Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi

Multi-resolution shape analysis based on discrete Morse decompositions

by

Iuricich Federico

Theses Series **DIBRIS-TH-2014-03**

DIBRIS, Università di Genova

Via Opera Pia, 13 16145 Genova, Italy `http://www.dibris.unige.it/`

Università degli Studi di Genova

Dipartimento di Informatica, Bioingegneria,

Robotica ed Ingegneria dei Sistemi

Dottorato di Ricerca in Informatica

Ph.D. Thesis in Computer Science

Multi-resolution shape analysis based on discrete Morse decompositions

by

Iuricich Federico

March, 2014

Dottorato in Scienze e Tecnologie dell'Informazione e della Comunicazione
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università degli Studi di Genova

DIBRIS, Univ. di Genova
Via Opera Pia, 13
I-16145 Genova, Italy
`http://www.dibris.unige.it/`

Ph.D. Thesis in Computer Science (S.S.D. INF/01)

Submitted by Iuricich Federico
DIBRIS, Univ. di Genova
`federico.iuricich@unige.it`

Date of submission: 21 February 2013

Title: Multi-resolution shape analysis based on discrete Morse decompositions

Advisor: Leila De Floriani
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università di Genova
`deflo@disi.unige.it`

Ext. Reviewers:
Silvia Biasotti
Istituto di Matematica Applicata e Tecnologie Informatiche
Consiglio Nazionale delle Ricerche, Genova
`silvia.biasotti@ge.imati.cnr.it`

Hans Hagen
Computer Graphics and HCI Group
University of Kaiserslautern
`hagen@informatik.uni-kl.de`

Abstract

Representing and efficiently managing scalar fields and morphological information extracted from them is a fundamental issue in several applications including terrain modeling, static and dynamic volume data analysis (i.e. for medical or engineering purposes), and time-varying 3D scalar fields. Data sets have usually a very large size and adhoc methods to reduce their complexity are needed. Morse theory offers a natural and mathematically-sound tool to analyze the structure of a discrete scalar field as well as to represent it in a compact way through decompositions of its domain. Starting from a Morse function, we can decompose the domain of the function into meaningful regions associated with the critical points of the field.

Such decompositions, called ascending and descending Morse complexes, are characterized by the integral lines emanating from, or converging to, some critical point of a scalar field. Moreover, another decomposition can be defined by intersecting the ascending and descending Morse complexes which is called the Morse-Smale complex. Unlike the ascending and descending Morse complexes, the Morse-Smale complex decomposes the domain into a set of regions characterized by a uniform flow of the gradient between two critical points. In this thesis we address the problem of computing and efficiently extracting Morse representations from a scalar field.

The starting point of our research is defining a representation for both ascending and descending Morse complexes. We have defined a dual representation for the two Morse complexes, called Morse incidence graph. Then we have fully investigated all the existing algorithms to compute a Morse or Morse-Smale complex. Thus, we have reviewed most important algorithms based on different criteria such as discrete complex used to describe the domain, features extracted by the algorithm, critical points used to perform the extraction and entities used by the segmentation process. Studying such algorithms has led us to investigate the strengths and weaknesses of both the Morse theory adaptations to the discrete case, piecewise-linear Morse theory and the discrete Morse theory due to Forman. We have defined and investigated two dimension-independent simplification operators to simplify a Morse complex and we have defined them in terms of updates on the Morse complexes and on the Morse

incidence graph.

Thanks to such simplification operators and their dual refinement operators, we have defined and developed a multi-resolution model to extract morphological representations of a given scalar field at different resolution levels. A similar hierarchical approach has been used to define and develop a multi-resolution representation of a cell complex based on homology-preserving simplification and refinement operators which allows us to extract representations of a cell complex at different resolutions, all with same homology of the original complex, and to efficiently compute homology generators on such complexes.

Table of Contents

Introduction

The aim of this thesis is the study of methods, rooted in Morse theory, for the topological analysis of scalar fields. Topological analysis of discrete scalar fields is an active research field in computational topology and in scientific data visualization. We will consider scalar fields $\mathbb{M}$ described as a pair (M, f) where M is a geometric object and f a function which defines a scalar value for any point in M. The manifold domain M of function f will be discretized through geometric objects represented as composition of simple cells or simplexes. The size and the complexity of available data sets defining scalar fields are increasing, and thus the definition of compact topological representations is a first step in building tools capable of analyzing large data sets effectively and efficiently. In the continuous case, Morse and Morse-Smale complexes have been recognized as convenient and theoretically well-founded representations for modeling both the topology of the manifold domain M, and the behavior of a scalar field f over M. Morse and Morse-Smale complexes have been introduced in geographic data processing and in computer graphics for the analysis of 2D scalar fields, and specifically for terrain modeling. In tho case, the domain is a region in the plane, and the scalar field is the elevation function. Recently, Morse and Morse-Smale complexes have been considered as a tool for analyzing 3D functions. They are used in scientific visualization, where data are obtained through measurements of scalar field values over a volumetric domain, or through simulation. With an appropriate selection of the scalar function, Morse and Morse-Smale complexes are also used for segmenting molecular models to detect cavities and protrusions, which influence interactions among proteins. Morse complexes of the distance function have been used in shape matching and retrieval. Scientific data, obtained either through measurements or simulation, is usually represented as a discrete set of vertices in a 2D or 3D domain M, together with function values given at those vertices. In this thesis we will consider mainly simplicial meshes as topological structures computed on a set of vertices on which f has been defined.

Our final goal is the development of multi-resolution techniques for scalar field analysis based on discrete Morse decompositions. We can describe the main contributions of these thesis in terms of four topics, necessary to the definition of a multi-resolution model: **computation**, **representation**, **simplification** and **multi-resolution** representation of Morse and Morse-Smale complexes.

We have started our investigation by studying the algorithms proposed in the literature for **computing** Morse and Morse-Smale complexes. Algorithms for extracting an approximation of Morse and Morse-Smale complexes, from a scalar field defined on the vertices of a simplicial mesh Σ, have been extensively studied in 2D. Recently, some algorithms have been proposed for dealing with scalar data in higher dimensions. They can be classified based on many criteria: the dimension on which they work, the Morse or Morse-Smale complexes they compute or the combinatorial manifold used to discretize the domain. We will classify them based on the approach used to extract the Morse and Morse-Smale cells, indicating them as *region-based*, *boundary-based*, *watershed-based* or *Forman-based.* Through theoretical and experimental comparisons we have analyzed their properties and drawbacks and we have adapted, the two most promising, for working with scalar fields defined on tetrahedral meshes. We have extended a watershed-based algorithm based on simulated immersion [VS91] and a Forman-based algorithm for computing a Forman gradient [RWS11]. We have adapted both of them to the simplicial case in 2D and 3D [CFI10, FIMS13, WIFF13] evaluating, through experimental comparisons, their performances [FIMS13].

We have then addressed the problem of **representing** Morse complexes in a compact way by exploiting their natural duality. We have focused on representing Morse and Morse-Smale complexes computed on unstructured simplicial meshes discretizing the domain of a scalar field. We have defined a dimension-independent data structure, where the topology of both the Morse complexes,Γ_a and Γ_d, is represented by encoding the immediate boundary and co-boundary relations of Γ_a and Γ_d in the form of an incidence graph [Ede87]; the structure, called a *Morse Incidence Graph (MIG)*, stores the top cells of both the ascending and descending Morse complexes. Then, we have proposed a new definition for the cells of the Morse complexes computed on a simplicial mesh [WIFF13]. We used a duality argument to define the cells of the descending Morse complex in terms of the primal simplicial mesh and the cells of the ascending Morse complex in terms of its dual mesh. The Morse-Smale complex is described, combinatorially, as collection of cells from the intersection of the primal and dual meshes. The description is entirely independent of the dimension of the complex. Inspired by this representation we have defined and implemented a simple compact encoding for the gradient field attached to the top simplexes of the primal mesh. The encoding is suitable to be combined with any data structure for the mesh which encodes just the vertices and the top simplexes. We have implemented this approach in combination with a common topological data structure, the *Indexed data structure with Adjacencies (IA data structure).*

Although Morse and Morse-Smale complexes represent the topology of M and the behavior of f in a much more compact way than the initial data set at full resolution, the **simplification** of these complexes is a necessary step for the analysis of noisy data sets. All the approaches known in the literature to the simplification of Morse and Morse-Smale complexes are based on the *cancellation* operator defined in Morse theory [Mat02]. In 2D, a cancellation eliminates critical points of f, reduces the incidence relation on the Morse complexes, and eliminates cells from the Morse-Smale complexes. The major problem with using cancellation is that it may increase the size of the Morse-Smale complex in three and higher dimensions, when the cancellation does not involve a minimum or a maximum, thus causing memory problems when dealing with large-size data sets. We have addressed the problem of **simplifying** Morse and Morse-Smale complexes and developed an approach based on an atomic simplification operator, called *remove* [CF11], which is entirely dimension-independent, never increases the size of the Morse or Morse-Smale complexes, and defines a minimally complete basis for expressing any simplification operator on such complexes.

We have studied the effect of $remove$ operator on the MIG and showed that it is simple to implement, in a completely dimension-independent way, and its effect on the MIG is always local [CFI11]. We have also implemented the cancellation operator on the MIG in the 3D case and compared it with the 3D instances of the $remove$ operator [CFI13b]. We have shown that the size of the simplified MIG produced by $remove$ is always smaller than the graph produced by cancellation. In addition to the $remove$ operator, we describe the dual $insert$ operator defined for refining a Morse complex. We present its effect in updating the ascending and descending Morse complexes and on the MIG. Simplification operators, together with their inverse refinement ones, form a basis for the definition of a multi-resolution representation of Morse and Morse-Smale complexes.

A **multi-resolution** representation of the topology of a scalar field is crucial for interactive analysis and exploration of terrain, static and time-varying volume data sets, in order to maintain and analyze their characteristic features at different levels of detail and reduce the size of their representation. Some approaches for building such multi-resolution representations in 2D have been proposed. In higher dimensions, such hierarchies are based on a progressive simplification of the initial full-resolution model. We have proposed a dimension-independent, graph-based model for multi-resolution representation of Morse and Morse-Smale complexes, which describes the topology of scalar fields in arbitrary dimensions [CFI12]. We call this model a *Multi-Resolution Morse Incidence Graph* ($MMIG$). An $MMIG$ is generated from the iterative simplification of the MIG at full resolution, and consists of the MIG representing the complexes at the coarsest resolution, of a set of refinement modifications reversing the simplification ones applied in the generalization phase, and of a dependency relation among such modifications. Note that an $MMIG$ provides also a multi-resolution description of the Morse-Smale complex. An $MMIG$ is capable of supporting the extraction of the graph which best approximates the topology of the field under given requirements, which depend on the specific application. It is possible to select from an $MMIG$ subsets of refinement modifications consistent with the partial order defined by the dependency relation. Their application on the graph at the coarsest resolution produces a variety of graphs in which the resolution (defined by a suitable error criterion) is uniform or varies over the domain of the scalar field.

The $MMIG$ however, provides a multi-resolution description of the topology of a scalar field $\mathbb{M}_\Sigma = (\Sigma, f)$, which is only a component of an effective tool for the analysis of $\mathbb{M}$. Thus, we have defined and developed a new multi-resolution model able to inspect a scalar field both from a geometric and morphological point of view. The model, called the *Multi-Tessellation based on Forman gradient*, developed for triangulated terrains, is built from a sequence of edge-collapse operators, reducing the size of the triangle mesh, and from a sequence of $remove$ operators, reducing the topology of $\mathbb{M}$.

Inspired by the multi-resolution Morse complex, we have defined a hierarchical model that we call a *Hierarchical Cell Complex (HCC)* based on a set of homology-preserving operators [CFI13c]. We have used such model to compute the homology and homology generators on various cell complexes efficiently. We describe, in Chapter 6, a set of modeling operators and we prove that they form a minimally complete basis for simplifying and refining cell complexes in arbitrary dimensions in a topologically consistent manner. We distinguish between operators that maintain the homology of the complex, and operators that modify it in a controlled manner. *Homology-preserving* operators add (or remove) a pair of cells of consecutive dimension, but they do not change the Betti numbers of the complex. *Homology-modifying* operators add (or remove) an i-cell, and increase (or decrease) the ith Betti number. We compare our modeling operators with other operators on cell complexes proposed in the literature, and we show how these latter can be expressed in terms of the former.

In our work, we have also investigated the use of the HCC model for homology computation [CFI13c]. We show that the HCC based on the homology-preserving operators enables us to obtain the homology (with coefficients in $\mathbb{Z}_2$) of all the complexes encoded in the model by computing the homology of the complex at the coarsest resolution using standard techniques [Ago05]. Moreover, we are able to construct homology generators of a complex at any intermediate resolution by computing generators on the coarsest complex and using the hierarchical model to propagate the computed generators to all the complexes at intermediate resolutions. Unlike approaches based on pyramids on n-maps and n-G-maps [DGDP12, PIH$^+$07], the version of the HCC based on homology-preserving operators can be applied to general complexes, not only to quasi-manifolds and it supports the extraction of homology generators at variable resolutions.

The last chapter of the thesis deals with the analysis of hypersurfaces in 4D space by computing Morse decompositions based on discrete 3D discrete distortion. Discrete distortion is defined by considering the graph of the discrete 3D field, which is a tetrahedral hypersurface in $\mathbb{R}^4$, and measuring the distortion of the transformation which maps the tetrahedral mesh discretizing the scalar field domain into the mesh representing its graph in $\mathbb{R}^4$. We analyze the 3D field by using a multi-resolution model based on clusters of tetrahedra, called diamonds, which enables the analysis of the field through crack-free approximations encoded as tetrahedral meshes. One important aspect of using mesh-based multi-resolution models is that the scale field can be analyzed by using much fewer samples than in the data set at full resolution. This facilitates our analysis of large 3D volume datasets by using significantly fewer resources. The aspect we will through our experiments is the utility of discrete distortion in analyzing approximated scalar field, thus giving good insights about the field behavior already at low resolutions [FIM$^+$12]. We then present a new discrete operator generalizing a discrete estimator for mean curvature, called *extrinsic distortion* to nD and deriving a weighting that can be provably used to compute mean curvature on such hypersurfaces. We analyze the behavior of the operator on 3-manifolds in 4D, comparing it to the well known Laplace-Beltrami operator, using ground-truth analytic surfaces with varying conditions of resolution, sampling distribution, and noise. We also investigate it in the context of an application that uses the mean curvature field to obtain a volumetric segmentation, examining the stability of the segmentations under increasing noise. In each case, we have shown that extrinsic distortion behaves similarly or better than the Laplace-Beltrami operator, while being intuitively simple and easy to implement [SFIM13].

This thesis is organized as follow. In Chapter 1 we will introduce some basic notions , fundamental for the understanding of the rest of the document. We will introduce the notion of *cell* and *cell complex* as well as those of *simplex* and of *simplicial complex*. Moreover, we will present *Morse theory* as the tool used to describe the morphology of a scalar field and the two approaches proposed in literature extending Morse theory to the discrete case;: piecewise linear Morse theory formalized by Banchoff [Ban67, Ban70] and discrete Morse theory, presented by Forman [For98, For02]. A Morse complex can also be defined using concepts related to the *watershed transform* [RM00]; we will introduce here its definition in terms of *catchment basin* and *watershed*. In Chapter 2, the state of the art on scalar field analysis, based on Morse theory, is presented. Here, we deal with three main topics: *computation*, *simplification* and *hierarchical representation* of Morse and Morse-Smale complexes. In Chapter 3 the actual contribution of our work is presented starting form the description of the *Morse Incidence Graph*. We present a characterization of the Morse complexes in terms of cells of the primal/dual mesh as well as a description of the Morse-Smale complex in terms of cells of the dually subdivided mesh obtained from the intersection of the primal and dual meshes. In Chapter 4, we present the $remove$ and $insert$ operators, we describe their behavior in terms of updates on the Morse complexes and the corresponding updates on the MIG structure. Moreover, we present the results we have obtained by comparing such operators with the $cancellation$ operator defined in Morse theory. In Chapter 5 we describe the multi-resolution model for the morphology of a scalar field $\mathbb{M}$. Beyond the description of the model, called the *Multi-resolution Morse complex ($MRMC$)*, we describe a practical implementation of the $MRMC$. The representations of the Morse complexes, that can be extracted from the $MRMC$, are encoded as a Morse Incidence Graph; thus we call the implemented multi-resolution model, Multi-resolution Morse Incidence Graph ($MMIG$). At the end of this chapter we present the results obtained for 2D and 3D scalar fields as well as a comparison with a similar model presented in [BHEP04]. As improvement of the $MMIG$ tool used to inspect only the morphology of a scalar field, in Chapter 7 we present the Multi-tessellation based on Forman gradient ($MTFG$); such model encodes different representations of a two-dimensional scalar field both from the morphological and geometrical point of view. We present a formal description as well as the encoding used to implement the model. We will also compare our model to the only multi-resolution model known, having similar properties, the Multi-resolution Morse Triangulation (MMT) described in [DDFMV10]. In Chapter 8 we will present our results in the analysis of 3D scalar fields by computing Morse decompositions based on 3D discrete distortion. We will present a new discrete 3D curvature estimator, the *extrinsic* distortion, and the results obtained applying distortion-based segmentations to scalar field analysis. Finally in Chapter 9, we present concluding remarks and discuss current and future developments of the work of this thesis. Appendix A contains a summary of the datasets used during our work.

Chapter 1

Background Notions

In this thesis we consider scalar fields $\mathbb{M}$ which can be described as a pair $\mathbb{M} = (M, f)$ where M is a manifold and f a function which defines a scalar value on the graph of M. Although manifolds exist in the continuum, we need discrete models in order to handle them computationally. In our work we consider discretizations of manifolds with pieces of simple geometry like simplexes or, more in general, cells.

We introduce the notions of *cell complex* and *simplicial complex* as well as a particular instance of a *cell*, the *simplex*. In Section 1.2 we present *Morse theory*, a tool used to described the morphology of a scalar field in the continuum studying its *critical points*, *integral lines* and their regions of influence.

In literature there are two approaches that extend the results of Morse theory to the discrete case: *piecewise-linear Morse* theory formalized by Banchoff [Ban67, Ban70] and *discrete Morse theory*, presented by Forman [For98, For02]. Piecewise-linear Morse theory transposes the results obtained on smooth functions to piecewise-linear functions. It has been widely used with two-dimensional scalar fields and adapted to three dimensions. We will present the main results provided by this theory in Section 1.3. Discrete Morse theory, extends Morse theory to the discrete case through a combinatorial point of view assigning a function value to all the cells of a regular cell complex Γ on which the original scalar field is defined. We will present the interesting results provided by discrete Morse theory in Section 1.4.

A Morse complex can also be defined using concepts related to the *watershed transform* [RM00]. The *watershed transform* induces a subdivision of the domain of a C^2 function in regions of influence associated with critical points. If f is a Morse function the regions of influence of the critical points are equivalent to the cells of the *Morse complex*. We will introduce the basic notions of the *watershed transform* in Section 1.5.

1.1 Cell complexes and Simplicial complexes

In algebraic topology many tools are defined to associate certain algebraic invariants to a space. The basic tool for computing these invariants consists of breaking a topological space into pieces (cells or simplexes) generating a combinatorial structure [LW69, Hat01].

A *topological space* is a mathematical structure used to extend concepts from the Euclidean space $\mathbb{R}^d$ to arbitrary sets of points. Continuity, closeness, limits are all example of concepts well defined on an Euclidean space $\mathbb{R}^d$ and generalized to a set of points using relationships between sets.

Definition 1.1.1. *A* **topological space** *is a set X, together with a collection S of subsets of X called* **open sets***, satisfying the following axioms:*

- *$\emptyset$ and X are open,*
- *arbitrary union of open sets is open,*
- *finite intersection of open sets is open.*

The collection S is also called a *topology* of X.

Definition 1.1.2. *A topological space in which distinct points have disjoint neighborhoods is called* **Hausdorff space***.*

An example of a Hausdorff space is the *manifold.* Intuitively, a *manifold* is a topological space, locally Euclidean, such that around each point there is a neighborhood that is topologically the same as the open unit ball in $\mathbb{R}^d$. More formally,

Definition 1.1.3. *A topological Hausdorff space M is called an* **n-dimensional manifold** *(n-manifold) if there is an open cover $\{U_i\}_{i \in A}$ of M such that for each $i \in A$ there is a map $\phi_i : U_i \to \mathbb{R}^n$ which maps U_i homeomorphically onto the open n-dimensional disk $D^n = \{x \in \mathbb{R}^n | \, ||x|| < 1\}$.*

Definition 1.1.4. *An* **n-manifold with boundary** *is a Hausdorff space in which each point has an open neighborhood homeomorphic either to the open disk D^n or the half-space $\mathbb{R}^{n-1} \times \{x_n \in \mathbb{R} | \, x_n \geq 0\}$.*

Many examples of topological spaces exist and some of them, such as cell and simplicial complexes, are also used in several application domains. Cell complexes and simplicial complexes are examples of topological spaces obtained through the structured composition of simple elements (cells or simplexes, respectively).

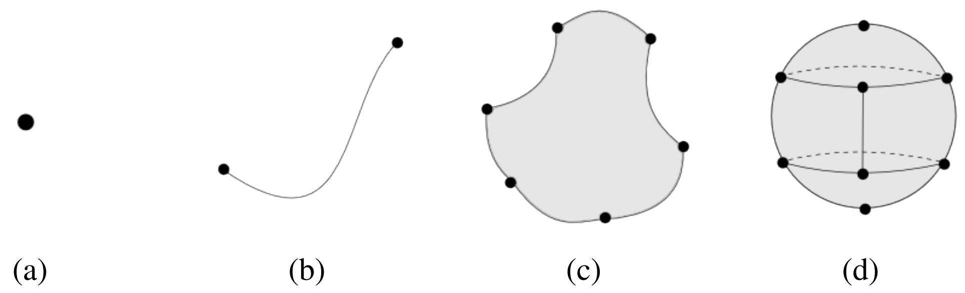


Figure 1.1: An example of (a) 0-cell; (b) 1-cell; (c) 2-cell and (d) 3-cell.

Definition 1.1.5. *A closed* **Euclidean cell** γ *of dimension* k *in* E^d *with* $0 \leq k \leq d$*, also called a* k*-cell, is a subspace of the Euclidean space* E^d *homeomorphic to the* k*-dimensional ball* $B^k = \{x \in \mathbb{R} \,|\, ||x|| \leq 1\}$.

Attaching the k-cell γ to a space X by the continuous map $\phi : S^{k-1} \to X$, with $S^{k-1} = \{x \in \mathbb{R}^k \,|\, ||x|| = 1\}$ the boundary of B^k, requires taking $X \cup B^k$, where each point $x \in S^{k-1}$ is identified with the point $\phi(x) \in X$. The space obtained is denoted $X \cup_\phi \gamma$.

Definition 1.1.6. *A* **Euclidean cell complex** Γ *of dimension* n *is the space* X *obtained by subsequently attaching finitely many cells.*

This means that there exists a finite nested sequence $\emptyset \subset X_0 \subset X_1 ... \subset X_i = X$ such that, for each $h = 1, 2, ..., i$ X_i is the result of attaching a cell to X_{i-1}.

In Figure 1.1 different examples of a cell are shown. In Figure 1.1(a) a 0-cell (a point) is shown. In Figures 1.1(b) and (c) a 1-cell and a 2-cell are shown, respectively. In Figure 1.1(d) a 3-cell is shown.

A cell is called *regular* if it has one attaching map ϕ bijective on all E^d. A cell complex is *regular* if all its cells are regular. A cell complex is *finite* or *countable* if its set of cells is finite or countable, respectively.

In Figure 1.2(a) a regular cell complex is shown. In Figure 1.2(b) an irregular cell complex, composed of one regular 0-cell, one regular 2-cell and an irregular 1-cell, is shown.

Let Γ' a cell complex, subcollection of a cell complex Γ, we say that Γ' is a *subcomplex* of Γ. Let $k \leq dim\Gamma$, the collection of all cells of Γ of dimension at most k is a subcomplex of Γ, it is called the k-skeleton of Γ and it is denoted by Γ^k. The 0-cells of collection Γ^0 are called *vertices* of Γ. Moreover, for $k \geq 0$, $\Gamma_k \subseteq \Gamma^k$, where Γ_k is the set of k-cells $\in \Gamma$.

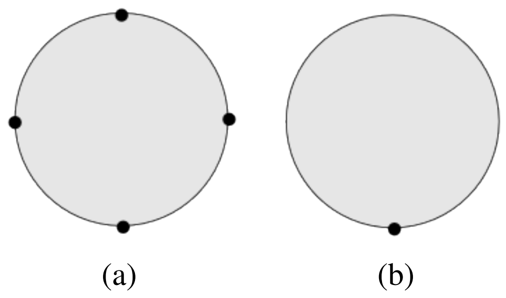


Figure 1.2: (a) A regular cell complex and (b) an irregular cell complex.

The *combinatorial boundary* of a cell $\gamma \in \Gamma$, denoted as $B(\gamma)$, is the set of cells $\gamma' \in \Gamma$ such that $\gamma' \cap \gamma = \gamma'$. Let γ' a k-cell on the boundary of γ then γ' is said to be a *k-face* of γ or incident in γ. The *immediate boundary* of a k-cell γ, denoted $b(\gamma)$, is the set of i-cells $\gamma' \in B(\gamma)$ with $i = k-1$.

The *combinatorial co-boundary* (or *star*) of a cell $\gamma \in \Gamma$, denoted $CB(\gamma)$ is the set of cells $\gamma' \in \Gamma$ that have γ in their combinatorial boundary. Let γ' a k-cell on the co-boundary of γ then γ' is said to be a *k-coface* of γ. The *immediate co-boundary* of a k-cell γ, denoted $cb(\gamma)$, is the set of i-cells $\gamma' \in CB(\gamma)$ with $i = k + 1$.

A k-cell γ is said to be *adjacent* to a k-cell γ' if they share a $k - 1$-face.

The *link* of a cell $\gamma \in \Gamma$, denoted as $Lk(\gamma)$ is the set of cells forming the combinatorial boundary of the cells in $CB(\gamma)$ and that do not intersect γ.

A cell $\gamma \in \Gamma$, which is not on the combinatorial boundary of any other cell $\gamma' \in \Gamma$, is said to be a *top cell*. If γ has also the same dimension of Γ it is called a *maximal cell*. An n-dimensional complex Γ is called *pure* if all its top cells are n-cells. Thus, in a pure cell complex, maximal cells and top cells are equivalent.

At this point we still need to relate the cellular structure of a cell complex to a topology on the set X. The result is called *CW complex* [LW69].

Definition 1.1.7. *A topological space X is a* **CW complex** *if and only if there is a sequence of closed subspaces $X_0 \subset X_1 \subset ... \subset X$ such that $X = \cup_i X_i$ and:*

- *every subset of X is open,*
- *for each i, X_i is obtained by attaching i-cells;*
- *the space X has the weak topology with respect to the closed sets X_i*

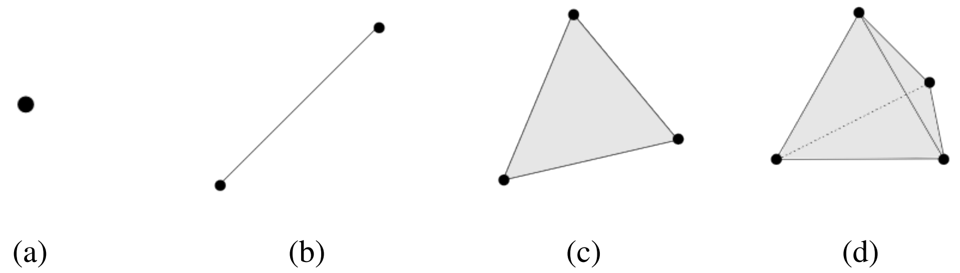


Figure 1.3: (a) A 0-simplex, (b) 1-simplex, (c) 2-simplex and (d) 3-simplex.

Although cell complexes represent a wide class of spaces, more restrictive classes with a stronger combinatorial nature are often preferred. One of the most relevant example of a combinatorial structure defined on a topological space is the *simplicial complex*, a complex obtained buy gluing together simple elements called *simplexes* [Mun84].

A set of points $\{v_1, v_2, ..., v_k\}$ in $\mathbb{R}^d$ is said to be *geometrically independent*, if for any real value t_i, the equations

$$\sum_{i=0}^{k} t_i = 0 \quad and \quad \sum_{i=0}^{k} t_i v_i = 0$$

is true only if $t_0 = t_1 = ... = t_k = 0$. This is equivalent to requiring that vectors $v_1 - v_0, ..., v_k - v_0$ are linearly independent over $\mathbb{R}$.

Definition 1.1.8. *Let $P = v_0, v_1, ..., v_k$ be a geometrically independent set in $\mathbb{R}^d$. A k-***dimensional simplex** *σ spanned by $v_0, v_1, ..., v_k$, also called k-simplex, is defined as the convex hull of $k+1$ points of P in $\mathbb{R}^d$, also called* vertices *of σ, such that,*

$$x = \sum_{i=0}^{k} t_i v_i \quad where \quad \sum_{i=0}^{k} t_i = 1,$$

and $t_i \geq 0$ for all i. The numbers t_i are uniquely determined by x and are called **barycentric coordinates** *of the point x of σ with respect to $v_0, v_1, ..., v_n$.*

In Figure 1.3 different examples of a simplex are shown. In Figure 1.3(a) a 0-simplex also called point is shown. In Figures 1.3(b) and 1.3(c) a 1-simplex, also called edge and a 2-simplex, also called triangle are shown, respectively. In Figure 1.3(d) a 3-simplex, also called tetrahedron is shown.

Definition 1.1.9. *A* **Euclidean simplicial complex** *Σ is a collection of simplices in $\mathbb{R}^n$ such that:*

- *every face of a simplex of Σ is in Σ,*
- *the intersection of any two simplices of Σ is a face of each of them.*

A simplicial complex can be seen as an instance of a cell complex in which all the cells are simplexes. Thus, the *dimension of a simplicial complex* Σ, denoted $dim\Sigma$, is defined as the dimension of the largest simplex of Σ and an n-simplex $\sigma \in \Sigma$ is called *maximal simplex* if $dim\Sigma = n$. A simplex of Σ which is not a proper face of any other simplex in Σ is called *top simplex*. A maximal simplex is also a top simplex while the converse does not hold in general.

Let us consider a simplicial n-complex Σ. An h**-path** between two $(h+1)$-simplices in Σ, where $h = 0, 1, \cdots, n-1$ is a path formed by an alternating sequence of adjacent h-simplices and $(h+1)$-simplices.

A complex Σ is said to be h**-connected** if and only if there exists an h-path joining every pair of $(h+1)$-simplices in Σ. Any maximal h-connected subcomplex of a n-complex Σ is called an h**-connected component**.

Definition 1.1.10. *A regular $(n-1)$-connected n-complex in which the star of any $(n-1)$-simplex consists of at most two n-simplices is called a n-**pseudo-manifold**.*

Definition 1.1.11. *A n-pseudo-manifold is called n-**manifold complex** if its underlying space in $\mathbb{E}^d$ is a n-manifold (with or without boundary), $n \leq d$.*

It can be easily shown that a simplicial n-complex is a manifold n-complex if and only if the link of every vertex is homeomorphic to the $(n-1)$-sphere S^{n-1} or to the $(n-1)$-disk D^{n-1} (see [Ago05]).

In the rest of this thesis we will work always with combinatorial simplicial complexes that we will call simplicial meshes. A simplicial meshes is called triangle mesh in the 2D case and a tetrahedral mesh in the 3D case.

Simplicial complexes are a natural choice to represent geometric objects computationally because they can be easily implemented by encoding the top simplexes. Form the top simplexes of a simplicial complex Σ we will combinatorially generate all simplexes in Σ , it is sufficient to take all their proper faces to construct the complex. Differently from other pure complexes, such as cubical complexes, simplicial complexes are particularly suitable to describe geometric objects defined by finite set of points irregularly distributed.

We recall that a scalar field $\mathbb{M}$ is defined, in the continuum, as a pair (M, f) where M is a manifold and fa function which defines a scalar value for any point in M. Working in a discrete domain M will be replaced by a cell complex Γ, or a simplicial mesh Σ. Thus, a scalar field defined on a discrete object will be indicated as a pair $\mathbb{M}_\Gamma = (\Gamma, f)$ or as a pair $\mathbb{M}_\Sigma = (\Sigma, f)$, when the object is discretized through a cell or simplicial complex, respectively. In both cases, function f will be defined on the vertices of the complex.

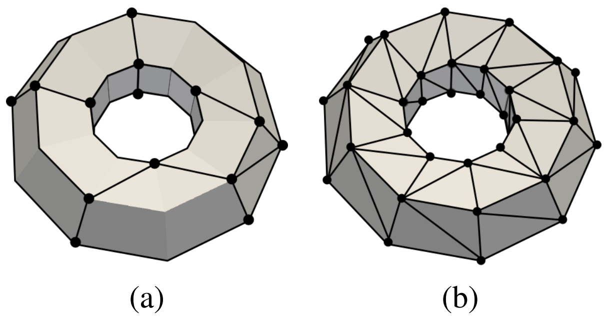


Figure 1.4: (a) A cell complex decomposing a torus; (b) a simplicial complex decomposing the same torus.

1.2 Morse Theory

Morse theory [Mil63, Mat02] studies the relationships between the topology of a manifold M and the critical points of a scalar function defined on it.

Definition 1.2.1. *Let f be a C^2-differentiable real-valued function defined over a domain $M \subseteq \mathbb{R}^d$. A point $p \in \mathbb{R}^d$ is a* **critical point** *of f if and only if the gradient $\bigtriangledown f$ of f vanishes on p, i.e.,*

$$\bigtriangledown f(p) = 0.$$

Definition 1.2.2. *The Hessian matrix of f, denoted $Hess(f)$, is the matrix of the second-order partial derivatives of the function f:*

$$Hess(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

The Hessian matrix gives fundamental information on the critical points of a function f. If the determinant of the Hessian matrix of f in p is either positive or negative (i.e. $det(Hess_p(f)) \neq 0$), then p is a *non-degenerate critical point.* Then, the number of negative eigenvalues of $Hess_p(f)$ is called the index i of a critical point p and p is called an i-saddle, with $0 \leq i \leq n$. A 0-saddle

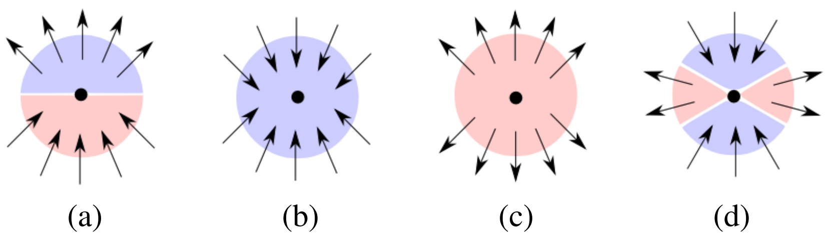


Figure 1.5: Classification of non-degenerate critical points in 2D. (a) A regular point, (b) a local maximum, (c) a local minimum and (d) a saddle. Arrows represent gradient arrows, red regions contain points with higher function value and blue regions contain points with lower function value.

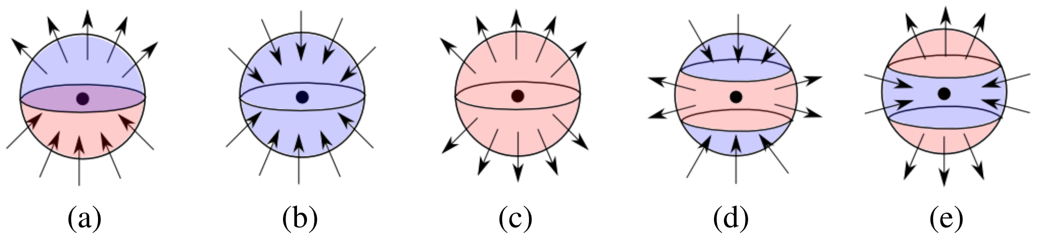


Figure 1.6: Classification of non-degenerate critical points in 3D. (a) A regular point, (b) a local maximum, (c) a local minimum, (d) a 1-saddle and (e) a 2-saddle. Arrows represent gradient arrows, red regions contain points with higher function value and blue regions contain points with lower function value.

is usually called *minimum* and an n-saddle *maximum*. The corresponding eigenvectors show the directions in which f is decreasing.

