Hierarchical Forman Triangulation: a Multiscale Model for Scalar Field Analysis

Federico Iuricich[a], Leila De Floriani[a]

[a]*University of Maryland, College Park, MD, USA*

Abstract

We consider the problem of analyzing the topology of scalar fields defined on a triangulated shape by using a multi-scale approach, which allows reducing storage costs and computation times, and supports interactive inspection and classification of topological features. We define and implement a multi-scale model that we call a *Hierarchical Forman Triangulation (HFT)*, where a 3D shape or a terrain is discretized as a triangle mesh, and its topology is described by defining a discrete Morse gradient field based on function values given at the vertices of the mesh. We introduce a new edge contraction operator, which does not change the behavior of the gradient flow and does not create new critical points, and we apply it in combination with a topological simplification operator which eliminates critical elements in pair. By combining the two operators in a sequence, we generate the HFT. We discuss and implement a compact encoding for the HFT that has a lower storage cost with respect to the triangle mesh at full resolution. We show the effectiveness of this new hierarchical model by extracting representations of terrains and shapes endowed with a scalar field at different, uniform and variable, scales and by efficiently computing topological features and segmentations.

Keywords: Topological data analysis, Simplification, Multi-scale models, Discrete Morse theory

1. Introduction

Computational topology is a rapidly developing field in data and shape analysis. It is used to support classification and understanding combined with machine learning techniques [1] and as the basis for interactive analysis and inspection through visualization [2]. Topological tools are rooted in Morse theory and persistent homology. This latter has produced shape signatures, like the barcode or the persistent diagram, while the former has been the basis for extracting topological features, like the Reeb graph, which describes the evolution of the level sets of a scalar function defined on a manifold shape, or the Morse and Morse-Smale complexes, which provide a segmentation of a shape induced by the regions of influence of the critical points of a scalar function defined on it [3]. These latter have been extensively used for terrain analysis [4], shape analysis [5, 6] or remeshing [7].

The purpose of our work is extracting Morse features, like the ascending and descending manifolds forming the Morse complexes, efficiently and effectively. We consider a terrain or a 3D shape discretized through a triangle mesh, with a scalar value associated with its vertices. Because of the large size of the meshes, most of the recent approaches to extract topological features from data are based on a discrete version of Morse theory for cell and simplicial complexes [8], which allows an entirely combinatorial and derivative-free approach to Morse feature computation.

Since data are affected by noise, many spurious critical points and cells can be generated. Simplification approaches have been defined for dealing with both noise and data redundancy [9, 10]. Simplifying a scalar field using topological simplifications means canceling critical points in pairs, thus reducing the number of cells in the Morse complexes. Simplification approaches have been recognized as effective but not efficient, especially to support data inspection and understanding through visualization. Multi-resolution approaches have been introduced for providing faster interactions and more degrees of freedom on the extracted representations [11, 12, 13]. Most of these models interact with the morphology but leave the underlying mesh untouched. This is a serious issue when working with big datasets since the complexity of extracting, representing and visualizing Morse features is mainly affected by the resolution of the mesh and not by the size of the Morse complexes.

Our work is inspired by [14] where the authors introduce the first edge contraction operator for a triangle mesh endowed with a scalar field which maintains the Forman gradient. The operator is used to create a progressive model, consisting of a sequence of simplifications of the mesh and of the Forman gradient. On the other hand, we define a multi-scale model for a triangulated shape endowed with a scalar field, that we call a *Hierarchical Forman Triangulation (HFT)*. The HFT allows extracting both a mesh and a topological representation at different levels of topological and geometric resolutions. The model is generated based on two simplification operators: the *edge contraction* operator, which simplifies the mesh, and the *cancellation* operator, which simplifies the morphology. The edge contraction operator that we will introduce in Section 5 is an improvement over the one defined in [14] since it imposes conditions on the Forman gradient V only. Given a triangle mesh Σ endowed with a scalar function given at its vertices, we build a discrete Morse gradient V on Σ compatible with the scalar field; then, through *edge contractions* and *cancellations*, we simplify both Σ and V. The HFT is built from the resulting sequence of simplifications. The inverse of the atomic simplifications used to generate the

HFT together with a partial order dependency relation between pairs of simplifications form the structure of the HFT, from which representations at different scales, also variable across the mesh, can be efficiently extracted. The major contributions of this work are the definition and implementation of:

- a simplification operator for triangle meshes endowed with a Forman gradient, which does not eliminate or generate critical elements, and generalizes the operator presented in [14];
- a new refinement operator, inverse of the latter, that operates on the mesh and on the Forman gradient without creating or eliminating critical elements;
- a new multi-scale model, the Hierarchical Forman Triangulation (HFT), which combines mesh and topological updates, based on discrete Morse theory;
- a dependency relation between topological updates, which is minimal in the number of dependencies required, thus greatly enhancing the expressive power of the multi-scale model.

The HFT has a low storage cost, lower than that of the mesh at full resolution, and provides a high flexibility in adjusting the resolution of the mesh to comply with the scale of the topological representation. This allows extracting variable-scale representations of the mesh endowed with the gradient as well as multi-scale Morse features efficiently.

2. Background notions

Morse theory [15, 16] is a mathematical tool studying the relationships between the topology of a manifold shape M and the critical points of a smooth scalar function f defined over M. Piecewise-linear Morse theory transposes some results from Morse theory to piece-wise linear functions [17]. Here, we focus on a combinatorial counterpart of Morse theory for cell complexes due to Forman and called *Discrete Morse Theory (DMT)* [8]. Since simplicial complexes are common discretization structures [18] for shapes in low and high dimensions, we review DMT focusing only on these latter. Recall that a *k-dimensional simplex*, or simply a *k-simplex*, σ is the convex hull of $k+1$ geometrically independent points in $\mathbb{R}^n$. 0-, 1- and 2-simplices are also called *vertices*, *edges*, and *triangles*, respectively. A *triangle mesh* is a special case of a simplicial complex: it is formed by vertices, edges and triangles and has a manifold domain.

We consider a pair (Σ, F), where Σ is a triangle mesh and $F : \Sigma \rightarrow \mathbb{R}$ is a scalar function defined on all the simplices of Σ. Function F is a *discrete Morse function* (also called a *Forman function*) if and only if, for every k-simplex $\sigma \in \Sigma$, all the $(k-1)$-simplices on the boundary of σ have a lower function value than σ, and all the $(k+1)$-simplices bounded by σ have a higher function value than σ, with at most one exception. If there is such an exception, it defines a pairing of cells, called a *gradient pair*. A gradient pair can be viewed as an *arrow* in which the head is a k-simplex and the tail a $(k-1)$-simplex. A simplex that is not a head or a tail of any arrow is a *critical simplex*. A *V-path* is a sequence of simplices $[\sigma_0, \tau_0, \sigma_1, \tau_1, ..., \sigma_i, \tau_i, ..., \sigma_q, \tau_q]$ such that σ_i and σ_{i+1} are on the boundary of τ_i and (σ_i, τ_i) are paired simplices, where $i = 0, ..., q$.

The collection of all paired and critical simplices of Σ forms a *discrete Morse gradient* (also called a *Forman gradient*) if there are no closed V-paths, i.e., if all V-paths are acyclic. In Figure 1(a)) a Forman gradient is shown: it has two critical triangles (t and t_1), one critical edge e, and one critical vertex v. Given a triangle mesh Σ endowed with a scalar function f defined on its vertices, we can always compute a Forman gradient V without computing the Forman function F explicitly. In our work we compute the Forman gradient through the algorithm in [19]. In the following, we denote a triangle mesh Σ endowed with a Forman gradient V as a pair (Σ, V). We call a *separatrix V_j-path* any V-path of the following form: $[\tau, \sigma_0, \tau_0, \sigma_1, \tau_1, ..., \sigma_i, \tau_i, ..., \sigma_q, \tau_q, \sigma]$, where τ and σ are two critical simplices of dimension $j+1$ and j, respectively. Thus, in a triangle mesh Σ we will have separatrix V_0-paths connecting a critical edge to a critical vertex and separatrix V_1-paths connecting a critical triangle to a critical edge (see Figure 1).

