Topologically-Consistent Simplification of Discrete Morse Complexes

Abstract

We address the problem of simplifying Morse-Smale complexes computed on volume datasets based on discrete Morse theory. Two approaches have been proposed in the literature based on a graph representation of the Morse-Smale complex (explicit approach) and on the encoding of the discrete Morse gradient (implicit approach). It has been shown that this latter can generate topologically-inconsistent representations of the Morse-Smale complex with respect to those computed through the explicit approach. We propose a new simplification algorithm that creates topologically-consistent Morse-Smale complexes and works both with the explicit and the implicit representations. We prove the correctness of our simplification approach, implement it on volume data sets described as unstructured tetrahedral meshes and evaluate its simplification power with respect to the usual Morse simplification algorithm.

Keywords: Discrete Morse theory, Morse and Morse-Smale complexes, Forman gradient simplification

1. Introduction

A volume dataset is characterized by a finite set of points, regularly or irregularly distributed over a domain, and by a scalar value associated with each of them. Morse theory [1] has been recognized as an important tool for studying the morphology of a scalar field in several applications, including physics, chemistry, medicine, geography, etc.

When working with real data, the size of the morphological segmentation and the presence of noise require specific tools for analysis and visualization. Multi-resolution models are then defined to provide domain experts with an interactive tool for the exploration of such data. At the base of the definition of a multi-resolution model stands the definition of a simplification algorithm, used for building the model.

The operator defined in Morse theory for topologically simplify a dataset is called cancellation. Cancellation removes two critical points by locally modifying the integral lines originating and converging in the two points [2]. Two different approaches have been defined for applying such operator onto real data. The first approach uses a graph representation of the connectivity of the critical points [3]. The geometry of the Morse complexes is explicitly stored in the representation, attached to the graph nodes. For this reason, this approach is also known as explicit. Removing two critical points corresponds to deleting two nodes of the graph and merging the attached entities.

The second approach is based on discrete Morse theory [4], a combinatorial counterpart of smooth Morse theory, where the notion of discrete Morse gradient (also called a Forman gradient) is defined. A Forman gradient field is a collection of critical simplices (corresponding to the critical points of a smooth function f) and a set of gradient paths, simulating the integral lines of f. From a Forman gradient field, the Morse cells can be computed navigating the gradient paths and, thus, they do not need to be stored explicitly.

Alongside with the notion of critical point and Morse complex also the cancellation operator has been defined in this combinatorial framework. Applying the cancellation operator on a Forman gradient corresponds to eliminating a pair of critical simplices and changing the direction of the gradient arrows along the path between them. This update implicitly modifies the Morse cells accordingly. Thus, this approach is also known as implicit.

Simplifications performed with the explicit method are generally faster thanks to the graph-based representation, and thus preferable when high performance is needed. On the other hand, the implicit method avoids the extraction of the Morse cells and is preferable when compactness is more relevant. However, even if the two methods are equivalent in 2D, the implicit representation may present inconsistencies when working in higher dimensions. The origin of the problem, described in [5] for 3D scalar fields, is attributed to the structure of the discrete gradient pairs along the paths connecting 1-saddles to 2-saddles. This makes the implicit approach useless in practice when simplifying volumetric data.

Different multi-resolution models have been defined in the literature based on an explicit simplification sequence [6, 7]. The resulting models have been proven to be efficient for interactively modifying and visualizing Morse features but they are lacking in compactness. In this direction, the Forman gradient would be a perfect candidate for defining a compact model but, due to the inconsistency problem described in [5], no models have been yet defined for volumetric data.

In this work we consider the problem of defining a simplification algorithm, for the implicit method, based on an efficient representation and free from the topologically inconsistencies that affect the standard implicit method. Our approach is described and implemented for unstructured tetrahedral meshes, but it is entirely dimension-independent. Thus, the major con-
tributions of this work are:

(i) the definition of a compact data structure for the efficient
simplification of a Forman gradient (see Section 4);
(ii) a method for removing shared gradient paths (see Section
7);
(iii) an algorithm for simplifying a volume dataset, that gen-
erates a topologically-consistent simplification sequence
(see Section 8).

By using this algorithm we are able to obtain a simplifica-
tion sequence free from topological inconsistencies, based on
which a compact multi-resolution model can be defined and im-
plemented.

2. Discrete Morse Theory

Morse theory [1] studies the relationships between the topol-
ogy of a manifold $M$ and the critical points of a real-valued
function $f$ defined on $M$. Recently, a discrete counterpart
of Morse theory has been proposed by Forman for cell com-
plexes [4]. Cell complexes provide a compromise between
the theoretical concept of topological space and the intuition
of a discretized shape. They encompass both the notions
of simplicial and cubical complex, which can be intuitively
described as collections of simplices or elementary cubes on a
regular grid, respectively. Here, we will review discrete Morse
theory in the context of simplicial complexes for simplicity.

A simplex of dimension $k$ (k-simplex) is the convex hull of
$k + 1$ affinely independent points. Given a k-simplex $\sigma$, any
simplex $\tau$ which is the convex hull of a non-empty subset of
the points generating $\tau$ is called a face of $\tau$. Conversely, $\tau$ is
called a coface of $\sigma$. A simplicial complex $\Sigma$ is a finite set
of simplices, such that each face of a simplex in $\Sigma$ belongs to $\Sigma$
and each non-empty intersection of any two simplices in $\Sigma$ is a
face of both. We define the dimension of a simplicial complex
$\Sigma$, denoted as $\text{dim}(\Sigma)$, as the largest dimension of its simplices.