Figures 1.5 and 1.6 illustrate a neighborhood of a non-degenerate critical point in two and three dimensions, respectively. In both Figure 1.5(a) and Figure 1.6(a) a regular point is shown. It is characterized by a single region of incoming gradient arrows and a single region of outgoing gradient arrows. In Figures 1.5(b) and 1.6(b) a minimun is shown: all the gradient arrows are incoming. In Figures 1.5(c) and 1.6(c) a maximum is shown: all the gradient arrows are outgoing. In the 2D case, a saddle is characterized by two different regions of incoming gradient arrows and two regions of outgoing gradient arrows (see Figure 1.5(d)). In the 3D case there are two kinds of saddle, 1-saddles and 2-saddles, shown in Figure 1.6(d) and Figure 1.6(e), respectively. A 1-saddle is characterized by two regions of outgoing arrows and a single region of incoming arrows, while a 2-saddle has the opposite set of arrows.

Consider a scalar field $\mathbb{M} = (M, f)$ where M is a manifold of dimension n and f is a C^2-differentiable real-valued function defined on M which associates a real value to each point of M.

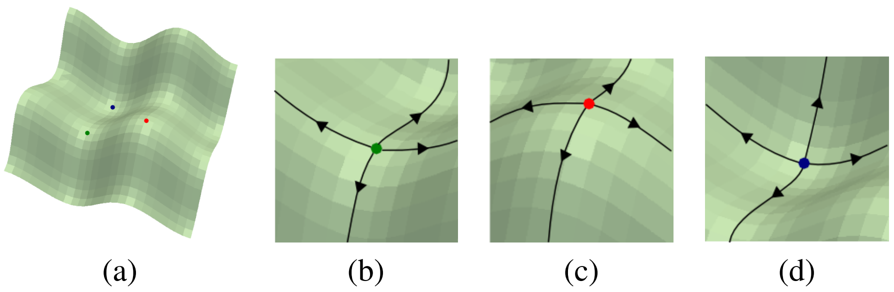


Figure 1.7: Example of a Morse function and three critical points on it, a saddle (b), a maximum (c) and a saddle (d).

Definition 1.2.3. *Function f is said to be a* **Morse function** *if and only if all its critical points are not degenerate.*

In Figure 1.7 an example of a Morse function is shown with the neighborhood of three critical points highlighted.

Morse Lemma [Mil63]. *For a Morse function, there is a neighborhood of a critical point $p = (p_1, ..., p_n)$, in which f can be expressed in a local coordinate system $(x_1, ..., x_n)$ as*

$$f(x_1, ..., x_n) = f(p_1, ..., p_n) - x_1^2 - ... - x_i^2 + x_{i+1}^2 + ... + x_n^2$$

where i is the **index** *of p in f.*

Thus, a Morse function can be expressed as a canonical quadratic form in some neighborhood of a critical point.

As a consequence of Morse Lemma, we see that non-degenerate critical points are isolated. Therefore, it is always possible, for a critical point p, to find a neighborhood of p not containing other critical points. In Figure 1.7 an example of a Morse function is shown with the neighborhood of three critical points highlighted.

Let M an n-manifold and β_i the ith Betti number of M, that measures the number of independent non-bounding i-cycles in M (see Section 6.1). Then the *Euler characteristic* of M, denoted $\chi(M)$, is $\chi(M) = \sum_{i=0}^{n}(-1)^i\beta_i$.

Critical Point Theorem [Ban67]. *If we denote with c_i the number of critical points of f with index i, and with $\chi(M)$ the Euler characteristic of M, then*

$$\chi(M) = \sum_{i=0}^{n} (-1)^i m_i.$$

Definition 1.2.4. *An* **integral line** *of a function f is a maximal path everywhere tangent to the gradient of f. An integral line l starts from a point $p = \lim_{t\to-\infty} l(t)$ called the* **origin** *of l and ends to a point $q = \lim_{t\to\infty} l(t)$ called the* **destination** *of l.*

Thus, an integral line follows the direction in which the function has the maximum increasing growth and two integral lines are either disjoint, or they are the same.

Definition 1.2.5. *An integral line which connect a critical point p of index i to a critical point q of index $i + 1$ is called* **separatrix line***.*

The integral lines cover the entire domain of f and they can be collected in cells corresponding to each critical point.

Definition 1.2.6. *Integral lines that converge to a critical point p of index i form an i-cell called the* **descending manifold** *of p.*

Definition 1.2.7. *Integral lines that originate from a critical point p of index i form an $(n-i)$-cell called the* **ascending manifold** *of p.*

The ascending/descending manifolds are pairwise disjoint and decompose the domain of M into open cells. The collection of all the cells form a cell complex since the boundary of each cell is the union of lower-dimensional cell.

Definition 1.2.8. *The collection of all the descending manifolds form the* **descending Morse complex***, Γ_d.*

Definition 1.2.9. *The collection of all the ascending manifolds form the* **ascending Morse complex***, Γ_a.*

The descending Morse complex is completely dual to the ascending Morse complex.

In Figure 1.8 red dots indicate maxima, green dots saddles and blue dots are minima. Black lines in Figures 1.8(a) and 1.8(b) represent the integral line and the separatrix lines of the dataset. Red dots indicate maxima, green dots saddles and blue dots are minima. In Figure 1.8(c) the descending Morse complex Γ_d is represented. For each maximum, the set of integral lines converging to it define a 2-cell. For each saddle the set of integral lines converging to it define a

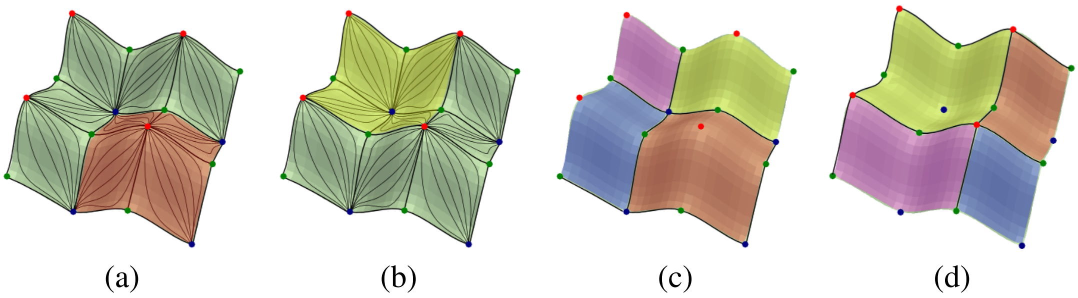


Figure 1.8: (a) The set of integral lines converging to a maximum and forming the (red) descending cell. (b) The set of integral lines originating from a minimum and forming the (yellow) ascending cell. The set of all the descending and ascending cells forming the descending Morse complex Γ_d (c) and the ascending Morse complex Γ_a (d).

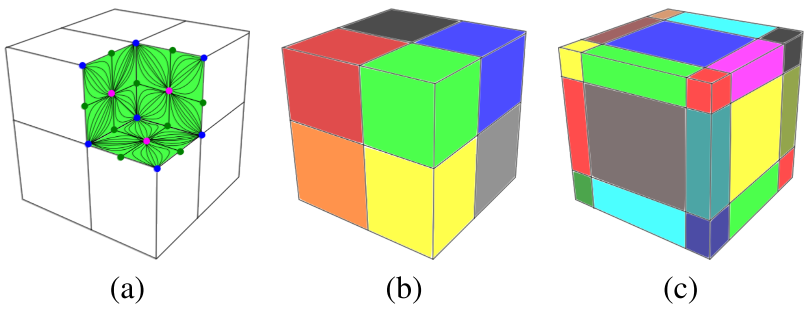


Figure 1.9: (a) The set of integral lines converging to the 2-saddles (purple dots) and forming three descending 2-cells. In (a) and (b) the Morse complexes, ascending and descending respectively, computed from Morse function $f(x, y, z) = sin(x) + sin(y) + sin(z)$.

1-cell represented as a black line. For each minimum the corresponding descending Morse cell is the point itself. In Figure 1.8(d), the ascending Morse complex Γ_a is represented. For each minimum the set of integral lines originating from it define a 2-cell. For each saddle, the set of integral lines originating from it define a 1-cell represented as a black line. For each maximum, the corresponding descending Morse cell is the point itself.

In Figures 1.9(b) and 1.9(c) the two ascending and descending Morse complexes, computed for the analytic Morse function $f(x, y, z) = sin(x) + sin(y) + sin(z)$, are shown. In Figure 1.9(a) a subset of the integral lines generating three descending 2-manifolds are shown. In the descending Morse complex Γ_d for each maximum the set of integral lines converging to it form a 3-cell (colored cubes shown in Figure 1.9(b)). For each 2-saddle (purple dots in Figure 1.9(a)) the set of integral lines converging to them form a 2-cell. For each 1-saddle (green dots in Figure 1.9(a)) the descending manifold is the 1-cell (separatrix line) connecting them to two minima (blue dots in Figure 1.9(a)) and for each blue point the corresponding descending manifold is the 0-cell itself.

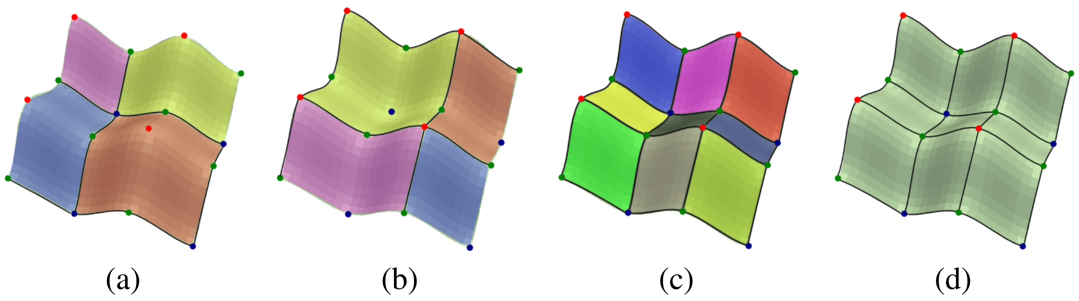


Figure 1.10: (c) The Morse-Smale complex originated from the intersection of the descending and ascending Morse complexes, (a) and (b) respectively. (d) The 1-skeleton of the Morse-Smale complex.

Dually, in the ascending Morse complex Γ_a, for each minimum the set of integral lines originating from it form a 3-cell. Moreover there is an ascending 2-cell for each 1-saddle, a 1-cell for each 2-saddle and a 0-cell for each maximum. In 2D it is often called *critical net*.

Definition 1.2.10. *A Morse function f is called a* **Morse-Smale function** *if and only if the descending and ascending Morse complexes intersects transversally,*

Thus each connected component originated by the intersection of a descending i-cell and an ascending $(n - j)$-cell, $i \geq j$, is an $(i - j)$-cell.

Definition 1.2.11. *The connected components of the intersection of descending and ascending cells of a Morse-Smale function f decompose M int a* **Morse-Smale complex***, denoted Γ_{MS}.*

If f is a Morse-Smale function, then there is no integral line connecting two different critical points of f of the same index. Each 1-saddle is connected to exactly two minima and each $(n-1)$-saddle is connected to exactly two maxima. The *1-skeleton* of the Morse-Smale complex is the sub-complex composed of 0-cells and 1-cells.

In Figure 1.10 an example of a Morse-Smale complex is shown. Each cell of the descending and ascending Morse complex in Figures 1.10(a) and 1.10(b) are intersected forming the Morse-Smale complex illustrated in Figure 1.10(c). For each critical point, the corresponding descending and ascending cells intersect at the critical point only. The cells of the Morse-Smale complex are always bounded by critical points. The 0-cells are the critical points. The 1-cells are the separatrix lines connecting pair of critical points (represented as black lines in Figure 1.10).

The 2-cells, also called *quads*, have critical points on their boundary (i-saddle, $(i - 1)$-saddle, $(i - 2)$-saddle, $(i - 1)$-saddle). In the 2D case, as shown in Figure 1.10(c), a quad is always bounded by the sequence (maximum,saddle,minimum,saddle).

In the 3D case a 3-cell, also called *crystal*, can be bounded by an arbitrary set of (maximum,2-saddle,1-saddle,2-saddle) or (2-saddle,1-saddle,minium,1-saddle) quads. However each crystal contains in its boundary only one maximum and one minimum. Only 1-saddles and 2-saddles are present, on the boundary of a crystal, in arbitrary number. The distinctive feature of Morse-Smale cells is the gradient flow: inside each cell of the Morse-Smale complex the gradient flow is uniform from one critical point to another.

In the literature two extension of Morse theory to a discrete domain can be found. Piecewise-linear Morse theory [Ban67, Ban70] transposes the results from smooth functions to piecewise-linear functions, while discrete Morse theory [For98, For02] considers a Morse function (also called a Forman function) defined on all the cells of a cell complex.

1.2.1 Simplification of Morse complexes

Simplification of Morse functions is achieved by applying an operator called *cancellation*. The cancellation operator, that we call $i\text{-}cancellation$, is a simplification operator defined in Morse theory [Mat02]. Let us consider a scalar field $\mathbb{M} = (M, f)$, an i-cancellation transform the Morse function f into a new Morse function g by removing two critical points, p and q, and modifying the gradient field of f around the integral lines of p and q.

Let p an $(i+1)$-saddle and q and i-saddle, $i = 0, \cdots, n-1$, critical points pair can be canceled if and only if there is a unique integral line connecting p and q.

After an $i\text{-}cancellation(p, q)$ the two critical points p and q are removed from the function and the integral lines originated/converging into them are modified as follows:

- the set of integral lines converging at p or q before the $i\text{-}cancellation$ are transformed into a set of integral lines converging to critical points of index $j > i$, that were the destination of integral lines starting at p before the cancellation,
- the set of integral lines that originated at q or p before the $i\text{-}cancellation$ are transformed into a set of integral lines originating at critical points v of index $k < i + 1$ that were the origin of integral lines ending at q before the cancellation.

In Figure 1.11, two examples of an $i\text{-}cancellation$ are shown. In Figure 1.11(a) a 0-cancellation is performed on a 0-saddle p and a 1-saddle q. The set of integral lines that converged to p and q are redirected into a set of integral lines converting to critical points connected with p. In Figure 1.11(b), a 1-cancellation is preformed on a 1-saddle q and a 2-saddle p. The set of integral lines, originated at p and q before the cancellation, are redirected into a set of integral lines originated at critical points connected with p.

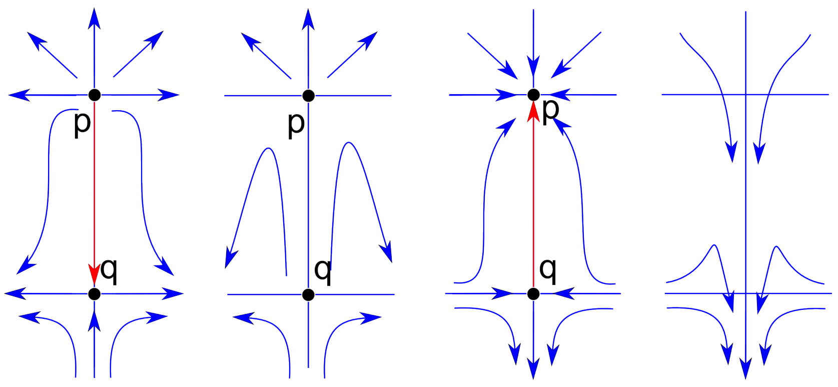


Figure 1.11: Processing of the lower star of vertex 9 using the algorithm in [RWS11].

An i-$cancellation$ transforms a Morse-Smale complex and the corresponding two Morse complexes into similar complexes with fewer cells. Recall that two cells in the Morse complexes of f are incident if and only if the corresponding critical points are connected through an integral line. An i-$cancellation$ in 2D consists of collapsing a maximum and a saddle pair into a maximum (*maximum-saddle cancellation*), or a minimum and a saddle into a minimum (*minimum-saddle cancellation*). In 3D, we have three kinds of cancellation:

- 0-$cancellation$, collapsing a minimum and a saddle pair into a minimum (*minimum-1-saddle cancellation*)
- 1-$cancellation$, collapsing a 1-saddle and a 2-saddle pair into a 1-saddle (*1-saddle-2-saddle cancellation*)
- 2-$cancellation$, collapsing a maximum and a saddle pair into a maximum (*maximum-2-saddle cancellation*)

Let p an $(i+1)$-cell and q and i-cell of a descending Morse complex Γ_d, $i = 0, \cdots, n-1$, an i-cancellation is performed on Γ_d if and only if i-cell q is incident in $(i+1)$-cell p only once. i-$cancellation(p, q)$ removes the cells p and q from the descending Morse complex changing the connectivity of the remaining cells as follows,

- all i-cells in the immediate boundary of cell p will become part of the immediate boundary of each $(i+1)$-cell incident in q,
- all the remaining cells in the neighborhood of p and q simply loose such cells from their immediate boundary/co-boundary.

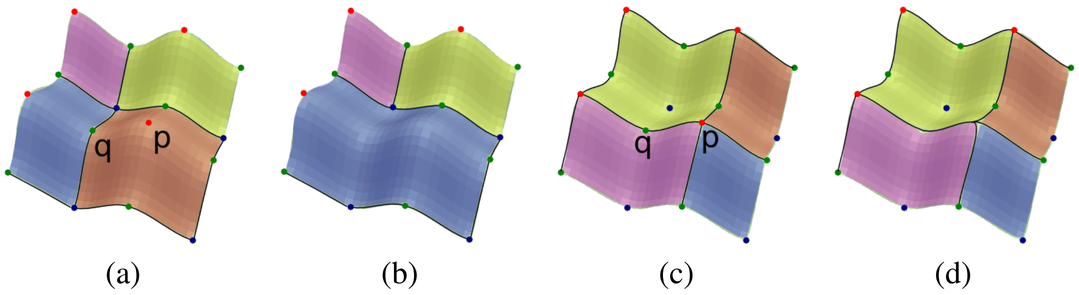


Figure 1.12: Effects of a 1-$cancellation(p, q)$ applied on a descending Morse complex Γ_d before (a) and after the simplification (b). (c) The corresponding ascending Morse complex Γ_a and (d) the effects of the same operator on Γ_a.

Dually, let p an $(i-1)$-cell and q and i-cell of a Morse complex Γ_a, $i = 0, \cdots, n-1$, the same i-cancellation is performed on Γ_a if and only if i-cell q is incident in $(i+1)$-cell p only once.

The i-cancellation removes cells p and q from the ascending Morse complex Γ_a changing the connectivity of the remaining cells as follows:

- all the $(i-1)$-cells in the boundary of cell p will become part of the immediate boundary of each $(i-1)$-cell incident in q,
- all the remaining cells in the neighborhood of p and q simply loose such cells from their immediate boundary/co-boundary.

In Figure 1.12, the effect of a 1-$cancellation(p, q)$ on the descending and ascending Morse complexes is shown. When applying the cancellation on the descending Morse complex Γ_d illustrated in Figure 1.12(a) the 2-cell p and the 1-cell q are removed from the complex. 1-cells on the boundary of p (corresponding to the green dots), are moved to the boundary of the maximum adjacent to p and the 0-cells (blue dots) in $b(q)$ are removed from the boundary of q. The same cancellation applied on the ascending Morse complex Γ_a in Figure 1.12(c) has a dual effect: 0-cell p is collapsed into 0-cell adjacent to p and the 1-cell q is deleted as well. All the 1-cells in the coboundary of p are extended to p' and q is removed from the boundary of all 2-cells in its coboundary (corresponding to blue dots).

1.3 Piecewise-linear Morse Theory

The first attempt to adapt Morse theory to the discrete case is provided by *Piecewise-linear Morse theory* [Ban67, Ban70] which extends all the results of Morse theory to polyhedral surfaces. The

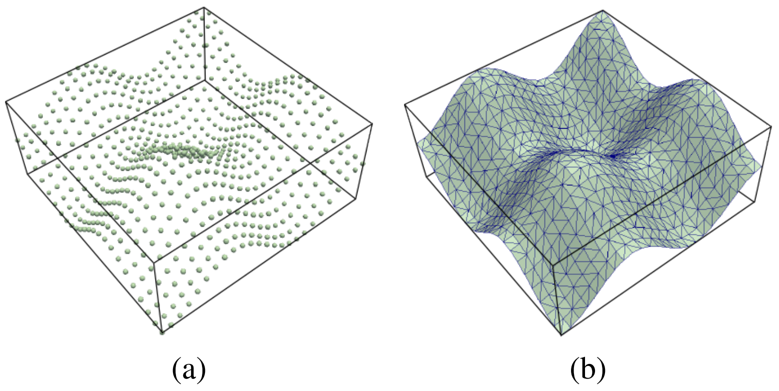


Figure 1.13: (a) Set of points, vertices of the simplicial complex Σ shown in (b).

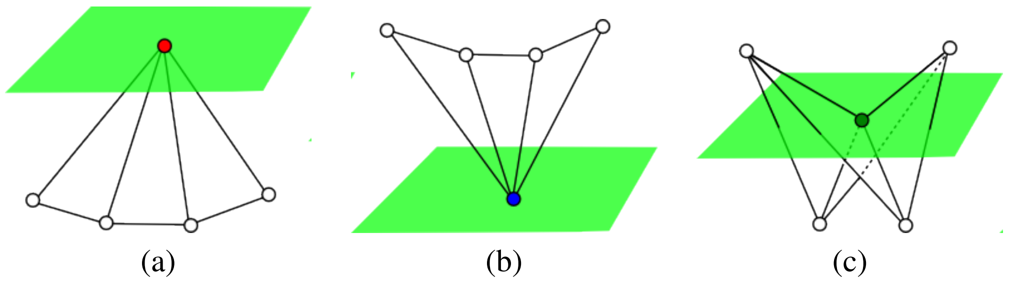


Figure 1.14: (a-b) the plane does not intersect the triangles incident in the maximum(red)/minimum(blue); (b) the plane intersects four triangles incident in the simple saddle;

basic assumption made by Banchoff is that every pair of points on the polyhedral surface have distinct field values. In Figure 1.13(a) a set of points sampled on a 2D grid and raised according the elevation at each of them (height function) is shown. In Figure 1.13(b) a simplicial complex built on this set of points is shown.

As described in [BFF$^+$08], Banchoff first introduces the definition for a critical point by using a geometric characterization [Ban67, Ban70]. Let us consider a triangle mesh as an example of a polyhedral surface. In order to define the conditions of a vertex v to be critical, the set of triangles incident to v is considered. A small neighborhood around a local maximum or a local minimum never intersects the tangent plane to the surface, intersecting the surface in v. A similar small neighborhood of a saddle is instead split into four pieces. The number of intersections is used to associate an *index* with each discrete critical point (see Figure 1.14).

Consider a triangle mesh Σ in $\mathbb{R}^3$ with height function $f : \mathbb{R}^3 \to \mathbb{R}$; f is called *general* for Σ if $f(v) \neq f(w)$ whenever v and w are distinct vertices of Σ. Then, critical points may occur only at the vertices of the triangles and the number of times that the plane, perpendicular to f, cuts the link of v is equal to the number of 1-simplexes in the link of p with one vertex above the plane

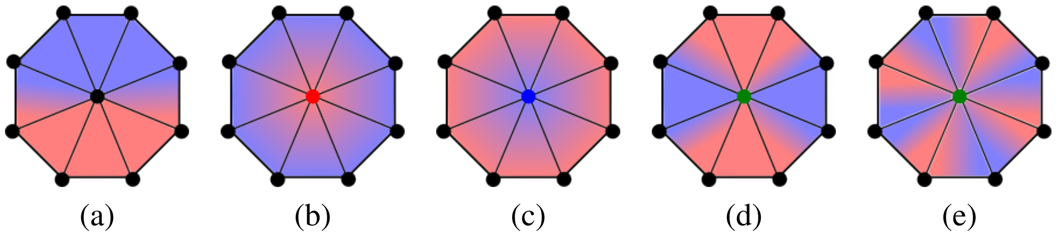


Figure 1.15: Classification of the critical points on a simplicial complex Σ. A regular point (a) maximum (b), minimum (c), simple saddle (d) and a multi saddle (e). Vertices surrounded by blue/red areas have lower/higher function value than the vertex in the center.

and one below.

In other words, the number of intersections identify the index of v. If there are no intersections v is a maximum or a minimum. If the plane divides the set of triangles in two sets the vertex is regular otherwise v is a saddle. The saddle can be *simple* if the plane divides the triangles in four pieces, otherwise it is *multiple* if the plane divides the triangles into six or more parts and it is called *multiplicity* of a saddle half of the number of these parts.

Formally, denoting n_v the number of triangles incident in v that intersect the plane, the index of v is defined as follows [Ban67]:

$$i_v = 1 - \frac{n_v}{2}$$

The critical points are defined as points with index different from 0. In particular, the index is equal to 1 for maxima and minima and an arbitrary negative integer value for saddles.

In Figure 1.15, the classification of the non-degenerate critical points is shown. In Figure 1.15(a) a regular point is shown, characterized by a sequence of vertices in the link with lower function value (vertices surrounded by blue area) and by a sequence of vertices with higher function value (vertices surrounded by red area). In Figure 1.15(b) a maximum has only vertices in the link with lower function value and dually a minimum (see Figure 1.15(c)) has only vertices in the link with higher function value. Saddle points are characterized by multiple sequences of vertices with lower and higher function value in the link. A simple saddle, shown in Figure 1.15(d), has two sets of vertices with lower function value and two sets of vertices with higher function value. A multiple saddle, often called *monkey saddle*, shown in Figure 1.15(e), has an arbitrary number, greater than two, of sets of vertices with higher/lower function value.

Moreover, Banchoff proved the *critical point theorem* [Ban67], as discrete counterpart of the critical point theorem described in Section 1.2, which holds for general height functions defined on polyhedral surfaces:

$$\sum_{v \in \Sigma} i_v = \chi(\Sigma)$$

Note that the theorem holds under the assumption that f is general and it also includes the case of isolated degenerate critical points, such as monkey-saddle, not considered by Morse theory.

The characterization provided by Banchoff correctly distinguishes critical points in dimension 2 and 3, while for higher-dimensional spaces the Betti numbers of the lower link provide a more complete characterization of discrete critical points as discussed in Section 2.1.1.

In [EHZ01, EHNP03] the notion of *Quasi Morse-Smale* (QMS) complex is introduced as a piecewise-linear counterpart of the Morse-Smale complex. The QMS is defined for 2D and 3D simplicial meshes. The QMS has the same combinatorial structure of a Morse-Smale complex; however the 1-cells in 2D and the 1-cells and 2-cells in 3D are not necessarily those of maximal ascent, or descent. The idea behind a QMS, called *simulation of differentiability* is that of extending the smooth notions to the piecewise-linear case so as to guarantee that the complex has the same structure of its smooth counterpart, and to achieve numerical accuracy via local transformations that preserve the structure of the complex. The QMS is a splittable quadrangulation of M whose vertices are the critical points of f and whose arcs are strictly monotonic in f. The 0-cells of a QMS complex are the critical points of f, the 1-cells connect minima to saddles (1-saddles in 3D), maxima to saddles (2-saddles in 3D), and, in the 3D case, 1-saddles to 2-saddles [EHNP03]. Once the QMS complex is computed, a series of operations, called *handle slides*, can be applied to turn the QMS into a Morse-Smale complex. For 2-manifolds, it is possible to find such a sequence of handle slides, while for 3-manifolds this is still an open question [EHNP03].

1.4 Discrete Morse Theory

The main purpose of *discrete Morse theory* [For98, For02] is to develop a discrete setting in which the main results from smooth Morse theory are extended to cell complexes. This goal is achieved by considering a function F defined on all the cells (and not only on the vertices) of a cell complex Γ. Since simplicial complexes are a subclass of cell complexes, all the results of discrete Morse theory on a cell complex Γ hold for simplicial complexes as well. We repeat here results from discrete Morse theory on simplicial complexes.

Let Σ be a n-dimensional simplicial complex. A function $F : \Sigma \rightarrow \mathbb{R}$, defined on Σ, is a *discrete Morse function* if for every i-simplex $\sigma \in \Sigma$, all the $(i-1)$-simplexes on the immediate boundary of σ have a lower function value than σ, and all the $(i+1)$-simplexes in the immediate co-boundary of σ have a higher function value than σ, with at most one exception.

Definition 1.4.1. *A* **discrete Morse function** *satisfies, for each i-simplex σ, both equations:*

$$\#\{\tau \in cb(\sigma)|F(\tau) \leq F(\sigma)\} \leq 1$$

$$\#\{\tau \in b(\sigma)|F(\tau) \geq F(\sigma)\} \leq 1$$

where $\#$ denotes the number of simplexes in the set.

Definition 1.4.2. *Given a discrete Morse function F, an i-simplex $\sigma \in \Sigma$ is said to be* **critical** *if and only if*

$$\#\{\tau \in cb(\sigma)|F(\tau) \leq F(\sigma)\} = 0$$

$$\#\{\tau \in b(\sigma)|F(\tau) \geq F(\sigma)\} = 0$$

and the index of σ is i.

From these two definitions it follows that an i-simplex σ is not critical if and only if exists an $(i-1)$-simplex τ such that $F(\tau) \geq F(\sigma)$ or if exist an $(i-1)$-simplex β such that $F(\beta) \leq F(\sigma)$. These two possibilities are exclusive; they cannot be true simultaneously for a given simplex σ and thus σ can be paired either with a non-critical simplex that is a co-face of σ or with one of its faces. A pair can be viewed as an *arrow* formed by a head (i-simplex) and a tail ($(i-1)$-simplex). A simplex that is not a head or a tail of any arrow is critical.

In Figure 1.16, two examples of functions defined on a simplicial complex are shown. In Figure 1.16(a), the function is not a Forman function since for the 1-simplex with function value 5 both the boundary 0-cells have higher function value (6 and 7). In Figure 1.16(b) on the same simplicial complex a Forman function is defined with a critical 0-cell with value 0. In Figure 1.16(c) the arrows indicating the pairing between the cells are shown, the critical vertex is the only one not paired with any other cell.

Forman demonstrates in [For98] the combinatorial counterpart of the *critical point theorem*.

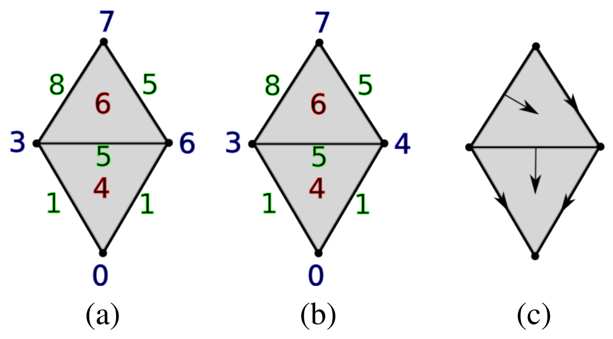


Figure 1.16: (a) Example of a function defined on a two-dimensional simplicial complex that is not a Forman function and (b) Forman function defined on the same complex. (c) The arrows defined by the function indicated in (b).

Definition 1.4.3. *Given m_i the number of critical cells of a discrete Morse function F defined on a simplicial complex Σ then,*

$$\chi(\Sigma) = \sum_{i=0}^{n} (-1)^i m_i;$$

m_i are called the Morse numbers of F.

Definition 1.4.4. *A V***-path** *is a sequence of simplexes*

$$(\sigma_0, \tau_0), (\sigma_1, \tau_1), ..., (\sigma_i, \tau_i), ..., (\sigma_n, \tau_n)$$

such that σ_i and σ_{i+1} are different faces of τ_i and (σ_i, τ_i) are paired simplexes.

The set of paired simplexes and critical simplexes of Σ forms a *Forman gradient V* if there are no closed V-paths in V or, in other words, if all the V-paths are acyclic.

In the combinatorial setup of discrete Morse theory, V-paths correspond to the gradient arrows of a Morse function f. We will call *separatrix V_i-path* the V-paths of the following form:

$$\tau, (\sigma_0, \tau_0), (\sigma_1, \tau_1), ..., (\sigma_i, \tau_i), ..., (\sigma_n, \tau_n), \sigma$$

where τ and σ are two critical simplexes of dimension i and $(i-1)$, respectively.

In Figure 1.17 the separatrix V-paths extracted from a Forman gradient V are illustrated. In Figure 1.17(a), the Forman function defined on the simplicial complex Σ is shown, while in

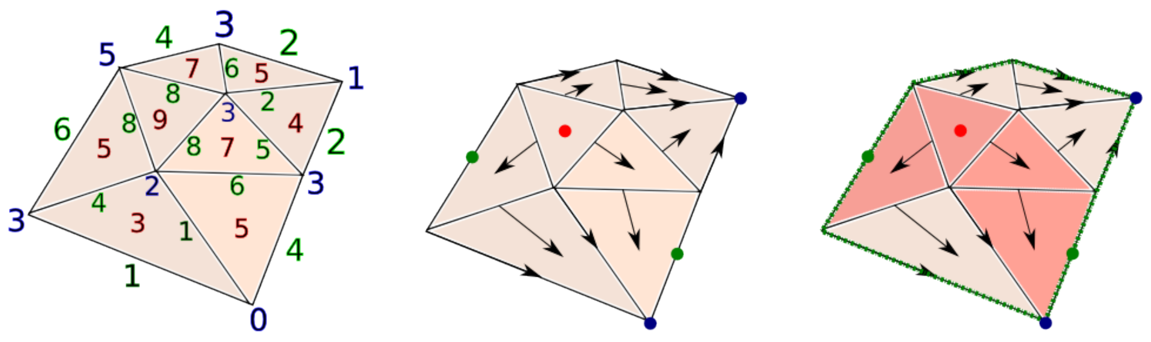


Figure 1.17: (a) The original simplicial complex Σ and the function defined on all the simplexes of Σ. (b) The corresponding Forman gradient computed on Σ. Red dots indicate critical triangles (maxima), green dots indicate critical edges (saddles) and blue dots correspond to critical vertices (minima).(c) Separatrix V_i-paths are highlighted: in green the separatrix V_1-paths connecting minima with saddles and in red separatrix V_2-paths connecting saddles to maxima.

Figure 1.17(b) the corresponding Forman gradient is shown. From the Forman gradient the separatrix V_1-paths are computed and highlighted in Figure 1.17(c). Green lines indicate separatrix V_2-paths connecting a saddle with a minimum, red triangles instead indicate separatrix V-paths connecting a maximum with a saddle.

The definitions introduced in discrete Morse theory are a discrete analogue of definitions for smooth functions we reviewed in Section 1.2. The V-paths forming a Forman gradient V can be seen as a combinatorial counterpart of the integral lines described in Morse theory as well as separatrix V-paths corresponds to separatrix lines. In a similar fashion, critical points correspond here to critical simplexes, involved in a number of outgoing or incoming V-paths based on their index. Observe that the index of a critical simplex is always equal to the dimension of the simplex.

1.5 Watershed Transform

Another way to define a Morse complex is through the *watershed transform* (see [RM00] for a survey). The watershed transform provides a decomposition of the domain M of a C^2 function f into regions of influence associated with the critical point of f. Such regions of influence are called *catchment basins*. The definition of *catchment basin* is based on a distance function. In this Section the notion of *watershed transform* is introduced based on the formalization proposed in [Mey94].

If f is at least C^1, it has a gradient $\bigtriangledown f$, except possibly at some isolated points.

Definition 1.5.1. *The* **topographic distance** $T_f(p, q)$ *between two points* p *and* q *belonging to the domain* M *of* f *is defined as,*

$$T_f(p, q) = inf_\gamma \int_\gamma || \bigtriangledown f(\gamma(s))||ds$$

where the infimum is considered over all paths γ *inside* M *originating at* p *and with destination* q.

The above definition of the topographic distance notion ensures that the path which minimizes the topographic distance between two points p and q in M is the path of steepest slope, if it exists. Thus, if p and q are two points in M:

- if $f(p) < f(q)$ and there is an integral line l which reaches both p and q, then $T_f(p, q) = f(q) - f(p)$,
- if $f(p) > f(q)$ and there is an integral line l which reaches both p and q, then $T_f(p, q) = f(p) - f(q)$.