Topological features are defined in the discrete case in terms of the Forman gradient and its paths. The *critical net* consists of the critical vertices, edges, and triangles plus the separatrix V_0- and V_1-paths connecting them. Any descending k-manifold (which is a k-cell of the descending Morse complex), is the collection of the k-simplices of Σ reached by the gradient paths starting from critical k-simplex. Dually, an ascending k-manifold (a k-cell of the ascending Morse complex) is the collection of the $(2-k)$-simplices reached by the gradient paths (visited backward) starting from a critical $(2-k)$-simplex. Figure 1(e) illustrates the descending 2-manifold associated with triangle t. A description of the algorithms used for extracting the ascending and descending manifolds from a Forman gradient can be found in Section 8.

3. Related work

Morse complexes can be simplified by applying an operator defined in smooth Morse theory, called *cancellation* [16]. A cancellation removes two critical points of consecutive index which are connected by a separatrix line. This operator has been investigated in 2D [20, 11, 21, 22] and 3D [10, 23], by considering piecewise-linear shape approximations.

Given a sequence of cancellations, a hierarchical model is built by organizing into a hierarchy the refinements which are inverse of such cancellations, each refinement (also called anti-cancellation) performing an undo of the corresponding cancellation. Several hierarchical models exist for representing the morphology of a triangle mesh endowed with a scalar field [3]. They can be classified as: *progressive models*, that just represent the sequence of refinements reversing the cancellation sequence, and *multi-resolution models*, that organize the refinements according to a partial order relation of mutual dependencies among the refinements. These latter have a much higher expressive power, since they support the extraction of a large number of

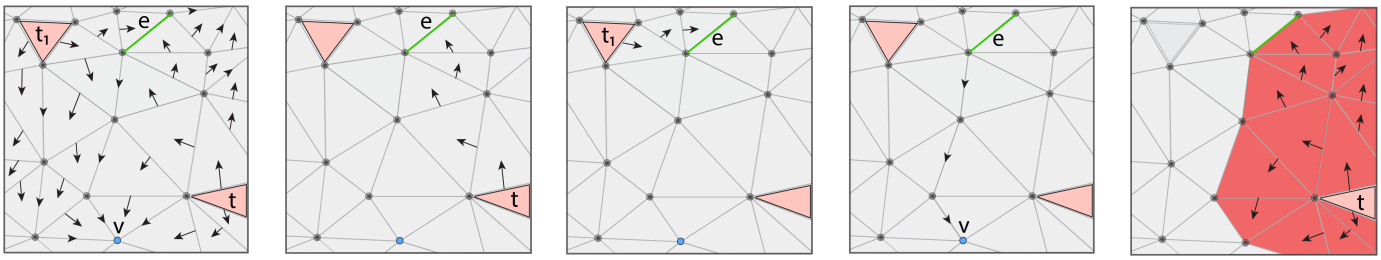


Figure 1: (a) Forman gradient with two critical triangles t and t_1, one critical edge e and one critical vertex v. (b) The separatrix V_1-path connecting e and t. (c) The separatrix V_1-path connecting e and t_1 and the (d) separatrix V_0-path connecting e and v. (e) Descending 2-cell corresponding to the critical triangle t.

representations not encountered during the cancellation process. The first progressive morphological model has been developed in the context of image analysis. The hierarchical approach described in [24] defines a containment hierarchy for the regions of the watershed segmentation computed on the image. In [20] a progressive model is defined for the morphology of a terrain. The hierarchy is created by applying cancellations on the critical net. The hierarchy is encoded as the critical net at the coarsest resolution plus the sequence of *anti-cancellations*, inverse to the cancellations used in the construction phase.

In [11, 4] a *dependency relation* among anti-cancellations is introduced for building a multi-resolution model. In [11], the dependency relation between two refinements is defined in terms of a *diamond*. The diamond associated with an anti-cancellation(q, p) is a quadrangle bounded by the maximum [minimum] p, the two (not necessarily distinct) minima [maxima] connected to saddle q and p', the other maximum [minimum] connected to q different from p. Two refinements are said *mutually dependent* if the associated diamonds have at least one vertex in common. Two refinements are said to be dependent in [4] if they share a pair of critical points.

In [13], a dimension-independent multi-resolution model has been defined and implemented for representing the morphology of an n-dimensional scalar field. The model is generated from the critical net computed on the scalar field at full resolution, by iteratively applying the dimension-independent cancellation operators defined in [23]. This produces a critical net describing the topology of the field at the lowest level of detail. The model organizes several representations of the critical net in a partial-order hierarchy and is capable of supporting the extraction of the net which best approximates the topology of the scalar field under application-dependent requirements.

In all these models the underlying mesh is always kept at full resolution. A first step towards combining mesh simplification with morphological cancellation is in [25], where the multiresolution model consists of three hierarchies, for the mesh Σ, for the critical net and for a purely combinatorial representation of the critical net. These three models are all generated by iteratively applying edge contraction on the triangle mesh. When a simplification only modifies Σ it is encoded in the first hierarchy, when it also modifies the critical net, it is encoded in the second hierarchy. When it deletes a critical point, it is encoded in all three of them. Besides the problems arising from encoding and navigating three different hierarchies, the major drawback of this approach is the fact that an edge contraction applied to Σ may generate new critical points.

4. Overview of the approach

We consider a triangle mesh Σ endowed with a Forman gradient V, denoted as (Σ, V). The gradient field V is computed based on the values of a scalar field given at the vertices of Σ. We apply a sequence of simplification operators S by starting from (Σ, V) until reaching the coarsest representation, i.e., the pair (Σ_B, V_B) on which no more simplifications can be applied. We call Σ_B the *base mesh* and V_B the *base Forman gradient*. We consider two simplification operators:

- a new simplification operator, that we call *gradient-aware edge contraction* which operates on Σ by reducing the number of simplices and on V by modifying the gradient without altering its number of critical simplices, as we prove in Section 5.

- a cancellation operator on V, which removes a pair of critical simplices from V without modifying the simplices in Σ; this is a classical gradient simplification operator defined in discrete Morse theory [8] and it is performed by reversing the gradient arrows in the V-path connecting the two critical simplices, as discussed in Section 6.

We then consider the inverse of the simplification operators, which undo their effect. We define a *gradient-aware vertex-split*, which is the inverse of the gradient-aware edge contraction: such operator modifies triangle mesh Σ and gradient field V without deleting or introducing critical simplices (see Section 5) We define the *insertion* operator as the undo operator with respect to a *cancellation*, and we define and prove the conditions for its feasibility (see Section 6) . As it happens for a *cancellation*, an *insertion* does not modify the triangle mesh Σ, but in order for it to be feasible, some requirements on the triangle mesh to which it can be applied need to be imposed, as discussed in Section 6.

From simplification sequence S, where the gradient-aware edge contraction and the cancellation are interleaved, we obtain a set R of refinement operators (the gradient-aware vertex-split and insertion) which undo the operators in S. We define a *dependency relation* between pairs of refinements that can be proven

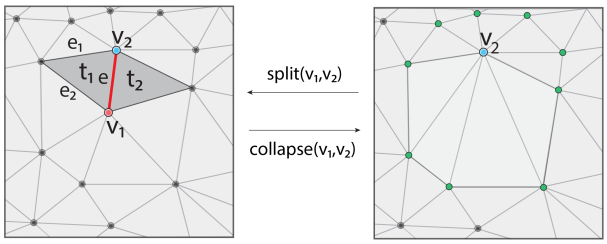


Figure 2: From left to right: we show the result of the contraction of edge e. From right to left: effect of its dual vertex split operator.

to be a partial order, as discussed in Section 7.1. The base mesh endowed with the base gradient (Σ_B, V_B), set R, and the dependency relation define a partial-order hierarchy of refinements, described by a Directed Acyclic Graph (DAG), the *Hierarchical Forman Triangulation (HFT)*. The HFT is a combined geometric/topological multi-scale model for supporting interactive inspection and understanding. In Section 7 we present the data structure for encoding an HFT, and we describe the algorithms for extracting representations at variable scales in Section 8.

5. Mesh simplification and refinement

In this section, we introduce the operator used for simplifying a mesh Σ endowed with a Forman Gradient V, (Σ, V), and its undo (refinement) operator.