In discrete Morse theory, a discrete function $F$ is defined on
all the simplices of $\Sigma$, and it is called a discrete Morse function
(or a Forman function) if any k-simplex $\sigma$ of $\Sigma$ has $F$ value
greater than its $(k - 1)$-faces and lower than its $(k + 1)$-faces,
with at most one exception (see Figure 1(b)). A k-simplex is
called a critical k-simplex (or a k-saddle) if there is no exception.
In particular, a 0-saddle is called a minimum and, a d-
saddle a maximum, when $d = \text{dim}(\Sigma)$. The unique exception to
the above rule, which holds for any non-critical simplex, allows
for pairing such simplex with one of its faces, or one of its co-
faces. Such pair can be depicted as an arrow from a k-simplex
$\sigma$ (tail) to a $(k + 1)$-simplex $\tau$ (head) (see Figure 1(c)).

A discrete vector field $V$ on a simplicial complex $\Sigma$ is a col-
lection of pairs $(\sigma, \tau)$ such that each simplex of $\Sigma$ is in at most
one pair of $V$. A Forman function $F$ induces a discrete vector
field $V_F$, called a discrete gradient Morse vector field (or, sim-
ply, a Forman gradient). In Figure 1(a) a scalar field defined on
a simplicial complex is shown. In Figure 1(b) a Forman func-
tion is defined which extends the values at the vertices to each
dge (green numbers) and triangle (red numbers). In Figure
1(c), the corresponding Forman gradient is built pairing each k-
simplex with the $(k + 1)$-simplex having the same value. Those
simplices that remain unpaired (vertex 1, edge 6 and triangle 8)
are the critical ones.

Many algorithms have been defined for building a Forman
gradient field on a simplicial or cubical complex by starting
from a function given at the vertices of the complex. Most of
such algorithms avoid the computation of the Forman function
and provides directly the gradient. In our work we have used
the algorithm described in [8] adapted for simplicial complexes.
The interested reader is referred to [9] for a survey of algorithms
for computing a Forman gradient field.

A V-path (or gradient path) is a sequence $[\sigma_1, \tau_1], [\sigma_2, \tau_2], \ldots, [\sigma_r, \tau_r]$ of pairs of k-simplices $\sigma_i$
and $(k + 1)$-simplices $\tau_i$, such that $\sigma_i, \tau_i \in V$, $\sigma_{i+1}$ is a face of
$\tau_i$, and $\sigma_i \neq \sigma_{i+1}$. A V-path with $r > 1$ is closed if $\sigma_1$ is a face
of $\tau_r$, different from $\sigma_{r-1}$. It can be proven that a discrete vector
field $V$ is the gradient vector field of a discrete Morse function
$F$ if and only if $V$ has no closed V-paths. Given two critical
simplices $\tau$ and $\sigma$, we call separatrix V-path between $\tau$ and $\sigma$
each sequence $[\tau, (\sigma_1, \tau_1), (\sigma_2, \tau_2), \ldots, (\sigma_r, \tau_r), \sigma]$ such that all
the pairs form a V-path from a face $\sigma_1$ of $\tau$ to a coface $\tau_r$ of $\sigma$.
Given two critical simplices $\tau$ and $\sigma$, we define the multiplicity
of the incidences between $\tau$ and $\sigma$ to be the number $\mu(\tau, \sigma)$ of
separatrix V-paths between $\tau$ and $\sigma$. In Figure 1(c) we have
two separatrix V-paths between the critical edge 6 and the
critical vertex 1. They are identified by following the sequence
of (vertex, edge) pairs starting from the boundary of edge 6 and
ending on vertex 1. The two separatrices are [6, (3, 3), 1] and
[6, (5, 5), (2, 2), 1]. The multiplicity of the incidences between
edge 6 and vertex 1 is two.

Given a Forman gradient $V$ on a simplicial complex $\Sigma$, the
notion of descending and ascending Morse complexes is de-
finite based on the behavior of the gradient arrows of $V$ on $\Sigma$.
Given a k-critical simplex $\tau$, its corresponding descending k-
cell is the collection of the k-simplices of $\Sigma$ belonging to a V-
path starting at $\tau$. Dually, its corresponding ascending $(d - k)$-
cell is the collection of all the k-simplices belonging to a V-path
that converges to $\tau$. The collection of all the descending [ascending] cells form the descending Morse complex $\Gamma_{d}$ [ascending Morse complex $\Gamma_{u}$] (see Figure 2(b-c)). The Morse-Smale (MS) complex $\Gamma_{MS}$ consists of the connected components of the intersection of descending and ascending Morse cells. The 1-skeleton of the Morse-Smale complex $\Gamma_{MS}$ is the subcomplex of $\Gamma_{MS}$ composed only of its 0-cells and 1-cells (see Figure 2(d)).

2.1. Simplifying Discrete Morse complexes

Topology-based simplification of scalar fields [10, 11] is a powerful tool known in literature for removing insignificant features while preserving relevant parts of the data (see Figure 2(e)).

An operator (called cancellation) has been defined in the literature for removing pairs of critical points [2]. The discrete counterpart of this operator has been introduced in [4] and allows the elimination of a pair of critical simplices.