Otherwise, if the integral line does not exist,

$$T_f(p, q) > |f(q) - f(p)|.$$

It follows from the above formulas that the topographic distance is unable to distinguish among points belonging to the same plateau since the topographic distance between two points p and q, belonging to the same flat region, is zero.

Definition 1.5.2. *Let* f *a function with minima* C_{min} *defined on a manifold domain* M. *The* **catchment basin** $CB(m_i)$ *of a minimum* m_i *is defined as the set of points* $x \in M$ *which are closer to* m_i*, in terms of topographic distance, than to any other minimum* m_j. *I.e.,*

$$CB(m_i) = \{x \in M | \forall m_j \in C_{min}, j \neq i : f(m_i) + T_f(x, m_i) < f(m_j) + T_f(x, m_j)\}$$

.

Definition 1.5.3. *The* **watershed** *of* f *is then defined as the set of points which do not belong to any catchment basin,*

$$Wshed(f) = D \cap (\cup_{m_i \in C_{min}} CB(m_i))$$

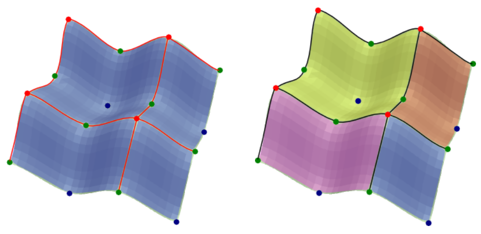


Figure 1.18: (a) The catchment basins corresponding to minima, 2-cells colored in blue, and the watershed lines, colored in red. (b) The corresponding ascending Morse complex Γ_a compared with the cell complex obtained with the watershed transform.

It follows from this definition that each catchment basin corresponds to a minimum while the watershed points to saddles and maxima. A similar definition can be stated for catchment basins related to saddle and maxima.

Definition 1.5.4. *Let f a Morse function with maxima C_{max} and saddles C_{sad} defined on a manifold domain M. The* **catchment basin** *$CB(max_i)$ of a maximum max_i is defined as max_i itself.*

$$CB(max_i) = \{max_i\}$$

.

Definition 1.5.5. *The* **catchment basin** *$CB(s_i)$ of a saddle s_i is defined as the set of points belonging to $Wshed(f)$ having their distance from s_i in terms of function value equal to the topographic distance value,*

$$CB(s_i) = \{x \in Wshed(f) | T_f(s_i, p) = f(p) - f(s_i)\}$$

.

For example, in the 1D case, the catchment basin relative to a minimum will be the line connecting the minimum to the maxima points and the watershed points will be the maxima.

In the 2D case, the catchment basins relative to minima are 2D surfaces, bounded by a sequence of watershed lines, related to saddles and maxima. The catchment basin for each saddle will be a ridge line bounded by two maxima; the catchment basin related to each maximum will be the maximum itself.

In Figure 1.18(a), an example of the watershed transform applied on a 2D manifold with boundary is shown. For each minimum (blue point), the corresponding catchment basin is a 2-cell (depicted in blue) while the watershed points are colored in red. Among such points, the catchment basins corresponding to the saddles (green points) are the 1-cells limited by two maxima (red points) and the catchment basin for each maximum is the point itself.

There is a one-to-one correspondence between catchment basins, watershed and the cells of the ascending Morse complex Γ_a defined on the same domain M by function a Morse f. For each i-saddle of f, the corresponding catchment basin and the $(n-i)$-cell of Γ_a cover the same domain. In the 2D case, each minimum catchment basin corresponds to a 2-cell in the ascending Morse complex Γ_a. For each saddle there is a catchment basin corresponding to the 1-cell of Γ_a and for each maximum its catchment basin is 0-cell of Γ_a.

Chapter 2

State of the Art

In this Section, we present a review of the state of art related to the analysis of scalar fields. Mainly, we concentrate on three topics: *computation* of Morse and Morse-Smale complexes, *simplification* of Morse and Morse-Smale complexes and *hierarchical representation* of Morse-Smale complexes.

Extracting morphological information from a scalar field, through the computation of a Morse or Morse-Smale complex, is the first issue we address. A variety of methods have been proposed in literature for the 2D and 3D cases. We will describe, in Section 2.1, the most widely used algorithms to compute Morse complexes and, with particular attention, we will focus on algorithms that computes Morse and Morse-Smale complexes on simplicial meshes. This section includes a review of methods we have defined and compared in [FIMS13]. At the beginning of Section 2.1, we will describe the state of the art for computing critical points on a combinatorial manifold, a common step for all the algorithms computing Morse complexes.

In many application domains, computing a Morse or Morse-Smale complex is insufficient to obtain a meaningful and significant structure useful for the analysis. This is due to the dimension of the original data, usually huge, and the noise naturally present in the data. Thus, a simplification process is required in order to clean up the data and to reduce drastically their dimensions. In Section 2.2 we will present an overview of the simplification algorithms, based on i-$cancellation$ operator described in Section 1.2.1, for the simplification of Morse and Morse-Smale complexes.

In the last part of this section we review hierarchical models that have been defined in literature for the interactive analysis of scalar fields. A hierarchical model can be described as a particular encoding, for a scalar field $\mathbb{M}$, from which is possible to extract representations of $\mathbb{M}$ at different level of details. It is usually built from a simplification process. The different representations are thus extracted from this model refining the interesting part of the scalar field. We will describe these models in Section 2.3.

2.1 Computing Morse and Morse-Smale complexes

In this section, we describe different techniques proposed in literature to compute Morse or Morse-Smale complexes. Many algorithms have been proposed to compute a Morse complex in 2D and 3D, we refer to [BFF$^+$08] and [CDFI13] for a complete overview. All the algorithms can be classified based on different criteria. A first distinction can be done based on the dimension on which the algorithms work; some of them are *dimension specific*, i.e. they are defined for working on specific dimensions (usually 2D and 3D), while others are *dimension-independent*.

Another classification can be done based on the complexes they compute. Some algorithms, i.e., compute the ascending and descending Morse complexes extracting the top cells of such complexes. Other algorithms compute the top cells of the Morse-Smale complex, or its 1-skeleton thus extracting all the 1-cells connecting pairs of critical points. Algorithms based on discrete Morse theory to compute a Forman gradient at first and then use the gradient to extract Morse or Morse-Smale complexes.

The combinatorial manifold, used to discretize the domain, offers a characterization for these algorithms. Many algorithms rooted in image processing have been defined on regular square grids and cubical complexes. Algorithms for terrain and 3D shape analysis instead are generally described on triangle and tetrahedral meshes.

The algorithms described in this section are presented according to the approach they use to compute the Morse or Morse-Smale complexes. Note that different approaches require the computation of different critical points as an initial step. *Region-growing* algorithms [DDFM03, MDDF$^+$08, GNPH07] compute only the minima and the maxima of the scalar field without extracting explicitly saddle points (see Section 2.1.2). *Boundary-based* algorithms [EHZ01, EHNP03, TIS$^+$95, Vit10] first extract all the critical points of f, and computing as a second step the ascending and descending 1-cells associated with saddles (see Section 2.1.3). The *watershed algorithms* [Mey94, VS91, Soi03, MW99, SS00] base their computation only on minima and extract the top cells of the ascending Morse complex (see Section 2.1.4). Clearly, they can be used to compute the descending Morse complex by inverting function f. Algorithms rooted in *discrete Morse theory* [KKM05, GBHP11, RWS11, GRWH12, SN12, SMN12] compute a Forman gradient V directly from a discrete scalar field $\mathbb{M}_\Gamma = (\Gamma, f)$ extending function f to all the cells of Γ (see Section 2.1.5).

2.1.1 Computing critical points

As described in [BFF$^+$08], most of the works known in literature refer to [Ban67, Ban70] for computing critical points on a surface in the discrete case [TIS$^+$95, EHZ01, NGH04]. The classification of a critical point can be done based on the function value at a vertex p, $f(p)$, and the simplexes in the link of p, $Lk(p)$ (or alternatively the simplexes in the star of p). The link of p is decomposed into two sets:

- the *upper link*, denoted $Lk^+(p)$, consists of the simplexes composed by vertices $v \in Lk(p)$ such that $f(v) > f(p)$,
- the *lower link*, denoted $Lk^-(p)$, consists of the simplexes composed by vertices $v \in Lk(p)$ such that $f(v) < f(p)$,

Based on the cardinality of these two sets, a point p can be classified as critical [EHZ01],

- if $Lk^+(p)$ is empty, then p is a minimum,
- if $Lk^-(p)$ is empty, then p is a maximum.

Otherwise, the number of connected components in $Lk^-(p)$ and $Lk^+(p)$, denoted $\#(Lk^-(p))$ and $\#(Lk^+(p))$ respectively, defines the index of p. In the 2D case [EHZ01]:

- if $\#(Lk^-(p)) = 1$ then p is regular,
- if $\#(Lk^-(p)) = n_p$, with $n_p > 1$, then p is a $\frac{n_p}{2}$-saddle.

In the 3D case [EHNP03]:

- if $\#(Lk^-(p)) = 1$ and $\#(Lk^-(p)) = 1$ then p is a regular point,
- if $\#(Lk^-(p)) = 1$ and $\#(Lk^-(p)) = 2$ then p is a 1-saddle,
- if $\#(Lk^-(p)) = 2$ and $\#(Lk^-(p)) = 1$ then p is a 2-saddle,
- if $\#(Lk^-(p)) + \#(Lk^-(p)) > 3$ then p is a multi-saddle.

Methods have been proposed to unfold multi-saddles in simple saddles in 2D [EHZ01] and in 3D [EHNP03]. In both cases, to split the saddle, the underlying geometrical structure is modified.

However, in the applications, much more problematic cases can occur. The problem of degenerate critical points, as flat areas naturally present in real data, can be handled either by replacing the notion of critical point with the notion of *critical area* [BFS02, CKF03] or by perturbing and unfolding the discrete surface (i.e. the simplicial complex) [EM90, AE98, EHNP03]. In [MFI13] we have proposed a preprocessing method to eliminate flat edges from a Triangulated

Irregular Network (TIN), perturbing their elevation, in a morphologically consistent way and without introducing new critical points.

The geometric characterization of a critical point has been also used to define critical points on a regular model. Adapting the characterization means, in this case, to adapt the concept of neighborhood to the connectivity of the regular grid. Moreover, in these cases the field is often viewed as a function defined on the n-cells of the grid instead of on the vertices [BPS98, WSHH02, Pap04]. Regular grids can also exploit the so called *analytic approaches* to compute critical points. An analytic approach relies on the general idea of fitting an approximating function on the vertices of the grid and then critical points are detected through numerical techniques [WLH85, SW04, WS04]. Such techniques are, in general, more precise and computationally more expensive than the combinatorial counterpart.

Approaches rooted in discrete Morse theory (see Section 1.4), compute critical cells of a cell complex Γ on which a Forman function has been defined, by computing a Forman gradient on Γ first.

2.1.2 Region-growing algorithms

Region-growing algorithms mimic in the discrete case the definition of an n-cell of a Morse complex. More precisely they expand, starting from a critical point p, the associated Morse n-cell following the integral lines originating or ending at p.

In [DDFM03], an algorithm for computing the descending and ascending Morse complexes has been presented for the segmentation of terrain models. The algorithm performs a breadth-first traversal of the dual graph of the triangle mesh in which the nodes correspond to triangles of the mesh and the arcs correspond to the edges shared by edge-adjacent triangles. A 2-cell is created for each minimum/maximum critical point and then expanded one triangle at time. In [DDFM03], the expansion criterion of a 2-cell is based on the height function value of the triangle vertex not yet included in the boundary of the 2-cell. In [MDDF$^+$08] such criterion has been improved to avoid over-segmentations. In [CMF11] the discrete gradient vector field, associated with the decomposition obtained with the algorithm described in [DDFM03], has been showed as a sub-field of the gradient field of a Forman function whose restriction over the vertices of the triangle mesh coincides with the given scalar field function f.

The algorithm proposed in [GNPH07], computes the Morse-Smale complexes of a function f defined over the vertices of a tetrahedral mesh Σ. Starting from the ascending Morse complex the 3-cells are computed labeling the vertices of the tetrahedral mesh Σ as belonging to the interior or the boundary of some ascending 3-cell. The region growing approach is applied recursively in order of decreasing cell dimension computing all the remaining ascending cells. Descending cells are computed inside the ascending 3-cells using the same region growing approach.

In the latter we describe two algorithms in details [MDDF$^+$08, GNPH07]. The first algorithm [MDDF$^+$08] works on triangle meshes and represents the most recent development, of algorithms of the same kind, known in literature. The second [GNPH07] is an example of region-growing algorithm defined specifically for the 3D case, even if it could be defined in a dimension-independent way.

2.1.2.1 STD algorithm

The *S*ource *T*hrough *D*rain algorithm (STD), proposed in [MDDF$^+$08], has been developed to extract the 2-cells of the ascending and descending Morse complexes from a discrete scalar field $\mathbb{M}_\Sigma = (\Sigma, f)$ where f is a scalar function defined on the vertices of the triangle mesh Σ.

The STD algorithm perform the three main steps described below:

(i) Classification - the vertices of each triangle t in Σ are classified based on their function value.

(ii) Extraction - the minima/maxima of f are extracted

(iii) Growth - for each minimum/maximum, the corresponding 2-cell is constructed.

We described here the computation of the ascending Morse complex, the computation of the descending Morse complex is entirely dual.

In the *classification* phase, all the vertices are labeled for each triangle. The objective is to simulate the gradient flow inside each triangle t. Then, by considering the three vertices of t in descending order of function value, the highest, middle and lowest vertices are labeled as *Source* (S), *Through* (T) and *Drain* (D), respectively. This labeling provides an easy way to *extract* the minima of f by considering only the vertices labeled as *Drain* in all their incident triangles.

For each minimum p a new 2-cell γ_p is created, formed by all the triangles incident in γ_p. Then, the region γ_p is grown iteratively by including the triangles adjacent to its boundary. The rationale for the inclusion of a triangle t in γ_p follows two main ideas:

- water flows from a higher to a lower elevation;
- triangle t can be included in only one region and the choice must be deterministic.

The algorithm maintains an invariant. If a triangle t has been included into a 2-cell γ_p, then the edge of t labeled $T - D$ is not on the boundary of γ_p.

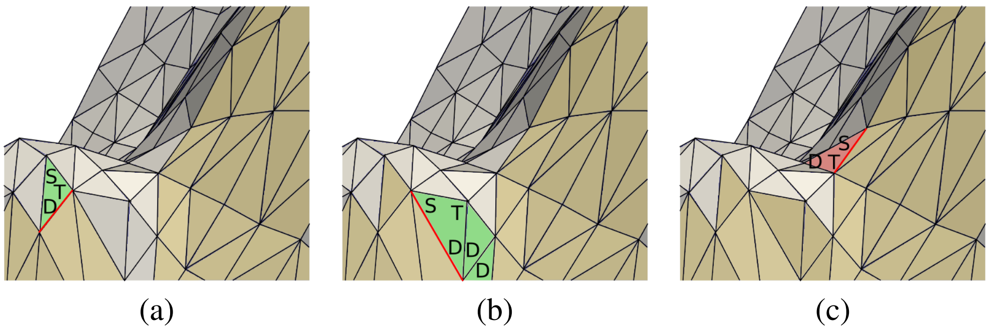


Figure 2.1: (a) The rule used for the inclusion of a boundary triangle if the opposite vertex is labeled S, in (b) and (c) the rules applied when the vertex is labeled T or S, respectively. Yellow triangles are included in the basin, gray triangles are excluded. Green triangles are the candidates to be included by the rule while red triangles will be excluded.

The rule that controls when a new triangle t is added to a cell γ_p consists of three cases which are based on the edge e incident in t and on the boundary of γ_p:

1 if the vertex v of t, opposite to e, is labeled D in t, then t is not included into γ_p;

2 if the vertex v of t opposite to e is labeled S in t, then t is included into γ_p;

3 if the vertex v of t opposite to e is labeled T in t, then surely water flows to vertex v', endpoint of e, which is labeled D in t. Starting from t, the maximal set of edge-adjacent triangles having their lowest vertex in v' is collected. Let w be the vertex of maximum elevation among the vertices of such triangles. The fan starting from t until the first triangle incident in edge (v', w) is included into γ_p.

In Figure 2.1 the expansion of a 2-cell is shown. Yellow triangles are already included in the 2-cell while gray triangles are the ones which are not yet included. In Figure 2.1(a), the triangle adjacent to the boundary edge (in red) is included since the vertex opposite to the edge is labeled D. In Figure 2.1(b) a fan of triangles is included starting from the triangle incident in the red edge, since the opposite vertex was labeled T. In Figure 2.1(c) an example of a triangle not included in the 2-cell, since the vertex opposite to the boundary edge is labeled D, is shown. In [MDDF$^+$08] a rigorous handling of the special cases, due to the flat triangles present in the data, can be found.

In the STD algorithm, every triangle of Σ is examined at most three times, one from each edge, before being included in one cell. A different situation occurs when a triangle is part of an edge-adjacent set of triangles (case 3). In such cases, having fixed edge (v', w), a triangle will be included with the set of triangles on the left or on the right of such edge. Thus, the worst-time complexity of the algorithm is always $O(|\Sigma_0|)$, where $|\Sigma_0|$ denote the number of vertices in Σ.

A dimension-independent definition for this algorithm would be hard to be stated. The labeling of the vertices could be extended for n-simplexes. However, the third rule requires an ordering of the top simplexes around a vertex, ordering that could be not uniquely determined in dimensions higher than two.

2.1.2.2 A Region-growing approach for computing 3D Morse-Smale complexes

The algorithm proposed in [GNPH07] computes the Morse-Smale complex of a function f defined over the vertices of a tetrahedral mesh Σ. The ascending cells are computed through a region-growing approach in the order of decreasing cell dimension. Descending cells are computed inside the ascending 3-cells, using the same region-growing approach. The ascending and descending cells are simply collections of vertices of Σ.

The computation of the ascending 3-cells consists of two steps:

- the set of minima of f are identified looking at the lower link of each vertex (see Section 2.1.1 for details). Each minimum will be the origin for a set of vertices representing an ascending 3-cell;
- each vertex p of Σ is classified as an *internal* or *boundary* vertex of an ascending cell. This depends on the number of connected components in the set of vertices in the lower link of p which are already classified as interior to some ascending 3-cell.

The classification is performed by sweeping Σ in the order of ascending function values.

Vertices classified as *boundary* in the first step of the algorithm are the input for the algorithm which builds the ascending 2-cells. An ascending 2-cell is created for each pair of adjacent 3-cells. The vertices of the 2-cells are classified as *interior* or *boundary*, based on local neighborhood information, similarly to the classification with respect to the 3-cells. A 1-cell is created everywhere ascending 2-cells meet. Each 1-cell is composed of vertices classified as *boundary* in the previous step. Finally, each vertex p of an ascending 1-cell is classified as interior or boundary. Maxima are created at the boundaries between ascending 1-cells. They form a small disjoint clusters of vertices.

An example of the extraction performed by the algorithm, is shown in Figure 2.2. In Figure 2.2(a) the set of internal vertices is computed and each of them forms the set of internal vertices of an ascending 3-cell. The set of boundary vertices are shown in black; they will form the ascending 2-cells. In the intersection points of such 2-cells the ascending 1-cells are computed (as

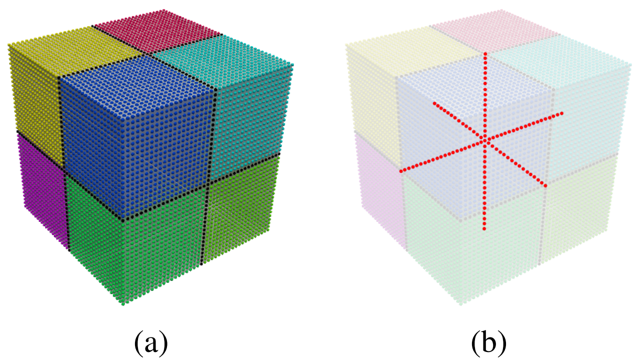


Figure 2.2: (a) The ascending 3-cells extracted on a synthetic data set with the algorithm presented in [GNPH07]. (b) The ascending 1-cells extracted.

illustrated in Figure 2.2(b)).

The descending cells are computed in their interior of the ascending n-cells. The region-growing steps are the same and again here iterations are performed in the order of decreasing dimension. The growth of the descending cells are forced to be in the interior of the ascending 3-cells. This corresponds to intersect the ascending and descending Morse complexes directly, thus obtaining the Morse-Smale complex .

Vertices are processed in increasing/decreasing order when the ascending/descending Morse cells are computed and each vertex is processed only once, thus leading to $O(|\Sigma_0|)$ complexity of the whole algorithm. Thus, the worst-time complexity is $O(|\Sigma_0|\, log\, |\Sigma_0|)$, because of the sorting of the vertices.

The algorithm has been used for tetrahedral meshes. However, the computation is vertex-based and the simplicial structure is only used to build the link of each vertex. Thus, the algorithm actually works on the 1-skeleton of the simplicial mesh and a dimension independent version could be defined.

2.1.3 Boundary-based algorithms

The algorithms described in this section compute the boundary of the cells of the Morse-Smale complexes and, thus, they are called *boundary-based* [TIS$^+$95, EHZ01]. A *boundary-based* algorithm generally extracts the critical points first and then traces the separatrix lines that are the 1-cells of the Morse-Smale complex. For each saddle q, with multiplicity k, k ascending paths and k descending paths are computed.

The steps performed by the algorithm are divided into:

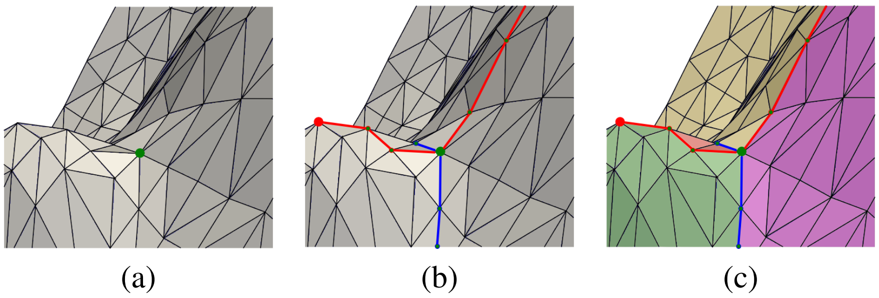


Figure 2.3: (a) Green point is the saddle point from which expand the Morse 1-cells. (b) The ascending and descending 1-cells are expanded, indicated with blue and red lines respectively. (c) The MS 2-cells delimited by the ascending and descending 1-cells are indicated with different colors.

- *classification* - the critical points are identified;
- *computation of upper/lower sequence* - for each saddle q the sequence of adjacent vertices with higher/lower elevation are computed. They are called upper and lower sequence respectively. For each sequence the highest/lowest vertex v is identified and, starting from v, the highest/lowest adjacent vertex v' is iteratively identified and included in the sequence connected with q until a maximum/minimum is reached.

During the classification step, all the critical points are computed. The extraction of critical points is usually performed based on techniques implementing the classification by Banchoff [Ban67]. The computation of the sequence simulates the reconstruction of the separatrix lines connecting pairs of critical points. Thus, in this step, a navigation of the mesh is performed on the vertices of the mesh following the maximum increasing/decreasing slope.

In Figure 2.3, the extraction of the 1-cells of a Morse-Smale complex is shown. In Figure 2.3(a), the first saddle is selected. In Figure 2.3(b) for each connected component in the lower link an ascending 1-cell is created (blue lines) following the steepest descent. For each connected component in the higher link a descending 1-cell is created (red lines) following the steepest slope. In Figure 2.3(c), 2-cells of the Morse-Smale complex are bounded by the ascending and descending 1-cells.

All boundary-based algorithms in the literature have these steps in common, to compute the 1-cells of the Morse-Smale complex. They differ in the way they handle special cases [EHZ01, EHNP03, Vit10, TIS$^+$95]. The authors of [TIS$^+$95] introduce a virtual pit (vertex with minimum field value) in order to handle the boundary of a triangle mesh. This method makes the mesh homeomorphic to a ball, and the critical point theorems is satisfied. Moreover minima on the boundary are impossible. If we consider a two-dimensional scalar field $\mathbb{M}_\Sigma = (\Sigma, f)$ and its dual with opposite field values, they will not have the same saddles on the boundary.

In [EHZ01] the approach presented by [TIS$^+$95] is extended. Function f is required to be a piecewise-linear Morse function, sequences are still computed by starting from the saddle points, but at each step the point with maximum slope is detected. The Smale condition is simulated a-posteriori by extending the path beyond each saddle, and forcing the path of the 1-skeleton in the Morse-Smale complex, not to intersect.

In [EHZ01, EHNP03] the notion of *Quasi Morse-Smale* is introduced (see Section 1.3). A *Quasi Morse-Smale complex* (QMS) is a complex that reflects the combinatorial structure of the Morse-Smale complex, but in which the arcs and quadrangles (1- and 2-cells) may not be those of maximal ascent and descent.

In the following we describe in detail only the algorithm presented in [EHNP03] being the only boundary-based method proposed in literature for computing Morse-Smale complexes for tetrahedral meshes. For a complete overview, we address to [BFF$^+$08] and [Vit10].

2.1.3.1 Computing Quasi Morse-Smale complexes in 3D

The boundary-based algorithm in [EHNP03] extracts the Quasi Morse-Smale complex for a tetrahedral mesh of a three-dimensional scalar field by computing first the critical points through the reduced Betti numbers of the lower link of the vertices (see Section 2.1.1).

Then, the three-dimensional quasi Morse-Smale complex is computed during two sweeps over the tetrahedral mesh Σ. The first sweep (in the direction of decreasing function value) computes the descending 1- and 2-cells and the second sweep (in the direction of increasing function value) the ascending 1- and 2-cells of the Morse complexes. The algorithm is boundary-based, as it computes the 1- and 2-cells which bound the 3-cells in the Morse complexes.

During the first sweep, the descending 1-cells and 2-cells are computed simultaneously. A 1-cell is built as follows:

- If a current vertex p in the sweep is a 1-saddle, a descending 1-cell is started. The two arcs of the corresponding 1-cell are initialized by edges from p to the lowest vertex in each connected component of the lower link of p, as illustrated in Figure 2.4(a).
- If there is a descending arc ending at a current vertex p, it is expanded by adding an edge from p to the lowest vertex in its lower link. If p is a 1-saddle, later an ascending 2-cell will start at p and each descending arc is extended to the lowest vertex in the specific connected component of the lower link of p that is not separated from the arc by the ascending 2-cell.
- If p is a minimum, it becomes a node of the quasi Morse-Smale complex, and the descending arcs end at p.

A 2-cell is built as follows:

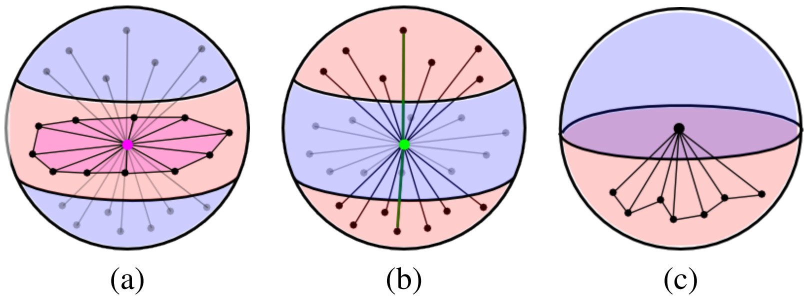


Figure 2.4: (a) 1-saddle, (b) 2-saddle and (c) a regular point found during the algorithm described in [EHZ01].

- If a current vertex p in the sweep is a 2-saddle, a descending 2-cell is started. A cycle of edges in the lower link is constructed, which contains the lowest vertex in the lower link of p. Triangles determined by p and edges of the cycle form the initial descending 2-cell of p, as illustrated in Figure 2.4(b). Initially, the entire boundary of the descending 2-cell is unfrozen.
- A 2-cell is expanded by constructing a shortest-path tree in the lower link of the current (highest) vertex q on the unfrozen boundary of the 2-cell associated with 2-saddle p, and connecting q to the edges of this tree. If q is a critical point (a 1-saddle or a minimum), it is declared frozen together with its two incident edges on the boundary.
- When the complete boundary of a 2-cell is frozen the 2-cell is completed.

The next step consists of building the intersections between the descending and ascending 2-cells by tracing ascending paths inside a descending 2-cell, starting from 1-saddles on the boundary of the descending 2-cell and ending at the 2-saddle that started the descending 2-cell. These intersections are used to guarantee the structural correctness of the extracted quasi Morse-Smale complex. Each 2-saddle starts two arcs of an ascending 1-cell, which must not cross any already established descending 2-cell. The intersection curves between descending and ascending 2-cell, and the ascending 1-cells decompose each ascending 2-cell into quadrangles. The ascending cells are built one quadrangle at a time, similarly to descending 2-cells.

An implementation of this algorithm is described also in [NP05]. The worst-case time complexity of the algorithm is bounded from above by the time for sorting the vertices, the input size for constructing the vertex link plus the output size for describing the resulting Morse-Smale complex. Sorting the $|\Sigma_0|$ vertices of the input complex takes $O(|\Sigma_0|\, log\, |\Sigma_0|)$, the input size is $O(|\Sigma_0|^2)$ while the size of the output depends on the cells describing the quasi Morse-Smale complex [EHNP03].

The extension of this algorithm to higher dimensions directly depends on the possibility to compute the critical points. Since a characterization for an i-saddle in dimensions higher then three is impossible, the first step of the algorithm is not defined. This is a common limitation for all the boundary-based approaches that are generally not well suited for the computation of Morse-Smale complexes in dimensions higher then two, also for their computational costs.

2.1.4 Watershed algorithms

The watershed transform, described in Section 1.5, provides a decomposition of the domain of a smooth function f into regions of influence associated to the critical points of f, called *catchment basins*. If the function f is smooth, the union of all the catchment basins related to the critical points forms the ascending Morse complex. Through a change in the sign of function f, the descending Morse complex can be defined in the same way.

As described in [BFF+08], the watershed transform has been applied, in the discrete case, in image processing for segmenting gray-scale images. Several algorithms have been developed to compute the discrete watershed transform [RM00, NC03]. Such algorithms can be divided in three groups:

(i) approaches based on *topographic distance* [Mey94]

(ii) approaches based on *simulated immersion* [VS91, Soi03]

(iii) approaches based on *rain falling simulation* [MW99, SS00]

The definition of a catchment basin, watershed line and watershed transform is similar in the discrete case to the definition in the continuous case. The main difference is that the continuous topographic distance is replaced with a (cost-based) discrete topographic distance. The *discrete topographic distance* between two points is then defined as the minimum-cost path joining them. The cost of a path is just the sum of the costs of the edges forming it. By selecting as a topographic distance a minimum-cost path, the catchment basin of a minimum p becomes the set of vertices in the simplicial model which are closers to p, in terms of the discrete topographic distance, than to any other minimum in the model. The watershed lines are the complement of the collection of the catchment basins of the minima, when the complement is taken on the set of vertices of the simplicial model. All the algorithms inspired by [Mey94] implement a modification of a classical shortest path algorithm in order to grow the ascending cell associated to each minimum. The worst-time complexity of the algorithm presented in [Mey94] is $O(|\Sigma_0|\ log|\Sigma_0|)$ due to the initial sorting of the vertices according to their elevation. However an alternative definition can be state as a graph traversal working in linear time.

Since originally applied in image processing, watershed algorithms only label the vertices of the mesh. They actually work on a graph structure connecting all the vertices of the discrete domain. In image processing, the domain is usually discretized through a regular grid where the function values are in correspondence of the top cells of the grid (called *pixels* in 2D and *voxels* in 3D). The graph structure is here the dual graph of the grid structure where the nodes corresponds to the pixels (or voxels) of the image; an arc of the graph corresponds to the adjacency between two pixels/voxels. There are different kinds of adjacency dependently on whether we consider in 2D only adjacency along edges or also adjacency between vertices. When the underlying structure, used to discretize the domain, is a triangle or tetrahedral mesh Σ, the function values are assigned to the vertices of the mesh; the graph structure corresponds here to the 1-skeleton of Σ where the vertices of Σ define the set of nodes and edges of Σ define the set of arcs of the graph. Thus, in order to segment the top simplexes of a simplicial mesh, the labeling computed on the vertices has to be propagated to the top simplexes according to some criterion. All the watershed algorithms compute the ascending and descending top-cells of the Morse complexes and then the Morse-Smale complex can be obtained by intersection of these cells. However, the extracted cells are not forced to respect the Smale condition. Since their origin, all the watershed algorithms provide a labeling of the vertices of a simplicial mesh Σ. We have defined a criterion for propagating this labeling to the top simplexes of Σ.

Note that the watershed algorithms discussed here are entirely discrete and differ from the approaches developed in the computational geometry literature for piecewise-linear triangulated terrains, in which the terrains are considered as continuous and several steepest descent paths can cross a single triangle [YVKS96, McA99, BCH$^+$07].

In the following, we will describe in more detail two definitions for the watershed transform different from the discrete topographic distance, the watershed by *simulated immersion* [VS91] and the watershed by *rain falling simulation* [MW99]. These two algorithms have given good results when applied in two dimensions for terrains [Vit10] and thus they have been selected in our work for the extension to higher dimensions.

2.1.4.1 Simulated immersion

The watershed algorithm by simulated immersion has been introduced in [VS91] for segmenting a 2D image into regions of influence of minima, which correspond to ascending 2-cells. The idea of simulated immersion can be described in an intuitive way. Let us consider the graph of a two-dimensional scalar field $\mathbb{M}$ and assume that holes are drilled in place of local minima. We immerse this surface in a pool of water, building dams to prevent water coming from different minima from merging. Then, the watershed of f is described by these dams and the catchment basins of the minima are delimited by the dams.

In [CFI10] we have introduced an extension of this algorithm to scalar fields defined at the vertices of a simplicial mesh in arbitrary dimension. The vertices of the simplicial mesh Σ are sorted in increasing order with respect to the values of the scalar field f, and are processed level by level in increasing order of function values. For each minimum p, an ascending cell C_p is iteratively constructed through a breadth-first traversal of the 1-skeleton of the simplicial mesh

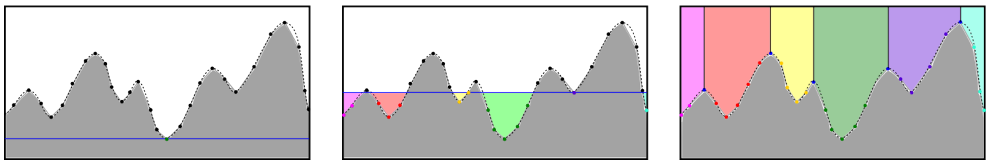

(a) (b) (c)

Figure 2.5: Immersion of a 1D terrain following the algorithm described in [VS91].

Σ (formed by its vertices and edges). For each vertex v, the labeling of v is based on the labels assigned to its adjacent vertices:

- if they all belong to the same ascending region C_p, or some of them are watershed points, then v is marked as belonging to C_p,
- if they belong to two or more ascending regions, then v is marked as watershed point,
- if they are all unmarked, then v is a minimum and a new ascending cell C_v is created.

In Figure 2.5 an example of the algorithm applied on a terrain is shown. In Figure 2.5(a), the blue line represents the level of immersion of the terrain. It starts from the lowest vertex recognized as a minimum and a new region (depicted in green) is initialized. In Figure 2.5(b), the line of immersion is raised and new minima are found (one minimum for each colored region) while the vertices inside each region are colored accordingly. In Figure 2.5(d), the line reaches the top. Points which are on the boundary of two regions are indicated as watershed (blue points). As discussed in Section 1.5, in 1D a watershed corresponds to a maximum.