5.1. Gradient-aware edge contraction

Edge contraction is a well-known simplification operator for triangle meshes and simplicial complexes in general [26, 27]. An *edge contraction* acts on a triangle mesh Σ by contracting an edge e, with endpoints v_1 and v_2, to one of its endpoints (i.e., v_2). We consider edge e as directed from v_1 to v_2, and we denote as t_1 the triangle on the left of e and as t_2 the one on the right (see Figure 2). The effect of the contraction of edge e on Σ is to remove edge e, vertex v_1 and triangles t_1 and t_2 and to transform all the edges and triangles incident in v_1 into edges and triangles incident in v_2. As discussed in [25], the edge contraction operator can modify the morphology of the scalar field in an uncontrolled way.

We define here a new operator called *gradient-aware edge contraction*, denoted $contract(v_1, v_2)$, on (Σ, V): the operator modifies the simplices in Σ as the edge contraction, but also removes the same simplices from the gradient pairs in V in such a way that the critical simplices remain unchanged. This operator is inspired by the one defined in [14]. As described in the following, the gradient-aware edge contraction operator presented here is an improvement over the latter since it imposes condition on the Forman gradient V only. This is guaranteed by the following feasibility requirement.

A gradient-aware edge contraction, $contract(v_1, v_2)$, can be considered as *feasible* on (Σ, V) if and only if the following conditions are satisfied:

(i) all the simplices to be removed, i.e. edge e, vertex v_1 and triangles t_1 and t_2, are not critical;

(ii) edge e is paired in V with v_1.

This latter condition guarantees that no simplices become critical. Clearly, no critical simplex is removed during simplification because of condition (i). Condition (ii) is required since otherwise an edge contraction might force the introduction of a new critical simplex, as shown in the example in Figure 3.

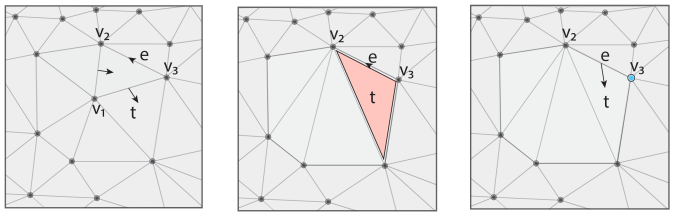


Figure 3: (a) Gradient configuration before edge contraction. After the edge contraction we either (a) keep e paired with vertex v_3, then introducing t as critical or (b) we keep t paired with e and we introduce v_3 as critical.

We now prove that if condition (ii) above holds, then no critical simplex is introduced. We always consider the contraction of edge e as oriented in the same direction as the gradient defined on e (so v_1 is the vertex paired with e). Refer to Figure 2 for notations. We recall that edge e and triangles t_1 and t_2 are removed by edge contraction. We know that e is paired with v_1 and, as effect of the contraction, pair (v_1, e) is removed from V. Because of condition (i), t_1 is not critical. Also, it is not paired with e because of condition (ii). Thus, t_1 has to be paired with either e_1 or e_2. As an effect of the contraction, we will remove t_1 and its paired edge from V. The same holds for t_2. No other simplex is removed from Σ and, since we have been removing simplices in pairs from V, we can conclude that no critical simplex is introduced.

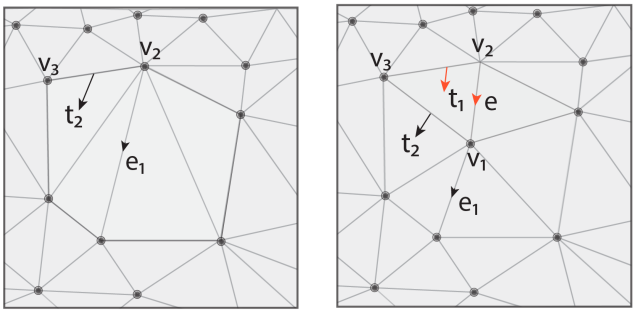
5.2. Gradient-aware vertex split

We define a *gradient-aware vertex split*, denoted $split(v_1, v_2)$, as the undo of $contract(v_1, v_2)$ on (Σ, V). The operator reintroduces vertex v_1 along with edge $e = (v_1, v_2)$ and the two triangles t_1 and t_2 incident in e (see Figure 2). By inserting triangles t_1 and t_2, two new edges are also created. The modification of the gradient consists of first pairing edge $e = (v_1, v_2)$ by looking at the path through v_2:

- if v_2 is paired with an edge e_1 which becomes incident into v_1 after the split, then v_2 is paired with e and v_1 is paired with e_1 (see Figure 4), otherwise e is paired with v_1;
- triangle t_1 is paired with the one edge between (v_1, v_3) or (v_2, v_3) which is unpaired (see Figure 4);
- the same happens for triangle t_2.

As a result of the above, we can easily prove that no critical simplex is deleted or introduced by a gradient-aware vertex split. We observe that:

- triangles t_1 and t_2 are paired with the newly introduced edges,



(a) (b)

Figure 4: (a) Gradient configuration before vertex split and (b) after, where the gradient path passing for e_1 is maintained by pairing v_2 with the new edge (v_2, v_1). Similarly, the new edge introduced is paired with the unpaired edge to maintain the gradient paths consistent.

- edge (v_1, v_2) is paired with either v_1 or v_2,
- v_1 is paired with either (v_1, v_2) or with the edge previously paired with v_2.

We conclude that no new simplex is unpaired after the vertex split, and thus no critical simplex is introduced or removed.

Let $contract(v_1, v_2)$ be the operator applied to a pair (Σ, V) which produces a new pair (Σ', V'), as described in the previous subsection. Let S_2 be the ordered sequence of vertices adjacent to v_2 in Σ' (green vertices in Figure 2). Intuitively, we consider a gradient-aware vertex split $split(v_1, v_2)$ applied to a pair (Σ'', V'') *feasible* if Σ'' is "locally equivalent" to Σ'. This translates into the following two conditions:

- vertex v_2 is in Σ'';
- S_2, the ordered sequence of vertices of Σ' adjacent to v_2, is in Σ''.

This condition will be at the base of one of the dependency relations of our multi-scale model (see Section 7.1).

6. Modifying the topological representation

In this Section, we describe the simplification operator used for coarsening the Forman gradient by reducing the number of its critical simplices, as well as its undo (refinement) operator.

6.1. i-cancellation

The simplification operator we use for modifying the Forman gradient V is called *cancellation* and it has been introduced in [8]. An *i-cancellation* deletes two critical simplices connected by a critical V_i-path. In the case of a triangle mesh we distinguish between two types of cancellation:

- A *1-cancellation* applied to V deletes a critical edge e and a critical triangle t connected by a separatrix V_1-path γ; the V_1-path connecting e to t is reversed by starting from e and ending into t (see Figure 5). We denote the new path γ^{-1}. This produces a new Forman gradient V', which is the same as V except for γ^{-1}, and where edge e and triangle t are no longer critical.

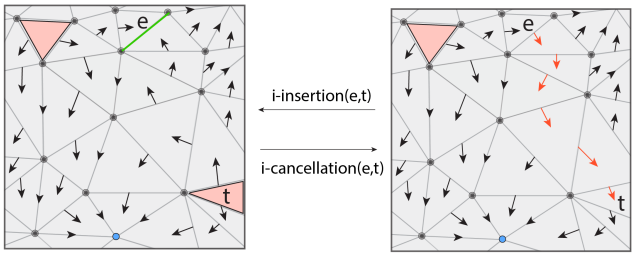


Figure 5: From left to right: the effect of 1-cancellation(e, t) on a Forman gradient V; the separatrix V_1-path connecting e and t is reversed by starting from e and by defining new pairs depicted as red arrows. From right to left: effect of its dual operator, 1-insertion(e, t).

- A *0-cancellation* applied to V deletes a critical vertex v and a critical edge e connected by a separatrix V_0-path. The effect of a 0-*cancellation*(e, v) on V is to delete v and e from the set of critical simplices of V and to reverse the gradient arrows on the one separatrix V_0-path between v and e.

6.2. i-insertion

An i-insertion is defined as the undo of an i-cancellation. A 1-insertion(e, t) introduces a critical edge e and a critical triangle t in a Forman gradient V', by inverting the V_1-path between e and t (see Figure 5). Symmetrically, a 0-insertion(e, v) introduces a critical edge e and a critical vertex v in V' by reversing the V_0-path between e and v.

