A discrete Morse-based approach to multivariate data analysis

Federico Iuricich*
University of Maryland,
College Park (MD), USA

Sara Scaramuccia†
University of Genova,
Italy

Claudia Landi‡
University of Modena
and Reggio Emilia, Italy

Leila De Floriani§
University of Maryland,
College Park (MD), USA

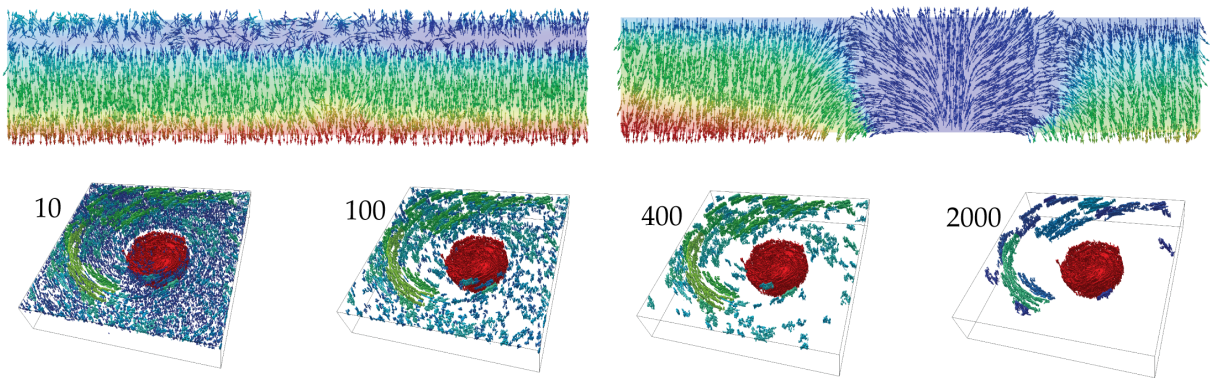

Figure 1: *The Hurricane dataset is a multivariate dataset defined on a cubical grid. Here we are considering two values per point describing the temperature and the pressure above ground. For each function, we are showing the function gradient computed on the original scalar values. Then, critical cells obtained with our method are collected in clusters called extrema-clusters. Using the extrema-clusters and their size (number of voxels composing each cluster) we provide an interactive method for filtering out uninteresting regions. The extrema-clusters are shown in the lower part. For each image, only the clusters bigger than the indicated size are shown. The color scheme is based on the cluster's size. Comparing the function gradients with the clusters obtained we notice that bigger clusters are created where the gradient disagree, for example in the eye of the hurricane.*

Abstract

Multivariate data are becoming more and more popular in several applications, including physics, chemistry, medicine, geography, etc. A multivariate dataset is represented by a cell complex and a vector-valued function defined on the complex vertices. The major challenge arising when dealing with multivariate data is to obtain concise and effective visualizations. The usability of common visual elements (e.g., color, shape, size) deteriorates when the number of variables increases. Here, we consider Discrete Morse Theory (DMT) [Forman 1998] for computing a discrete gradient field on a multivariate dataset. We propose a new algorithm, well suited for parallel and distribute implementations. We discuss the importance of obtaining the discrete gradient as a compact representation of the original complex to be involved in the computation of multidimensional persistent homology. Moreover, the discrete gradient field that we obtain is at the basis of a visualization tool for capturing the mutual relationships among the different functions of the dataset.

Keywords: Multivariate topology, Persistent homology, Segmentation analysis

Concepts: •**Computing methodologies** → *Shape analysis;*

*e-mail:iurif@umd.edu
†e-mail:sara.scaramuccia@dibris.unige.it
‡e-mail:claudia.landi@unimore.it
§e-mail:leila.defloriani@unige.it

1 Introduction

Topological methods aim at creating compact representations of a data set, focusing on its local features without losing global information. Among them, Morse theory [Milnor 1963] studies the relationships between the topology of a shape and the critical points of a real-valued smooth function defined on it. It has been used both for improving the performances of algorithms for computing homology and persistent homology and as the basis for creating segmentations of terrain and volume data based on the critical points of the scalar field defined on them. Discrete Morse theory [Forman 1998] due to Forman, a combinatorial counterpart of Morse theory, provides the notion of discrete gradient field (also called Forman gradient) for an efficient and derivative-free analysis of a scalar field. In the case of univariate data, a Forman gradient has been extensively used because, due to its discrete nature, it can be easily represented [De Floriani et al. 2015]. Multivariate data are characterized by several function values. A multivariate dataset usually consists of a cell complex discretizing the domain, and a vector-valued function defined on the complex vertices. In this context, the objective is to study the relations among multiple functions defined on the dataset rather than collecting results according to a single field. Such a component-wise segmentation would, in general, lose some functional relationships.

Given a multivariate data set, we propose an algorithm for computing a discrete gradient field compatible with all multiple functions defined on a cell complex. The problem has been faced from a theoretical point of view in [Allili et al. 2016], leading to a solution discussed in [Allili et al. 2015] of no practical interest, since it does not scale with the size of the data set. On the other hand, in order for

SA '16 Symposium on Visualization , December 05-08, 2016, , Macao
ISBN: 978-1-4503-4547-7/16/12...$15.00
DOI: http://dx.doi.org/10.1145/3002151.3002166

topological data analysis to be a valuable tool in the applications, we need to develop efficient algorithms that can effectively deal with current big data sets. We propose here the first algorithm capable of computing a discrete gradient field on real-world data. Our approach is easy-to-use and well suited for parallel and distributed implementations. We consider two applicative domains where the obtained representation can be successfully adopted, namely, for reducing the complexity of computing multipersistent homology and for the visual analysis of multivariate data.

In Section 2 we introduce the notions at the basis of our work. Related work is reviewed in Section 3. The new approach is described in Section 4. We discuss the data structures used in our implementation in Section 5. In Section 6 we present our experimental results and, in Section 7, we draw concluding remarks and we discuss future developments.

2 Background notions

In this section, we describe our setting. We will discuss our work in terms of simplicial complexes, although, all the results are valid for any kind of regular cell complex. Formalism is avoided to provide hands-on examples and intuitive definitions.

