a) (b) (c)

Fig. 13: (a) The wind speed field of the 1^{st} time step over the surface (function normalized to $[0,1]$). (b) The full resolution ascending Morse complex (c) The simplified MS complex retains significant critical points. The most persistent minimum corresponds to the eye of the hurricane.

velocity fields over time. We compute the MS complex for all three scalar fields in each time step using our parallel algorithm and track significant features in the data. Figure 12 shows the execution profile, along with the stage wise breakup of time, for the pressure and magnitude of wind velocity fields, for various simplification thresholds. Since the data in each field of each time-step fits in GPU memory, the merge and history tree traversal stages are not present. We observed that the time required for computation of the MS complex for most time steps was below 0.5 seconds. Without simplification, the time required to compute the MS complex increased up to 6 seconds. However, it dropped below 0.5 seconds for several time steps once we simplified critical pairs below a 0.1% persistence threshold.

Figure 13 shows the decomposition of the domain into ascending manifolds of the critical points of wind speed. Our implementation supports the interactive extraction of these manifolds using a parallel algorithm. We simplify the wind speed field within each time step to identify significant features after removing all the small features. Figure 1 in the appendix shows the result of this experiment using the wind speed, where we track the ascending manifold of the most persistent minimum corresponding to the eye of the hurricane. Currently, we are able to process each time-step of the speed within 0.5 seconds for simplification threshold of above 5%, thereby supporting interactive analysis of the data. With additional optimizations we hope to be able to further reduce the processing time and hence enable real-time analysis and feature tracking on larger time-varying data.

8 Conclusions

We have described the first parallel algorithm to compute MS complexes. Our approach is based on two key lemmas on the gradient-based pairing of cells and critical pair cancellations. The algorithm performs well on both multicore environments like the CPU and massively parallel architectures like the GPU. We describe fast methods for querying the MS complex, in particular to extract the ascending / descending manifolds of a query critical point.

In future, we plan to extend our implementation to three-dimensional grid and unstructured meshes. The algorithm can be implemented for multicore CPUs without change. For GPUs, the key challenge is the design of an efficient parallel BFS procedure. Unlike 2D discrete gradient fields that decompose into trees (Section 4.4), 3D gradient fields do not. Specifically the discrete gradient field restricted to paths that originate at 2-saddles or terminate at 1-saddles is, in general, a directed acrylic graph whereas discrete gradient paths that originate/terminate at extrema form a tree. We

believe a synergistic approach, using both the CPU and the GPU should yield good performance.

Our algorithm does not implicitly handle noisy data. This can be a problem in case of large datasets because of the large number of low persistence critical points which will have to be tracked through the various stages of the algorithm. For large noisy datasets, we advocate the approach adopted by Gyulassy et al[11], where a first round of simplification is done to eliminate low persistence critical point pairs within sub-domains before they are merged.

The development of fast parallel methods for other tasks like creation of a multi-resolution representation and segmentation using the MS complex are also interesting open problems.

ACKNOWLEDGMENTS

This work was supported by grants from Intel and Department of Science and Technology, India (SR/S3/EECE/048/2007). Hurricane Isabel data produced by the Weather Research and Forecast (WRF) model, courtesy of NCAR and the U.S. National Science Foundation (NSF).

REFERENCES

[1] U. Bauer, C. Lange, and M. Wardetzky. Optimal topological simplification of discrete functions on surfaces. *Discrete and Computational Geometry*, pages 131, April 2011.

[2] G. E. Blelloch. Scans as primitive parallel operations. *IEEE Transactions on Computers*, 38:15261538, 1989.

[3] G. E. Blelloch. Prefix sums and their applications. In John H. Reif, editor, *Synthesis of Parallel Algorithms*. Morgan Kaufmann, 1991.

[4] Peer-Timo Bremer, Herbert Edelsbrunner, Bernd Hamann, and Valerio Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385 396, 2004.

[5] F. Cazals, F. Chazal, and T. Lewiner. Molecular shape analysis based upon the Morse-Smale complex and the Connolly function. In *Proc. 19th Ann. ACM Sympos. Comput. Geom.*, pages 351360, 2003.

[6] 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 361370, 2003.

[7] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse- Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry*, 30(1):87107, 2003.

[8] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28(4):511533, 2002.

[9] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66104, 1990.

[10] R. Forman. A users guide to discrete Morse theory. *Seminaire Lotharingien de Combinatoire*, 48, 2002.

[11] A. Gyulassy, P. T. Bremer, V. Pascucci, and B. Hamann. A practical approach to Morse-Smale complex computation: scalability and generality. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):16191626, 2008.

[12] A. Gyulassy, M. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically clean distance fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):14321439, 2007.

[13] 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*, 12(4):474484, 2006.

[14] 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):14401447, 2007.

[15] A. Hatcher. Algebraic Topology. *Cambridge University* Press, 2001.

[16] H. King, K. Knudson, and N. Mramor. Generating discrete morse functions from point data. *Experiment. Math.*, 14(4):435444, 2005.

[17] D. Laney, P. T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):10531060, 2006.

[18] T. Lewiner, H. Lopes, and G. Tavares. Applications of formans discrete Morse theory to topology visualization and mesh compression. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):499508, 2004.

[19] Y. Matsumoto. An Introduction to Morse Theory. *Amer. Math. Soc.*, 2002. Translated from Japanese by K. Hudson and M. Saito.

[20] J R Munkres. Elements of Algebraic Topology. *Addison-Wesley*, 1984.

[21] J. Reininghaus and I. Hotz. Combinatorial 2d vector field topology extraction and simplification. In *Topological Methods in Data Analysis and Visualization. Theory, Algorithms, and Applications. (TopoInVis09)*, 2010.

[22] J. Reininghaus, C. Löowen, and I. Hotz. Fast combinatorial vector field topology. *IEEE Transactions on Computer Graphics and Visualization*, in press, 2010.

[23] V. Robins, P. J. Wood, and A. P. Sheppard. Theory and algorithms for constructing discrete morse complexes from grayscale digital images. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 99(PrePrints)*, 2011.

[24] S. Smale. Generalized Poincaré's conjecture in dimensions greater than four. *Ann. of Math.*, 74:391406, 1961.

[25] S. Smale. On gradient dynamical systems. *Ann. of Math.*, 74:199206, 1961.

[26] W. Wang, C. Bruyere, and B. Kuo. Competition data set and description in 2004 IEEE Visualization design contest. *http://vis.computer.org/vis2004contest/data.html*.

Nithin Shivashankar is a doctoral candidate at the Department of Computer Science and Automation, Indian Institute of Science, Bangalore. He holds a bachelors degree in computer science from the Manipal Institute of Technology. His research interests include scientific visualization and computational topology.

Senthilnathan M is a doctoral candidate at the Department of Computer Science and Automation, Indian Institute of Science, Bangalore. His research interests include topology based methods in visualization, parallel computing and GPGPU.

Vijay Natarajan is an assistant professor in the Department of Computer Science and Automation and the Supercomputer Education and Research Centre at the Indian Institute of Science, Bangalore. He received the Ph.D. degree in computer science from Duke University in 2004 and holds the B.E. degree in computer science and M.Sc. degree in mathematics from Birla Institute of Technology and Science, Pilani, India. His research interests include scientific visualization, computational geometry, computational topology, and meshing.