Algorithm 1 provides a description of the effect of a 1-insertion. The algorithm starts from the triangle *tri* which will become critical, and retrieves its paired edge *edg* from V (row 7). The corresponding pair (edg, tri) is then removed from V (row 9). The other triangle adjacent to *edg* is retrieved (row 11) as well as its paired edge (row 12). Pair $(nextEdg, nextTri)$ is removed from V and pair $(edg, nextTri)$ is added (row 13-14). Pairs are then inverted in sequence until the edge e, which will become critical, is reached. A 0-insertion works in a completely dual fashion by working on vertices and edges.

We consider a generic i-cancellation(σ, τ) applied on (Σ, V). This produces a new pair (Σ, V'). To apply its inverse i-insertion(σ, τ), we do not need to work on (Σ, V'), but on a "locally equivalent" (Σ', V''). Thus, we say that an i-insertion(σ, τ) applied to the pair (Σ', V'') is *feasible* if the following two conditions are satisfied:

- σ and τ are in Σ';
- critical simplex τ_1, which is the other critical simplex, besides τ. connected to σ through a separatrix in (Σ, V) is in V''.

Generally speaking we should also guarantee that σ and τ are connected by a separatrix path for the insertion to be valid. However, the presence of τ_1 is a sufficient condition to guarantee this, as explained below. Since we are working with triangle meshes we know that:

Algorithm 1 1-insertion(e,s,V,Σ)

Input: e, future critical edge
Input: t, future critical triangle
Input: V, Forman gradient
Input: Σ, triangle mesh
Output: V, updated Forman gradient

edg = getPair(t,V)
$tri = t$
removePair(edg, tri, V)
while $edg \neq e$ **do**
  $nextTri$ = getAdjacent(edg,tri,Σ)
  $nextEdg$ = getPair($nextTri$,V)
  removePair($nextEdg$, $nextTri$, V)
  setPair(edg, $nextTri$, V)
  $edg = nextEdg$
  $tri = nextTri$
return V

(i) each edge σ is connected to, at most, two critical triangles t and t_1,

(ii) each edge σ is connected to, at most, two critical vertices v and v_1,

(iii) a V_1-path cannot be modified by inverting a V_0-path (and vice-versa).

Let us consider a 1-cancellation(e,t). We denote as γ the separatrix V_1-path connecting e and t and γ^{-1} the gradient path obtained by inverting all the arrows of γ as the result of the cancellation. We know from (iii) that γ^{-1} can be further modified by performing a 1-cancellation only and, because of (i), such cancellation necessarily involves t_1. Thus, we conclude that if $t_1 \in V$,then also $\gamma^{-1} \in V$, and the 1-insertion(e, t) is feasible. The same holds for a 0-insertion(e,v) by using (ii) instead of (i).

The feasibility condition above implicitly defines a *dependency relation* among i-insertions, as discussed in Section 7.1.

7. The Hierarchical Forman Triangulation

In this Section, we define a multi-scale model for a triangulated shape endowed with a Forman gradient (Σ, V), generated on the basis of the combined simplification of Σ and V. We call such a model a *Hierarchical Forman Triangulation (HFT)*. An HFT is built by applying an interleaved sequence of *gradient-aware edge contractions* and *cancellations* to (Σ, V) until no more simplifications can be performed. An HFT consists of:

- the triangle mesh Σ_B (the *base mesh*) obtained by simplifying Σ endowed with the Forman gradient V_B (the *base Forman gradient*) obtained by simplifying V;
- the set of all refinements which undo the simplifications performed on (Σ, V);
- a direct dependency relation between pair of refinements, defined below based on the feasibility conditions discussed in Sections 5.2 and 6.2.

7.1. Direct dependency relation

Generally speaking, a refinement r_i depends on another refinement r_j if r_j introduces new simplices or makes a simplex critical, as required for r_i to be feasible (see Sections 5.2 and 6.2 for the definitions of feasibility). Recall that vertex split $split(v_1, v_2)$ creates a new vertex v_1 and a new edge (v_1, v_2), while an i-insertion(e, σ) makes edge e and simplex σ critical. We have three kinds of dependency:

- A vertex split $split(v_1, v_2)$ *directly depends* on another vertex split operator $split(v'_1, v'_2)$ if and only if v'_1 is required in order that $split(v_1, v_2)$ can be correctly applied. This means that v'_1 is either the same as v_2 or is one of the vertices in the sequence S_2, as defined by the feasibility condition in Section 5.2;
- An i-insertion(e, σ) *directly depends* on a single i-*insertion*(e_1, σ_1), which is the one making σ_1 critical, where σ_1 is the critical simplex (vertex or triangle) connected through a separatrix path to edge e, as required by the feasibility condition in Section 6.2;
- An i-*insertion*(e, σ), *directly depends* on a vertex split $split(v_1, v_2)$ if v_1 is one of the vertices of e or of σ.

Note that simplices are never introduced twice in Σ, since each vertex, edge and triangle is introduced by a single refinement, and similarly critical simplices are never introduced twice in V. Thus, the direct dependency relation defined above is a partial order relation. The hierarchical structure of the HFT is thus described by a Directed Acyclic Graph (DAG), in which the nodes are the refinement operators and the arcs encode their mutual direct dependencies.

Comparing to multi-scale models that represent only the morphology of a scalar field, we can notice a relevant improvement in the dependency relation for the i-insertions. We recall that in [11, 4] the dependency relation is defined in terms of a *diamond* (see Section 3). Each anti-cancellation, the equivalent of our insertion, corresponds to a diamond. Two anti-cancellations are dependent if the corresponding diamonds share at least one vertex [11] or at least one edge [4]. Recalling that a diamond is a quad having four critical maxima/minima as vertices, it is easy to see that the relation in [11] has a minimum bound of four (i.e., one dependent diamond for each vertex of the quad). The upper bound is not limited and it depends on the number of diamonds incident in each vertex of the quad. The relation described in [4] has a limited upper bound since we have at most four dependent anti-cancellations (one for each edge of the diamond).

On the other hand, an i-insertion is dependent only on one other i-insertion, which translates in having only a single link in the hierarchy. Thus, it is minimal. This is an important aspect, since fewer dependencies imply a higher expressive power for a multi-scale model, where the expressive power is evaluated in terms of the number of representations which can be extracted from the model.

7.2. HFT encoding

We represent the base mesh Σ_B by encoding triangles and vertices only. We use the Indexed-based data structure with Adjacencies (IA) described in [28]. Both triangles and vertices are indexed in two different arrays and can be referred by using four bytes. For each triangle, we store the indices of its three vertices and the indices of its three adjacent triangles. For each vertex, we store its coordinates and the index of one of the triangles incident in it. We denote as $|\Sigma_0|$ and $|\Sigma_2|$ the number of vertices and triangles in Σ_B, respectively. Encoding the vertices of Σ_B requires 28 bytes per vertex, i.e., 24 *bytes* for its coordinates and 4 *bytes* for indexing one of its incident triangles. Each triangle in Σ_B requires 4 *bytes* for each incident vertex and for each adjacent triangle.

Forman gradient V is encoded by associating, with each triangle, the subset of pairs involving its edges and vertices. We call this collection of pairs as *local frame*. We refer to [22] for a detailed description of this encoding, which results in only one byte per triangle for storing the entire V. Thus, encoding Σ_B plus V_B requires in total

$$28|\Sigma_0| + 25|\Sigma_2| \; bytes.$$

The direct dependency relation in an HFT is encoded in a DAG. Nodes are of two types for indicating mesh updates ($Node_g$) and topological updates ($Node_m$). Nodes of each type are indexed in an array denoted N_g and N_m, respectively.

Each $Node_g$ represents a vertex split $split(v_1, v_2)$, undo of an edge contraction $contract(v_1, v_2)$. It encodes:

- the coordinates of newly inserted vertex v_1;
- one index to each node representing a vertex split on which $split(v_1, v_2)$ depends.

This requires storing 4 *bytes* for each split it depends on and 32 *bytes* for the coordinates of v_1 (3+1 floats). Encoding each node $Node_g \in N_g$ requires

$$32 + 4d \; bytes,$$

where d denotes the number of nodes $Node_g$ depends on, which can be a variable number depending on the number of vertices adjacent to v_2 before the corresponding edge contraction.