2.1 Simplicial and chain complexes

A k-dimensional simplex, or k-simplex, σ is the convex hull of $k+1$ affinely independent points. A *face* τ of σ is the convex hull of any subset of $k-1$ points of σ (indicated $\tau < \sigma$), while σ is a coface of τ (indicated $\sigma > \tau$). A *simplicial complex* Σ is a collection of simplices such that every face of a simplex in Σ is also in Σ and the *intersection property holds*, i.e., the intersection of any two simplices of Σ is a single shared simplex, possibly empty. The *boundary* of a simplex σ is the set $\mathrm{bd}(\sigma)$ of all $\tau < \sigma$. The *coboundary* $\mathrm{cb}(\sigma)$ (or the star) of σ is the set of simplices $\tau \in \Sigma$ such that $\tau > \sigma$. A simplicial complex Σ has dimension d (d-complex for short) if the maximum of the dimensions of its simplices is d. We will denote by Σ_k the set of k-simplices in Σ. An element in Σ_0 is also called a *vertex*. Given a simplicial complex Σ, it is possible to compute the *chain complex* associated with Σ. The chain complex is an algebraic representation of a simplicial complex, necessary for studying its homology [Hatcher 2002]. Intuitively, the homology of a simplicial complex Σ detects k-dimensional cycles of Σ, i.e. connected components (0-cycles), tunnels (1-cycles), voids (2-cycles), and so on. For a combinatorial description of the chain complex is enough to know the incidence relations in the complex, intuitively represented by a graph. Each node in the graph represents a k-simplex in Σ. Each arc connects two nodes if for the corresponding simplices, τ and σ, either $\sigma < \tau$ or $\sigma > \tau$.

2.2 Filtrations and Persistent Homology

A scalar field is generally represented as a pair (Σ, f), where Σ is a simplicial complex and f is a function $f : \Sigma_0 \longrightarrow \mathbb{R}$ defined on the vertices of Σ. Having defined a *filtration* as a sequence of simplicial complexes $\Sigma^0 \subset \Sigma^1 \subset ... \subset \Sigma^n = \Sigma$, using f we can induce a filtration on Σ by assigning, to each simplex of Σ, the maximum function value of its vertices. The filtration of Σ is then defined as the sequence of sub-level complexes $\Sigma^u = f^{-1}(-\infty, u]$. In this context, persistent homology [Edelsbrunner and Harer 2008] is used to study the homological changes of the sub-level sets.

In Figure 2, we show two different functions defined on the same 1-dimensional simplicial complex Σ. Filtering Σ according to f_1 means sweeping the graph of f_1, from bottom to top, introducing simplices along the way. By starting from the lowest function value,

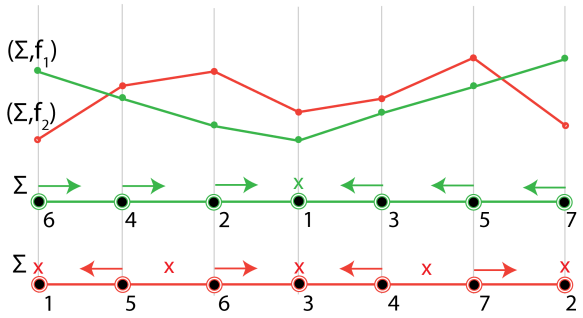

Figure 2: *Two filtering functions f_1 and f_2 defined on the same simplicial complex Σ. The graph of both functions is shown and the values of both f_1 and f_2 are explicitly indicated for each vertex of Σ. The discrete gradient computed on each function is also indicated. Arrows represent gradient pairs and exes represent critical simplices.*

we introduce vertex 1, then vertex 2 along with edge $[2,1]$, and so on. Looking at the filtration induced by function f_1, the only homological change occurs when vertex 1 is introduced, creating the first component. The filtration induced by f_2 instead provokes more changes. Simplices responsible of said changes are indicated with an ex. At the time vertices 1, 2 and 3 are introduced they all originate a new component. Successively vertices 4, 5 and 6 are all introduced together with edges $[3,4]$, $[1,5]$ and $[6,3]$ without affecting the homology. By adding edge $[5,6]$, two distinct components become connected, changing the homology again. The same holds for edge $[4,7]$.

In this paper, we focus on multivariate data having a function of the form $f : \Sigma_0 \longrightarrow \mathbb{R}^n$. We consider the partial order $\preceq$ on $\mathbb{R}^n$ defined component-wise by $u \preceq v$ if, for all component indexes i from 1 to n, $u_i \leqslant v_i$. A *multifiltration* is then a collection of simplicial complexes Σ^u, for $u \in \mathbb{R}^n$ such that, for every two parameters $u \preceq v$, it holds that $\Sigma^u \subseteq \Sigma^v$.

A multifiltration can be seen as an n-dimensional array where the indexes of each cell of the array correspond to the coordinates at which a simplex is introduced. Figure 3 shows the matrix representation for the simplicial complex Σ composed by two adjacent triangles. The bifiltration defined on Σ_0 is indicated for each vertex. The filtration value for all the other simplices σ is computed by taking, for each component, the maximum of the component value of σ's vertices. In the matrix representation we can find simplices depicted in red when the corresponding filtration value is equal to the cell index. We can notice that the cell of coordinates (0,0) corresponds to the empty space while the cell of coordinates (2,2) corresponds to Σ. In this context, multidimensional persistent homology aims at detecting the homological changes among nested pairs of complexes. While referring to [Carlsson and Zomorodian 2007] for a precise definition of multidimensional persistence, for the sake of this work we consider it as the generalization of persistence homology for a multifiltration.

2.3 Discrete Morse theory

The algorithm proposed in this paper retrieves an acyclic discrete vector field (called discrete gradient for brevity) over the domain Σ. The relevance of this output has to be seen within the framework of Forman's discrete Morse Theory [Forman 1998]. In discrete Morse Theory, a *(discrete) vector* is a pair of simplices (σ, τ) such that $\sigma < \tau$. A *discrete vector field V* is any collection of vectors over a simplicial complex such that each simplex belongs to at most one

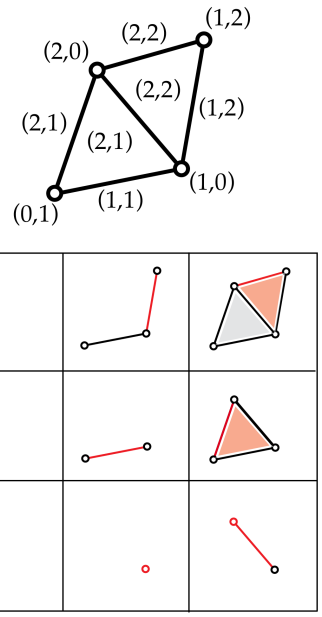

Figure 3: *A bifiltration, defined on a simplicial complex, represented as a matrix. In each cell we depict in red simplices that are introduced at the corresponding filtration level.*