We have defined a criterion for propagate the labeling on vertices to the maximal simplexes. Each maximal simplex σ (an n-simplex if we consider an n-dimensional simplicial mesh) is assigned to an ascending region based on the labels of its vertices. If all vertices of σ, that are not watershed points, belong to the same cell C_p, then σ is assigned to C_p. If the vertices belong to different ascending cells C_{p_i}, then σ is assigned to the cell corresponding to the lowest minimum. The assignment of top simplexes inside an ascending cell based on the labeling of its vertices is trivial. The challenging part is assign a top simplex on the boundary of two or more ascending cells. In some cases the creation of smooth boundaries for the cells can be preferred. The method proposed instead assigns a simplex to the ascending cell corresponding to the deepest minimum, considering such minimum as more influential. Descending regions associated with maxima are computed in a completely similar fashion.

The worst-time complexity of the labeling phase is $O(|\Sigma_0|\, log|\Sigma_0|)$ due to the initial sorting of the vertices. Also for the simulated immersion approach, this computation could be performed in linear time. Once the vertices are all labeled the labeling is propagate to the top simplexes. Each top simplex is visited only once for a time complexity of $O(|\Sigma_n|)$.

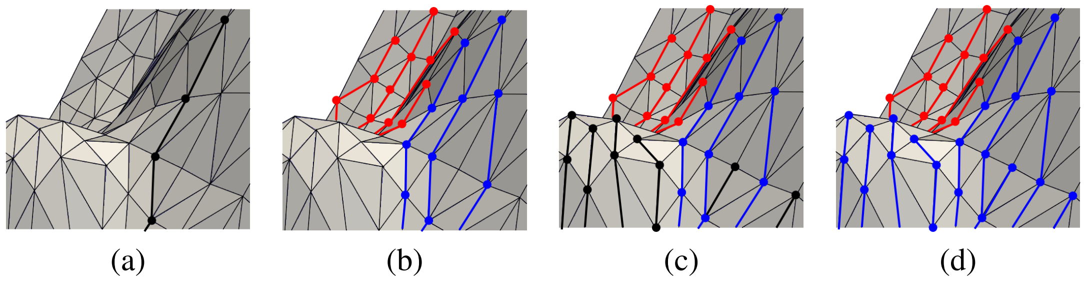


Figure 2.6: Labeling of the vertices of a terrain following the algorithm described in [MW99].

2.1.4.2 Rain falling simulation

The rain falling simulation approach presented in [MW99] uses a reverse strategy respect to the simulated immersion approach.

The general idea is to construct steepest descending paths from each vertex until a minimum, or a vertex already inserted in an ascending cell, is reached. The main steps of this algorithm are:

- *minima computation* - minima are identified and for each of them an ascending cell is created,
- *sorting of the vertices* - all the vertices are sorted in descending order of function value,
- *assignment* - each vertex is assigned, recursively, to the cell of its lowest adjacent vertex.

In Figure 2.6, the labeling propagation used by the algorithm is illustrated. In Figure 2.6(a), a rain falling line is propagated from a maximum to the basin following the steepest descent. In Figures 2.6(b) and 2.6(c), more lines are added with different labels (blue or red) based on the reached basin.

After the assignment step, all the vertices are labeled as belonging to some ascending region. No watershed lines are created with this approach. Thus, each maximal simplex σ can be labeled following the same approach used by the simulated immersion algorithm. The possible problems arising from the rain falling approach are the non-uniqueness of the lowest neighbor q of a vertex p and the occurrence of flat areas. Possible solutions are discussed in [SS00]. The first problem can be solved postponing the decision on the steepest path examining all the paths with equal steepest descent until a difference is found. Problem of flat areas are generally solved considering a flat region as a single vertex, thus labeling all the vertices inside in the same way [MW99] or splitting the flat areas among different ascending cells [SS00]. The worst-time complexity of the algorithm is the same as the already discussed watershed approaches. We have developed an implementation of this algorithm for triangle and tetrahedral meshes, even if the approach is dimension independent, extending the labeling, from vertices to maximal simplexes, as discussed in Section 2.1.4.1.

2.1.5 Algorithms rooted in discrete Morse theory

The algorithms proposed in the literature based on discrete Morse theory do not extract explicitly cells from the Morse or Morse-Smale, complexes but compute a Forman gradient V extending the scalar function f, defined on the vertices of a CW complex Γ, to all the cells of Γ. Generally such algorithms are dimension independent by definition.

Generally speaking, a scalar function defined on the vertices of a CW complex Γ is not required in order to compute a discrete Morse complex on Γ [HMMN14, HMM$^+$10]. This is the case where a discrete Morse complex is used for homology or persistent homology computation. Since we are interested in the analysis of a scalar field $\mathbb{M}$, we will focus on methods for computing a Forman gradient on the underlying geometry of $\mathbb{M}$ [KKM05, GBHP11, RWS11, GP12, GRWH12, SMN12, SN12].

Since discrete Morse theory has been defined for CW complexes (see Section 1.4) most of the algorithms proposed are defined for this kind of complexes. However, they are easily adaptable to other kind of complexes like cubical complexes or simplicial meshes.

We can classify the algorithms proposed in literature in two groups:

- algorithms focused on the Forman gradient computation [KKM05, GBHP11, RWS11, SMN12]
- algorithms for the efficient extraction of Morse and Morse-Smale complexes from a Forman gradient [GP12, GRWH12, SN12, WIFF13] .

Algorithms of the first group are focused in constructing a Forman gradient on a regular CW complex. One of the first algorithm proposed is described in [KKM05]. The algorithm splits the computation of a Forman gradient on the link of the vertices of a tetrahedral mesh. The Forman gradient is computed recursively on the link of such vertices and the pairing obtained is propagated backward to obtain a global result. We will describe this algorithm in detail in Section 2.1.5.1.

In the algorithm proposed in [GBHP11], a Morse-Smale complex is computed starting from a regular n-dimensional CW-complex Γ with scalar field f defined at the vertices of Γ. Function f is extended to a Forman function F, defined on all cells of Γ, such that $F(\sigma)$ is slightly larger than $F(\tau)$ for each cell σ and each face τ of σ. Based on the new Forman function F a pairing for the cells of Γ is computed and the Morse-Smale complex extracted. The same algorithm has been successively optimized for the parallel computation of a Forman gradient on a two-dimensional CW-complex in [SMN12] and for a three-dimensional CW-complex in [SN12]. A description of the algorithm is presented in Section 2.1.5.2.

One of the most widely used algorithm to compute a Forman gradient on a n-dimensional CW-complex has been presented in [RWS11]. The algorithm is presented for 3D cubical complexes for study to study the homology of a 3D gray-scale digital image. The strategy used to build in an efficient way a discrete Morse function on the whole complex is to partition it into small groups of cells, based on the vertices, and analyze each group separately. The lower stars of all vertices are then treated independently for computing a pairing of cells. In our work, we have developed an implementation of this algorithm for triangular and tetrahedral meshes [WIFF13] working also in parallel. We will describe such algorithm in detail in Section 3.4.

The challenging problem, when working with a Forman gradient V, is the traversal of the V-paths of V. In 2D, all the V-paths can be visited in linear time visiting all the cell pairs at most once. However, extracting only the separatrix V-paths the gradient paths can be visited in reverse order reducing the numbers of pairs visited. In higher dimensions the situation is more complicate. In 3D for example, gradient paths can branch and merge multiple times resulting in a many-to-many adjacency relationship between critical 1- and 2-cells. This produces a discrete Morse complex containing $O(|\Gamma_0|^2)$ gradient paths between critical 1- and 2-cells and, since the number of critical 1- and 2-cells is bounded by $O(|\Gamma_0|)$ this lead the complexity to $O(|\Gamma_0|^3)$. To solve this problem some modifications to the extraction algorithm has been proposed. In [WIFF13, GRWH12] the worst-time complexity is reduced to $O(|\Gamma_2|)$, where $|\Gamma_2|$ denotes the number of 2-cells in Γ, forcing a single path not to visit the same simplex more than once. Simplexes are marked, as visited or not visited by a V-path, and this leads to an increasing in the storage. Overcoming this rise in the storage cost is crucial for a parallel implementation. The idea described in [SN12] is to interpret the substructure of the gradient paths as a Directed Acyclic Graph (DAG). Then, avoiding the standard breadth first traversal of the DAG, the path exiting from a DAG node is visited only when all the entering paths have been visited. This way the common paths are visited only once and only the initial simplexes, where the common path starts, are visited more than once. This leads to a worst time complexity of $O(|\Gamma_2|\, log|\Gamma_2|)$ without increasing the storage cost. We will described in details the path traversal algorithm, based on our encoding defined for triangle and tetrahedral meshes, in Section 3.3.1.

2.1.5.1 Recursive construction of a Forman gradient

The algorithm proposed in [KKM05] takes as input a scalar field f defined over the vertices of a 3D simplicial complex Σ and a persistence value $\rho \geq 0$ (see Section 2). It computes a Forman gradient V by subdividing the simplexes of Σ into three lists, denoted as A, B and C, such that:

- lists A and B have the same length,
- for each i-simplex $\sigma_j \in A$, $V(\sigma_j) = \tau_j$, where τ_j is the $(i+1)$-simplex in B,
- C is the set of critical simplexes.

The algorithm builds the Forman gradient V recursively on the *lower link* $Lk^-(v)$ of each vertex v in Σ (see Section 2.1.1). Lists A, B and C are initialized as empty. For each vertex v in Σ, if

$Lk^{-}(v)$ is empty, then v is a minimum and it is added to C. Otherwise, v is added to A and the algorithm is recursively applied on the lower link $L^{-}(v)$, producing lists A', B', C'. Such lists define the Forman gradient V' on $Lk^{-}(v)$. The recursive call is performed until all the lower links are empty.

Then, the Forman gradient V on v is defined as follows:

- the lowest critical vertex w is chosen from C' and the edge $v * w$ is added to B,
- for each i-simplex σ (different from w) in C', the $(i+1)$-simplex $v * \sigma$ is added to C,
- for each i-simplex σ in A' the $(i+1)$-simplex $v * \sigma$ is added to A and the $(i+2)$-simplex $v * V'(\sigma)$ is added to B.

Once all the lower links of vertices in Σ have been processed, a *persistence canceling step* is performed in increasing order of dimension. For each critical i-simplex σ, all the gradient paths to critical $(i-1)$-simplexes are found. A critical i-simplex σ can be canceled with critical $(i-1)$-simplex γ if and only if there is only one gradient path from σ to γ. The effect of a cancellation is to reverse the gradient path connecting σ and γ. A further description of i-cancellations is presented in Section 2.2.

Cancellations are applied in the order of increasing persistence while the function that extends the scalar field f to the simplexes of Σ, and whose values are considered in the definition of persistence, is given by $fmax(\sigma) = \max\limits_{p \in \sigma} f(p)$.

In Figure 2.7, an example of the Forman gradient extraction on the link of a vertex is illustrated. The star of vertex 9 is shown in Figure 2.7(a). The application of the algorithm to the lower link of vertex 9, illustrated in Figure 2.7(b), produces the Forman gradient V' composed by the following lists :

A'	= 3; 4; 6; 7; 8
B'	= [3,2]; [4,1]; [6,5]; [7,1]; [8,2]
C'	= 1; 2; [4,3]; 5; [7,6]; [8,5]

In Figure 2.7(c), the lists are updated after the cancellations performed on V'. Vertex 2 is deleted with edge (3,4) while vertex 5 is deleted with edge (6,7). Then, V' is extended to V obtaining the following lists (showed in Figure 2.7(d)).

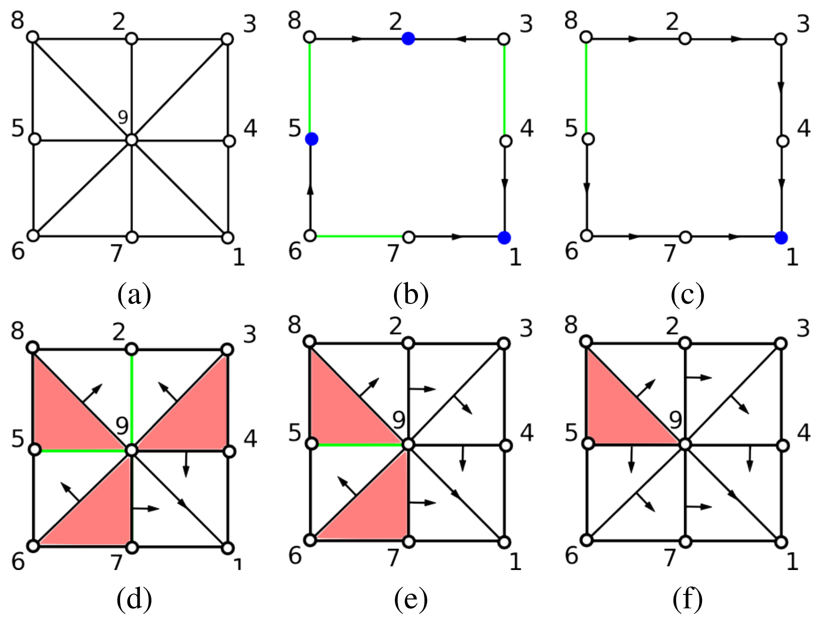


Figure 2.7: (a) The lower star of vertex 9. The Forman gradient V' on the link of 9 before (b) and after (c) the cancellation of critical edge (4,3) and vertex 2, and edge (7,6) and vertex 5. The critical vertices are blue, critical edges are green and critical triangles are red. (d) The Forman gradient V in the lower star of vertex 9.

A = 1; [3,9]; [8,9]; [5,9]; [7,9]; [9,4]
B = [1,9]; [3,9,2]; [9,8,2]; [5,9,6]; [9,7,1]; [9,4,1]
C = [2,9]; [5,9]; [3,4,9]; [5,8,9]; [6,7,9]

In Figure 2.7(e), the cancellation is performed also on V on the first pair of critical simplexes, the edge (2,9) and the triangle (3,4,9). In Figure 2.7(f), the final gradient setting after all the possible cancellations have been performed, is shown.

2.1.5.2 Forman gradient computation based on Forman function.

The algorithm proposed in [GBHP11] computes the Morse-Smale complex starting from a regular n-dimensional CW-complex Γ with scalar field f defined at the vertices of Γ, that is a n-dimensional cubical complex. Function f is extended to a Forman function F, defined on all cells of Γ, such that $F(\sigma)$ is slightly larger than $F(\tau)$ for each cell σ and each face τ of σ. For the defined Forman function F, all cells of Γ are critical. A discrete gradient vector field is computed by assigning gradient arrows in a greedy manner in ordered sweeps over the cells of Γ according to increasing dimension and increasing F value. Each current non-paired and non-critical cell in the sweep is paired with its coface with only one facet not marked (as critical or as already

Algorithm	Worst-time complexity	Input	Output
Magillo et al. [MDDF+08]	$O(\|\Sigma_n\|)$	triangle mesh	Morse complexes
Gyulassy et al. [GNPH07]	$O(\|\Sigma_0\| log \|\Sigma_0\|)$	tetrahedral mesh	Morse-Smale complex
Takahashi et al. [TIS+95]	$O(\|\Sigma_0\|)$	triangle mesh	Morse complexes
Edelsbrunner et al. [EHNP03]	$O(\|\Sigma_0\|)^2$	tetrahedral mesh	Quasi Morse complex
Meyer et al. [Mey94]	$O(\|\Gamma_0\| log \|\Gamma_0\|)$	reqular square grid	Morse complexes
Vincent et al. [VS91]	$O(\|\Gamma_0\| log \|\Gamma_0\|)$	reqular square grid	Morse complexes
Mangan et al. [MW99]	$O(\|\Gamma_0\| log \|\Gamma_0\|)$	reqular square grid	Morse complexes
King et al. [KKM05]	$O(\|\Sigma_0\|)$	tetrahedral mesh	Forman gradient
Gyulassy et al. [GBHP11]	$O(\|\Sigma_0\|)$	cubical complex	Forman gradient
Robins et al. [RWS11]	$O(\|\Sigma_0\|)$	cubical complex	Forman gradient
Shivashankar et al. [SMN12]	$O(\|\Sigma_0\|)$	regular square grid	Forman gradient

Table 2.1: Algorithms for the extraction of ascending or descending Morse complexes, Morse-Smale complexes or Quasi Morse-Smale complexes. For each algorithm, as it is in its original definition, the worst-time complexity is indicated as well as the expected input and the output generated.

paired). If there are several of such cofaces the lowest is taken. If there is no such coface, a cell cannot be paired, and it is critical. This pairing defines a discrete gradient vector field.

The 1-skeleton of the Morse-Smale complex is computed starting from this gradient vector field. Critical cells of F (and not critical points of f) and the discrete gradient paths connecting them determine the nodes and arcs in the 1-skeleton of the Morse-Smale complex (incidence graph). In [GP12], this algorithm has been extended to extract also 2- and 3-cells of the Morse-Smale complex when the input is a hexahedral mesh.

The order in which the cells in Γ are processed by the algorithm is not completely deterministic, since there could be many different i-cells in Γ with the same value of function F. As a consequence, some unnecessary critical cells may be produced by the algorithm.

In Table 2.1 we show a summary of the algorithms described in detail within this section indicating the input format, the cells of the Morse or Morse-Smale complex computed as output and the worst-time complexity. We denote with Σ_0 the collection of the vertices of the n-dimensional simplicial mesh Σ, and with Σ_n the collection of n-cells of Σ.

2.2 Simplification of Morse complexes

One of the major issues that arises when extracting Morse and Morse-Smale complexes from a discrete scalar field is the over-segmentation due to the presence of noise. To deal with this problem, simplification algorithms have been developed in order to remove less significant features from Morse-Smale complexes. All of them are based on simplification operator defined in the Morse theory called i-*cancellation* [Mat02]. A cancellation removes a pair of critical points of the function f (see Section 1.2.1) smoothing the function in the neighborhood of the critical point.

i-$cancellation$ in 2D consists of collapsing a maximum and a saddle pair into a maximum (*maximum-saddle* cancellation), or a minimum and a saddle into a minimum (*minimum-saddle* cancellation). In 3D, it consists of collapsing a maximum-2-saddle pair into a maximum, a minimum-1-saddle pair into a minimum, or collapsing a 1-saddle and a 2-saddle into a 1-saddle (or 2-saddle). Updates resulting from an i-$cancellation$ are usually described in terms of updates on Morse-Smale complex.

In 2D, the saddle-extremum cancellation, called saddle-minimum (0-$cancellation$) or saddle-maximum (1-$cancellation$), removes two critical points from the Morse-Smale complex merging the 2-cells and 1-cells around the critical points pair to reconnect the complex. In 3D, a saddle-extremum cancellation is analogue to the 2D case. A saddle-saddle cancellation removes a saddle pair (1-saddle-2-saddle) merging the 2-cells and 1-cells in the neighborhood the critical point pair. In Section 4.3.2, we will described the updates resulting from this operators in terms of modifications on the Incidence-Graph (IG) [Ede87]: this is actually the best way to see how the cells of the Morse-Smale complex reconnect after a saddle-saddle cancellation. In [GN05], it has been shown that the Morse-Smale complex gains cells after a saddle-saddle cancellation because re-routing the descending cells creates new intersections with the ascending cells. To overcome this problem, in [CF11] a minimally complete set of operators for simplifying n-dimensional Morse complexes has been presented. We will describe such operators in Section 4.

The pairs of critical points removed in a cancellation are usually computed through a filtration of the cell complex Γ discretizing the domain of function f, following the persistence algorithm described in [ELZ02]. The persistence of a critical pairs is defined in 2D as the absolute difference between the function values of the two critical points and indicates the importance of a critical point. The definition of persistence pair relates to persistence homology [Ede87]. A critical point is called *positive*, if it creates a component in the filtration of Γ, or *negative*, if it destroys a component in Γ. Critical pairs are defined as the pairs of negative saddles with preceding positive minima and of negative maxima with preceding positive saddles. Persistence can be seen as the difference in the birth times of the two critical points.

However, the unique persistence pairing defined in [ELZ02] is too restrictive to support a flexible simplification of a scalar field. In [BPH09] the notion of *variance* is introduce relaxing the unique pairing of persistence. *Variance* is defined, for each pairs of adjacent critical points p and q in a scalar field, as $v = |f(p) - f(q)|$. We will compute the importance of a pair of critical point in the same way extending the idea in dimensions higher than two.

Topological persistence has been demonstrated as a powerful importance measure and has been used in many applications including scalar field simplification and shape matching. However, such measure lacks of information related to the geometry on which the function is defined. In [DSNW13] the notion of *topological saliency* is introduced with the scope of reflecting the importance of a feature related to other features in its neighborhood. Intuitively, the topological saliency of a feature, normalizes the persistence of features present in its neighborhood. Let $\rho^{(i)}$ be the persistence of the topological feature created at minimum $q_i \in Q = \{q_1, \cdots, q_k\}$ and let $d_g(x, c_i) \leq r$ the geodesic distance between two points $p, q \in \mathbb{M}$. Consider a neighborhood $N_r(i) + \{x \in \mathbb{M} | d_g(x, q_i) \leq r\}$ which is the geodesic ball of radius r centered at q_i. The *topological saliency* $T_r(i)$ of the feature created at q_i is:

$$T_r(i) = \frac{\omega_i^i \rho^{(i)}}{\sum_{q_j \in Q} \omega_j^i \rho^{(j)}}$$

where ω_j^i is a weighting function for the feature j with respect to i. The topological saliency for a maximum is defined in symmetrical manner while has not be extended yet to critical points of other indices.

We can classify existing simplification algorithms for Morse or Morse-Smale complexes as *topology-based* or *geometry-based*. Topological simplification algorithms developed for 2D scalar fields eliminate less significant features from a Morse-Smale complex removing pairs of critical points through cancellations with increasing persistence values [EHZ01, BHEP04, DDFVM07, TP12]. In [EHZ01, BHEP04] the cancellations of a critical pair (p, q) in a Morse-Smale complex is treated as a contraction of the extremum p and saddle q into the other extremum p' connected with q of the same type of p. After the cancellation the 1-cells in the neighborhood of the p and q are modified accordingly to updates required by cancellation operator (see 1.2.1). In [BHEP04] it is required also that $f(q) > f(p')$ before the cancellation. This guarantees that all the paths recomputed after the cancellation remain monotonic and ensures that no level sets are eliminated except the ones between $f(p)$ and $f(q)$. In [DDFVM07], standard minimum-saddle-minimum and maximum-saddle-maximum operators have been extended to deal with macro-saddles and multiple-saddles naturally present in a Triangulated Irregular Network (TIN) discretizing a terrain.

In the 3D case, the simplification problem has been studied both for Morse-Smale [GN05] and Morse complexes [CF11]. The main problem in the 3D case is the non-trivial extension of the 2D operators. While for the (maximum, 2-saddle) and (minimum, 1-saddle), the extension from the 2D case is straightforward, the (1-saddle, 2-saddle) simplification, according to the cancellation operator, could force the introduction of some new cells in the complex in order to guarantee the combinatorial correctness in the neighborhood of the two saddles involved.

In [GNP$^+$06], a new data structure is defined encoding both the topological relations between the cells of the MS complex Γ_{MS} and the geometrical features of the descending and ascending Morse complexes corresponding to the vertices of Γ_{MS}. The geometrical primitives of the Morse complexes are stored in the leaves of a Directed Acyclic Graph (DAG). Each feature of the Morse complexes correspond to a node in the DAG and such node references the leaves encoding the geometrical primitives composing it. During the simplification algorithm the topological relations among the vertices of the MS complex are updated and the DAG is modified as well. Nodes can be deleted from the DAG when a feature is removed from the Morse complexes, or new DAG nodes are created to represent the merging of two features. Moreover, it has been shown that during a saddle-saddle cancellation no features are removed since the surface geometry become part of many ascending/descending 2-cells. Thus, the MS complex actually increases since new intersections are created during such cancellation. However, it is shown that the new cells introduced by a saddle-saddle cancellation could be removed by subsequent extrema-saddle cancellation.

In [GRSW13], the i-cancellation operator is applied on two different representations of a Morse-Smale complex, the so called *explicit representation*, corresponding to the 1-skeleton of the MS-complex, and the *implicit representation*, corresponding to the Forman gradient V, as described by Forman [For98]. Simplifying the explicit representation corresponds to apply i-cancellations on pair of critical points updating the 1-skeleton of the Morse-Smale complex in the neighborhood of the critical points removed [EHZ01]. Simplifying the implicit representations, corresponds to reverse the gradient path between two critical simplexes corresponding to the pair of critical points removed (see Section 7.1.2 for details). In 2D, an i-cancellation performed on both representations leads to equivalent results. In higher dimensions, this is no longer guaranteed and the same simplification applied on the two representations can lead to different geometric embedding of the separatrices and, consequently, to a different order of simplifications in the simplification algorithm [GRSW13]. Differences occurs only when a single separatrix can merge and split and, hence, in the 3D case this occurs at saddle-saddle simplifications only. During an 1-cancellation, of 1-saddle p and 2-saddle q, q can be connected to an arbitrary number of 1-saddles different from p. In the explicit representation each connection between saddle points is independently treated from the topological and geometric points of view; thus the 1-cancellation requires only the removal of the critical point pair and the update of the connection in the neighborhood. In the implicit representation, changing the direction of a V-path connecting p and q, can lead to a change in the connectivity of the critical points remaining if a subset of the V-path is shared by multiple separatrices. However, having an initial Forman gradient with no overlapping separation lines, both representations would yield to the same result.

In [MB09] and [CFI13c], the simplification of cell complexes in arbitrary dimension is considered to compute the homology of the complex by reducing it to a smaller complex having the same homology. The simplification operators described in [MB09], called *reduction* and *co-reduction*, are special cases of the $KiC(i+1)C$ and $KiC(i-1)C$ homology-preserving operators, described in Section 6.2, defined for cell complexes in arbitrary dimensions. A $KiC(i+1)C(q,p,p')$ removes an i-cell q and an $(i+1)$-cell p from a cell complex Γ. Recall that $b(p)/cb(p)$ denotes the immediate boundary/co-boundary of p, the cells in the neighborhood of p and q are modified as follows:

- all the cells $r_j \in b(p)$ are moved to the immediate boundary of p';
- all the cells $s \in cb(p)$ get p' in their immediate boundary.

$KiC(i-1)C$ operator has a similar behavior:

- $(n-i)$-cell q and $(n-i+1)$-cell p are removed from Γ ;
- all the cells $r \in b(p)$ are moved to the immediate boundary of p';
- all the cells $s \in cb(p)$ get p' in their immediate boundary.

A reduction corresponds to a $KiC(i+1)C(q,p,p')$ where the i-cell is incident into a single $(i+1)$-cell and dually, a co-reduction corresponds to a $KiC(i-1)C(q,p,p')$ where the $i-cell$ is incident in a $(i-1)$-cell only. Aside to the homology-preserving operators, a set of homology-modifying operators have been defined, called $KiCiCycle$(*Kill i-Cell and i-Cycle*) which kill an i-cell and an i-cycle in Γ. The set of homology-preserving and homology-modifying operators forms a minimal set of Euler operators on cell complexes in arbitrary dimensions, which subsume all the other Euler operators proposed in the literature (see Section 6.2). The set of homology-preserving operators can be seen in terms of updates on the ascending/descending Morse complexes. Operators $KiC(i+1)C(q,p,p')$ and $KiC(i-1)C(q,p,p')$ are equivalent to the $removal_{i,i+1}(q,p,p')$ and $removal_{i,i-1}(q,p,p')$ operators presented in [CF11] and they form a basis of the set of topological operators for modifying Morse complexes in arbitrary dimensions (see Section 4 for details on the effect of the operators).

In [TP12], a new approach is proposed for the topological simplification of 2D scalar fields. The simplification algorithm does no longer rely on persistent homology but it is based on the enumeration of the interesting, non-removable, critical points. Using such algorithm all the removable critical points are eliminated from the field, thus enabling the development of a more general simplification algorithm.

There have been some proposals in the literature to modify not only the Morse and Morse-Smale complexes using cancellations, but to modify also the scalar function f thus constructing a function g that corresponds to the simplified field. The first work in 2D, presented in [BHEP04] and improved in [WGS10], modifies function f numerically, using Laplacian smoothing.

In [BHEP04], after each cancellation function f is locally modified in order to agree with the new topology, by minimizing the error and obtaining a smooth approximation. The error is measured as the difference between function values at a point while the persistence ρ of the critical points involved implies a lower bound on the error. Any monotonic approximation of the curve between the two points has an error of at least $\rho/2$. Thus, the goal is to find monotonic patches that minimize the error. The technique used in [BHEP04] provides a smooth C^1-continuous approximation within an error bound along the boundaries of the quadrangular patches of the MS complex and a similar approximation, not observing error bounds, in the interior.

The steps of the geometry fitting process after each i-cancellation(p, q) are:

(i) all paths affected by the i-cancellation are found;

(ii) critical points are removed by gradient smoothing;

(iii) old regions are smoothed until they are monotonic;

(iv) new paths are recomputed using new geometry;

(v) one-dimensional gradient smoothing forces new paths to comply with constraints;

(vi) new regions are smoothed until all points are regular.

As described in [BHEP04] the paths built in step (iv) are not guaranteed to satisfy the error bounds. For this reason in step (v) gradient smoothing is used repeatedly.

In [WGS10], the bottleneck of the smoothing step performed after each cancellation in [BHEP04] is solved constructing a topologically valid function after all cancellation steps. The two C^0 methods give comparable results but the one in [WGS10] is faster. Moreover in [WGS10] a novel schema is devised to C^1-continuity.

The smoothing algorithm provided in [WGS10] consist of three steps:

(i) all the critical points whose persistence is lower then a given threshold are removed via i-cancellation;

(ii) a quick preview reconstruction of the smoothed scalar field by computing a harmonic function in the interior of each cell of the MS complex. Only C^0-continuity is guaranteed by the harmonic function;

(iii) a C^1-continuity is computed by constrained bi-Laplacian smoothing of the input scalar field where constraints force the scalar field to be monotonic within each Morse cell.

Step iii is converted to an unconstrained numerical optimization problem and solved numerically.

2.3 Hierarchical models for Morse complexes

One of the most relevant problems dealing with scalar fields is the huge size of the data set and their complexity. To represent and handle such data a hierarchical model or a multi-resolution model becomes crucial. First investigations for a hierarchical representation have been done in image analysis to reduce over-segmentation, naturally present using watershed transformation as a segmentation tool [Beu94].

The hierarchical watershed approach described in [Beu94] is based on the simplification process performed on a new image, called *mosaic image*, obtained from the original image (scalar field $\mathbb{M}_\Gamma = (\Gamma, f)$ where Γ is a regular square grid and f associates a function value to each pixel of Γ). Starting from the original image M_Γ and the catchment basins CB_i computed on M_Γ (see Section 1.5) a new function f' is defined on the pixels of Γ by extending the value of $f(p)$ (where p is the seed of the catchment basin CB_p) to the entire CB_p. Note that the watershed of this new image $M'_\Gamma = (\Gamma, f')$ is equal to the watershed of the initial image. Then, the hierarchy and suppression of the over-segmentation is obtained by merging adjacent catchment basins. A graph structure $G = (N, A)$ is defined on the mosaic image. For each 1-cell σ_{ij} of the watershed lines (1-cell dividing two catchment basins CB_i and CB_j) a node in the graph is instantiated with value $h(\sigma_{ij}) = |f(i) - f(j)|$. Then, for σ_{ij} an arc is introduced in the graph for each 1-cell bounding the same catchment basins of σ_{ij}. The watershed transformation is performed on G, flooding starts from the arcs with minimum evaluation and propagates towards the other arcs. Finally, the watershed arcs are made of those arcs which surround the homogeneous regions. This produces the desired result. This hierarchy works specifically for images since is based on the observation that over-segmentation is produced by low contrast variations inside the image. For this reason the criterion used inside the hierarchy is not a threshold but simply the fact that the values of function values of h are lower in the over-segmented part of $\mathbb{M}_\Gamma$ and higher in the real boundaries. The watershed transformations applied on h can be seen as a series of minimum-saddle cancellations applied on the ascending Morse complex Γ_a computed on f. The same idea can be applied to the descending Morse complex Γ_d applying the hierarchy to the opposite function $-f$. Such hierarchical structure can be generalized to higher dimensions since the simplification performed correspond to general minimum-1-saddle cancellations (or dual maximum-(n-1)-saddle cancellations). We will see that such operators are dimension independent (see Section 4).

Most common hierarchical models for Morse complexes have been applied to terrain modeling; Generally they can be applied to 2-manifolds embedded in 3D space endowed with a scalar field. Hierarchical models can be classified in two sets on the basis of the information in the hierarchy are *topological* or *geometric*. In [EHZ01] a hierarchy for the topology of terrains, is defined. The hierarchy is created by applying cancellations on the 1-skeleton of a Morse-Smale complex and is encoded as the MS complex at the coarsest resolution plus the sequence of *anticancellation* inverse to the cancellation used in the construction phase. These models are also called *progressive models* since all the MS complexes generated in the simplification phase can be obtained from the coarsest MS complex by applying the anticancellations in sequence. In [BHEP04, BPH09] a hierarchy of anti-cancellations is constructed. The hierarchy is built from a sequence of simplifications as defined in [EHZ01]. Then a relation between the simplifications is defined. Two cancellation are called *independent*, if they do not modify the same neighborhood of the MS complex, otherwise they are *dependent*. A *dependency graph* is derived from these relations. More precisely, the dependency relation between refinements is defined in terms of a *diamond*. The diamond associated with a anti-1-cancellation(q, p) is a quadrangle z_1, p, z_2, p', where z_1 and z_2 are the two (not necessarily distinct) minima connected to 1-saddle q and p' is the other maxima connected to q different from p. Dually, in the diamond associated with an anti-i-cancellation$(q, p$, z_1 and z_2 are the two maxima connected to q and p' is the other minima connected to q different from p. Two refinements are dependent if the associated diamonds have at least one vertex in common. In [BPH09], the dependency relation has been modified. Two refinements are dependent if if they share an edge. The dependency relation in [BPH09] is clearly less restrictive than the one in [BHEP04] (we will compare this approaches with our multi-resolution model in Section 5).

A first attempt to couple the inspection of both the topology and the geometry of a terrain has been studied in [DDFMV10]. The multi-resolution model, called *Multi-resolution Morse triangulation* (MMT) is based on the topological model discussed in [DDFVM07] and encodes an MS complex for a terrain $\mathbb{M}_\Sigma = (\Sigma, f)$ defined on a triangle mesh Σ. The model is constructed from a sequence of simplifications which can be of three types:

1. simplifications affecting the interior of the ascending/descending 2-cells only;
2. simplifications affecting the interior of the ascending/descending 1-cells only;
3. simplifications affecting the ascending/descending 0-cells.

Simplifications applied on the triangulation are *half-edge collapse*. An half-edge collapse, denoted $(v_i, v_j) \rightarrow v_j$, collapses two vertices v_i and v_j connected by an edge, into v_j. Triangles incident in the edge (v_i, v_j) are removed and triangles incident only in v_i become incident in v_j. The simplification of type 1 performs an half-edge collapse inside a Morse 2-cell (thus not on the 1-skeleton of the MS complex) modifying only the triangular mesh Σ. Simplifications of type 2 are similar since they do not change the topological structure of the 1-skeleton of the MS complex but they modify the set of edges forming it. Simplifications of type 3 collapse a critical point from the 1-skeleton of the MS complex and this trigger a simplification operation on the combinatorial representation of the 1-skeleton of the MS complex. An i-cancellation is applied, collapsing a minimum and a saddle into an adjacent minimum or a maximum and a saddle into an adjacent maximum. From the set of performed simplifications three hierarchies are constructed. Each hierarchy encodes all the inverse of the simplifications (refinements) of a certain type (1, 2 or 3) connected by a dependency relation based on the vertices of Σ. The hierarchy encoding the representations of Σ is interlinked with the hierarchy encoding the representations of the 1-skeleton of the MS complex. The latter is connected to the hierarchy representing the combinatorial structure of the 1-skeleton. The links go from refinements on the combinatorial structure to refinement on the geometry of the 1-skeleton to refinement on the triangle mesh.

Chapter 3

Computing and Representing Morse Complexes

As described in Section 2.1, several algorithms have been proposed in the literature to extract morphological information from a scalar field. A relevant issue in the application domains, such as terrain modeling, volume and data analysis and visualization, is representing such morphological information in an efficient and compact way. In this chapter we describe our contributions on representing and computing Morse and Morse-Smale complexes.