Each $Node_m$ represents an i-insertion, undo of an i-cancellation applied to V. It encodes:

- the persistence value of the pair that will be introduced, where the persistence is defined as the absolute value of the difference between the field values at the two critical simplices introduced;
- an index to the only topological node this refinement depends on;
- an index to each geometric node this refinement depends on.

Algorithm 2 selectiveRefinement(V_B,N_m,e_p,Σ_B,N_g,e_l)

Input: V_B, base Forman gradient
Input: N_m, updates for V_B
Input: e_p, threshold
Input: Σ_B, base mesh
Input: N_g, updates for Σ_B
Input: e_l, threshold
Output: (Σ', V'), mesh and gradient extracted

$V_E = V_B$
$\Sigma_E = \Sigma_B$
for each n in N_g **do**
  if $e_l <$ error(n) **then**
    break;
  refineMesh(n, Σ_E)
for each n in N_m **do**
  if $e_p <$ error(n) **then**
    break;
  refineMorphology(n, V_E, Σ_E)
return (Σ_E, V_E)

We use a float for encoding the persistence value (8 byte). Since each refinement depends on only one $Node_m$ (see Section 7.1), encoding this relation will require 4 byte only. Note that there can be five or three geometric nodes to refer, which correspond to the vertices of the two critical simplices it depends. One of the two simplices is an edge, which is bounded by two vertices, while the other simplex can be a triangle, which accounts for three vertices or a single vertex. We overestimate the storage cost required for each node in N_m as $8 + 4 + (5 * 4) bytes$.

An experimental evaluation performed on the test datasets is provided in Section 9.

8. Querying an HFT

We perform two kinds of queries: *selective refinement* queries on an HFT, which extract representations of (Σ, V) at different scales, and *topological* queries, which extract topological features, like the critical net, ascending or descending manifolds, Morse or Morse-Smale complexes.

8.1. Selective refinement

A selective refinement extracts from an HFT a mesh with the *minimum* number of triangles endowed with a Forman gradient with the *minimum* number of critical simplices based on some prescribed criterion. In our current implementation, we have defined two criteria based on edge length and persistence [9]. Specifically, a vertex split $split(v_1, v_2)$ is performed if the length of edge (v_1, v_2) is greater than a user-defined threshold value e_l, and an i-$insertion(\sigma, \tau)$ is applied if the difference in absolute value of $f(\sigma)$ and $f(\tau)$ is greater than some threshold value e_p.

Algorithm 2 describes how a selective refinement is performed, starting from (Σ_B, V_B), where Σ_B is the base mesh and V_B the base Forman gradient V_B. The algorithm extracts a representation at variable scale, that we denote as (Σ_E, V_E), initialized

Algorithm 3 refineMesh(n, Σ_E)

```
1: Input: n, update operator
2: Input: Σ_E, front mesh
3:
4: if refined(n) then
5:     return;
6: if notReady(n) then
7:     D = dependencyMesh(n)
8:     for each d in D do
9:         refineMesh(d,Σ_E)
10: split(n,Σ_E)
```

Algorithm 4 refineMorphology(n, V_E)

```
1: Input: n, update operator
2: Input: V_E, front Forman gradient
3: Input: Σ_E, front mesh
4:
5: if refined(n) then
6:     return;
7: if notReady(n) then
8:     D = dependencyMesh(n)
9:     for each d in D do
10:        refineMesh(d,Σ_E)
11:    D = dependecyMorphology(n)
12:    for each d in D do
13:        refineMorphology(d,V_E,Σ_E)
14: insert(n,Σ')
```

with (Σ_B, V_B). It performs a visit of the refinements, that are stored in descending order of edge length and persistence for N_g and N_m, respectively. When reaching a node n, the algorithm checks first if all the refinements on which n depends upon have been performed, and, if they have not been, it activates them by recursively visiting the predecessors of node n in the DAG. The refinement corresponding to node n will be performed at the end of this process.

The algorithm considers mesh refinements N_g (row 11) and activates vertex splits as long as the geometric *error* in the currently extracted mesh Σ_E is larger than threshold e_l (row 12). Performing all possible mesh refinements first reduces the activation of a vertex split due to an i-insertion. Since an i-refinement depends on some vertex splits, increasing the resolution of the mesh at first will increase the number of satisfied dependencies once we are refining the nodes in N_m.

Algorithm 3 describes the procedure for updating the mesh by performing the vertex split associated with a node $n \in N_g$. Function $refined(\cdot)$ returns the value *true* if the update has been already performed (row 4), *false* otherwise. Function *notReady*$(\cdot)$ (row 6) returns the value *true* if the corresponding vertex split requires one or more vertex splits to be performed first. Function $refineMesh(\cdot)$ recursively calls algorithm 3 on all such required splits. Note that the splits triggered by a vertex split are applied weather or not the error they introduce is lower than e_l. After all the dependencies have been solved, the vertex split is performed on Σ_E (row 10).

Algorithm 5 extractDescendingCell(s_c,V,Σ)

```
1: Input: s_c, critical k-simplex
2: Input: V, Forman gradient
3: Input: Σ, triangle mesh
4: Output: M, descending k-manifold
5:
6: M = ∅
7: Q = ∅
8: add(M,s_c)
9: enqueue(Q,s_c)
10: while Q.notEmpty() do
11:     s = dequeue(Q)
12:     BD_s = boundary(s,Σ)
13:     for each b in BD_s do
14:         s_1 = getPair(b,V)
15:         if s_1 ≠ ∅ AND s_1 ≠ s then
16:             add(M,s_1)
17:             enqueue(Q,s_1)
18: return M
```

Once the desired resolution for Σ_E has been achieved, the refinement of gradient V_E starts (Algorithm 2, row 15). For each node $n \in N_m$, the corresponding i-insertion is performed if the *error* associated with n is larger than threshold e_p. Algorithm 4 works in the same way as algorithm 3 with only one difference. Since an i-insertion corresponding to a node n in N_m may depend on nodes in both N_g and N_m, the algorithm has to ensure that all such refinements are performed before refining the gradient V_E with n (rows 8 and 11).

8.1.1. Extracting topological features

Several topological features can be extracted from an HFT at different scales. We have developed algorithms for extracting descending and ascending manifolds from an extracted triangle mesh Σ_E endowed with the Forman gradient V_E. The collection of the descending and ascending manifolds forms the discrete ascending and descending Morse complexes, respectively. From these latter, all the other topological features can be computed. The Morse-Smale complex is obtained by intersecting the ascending and descending manifolds. The critical net is computed by considering the critical simplices and the 1-manifolds of the ascending and descending Morse complexes connecting them.

Algorithm 5 describes the process of extracting a descending k-manifold. The gradient is visited by using a breadth-first traversal (row 6). All boundary $(k-1)$-simplices are computed for each k-simplex s extracted from the queue, (row 10). If a $(k-1)$-simplex b is paired with a k-simplex s_1, different from s, s_1 is added to the queue (row 11). Each k-simplex is visited at most once, so this operation has a time complexity of $O(\Sigma_k)$ where Σ_k is the number of k-simplices in Σ_E. The algorithm for extracting an ascending manifold is the same as algorithm 5, except for row 11 where, for each k-simplex s, the $(k+1)$-simplices in the co-boundary of s will need to be extracted.

Dataset	$\|\Sigma_0\|$	$\|\Sigma_2\|$	$\|Node_g\|$	$\|Node_t\|$	$\|Node_g\|_{\%}$	$\|Node_t\|_{\%}$
NEPTUNE	2.0M	4.0M	2.0M	2.2K	99.8%	0.11%
DRAGON	3.6M	7.2M	3.6M	5.7K	99.8%	0.11%
STATUETTE	4.9M	10M	4.9M	11K	99.7%	0.22%
COMOLAKE	810K	1.6M	806K	25K	96.9%	3.07%
MAJORLAKE	810K	1.6M	806K	29K	96.4%	3.57%
MAUI	2.0M	4.0M	2.0M	25K	98.7%	1.2%
BAIA	4.1M	8.3M	4.1M	16K	99.5%	0.41%
PUGET	9.7M	19M	9.7M	17.1K	98.2%	1.75%

Table 1: Characteristics of the datasets used in our experiments. For each mesh we show the dataset name (column *Dataset*), the number of vertices and triangles (columns $|\Sigma_0|$ and $|\Sigma_2|$ respectively), the number of geometric and topological nodes in the HFT (columns $|Node_g|$ and $|Node_t|$ respectively) and the percentage of such nodes respect to the total number of nodes (columns $|Node_g|_{\%}$ and $|Node_t|_{\%}$ respectively).