Figure 4: *The discrete gradient computed on* (Σ, f) *according to the filtering function* $f = (f_1, f_2)$*. Arrows represent gradient pairs and exes represent critical simplices.*

vector. Given a discrete vector field V, if a simplex belongs to no vector, it is called *critical*. A V-path is a sequence of vectors (σ_i, τ_i) belonging to V , for $i = 1, \ldots, r$, such that, for all indexes $i \leqslant r - 1$, $\sigma_{i+1} < \tau_i$ and $\sigma_i \neq \sigma_{i+1}$. A V-path might be *closed* if $\sigma_1 = \sigma_r$ and *trivial* if $r = 1$. A *discrete gradient* V is a discrete vector field whose closed V-paths are all trivial. Given a discrete gradient V on a simplicial complex Σ, we can compute the chain complex associated with V represented as a graph G where,

- the nodes are the critical k-simplices of V, and
- the arcs are obtained by following the gradient paths of V.

Forman [Forman 1998] proves that the homology of Σ is always isomorphic to the homology of complex encoded by G. The discrete gradient V can be adapted to preserve the sub-level structure with respect to a filtering function. In the univariate setting, Theorem 4.3 in [Mischaikow and Nanda 2013] proves that the persistent homologies obtained by filtering the complex in G coincides with those of Σ^f. In Figure 2, we depict with an arrow the gradient pairs of the Forman gradient V computed for each filtering function. In this case, we can notice that the unpaired (critical) simplices are exactly those responsible for a topological change. This means that computing persistent homology, using all the vertices and edges of Σ or using only the simplices marked by an ex, lead to the same result. For multidimensional persistent homology, the analogous result is guaranteed by Corollary 3.12 in [Allili et al. 2016]. Referring to Figure 2, an example of discrete gradient compatible to $f = (f_1, f_2)$ is shown in Figure 4. In Section 4, we will discuss how to compute a discrete gradient compatible to a multivariate function.

3 Related Work

Different tools related to Morse theory [Milnor 1963] exist for dealing with with multivariate data. Jacobi sets [Edelsbrunner and Harer 2004] provide a method of studying the relationship between multiple Morse functions. Reeb spaces [Edelsbrunner et al. 2008], and their discrete counterpart, Join Contour nets [Carr and Duke 2013], are obtained as a generalization of the notion of Reeb graph. Huettenberger et al. [Huettenberger et al. 2013; Huettenberger and Garth 2015] use the concept of dominance relation and Pareto optimality to visualize k scalar-valued fields on a common domain giving an algorithm to compute Pareto sets for piecewise linear functions.

Few methods exist generalizing the notion of Forman gradient to the multivariate case. Discrete Morse Theory (DMT) [Forman 1998] has been used for both developing visualization tools [Gyulassy et al. 2012] and computing persistent homology [Mischaikow and Nanda 2013]. Many proposals exist for computing a discrete gradient from a function sampled at the vertices of a cell complex. The algorithm described in [King et al. 2005] is the first one to introduce a divide-and-conquer approach for computing a Forman gradient on real data. However, the latter has the main drawback of introducing many spurious critical simplices. Two approaches have been defined in [Shivashankar et al. 2012; Shivashankar and Natarajan 2012] for 2D and 3D images respectively. Focusing on a parallel implementation, they provide a substantial speedup in computing the discrete gradient still creating spurious critical simplices. In [Robins et al. 2011], a dimension-agnostic algorithm is proposed that processes the lower star of each vertex independently. It has been proved that up to the 3D case, the critical cells identified are in one-to-one correspondence with the topological changes in the sublevel sets, i.e. no spurious critical simplices are created. An efficient implementation of the latter, focused on regular grids, is discussed in [Günther et al. 2012] while, for simplicial complexes, the same algorithm as been extended to triangle [Fellegara et al. 2014] and tetrahedral meshes [Weiss et al. 2013]. The first dimension independent implementation for simplicial complexes is presented in [Fugacci et al. 2014].

The first attempt for extending the concept of discrete gradient to the multivariate case can be found in [Allili et al. 2015]. Given a multivariate function $\tilde{f} : \Sigma \longrightarrow \mathbb{R}$ in input, the idea at the base of [Allili et al. 2015] is to group cells where possible pairings can be found. We refer to these subsets as *filtration-based lower stars* (denoted L_f) which is defined on each simplex σ of Σ as,

$$\mathrm{L}_f(\sigma) := \{\alpha \in \Sigma \mid \alpha \supseteq \sigma \wedge \tilde{f}(\sigma) \preceq \tilde{f}(\sigma)\}.$$

The approach also requires following an order for visiting the simplices of Σ. For each simplex σ, if σ has been already classified (i.e., paired or defined as critical), it is ignored. Otherwise, the set of simplices in $\mathrm{L}_f(\sigma)$ are extracted and paired via homotopy expansion. The resulting discrete gradient is proved to have the same *multidimensional persistence* [Carlsson and Zomorodian 2007] of the original simplicial complex. However, the algorithm is linear in the number of simplices of Σ, and all the simplices have to be explicitly represented to avoid some simplex to be visited more than once. Thus, using the algorithm on real-world data is not feasible.

4 Computing a discrete gradient on multivariate data

In this section, we describe our algorithm for computing a discrete gradient on multivariate data. We consider a finite simplicial complex Σ and a *component-wise injective* function $f : \Sigma_0 \longrightarrow \mathbb{R}^n$ defined on its vertices Σ_0. Let $f = (f_1, \ldots, f_n)$, we get injectivity for

Algorithm 1 ComputeDiscreteGradient(Σ, f)

Input: Σ simplicial complex
Input: $f : \Sigma \longrightarrow \mathbb{R}^n$ component-wise injective function
Output: V gradient pairs
Output: C critical simplices

```
1: V ← ∅
2: C ← ∅
3: I =ComputeIndexing(Σ_0, f)
4: for all v ∈ Σ_0 do
5:     T_v =ComputeIndexLowerStar(v, I)
6:     K_v =SplitIndexLowerStar(f, T_v)
7:     for all S ∈ K_v do
8:         (V_S, C_S)=HomotopyExpansion(S, I)
9:         V = V.append(V_S)
10:        C = C.append(C_S)
11:    end for
12: end for
13: return (V, C)
```

each component f_i by means of simulation of simplicity [Edelsbrunner and Mücke 1990]. Function $\tilde{f}$ is defined on all the simplices of Σ by extending function f. For each simplex $\sigma \in \Sigma$ and for each component i of f we define

$$\tilde{f}_i(\sigma) = \max_{v \in \sigma} f_i(v).$$

The proposed algorithm (see Algorithm 1) takes inspiration from the one presented in [Robins et al. 2011] for scalar fields. The output of the algorithm consists of two lists:

- V, the list of pairs (σ, τ) with σ face of τ, and
- C, the list of critical simplices.