We start presenting an efficient representation for the Morse complexes. In [CFI10] we have proposed a compact dimension-independent graph-based representation, called *Morse Incidence Graph MIG*, for both the ascending and descending Morse complexes. By exploiting the duality between the ascending and descending Morse complexes, Γ_a and Γ_d, the MIG represents the incidence relations between the cells of Γ_a and Γ_d, as well as the 1-skeleton of the Morse-Smale complex. We will describe such representation in Section 3.1.

Since we are interested in computing Morse and Morse-Smale complexes on triangular and tetrahedral meshes, we focus our attention on representations for simplicial complexes well-suited for computing and encoding Morse complexes. We start observing the relationships between a simplicial mesh Σ and the Morse complexes and we express (in Section 3.2) the primal/dual relationship between the *descending* and *ascending* Morse complexes in terms of the supplied simplicial mesh Σ, and its dual mesh Σ_d. The two complexes, Σ and Σ_d, are dual to each other and the cells of the descending complex can be expressed in terms of the simplices of the primal tetrahedral mesh Σ, while the ones of the ascending complex in terms of the cells of the dual mesh of Σ. Furthermore, we express the combinatorial structure of the MS complex as collection of cells in the mesh obtained by the intersection of Σ and Σ_d, which we refer to as the *dually subdivided* mesh Σ_S. This leads to a compact encoding of these complexes in terms of only the vertices and tetrahedra of the primal mesh. In this way, we can efficiently express morphological structures of the scalar field, such as the regions of influence of critical points (i.e. the maxima, minima, and saddles), the arcs of the extrema graphs and the 1-skeleton of the Morse-Smale complex [WIFF13].

Then, inspired by the discrete Morse theory, we define a new encoding for a Forman gradient defined over irregular simplicial meshes [WIFF13]. In such representation, described in Section 3.3, information are attached only to the triangles and tetrahedra and has been used as a compact representation for the Forman gradient of a discrete Morse function defined over a mesh (see Section 1.4 for details on the discrete Morse theory). The compact encoding and the primal/dual relationship between the Morse complexes defined on a simplicial mesh are suitable to be combined with any topological data structure encoding just the vertices and top-simplexes of the mesh.

Another relevant contribution is an efficient extension of the algorithm presented in [RWS11] for computing a Forman gradient on cubical complexes, to simplicial meshes with irregular connectivity. We describe our algorithm in Section 3.4, discussing its time and space complexity. In our work we have also extended to simplicial meshes the *watershed by simulated immersion* algorithm presented in [VS91]. We have implemented such algorithm for computing Morse complexes on triangle and tetrahedral meshes. Results obtained from the Forman based algorithm has been compared with the results obtained with *watershed by simulated immersion* algorithm [VS91] (see Section 2.1.4.1). A full comparison on these two approaches is done, as a means of obtaining Morse decompositions of tessellated manifolds endowed with scalar fields, in Section 3.5.

3.1 The Morse Incidence Graph (MIG)

The *incidence-based representation* presented in [CFI10] is a dual representation for the ascending and descending Morse complexes, Γ_a and Γ_d, extracted from a scalar field $\mathbb{M} = (M, f)$. We call this graph *Morse incidence graph* (MIG).

The MIG is a dimension-independent representation for Morse and Morse-Smale complexes. A MIG only encodes the top cells of the Morse complexes. Thus a MIG is defined in the continuous as well as the discrete case and it is independent of the combinatorial manifold used to describe M. In the following we will describe the MIG representation based on the most general class of discrete manifolds, the regular CW-complex Γ.

Recall that there is a one-to-one correspondence between i-saddles p with i-cells p in the descending Morse complex Γ_d, and dual $(n - i)$-cells p in the ascending Morse complexes Γ_a, $0 \leq i \leq n$. This duality is exploited to define a representation which encodes both the ascending and the descending complexes at the same time, as an *incidence graph* [Ede87]. The incidence graph encodes the cells of a complex as nodes, and a subset of the boundary and co-boundary

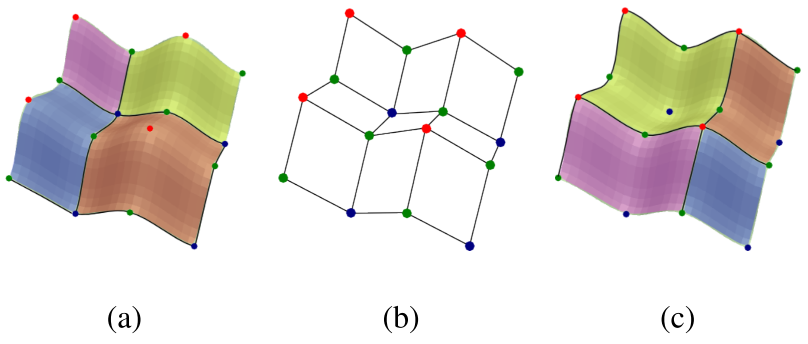


Figure 3.1: (b) MIG representing the incidence relations between the cells of the descending (a) and ascending (c) Morse complexes.

relations between cells as arcs.

The *Morse Incidence Graph* (MIG) associated with an n-dimensional descending and ascending Morse complex Γ_d and Γ_a is a graph $G = (N, A, \varphi)$, in which:

- the set of nodes N is partitioned into $n + 1$ subsets N_0, N_1,...,N_n, such that there is a one-to-one correspondence between nodes in N_i (which we will call *i-nodes*) and the i-cells of Γ_d (and thus the $(n - i)$-cells of Γ_a),
- there is an arc joining an i-node p with an $(i + 1)$-node q if and only if i-cell p is on the boundary of $(i + 1)$-cell q in Γ_d (q is on the boundary of p in Γ_a),
- each arc connecting an i-node p to an $(i + 1)$-node q is labeled by the number of times i-cell p (corresponding to i-node p in Γ_d) is incident to $(i + 1)$-cell q (corresponding to $(i + 1)$-node q in Γ_a). The label, denoted $\varphi((p, q))$, is also called the *multiplicity* of the arc (p, q).

The MIG is also called combinatorial representation of the 1-skeleton of the Morse-Smale complex. Figure 3.1 shows the MIG representing the incidence relations between an ascending (a) and descending (c) Morse complexes. Each node in the graph (b) corresponds to a critical point in (a) and (c) and for each arc connecting two nodes in (b) the two cells corresponding to such nodes are incident to each other in (a) and (c).

3.1.1 Encoding the Morse incidence graph

The data structure designed for encoding a MIG couples the graph, representing the incidence relations between the cells of the Morse complexes, with a representation of the cell complex

Γ discretizing the domain of the Morse function f. It associates with each node representing a minimum, the list of cells in Γ forming its ascending Morse cell Γ_a and with each node representing the maximum, the list of cells in Γ forming its descending Morse cell Γ_d. Note that an ascending/descending Morse cell can be represented as a collection of top cells as well as a collection of 0-cells.

In the MIG encoding, the incidence graph $G = (N, A, \varphi)$ is encoded as three arrays of nodes (one for minima, one for maxima and one for k-saddles) plus an array of arcs. Each element of the array of the nodes corresponding to minima encodes a minimum p and contains:

- 0-cell $p \in \Gamma$,
- the list of the 0-cells or top cells in the underlying complex Γ forming the corresponding ascending Morse cell,
- a list of the arcs in G incident in p.

Dually, each element of the array of the nodes corresponding to maxima encodes a maximum p and contains:

- 0-cell $p \in \Gamma$,
- the list of the 0-cells or top cells in the underlying complex Γ forming the corresponding descending Morse cell,
- a list of the arcs in G incident in p.

Each element of the array of the saddles contains the lists of all saddles with the same index i and, for each saddle q,

- the index of 0-cell $q \in \Gamma$,
- the lists of arcs joining q to nodes of index $i + 1$,
- the lists of arcs joining q to nodes of index $i - 1$.

Arcs are explicitly encoded in an array of lists. The i-th element of the array contains a list of arcs connecting nodes corresponding to i-saddles to nodes corresponding to $(i+1)$-saddles. Each element of any of such lists corresponds to an arc a and contains the indexes of the two nodes in which a is incident plus an integer indicating the multiplicity of the arc. The resulting data

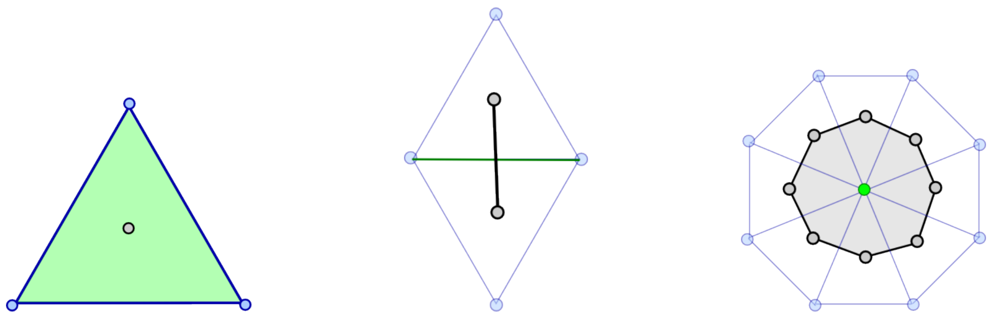


Figure 3.2: The primal/dual relationships in a triangle mesh. Each dual cell i_d (gray) is an i-polytope contained within the *star* (blue) of its corresponding primal k-simplex k_p (green), where $i + k = 2$

structure is completely independent by the dimension of $\mathbb{M}$ and by the discrete representation of Γ. All the information stored in the incidence graph regarding the incidence relations between the Morse cells are referred as *topological MIG* while all the geometrical information regarding the cells of the Morse complexes are also called *geometrical MIG*.

3.2 Primal/dual representation for discrete Morse complexes on simplicial meshes

As described in the previous section, representations for a Morse complex can be defined independently from the manifold used to discretize the domain of the Morse function f. However, in application domains a specific representation has to be chosen in order to optimize both the computation and the analysis of such complexes. Simplicial meshes are widely used representations for discretize a continuos domain. Experimental data are often characterized by a cloud of points, regularly or irregularly distributed on the domain, with a single value associated to each point. A simplicial mesh is built on such cloud of points to induce a topology on such set. Computing the Morse and Morse-Smale complexes on a simplicial mesh Σ, using discrete Morse theory (see Section 1.4), means to group the k-simplexes, with $0 \leq k \leq n$, forming all the ascending and descending Morse cells or the cells of the Morse-Smale complex. In this section, we present an interpretation of the Morse and Morse-Smale (MS) complexes in terms of the provided simplicial mesh and its dual mesh.

We denote the *primal* simplicial mesh as Σ. The *dual* mesh of Σ, which we denote as Σ_d is a polyhedral mesh in which the 0-cells (vertices) correspond to the n-simplexes of Σ, the 1-cells (edges) correspond to the $(n-1)$ simplexes of Σ and so on. In the following, we will call *primal*

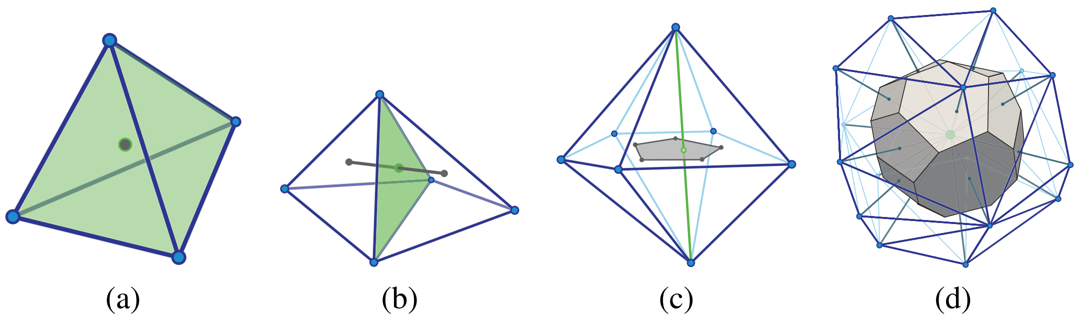


Figure 3.3: The primal/dual relationships in a tetrahedral mesh. Each dual cell i_d (gray) is an i-polytope contained within the *star* (blue) of its corresponding primal k-simplex k_p (green), where $i + k = 3$

the simplices of Σ and *dual* the cells of Σ_D. The 0-cells of Σ_D can be geometrically placed at the centroid of the corresponding d-simplex.

Let us consider first the 2D case (see Figure 3.2). Each primal triangle corresponds to a dual vertex (Figure 3.2(a)). Primal edges, on the boundary of two primal triangles, correspond to dual edges having two dual vertices in their boundary (see Figure 3.2(b)) and, similarly, primal vertices correspond to dual 2-cells (see Figure 3.2(c)).

In the 3D case (see Figure 3.3) each primal tetrahedron corresponds to a dual vertex (see Figure 3.3(a)). Primal triangles, on the boundary of two primal tetrahedra, correspond to dual edges having two dual vertices in their boundary (see Figure 3.3(b)). Primal edges instead correspond to dual 2-cells and primal vertices correspond to dual 3-cells (see Figures 3.3(c) and 3.3(d)).

As described in Section 1.4, in a triangle mesh Σ, maxima correspond to triangles, minima correspond to vertices, and, saddles to edges. In the Morse complexes defined on Σ a *descending 2-manifold* corresponds to a maximum, and thus to a collection of primal triangles. An *ascending 2-manifold* corresponds to a minimum, and thus to a collection of dual 2-cells, each of which is a primal vertex. A *descending 1-manifold* corresponds to a saddle, and thus to a sequence of primal edges. An *ascending 1-manifold* corresponds to a saddle as well, and thus to a sequence of dual edges, each of which corresponds to a primal edge. Therefore, the descending Morse complex consists of elements from the primal mesh, while the ascending Morse complex consists of elements from the dual mesh.

Let us consider now the 3D case:

- A *descending 3-cell* corresponds to a maximum, and thus to a collection of (primal) tetrahedra. Dually, an *ascending 3-cell* corresponds to a minimum, and thus to a collection of dual 3-cells (i.e., primal vertices).
- A *descending 2-cell* corresponds to a 2-saddle, and thus to a collection of primal trian-

gles, each of which can be expressed as a pair of primal tetrahedra. An *ascending 2-cell* corresponds to a 1-saddle, and thus to a collection of dual 2-cells, each of which can be expressed as a pair of dual 3-cells, corresponding to a primal edge.

- A *descending 1-cell* corresponds to a 1-saddle and thus to a sequence of primal edges, or, equivalently, as a sequence of primal vertices. An *ascending 1-cell* corresponds to a 2-saddle and thus to a sequence of dual edges, which can be seen as a pair of dual vertices (i.e., as a sequence of primal tetrahedra).

We can observe that, since the primal edges can be expressed as pair of vertices and the primal faces can be expressed as pair of top simplexes, all the Morse cells can be expressed in terms of vertices and top simplexes.

Figure 3.10 illustrates the above observations for a discrete Morse function defined on a triangle mesh, where the descending 2-manifolds (see Figure 3.10(c)) are collections of triangles from Σ associated with the maxima (red critical points), while the ascending 2-manifolds (see Figure 3.10(e)) are collections of dual 2-cells (corresponding to vertices from Σ) associated with the minima (blue critical dots). Similarly, the descending (see Figure 3.10(d)) and ascending 1-manifolds (see Figure 3.10(e)) correspond to collections of primal and dual edges, respectively, associated with the saddles (green critical points).

Using the above correspondences, we observe that the k-saddles of the Morse function correspond to k-simplices in the primal mesh Σ and to $(n-k)$-cells in the dual mesh Σ_d. Equivalently a *descending k-cell* corresponds to a k-saddle, thus it correspond to a collection of primal k-simplices and thus to a collection of dual $(n-k)$-cells. Dually, an *ascending k-cell* corresponds to a $(n-k)$-saddle, and thus to a collection of dual k-cells, and thus to collection of primal $(n-k)$-simplices.

Note that all the descending and ascending manifolds are expressed entirely in terms of top simplexes and vertices. This is a relevant issue from an implementation point of view, since there is no need to encode the primal k-simplexes (with $0 \leq k \leq n$), and of course no need for encoding the dual mesh. In particular, each manifold of the Morse complex can be described in terms of collections of cells of uniform dimension from the primal or dual mesh, each of which can be expressed using at most $\lceil (n+1)/2 \rceil$ vertices or n-simplices.

3.2.1 Representing the cells of the MS complex

We define the *dually subdivided mesh*, denoted as Σ_S, as the mesh obtained by the intersection of the primal mesh Σ and its dual Σ_d (see Figure 3.10(a) for a 2D example).

Each top cell of Σ_S, called top *micro-cell*, is the intersection of a top simplex σ and of a dual top cell σ_d (corresponding to a vertex). A n-micro-cell can be complete described by a pair of

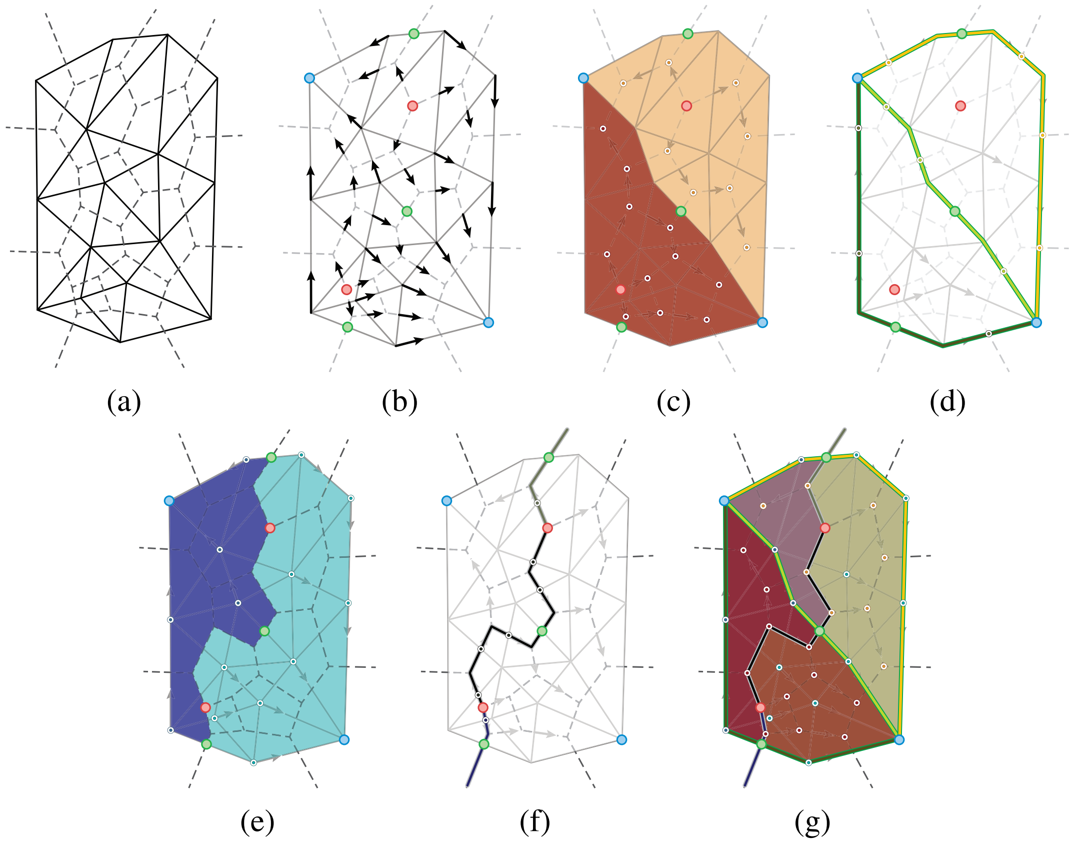


Figure 3.4: Two dimensional example of our scheme for simplicial meshes. (a) We overlap the triangle mesh Σ (solid lines) with the dual mesh Σ_d (dashed lines). (b) Encoding the *Discrete Morse gradient field* entirely with the triangles enables the use of compact topological data structures for morphological extraction. We associate the *descending Morse complexes* with the cells of Σ (c-d) and the *ascending Morse complexes* with the cells of Σ_d (e-f). Finally, we associate the *Morse-Smale complex* with the *dually subdivided* tetrahedral mesh Σ_S (g) whose hexahedral cells are defined by a tetrahedron and one of its vertices. All relations are encoded strictly in terms of the vertices and tetrahedra of Σ.

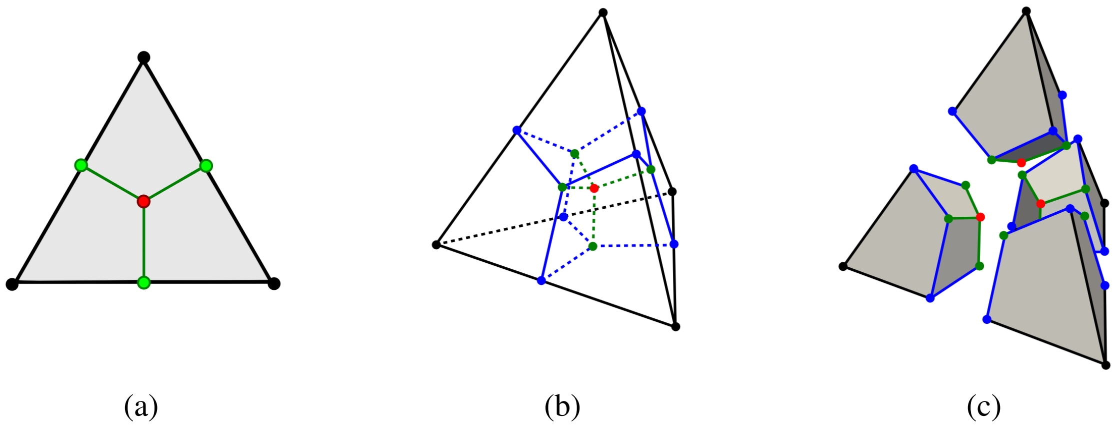


Figure 3.5: A dually-subdivided triangle (a) is decomposed into three quads. Similarly a tetrahedron σ (b) is decomposed into four hexahedra (c), each defined by a vertex of σ (black) and interior points from each incident face within σ.

indices $<$ n-simplex,vertex$>$, both within the primal mesh.

For the 2D case, the 2-micro-cells are quadrilateral cells (quads) that we call micro-quads whose boundary consists of four *micro edges*. Micro-quads are defined by a triangle σ and a 2-cell in the dual mesh Σ_d (which is a vertex v in the primal mesh). Then they are encoded as a pair (σ, v). Each triangle is decomposed into three micro-quads. (see Figure 3.5(a))

In 3D, the intersection of a tetrahedron σ with the dual 3-cell σ_d defines a hexahedron, which we refer to as a *micro-hex*, whose boundary consists of six quadrilateral *micro-quads* and twelve *micro-edges*. Each tetrahedron is decomposed into four hexahedra (see Figure 3.5(b-c)). Thus, a tetrahedron τ in Σ is decomposed into four hexahedra in Σ_S, each corresponding to a vertex of τ (see Figure 3.5(b)). Similarly, a triangle σ in primal mesh Σ is decomposed into three micro-quads in Σ_S, namely those micro-quads corresponding to the three vertices of σ in the two tetrahedra of Σ sharing σ. An edge e in Σ is the union of two micro-edges in Σ_S belonging to the micro-hexes forming the tetrahedra incident in e and corresponding to its endpoints.

A micro-hex, being the intersection of a tetrahedron σ and a 3-cell in the dual mesh (a vertex v in the primal mesh), is encoded as a pair (σ, v). A micro-quad γ in Σ_S separates two hexahedral cells, which can either share a tetrahedron τ or a vertex v, depending on whether they are part of the same primal tetrahedron or the same dual 3-cell (corresponding to v). This leads to an encoding for γ either as a triple (τ, v_1, v_2), where v_1 and v_2 are the vertices of τ defining the two hexahedral cells (which are pairs (τ, v_1) and (τ, v_2)), or as a triple (τ_1, τ_2, v), where τ_1 and τ_2 are the two tetrahedra defining the two hexahedral cells (which are pairs (τ_1, v) and (τ_2, v)).

If the cells of the Morse complexes consist of elements from the simplicial mesh Σ and from the dual mesh Σ_d, the (macro) cells of the MS complex consist of elements from the dually subdivided mesh Σ_S.

In the 2D case:

- a Morse-Smale 2-cell is a collection of micro-quads (see Figure 3.10(g)),
- a Morse-Smale 1-cell is a collection of micro-edges on the boundary of two micro-quads (see Figure 3.10(g)).

Figure 3.10(g) illustrates the Morse-Smale complex associated with the discrete Morse gradient field of Figure 3.10(b). Note that each micro-quad is defined by the intersection of a triangle (a primal 2-cell) and a dual 2-cell associated with one of its boundary vertices, and that each (macro) 2-cell of the MS complex is defined by a maximum (red critical point), a minimum (blue critical point) and two saddles (green critical points).

In the 3D case:

- a Morse-Smale 3-cell is a collection of micro-hexes,
- a Morse-Smale 2-cell is a collection of micro-quads on the boundary of two micro-hexes,
- a Morse-Smale 1-cell is a collection of micro-edges on the boundary of multiple micro-edges.

The 1-cells of the MS complex in 3D are all sequences of *micro-edges*:

- *Minimum–1-saddle connector:* Each micro-edge in the sequence connects a primal vertex and edge. Thus, the connector is formed by primal edges, and can be encoded as a sequence of primal vertices, where the last two vertices define the critical edge. In Figure 3.5 the black edges connect primal vertices (black dots) to primal edges (blue dots).
- *Maximum–2-saddle connector:* Each micro-edge in the sequence connects a primal tetrahedron and face (i.e. a dual vertex and edge). Thus, the connector is formed by a sequence of dual vertices and is encoded as a sequence of primal tetrahedra. In Figure 3.5, the green edges connect primal tetrahedra (red dot) to primal faces (green dots).
- *Saddle–connector:* Each micro-edge in the sequence connects a primal triangle and edge (or, dually, a dual edge and 2-cell). In Figure 3.5, the blue edges connect primal edges (blue dots) to primal faces (green dots). The 2-saddle-1-saddle connector is a path moving from the centroid of a triangle to the centroid of one of its edges to the adjacent triangle and so on until the centroid of the critical edge is reached. We encode it as a sequence of primal triangles, whose first triangle is critical and whose last two intersect in the critical edge. Note that a saddle connector is not a subset of 1-cells of the ascending or descending Morse complexes.

3.3 A compact representation for the Forman gradient

Accordingly to the primal/dual representation for discrete Morse complexes described in Section 3.2, the Morse and Morse-Smale complexes computed on a simplicial mesh Σ can be fully described in terms of vertices and top simplexes of Σ. On the contrary, since a Forman gradient is defined on all the simplices of the mesh, a natural representation for the mesh would be the *incidence graph*.

An *incidence graph*(IG) [Ede87] is a topological data structure encoding explicitly all the cells of a cell complex Γ and all the incidence relations among such cells. For each i-cell p,

- the immediate boundary relations $b(p)$ between p and the cells in $b(p)$ are stored;
- the immediate coboundary relations between p and the cells in $cb(p)$ are stores.

If we consider as discrete model a cubical complex, we can implicitly represent all such boundary/coboundary relations among the cells of the cubical complex. Due to its regularity all the relations can be represented indexing the voxels of the cubical complex. Moreover, since a Forman gradient V defines a pairing between incident cells, V can be defined on such representation as a bit vector based on the same indexing [GRWH12].

When encoding unstructured simplicial meshes instead, the IG can be verbose, since we would explicitly encode all simplexes in the mesh plus the incidence relations above. On the contrary, data structures which encode only the vertices and the top simplexes [PBCF93, GR10] have been shown to be much more compact [CFW11]. Thus, inspired by the primal/dual interpretation for discrete Morse complexes, we have defined a new encoding for a Forman gradient defined over irregular simplicial meshes [WIFF13], called *compact gradient.*

We represent the underlying simplicial mesh Σ through an *incidence-based data structure with adjacency* (IA) introduced in [Nie97]. The IA data structure is a dimension-independent data structure encoding the 0- and n-simplexes of Σ explicitly, plus the following relations:

- for each n-simplex σ, we encode
  - the $n+1$ vertices of σ;
  - the $n+1$ n-simplexes which share an $(n-1)$-simplex with σ;
- for each 0-simplex v, we encode
  - the $n+1$ coordinates of v;
  - one n-simplex incident in v;

All the 0-simplexes (vertices) of Σ are stored in an array Σ_0, and $|\Sigma_0|$ is the number of 0-simplexes of Σ. Similarly all the n-simplexes of Σ are stored in an array Σ_n, and $|\Sigma_n|$ is the number of n-simplexes of Σ. Note that each vertex v of an n-simplex σ defines a unique $(n-1)$-face γ of σ which does not contain v.

Then, information regarding the Forman gradient V are attached only to the top simplexes. Let us consider an n-dimensional simplicial mesh Σ and an n-simplex σ in Σ. Recall that we denote as $CB(\sigma)$ the coboundary of simplex σ and as $cb(\sigma)$ the immediate coboundary of σ (see Section 1). The encoding associates with σ a subset of the discrete vector pairs involving its faces. Specifically, it encodes all vector pairs

- (τ_i, τ_j), with $\tau_i, \tau_j \in CB(\sigma)$;
- (τ_i, σ'), with $\tau_i \in cb(\sigma)$ and σ' one of the top simplexes adjacent to σ.

In an n-dimensional simplicial mesh Σ, a top simplex σ has $\binom{n}{i+1}$ faces of dimension i, and each face has $(i+1)$ simplexes of dimension $(i-1)$ on its boundary. Since each k-simplex can be paired with any of the simplexes on its boundary or coboundary, there are

$$\sum_{i=1}^{n-1} \binom{n}{i+1} \cdot (i+1)$$

possible discrete vector pairs in the restriction of the Forman gradient V to σ. Adding the n additional vector pairs from a $(n-1)$-simplex in the immediate boundary of σ to an adjacent top simplex, it gives a total of

$$\sum_{i=1}^{n-1} \binom{n}{i+1} \cdot (i+1) + n$$

possible discrete vector field pairings.

Let us consider a triangle mesh Σ as an example. The encoding associates to a triangle (2-simplex) σ in Σ a subset of the vector pairs involving its faces. In particular, σ encodes all the vectors pairs:

- (τ_1, σ'), corresponding to an arrow from an edge τ_1 to a triangle σ' (red arrows in Figure 3.6);
- (τ_0, τ_1), corresponding to an arrow from a vertex τ_0 to an edge τ_1 (blue arrows in Figure 3.6).

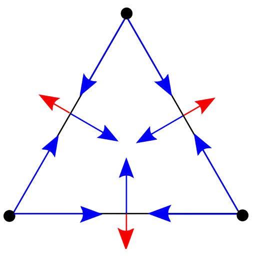


Figure 3.6: Set of arrows inside per triangle σ. Blue arrows indicate pairings between simplexes belonging to the boundary of σ (and eventually σ itself). Red arrows indicate pairings between the edges of σ and the adjacent triangles.

Then, in a triangle mesh, a triangle has $\binom{3}{i+1}$ faces of dimension i, and each face has $(i+1)$ simplexes of dimension $(i-1)$ on its boundary. Thus, there are

$$\sum_{i=1}^{2}\binom{3}{i+1}\cdot(i+1) = 3\cdot 2 + 1\cdot 3 = 9$$

possible gradient pairs in the restriction of vector field V to σ. Adding the three additional gradient pairs from an edge of σ to an adjacent triangle gives a total of 12 possible gradient pairings.

Similarly, a tetrahedron σ in a tetrahedral mesh has $\binom{4}{i+1}$ faces of dimension i, and

$$\sum_{i=1}^{3}\binom{4}{i+1}\cdot(i+1)+4 = 6\cdot 2 + 4\cdot 3 + 1\cdot 4 + 4 = 32$$

possible gradient pairs.

Such collection of pairs from the Forman gradient V in the vicinity of a top simplex σ is referred as a *local frame* of the Forman gradient. Since each such pairing within a local frame encodes a single bit of information (i.e. the presence or absence of that particular pairing), each local frame can be encoded using $\sum_{i=1}^{n-1}\binom{n}{i+1}\cdot(i+1)+n$ bit flags per top simplex. This bit flag representation simplifies testing for the presence of vector pairings as well as updates to the discrete vector field.

The restrictions imposed by discrete vector fields (i.e. that each simplex can be involved in at most one pairing) imply that there are significantly fewer valid local frame configurations than the possibilities provided by the bit flag representation. Then, we are able to encode a local frame *compressed* representing only the valid configurations.

In 2D for example, we have 12 arrows for a total of $2^{12} = 4,096$ cases. However, since we are considering a Forman gradient we have only 97 valid cases for a triangle. Thus we can encode all the possible configurations using only 1 byte per triangle. Similarly, in the 3D case we have

32 arrows for a total of $2^{32} = 4,294,967,296$ possible configurations. Considering the valid configurations we have only $51,030$ cases that we can represent with 2 bytes per tetrahedron. The efficient encoding is thus based on two representations of a local frame:

- *expanded frame* is the 12 bit flag representation for a frame in 2D (32 bits in 3D),
- *compact frame* is the 1 byte representation encoding the expanded frame compactly (2 bytes in 3D).

To simplify an easy conversion between the two representations, two small auxiliary lookup tables are used:

- *expandFrame*$[\cdot]$ is an array with 97 bit flag entries ($51,030$ entries in 3D)
- *compressFrame*$[\cdot]$ is a map from the 97 bit flags to the compressed local frame representation ($51,030$ bit flags in 3D).

We combine such compact encoding for a Forman gradient V with the IA data structure and we compare the resulting storage cost with two graph-based representations for a simplicial mesh Σ. The first graph-based structure is the IG. As discussed before the IG encodes explicitly, through a graph structure, all the simplexes σ of Σ and all the immediate boundary/co-boundary relations of σ. Let us consider the graph $G_{IG} = (N_{IG}, A_{IG})$ where N_{IG} are the k-simplex, with $0 \leq k \leq n$, and A_{IG} are the immediate boundary/co-boundary relations among all the simplexes in N_{IG}. Each element in A_{IG} is encoded with two 4 bytes pointers one for each simplex involved in the relation. Only elements in N_{IG} encoding a vertex v stores additional information regarding the coordinates of v; we are ignoring this additional storage cost since common to all the data structures considered. The Forman gradient V can be encoded on the IG adding 1 bit flag for each element in A. Recall the an arrow in V is defined only between incident simplexes. Thus, the bit flag, for each arc in A_{IG}, will indicate the presence or absence of the vector pairing between the two simplexes involved.

However, beeing the IG one of the most general representations for cell complex it is, as mentioned before, also one of the most verbose, in particular for encoding simplicial complexes. Thus we consider another graph-based structure for our comparisons which is called Simplified Incidence Graph (SIG) [FGH04]. The SIG encodes all simplexes in a simplicial complex Σ as well as all the immediate boundary relations; however only one immediate co-boundary relation for each simplex is stored. For each k-simplex σ:

- the immediate boundary relations with the $k+1$ simplexes in $b(\sigma)$ are stored;
- the immediate co-boundary relation is stored only with one simplex in $cb(\sigma)$. We call this relation a partial co-boundary relation (denote $cb^*(-)$)

Let us consider the graph $G_{SIG} = (N_{SIG}, A_{SIG})$ where N_{SIG} are the k-simplex, with $0 \leq k \leq n$, and A_{SIG} are the immediate boundary relations among all the simplexes in N_{IG} and the subset of the coboundary relations. Encoding a boundary/coboundary relation with a 4 bytes pointer, each k-simplex expect $k+1$ boundary relations and one co-boundary relation, thus $4(k+2)$ bytes (except top simplexes that have no co-boundary relations and the vertices that have no boundary relations).

Recall that $|\Sigma_k|$ denotes the number of k-simplexes in the simplicial mesh Σ. In the 3D case, the storage cost of the G_{SIG} would be:

- $|N|$ for encoding the simplexes;
- $4 \cdot (4|\Sigma_3| + (3+1)|\Sigma_2| + (2+1)|\Sigma_1| + |\Sigma_0|)$ for encoding the immediate boundary/co-boundary relations.