9. Experimental results

In this Section, we discuss our experimental evaluation of the HFT. The purpose of our experiments is to show the efficiency of our approach in computing Morse complexes on representations, at different levels of resolution, extracted from the HFT. We present only the results obtained in extracting the ascending and descending Morse complexes, for the sake of brevity, but the Morse-Smale complex can be efficiently computed from them, as well as the critical net by following the separatrix gradient paths. Our experiments have been performed on a desktop computer with a 3.2Ghz processor and 16GB of memory.

We have tested our implementation by computing the storage cost of the HTF for five terrain datasets. Since our implementation is defined for triangle meshes, we have also tested it by using three triangulated shapes. The field value for each vertex is initialized with the corresponding coordinate on the z axis. Figures focus on terrain datasets since improving efficiency for geospatial applications is the main focus of our work. Numerical results are presented for all datasets. In Table 1, we present the datasets used for our experiments. All the datasets are courtesy of the Virtual Terrain Project (VTP), the Stanford 3D Scanning Repository and the Aim@Shape repository. The size of the triangle meshes used in the experiments is between 1.6M and 19M triangles.

For each dataset, we have built an HFT by simplifying it with an interleaved sequence of mesh and topological simplifications. We recall that a topological refinement requires a specific resolution of the mesh (see the dependency relation defined in Section 7.1). This is imposed by the resolution of the mesh when the corresponding simplification has been performed. Then, we prioritize geometrical simplifications. Once all viable mesh simplifications have been performed, and all the remaining ones are blocked because of some critical simplex, we perform a subset of the viable topological simplifications. We simplify 10% of the initial number of critical simplices at each iteration. At this point, new mesh simplifications become available, and the process can continue until no more mesh or topological simplifications can be performed. This way, the resolution required by each topological refinement will be as low as possible and we will not be forced to excessively refine the mesh while performing the topological refinements.

For each dataset, we show the number of vertices and triangles in the mesh Σ at full resolution as well as the number of nodes in the corresponding HFT, classifying them as mesh refinements ($Node_g$) and as morphological refinements ($Node_t$). All datasets are simplified by interleaving sets of topological simplifications with sets of morphological simplifications. In the last two columns we show the percentage of nodes of a certain type with respect to the total number of nodes. We can notice that mesh refinements ($Node_g$) are almost 90% of the total number of nodes, while the morphological refinements account for a small part of the entire model. This emphasizes the importance of simplifying the mesh when we are interest in topological data analysis.

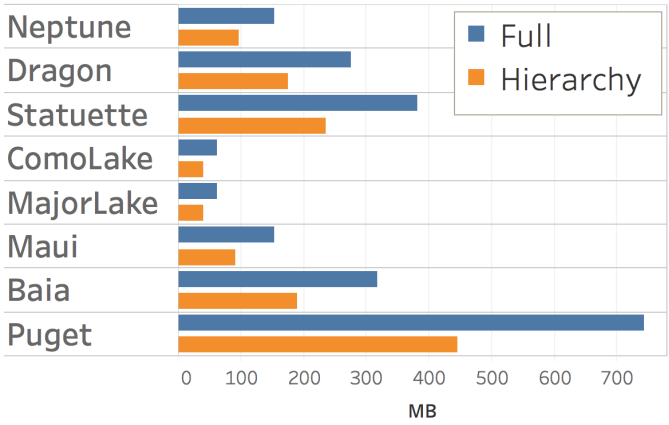


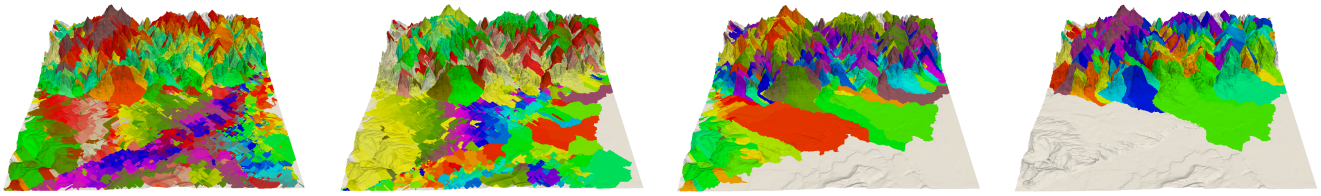
Figure 6: Comparison of the memory consumption for storing a mesh a full resolution and the corresponding Hierarchical Forman Triangulation.

In Figure 6, we compare the storage cost for encoding the HFT, i.e., the base mesh and base gradient (Σ_B, V_B) plus the data structure encoding the refinements and the DAG of the dependencies, and that for encoding (Σ_F, V_F), i.e. the mesh at full resolution endowed with the Forman gradient.

The storage costs are estimated as discussed in Section 7.2. The pair (Σ_F, V_F) is encoded in the same data structure used for the base mesh and base gradient (Σ_B, V_B). We have estimated that the HFT provides a saving in storage between 34% and 40% with respect to the storage cost ofy (Σ_F, V_F).

The higher compression achieved does not impact efficiency. We have computed the time required by performing all simplifications which from Σ_F and V_F, i.e., the representation at full resolution, lead to Σ_B and V_B. As opposed, we have tested our model by starting from the base representation Σ_B and V_b encoded in the HFT, applying all the possible refinements until reaching Σ_F and V_F. On the largest dataset terrain dataset Puget), the simplification process takes more than 1 hour, while the full refinement of the model takes less than 2 minutes.

The main purpose of the HFT is to allow an efficient interactive topological analysis of a dataset by overcoming all the limitations deriving by a fixed resolution in the underlying mesh representation. In Figure 7 we show an example of a selective refinement performed by varying the topological level of detail while keeping the mesh at full resolution. For each image, we are showing the descending 2-manifolds (which form the descending Morse complex). We are also showing the total



C=34K, 66sec. C=12K, 62sec. C=3.1K, 59sec. C=0.9K, 55sec.

Figure 7: Selective refinement performed by extracting the mesh at full resolution and the topology at variable resolution. For each representation extracted we show the total number of critical simplices (denoted C) and the time (in seconds) required for extracting all the ascending and descending manifolds. Colored regions superimposed on the triangulation correspond to the descending 2-manifolds.

number of critical simplices extracted and the time required for computing all the ascending and descending manifolds. We notice that, independently of the scale of the topological representation, the timings for extracting the Morse complexes are almost the same. This is due to the underlying mesh, which is at full resolution.

On the other hand, as shown in Figure 8, varying the resolution of the underlying mesh can affect the computation of the ascending and descending manifolds considerably. In these experiments, we have performed extractions by varying the resolution of the mesh, but always extracting the topological representation at full scale. For each image, we show the time required for computing ascending and descending Morse complexes. Depending on the mesh resolution, computing the ascending and descending manifolds is generally 3 to 30 times faster than computing them on the mesh at full resolution. Figure 8 underlines the importance of having an efficient multi-scale model for data exploration. The level of detail provided by Figure 8(b) or (c) is adequate for studying the morphology of the dataset globally. When the user is interested in a broad overview, the less refined representation of Figure 8(a) can be sufficient. On the other hand, the speed-up obtained by using the latter is clear. The key aspect and distinctive feature of our model is in supporting an efficient and effective extraction of representations at scales which can vary in different parts of the mesh. This provides the capability to zoom in on the overview picture, increasing the level of detail only on small parts of the domain.

By using the HFT, we can use a high mesh resolution only on specific regions of interest. The extractions at variable scale are performed by setting as input parameter a window inside the domain in which to concentrate the resolution. Figure 9 shows the results of a window query where we extract, at full mesh resolution, only the subset of the domain inside the window. From top to bottom we show three images obtained by increasing the far bound of the viewport. For each image, we report the number of vertices in the extracted mesh and the time required for extracting the descending 2-manifolds. Critical triangles (depicted in red), edges (depicted in green) and vertices (depicted in blue) are also shown for giving a visual representation of the morphological resolution.