Efficiency is achieved by using a *well-extensible* indexing I, computed by defining a total order over the vertices of Σ (function `ComputeIndexing`). The domain of I is extended to the whole complex Σ via $\tilde{I}(\sigma) := \max_{v \in \sigma} I(v)$. We say that I is *well-extensible* with respect to function f when, for every two simplices σ_1 and σ_2,

$$\tilde{f}(\sigma_1) \preceq \tilde{f}(\sigma_2) \implies \tilde{I}(\sigma_1) \leqslant \tilde{I}(\sigma_2).$$

There are different ways to obtain a well-extensible indexing and any strategy is equivalent as we will see later. In our approach, we achieve this result by ordering the vertices in ascending order based on a single component f_i. Notice that, since f is component-wise injective, $f_i(v_1) \neq f_i(v_2)$. If $\tilde{f}(\sigma_1) \preceq \tilde{f}(\sigma_2)$, then $\tilde{f}_i(\sigma_1) \preceq \tilde{f}_i(\sigma_2)$. For both σ_1 and σ_2, their function value under f_i is given by one vertex with maximum index I in the simplex. Thus, our choice ensures I to be well-extensible with respect to f. In Figure 5(a) I (indicated within square brackets) is computed by ordering the vertices with respect to the first component of f.

The algorithm processes the vertices one by one (possibly in parallel) and, for each vertex $v \in \Sigma_0$, the index-based lower star $\mathrm{L}_I(v)$ is computed (function `ComputeIndexLowerStar`) as

$$\mathrm{L}_I(v) := \{\sigma \in \Sigma \mid \tilde{I}(\sigma) = I(v).$$

The union of all the index-based lower stars is a partition of Σ. Each index-based lower star $\mathrm{L}_I(v)$ can possibly consists of a single simplex, i.e. the vertex v, or multiple simplices. When $\mathrm{L}_I(v)$ is composed by more than one simplex, we know that for each $\sigma \in \mathrm{L}_I(v)$, $\tilde{f}(\sigma) \geqslant \tilde{f}(v)$ by construction. Most importantly, we are guaranteed that for any pair of simplexes σ, τ, if $\tilde{f}(\sigma) = \tilde{f}(\tau)$ they both belong to the same lower star. Intuitively, if σ and τ have the same value of $\tilde{f}$, there exists a vertex v on the boundary of both such that $f_i(v) = \tilde{f}_i(\sigma) = \tilde{f}_i(\tau)$. Since we built I based on f_i, v is also the vertex with maximum index I among those incident in both σ and τ. It follows that $\{\sigma, \tau\} \in L_I(v)$. However, we may use any method for building I as long as the obtained indexing is well-extensible. In other words, simplices belonging to the same sub-level set (i.e., simplexes having the same value of $\tilde{f}$) will end up in the same L_I. Once the indexed-based lower stars have been computed, simplices have to be organized to be correctly paired. We recall that two simplices σ, τ can be paired if and only if $\tilde{f}(\sigma) = \tilde{f}(\tau)$. Function `SplitIndexLowerStar` is then used for subdividing the simplices in $T_v = \mathrm{L}_I(v)$ according to $\tilde{f}$. More precisely, `SplitIndexLowerStar`(f, T_v) returns K_v, that is the quotient of index-based lower star obtained through the equivalence relation $\sigma \sim \tau$ if and only if $\tilde{f}(\sigma) = \tilde{f}(\tau)$.

For each equivalence class S of K_v we compute the local discrete gradient (V_S, C_S) via homotopy expansion. Two simplices, say k-simplex σ and $(k+1)$-simplex τ, are paired via homotopy expansion when σ has no unpaired faces and τ has only one unpaired face (i.e. σ). Each local pair in V_S and each critical simplex in C_S contributes to the global output (V, C). At this point the procedure `HomotopyExpansion` has no conceptual differences from the one described in [Robins et al. 2011] except that we will work with the simplices in S and not on the full lower star. Two priority queues `PQ0` and `PQ1` are used for selecting which simplex in S needs to be paired respectively to a higher or to a lower dimensional simplex. The priority queues are organized by listing, in lexicographic order, the tuples containing the values of I for the vertices of each simplex σ. This ensures that if $\sigma \subsetneq \tau$ in S, then σ takes priority over τ. We have demonstrated the equivalence between our output and the one computed by the algorithm `Matching` described in [Allili et al. 2015] (proof is omitted for brevity). Thus, algorithm produces a discrete gradient V compatible with the multivariate function $f = (f_1, \ldots, f_n)$, reducing the input complex Σ without affecting the persistence module with respect to f. In Figure 5, we can see a working example for the procedure `ComputeDiscreteGradient`. Figure 5(a) shows the simplicial complex indicating the function $\tilde{f}$ for each simplex and the computed indexing I. In Figure 5(b) the index-based lower stars are extracted. Simplices having the same value belong to the same lower star $\mathrm{L}_I(v)$. Notice that a single $\mathrm{L}_I(v)$ may enclose simplices with different values of $\tilde{f}$. For example, the lower star $\mathrm{L}_I([3])$ contains simplices with values $(3,0), (3,1), (3,2)$ and $(3,3)$. `HomotopyExpansion` is called independently for each equivalence class in $\mathrm{L}_I([3])$ (Figure 5(c)). The set of critical simplices and gradient pairs obtained from each lower star are combined in the final discrete gradient depicted in Figure 5(d).