Also using a SIG the Forman gradient V can be encoded adding 1 bit flag for each element in A_{SIG}. Each simplex is paired with at most one simplex in its immediate boundary or at most one simplex in its immediate coboundary. When σ is paired with a $(k+1)$-simplex we store the relation with such simplex in the partial co-boundary relation $cb*(\sigma)$ of σ. When σ is paired with a $(k-1)$-simplex we store the relation with such simplex in the immediate boundary relation $b(\sigma)$ of σ; this way, all the pairings in V are encoded as a bit flag on some boundary/co-boundary relation. Thus, the bit flag, for each arc in A_{SIG}, will indicate the presence or absence of the vector pairing between the two simplexes involved. This augment the storage cost of the SIG structure by

$$\frac{1}{8} \cdot (4|\Sigma_3| + (3+1)|\Sigma_2| + (2+1)|\Sigma_1| + |\Sigma_0|) \ bytes$$

for encoding the Forman gradient.

To simplify our evaluation we consider an approximation of the number of simplexes, relating to the number of vertices, as $|\Sigma_3| \sim 6|\Sigma_0|$, $|\Sigma_2| \sim 12|\Sigma_0|$ and $|\Sigma_1| \sim 7|\Sigma_0|$ [FGH04]. Thus the total cost for the SIG with Forman gradient in the 3D case is:

$$26|\Sigma_0| + 4 \cdot (94|\Sigma_0|) + \frac{1}{8} \cdot (94|\Sigma_0|) \simeq 413|\Sigma_0|.$$

Considering a 3D instance of our representation, encoding each relation with a 4 byte pointer, the IA data structure encodes only the vertices and the tetrahedra of a tetrahedral mesh Σ. Each vertex encodes only one co-boundary relation with one incident tetrahedron (4 bytes) and each tetrahedron encodes the boundary relations with its 4 vertices ($4 \cdot 4$ bytes). Moreover for each tetrahedron we encode the relation with its four adjacent tetrahedra ($4 \cdot 4$ bytes). We will need:

Dataset	$\lvert\Sigma_0\rvert$	$\lvert\Sigma_1\rvert$	$\lvert\Sigma_2\rvert$	$\lvert\Sigma_3\rvert$	IA	SIG	IG
VISMALE	5M	36M	62M	31M	1.09 GB	1.95 GB	5.89 GB
FOOT	5M	33M	57M	28M	1.01 GB	1.8 GB	5,4 GB
BONSAI	5.9M	40M	69M	34M	1.21 GB	2.1 GB	6.54 GB

Table 3.1: Evaluation of the data structures exemplified on some real datasets. For each dataset we show, number of vertices Σ_0, number of edges Σ_1, faces Σ_2 and tetrahedra Σ_3. Cost in Gigabytes (GB) of the IA, SIG and IG encoding the tetrahedral mesh Σ and the Forman gradient

- $|\Sigma_3| + |\Sigma_0|$ bytes for encoding the simplexes;
- $4 \cdot (8|\Sigma_3| + |\Sigma_0|)$ bytes for the relations involving all the tetrahedra (Σ_3) and all the vertices (Σ_0) in Σ;
- $2|\Sigma_3|$ bytes for encoding the Forman gradient *local frame* for each tetrahedron.

Thus, simplify our evaluation as before, the total cost for the IA with Forman gradient in the 3D case is:

$$7|\Sigma_0| + 196|\Sigma_0| + 12|\Sigma_0| \simeq 215|\Sigma_0|.$$

In our work we have compared the two data structures evaluating the storage cost required by some real datasets (VISMALE, FOOT, BONSAI) what we subsume in Table 3.1. We can notice that the IA data structure occupies a little more than half the SIG. The IG occupies 4-5 times more space than the IA data structure.

3.3.1 Computing Morse and Morse-Smale complexes from a compact gradient

In this section, we discuss how to retrieve the cells of the Morse complexes (i.e. the descending and ascending manifolds), the cells of the Morse-Smale complex, and the Morse Incidence Graph from a simplicial mesh Σ endowed with a local discrete gradient field encoded with a *compact gradient*. We discuss how to extract the descending and ascending manifolds based on the simplices of the primal mesh Σ and the topological relations involved. Generally speaking, a descending or ascending k-cell is extracted by traversing the primal/dual mesh following the pairings of the gradient field, and starting from the k-simplex corresponding to the critical point associated with the descending/ascending k-cell.

Recall that a V-path of the Forman gradient V, corresponds to a sequence simplexes pairs $(\sigma_0, \tau_0), (\sigma_1, \tau_1), ..., (\sigma_i, \tau_i), ..., (\sigma_n, \tau_n)$ such that σ_i and σ_{i+1} are different faces of τ_i and (σ_i, τ_i) are paired simplexes.

Descending Morse complex The k-cells of the descending Morse complex Γ_d are naturally defined as a collection of k-simplexes of Σ. The computation of a descending k-cell always starts from a critical k-simplex σ. All the $(k-1)$-simplexes in the immediate boundary of σ are selected and, among them, only the $(k-1)$-simplexes paired with a k-simplex different from σ are considered. From such k-simplexes the breadth-first traversal of the complex continues until all the V-paths starting from σ have been visited.

Let us consider the computation of a descending 2-cell on a triangle mesh Σ. The computation starts from a critical triangle (maximum) σ. All the edges on the immediate boundary of σ are considered; among such simplexes only the edges paired with a triangle different from σ are considered. Navigated such arrows, the triangles reached are enqueued in a breadth-first traversal of the top simplexes of Σ until all the V-paths starting from σ have been visited.

Ascending Morse complex The k-cells of the ascending Morse complex Γ_a are naturally defined as a collection of k-cells of the dual mesh Σ_d or equivalently as a collection of $(n-i)$-simplexes of Σ. The computation of an ascending k-cell starts from a critical $(n-i)$-simplex σ of Σ. All the $(k+1)$-simplexes in its immediate co-boundary are selected and, among them, only the $(k+1)-simplexes$ paired with a k-simplex different from σ are considered. From such k-simplexes the breadth-first traversal of the complex continues until all the V-paths ending in σ have been visited in reverse order.

Let us consider the computation of an ascending 2-cell on a triangle mesh Σ. The computation starts from a critical vertex (minimum) σ. All the edges on the immediate co-boundary of σ are considered; among such simplexes only the edges paired with a vertex different from σ are considered. Navigated such arrows, the vertexes reached are enqueued in a breadth-first traversal of the 0-simplexes of Σ until all the V-paths ending in σ have been visited in reverse order.

The computation of the ascending/descending Morse complex is performed through constant time operations at each cell on the V-paths visited. In 2D, the extraction of a descending k-cells takes a worst-time complexity of $O(|\Sigma_k|)$ since all the k-simplexes are visited only once traversing the V-paths. Dually the complexity for the extraction of an ascending k-cell is $O(\Sigma_{(n-k)})$. In higher dimensions, the situation is more complicate. For instance, in 3D, as introduced in Section 2.1.5, gradient paths can branch and merge, potentially resulting in many-to-many adjacency relationships between critical 1-cells and critical 2-cells. For example, on a tetrahedral mesh Σ with of $|\Sigma_0|$ vertices whose discrete Morse function contains $O(|\Sigma_0|)$ critical 1-cells, each of which connects to $O(|\Sigma_0|)$ critical 2-cells. This produces a discrete Morse complex containing $O(|\Sigma_0|^2)$ gradient paths between critical 1 and 2-simplexes. Since the number of critical 1 and 2-simplexes is bounded by $|\Sigma_0|$, the number of visits to any cell during the breadth first search is also bounded by $|\Sigma_0|$ and so the complexity of the whole extraction is $O(|\Sigma_0|^3)$. Although, using a bit flag array to maintain the visited simplexes the standard breadth first traversal of the 1- 2-simplexes can be employed.

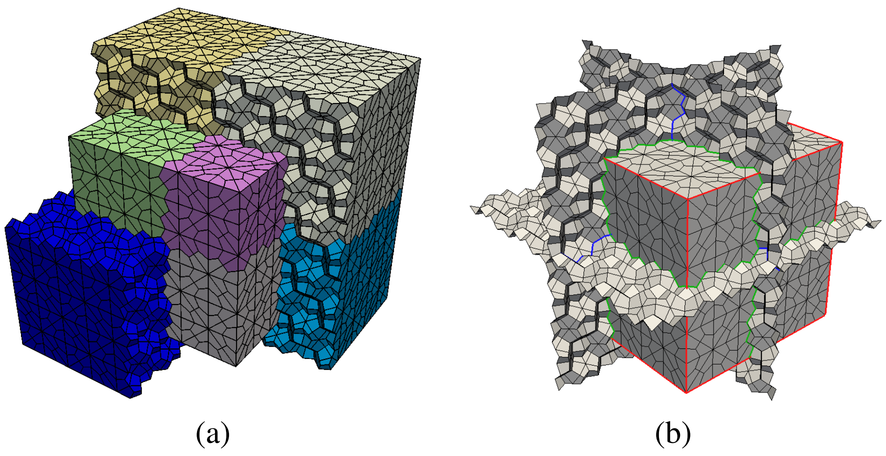


Figure 3.7: Geometrical representation of a Morse-Smale complex computed on a synthetic dataset. (a) Filtered view of the (macro) MS complex 3-cells (unique colors) composed of a set of micro-hexahedra. (b) The MS 2-cells bounded by the MS 1-skeleton. Each 2-cell is composed of a set of micro-quads and is bounded by 1-cells (saddle connectors).

Morse-Smale complex The k-cells of the Morse-Smale complex are defined as a collection of the k-cells of the *dually subdivided mesh* Σ_S obtained by intersecting the primal mesh Σ with its dual mesh Σ_d (see Section 3.2).

A top cell of the Morse-Smale complex corresponds to a pair of critical points (a maximum and a minimum) and is encoded as a collection of micro-top-cells in the dually subdivided mesh Σ_S obtained by intersecting the descending top cell (corresponding to the maximum) and the ascending top cell (corresponding to the minimum), which are collections of top simplexes and vertices, respectively.

Let us consider the 3D case. A top cell of the Morse-Smale complex corresponds to a pair of critical points (a maximum and a minimum) and is encoded as a collection of micro-hexahedra in the dually subdivided mesh Σ_S obtained by intersecting the descending 3-cell (corresponding to the maximum) and the ascending 3-cell (corresponding to the minimum), which are collections of tetrahedra and vertices, respectively (see Figure 3.7(a)). Moreover in the 3D case the 3-cells of the Morse-Smale complex are bounded by a set of 2-cells corresponding to pairs of saddles (1-saddle and 2-saddle) and composed by a collection of micro-quads. Considering the primal/dual representation described in Section 3.2, for each pair of face-adjacent micro-hexes the common micro-quad is part of the Morse-Smale 2-cell if the labels of the two hexahedra are different (see Figure 3.7(b)).

The 1-skeleton of the Morse-Smale complex is composed of different sets of 1-cells. For the 2D case, the 1-cells corresponding to a maximum-saddle or a minimum-1saddle are the 1-manifolds of the ascending and descending Morse complex, respectively. In the 3D case the same sets of 1-cells is combined with a the 1-cells called saddle-connector which connect 1-saddles with 2-

saddles. A saddle connector, between a 1-saddle p and a 2-saddle q, is computed by extracting the descending and ascending 2-manifolds associated with p and q. The descending 2-manifold extraction is performed first, and all the traversed triangles are marked as visited. Then, starting from the critical primal edge e corresponding to p and its adjacent edges, the same process as for extracting ascending 2-manifolds is performed, but only the triangles previously marked as visited are considered.

3.3.2 Computing the Morse incidence graph from a compact gradient

The Morse Incidence Graph, described in Section 3.1, represent the incidence relations between the cells of the Morse and Morse-Smale complexes defined on a simplicial mesh Σ. Such relations are computed traversing the V-paths of the *compact gradient* V defined on Σ, computing all the Morse cells in one of the two complexes, for instance, the descending complex, saving one node for each critical simplex and connecting two nodes in the graph with an arc if there is a separatrix V-path in V connecting the two corresponding critical simplexes.

Let us consider the 2D case. When computing the $MIG\ G = (N, A, \varphi)$ on a two-dimensional scalar field $\mathbb{M}_\Sigma = (\Sigma, f)$ on which has been defined a Forman gradient V, we start adding to N one node p for each critical simplex σ in V. We consider the vertex v with highest function value f in σ, the index of v is stored in p. Then, for each maximum node p corresponding to a critical triangle σ, the descending 2-cell of σ is computed. The set of triangles visited during the V-path traversals are stored in the node and p is connected, with an arc in A, to all the saddle nodes corresponding to critical edges reached by V-paths. Dually, for each minimum node p corresponding to a critical vertex v, the ascending 2-cell of v is computed. The set of vertices visited during the V-paths traversal are stored in the node and p is connected, with an arc in A, to all the saddle nodes corresponding to critical edges reached by the V-paths ending at v. Note that the descending 2-cells are stored as collections of triangles while the ascending 2-cells are stored as collections of primal vertices (and, thus, of dual 2-cells).

Let us consider the extremal graphs, i.e. the relations of the incidence graph between nodes corresponding to minima and nodes corresponding to 1-saddles (or maxima and 2-saddles). Such extremal graphs are dimension independent. In the 3D case, these two subgraphs of the MIG are computed in the same way as in 2D case. A new step is introduced to compute the saddle connectors, i.e. arcs of the MIG between 1-saddles and 2-saddles. Considering the 1-skeleton extraction of the Morse-Smale complex described in the previous section, saddle connectors are extracted in a similar fashion. All the descending 2-manifolds are extracted first, and all the traversed triangles are marked as visited. Then, starting from each critical edge e corresponding to a 1-saddle p the same process as for extracting ascending 2-manifolds is performed but only the triangles previously marked as visited are considered. This way only the separatrix V_2-pahts are traversed and, for each of them, an arc connecting the two saddle nodes is created.

The 3D instance of the Morse incidence graph has been compared to the data structure proposed in [GNP$^+$06] for encoding Morse-Smale complexes in 3D. This latter encodes the critical points (together with their geometric location) and, for each critical point p, the sets of all 3-simplexes and of all 2-simplexes, forming the (descending or ascending) 3-cell and 2-cell associated with p. Moreover, all 1-simplexes of Σ, which are the edges in the Morse-Smale complex, are maintained. The topological part of such data structure is substantially equivalent to the *topological MIG* even if the nodes corresponding to the different saddles are not organized based on the saddle they represent. In the 3D instance of the MIG, we encode only the simplexes defining the ascending and descending 3-cells associated with the minima and maxima, respectively, while the geometry of the edges in the Morse-Smale complex needs to be computed from the boundaries of such 3-cells. Thus, the *MIG* is definitely more compact even if require much more effort when the geometry of Morse-Smale 1-skeleton has to be computed.

3.4 Computing the Forman gradient on a simplicial mesh

In [RWS11], a dimension-independent algorithm is proposed for constructing a Forman gradient vector field on a cubical complex with scalar field values given at the vertices, and applications to the persistent homology computation of 2D and 3D images are presented. In [WIFF13], we have adapted such algorithm for computing a *compact gradient* on a sclara field $\mathbb{M}_\Sigma = (\Sigma, f)$ defined on a simplicial mesh.

The algorithm processes the lower star of each vertex v in Σ independently. First of all, it is convenient to require that the value of f on each vertex is distinct, so that the vertices in Σ can be ordered as:

$$g(v_1) < g(v_2) < \cdots < g(v_N).$$

To ensure unique values, g may need to be perturbed with a tie-breaking scheme.

The lower star of a vertex v is defined as follows,

$$L(v) := \big\{\sigma \in \Sigma \,|\, v \in \sigma \text{ and } g(v) = \max\{g(v_1) \,|\, v_1 \in \Sigma_0,\, v_1 \in b(\sigma)\}\big\}.$$

For each simplex σ in the lower star, the value $\max\limits_{p\in\sigma} f(p) = fmax(\sigma)$ is considered. Each cell σ is considered in ascending order of function values $fmax(\sigma)$ and of dimension, such that each cell σ is considered after its faces.

The lower stars of all vertices $v \in \Sigma_0$ form a disjoint partition of Σ. In Algorithm 1 we describe the procedure for computing a Forman gradient on lower star of each vertex of Σ. Since each lower star can be treated independently this part is well suited for a parallel implementation.

The functions used by the algorithm are:

- *lowest_edge(·)* returning the edge in the lower star with lowest function value g,
- *num_unpaired_faces(·)* returning the number of unpaired faces in the boundary of the input simplex,
- *unpaired_face(·)* returning the single unpaired face of the input simplex.

If the lower star St of vertex v is v itself, then v is a local minimum and it is added to the set C of critical cells (Algorithm 1, line 8). Otherwise, the first edge e in the order is chosen and vertex v is paired with e (Alg. 1, line 11). The star of v is processed using two queues,

- $PQone$, corresponding to the k-simplexes with one unpaired face,
- $PQzero$, corresponding to the k-simplexes with zero unpaired faces.

All the edges in the star of v different from e are added to $PQzero$ (Alg. 1, line 13). All co-faces of e are added to $PQone$ if the number of unpaired faces is equal to one (Alg. 1, line 16).

From this point the two queues control the flow of the algorithm for each vertex:

- when queue $PQone$ is not empty, the first simplex σ is removed from the queue. If the number of unpaired faces of σ has become zero, σ is added to $PQzero$. Otherwise, the vector field at the unique unpaired face $pair(\sigma)$ of σ is defined as $V(pair(\sigma)) = \sigma, pair(\sigma)$ is removed from $PQzero$ and all the co-faces, of either σ or $pair(\sigma)$, with number of unpaired faces equal to one are added to $PQone$ (Alg. 1, lines 18-27);
- when $PQone$ is empty and $PQzero$ is not empty, a simplex σ is taken from $PQzero$. Simplex σ is added to the set C of critical simplexes and all the co-faces of σ with number of unpaired faces equal to one are added to $PQone$ (Alg. 1, lines 28-33);
- when both $PQzero$ and $PQone$ are empty, then the next vertex is processed. Result of the algorithm is the set C of critical cells and the pairing of non-critical cells, which define the Forman gradient V.

Algorithm 1 Process_Lower_Star(v,Σ,V,C)

```
Require: $\Sigma$ is a simplicial complex
Require: $v$ is a vertex in $\Sigma$
Require: $V$ is a Forman gradient
Require: $C$ is a set of critical simplexes
 1:
 2: for all $v \in \Sigma_0$ do
 3:    // PQone and PQzero are two priority queues.
 4:    PQone $\leftarrow \emptyset$
 5:    PQzero $\leftarrow \emptyset$
 6:    $St \leftarrow L(v)$ // $L(v)$ is the lower star of $v$
 7:    if $St = \{v\}$ then
 8:       C.add($v$) // add $v$ to the set of minima
 9:    else
10:       $e \leftarrow$ lowest_edge(St) // extract the 1-cell with lowest function value
11:       V.add_pair($v, e$)
12:       for all 1-cells $\sigma \in St$ do
13:          PQzero.enqueue($\sigma$)
14:       for all $\sigma \in St$ do
15:          if $e \in b(\sigma)$ AND num_unpaired_faces($\sigma$)=1 then
16:             PQone.enqueue($\sigma$)
17:       while $PQone \neq \emptyset$ OR $PQzero \neq \emptyset$ do
18:          while $PQzero \neq \emptyset$ do
19:             $\sigma \leftarrow$ PQone.pop()
20:             if num_unpaired_faces($\sigma$)=0 then
21:                PQzero.enqueue($\sigma$)
22:             else
23:                V.add_pair(unpaired_face($\sigma$),$\sigma$)
24:                PQzero.remove(unpaired_face($\sigma$))
25:                for all $\tau \in St$ do
26:                   if ($\tau \in$ cb($\sigma$) OR $\tau \in$ cb(unpaired_face($\sigma$))) AND num_unpaired_faces($\tau$)=1
                      then
27:                      PQone.enqueue($\tau$)
28:          if $PQzero \neq \emptyset$ then
29:             $\sigma \leftarrow$ PQzero.pop()
30:             C.add($\sigma$)
31:             for all $\tau \in St$ do
32:                if $\sigma \in b(\tau)$ AND num_unpaired_faces($\tau$)=1 then
33:                   PQone.enqueue($\tau$)
34: return ($V, C$)
```

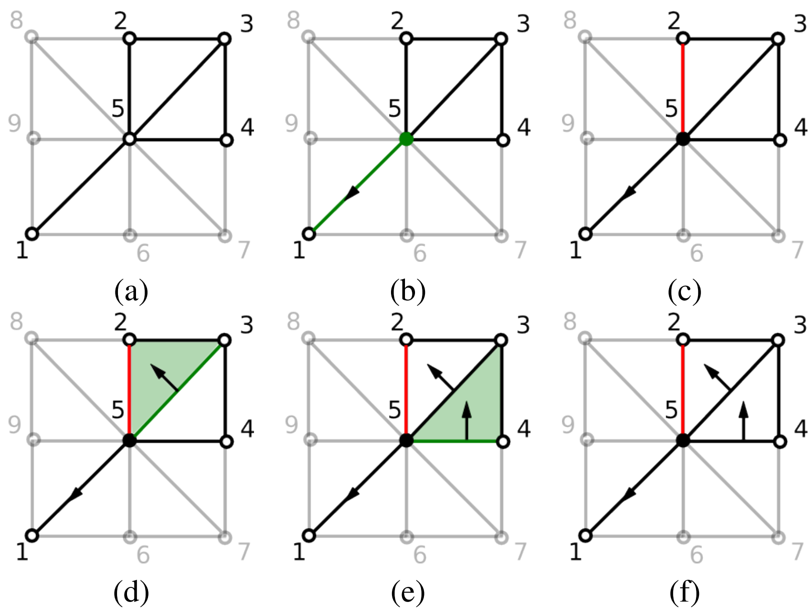


Figure 3.8: Processing of the lower star of vertex 9 using the algorithm in [RWS11].

In Figure 3.8 the main steps of the algorithm when processing the lower star of vertex 5 are illustrated. Each vertex is labeled by its scalar field value (see Figure 3.8(a)). Other simplexes are labeled based on the labels of the vertices on their boundary in lexicographic order (i.e. edge [5,2] is labeled 52). The lower star of 5 is not 5 itself, and thus 5 is not a minimum. The lowest edge starting from 5 (edge 51), is chosen to be paired with 1 (Figure 3.8(b)). All the other edges are inserted in $PQzero$ while 51 has no co-faces unpaired to be inserted in $PQone$. Since $PQone$ is empty the lowest unpaired edge, [5,2], is marked as critical and its co-face with exactly one unpaired face (triangle 532) is inserted in $PQone$ (Figure 3.8(c)). The triangle 532 is taken from $PQone$ and paired with its single unpaired face (edge 53) which is removed from $PQzero$ (Figure 3.8(d)). In a similar fashion triangle 543 is paired with the last edge 543 (Figure 3.8(e)).

From a computational point of view, algorithm 1 divides the simplicial mesh into $|\Sigma_0|$ disjoint sets called lower stars, each associated with a single vertex. The lower star $L(v)$ of each vertex v is processed independently. In order to process $L(v)$, we need two auxiliary structures $PQzero$ and $PQone$. They are priority queues and, since Algorithm 1 requires removal operations, can be implemented by using *AVL trees* to reduce time complexity. Consider a lower star $L(v)$ comprising p_v cells. As stated earlier, Process Lower Star algorithm inserts each $\sigma \in L(v)$ into $PQone$ exactly once; all cells are popped from $PQone$ one at a time. Therefore the body of the inner while loop is executed exactly p_v times. All operations are constant time except the priority queue operations which are logarithmic in the queue size. Since the sizes of $PQone$ and $PQzero$ are bounded by $p := \max_{v \in \Sigma^0} p_v$, then the running time of Process Lower Star is $O(|\Sigma_0| p \log p)$. Since that in the most of application, $p << |\Sigma_0|$, $O(p \log p)$ can be considered negligible. Therefore the complexity of the entire Process Lower Stars algorithm is $O(|\Sigma_0|)$.

3.4.1 Experimental results

We have evaluated the performances of our discrete gradient encoding and morphological feature extraction algorithms through implementations of the IA data structure. Moreover we have implemented the same algorithms for the PR-star octree and we have compared the two implementations [WIFF13]. The PR-star octree is based on the *Point Region octree (PR octree)* [Sam06], a spatial index on a set of irregularly distributed points. The domain decomposition is controlled by a single parameter, that we denote as k_v, which determines the maximum number of points indexed by a leaf node.The insertion of a new point into a *full* leaf in the tree causes the leaf to split and its indexed points to be redistributed among its children.Thus, the domain decomposition induced by a PR octree is independent of the insertion order of its points.The *PR-star octree* for a tetrahedral mesh Σ encodes the vertices and the tetrahedra of Σ and consists of:

- an array of vertices, encoding the geometry of Σ;
- an array of indexed tetrahedra, where each element is encoded in terms of the indices of its four vertices;
- an augmented PR octree, whose leaf nodes index a subset of vertices, as well as all tetrahedra in incident in these vertices.

Besides the hierarchical information associated with the octree (e.g. pointers to the parent node and to the set of children nodes),each leaf node N_l encodes: the range of indices v_{start} and v_{end} of the vertices contained in N_l;the range of indices t_{start} and t_{end} of the tetrahedra that are completely contained in N_l;and a pointer to the list of the remaining tetrahedra incident in these vertices. i.e. each such tetrahedron has at least one vertex inside and outside the domain of N_l.

The basic paradigm for performing operations on a mesh encoded as a PR-star octree is to locally process the mesh in a streaming manner by iterating through the leaf nodes of the octree.For each leaf node, a local application-dependent data structure, which we refer to as an *expanded leaf node* is generated and used to process the local geometry. After we finish processing a leaf node, we discard this local data structure and move on to the next leaf node.

Since connectivity relations are reconstructed within leaf nodes, the PR-star is ideally suited for situations in which the geometry will be processed in batches. In such cases, the connectivity reconstruction costs can be amortized over multiple mesh processing operations and a more verbose application-dependent local data structure can be utilized. We report only a single value of k_v for each dataset.In general, increasing the value of k_v reduces the overall storage requirements but increases its local storage requirements and connectivity reconstruction times.

We present experiments on five tetrahedral meshes whose sizes vary from 6 to 30 million tetrahedra. The semi-regular meshes (BONSAI, VISMALE, FOOT) were extracted from a regular grid

Table 3.2: Absolute and relative timings (in seconds) and storage costs (expressed in MBs) for the implementations based on the PR-star octree and the two version based on the IA (with and without the ET* relation explicitly stored). The first two datasets are irregular tetrahedral meshes, while the final three are irregularized tetrahedral meshes derived from regular grids.

Data set	$\|\Sigma_3\|$	k_v	**Storage**					**Timing**							
			Mesh	**Connectivity**		**Max total**		**Gradient**		**Descending**		**Ascending**		**Total**	
				tot	**%**	**tot**	**%**	**tot**	**%**	**tot**	**%**	**tot**	**%**	**tot**	**%**
F16	6.35M	IA	114	115	**–**	231	**–**	206.85	–	56.64	–	133.88	–	397.37	–
		IA$_{ET}$		189	**164**	305	**132**	212.13	**102**	23.45	**40**	133.88	**100**	369.47	**92**
		800		38	**33**	178	**77**	100.67	**49**	313.30	**549**	591.92	**442**	1005.89	**253**
SAN FERNANDO	12.4M	IA	223	225	**–**	448	**–**	358.02	–	6.90	–	260.89	–	625.81	–
		IA$_{ET}$		370	**164**	593	**134**	362.97	**101**	2.48	**29**	260.89	**100**	626.34	**100**
		800		68	**30**	329	**73**	186.27	**52**	89.85	**1286**	728.69	**279**	1004.81	**161**
BONSAI	24.4M	IA	437	445	**–**	823	**–**	732.76	–	75.23	–	147.61	–	955.60	–
		IA$_{ET}$		723	**162**	1101	**134**	748.19	**102**	15.96	**21**	147.61	**100**	911.76	**95**
		800		130	**29**	577	**70**	370.31	**50**	309.33	**412**	353.97	**240**	1033.61	**108**
VISMALE	26.5M	IA	475	484	**–**	959	**–**	796.68	–	113.75	–	217.87	–	1128.30	–
		IA$_{ET}$		786	**162**	1261	**131**	809.64	**102**	22.19	**19**	217.87	**100**	1049.70	**93**
		800		141	**29**	725	**76**	400.85	**50**	288.44	**253**	355.59	**163**	1044.88	**92**
FOOT	29.5M	IA	527	541	**–**	1068	**–**	868.29	–	138.78	–	201.60	–	1208.67	–
		IA$_{ET}$		875	**162**	1402	**131**	892.89	**103**	27.15	**19**	201.60	**100**	1121.64	**93**
		800		164	**30**	691	**65**	454.52	**52**	699.86	**504**	395.69	**196**	1550.07	**128**

using Regular Simplex Bisection [WDF11] and irregularized through a half edge collapse-based simplification process that removed approximately 15% of the vertices. We also simplified about 10% of the vertices of irregular dataset SAN FERNANDO to remove 'flat' regions (i.e. regions with very low persistence) from the mesh, yielding a more meaningful feature extraction.

Since each vertex v must reference a single (arbitrary) incident tetrahedron, we use the convention that the encoded tetrahedron either contains the edge that is paired with v in V, or v is critical. We find this tetrahedron during our gradient vector field generation. This optimization accelerates the descending 1-manifold and the ascending 3-manifold extraction steps without any impact on storage cost. We noticed room for optimization in the descending 2-manifold extraction by using a similar trick to encode the Edge-Face gradient relation. That is, for each edge e paired with a face, we encode a single tetrahedron whose gradient contains the face pointed to by e. We refer to the IA data structure with this optimization as IA_{ET}.

We have adapted the algorithm by Robins et al. [RWS11] for working with the PR-star structure. Since the PR-star octree efficiently reconstructs the local connectivity for the entire submesh indexed by a leaf node, rather than for each individual vertex (as in the IA) it is able to compute the gradient field in about half the time (see column GRADIENT in Table 3.2). For storage comparisons, we considered the *topological (connectivity) overhead* of the data structures. The IA requires about the same amount of information for connectivity as it does for the base mesh, the IA_{ET} requires about 1.6 times as much space, and the PR-star requires less than a third of the storage. (see column CONNECTIVITY in Table 3.2). In terms of overall storage requirements, which includes the base mesh, gradient, topological overhead and auxiliary data structures, the PR-star requires about 30% less storage space than the IA, while the IA_{ET} requires about 30% more space.

All data structures require a small amount of additional memory to perform feature extraction.The IA requires a global queue to perform the graph traversal of the gradient field, while the PR-star utilizes a cache of octree nodes with expanded connectivity information, as well as a list of *dangling paths* for each visited leaf node. For both data structures, this additional storage was negligible (0.01%–0.1% the size of the mesh). As can be seen in Table 3.2, the IA data structures perform best on descending manifold extraction, where the relevant topological connectivity relations are explicitly encoded. Furthermore, the ET* optimization in IA_{ET} accelerates the 2-manifold extractions, reducing the overall extraction times for descending manifolds to 20%–40% that of IA. In contrast, since the PR-star needs to explicitly reconstruct these relations, it can take several times as long to extract the descending manifolds (see column DESCENDING). The timings are significantly closer for the ascending manifold extractions (see column ASCENDING), where the connectivity relations need to be extracted for all data structures. Overall, the PR-star is the smallest data structure, but requires additional time to reconstruct the connectivity of the mesh at runtime.

Figure 3.9 illustrates features extracted from the BUCKY dataset, including the 3-cells of the MS complex, the intersection of ascending and descending cells of the Morse complexes, the 1-skeleton of the Morse-Smale complex and its combinatorial structure. Note that many arcs of the extracted 1-skeleton are shared (Figure 3.9(a)), while they are explicit in the combinatorial representation (Figure 3.9(d)). In Figure 3.10 an example of features extracted from the BONSAI dataset are shown. In Figure 3.10(a) the descending 3-cells filtered in order to show the interior of the dataset and in Figure 3.10(b) the ascending 1-cells overlapping the descending 3-cells.

3.5 Watershed and Forman based approaches: an experimental comparison.

In this section we provide an experimental comparison between the watershed by simulated immersion, described in Section 2.1.4.1, and a Forman approach [FIMS13].

The main difference between the two approaches is that the Forman approach generally requires more time since it requires the pre-computation of the Forman gradient, while the watershed approach can begin immediately with the manifold extraction. On the other hand, much more information can be easily extracted from the gradient field, *e.g.* number of critical points, not only top cells of the Morse complexes but all the cells of the Morse-Smale complex, or the incidence graph, while the same extractions based on the watershed decomposition using the MIG

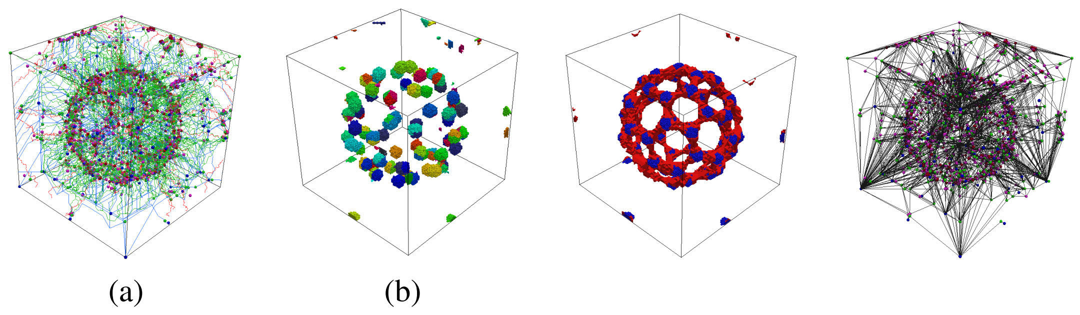


Figure 3.9: Example of features extracted from the BUCKY dataset. (a) The Morse-Smale 1-manifolds: maxima–2-saddle connectors (red), 2-saddle–1-saddle connectors (green) and 1-saddle–minima connectors (blue). (b) The Morse-Smale 3-cells, thresholded by region sizes to highlight the larger 3-cells decomposing the inner spheres. (c) The intersection of 3-cells from the ascending (blue) and descending (red) Morse complexes, filtered to highlight 3-cells decomposing the inner spheres. (d) The graph representing the combinatorial structure of the MS complex.

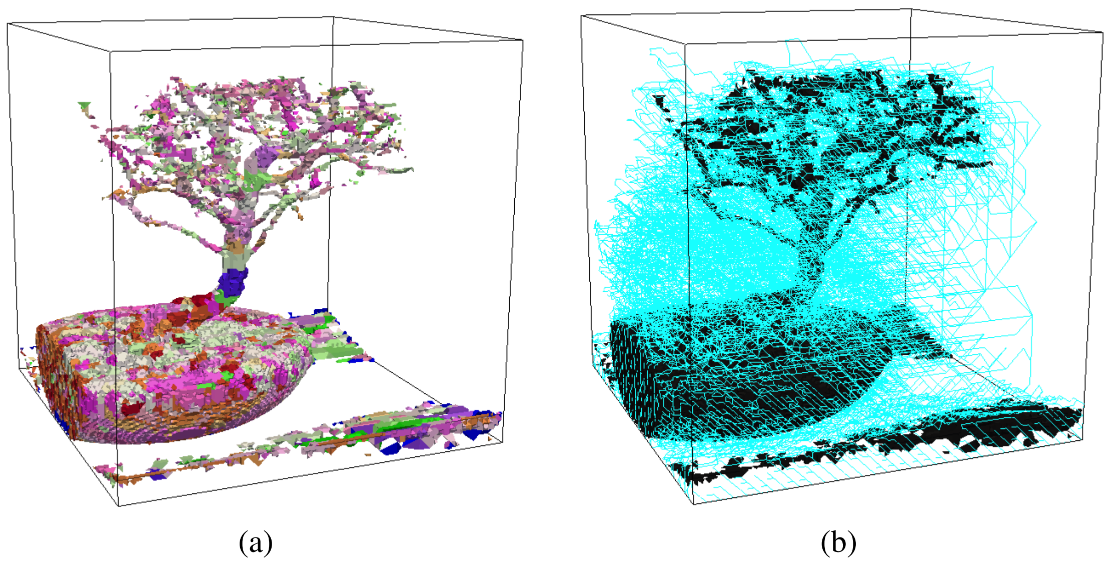


Figure 3.10: Example of features extracted from the BONSAI dataset. In (a) the descending 3-cells filtered in order to show the interior of the dataset. In (b) the ascending 1-cells overlapping the descending 3-cells.

representation are inefficient and, in many cases, incorrect, or impossible.