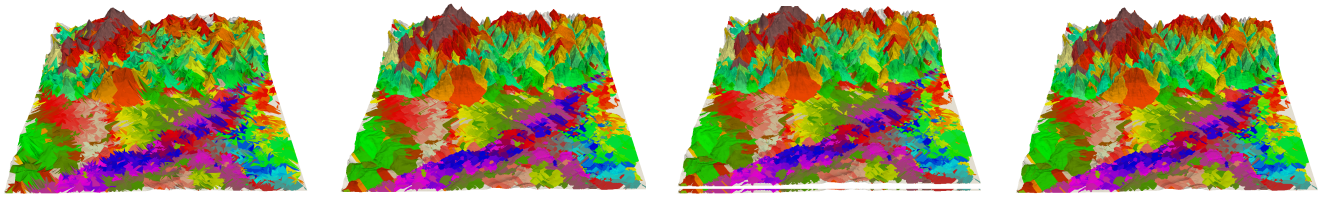
We have evaluated our model, by defining a window-based query automatically, by centering the window inside the dataset and defining its size as a percentage (10% 30% and 50%) of the size of the domain. We require a maximum mesh resolution inside each window and no constraint outside. The operation achieves a 30x speed up with respect to working on the mesh at full resolution for the smallest window. The speed up reduces to 8x when using the largest window.

10. Concluding remarks

We have presented a new multi-scale model, the *Hierarchical Forman Triangulation (HFT)*, which is a combined geometric and topological representation for a triangle mesh Σ endowed with a Forman gradient V. The HFT is generated through an edge contraction operator for the mesh, which preserves the gradient, and through a topological operator, which reduces the number of critical simplices in V without modifying the simplices in Σ. The model has a low storage cost, lower than that of the mesh at full resolution, and provides a high flexibility by adjusting the resolution of the mesh to comply with the scale of the topological representation. We obtain a consistent saving in computation times for extracting Morse complexes both when reducing the resolution globally and when performing adaptive refinements inside a window. There are several avenues for developing the work presented here. In the current model generation, the edges are contracted in order of their length. This choice has been performed since our objective has been to develop and experiment with the entire framework without worrying about the quality of the approximation of the terrain or 3D shape we produce. To obtain a higher fidelity to the original shape, we plan to experiment with other quality measures, such as the Quadric Error Metrics(QEM) [29].

If we consider any triangle mesh Σ_E endowed with a Forman gradient V_E, extracted from the HFT, we notice that, if we attach to the vertices of Σ_E the scalar field values of the corresponding vertices of the original mesh, they may not agree with the gradient V_E. This is not an issue when we want to compute topological features since the gradient field is always correct. Methods exist that modify a piecewise-linear function while preserving the morphological structure of the Morse-Smale complex [30, 31, 32, 33]. We can apply such techniques to our extracted representations, which have a lower complexity due to the reduced size of the mesh. We are also planning to modify the function values at the vertices to obtain a function admissible for V_E, by considering techniques which generate from a Forman gradient a discrete Morse function compatible with it.

We observe that the operations of computing and simplifying a Forman gradient are not limited to triangle meshes, but we



(a) $|\Sigma_0|$=96K,2.1sec. (b) $|\Sigma_0|$=400K,3.3sec. (c) $|\Sigma_0|$=1M,7.2sec. (d) $|\Sigma_0|$=4M,66sec.

Figure 8: Selective refinements performed by extracting topological features at full resolution and the mesh at variable resolutions. For each extraction we show the total number of vertices (denoted $|\Sigma_0|$) and the time (in seconds) required for extracting the ascending and descending Morse cells. Colored regions superimposed on the triangulation correspond to the descending 2-manifolds. Figure (d) shows the mesh at full resolution

can consider other discretization of a 3D shape, for instance as a quad mesh. A quad mesh can be represented with a very similar data structure as the one we use for triangle meshes, and an extension of the gradient encoding for triangle meshes to quad meshes seem to be straightforward. A Forman gradient is defined for cell complexes, and the cancellation and insertion will work in the same way. The triangle-specific part is the mesh simplification and refinement. We are currently studying the definition of new gradient aware simplifications for simplifying quad meshes. A set of local operators for simplifying a quad mesh is discussed in [34]. Besides having sequences of edge contractions, we are considering the extension of edge/vertex rotate and diagonal collapse as well.

Both the definition and the implementation of the hierarchical model and the topological simplification operators are dimension-independent. Our current plan is to define a gradient-aware edge contraction and its inverse, vertex split, for tetrahedral meshes endowed with a Forman gradient. Multi-resolution models for tetrahedral meshes can be found in the literature based on edge contraction and vertex split [35]. We plan to develop a hierarchical Forman tetrahedralization for 3D scalar fields defined on unstructured tetrahedral meshes for volume data analysis and visualization.

Finally, a Forman gradient has been recognized as a powerful tool for computing the homology of a shape in an efficient manner. In this alternative framework, the Forman gradient is used as a compact representation of the triangle mesh on which computing homology or persistent homology efficiently. By applying the *cancellation/insertion* operators we are guaranteed not to change the homology of the shape. In the same way, the edge contraction/vertex split operators are guaranteed to preserve its persistent homology. We are planning to make an exploratory use of the HTF for extracting homology generators on a triangulated shape, computing specifically homology generators which are "localized" [36] in the sense of being independent and as smaller as possible.

11. Acknowledgments

This work has been partially supported by the US National Science Foundation under grant number IIS-1116747.

References

[1] Uzunbaş MG, Chen C, Metaxsas D. Optree: A Learning-Based Adaptive Watershed Algorithm for Neuron Segmentation. Cham: Springer International Publishing. ISBN 978-3-319-10404-1; 2014, p. 97–105. doi:10.1007/978-3-319-10404-1_13.

[2] Heine C, Leitte H, Hlawitschka M, Iuricich F, De Floriani L, Scheuermann G, et al. A survey of topology-based methods in visualization. Computer Graphics Forum 2016;35(3):643–67. doi:10.1111/cgf.12933.

[3] De Floriani L, Fugacci U, Iuricich F, Magillo P. Morse complexes for shape segmentation and homological analysis: discrete models and algorithms. In: Computer Graphics Forum; vol. 34. Blackwell Publishing Ltd; 2015, p. 761–85. doi:10.1111/cgf.12596.

[4] Bremer PT, Pascucci V, Hamann B. Maximizing Adaptivity in Hierarchical Topological Models Using Cancellation Trees. In: Möller T, Hamann B, Russell RD, editors. Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration. Mathematics and Visualization; Springer Berlin Heidelberg. ISBN 978-3-540-25076-0; 2009, p. 1–18. doi:10.1007/b106657_1.

[5] Dey TK, Giesen J, Goswami S. Shape Segmentation and Matching with Flow Discretization. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-45078-8; 2003, p. 25–36. doi:10.1007/978-3-540-45078-8_3.

[6] Feng W, Huang J, Ju T, Bao H. Feature correspondences using morse smale complex. The Visual Computer 2013;29(1):53–67. doi:10.1007/s00371-012-0674-3.

[7] Dong S, Bremer PT, Garland M, Pascucci V, Hart JC. Spectral surface quadrangulation. In: ACM SIGGRAPH 2006 Papers. SIGGRAPH '06; New York, NY, USA: ACM. ISBN 1-59593-364-6; 2006, p. 1057–66. doi:10.1145/1179352.1141993.

[8] Forman R. Morse Theory for Cell Complexes. Advances in Mathematics 1998;134(900145):90–145. doi:http://dx.doi.org/10.1006/aima.1997.1650.

[9] Edelsbrunner H, Letscher D, Zomorodian A. Topological persistence and simplification. Discrete and Computational Geometry 2002;28(4):511–33. doi:10.1007/s00454-002-2885-2.

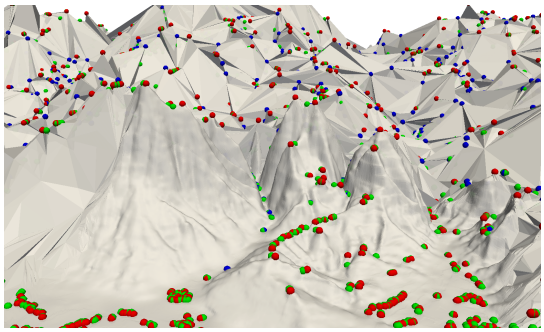
[10] Gyulassy A, Natarajan V, Pascucci V, Bremer PT, Hamann B. A Topological Approach to Simplification of Three-Dimensional Scalar Functions. IEEE Transactions on Visualization and Computer Graphics 2006;12(4):474–84.