Given a simplicial complex Σ and a simplex $\sigma \in \Sigma$, $\tilde{f}(\sigma)$ can be retrieved in linear time in the number of the vertices of σ. We also consider the extraction of the boundary/coboundary of a given simplex σ as a linear operation with respect to the number of simplices incident in σ. The main contribution to the complexity of the algorithm comes from procedure `ComputeIndexing` and from cycling over the vertices of Σ. Functions `ComputeIndexLowerStar` and `SplitIndexLowerStar` only require the retrieval of the coboundary of each vertex. In `ComputeIndexing` vertices are sorted according to a single component of the input function. Thus, it requires $O(m_0 \cdot \log m_0)$ operations, with m_0 the number of vertices in Σ. The computational complexity of `HomotopyExpansion` is determined by the updates of the two queues `PQ0` and `PQ1`. We have implement each queue as a heap. Their maximal length is

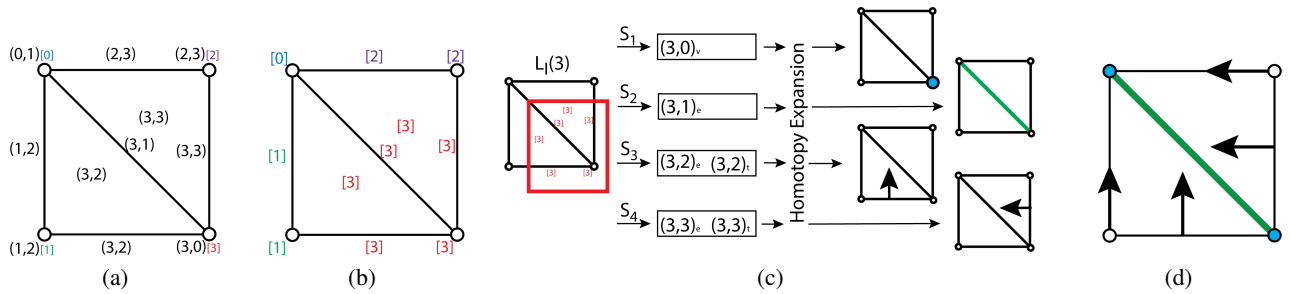

Figure 5: *(a) The filtering function values are depicted as pairs. The vertex indexing I is depicted within square brackets. (b) $\tilde{I}$ is propagated to all the simplices. Colors indicate simplices belonging to the same index-based lower star. (c) For each lower star, the level sets S_i are built collecting simplices having the same function value $\tilde{f}$. Pairs and critical simplices are created via homotopy expansions within each S_i. (d) The discrete gradient is obtained as the union of all the pairs and critical simplices classified in the lower stars. Critical vertices are depicted in blue and the critical edge is depicted in green.*

the cardinality of S, denoted Λ. It is easy to see that each simplex in S always enters some queue and at most once for each of the two queues, as in [Robins et al. 2011]. As a result, each call of `HomotopyExpansion` consists of a number of operations of $O(\Lambda)$. When we are in low dimensions, as it is the case in real-world applications of multivariate data (d equal to 2 or 3), the number of simplices per level set (i.e., Λ) is always negligible, as well as the number of total level sets S. Thus, the complexity is dominated by the for-cycle (and thus it is linear in the number of vertices m_0). This represents a substantial improvement with respect to the algorithm presented in [Allili et al. 2015] that is linear in the number of simplices.

5 Data Structures

In the following, we will present the results obtained by running our algorithm on datasets having a triangulated domain (triangle meshes) and volumetric images discretized as cubical grids. We will describe first the data structures used for encoding the input complexes and the gradient in output.

The memory requirements of our algorithm are determined by the overhead necessary for encoding the input complex and the discrete gradient. The data structures used for the latter can be specialized based on the type of the input complex, namely triangle meshes or cubical grids. In both cases, the multivariate function is defined on the vertices of Σ. This means storing a floating point value for each component of f, for each vertex.

Triangle meshes A triangle mesh is a 2-dimensional simplicial complex formed by vertices, edges, and triangles. To compactly encode the relations among these simplices we are using an incidence-based data structure with adjacency (IA) introduced in [Nielson 1997]. The IA data structure encodes vertices and triangles explicitly, plus some additional relations. For each triangle σ we encode a reference to its three vertices and a reference to the triangles sharing a face with σ. For each vertex instead, we encode a single triangle incident in it. Let m_0 and m_2 the number of vertices and triangles in Σ, respectively, the total footprint required for encoding Σ is $m_0 + 6m_2$. The discrete gradient is here encoded by adopting the representation described in [Weiss et al. 2013]. The latter focuses on encoding all the gradient pairs locally to a triangle. The faces of dimension 1 of a triangle σ are the three edges each one having two faces of dimension 0 (vertices). Thus there are 9 possible gradient pairs internally to a triangle. If we consider also the possible pairs between ad edge of σ and an adjacent triangle we get 12 possible gradient pairs and thus $2^{12} = 4096$ possible combinations. However, since the restrictions imposed by the discrete gradient (i.e. that each simplex can be involved in at most one pairing) we have only 97 valid cases for a triangle. We can encode all these cases using only 1 byte per triangle and encoding the gradient only requires m_2 bytes.

Cubical Grids Representing a cubical grid is a much easier task than unstructured meshes. Let us consider the graph $G = (N, A)$ where N is the set of cells composing the cubical grid and A is the set of incidence relations between the cells in N. The regular distribution and connectivity of the cells in G makes possible to encode the topology of the complex implicitly. By enumerating the cells in N, we can extract any relation in A using index calculations without any overhead. Based on the same rationale, the discrete gradient is encoded by assigning a Boolean value to each arc α in A, where the two cells connected by α are paired in V. Then, the discrete gradient is encoded as an array of bits of length $|A|$.

6 Experimental results

Experiments have been performed on a MacBook Pro with a 2.8GHz quad-core processor and 16GB of memory. The first three datasets are triangle meshes, each having three scalar fields defined on the vertices (see [Cerri et al. 2014] Section 6.2 - Db2 for details on the functions). The last four datasets are collections of time-varying scalar fields defined on a cubical grid. For our experimental analysis, we have coupled a subset of the original scalar fields, on a single timestep. The Hurricane Isabel WRF Model Data describes the simulation of the Hurricane Isabel. The first dataset $Hurricane_{pt}$ is created by coupling the scalar fields describing pressure (weight of the atmosphere above a grid point) and temperature. The second dataset $Hurricane_{rit}$ still combines the temperature measurement with two scalar fields describing the density of clouds in the atmosphere distinguishing between rain clouds and ice clouds. The turbulent combustion simulation instead is a simulation of temporally-evolving plane jet flames. $Combustion_{vo}$ presents two scalar fields describing vorticity and combustion-generated OH. $Combustion_{hm}$ instead combines the scalar field describing mixture fraction and the one for chi distribution. Results are reported in Table 1.