As discussed in Section 2.1.4.1, the watershed by simulated immersion approach computes a labeling on the vertices of a simplicial mesh Σ, on which a functions f has been defined. The labeling corresponds to the n-cells of the ascending Morse complex Γ_a. The worst-time complexity of the labeling phase is $O(|\Sigma_0|\, log|\Sigma_0|)$ due to the initial sorting of the vertices required by the algorithm. The n-cells of the descending Morse complex are computed using the same algorithm and inverting the function f. Thus, the space complexity of the output is $O(|\Sigma_0|)$.

We used the algorithm described in Section 3.4 to compute the Forman gradient ($O(|\Sigma_0|)$). Extractions from the Forman gradient have different time complexity based on the k-cell we want to extract. The worst-time complexity for extracting an ascending n-manifold is linear since each vertex is visited at most once and the space complexity of the output is $O(|\Sigma_0|)$. To obtain a labeling on vertexes also for the descending n-cells the Forman gradient is computed on the inverse of function f.

It is clear that the computation of the Forman gradient plus n-cells extraction takes more time than the same n-cell computation using the watershed approach. However, let us first consider that the processes required by Forman are easily parallelizable, providing a relevant speedup and making the overall process comparable or even faster than the simulated immersion approach. Secondly, the Forman approach has a significant advantage when we want to extract many features from our dataset and not only the top manifolds. This task can be easily performed using the single gradient pre-extraction similarly to what we have described for the top manifolds, but is much more difficult (and sometimes incorrect) using the watershed decomposition. Finally, the Morse incidence graph compactly encodes all the information for homology computation, so its efficient computation is crucial for efficient homology computation.

In the latter of this section some practical comparison on the segmentations obtained for some 2D terrain and 3D volumetric datasets generated from analytic functions and from real-world data are shown. All datasets used in our experiments satisfy the theoretical conditions that no two adjacent vertices have the same elevation. Therefore, they have no flat simplexes. For each dataset, we extract the corresponding ascending and descending decomposition using the Forman and watershed approaches and we compare them by using the Rand Index (RI) [Ran71] and the Hamming distance (HD) [HD95] metrics, two common similarity metrics used to compare segmentations and adapted in [CGF09a] to mesh data.

As described in Section 2.1.4, the simulated immersion algorithm leaves some unclassified vertices (called watershed vertices) at the boundaries between two regions. The metrics have been evaluated considering such vertices as neutral, therefore always considering them as labeled in agreement with their counterpart derived by the Forman extraction. In Table 3.3 (firsts six rows)

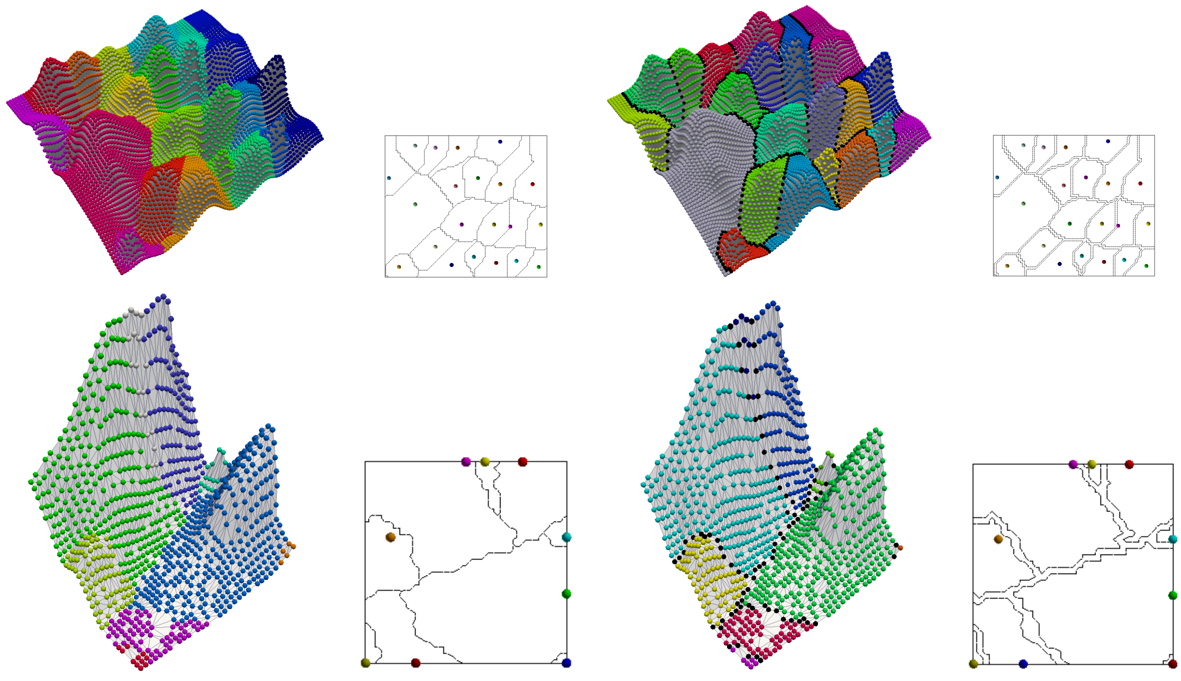

(a) (b)

Figure 3.11: Segmentations computed on the EGGS (above) and MARCY (below) obtained with the Forman approach (a) and with the watershed by simulated immersion (b) where black spheres correspond to watershed vertices. For each image on the left we shown the vertex labeling and on the right the boundaries of each region as well as their seeds. Corridors in the segmentations obtained from the simulated immersion algorithm correspond to unclassified vertices.

is shown a numerical description of the datasets as well as the number of regions computed by the two algorithms in the ascending and descending decomposition. Can be noticed that the two algorithms compute always the same number of regions and the metrics have values always greater than 0.98 for all the datasets. We cannot claim that one decomposition behaves better than the other, since sometimes ascending manifolds have higher metric values (with EGGS, for example) while in other cases the descending decomposition works better. Regarding watershed vertices, we notice that, for these datasets, they are about 1.5-9% of the total number of vertices, remarking that this number is not only influenced by the dimensionality of the dataset but also by the number of regions and how these regions intersect. Figure 3.11 shows the computed mountains for the EGGS and MARCY datasets.

Name	W. vertices		W. regions		F. regions		RI		HD	
	Asc.	Desc.	Asc.	Desc.	Asc.	Desc.	Asc.	Desc.	Asc.	Desc.
EGGS	569	567	23	21	23	21	1.00	0.99	1.00	0.99
MARCY	38	85	3	9	3	9	0.98	0.99	0.98	0.98
MALLORCA	29	88	4	4	4	4	0.99	0.99	0.98	0.99
USTICA	120	104	8	8	8	8	0.99	0.96	0.99	0.94
ANALYTIC1	4921	0	8	1	8	1	0.98	1.00	0.96	1.00
ANALYTIC2	4921	4921	8	8	8	8	0.97	0.97	0.96	0.96
BUCKY	9472	14061	178	223	178	223	0.99	0.99	0.87	0.88
FUEL	1947	13075	33	58	33	58	0.99	0.80	0.99	0.86
NEGHIP	18423	34095	88	74	88	74	0.79	0.95	0.80	0.77
SILICIUM	19503	23862	99	124	99	124	0.94	0.99	0.89	0.90

Table 3.3: Comparisons between pairs of ascending and descending segmentations found on four terrain datasets and six volume datasets. Columns describe (from left to right), dataset name (*Name*), number of vertexes (*V*) , number of triangles/tetrahedra (*T*), watershed vertices (*W. vertices*) found in the ascending (*Asc.*) or descending (*Desc.*) decomposition, number of regions found with the Watershed approach (*W. regions*), number of regions found with the Forman approach (*F. regions*), Rand index value (*RI*) and Hamming distance value (*HD*).

Once assured of the similarity between these two methods in 2D, results have been obtained for some volume datasets. Two synthetic and four real datasets has been chosen. In Table 3.3 (lasts six rows) are summarized the results obtained with these datasets. Also for the 3D case it cannot be claimed that one decomposition behaves better than the other while the percentage of the watershed vertices with respect to the total number of vertices of the dataset is about 7-20%, and up to 40% for the case of BUCKY dataset. Figures 3.12 illustrate the results obtained on the ANALYTIC1 and NEGHIP datasets.

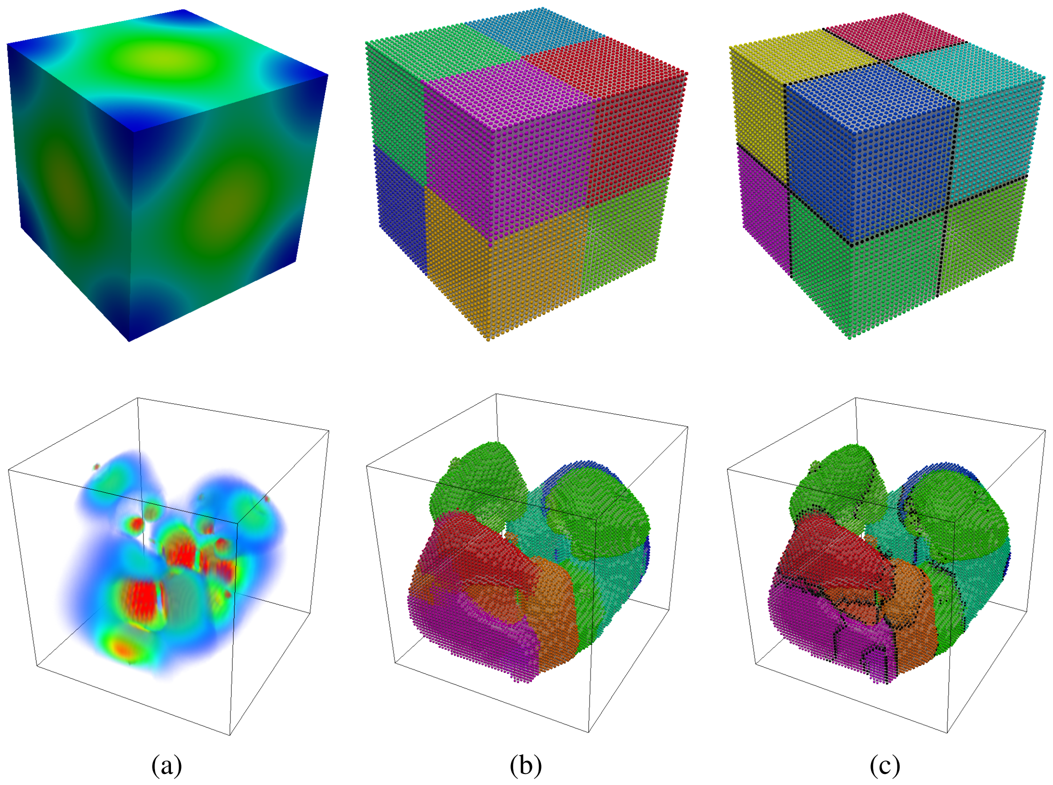


Figure 3.12: Example of the results obtained on the ANALYTIC1 dataset (above) and NEGHIP dataset (below). In (a) a visualization of the field. The labeling for the vertices obtained computing the ascending 3-cells on with the Forman approach is shown in (b) and using the Watershed by simulated immersion in (c) where black spheres indicate watershed vertices. Visualizations for the *Neghip* dataset are obtained filtering out the part of the dataset corresponding to empty air.

Chapter 4

Update Operators on a Scalar Field

A fundamental issue in the scalar field analysis is the possibility to simplify a scalar field $\mathbb{M}$. Due to the huge dimensions of the data sets and the presence of noise, a tool to eliminate the uninteresting features of $\mathbb{M}$ is fundamental. In Morse theory an operator, called *cancellation*, has been defined which removes two critical points by locally modifying the integral lines originating and converging in those two points [Mat02]. In terms of Morse complexes, a cancellation produces the removal of two Morse cells, both in the ascending and descending Morse complexes, as well as the local modifications of the incidence relations between the remaining Morse cells. A cancellation may increase the incidence relations among such cells when applied on a complex in dimension higher than two.

For this reason in [CF11], two dimension-independent simplification operators for Morse complexes have been defined, alongside with the inverse refinement operators defined as the undo of the simplification operators. These new operators constantly reduce the number of cells in the Morse and Morse-Smale complexes as well as the number of incidences among such cells. In Section 4.1 the simplification operators are introduced describing their effect on the integral lines of the scalar field and on the cells of the Morse and Morse-Smale complexes. Moreover we describe also the refinement operators, undo of the simplification operators.

In [CFI11] we have defined the updates imposed by the two simplification operators, called $removal_{i,i+1}$ and $removal_{i,i-1}$ and introduced in [CF11], on the Morse incidence graph and we have compared their behavior with the cancellation operator defined by Morse theory [CFI11, CFI13b]. Such simplification operators are completely dimension independent and their number depends on the dimension of the domain. In the n-dimensional case we have $2(n-1)$ simplification operators where $(n-1)$ are $removal_{i,i+1}$ and $(n-1)$ are $removal_{i,i-1}$. The effect of a removal and a contraction has been defined on the Morse incidence graph as the set of nodes removed, set of arcs removed and set of arcs inserted by each operation. We have also developed an implementation of two macro-operators for 1-saddle-2-saddle cancellation in the 3D case. They are implemented as a sequence of extremum-saddle operations aimed to the simplification of the neighborhood of the 1-saddle-2-saddle in order to obtain the feasibility condition. In Section 4.2 the update operators are presented, describing their effect on the incidence graph structure encoding the Morse complexes, the Morse Incidence-Graph (MIG) and in Section 4.3 we described the experimental results obtained implementing such operators.

4.1 Update operators on Morse complexes

The removal operators defined in [CF11], remove two critical points from a scalar field $\mathbb{M} = (M, f)$ modifying the integral lines in the neighborhood of the two critical points. The updates imposed on the function are similar to the one described in Section 2.2 for the cancellation.

Two critical points can be canceled, from f, if and only if the following conditions are satisfied:

- p is an $(i + 1)$-saddle,
- q is an i-saddle,
- there is a unique integral line connecting p and q.

After the i-cancellation the two critical points p and q are removed from the function and the integral lines originated or converging into them are modified as follows:

- the set of integral lines converging at p or q before the i-cancellation are transformed into a set of integral lines converging to critical points of index $j > i$ that were the destination of integral lines starting at p before the cancellation,
- the set of integral lines that originated at q or p before the i-cancellation are transformed into a set of integral lines originating at critical points v of index $k < i + 1$ that were the origin of integral lines ending at q before the cancellation.

The simplification operators are called $removal_{i,i+1}$ and $removal_{i,i-1}$ and they differs based on the critical points on which they operate.

4.1.1 Simplification operators on Morse complexes

Given a scalar field $\mathbb{M} = (M, f)$, a $removal_{i,i+1}(q, p, p')$ removes an i-saddle q and an $(i + 1)$-saddle p from f if and only if

- p and q are connected through a unique integral line;
- q is connected through integral lines at most to another $(i + 1)$-saddle p' different from p.

The effect of a removal $removal_{i,i+1}(q, p, p')$ is to delete p and q and to transform the set of integral lines converging to p, and those converging to q, into a set of integral lines converging to p'. Each critical point that was the origin of an integral line converging either to p or q becomes the origin of an integral line converging to p'. A $removal(q, p, p')$ has a specific effect on the Morse complexes computed on f.

The descending Morse complex Γ_d is modified as follow:

- i-cell q is removed from Γ_d;
- $(i + 1)$-cell p is merged into $(i + 1)$-cell p';
- all the cells $r_j \in b(p)$ are moved to the immediate boundary of p';
- all the cells $s \in cb(p)$ get p' in their immediate boundary.

The effects of the $removal_{i,i+1}(q, p, p')$ on the ascending Morse complex Γ_a are defined in a entirely dual manner:

- $(n - i)$-cell q is removed from Γ_a;
- $(n - i + 1)$-cell p is merged into $(n - i + 1)$-cell p';
- all the cells $r \in b(p)$ are moved to the immediate boundary of p';
- all the cells $s \in cb(p)$ get p' in their immediate boundary.

In Figure 4.1 the effect of a $removal_{1,2}(q, p, p')$ operator on the descending and ascending Morse complexes is shown. When applying the operator on the descending Morse complex Γ_d illustrated in Figure 4.1(a) the 2-cell p and the 1-cell q are removed from the complex. 1-cells in $R = \{r_1, r_2, r_3\}$, on the boundary of p, are moved to the boundary of p' and the 0-cells $Z = \{z_1, z_2\}$ are removed from the boundary of q. The same operator applied on the ascending

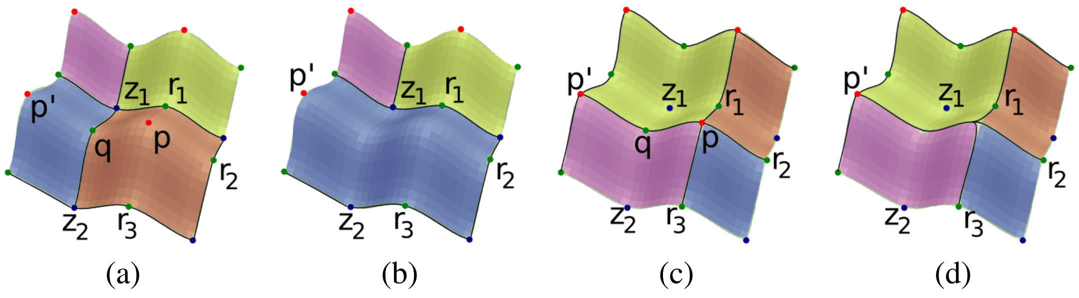


Figure 4.1: Effects of a $removal_{1,2}(q, p, p')$ applied on a descending Morse complex Γ_d before (a) and after the simplification (b). In (c) the corresponding ascending Morse complex Γ_a is shown and in (d) the effects of the same operator on Γ_a.

Morse complex Γ_a in Figure 4.1(c) has a dual effect. 0-cell p is collapsed into 0-cell p' and the 1-cell q is deleted as well. All the 1-cells $R = \{r_1, r_2, r_3\}$ are extended to p' and q is removed from the boundary of all 2-cells in $Z = \{z_1, z_2\}$.

The other removal operator, $removal_{i,i-1}(q, p, p')$ removes an i-saddle q and an $(i-1)$-saddle p if and only if

- p and q are connected through a unique integral line;
- q is connected through integral lines at most to another $(i-1)$-saddle p' different from p.

The effect of a removal $removal_{i,i-1}(q, p, p')$ is to delete p and q and to transform the set of integral lines originating at p or q into a set of integral lines originating at p'. Each critical point that was the destination of an integral line originating at p or q becomes the destination of an integral line converging to p'.

The descending Morse complex Γ_d is modified as follow:

- i-cell q is removed from Γ_d;
- $(i-1)$-cell p is merged into $(i-1)$-cell p';
- all the cells $r \in b(p)$ are moved to the immediate boundary of p';
- all the cells $s \in cb(p)$ get p' in their immediate boundary.

The effect of $removal_{i,i-1}(q, p, p')$ on the ascending Morse complex Γ_a are defined in a entirely dual way:

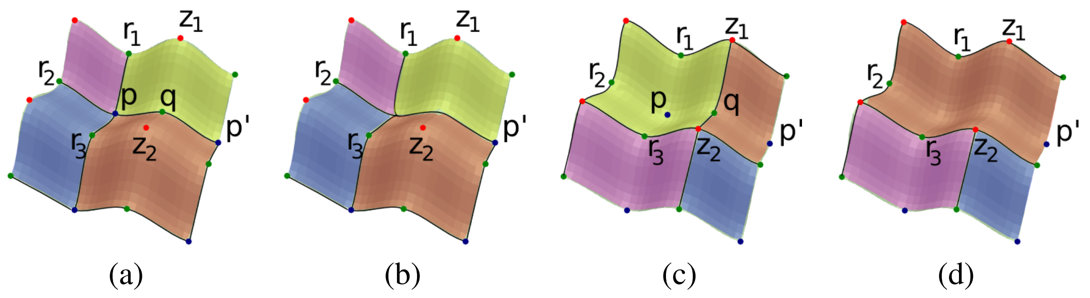


Figure 4.2: Effects of a $removal_{1,0}(q,p,p')$ applied on a descending Morse complex Γ_d before (a) and after the simplification (b). In (c) the corresponding ascending Morse complex Γ_a and in (d) the effects of the same operator on Γ_a.

- $(n-i)$-cell q is removed from Γ_a;
- $(n-i-1)$-cell p is merged into $(n-i-1)$-cell p';
- all the cells $r \in b(p)$ are moved to the immediate boundary of p';
- all the cells $s \in cb(p)$ get p' in their immediate boundary.

In Figure 4.2 the effect of a $removal_{1,0}(q,p,p')$ operator on the descending and ascending Morse complexes is shown. Applying the operator on the descending Morse complex illustrated in Figure 4.2(a) the 0-cell p is collapsed in the 0-cell p' and the 1-cell q is removed as well from the complex. The 1-cells $R = \{r_1, r_2, r_3\}$ expand their boundary to p' and the 2-cells $Z = \{z_1, z_2\}$ are removed from the boundary of q. The same operator applied on the ascending Morse complex Γ_a Figure 4.2(c) has a dual behavior. 2-cell p is merged with the 2-cell p' and p' gets all the 1-cells $R = \{r_1, r_2, r_3\}$,previously on the boundary of p, on its boundary. The 1-cell q is deleted and the incidences with the 0-cells $Z = \{z_1, z_2\}$ are removed.

It can be noted that $removal_{i,i+1}$ and $removal_{i,i-1}$ have similar effects on the ascending and descending Morse complexes. They remove cells of the same dimension based on the index of the removal and the Morse complex on which they are working. In general, a $removal_{i,i+1}$ applied on a descending Morse complex remove cells of the same dimension of a $removal_{i,i-1}$ applied on the ascending Morse complex (and vice versa).

For example, both operations showed in Figure 4.1(b) and 4.2(d) delete a 2-cell and a 1-cell from the descending and ascending Morse complex, respectively. Similarly, $removal_{1,2}$ applied in Figure 4.1(d) removes a 0-cell and a 1-cell from the descending Morse complex Γ_d as $removal_{1,0}$ applied in Figure 4.1(b) on the ascending Morse complex. In Sections 4.2 this duality will be highlighted when applying the simplification operators on the Morse Incidence Graph representation (MIG).

	$removal_{i,i+1}$		$removal_{i,i-1}$	
	Γ_d	Γ_a	Γ_d	Γ_a
R	i	n-i	i	n-i
Z	(i-1)	n-(i-1)	(i+1)	n-(i+1)
S	(i+2)	n-(i+2)	(i-2)	n-(i-2)

Table 4.1: Dimension of the cells involved in a simplification on the two Morse complexes.

4.1.2 Refinement operators on Morse complexes

Based on the simplification operators defined in Section 4.1.1, two refinement operators have been defined called $insert_{i,i+1}$ and $insert_{i,i-1}$İntuitively, an insertion is defined as the undo of the corresponding removal operator, it introduces two cells of consecutive dimension in the Morse complexes by splitting an existing cell and adapting the incidence relations among the cells in the neighborhood. The key precondition for an $insert$ to be applied is the configuration of the cells around p and q, such cells can be divided in three sets:

- $R = \{r_j, j = 1, ..., j_{max}\}$ is the set of cells that were incident in p before the simplification. They are incidence in p' after the simplification.
- $Z = \{z_h, h = 1, ..., h_{max}\}$ is the set of cells that were incident in q before the simplification but different from p and p'
- $S = \{s_k, k = 1, ..., k_{max}\}$ is the set of cells that were incident in p before the simplification and that are not incident in p' after.

These sets are composed of cells of different dimension if the simplification performed is a $removal_{i,i+1}$ or a $removal_{i,i-1}$ and depending from the Morse complex on which it is applied. Table 4.1 summarizes the dimension of the cells in sets R, S and Z when varying the operation and the Morse complex.

An $insert_{i,i+1}(q, p, p')$, inverse to a $removal_{i,i+1}(q, p, p')$, is feasible if:

- all cells set $\{p'\} \cup R \cup Z \cup S$ need to be present in the Morse complexes and
- the cells in R are cells incident in p'.

Applied on a descending Morse complex Γ_d $insert_{i,i+1}(q, p, p')$ has the following effect:

- i-cell q and $(i + 1)$-cell p are reintroduced in Γ_d,

- all i-cells in R are moved to the boundary of cell p,
- all $(i+1)$-cells in Z get p on their co-boundary,
- all $(i-1)$-cells in S are moved to the boundary of q.

The effect of $insert_{i,i+1}(q,p,p')$ on the ascending Morse complex Γ_a is entirely dual:

- the $(n-i)$-cell q and the $(n-i+1)$-cell p are reintroduced,
- all $(n-i)$-cells in R takes p on their boundary,
- all $(n-i+1)$-cells in Z are moved to the boundary of p,
- all $(n-i-1)$-cells in S get q on their boundary.

The inverse of $removal_{i,i-1}(q,p,p')$ is $insert_{i,i-1}(q,p,p')$ operator. The feasibility preconditions are the same as the preconditions of $removal_{i,i+1}(q,p,p')$:

- the cells in $\{p'\} \cup R \cup Z \cup S$ need to be in the Morse complexes and
- the cells in R are all incident to p'.

The effects of the operator on the two Morse complexes are dual with respect to the $insert_{i,i+1}$-(q,p,p'). Specifically, an $insert_{i,i-1}(q,p,p')$, applied on a descending Morse complex, inserts cells of the same dimension of an $insert_{i,i+1}(q,p,p')$ applied on an ascending Morse complex.

Applying $insert_{i,i+1}(q,p,p')$ on a descending Morse complex Γ_d the operator has the following effect:

- i-cell q and $(i-1)$-cell p are reintroduced in Γ_d;
- all i-cells in R get p on their coboundary;
- all $(i-2)$-cells in Z are moved to the boundary of p;
- all $(i+1)$-cells in S get q on their boundary.

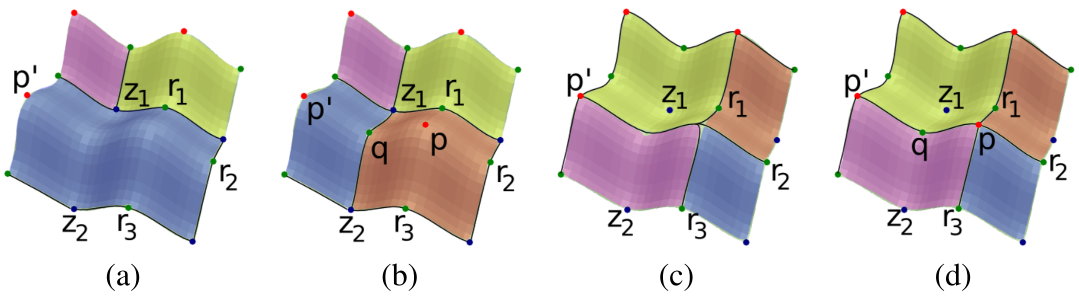


Figure 4.3: Effects of a $removal_{1,2}(q, p, p')$ applied on a descending Morse complex Γ_d before (a) and after the simplification (b). In (c) the corresponding ascending Morse complex Γ_a is shown and in (d) the effects of the same operator on Γ_a.

Figure 4.3 shows the effect of $insert_{1,2}(q, p, p')$ operator on the descending and ascendin Morse complexes. Applying the operator on the descending Morse complex illustrated in Figure 4.3(a) the 2-cell p and the 1-cell q are introduced into the complex. 1-cells in $R = \{r_1, r_2, r_3\}$, on the boundary of p', are moved to the boundary of p and the 0-cells $Z = \{z_1, z_2\}$ are added from the boundary of the 1-cell q. The same operator applied on the ascending Morse complex Γ_a in Figure 4.3(c) has a dual effect. 0-cell p is introduce aside 1-cell q. All the 1-cells $R = \{r_1, r_2, r_3\}$ incident into p' now end at p and q is inserted as part of the boundary of 2-cells in $Z = \{z_1, z_2\}$.

Figure 4.4 shows two examples of a refinement applied on a descending Morse complex Γ_d. In Figure 4.4(a) the precondition of the refinement $insert_{2,3}(q, p, p')$ is the presence of cells p' and the sets R and Z. In Figure 4.4(b) 3-cell p and 2-cell q are introduced in $\Gamma_d\dot{2}$-cells in R are removed from the boundary of p' and inserted, as well as q, on the boundary of p. 1-cells in Z instead are inserted in the boundary of q. In the descending Morse complex Γ_d showed in 4.4(c) an $insert_{1,2}(q, p, p')$ is applied. In (d) 2-cell p and 1-cell q are introduced in the complex Γ_d. 1-cells r_j are removed from the boundary of p' and redirected, as well as q, to the boundary of p. 3-cells s_k extend their boundary on p and q, extend its boundary on the 0-cells in Z.

4.2 Update operators on the MIG

Alternative definitions, for the operators described in Section 4.1, can be furnished based on the encoding used to represent the Morse complexes. In this section a new definition for the operators is given based on the Morse Incidence Graph described in Section 3.1. Since the MIG is a dual representation of both the descending and ascending Morse complexes, the two instances of the removal operator can be rewritten without distinction.

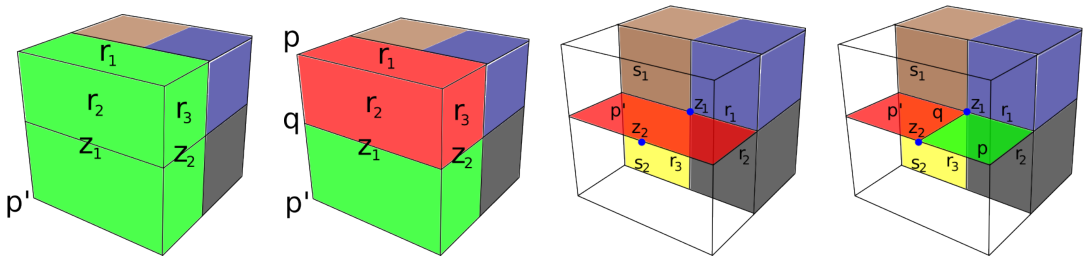


Figure 4.4: Effects of an $insert_{2,3}(q, p, p')$ applied on a descending Morse complex Γ_d before (a) and after the refinement (b). Cells occluded are not indicated with letters. In (c) the effects of an $insert_{1,2}(q, p, p')$ applied on a descending Morse complex Γ_d before (c) and after (d) the refinement.

A $remove(q, p, p')$ simplifies a MIG $G = (N, A, \psi)$ $A \subset N \times N$, removing two nodes p and q from N and removing the arcs connected with either p or q from A. New arcs are reintroduced in A.

Considering the nodes, in the neighborhood of the two removed nodes, they can be divided in three sets:

- $R = \{r_j, j = 1, ..., j_{max}\}$ is the set of nodes incident to p before the simplification of the same dimension of q,
- $Z = \{z_h, h = 1, ..., h_{max}\}$ is the set of nodes incident to q but different from p and p',
- $S = \{s_k, k = 1, ..., k_{max}\}$ is the set of nodes incident to p and different from r_j.

4.2.1 Simplification operators on the MIG

Formally a $remove_{a,b}(q, p, p')$, with either $a = b + 1$ or $a = b - 1$, is feasible on a MIG $G = (N, A, \psi)$ if:

- node q is connected to, at most two, nodes of the same dimension, p and p',
- the label of the arc (p, q) is 1 $(\psi((p, q)) = 1)$.

Applying $remove_{a,b}(q, p, p')$ on G, let us define

- $A^- = \{(p, q)\} \cup \{(p', q)\} \cup \{(p, r_j)|r_j \in R\} \cup \{(p, s_k)|s_k \in S\} \cup \{(q, z_h)|z_h \in Z\}$ the set of arcs removed and

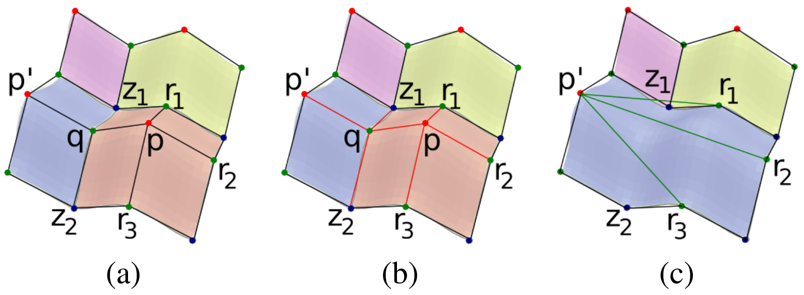


Figure 4.5: Effects of the $remove_{1,2}(q, p, p')$ applied on the 2D MIG showed in (a). In (b) the red arcs are removed from the MIG and in (c) the green arcs are introduced. Blu dots correspond to minima, green dots to saddles and red dots to maxima.

- $A^+ = \{(p', r_j) | r_j \in R\}$ the set of arcs inserted.

Then $remove_{a,b}(q, p, p')$ produces a simplified graph $G = (N', A', \psi')$ where,

- $N' = N \setminus \{p, q\}$
- $A' = A \setminus A^- \cup A^+$
- $\psi'(p', r_j) = \psi(p', q) \cdot \psi(p, r_j) + \psi(p', r_j)$

The labels of the arcs different from (p', r_j) remain unchanged.

Figure 4.5 shows an example of a $remove_{1,2}(q, p, p')$ applied on the Morse Incidence Graph illustrated in 4.5(a). In 4.6(b) the arcs connecting p with nodes in the set $R = \{r_1, r_2, r_3\}$ are removed as well as arcs connecting q to nodes in the set $Z = \{z_1, z_2\}$ and arcs (p, q) and (p', q);p and q are deleted as well. In 4.6(c) three new arcs are inserted in the graph connecting nodes in set $R = \{r_1, r_2, r_3\}$ with p'.

In Figure 4.6(a) the same $remove_{1,2}(q, p, p')$ is applied on the MIG extracted from a 3D synthetic scalar field. In Figure 4.6(b) the arcs connecting p with nodes in $R = \{r_1, r_2, r_3\}$ and in $S = \{s_1, s_2\}$ are removed. Arcs connecting q to nodes in $Z = \{z_1, z_2\}$ are removed as well as arcs (p, q) and (p', q). After removing nodes p and q, in Figure 4.6(c) p' is connected with the nodes $R = \{r_1, r_2, r_3\}$.

4.2.2 Refinement operators on the MIG

The refinement operators $insert_{i,i+1}(q, p, p')$ and $insert_{i,i-1}(q, p, p')$ can be formalized as a unique operator working on the MIG, the $insert_{a,b}(q, p, p')$ operator undo of the $remove_{a,b}(q, p, p')$.

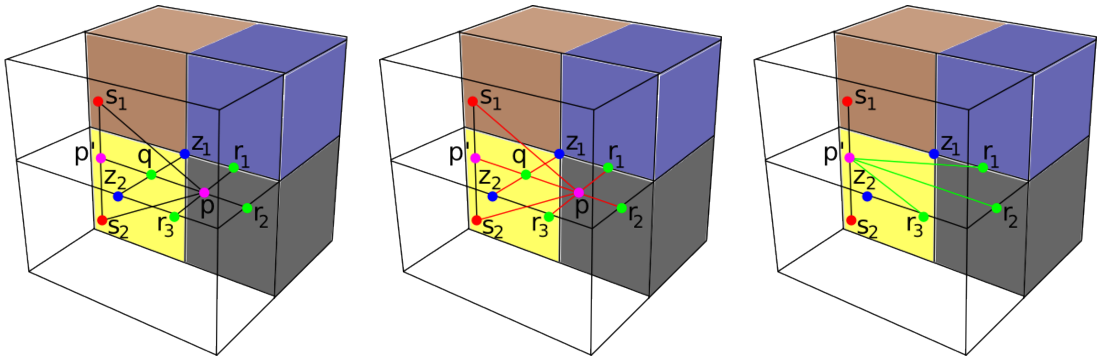


Figure 4.6: Effects of the $remove_{1,2}(q, p, p')$ applied on the 3D MIG showed in (a). In (b) the red arcs are removed from the MIG and in (c) the green arcs are introduced. Blue dots correspond to minima, green dots to 1-saddles, purple dots to 2-saddles and red dots to maxima.