Let $\Sigma$ be a simplicial complex endowed with a Forman gradient $V$. Given two critical simplices $\sigma$ and $\tau$ of $\Sigma$ with dimension $k + 1$ and $k$ respectively, $(\sigma, \tau)$ is a valid cancellation pair for $(\Sigma, V)$ if $\mu(\tau, \sigma) = 1$ i.e., if the two simplices are connected through a unique separatrix $V$-path.

Under such assumption, $k$-cancellation($\sigma, \tau$) is the operator which removes the critical simplices $\sigma$ and $\tau$, reversing the gradient arrows along the unique separatrix $V$-path from $\sigma$ to $\tau$. More precisely, if $[\tau, (\sigma_1, \tau_1), (\sigma_2, \tau_2), \ldots, (\sigma_r, \tau_r), \sigma]$ is a separatrix $V$-path, a new $V$-path on $\Sigma$ is created as $[(\sigma, \tau), (\sigma_1, \tau_{r-1}), \ldots, (\sigma_r, \tau_1), (\sigma_1, \tau)]$. The Forman gradient $V'$ obtained in this way is still a Forman gradient on $\Sigma$ with the same critical simplices with the exception of $\sigma$ and $\tau$.

Figure 3: Effect of the 1-cancellation($\sigma, \tau$) on a Forman gradient $V$ defined on a triangle mesh. The original $V$ (left side) has two critical triangles $\tau$ and $\tau'$ (in red) and one critical edge $\sigma$ (in green). Red arrows indicate the $V$-path involved in the simplification.

Figure 3 shows the effect of 1-cancellation($\sigma, \tau$) on a Forman gradient $V$ defined on a triangle mesh $\Sigma$: $\sigma$ is a critical edge and $\tau$ and $\tau'$ are two critical triangles. Starting from $\sigma$, the separatrix $V$-path, connecting $\tau$ to $\sigma$, is reversed. As a consequence, $\tau$ and $\tau'$ are not critical. The two separatrix $V$-paths, connecting $\tau$ to $\alpha_1$ and $\alpha_2$, are extended with the reversed $V$-path, and now connect $\tau'$ to $\alpha_1$ and $\alpha_2$. The two separatrix $V$-paths starting from $\sigma$ and reaching $\rho_1$ and $\rho_2$ become non-separatrix $V$-paths.

3. Related Work

In the literature, several strategies have been proposed for topologically simplifying the morphological representation of a dataset [12].

The problem of simplifying a Morse-Smale complex has been addressed in 2D [3, 13, 14], 3D [6, 15] and in nD [16]. A common characteristic of all simplification algorithms is the ordering of the available simplifications based on a priority schema. Priority measures the importance of pairs of critical points which are candidate for deletion, and is defined in such a way to cause the removal of less important critical pairs first.

Algorithms have been proposed based on different priority measures. Persistence [10] is the most widely used; it estimates the absolute difference of function values between the two points.

More recently, other methods for measuring the importance of pairs of critical simplices have been proposed with the purpose of taking into account also the geometry of the underlying simplicial or cubical complex, namely separatrix persistence [14, 17] and topological saliency [18].

We distinguish between two types of algorithms for simplifying an MS complex: algorithms working on a graph-based representation of the complex [13, 6, 19] (also called explicit methods), like the Morse incidence graph discussed in Section 4, and algorithms based on the Forman gradient [14, 15, 20] (also called implicit methods). All algorithms for 2D scalar fields are equivalent in the sense that they can produce the same simplification sequence, and the resulting simplification process is monotonic, i.e., after each simplification, all the new simplifications have higher persistence value. Differences arise when working in three or higher dimensions (see [5] and Section 5).

In [13, 6], two data structures have been defined implementing the graph representation for triangle meshes and for cubical complexes, respectively. In both cases, geometric attributes of the Morse cells and of the 1-skeleton of the MS complex are explicitly encoded (i.e. vertices, edges, triangles and voxels forming such cells). Simplifications are performed by deleting nodes in pairs and merging together the geometrical representations of the Morse cells. In [19] a lightweight version of the same structure has been used encoding only the d- and 0-cells of the two Morse complexes, all the other cells being retrieved by intersection. However, since this latter operation is particularly time-consuming, the resulting data structure is less significative for practical usage.

Algorithms defined in [14, 20, 15] take full advantage of the Forman gradient for defining a simplification algorithm with a low storage consumption. In [14, 20] simplifications are performed on the Forman gradient defined on a 2D regular grid and on a 2D simplicial complex, respectively. In [15] a similar simplification algorithm is implemented for the Forman gradient defined on a 3D regular grid. Due to the inconsistency problem, that we will discuss in Section 5, the
Two kinds of representation are used for Morse complexes: a graph-based representation \([10, 13, 6]\), which explicitly encodes the cells of the Morse complexes and their topological relations, and one based on the encoding of the Forman gradient, which represents such relations implicitly \([15, 5]\). We motivate why the implicit representation is preferable when aiming at a compact data structure for simplifying Morse complexes. Moreover, we propose a new compact graph-based representation coupling the efficiency of the explicit graph-based representation and the compactness of gradient-based one.