6.1 Multipersistence homology

When considering topological data analysis, the discrete gradient may be used as a compact representation of the original complex to be involved in the computation of multidimensional per-

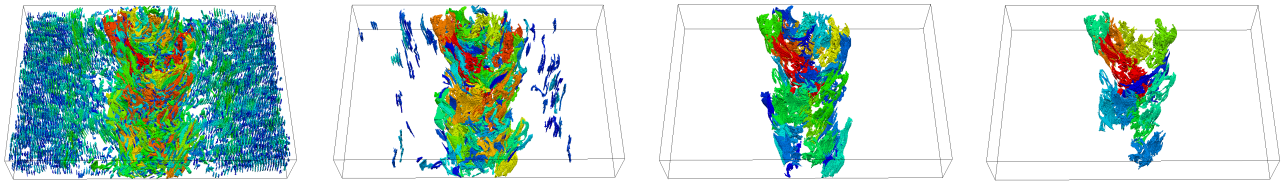

Figure 6: *Critical cells are computed and collected in clusters called extremal-clusters. Using the extremal-clusters and their size (number of voxels composing each cluster) we can visualize them on screen, filtering out the smaller ones. Here we are showing the extrema-clusters found on the Turbulent Combustion Simulation drawing only those larger than the indicated size. Smaller clusters are depicted in blue, the larger ones in red.*

Dataset	Size	Fields	$\|C\|$	Time G.		Time M.
				Single	Parallel	
Neptune	12M simpl	3	1.4M (8x)	2.1m	0.8m (2.5x)	7.3m
Statue	30M simpl	3	2.1M (15x)	6.1m	2.1m (2.9x)	12.1m
Lucy	84M simpl	3	5.6M (15x)	15.4m	5.6m (2.7x)	22.9m
Hurricane_{pt}	$[500\times500\times100]$	2	12.2M (16x)	46.1m	11.2m (4.1x)	13.4m
Hurricane_{rst}	$[500\times500\times100]$	3	5.7M (40x)	47.8m	12.2m (3.9x)	17.0m
Combustion_{vo}	$[480\times720\times120]$	2	43.1M (7.6x)	58.2m	18m (3.2x)	23.7m
Combustion_{hm}	$[480\times720\times120]$	2	45.9M (7.3x)	67.2m	18.6m (3.5x)	20.1m

Table 1: *Results obtained computing the discrete gradient the the boundary maps on three triangle meshes and four volumetric images. For each dataset we indicate its size (number of cells for triangle meshes or resolution of the volume datasets), number of scalar values for each point (*Fields*), number of critical cells obtained (*C*), and timings required for computing the discrete gradient (*Time G.*) and the boundary maps (*Time M.*).*

sistent homology. By computing the multidimensional persistent homology one means retrieving the persistence module [Carlsson and Zomorodian 2007], that is the algebraic entity encoding all the multiparametrized family of homologies and the linear maps among them. The algorithm proposed in [Carlsson et al. 2009] for computing the persistence module of a multifiltration has worst time complexity $O(n^4m^3)$, where n is the number of functions and m is the number of simplices. Other targets for the multidimensional persistent homology computation are the *rank invariant* and the *multigraded Betti numbers* [Carlsson and Zomorodian 2007]. Even if easier to compute, the main drawback of the rank invariant is the number of comparable multiparameters $O(m^{2n})$ where it should be computed. To this aim, the tool RIVET [Lesnick and Wright 2015] is an interesting optimized visualization tool for the $n = 2$ case. The tool constructs a suitable *barcode template* in $O(m^3\kappa + (m + \log\kappa)\kappa^2)$, where $\kappa = \kappa_1\kappa_2$ with κ_i the number of different coordinates for the i^{th}-component in the support of the multigraded Betti numbers of any index. Then, the tool manages to complete the rest of the information by updating those results in linear time with respect to m. By removing unnecessary cells, our contribute is twofold. On the one hand, it allows reducing the impact of parameter m in a single and consistent way in multidimensional persistent homology computation without the need of repeating the procedure for each slice. Moreover, for the rank invariant, entire function level sets might disappear and this possibly reduces the impact of parameter κ. For each dataset in Table 1 we are reporting the compression factor, i.e. the ratio between the number of cells in the original complex and the number of critical cells in the discrete gradient and we also indicate the timings for computing the discrete gradient and the boundary matrices (connection between the critical cells). We can see that the compression factor obtained goes from 40x to 7.6x in the worst case. The efficiency provided by our algorithm in computing the discrete gradient is remarkable. We have compared a single threaded implementation with a parallel implementation based on OpenMP. On average, the parallel implementation achieves a 2.7x speedup when working on triangle meshes and a 3.6x speedup when working with cubical grids. Starting from the discrete gradient we are also computing the boundary maps of the corresponding chain complex. This can be done by retrieving the connections between pairs of critical simplices, using the gradient paths of V, as described in [Fugacci et al. 2014]. Timings required for computing the boundary maps are reported in Table 1 (column Time M.). Here the implementation is single threaded only.

6.2 Topology-based visualization

While visualizing the information provided by multipersistent homology is currently a long term goal, relevant information about a multivariate dataset can be obtained by studying the discrete gradient alone. Referring to Figure 1, we notice that clusters of critical cells appear where the gradients of the two functions are in an opposite direction. These clusters can be analyzed to get some hints for where to study the functions behavior with greater accuracy.