[11] Bremer PT, Edelsbrunner H, Hamann B, Pascucci V. A topological hierarchy for functions on triangulated surfaces. IEEE Transactions on Visualization and Computer Graphics 2004;10(4):385–96. doi:10.1109/TVCG.2004.3.

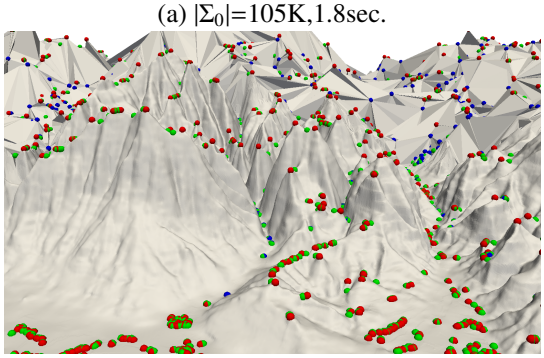
[12] Gyulassy A, Kotava N, Kim M, Hansen CD, Hagen H, Pascucci V. Direct feature visualization using morse-smale complexes. IEEE Transactions on Visualization and Computer Graphics 2012;18(9):1549–62. doi:10.1109/TVCG.2011.272.

[13] Čomić L, De Floriani L, Iuricich F. Dimension-independent multiresolution Morse complexes. Computers and Graphics (Pergamon) 2012;36(5):541–7. doi:10.1016/j.cag.2012.03.010.

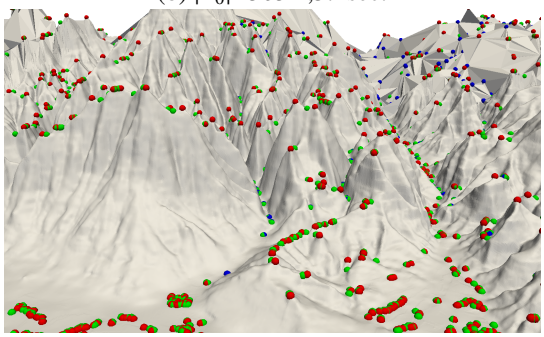
[14] Iuricich F, De Floriani L. A combined geometrical and topological simplification hierarchy for terrain analysis. In: Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. SIGSPATIAL '14; New York, NY, USA: ACM. ISBN 978-1-4503-3131-9; 2014, p. 493–6. doi:10.1145/2666310.2666487.



(a) $|\Sigma_0|$=105K,1.8sec.



(b) $|\Sigma_0|$=303K,5.2sec.



(c) $|\Sigma_0|$=1.0M,16.1sec.

Figure 9: Three selective refinement queries where the mesh is extracted at maximum resolution inside a window with size as 10%, 30% and 50% of the domain. For each image we report the number of vertices in the extracted representation and the time required for extracting the Morse complexes. Critical triangles (depicted in red), edges (depicted in green) and vertices (depicted in blue) are also shown.

[15] Milnor JW. Morse theory. Princeton University Press, New Jersey; 1963. ISBN 0691080089 9780691080086.

[16] Matsumoto Y. An Introduction to Morse Theory; vol. 208. American Mathematical Society; 2002. ISBN 0821810227.

[17] Banchoff TF. Critical Points and Curvature for Embedded Polyhedral Surfaces. The American Mathematical Monthly 1970;77(5):475–85.

[18] Agoston MK. Computer Graphics and Geometric Modeling: Implementation and Algorithms. Springer-Verlag London Ltd.; 2005. ISBN ISBN-10: 1-85233-818-0. doi:10.1016/j.cagd.2009.02.005.

[19] Robins V, Wood PJ, Sheppard AP. Theory and algorithms for constructing discrete Morse complexes from grayscale digital images. IEEE Transactions on Pattern Analysis and Machine Intelligence 2011;33(8):1646–58.

[20] Edelsbrunner H, Harer J, Zomorodian A. Hierarchical Morse Complexes for Piecewise Linear 2-Manifolds. In: Proc. 17th ACM Symposium on Computational Geometry. 2001, p. 70–9. doi:10.1145/378583.378626s.

[21] Tierny J, Pascucci V. Generalized topological simplification of scalar fields on surfaces. IEEE Transactions on Visualization and Computer Graphics 2012;18(12):2005–13. doi:10.1109/TVCG.2012.228.

[22] Fellegara R, Iuricich F, De Floriani L, Weiss K. 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 - SIGSPATIAL '14. ISBN 9781450331319; 2014, p. 223–32. doi:10.1145/2666310.2666412.

[23] Čomić L, De Floriani L, Iuricich F. Simplifying morphological representations of 2D and 3D scalar fields. In: Cruz IF, Agrawal D, Jensen CS, Ofek E, Tanin E, editors. Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '11. ACM. ISBN 9781450310314; 2011, p. 437. doi:10.1145/2093973.2094042.

[24] Beucher S. Watershed, Hierarchical Segmentation and Waterfall Algorithm. In: Serra J, Soille P, editors. Mathematical Morphology and Its Applications to Image Processing; vol. 2 of *Computational Imaging and Vision*. Springer Netherlands. ISBN 978-94-010-4453-0; 1994, p. 69–76. doi:10.1007/978-94-011-1040-2_10.

[25] Danovaro E, De Floriani L, Magillo P, Vitali M. Multiresolution morse triangulations. In: Proceedings of the 14th ACM Symposium on Solid and Physical Modeling - SPM '10. SPM '10; New York, NY, USA: ACM. ISBN 9781605589848; 2010, p. 183. doi:10.1145/1839778.1839806.

[26] Popović J, Hoppe H. Progressive simplicial complexes. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97. SIGGRAPH '97; New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. ISBN 0897918967; 1997, p. 217–24. doi:10.1145/258734.258852.

[27] Dey TK, Edelsbrunner H, Guha S, Nekhayev DV. Topology preserving edge contraction. Publ Inst Math (Beograd) (NS 1998;66:23–45.

[28] Canino D, Floriani LD, Weiss K. IA*: An adjacency-based representation for non-manifold simplicial shapes in arbitrary dimensions. Computers & Graphics 2011;35(3):747–53. doi:10.1016/j.cag.2011.03.009.

[29] Garland M, Heckbert PS. Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97. ACM Press/Addison-Wesley Publishing Co. ISBN 0897918967; 1997, p. 209–16. doi:10.1145/258734.258849.

[30] Weinkauf T, Gingold YI, Sorkine O. Topology-based smoothing of 2d scalar fields with C^1-continuity. Comput Graph Forum 2010;29(3):1221–30. doi:10.1111/j.1467-8659.2009.01702.x.

[31] Magillo P, De Floriani L, Iuricich F. Morphologically-aware elimination of flat edges from a tin. In: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. SIGSPATIAL'13; New York, NY, USA: ACM. ISBN 978-1-4503-2521-9; 2013, p. 244–53. doi:10.1145/2525314.2525341.

[32] Günther D, Jacobson A, Reininghaus J, Seidel H, Sorkine-Hornung O, Weinkauf T. Fast and memory-efficienty topological denoising of 2d and 3d scalar fields. IEEE Trans Vis Comput Graph 2014;20(12):2585–94. doi:10.1109/TVCG.2014.2346432.

[33] Allemand-Giorgis L, Bonneau GP, Hahmann S. Piecewise Polynomial Reconstruction of Functions from Simplified Morse-Smale complex. In: IEEE Visualization conference 2014, SciVis Posters. Paris, France: IEEE; 2014,.

[34] Tarini M, Pietroni N, Cignoni P, Panozzo D, Puppo E. Practical quad mesh simplification. Computer Graphics Forum 2010;29(2):407–18. doi:10.1111/j.1467-8659.2009.01610.x.

[35] Cignoni P, Floriani LD, Magillo P, Puppo E, Scopigno R. Selective refinement queries for volume visualization of unstructured tetrahedral meshes. IEEE Trans Vis Comput Graph 2004;10(1):29–45. doi:10.1109/TVCG.2004.1260756.

[36] Zomorodian A, Carlsson GE. Localized homology. Comput Geom 2008;41(3):126–48.