**Gradient-based representation.** The standard gradient-based representation encodes the arrows defining a Forman gradient field \(V\) on a simplicial or cubical complex \(\Gamma\). Since a Forman gradient field is a collection of pairs of cells on \(\Gamma\), we need a representation for \(\Gamma\), in which all cells and their mutual incidence relations are explicitly encoded, as in the Incidence Graph \((IG)\) \([21]\). This latter is the most common and general data structure for cell complexes, being an implementation of the Hasse diagram of the complex. The Forman gradient \(V\) can be implemented in a straightforward way on an IG as a Boolean function associated with the arcs of the IG. For a cubical complex, the arcs of the IG are encoded implicitly by indexing the cells of the complex. Moreover, since \(V\) defines a pairing between incident cells, \(V\) is encoded as a bit vector based on the same indexing \([15]\).

For simplicial complexes, a data structure encoding all simplices and their incidence relations is too verbose. The most compact data structures for simplicial complexes only encode the vertices and the top simplices, i.e., those simplifies which are not on the boundary of any simplex \([22]\). Such data structures make the computation of the Forman gradient and of the Morse and Morse-Smale complexes on simplicial complexes of large size feasible. In \([23]\), an encoding for the Forman gradient, which associates the gradient pairs with the top simplices, has been defined for tetrahedral meshes. This compact gradient encoding associates with a tetrahedron \(\sigma\) a subset of the pairs involving its faces (i.e., triangles, edges and vertices). All the gradient arrows inside each tetrahedron can be represented with just two bytes.

**Graph-based representation.** The graph representation is the so-called Morse Incidence Graph \((MIG)\), which is a weighted graph \(G = (N, E, \mu)\) in which: the set of nodes \(N\) is partitioned into \(d + 1\) subsets \(N_0, N_1, \ldots, N_d\), such that there is a one-to-one correspondence between the nodes in \(N_0\) (k-nodes), the \(k\)-cells of the descending complex \(\Gamma_d\), and the \((d - k)\)-cells of the ascending complex \(\Gamma_e\). Each arc in \(E\), connecting a k-node \(\sigma\) to a \((k + 1)\)-node \(\tau\), represents the incidence relation between the Morse cells corresponding to \(\sigma\) and \(\tau\), and is labeled with the number \(\mu(\tau, \sigma)\) of times that \(k\)-cell \(\sigma\) is incident into \((k + 1)\)-cell \(\tau\). Thus, the MIG is an incidence-based representation of the two Morse complexes and provides also a combinatorial representation of the 1-skeleton of the Morse-Smale complex. In the applications, attributes are attached to the nodes in \(N\) storing the geometric information associated with the Morse cells. In Figure 4, an example of a 2D MIG is shown, representing the combinatorial structure of the 1-skeleton of the MS complex depicted in Figure 2(d).

In \([6]\), an extended MIG has been defined storing the cells of both the ascending and descending Morse complexes explicitly. We discuss here such representation for the case of tetrahedral

Figure 2: (a) For a terrain dataset function \(f\) corresponds to the height function and its critical points are peaks (red dots), saddles (green dots) and pits (blue dots). (b) Descending Morse complex decomposes the terrain in a collection of 2-cells in one to one correspondence with the peaks while (c) the 2-cells forming the ascending Morse complex are in one-to-one correspondence with the pits. (d) The separatrices lines for a terrain dataset always connect a saddle with a maximum or a saddle with a minimum. (c) Effects of topological simplification performed on the 1-skeleton of the Morse-Smale complex shown in (d). Note that function values (height values of the terrain) are not modified by the topological simplification; the simplified 1-skeleton represents the two main peaks and the pit only.

Figure 4: The MIG computed on the terrain dataset shown in Figure 2(e). The nodes of the graph are the maxima (red nodes), saddles (green nodes) and minima (blue nodes) of the scalar field function. Arcs (black lines) connect two nodes if there exist a separatrix line connecting the corresponding critical points. Nodes corresponding to maxima are enhanced with the geometrical representation of the corresponding descending 2-cells (relation depicted with red lines) while minima nodes refer to the ascending 2-cells (relation depicted with blue lines).
5. Simplifying an MIG

The Morse Incidence Graph (MIG) can be simplified by adapting the cancellation operator. We consider an MIG $G = (N,E,\mu)$, and a pair of nodes $\tau$ and $\sigma$ in $N$ of dimension $k + 1$ and $k$, respectively, connected through an arc in $E$. We denote as $A = \{x_i, i = 1, \ldots, i_{max}\}$ the $k$-nodes of the MIG different from $\sigma$ and connected to node $\tau$, and as $B = \{y_j, j = 1, \ldots, j_{max}\}$ the $(k + 1)$-nodes of the MIG different from $\tau$ and connected to the node $\sigma$.

As investigated in [5], the simplification of the same pair of critical simplices performed on an MIG and on the corresponding Forman gradient may give different results on the connectivity of the critical simplices when working in three dimensions or higher. We illustrate this problem by using the example in Figure 6. Recall that the weighted arcs in the MIG are in correspondence with the separatix V-paths in the Forman gradient. Figure 6(a) shows a cancellation applied to delete 1-saddle $\sigma$ and 2-saddle $\tau$ on the MIG. As a result of the cancellation, all the arcs connected to either $\sigma$ or $\tau$ are deleted, and the new arcs introduced connect nodes which were previously connected with $\sigma$ and $\tau$. In Figure 6(b), the same configuration is depicted on a Forman gradient showing the separatix V-paths between the critical simplices connected to $\sigma$ and $\tau$. When permuting the same cancellation as before, the arrows in the separatix V-path between $\sigma$ and $\tau$ are swapped. As a consequence, following the gradient arrows originating from the remaining 2-saddles (purple triangles), the new separatix V-paths will end
at the two 1-saddles (green edges), on the left, only. The MIG configuration extracted from the original Forman gradient and the one simplified are shown in Figure 6(c).