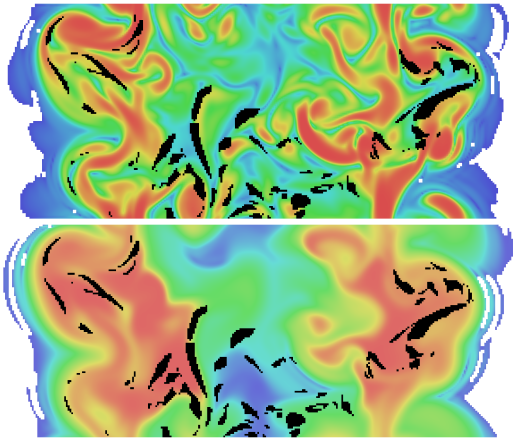

Figure 7: *Slices of the two fields on the Turbulent Combustion Simulation (Vorticity) and (OH) are shown with superimposed the extrema-clusters occupying the same area (depicted in black).*

Extrema-clusters Starting from the discrete gradient we collect the sets of connected critical cells. We call *minima-clusters* the collection of critical 0-cells connected by critical 1-cells. They can be easily computed using an union-find data structure defined on the critical 0-cells and by linearly processing all the critical 1-cells. Each 0-cell initially forms a minima-cluster on its own. Sweeping on the critical 1-cells, we consider only those having two critical 0-cells v_1,v_2 on their boundary. If v_1,v_2 belong to different minima-clusters, they are merged. Dually, *maxima-clusters* are defined as collections of critical d-cells connected by critical $(d-1)$-

cells and they are computed in a similar fashion. Maxima- and minima-clusters are easily encoded as segmentations defined on the vertices/d-cells of the dataset (i.e. an integer label for each element). Minima- and maxima-clusters overlap in those areas where the function is more chaotic. Then, we can identify these regions by intersecting the two decompositions and originating what we call the *extremal-clusters*. Each voxel is labeled with an index indicating the pair of minima and maxima clusters it belongs to. While the extremal-clusters are not a rigorous estimator for evaluating differences among multiple functions, they are fast to compute and, more importantly, they provide an interactive framework to help the user to eliminate noise.

In Figure 1 we show the result obtained on the $Hurricane_{pt}$ dataset. In this dataset, temperature values are decreasing from the lower part of the dataset to the upper part. Pressure has an opposite trend in the eye of the hurricane and behaves similarly to the temperature in the remaining part (see the function gradients shown in the figure). This result is evident at a glance by looking at the large red cluster appearing in all the images in Figure 1. The small regions, concentrated on the top of the dataset, are clearly visible when the threshold used is small (10 - 400). They are originated by the chaotic direction of the gradient in the higher part of the temperature function. In Figure 6, we show an example illustrating the extremal-clusters obtained on the $Combustion_{vo}$ dataset. Once the clusters have been computed, we filter out the smaller ones based on the size of the cluster (number of voxels). In this sense, the size of each cluster works as an intuitive value for defining the importance of the area where functions disagree. In the leftmost image of Figure 6, all extremal-clusters formed by more than 50 voxels are shown. This decomposition is progressively refined by increasing the minimum size of an extremal-cluster to 1000, 5000 and 18000 voxels. Figure 7 better describes the way extremal-clusters identify the area in which the functions differ. The same slice is shown coloring accordingly to vorticity (image above) and to the *OH* produced (image below). The extremal-clusters shown in Figure 6 (with threshold 18K) are sliced as well and depicted in black. Focusing on the neighborhood of the black segments in the two images we can notice the functions, having an opposite behavior, by looking at the color patterns.

7 Concluding Remarks

We have proposed a new approach for computing a discrete gradient vector field on multivariate data. The key idea underlying our approach is avoiding to retrieve the lower star of all the cells of the input complex. Moreover, by using the indexed-based lower star we produce a partition of the input complex. The algorithm is easy to parallelize and the actual performances have been tested on real-world datasets proving its practical importance. By proving the equivalence with the output of [Allili et al. 2015] we have demonstrated that the discrete gradient computed is compatible with the multidimensional persistent homology induced by the multiple functions. Thus, our contribution represents a first step towards an efficient computation of multidimensional persistent homology. Based only on the information provided by the discrete gradient, we have defined a new visualization tool. Based on the clusters of critical cells in the discrete gradient, we are able to identify the subsets of the dataset where the input functions have an uneven behavior. By classifying the clusters based on their dimension, we are able to filter out the noise and uninteresting regions. The visualization framework described shares common traits with others methods defined for studying interactions among multiple scalar fields.

The work described in [Nagaraj et al. 2011] shares our same rationale, focusing on computing a gradient-based comparison measure. The measure, called *multifield comparison measure* (denoted η_F), captures the extent of alignment of the gradient vectors at a point. The multifield comparison measure can be seen as a scalar field describing the behavior of the multiple functions. Looking at the distribution of η_F, the relationships of the input functions can be studied. The approach is numerical, as opposed to ours that is combinatorial. Results are mainly shown on 2D grids while timings for computing η_F are not provided. Visualizing η_F in the case of 3D scalar fields is not straightforward. As opposed, we are able to obtain clear three-dimensional representations of the input fields also providing an interactive framework for studying the extrema clusters. From a computational complexity point of view, both methods depend only on the dimension of the domain, being independent of the number of fields.

Pareto sets [Huettenberger et al. 2013] and Joint Contour Nets [Huettenberger et al. 2013; Carr and Duke 2014] aim at subdividing the domain with respect to the vector-valued function behavior. A Pareto set includes Pareto optima (points with incomparable neighborhood), Pareto minima (points with neighborhood either incomparable or greater), and Pareto maxima (points with neighborhood either incomparable or lower). The *reachability graph* is introduced in [Huettenberger et al. 2013] for encoding the relations among Pareto optima, minima and maxima so as to provide a meaningful representation of the function under investigation. Joint Contour Nets (JCNs) [Carr and Duke 2014] split each cell into connected regions of points having equal quantized isovalue for all the fields. Each connected region is called a *slab*. Adjacency relations between slabs are encoded as arcs in the net. JCNs are related to Jacobi sets [Edelsbrunner and Harer 2004] since these latter correspond to a subset of the arcs in the net. In our method, cells are defined as critical when the gradients of their boundary points are opposite. For the same reason, a point is defined Pareto. As described in [Huettenberger et al. 2013], the Jacobi sets for two functions are the points where the gradients are parallel. Thus, in this case, both the method in [Huettenberger et al. 2013] and the critical vertices found by our method are a subset of the Jacobi sets. We have not been able to compare the two methods on the same dataset but, for the volume dataset shown in [Huettenberger et al. 2013] the reported timing for extracting the reachability graph on a tetrahedral mesh of roughly 750K simplices is around 20 minutes. By comparing with our results, we can see that our method is more efficient. Taking inspiration from what has been done in the univariate case we are currently studying the information retrieved by visiting the gradient V-paths. Using ascending/descending paths of the discrete gradient computed we will be able to design an adjacency relation among the extrema-clusters and possibly to define a simplification process for building hierarchical representations of such clusters.