We can observe that this situation occurs each time a cancellation involves a separatrix V-path originating from different critical simplices and converging to different critical simplices, which merge and split in a common V-path, that we call a shared V-path. More precisely, a V-path \( \pi \) is called a shared V-path if it is contained in at least two separatrix V-paths \( \pi', \) between \( \tau' \) and \( \sigma' \), and \( \pi'' \), between \( \tau'' \) and \( \sigma'' \), such that \( \tau' \neq \tau'' \) and \( \sigma' \neq \sigma'' \).

6. Shared V-paths and the remove operator

In [16], a dimension-independent simplification operator, called \( k\text{-remove}(\sigma, \tau) \), has been defined for limiting the number of new arcs introduced in the MIG during a \( k\text{-cancellation}(\sigma, \tau) \), since this latter deletes two nodes (the two critical points) but it is likely to increase the number of mutual incidences among critical cells represented as arcs in the MIG. When \( k = 0, d - 1 \), \( k\text{-remove}(\sigma, \tau) \) is equivalent to \( k\text{-cancellation}(\sigma, \tau) \). When \( 1 < k < d - 1 \), \( k\text{-remove}(\sigma, \tau) \) operator can be considered as a cancellation with stronger feasibility conditions.

A \( k\text{-remove}(\sigma, \tau) \) collapses a \( k \)-saddle \( \sigma \) and a \((k + 1)\)-saddle \( \tau \), that are connected through a unique separatrix V-path, if there is at most one \( k \)-saddle, different from \( \sigma \), connected with \( \tau \), or at most one \((k + 1)\)-saddle, different from \( \tau \), connected with \( \sigma \). Since the similarity between a \( k\text{-remove}(\sigma, \tau) \) and a \( k\text{-cancellation}(\sigma, \tau) \), we can describe the effects of a \( k\text{-remove}(\sigma, \tau) \) on the MIG as a \( k\text{-cancellation}(\sigma, \tau) \) in which \#A \( \leq 1 \) or \#B \( \leq 1 \) (see Section 5). Its effect in terms of updates on the Forman gradient or on the MIG are the same as the cancellation operator.

When feasibility conditions are not satisfied, i.e., when \#A > 1 and \#B > 1, a suitable sequence of extremum-saddle operators is performed to obtain a valid configuration for \( k\text{-remove}(\sigma, \tau) \). Such sequence of simplifications forms a macro-operator. As an example, we consider the macro-operator which collapses a \( 2 \)-saddle \( \tau \) and a \( 1 \)-saddle \( \sigma \) into another \( 2 \)-saddle \( \tau' \) (see Figure 7). For all the \( 2 \)-saddles \( \beta_i \) connected to \( \sigma \) and different from \( \tau \) and \( \tau' \), a \( 2\text{-remove} \) involving \( \beta_i \) is performed. When \( \tau \) and \( \tau' \) are the only \( 2 \)-saddles connected to \( \sigma \), the \( 1\text{-remove}(\sigma, \tau) \) is performed.

Because of the similarity between \( k\text{-remove}(\sigma, \tau) \) and \( k\text{-cancellation}(\sigma, \tau) \), the remove operator is still affected by the problems of inconsistencies arising when performing the graph-based or the gradient-based simplification. However, it guarantees a fundamental property that makes \( k\text{-remove}(\sigma, \tau) \) the first ingredient for our simplification algorithm: a \( k\text{-remove}(\sigma, \tau) \) never introduces shared V-paths in \( V \).

Proposition 1. Let \( \Sigma \) be a simplicial complex endowed with a Forman gradient \( V \), which does not contain any shared V-path. Let \( (\sigma, \tau) \) be a valid cancellation pair for \( (\Sigma, V) \), let \( V' \) be the Forman gradient obtained from \( V \) by applying \( k\text{-cancellation}(\sigma, \tau) \). Then, \( V' \) does not contain any shared V-path if and only if \( k\text{-cancellation}(\sigma, \tau) \) is a feasible \( k\text{-remove}(\sigma, \tau) \) for \( (\Sigma, V) \).

Let us assume that \( k\text{-remove}(\sigma, \tau) \) is feasible for \( (\Sigma, V) \). By hypothesis, \( V \) has no shared V-path. Thus, any shared V-path in \( V' \) should be contained in one of the separatrix V-paths newly created by \( k\text{-remove}(\sigma, \tau) \). Since \( k\text{-remove}(\sigma, \tau) \) is feasible for \( (\Sigma, V) \), at least one of the sets \( A \) and \( B \) has cardinality equal to one, so no shared V-path can be created in \( V' \).

An example of this is shown in Figure 8. Since 1-
\text{remove}(\sigma, \tau) \) is feasible, at most one simplex \( \alpha_i \) of the same dimension of \( \tau \) is connected to \( \sigma \). Thus, the new created V-paths cannot be shared V-paths since they will have a common origin (i.e., \( \tau_i \)).
the first simplex of π which belongs to a separatrix V-path between βj ∈ B and σ. Dually, let τm be the last simplex of π which belongs to a separatrix V-path between αi ∈ A and τ. Since V has no shared V-path, m < l and each newly created separatrix V-path between βj and αi will contain the V-path π′ = [(σ1, τ1), . . . , (σm+1, τm)]. Since both #A and #B are greater than 1, π′ is a shared V-path for (Σ, V).

Conversely to the example shown in Figure 8, the configuration depicted in Figure 9 is not valid for 1-remove(σ, τ) since multiple 2-saddles are connected with σ (i.e. α1, α2 and α3). As a result of applying 1-cancellation(σ, τ) we introduce a shared V-path, depicted in red, created overlapping the new V-paths having different origin and destination.

7. Shared V-path disambiguation algorithm

In this section, we propose a preprocessing step aimed to untie the shared V-paths in a tetrahedral mesh Σ endowed with a Forman gradient V. The idea at the basis of the shared V-path disambiguation algorithm is to modify the separatrix V-paths between 1-saddles and 2-saddles, inserting new dummy critical simplices in such a way that all the separatrix V-paths sharing the same path will end (or start) at the same critical saddle.

When looking at the separatrix V-paths connecting maxima with 2-saddles and minima with 1-saddles, this property is guaranteed by construction, i.e., V-paths starting from a maximum can only split, while V-paths reaching a minimum can only merge.

Figure 10 illustrates the key ideas of the algorithm. The traversal starts from critical edge σ and continues visiting the triangles in the separatrix V-path by navigating the arrows in reverse order. At triangle τ1, three separatrix V-paths split, then the triangle is identified as part of the shared path. Continuing the traversal, on edge σ1 different separatrix V-paths merge. Thus, σ1 is identified as the beginning of the shared path, and τ1 and σ1 are introduced as critical (see Figure 10(b)).

Algorithm 1 shows the pseudocode description of the algorithm for disambiguation of V-paths. Starting from a critical edge σ, the separatrix V-paths converging in it are considered (lines 2-4). For each separatrix V-path, the first triangle incident into σ and belonging to the path is pushed onto a stack S (lines 4-6). While S is not empty, the first triangle τi is popped from the stack and the number of separatrix V-paths outgoing from its boundary edges are computed (function countS splittingS separatrix(τi)). If there are multiple separatrix V-paths that split at τi (see τ1 in Figure 10(a)), the visit of a shared path begins (line 11).
Algorithm 1 IdentifySharedPath(V)

1: INPUT: V is an discrete gradient field
2: for all critical edges σ in V do
3: \( F := \text{startingVPaths}(\sigma); \)
4: for all triangles \( \tau_i \) in \( F \) do
5: Stack \( S := \emptyset \)
6: \( S.push(\tau_i); \)
7: while \( S.notEmpty(); \) do
8: \( \tau_i := S.pop(); \)
9: \( \text{nSplit} := \text{countS splittingSeparatrix}(\tau_i, V); \)
10: if \( \text{nSplit} > 1 \) then
11: \( \sigma_f := \text{visitSharedPath}(\tau_i, V); \)
12: \( F := \text{adjacentPaired}(\tau_i, V); \)
13: for all triangles \( \tau_j \) in \( F \) do
14: \( S.push(\tau_j); \)

Algorithm 2 VisitSharedPath(\( \tau_i, V \))

1: INPUT: \( \tau_i \) is a triangle
2: INPUT: \( V \) is an discrete gradient field
3: \( \sigma_i := V.getFePair(\tau_i); \)
4: \( F := \text{adjacentPaired}(\sigma_i, V); \)
5: if \( \#F = 1 \) then
6: \( \tau_j := F; \)
7: \( \text{nSplit} := \text{countS splittingSeparatrix}(\tau_j, V); \)
8: if \( \text{nSplit} > 1 \) then
9: // if a new splitting face is found \( \tau_i \) is updated
10: \( \tau_i := \tau_j; \)
11: return \( \text{visitSharedPath}(\tau_j, V); \)
12: if \( \#F > 1 \) then
13: \( \text{reversePath}(\sigma_i, \tau_i, V); \)

Algorithm 2 describes the traversal of a shared V-path. Starting from the triangle \( \tau_i \) on which the shared V-path splits, the edge \( \sigma_i \) paired with it, is extracted (line 3). Function \( \text{adjacentPaired} \) returns the set \( F \) of triangles different from \( \tau_i \) and incident in \( \sigma_i \) that are in some separatrix V-path (line 4). If \( F \) has cardinality equal to one, we are still visiting the shared V-path (line 5). Otherwise, if the cardinality of \( F \) is greater than one, we are on an edge \( \sigma_j \) on which multiple separatrix V-paths are collapsing (see \( \sigma_1 \) in Figure 10(a)). If this is the case, \( \tau_i \) and \( \sigma_i \) are introduced as dummy critical simplices and the arrows between them are reversed (lines 12-13). If \( \#F \) was zero, we ended into a single critical triangle, thus we were not on a real shared V-path and no critical simplices are introduced. Note that, during the visit of a shared V-path, triangle \( \tau_i \) can be updated if another triangle, closer to \( \tau_i \), is found on which separatrix V-paths split (lines 7-10).