Acknowledgements

This work has been partially supported by the US National Science Foundation under grant number IIS-1116747. Hurricane Isabel data produced by the Weather Research and Forecast (WRF) model, is courtesy of NCAR, and the U.S. National Science Foundation (NSF). The turbulent combustion simulation is made available by Dr. Jackqueline Chen at Sandia Laboratories through US Department of Energy's SciDAC Institute for Ultrascale Visualization. Meshes are courtesy of The Stanford 3D Scanning Repository and the Aim@Shape repository.

References

ALLILI, M., KACZYNSKI, T., LANDI, C., AND MASONI, F., 2015. A new matching algorithm for multidimensional persistence. ArXiv, Id:1511.05427, Nov.

ALLILI, M., KACZYNSKI, T., AND LANDI, C. 2016. Reducing complexes in multidimensional persistent homology theory. *Journal of Symbolic Computation*.

CARLSSON, G., AND ZOMORODIAN, A. 2007. The theory of multidimensional persistence. In *SoCG '07 Proceedings of the twenty-third annual symposium on Computational geometry*, ACM New York, Gyeongju, South-Korea, vol. 392, 184–193.

CARLSSON, G., SINGH, G., AND ZOMORODIAN, A. 2009. Computing multidimensional persistence. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5878 LNCS*, 1, 730–739.

CARR, H., AND DUKE, D. 2013. Joint contour nets: Computation and properties. In *Visualization Symposium (PacificVis), 2013 IEEE Pacific*, 161–168.

CARR, H., AND DUKE, D. 2014. Joint contour nets. In *IEEE Transactions on Visualization and Computer Graphics*, IEEE Computer Society, vol. 20, 1100–1113.

CERRI, A., FABIO, B. D., JABLONSKI, G., AND MEDRI, F. 2014. Comparing shapes through multi-scale approximations of the matching distance. *Computer Vision and Image Understanding 121*, 43–56.

DE FLORIANI, L., FUGACCI, U., IURICICH, F., AND MAGILLO, P. 2015. Morse Complexes for Shape Segmentation and Homological Analysis: Discrete Models and Algorithms. *Computer Graphics Forum 34*, 2, 761–785.

EDELSBRUNNER, H., AND HARER, J. 2004. Jacobi sets of multiple Morse functions. In *Foundations of Computational Mathematics, Minneapolis 2002*, vol. 312 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 35–57.

EDELSBRUNNER, H., AND HARER, J. 2008. Persistent homology-a survey. *Contemporary mathematics 453*, 257–282.

EDELSBRUNNER, H., AND MÜCKE, E. P. 1990. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph. 9*, 1, 66–104.

EDELSBRUNNER, H., HARER, J., AND PATEL, A. K. 2008. Reeb spaces of piecewise linear mappings. In *Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry*, ACM, New York, NY, USA, SoCG '08, 242–250.

FELLEGARA, R., LURICICH, F., DE FLORIANI, L., AND WEISS, K. 2014. Efficient computation and simplification of discrete morse decompositions on triangulated terrains. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, New York, NY, USA, SIGSPATIAL '14, 223–232.

FORMAN, R. 1998. Morse theory for cell complexes. *Advances in mathematics 134*, 1, 90–145.

FUGACCI, U., IURICICH, F., AND DE FLORIANI, L. 2014. Efficient computation of simplicial homology through acyclic matching. In *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2014, Timisoara, Romania, September 22-25, 2014*, IEEE, F. Winkler, V. Negru, T. Ida, T. Jebelean, D. Petcu, S. M. Watt, and D. Zaharie, Eds., 587–593.

GÜNTHER, D., REININGHAUS, J., WAGNER, H., AND HOTZ, I. 2012. Efficient computation of 3d morse–smale complexes and persistent homology using discrete morse theory. *The Visual Computer 28*, 10, 959–969.

GYULASSY, A., KOTAVA, N., KIM, M., HANSEN, C., HAGEN, H., AND PASCUCCI, V. 2012. Direct Feature Visualization Using Morse-Smale Complexes. *IEEE transactions on visualization and computer graphics 18*, 9, 1549–62.

HATCHER, A. 2002. Algebraic topology. *Cambridge UP, Cambridge*.

HUETTENBERGER, L., AND GARTH, C. 2015. *A Comparison of Pareto Sets and Jacobi Sets*. Springer Berlin Heidelberg, Berlin, Heidelberg, 125–141.

HUETTENBERGER, L., HEINE, C., CARR, H., SCHEUERMANN, G., AND GARTH, C. 2013. Towards multifield scalar topology based on Pareto optimality. *Computer Graphics Forum 32*, 3 (Jun), 341–350.

KING, H., KNUDSON, K., AND MRAMOR, N. 2005. Generating Discrete Morse Functions from Point Data. *Experimental Mathematics 14*, 4, 435–444.

LESNICK, M., AND WRIGHT, M. 2015. Interactive Visualization of 2-D Persistence Modules. *Preprint ArXiv*, 1–75.

MILNOR, J. W. 1963. *Morse Theory*. Annals of Mathematics Studies. Princeton University Press.

MISCHAIKOW, K., AND NANDA, V. 2013. Morse Theory for Filtrations and Efficient Computation of Persistent Homology. *Discrete & Computational Geometry 50*, 2, 330–353.

NAGARAJ, S., NATARAJAN, V., AND NANJUNDIAH, R. S. 2011. A gradient-based comparison measure for visual analysis of multifield data. *Computer Graphics Forum 30*, 3, 1101–1110.

NIELSON, G. M. 1997. Tools for triangulations and tetrahedralizations and constructing functions defined over them. In *Scientific Visualization: overviews, Methodologies and Techniques*, G. M. Nielson, H. Hagen, and H. Müller, Eds. IEEE Computer Society, Silver Spring, MD, ch. 20, 429–525.

ROBINS, V., WOOD, P. J., AND SHEPPARD, A. P. 2011. Theory and algorithms for constructing discrete morse complexes from grayscale digital images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 8, 1646–1658.

SHIVASHANKAR, N., AND NATARAJAN, V. 2012. Parallel computation of 3d morse-smale complexes. *Comput. Graph. Forum 31*, 3, 965–974.

SHIVASHANKAR, N., MAADASAMY, S., AND NATARAJAN, V. 2012. Parallel computation of 2d morse-smale complexes. *IEEE Trans. Vis. Comput. Graph. 18*, 10, 1757–1770.

WEISS, K., IURICICH, F., FELLEGARA, R., AND DE FLORIANI, L. 2013. A primal/dual representation for discrete Morse complexes on tetrahedral meshes. *Computer Graphics Forum 32*, 3pt3, 361–370.