Algorithm 2 has a linear time complexity in the number of simplices in the identified shared V-path. Algorithm 1, instead, visits all the separatrix V-paths once for each 1-saddle. Thus, it has a worst-case time complexity of \( O(s_1 \cdot s_V) \), where \( s_1 \) is the number of 1-saddles and \( s_V \) the number of simplices forming the separatrix V-paths.

Once all shared V-paths have been identified and disambiguated, we perform a simplification step for removing all the dummy critical simplices. Since the insertion of a pair of critical simplices \( (\sigma, \tau) \) can be seen as the undo of a cancellation, performing cancellations would restore the initial inconsistency situations in the complex. Thus, we use only remove operators that will trigger macro-operators working on extremum-saddle pairs.

7.1. Dummy critical points and obstructions

Obstructions are critical point configurations that cannot be simplified either using a cancellation or a remove operator. Specifically an obstruction is a pair of critical points, of consecutive index connected by multiple paths. The presence of obstructions can lead to degenerate configurations, called fingers, that cannot be simplified. Such configurations typically do not appear in the initial state of the dataset but arise, with the undergoing of simplifications, in flat areas [24]. Even if flat areas are not allowed, when computing a Forman gradient with the algorithm described in [8], obstructions are still present in the data since they describe the natural behavior of the field.

Let us consider the shared V-paths. When the obstruction is present inside a shared V-path, the introduction of a dummy critical simplex can be avoided (since any simplification passing by that part will be unfeasible). When obstructions involve critical simplices in the neighborhood, there is a degenerate configuration that could prevent the removal of the dummy critical points. We show such configuration in Figure 11. The 2-saddles (purple triangles) and the 1-saddles (green edges) are all connected with extrema (maxima and minima respectively) through multiple paths. Thus, the macro-operator cannot remove the two dummy critical points since none of the 2- or 1-saddles in the neighborhood can be removed. However, it is still important to introduce the pair since otherwise, the shared V-path could be affected by a swap during the simplification algorithm. Note that if this is the case, it means that the dummy pair will be removed in the future.

Dummy critical points that have not been removed during the simplification process can be removed at the end, with a cancellation, avoiding the visualization of spurious cells.

Even if this can be seen as a degenerate problem that could bring to uncontrolled results, it is important to notice that the introduction of a dummy pair never inhibits the application of other remove operators. In other words, the number of remove operators, preserving the shared V-paths, that can be applied on a Forman gradient V without a dummy pair, is always less or equal to the number of remove operators that can be applied on V after the insertion of the dummy pair. This is important to guarantee that the simplification is never obstructed by our disambiguation method.
8. Experimental results

We have combined the shared V-path disambiguation algorithm with a simplification algorithm based on the remove operator. In this section, we discuss the results obtained when simplifying real datasets. Experiments have been performed on a desktop computer with a 3.2Ghz processor and 16GB of memory. Datasets chosen for our experiments are originated from regularly distributed data. The unstructured tetrahedral meshes are obtained by removing points (and tetrahedra) corresponding to the empty space and removing flat areas (adjacent vertices with the same field value) through edge contractions.

We use a Discrete Morse Incidence Graph (DMIG) (see Section 4) for representing the pairs of critical simplices connected by a separatrix V-path. remove operators are applied in ascending order of persistence using a priority queue. At each step, the simplification with the lowest persistence value is performed, the gradient arrows along the path are updated as well as the DMIG, and the new available simplifications are inserted in the priority queue. Once the queue is empty, or all the valid simplifications have a persistence value higher than a user defined threshold, the simplification algorithm ends.

Table 2: Evaluation of the preprocessing step and the remove-based simplification. For each dataset we indicate, the original size and the number of vertices, tetrahedra and critical points (columns Size, Σ0, Σ1 and #C respectively) in the tetrahedral mesh. In column Preprocessing, we show the number of critical points introduced during the preprocessing step and the timings for: identifying the shared V-paths, insert the critical points and remove them. Column Simplification shows the total number of simplifications performed and the time required by the algorithm. Column Mem. Peak indicates the maximum amount of memory used.

| Dataset      | Size | | | #C | Preprocessing | | Simplification | Mem. Peak |
|--------------|------|-----|-----|-----|---------------|---------------|-------------|
|              |      | Σ0  | Σ1  |     | #Cins         | Rem Time      | Time        | (GB)       |
| BUCKY        | 32³  | 32K | 0.17M| 2K  | 156 2.4 sec   | 1K 6.39 sec   | 0.09        |
| FUEL         | 64³  | 13K | 0.06M| 2.7K| 54 0.65 sec   | 1.3K 4.13 sec | 0.15        |
| SILICIUM     | 98³  | 66K | 0.36M| 2.1K| 290 1.6 sec   | 1K 17.5 sec   | 0.1         |
| NEGHIP       | 64³  | 0.12M| 0.64M| 12.6K | 234 10.7 sec  | 6.3K 3.8 min | 0.2         |
| SHOCKWAVE    | 64³  | 1.2M| 7M  | 1.1K| 55 20.1 sec   | 582 2.8 sec   | 2.4         |
| BLUNT        | 256³ | 1.0M| 6M  | 11.2K| 1378 10.4 min | 5.5K 22.2 min | 1.9         |
| HYDROGEN     | 128³ | 0.6M| 3.9M| 15.1K| 2133 24.1 min | 7.5K 24.3 min | 2.2         |

We have studied the preprocessing step by evaluating its impact on the overall computation. In Table 2, we present the results obtained. We can notice that the number of critical simplices artificially introduced (column #C_ins) varies depending on the dataset and is between 2-13% of the total number of critical simplices and all of the are removed during this phase. The timings of the preprocessing algorithms can be relevant with respect to the whole simplification process and, in a worst-case scenario (HYDROGEN), the time required for identifying and disambiguating shared V-paths and removing the dummy critical simplices is equal to the time required for simplifying the entire mesh. The complexity of the preprocessing step depends on the number of separatrices V-paths between saddles and on their size, i.e., on the number of simplices forming them. In Figure 13, we show the results obtained by simplifying FUEL, BUCKY, NEGHIP and HYDROGEN tetrahedral meshes. For HYDROGEN mesh, we can notice that shared V-paths are quite numerous and spread around the entire mesh, unlike what happens with FUEL, BUCKY and NEGHIP.

We have also studied the remove operations triggered by the macro-operators during the removal of the dummy critical simplices. Specifically, we focus on studying the persistence associated with the deleted nodes in order to ensure that interesting features were not deleted during the preprocessing step. As discussed in [5], there is a correlation between noise...
Figure 13: Topologically-consistent simplification of the FUEL, BUCKY, NEGHIP and HYDROGEN. The original scalar field (a) and the shared paths depicted in red (b). The original 1-skeleton of the MS complex (c) and its simplified version (d) computed with a persistence threshold of 0.01% with respect to the maximum persistence for FUEL, 0.2 for BUCKY and HYDROGEN and 0.3% for NEGHIP.
and shared V-paths. We have found that 98% of the removals applied during the preprocessing step delete nodes that would be removed by the classical algorithm using a persistence threshold lower than 0.01% of the maximum persistence. Nodes in the remaining 2% have a persistence lower than 0.1% of the maximum persistence. Typically, values of persistence lower than 0.2% of the maximum persistence are considered noise.

Studying the entire simplification algorithm, we have verified experimentally the correctness of our approach comparing the graph updated during the simplification process and the one extracted from the simplified Forman gradient after each simplification step.

Moreover, we have compared our remove-based simplification algorithm with a standard cancellation-based algorithm testing whether the number of critical simplices in the fully simplified Forman gradient are comparable. The graph depicted in Figure 12 shows the number of critical simplices deleted using different simplification errors. For each mesh, the column on the right indicates the results obtained with the remove-based algorithm, while the results obtained with the cancellation-based algorithm are shown in the columns on the left. As we can notice, the number of critical simplices removed is comparable in both approaches. This result guarantees that the simplification sequence obtained using the remove operator removes features in a controlled and progressive way, as the cancellation-based method.

9. Concluding remarks

We have presented a new simplification algorithm for a discrete Morse gradient that guarantees the topological consistency of the Morse and Morse-Smale complexes generated from the simplification. The algorithm works on a new compact graph-based data structure representing such complexes efficiently with a minimum loss in storage cost. We have proved the correctness of our approach, and we have evaluated experimentally its performances with respect to a classical cancellation-based approach. Note that the remove operators and the Forman gradient have been defined in a dimension-independent way, and also the gradient encoding proposed is dimension-independent. A further development of the work presented here is to apply the proposed simplification approach in higher dimensions for computing homology and persistent homology efficiently.

The algorithm proposed here is the basis for a tool on which both geometric and morphological simplifications can operate concurrently to reduce the complexity and to enhance the understanding of available volume datasets. In our future work, we plan to combine the simplification algorithm proposed with a simplification of the underlying tetrahedral mesh which does not affect the critical simplices, thus being able to control also geometric resolution.

In the applications, however, topological simplification cannot be considered as a suitable tool on its own. In several contexts, multi-resolution models are preferable to produce an interactive framework for scientists and domain experts.

Generally speaking, a multi-resolution model is the basic tool for producing different representations of a spatial object at different levels of detail, which can be uniform or vary over the object. In order to define a multi-resolution topological model based on the Forman gradient, we need to encode a coarse Forman gradient \( V \) (i.e., the gradient with the minimum number of critical simplices) obtained from the initial gradient through a sequence of simplifications, a set of refinement operators and a dependency relation between such operators.

Each refinement operator will be the inverse of a remove, thus introducing a pair of critical simplices under the same assumptions as in remove. The dependency relation will make a refinement operator introducing two critical simplices \( \sigma \) and \( \tau \) depend on all refinements in the multi-resolution model which introduce critical simplices to which \( \sigma \) and \( \tau \) need to be connected. If we use an implicit representation, because of the undecidability introduced by the shared V-paths, these conditions could not be determined a priori but they would have to verified on the fly, navigating the Forman gradient, before each refinement. The resulting loss of efficiency would make the model useless for an interactive experience. This is the reason why our proposed approach is fundamental for designing and implementing a topological multi-resolution model.

Finally, inspired by the work done in [20] for the 2D case, we plan also to adapt our data structure for working with the spatio-topological index there defined, the PR-star tree. This would lead to a distributed approach for the simplification algorithm as well as to a consistent reduction in the storage cost for encoding the underlying complex on which the Forman gradient is defined.

Acknowledgements

The data sets used are courtesy of http://www.volvis.org/. This work has been partially supported by the National Science Foundation under grant number IIS-1116747.

References