The feasibility condition for a $insert_{a,b}(q, p, p')$ operator is the presence of nodes $p' \cup R \cup Z \cup S$ in G and if the label $\psi(p', r)$ is greater than or equal $\psi'(p', q) \cdot \psi'(p, r)$.

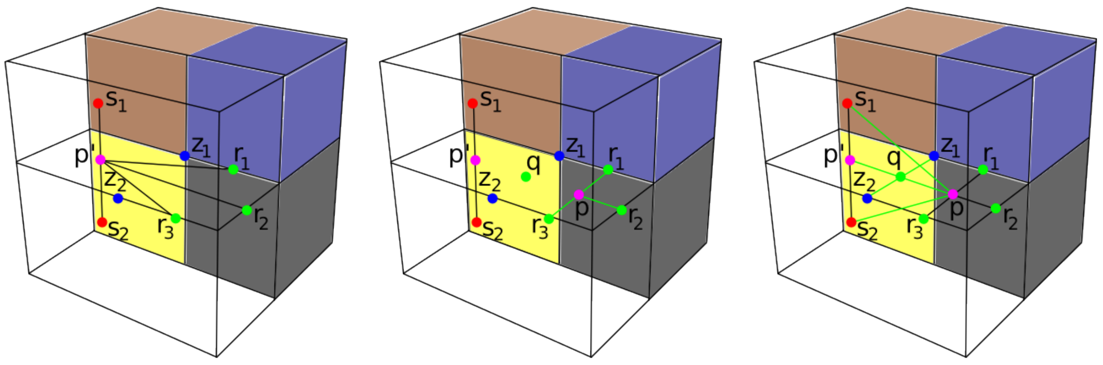
Then, applying a feasible $insert_{a,b}(q, p, p')$ let us define:

- $A^- = \{(p', r_j)|r_j \in R\}$ the set of arcs removed;
- $A^+ = \{(p, q)\} \cup \{(p', q)\} \cup \{(p, r_j)|r_j \in R\} \cup \{(p, s_k)|s_k \in S\} \cup \{(q, z_h)|z_h \in Z\}$ the set of arcs inserted.

$insert_{a,b}(q, p, p')$ applied on G produce a refined graph $G' = (N', A', \psi')$ where:

- $N' = N \cup \{p, q\}$
- $A' = A \setminus A^+ \cup A^-$
- $\psi'(p', r_j) = \psi(p', r_j) - \psi'(p', q) \cdot \psi'(p, r_j)$

In Figure 4.7 is shown an example of an $insert_{1,2}(q, p, p')$ applied on the 3D MIG illustrated in Figure 4.7(a). In Figure 4.7(b) the two nodes p and q are introduced in the graph. The three arcs connecting p' with nodes in $R = \{r_1, r_2, r_3\}$ are deleted and three new arcs are created connecting p with nodes in R. In Figure 4.7(c) the new arcs are introduced connecting q with nodes p, p' and with nodes in $Z = \{z_1, z_2\}$ and connecting node p with nodes $S\{s_1, s_2, s_3\}$.



(a) (b) (c)

Figure 4.7: Effects of the $insert_{1,2}(q, p, p')$ applied on the 3D MIG shown in (a). In (b) the green arcs are redirected from p' to p in the MIG and in (c) the green arcs are the newly introduced. Blu dots correspond to minima, green dots to 1-saddles, purple dots to 2-saddles and red dots to maxima.

4.3 Experimental Results

In this section we will described our experimental results obtained on scalar fields $\mathbb{M}_\Sigma = (\Sigma, f)$ defined on triangular and tetrahedral meshes.

In Section 4.3.1 we will present a persistence based simplification algorithm we have defined for simplifying a MIG computed on $\mathbb{M}_\Sigma$. In Section 4.3.2 we will present results obtained comparing i-$cancellation$ and $remove$ operators for simplifying scalar fields in 3D.

4.3.1 Simplification on the MIG

Working with scalar fields in dimensions higher than two, it can be showed that, at any time, the Morse complexes will admit a number of feasible i-$cancellation$ greater or equal than the number of feasible $remove(q, p, p')$. In particular,

- an i-$cancellation$ is feasible for any p and q simply connected
- a $remove(q, p, p')$ is feasible for any p and q simply connected only if q is connected to, at most, another cell p' different from p.

Thus, in the 3D case two macro-operators has been defined to increase the number of viable simplifications at any time. These macro-operators are implemented as a sequence of extremum-saddle operators followed by a saddle-saddle removal operator. We will describe the implementation of such macro-operators in terms of updates on the MIG for seek of clarity even if a similar definition can be done in terms of Morse complexes and Forman gradient.

The first macro-operator collapse a 2-saddle p and a 1-saddle q into another 2-saddle p'. For all the 2-saddles t_j connected with q a $remove_{2,3}(t_j, p_1, p'_1)$ is performed to eliminate the 2-saddles different from p and p'. When p and p' are the only 2-saddles connected with q the $remove_{1,2}(q, p, p')$ is performed.

Dually, the second macro-operator collapse a 1-saddle p and a 2-saddle q into another 1-saddle p'. For all the 1-saddles t_j connected with q a $remove_{1,0}(t_j, p_1, p'_1)$ is performed to eliminate the 1-saddles different from p and p'. When p and p' are the only 1-saddles connected with q the $remove_{2,1}(q, p, p')$ is performed.

Then, an experimental evaluation of the macro-operators has been performed in [CFI11].The simplification algorithm developed simplify Morse complexes in arbitrary dimensions using $remove_{i,i+1}$ and $remove_{i,i-1}$. A *persistence* value is associated with any $remove$ operator by considering the function values of the two critical points p and q deleted by the operator. Intuitively, the *persistence* of a pair of critical points measures the importance of the pair and is equal to the absolute difference in function values between the two points [EHZ01]. The objective of the simplification algorithm is to reduce the size of the Morse complex by removing critical points which are due to the presence of noise or which are not relevant for the need of a specific application. Simplification is also applied when the size of the original Morse complex is too large for the computation resources available.

The simplification algorithm starts by computing all feasible simplifications, evaluates their persistence and inserts them in an ordered queue in increasing order of persistence.

At each step, a simplification is removed from the queue and applied to the current MIG. The process terminates when either a certain number of simplifications has been performed or when a specified value of persistence is reached.

Algorithm 4.3.1 shows the pseudo-code description of the simplification algorithm in the case when the stopping condition is determined by a persistence threshold. The $SimplifyGraph$ function requires two input parameters:

- a float value t indicating the threshold for the simplifications. All the simplifications performed will have persistence value associated lower than t,
- a MIG $G = (N, A, \varphi)$ composed by nodes N and arcs A.

As initialization, all the arcs are pushed inside a priority queue Q (rows 3-4) and organized in ascending order of persistence value. The persistence value associated to each arc is computed as the absolute value of the difference between the function values of the vertices associated to each node incident a. Each simplification s in Q is considered if the associated persistence value is lower than t (row 7). If s is a feasible simplification ($removal_{i,i+1}$ or $removal_{i,i+1}$) or a simplification valid for a macro-operator, it is performed and all the new arcs generated are introduced in Q (row 9-12). Otherwise, if s is not valid it is ignored, while if it has a persistence valued equal or higher than t the algorithm ends.

Algorithm 2 SimplifyGraph(t,Q,G)

```
Require: $t$ is a float value indicating a threshold;
Require: $G = (N, A)$ is a $MIG$ with nodes $N$ and arcs $A$;
 1: $Q = \emptyset$ // an empty priority queue
 2: // All the arcs are pushed into the priority queue based on their persistence value;
 3: for all arcs $a \in A$ do
 4:    if isFeasibleSimpl($a$) then
 5:       $Q.push(a)$;
 6: while $Q.isNotEmpty()$ do
 7:    $s \leftarrow Q.pop()$ // The next simplification $s$ is popped from the queue;
 8:    if $s.persistence() < t$ then
 9:       // If $s$ has persistence value lower than $t$
10:       if isFeasibleSimpl($s$) then
11:          $a_{new}$ = simpl($s$); // $a_{new}$ are the new arcs created by the simplification;
12:       else if isFeasibleMacro($s$) then
13:          $a_{new}$ = macro_operator($s$); // $a_{new}$ are the new arcs created by the macro-operator;
14:       $Q \leftarrow a_{new}$;
15:    else
16:       break;
```

Different thresholds on persistence value has been used: 1% of the max persistence value for light noise removal, 10% for stronger noise removal, and 20% or greater for consistently reducing the complexity of the MIG. The storage cost of the simplified MIG using these three different thresholds is equal to 95%, 65% and 35% of the cost of the MIG at full resolution.

Then the distribution of the saddle-saddle operator among all the simplifications is studied. Such kind of simplifications are likely to be performed early in the simplification process. If the simplification algorithm is based on persistence, this means that a large number of arcs will be introduced in the MIG early in the simplification process, influencing both the efficiency (speed) of the algorithm and its versatility (the number of feasible simplifications). This result underlines the importance of having an efficient operator for simplifying saddles.

4.3.2 Removal operator and cancellation

The cancellation operator described in Section 2.2 is a simplification operator defined in Morse theory [Mil63]. Comparing the i-cancellation to the simplification operators described in the previous sections it can be showed that there is a specific relation between an i-cancellation and a $remove$ operator.

In general,

- a $remove_{n-1,n}(q, p, p')$ remains equivalent to a (n-1)-$cancellation(q, p)$;
- a a $remove_{1,0}(q, p, p')$ remains equivalent to a 0-$cancellation(q, p)$.

The i-cancellation involving only saddles is more complicated (as described in Section 2.2) and generally different from a remove operator involving saddles. To highlight the differences between the two operators we will compare their effects on the same MIG.

In the 2D case, a $remove_{1,2}(q, p, p')$ is the same as the maximum-saddle 1-cancellation(q,p) and the $remove_{1,0}(q, p, p')$ is the same as the minimum-saddle 0-cancellation. In the 3D case the relations become more various. The $remove$ operators involving an extremum still remain the same, thus:

- $remove_{2,3}(q, p, p')$ is equivalent to a 2-$cancellation(q, p)$
- $remove_{1,0}(q, p, p')$ is equivalent to a 0-$cancellation(q, p)$.

As an example, let us consider the $1 - cancellation$ of a 1-saddle and a 2-saddle in 3D. The 1-$cancellation(q, p)$ of 1-node q and 2-node p is feasible on the MIG $G = (N, A, \psi)$ if nodes q and p are connected, and the label of arc (p, q) is 1 ($\psi(p, q) = 1$).

Let $G' = (N', A', \psi')$ be the graph after 1-$cancellation(q, p)$. The effect of 1-$cancellation(q, p)$ consists of deleting nodes p and q, as well as all the arcs incident in nodes p and q, and adding one arc for each pair (r_j, t_l) where r_j belongs to R and t_l belongs to set T, representing the set of $(i + 1)$-nodes different from $(i + 1)$-node p and connected to q.

Thus, the 1-$cancellation$ operator deletes two nodes from N, but possibly increasing the number of arcs connecting 1-nodes to 2-nodes in the graph by deleting $|R| + |T| + 1$ of such arcs, but adding $|R| * |T|$ of them. Thus, it is not a simplification operator, since it does not reduce the size of the graph. In [GBHP11], this issue has been discussed at length, since it can cause

Name	N Simpl	Nodes	Arcs	Cost (MB)	Time (sec.)
NEGHIP		3K	11512	0.17	-
i-cancellation	1200	700	2621	0.04	0.68
remove	1200	700	2395	0.03	0.62
HYDROGEN	-	23K	65961	1.0	-
i-cancellation	7000	9K	35123	0.53	25.1
remove	7000	9K	23091	0.35	17.86
BUCKY	-	46K	157984	2.4	-
i-cancellation	7000	32K	128231	1.95	73.5
remove	7000	32K	84487	1.23	33.4
ANEURISM	-	125K	1015724	15.49	-
i-cancellation	10000	105K	748192	11.41	233.28
remove	10000	105K	435910	6.65	70.54
VISMALE	-	900K	3588570	54.75	-
i-cancellation	10000	880K	3513889	53.61	37.12
remove	10000	880K	3107124	47.41	9.43
FOOT	-	1550K	7178384	109.5	-
i-cancellation	45000	1460K	6137199	93.64	2882.1
remove	45000	1460K	5413683	82.6	1187.3

Table 4.2: Comparison of *cancellation* and *remove* operators.

computational problems and, more importantly, make the application of $i\text{-}cancellation$ operator unfeasible on large-scale data sets. Several strategies are proposed in [GBHP11], which aim at postponing an $i\text{-}cancellation$ that would introduce a number of arcs greater than a predefined threshold, or vertices with valence greater than a predefined threshold. On the contrary, the $remove$ operator always reduces the size of the graph. In [CFI13b] the macro-operators has been compared with the cancellation implementing a 3D version of the simplification algorithm, described in Section 4.3.1, based on cancellation. In this case, the number of critical nodes is reduced at each step, but not necessarily the number of arcs.

In Table 4.2, the results obtained by comparing the $remove$ operator with the $cancellation$ operator are shown. For each 3D data set, in the first row illustrates the number of nodes and arcs in the full resolution MIG. In the second and third rows, are shown the statistics related to $cancellation$ and $remove$ operators, respectively: the number of simplifications applied, the number of nodes and arcs in the simplified MIG, the cost of the data structure encoding the MIG (in MB), and the time (in sec) needed to perform the simplifications.

The number of arcs in the graph simplified with $cancellation$ always exceeds the number of arcs in the graph simplified with the same number of $remove$. Such behavior influences the efficiency of the whole algorithm, doubling the time needed to manage and enqueue a larger number of arcs (and thus, a greater number of possible simplifications) for large data sets.

When the data set is small and the number of simplifications is high compared to the total number of nodes the two methods are quite similar (NEGHIP). With the growth of the dimension of the data set the two methods start to differ (HYDROGEN): by using $remove$ we can get a 20% more compressed MIG in about half the time than by using $cancellation$. In particular, the $remove$ operator is particularly useful in the first simplifications performed on a data set (simplifications that can be interpreted as noise removal). On many data sets we have noticed that by using $cancellation$ the number of arcs remains approximately the same while by using $remove$ their number immediately decreases (VISMALE). In general, the cost of the MIG is reduced by 10% to 20% by using $remove$ instead of $cancellation$ and the same number of simplifications can be performed in half the time.

Chapter 5

Multi-resolution Models for Morse Complexes

Due to the complexity of real data sets, such as terrain models or volumetric scalar fields, the investigation of hierarchical methods to control and adjust the *Level of Detail* (*LOD*) of a given dataset is an active research area. A *multi-resolution model* (or *LOD model*) permits to obtain different representations of an object at different levels of detail. The level of detail can be uniform or vary other the object.

Multi-resolution approaches can be subdivided into two categories: *geometry-based* approaches, where the approximation is guided by the refinement of the geometrical shape, and *wavelet-based* approaches, where the multi-resolution behavior is determined by the space of functions.

Multi-resolution models for geometrical objects support representation and processing of spatial entities at different levels of detail [FMP97, FMP99, DFKP05]. Such representations are especially interesting because of their potential impact on applications, such as terrain modeling in Geographic Information Systems (GIS) and scientific data visualization. The basis for a multi-resolution geometric model of a shape is the decomposition of the shape into simple elements, called *cells*. The *accuracy* of a cell complex in representing a shape depends on the size, number, and density of its cells: a parameter that we call the *resolution* of the complex. A high resolution, and thus a high number of cells, is needed to produce accurate descriptions. On the other hand, a *maximum accuracy* is not always required in the whole shape, but a sufficient high accuracy for the specific application task can be achieved by *locally adapting* the resolution of a complex in different parts of the shape, thus reducing processing costs and memory space.

The basic ingredients of a multi-resolution model for a spatial object are a *base complex*, that defines the coarsest representation of the object, a set of *updates* that provide variable resolution representation of the base complex when applied to it, and a *dependency relation* among updates which allows combining them to extract consistent intermediate representations. The basic operation in a multi-resolution model is called *selective refinement* and consists of extracting a mesh satisfying some application-dependent criterion based on level of detail, such as approximating a surface or scalar field with a certain accuracy.

The process of building a multi-resolution model depends on the simplification of a cell complex. Usually, such operation is time consuming because sophisticated techniques to optimize the shape of the cells and to bound the approximation error are required. However, in a multi-resolution model, such operations are performed off-line, in order to build the structure which can be queried efficiently on-line, according to some application-dependent parameters.

Several multi-resolution models for cell complexes have been proposed in the literature [Mag98, FMP99, DFKP05]. Most existing models are designed for specific applications, or classes of applications (rendering, terrain modeling, ...). Existing models usually rely on a specific construction technique: such models can be obtained only from an initial cell complex by applying a *specific type* of transformation operator. Most proposed models are a direct abstraction of the data structure used to implement them. We will introduce here a multi-resolution model for the analysis of a scalar field $\mathbb{M}_\Sigma = (\Sigma, f)$, which encodes the Morse and Morse-Smale complexes computed on $\mathbb{M}$.

In Section 5.1, we introduce the multi-resolution model consisting of a hierarchy of combinatorial representations of descending and ascending Morse complexes in the form of an MIG. Note that the model is geometry-based since extracts representations of cell complexes, Γ_d and Γ_a, at different levels of detail but the simplicial mesh Σ, discrete domain of f, remains unchanged.

We call this model a *Multi-Resolution Morse Incidence Graph ($MMIG$)* [CDFI13], and we define it in terms of refinement modifications on the MIG. In Section 5.1.1 we describe two different encodings for an $MMIG$, called the *implicit* and *explicit Multi-resolution Morse Incidence Graph* ($MMIG$). In Section 5.1.3 experimental results obtained implementing such two data structures are shown and we compare the 2D instance of the $MMIG$ with existing proposal for hierarchical topological representations of 2D scalar fields.

5.1 The multi-resolution Morse incidence-graph

A *Multi-resolution Morse Incidence Graph* ($MMIG$) is a multi-resolution model representing the topology of the two Morse complexes as well as the 1-skeleton of the Morse-Smale complex at different level of details [CFI12].

An $MMIG$ is generated from the MIG representing the two Morse complexes at full resolution by iteratively applying the simplification operator $remove$ discussed in Section 4 (see Figure 5.1). Simplifications are applied according to their value of persistence, and the graph obtained as a result of the simplification sequence is the coarsest representation of the two complexes. We denote such coarse MIG as $G_B = (N_B, A_B, \varphi_B)$, and we call it the *base graph*. We denote as

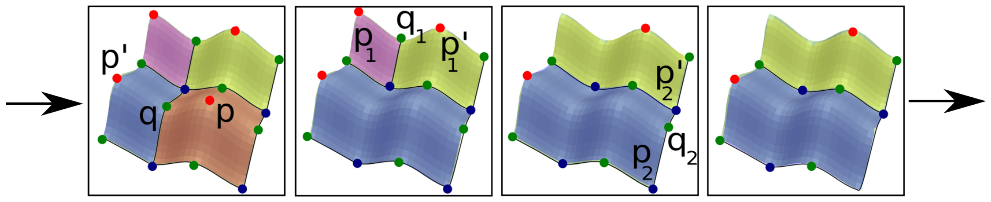


Figure 5.1: A sequence of simplifications, $remove_{i,i+1}(q, p, p')$, $remove_{i,i+1}(q_1, p_1, p'_1)$, *remove*$_{i,i-1}(q_2, p_2, p'_2)$, applied on a descending Morse complex to produce a coarser representation. Red dots correspond to 2-cells, green dots to 1-cells and blue dots to 0-cells.

$\mathcal{S}$ the set of *remove* simplifications that generates G_B from G. An $MMIG$ is a compact way of representing the coarse graph G_B plus the inverse of the simplification modifications applied in the generalization process. The objective is to allow an effective and efficient extraction of approximate representations of Morse complexes at uniform or variable levels of detail. The level of detail is, in this case, determined by the persistence values that we require in the extracted complexes.

The basic ingredients of an $MMIG$ are:

- an MIG $G_B = (N_B, A_B, \varphi_B)$ representing the two Morse complexes, Γ_d and Γ_a at the coarsest resolution,
- a set $\mathcal{M}$ of refinements, inverse to the simplifications in $\mathcal{S}$,
- a dependency relation between refinements in $\mathcal{M}$, which defines a partial order relation on $\mathcal{M}$.

A refinement modification is defined as $\mu = insert(q, p, p')$. Recall that $insert(q, p, p')$ is specified by the nodes q and p it introduces, and by the nodes in $N^- = \{p'\} \cup Z \cup S \cup R$, as defined in Section 4.1. Intuitively, refinement modification μ depends on all refinements $\mu*$ which introduce nodes belonging to N^-, and on the modification μ_0 creating the base graph G_B, in case some node in N^- belongs to G_B.

Let us denote as $\mu = (G^-, G^+)$ a refinement modification on the MIG, where

- $G^- = (N^-, A^-, \varphi^-)$ is the set of nodes N^- and arcs A^- involved in the refinement μ
- $G^+ = (N^+, A^+, \varphi^+)$ is the resulting set of nodes N^+ and arcs A^+ when refinement μ has been applied

Let us consider the set $\mathcal{M}$ of all refinements and the creation of the coarse representation G_B as a dummy refinement that we denote as μ_0 (i.e., μ_0 generates G_B). Note that $\mu_0 = (\emptyset, G_B)$, i.e., for μ_0, G^- is a null graph (without nodes), and $G^+ = G_B$.

We can define over set $\mathcal{M}$ plus μ_0 a *dependency* relation as follows.

Definition 5.1.1. *Refinement modification $\mu_1 = (G_1^-, G_1^+)$,* directly depends *on refinement modification $\mu_2 = (G_2^-, G_2^+)$, if and only if*

- *μ_2 creates one or more nodes in G_1^-; or*
- *μ_2 creates one or more arcs in G_1^-.*

From an $MMIG$, a large number of complexes at intermediate resolution can be obtained by applying a sequence $\mathcal{U} = [\mu_0, \mu_1, .., \mu_k]$ of refinement modifications in $\mathcal{M}$ to the base graph G_B. Applying a sequence of refinements $\mathcal{U} = [\mu_0, \mu_1, .., \mu_k]$, the first refinement μ_0 is applied to G_B obtaining the graph $G_{(1)}$. All the refinements μ_i in $\mathcal{U}$ are applied to the graph $G_{(i)}$ obtained from the refinement $\mu_{(i-1)}$ in the sequence.

In a generic sequence a node can be created and deleted several times and this creates cycles in the relation of dependency. However, in the $MMIG$, a node is introduced by one modification only and never deleted. Thus, any sequence of modifications is always *non-redundant*.

If all the possible sequences of refinement modifications are *non-redundant* the dependency relation is a partial order relation since a node or an arc is never introduced twice by the modifications in $\mathcal{M}$. An $MMIG$ is thus a triple $(G_B, \mathcal{M}, \mathcal{R})$, where $\mathcal{R}$ denotes the direct dependency relation defined above. We can alternatively define the dependency relation between refinement modifications only in terms of the nodes of the graphs defining those refinements.

Proposition 5.1.1. *Refinement modification $\mu_1 = (G^-, G^+)$ directly depends on refinement modification $\mu_2 = (G_2^-, G_2^+)$, if and only if μ_2 creates one or more nodes in G^-, i.e., if and only if $\{p_2, q_2\} \cap N^- \neq \emptyset$.*

Proof. ($\Rightarrow$) Let refinement μ_1 directly depend on refinement μ_2. We need to show that if μ_2 creates an arc in A^-, then it creates also a node in N^-. From the definition of a refinement modification, we know that if a refinement creates an arc, it creates at least one of its incident nodes. Refinement μ_2 creates an arc in A^-, thus it creates at least one if its incident nodes. Nodes incident in an arc in A^- are node p', and a node in R. Thus, refinement μ_2 creates a node in $\{p'\} \cup R \subseteq N^-$.

($\Leftarrow$) If a refinement μ_2 creates a node in N^-, then refinement μ_1 directly depends on refinement μ_2 by definition. $\square$

Definition 5.1.2. Dependency relation *between refinement modifications is the transitive closure of the direct dependency relation.*

In Figure 5.2 an $MMIG$ built from the sequence of simplifications shown in Figure 5.1 is illustrated (for the seek of clarity, we are showing only the descending Morse complex and not

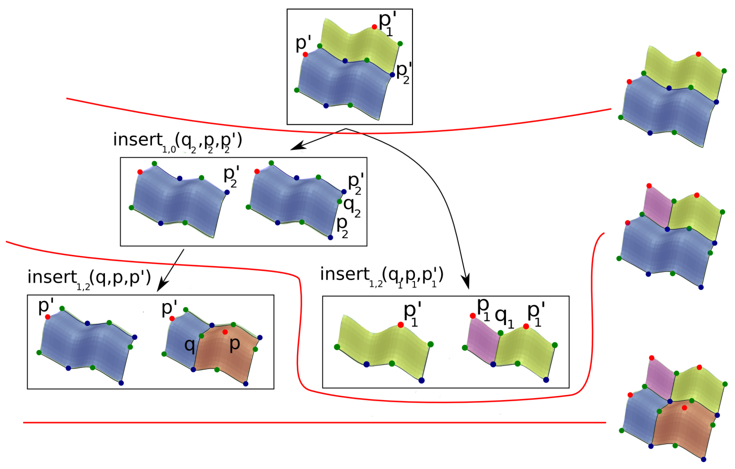


Figure 5.2: $MMIG$ built from the sequence of simplifications on a descending Morse complex Γ_d illustrated in Figure 5.1. The root refinement correspond to the base complex Γ_B. Remaining nodes indicate refinement operations. Each node represents only the incidence relations among cells in the immediate boundary/coboundary of the cells involved in the refinement. Arrows represent the dependency relation, red points correspond to 2-cells, green points to 1-cells and blue points are 0-cells.

the MIG representation). The base complex Γ_B is indicated as the root of the model while the remaining nodes store the refinement operations. An arrow connects two nodes when there is a direct dependency relation among them.

Figure 5.2 shows the dependency relations among three different refinements and the root. To perform $insert_{1,2}(q, p, p')$ a specific set of cells are required in the boundary and co-boundary of the newly introduced cells p and q. The dependency relation with refinement $insert_{1,2}(q_2, p_2, p'_2)$ assures that such cells q_2 will be in the immediate boundary of cell p once $insert_{1,2}(q, p, p')$ is performed. Refinement $insert_{1,2}(q_1, p_1, p'_1)$ instead is independent from other dag nodes since all the cells required are already in the base complex.

Given a multi-resolution, a sequence of modifications has to satisfy specific properties related to the dependency relation among the modifications of the model.

Definition 5.1.3. *A refinement modification $\mu_i = (G^-, G^+)$, is* **feasible** *on a graph $G_{(i)}$ if and only if the graph $G_{(i)}$ contains all the nodes in G^-.*

Then, from a set of feasible modifications, the notion of feasible sequence is introduced.

Definition 5.1.4. *A sequence $U = [\mu_0, \mu_1, \mu_2, ..., \mu_m]$ of refinement modifications in $\mathcal{M}$ is* feasible *if μ_0 is feasible on G_B and each modification μ_i, $1 \leq i \leq m$ is feasible on a graph $G_{(i)}$*

obtained from the base graph G_B by applying on it a sequence $[\mu_1, .., \mu_{i-1}]$.

Definition 5.1.5. *Let $U = [\mu_0, \mu_1, \mu_2, ..., \mu_m]$ be a feasible sequence of refinement modifications in $\mathcal{M}$. The* front graph *G_U associated with sequence U is the graph obtained from the base graph G_B by applying on it the sequence of refinements $[\mu_1, \mu_2, ..., \mu_m]$.*

The front graph is a MIG representing the two Morse complexes at an intermediate level of detail. If a feasible sequence U contains all refinements in $\mathcal{M}$, then the front graph G_U associated with U is the same as the MIG at full resolution.

This means that a refinement μ is feasible on a graph $G = (N, A, \varphi)$ if and only if all nodes in N^- are contained in N ($N^- \subseteq N$). If the nodes in N^- are in N, then graph G^- that defines the feasibility of refinement μ is a subgraph of graph $G = G_U$.

Recall that dependency relation $\mathcal{R}$ is a partial order relation, and thus it defines a closure operator on the set $\mathcal{M}$ of refinement modifications. We denote a closed set of such refinement modifications as $\mathcal{U}$. The set $\mathcal{U}$ implicitly defines an incidence graph representing an approximation of the original Morse complexes.

Definition 5.1.6. *Let $(G_B, \mathcal{M}, \mathcal{R})$ be an $MMIG$, and let $\mathcal{U} = \{\mu_0, \mu_1, \mu_2, ..., \mu_m\}$ be a set of refinement modifications in $\mathcal{M}$. The set $\mathcal{U}$ is* closed *with respect to the dependency relation $\mathcal{R}$ if for each i, $1 \leq i \leq m$, each refinement modification on which refinement μ_i depends is in $\mathcal{U}$.*

Proposition 5.1.2. *If a sequence $U = [\mu_0, \mu_1, \mu_2, ..., \mu_m]$ of refinement modifications in $\mathcal{M}$ is feasible, then the set $\mathcal{U} = \{\mu_0, \mu_1, \mu_2, ..., \mu_m\}$ is* closed *with respect to the dependency relation $\mathcal{R}$.*

Proof. Let $U = [\mu_0, \mu_1, \mu_2, ..., \mu_m]$ be a feasible sequence of refinement modifications in $\mathcal{M}$, and let $\mu_i \in U$. Since $\mu_i = (G_i^-, G_i^+)$ is feasible, each refinement modification μ_j that introduces a node in N_i^- (on which refinement μ_i directly depends) is in $U_{i-1} = [\mu_0, \mu_1, .., \mu_{i-1}]$. This argument can be repeated iteratively on modifications μ_j. This implies that all the refinement modifications on which μ_i depends have a position $p < i$ in the sequence U, and the set $\mathcal{U} = \{\mu_0, \mu_1, \mu_2, ..., \mu_m\}$ is closed with respect to the dependency relation $\mathcal{R}$. □

Definition 5.1.7. *Let μ_1 and μ_2 be two feasible refinement modifications on a graph G. We say that the refinement modifications μ_1 and μ_2 are* interchangeable *if a sequence (μ_1, μ_2) of refinements (consisting of μ_1 followed by μ_2) on graph G produces the same simplified graph G' as a sequence (μ_2, μ_1) (consisting of μ_2 followed by μ_1).*

Proposition 5.1.3. *Two independent refinement modifications μ_1 and μ_2 are interchangeable.*

Proof. Let μ_1 and μ_2 be two feasible independent refinements. Let us first consider the sequence (μ_1, μ_2) applied on G. The effect of μ_1 is to delete arcs $(p_1', r_{1,j})$, insert nodes q_1 and p_1 and arcs (q_1, p_1), (q_1, p'_1), $(q_1, z_{1,h})$, $(p_1, s_{1,k})$ and $(p_1, r_{1,j})$, thus creating graph $G_1 = (N_1, A_1, \varphi_1)$, $N_1 = N \cup \{q_1, p_1\}$, $A_1 = A \backslash A_1^- \cup A_1^+$. The effect of μ_2 on graph G_1 is to delete arcs $(p'_2, r_{2,j})$ (which are also arcs in G since μ_2 is feasible on G, and p_1 and q_1 are not in N_{p_2,q_2}) and insert nodes q_2 and p_2 and arcs (q_2, p_2), (q_2, p'_2), $(q_2, z_{2,h})$, $(p_2, s_{2,k})$ and $(p_2, r_{2,j})$, thus creating graph $G_{1,2} = (N_{1,2}, A_{1,2}, \varphi_{1,2})$. Here, $N_{1,2} = N_1 \cup \{q_2, p_2\}$ and $A_{1,2} = A_1 \backslash A_2^- \cup A_2^+$.

Similarly, we denote as $G_2 = (N_2, A_2, \varphi_2)$, $N_2 = N \cup \{q_2, p_2\}$, $A_2 = A \backslash A_2^- \cup A_2^+$ the MIG obtained from $MIG\ G = (N, A, \varphi)$ by applying refinement μ_2, and as $G_{2,1} = (N_{2,1}, A_{2,1}, \varphi_{2,1})$ the MIG obtained from $MIG\ G_2$ by applying refinement μ_1. Then we have that $N_{2,1} = N_2 \cup \{q_1, p_1\}$ and $A_{2,1} = A_2 \backslash A_1^- \cup A_1^+$.

Since both μ_1 and μ_2 are feasible on G, arcs in $A_1^- \cup A_2^-$ are in A. From the definition of a refinement it follows that $A_1^- \cap A_1^+ = \emptyset$, and similarly $A_2^- \cap A_2^+ = \emptyset$. Independence of μ_1 and μ_2 implies that $A_1^- \cap A_2^+ = \emptyset$ and $A_2^- \cap A_1^+ = \emptyset$. Also $A_1^- \cap A_2^- = \emptyset$ (if each arc in A_1^- is labeled by the inverse simplification of the refinement μ_1, and similarly for A_2^- and μ_2). Finally, $A_1^+ \cap A_2^+ = \emptyset$, since each arc in A_1^+ has either q_1 or p_1 as one endpoint, each arc in A_2^+ has either q_2 or p_2 as one endpoint, and $\{q_1, p_1\} \cap \{q_2, p_2\} = \emptyset$.

We can conclude that $N_{1,2} = N \cup \{q_1, p_1, q_2, p_2\} = N_{2,1}$ and $A_{1,2} = A_1 \backslash A_2^- \cup A_2^+ = (A \backslash A_1^- \cup A_1^+) \backslash A_2^- \cup A_2^+ = A \backslash (A_1^- \cup A_2^-) \cup (A_1^+ \cup A_2^+) = A_{2,1}$, thus showing that two independent refinements are interchangeable. □

In Figure 5.2 for example, refinement $insert_{1,0}(q_2, p_2, p'_2)$ and $insert_{1,2}(q_1, p_1, p'_1)$ are interchangeable.

Proposition 5.1.4. *Two interchangeable refinement modifications μ_1 and μ_2 are independent.*

Proof. If the refinements μ_1 and μ_2 are interchangeable, then they are both feasible on the MIG $G = (N, A, \varphi)$, i.e., both G_1^- and G_2^- are subgraphs of G, and q_1, p_1, q_2 and p_2 are not nodes in N. This implies that q_1 and p_1 are not in N_2^- and similarly q_2 and p_2 are not in N_1^-. Thus, refinements μ_1 and μ_2 are independent. □

Definition 5.1.8. *Let $(G_B, \mathcal{M}, \mathcal{R})$ be an $MMIG$, and let $U = [\mu_0, \mu_1, \mu_2, ..., \mu_m]$ be a feasible sequence of refinement modifications in $\mathcal{M}$. A permutation $\mu_0, \mu_{i1}, \mu_{i2}, ..., \mu_{im}$ of the refinement modifications in U is* consistent *if the sequence $V = (\mu_0, \mu_{i1}, \mu_{i2}, .., \mu_{im})$ is a feasible sequence of refinements in $\mathcal{M}$.*

Proposition 5.1.5. *Let $U = [\mu_0, \mu_1, \mu_2, ..., \mu_m]$ be a feasible sequence of refinement modifications in $\mathcal{M}$, and let sequence $W = [\mu_0, \mu_{i1}, \mu_{i2}, .., \mu_{im}]$ be obtained from U through a consistent permutation of refinements in U. Then, the front graph G_U associated with sequence U is the same as the front graph G_W associated with sequence W.*

Proof. A permutation that defines W starting from U is consistent if each refinement μ_{ij} is feasible in sequence W. This means that each refinement μ_{ik} on which μ_{ij} depends has a position

$p < ij$ in V. Thus, a permutation defining W from U is a composition of adjacent transpositions of two independent refinements (composition of permutations obtained by reversing the order of two consecutive refinements). For each such transposition, the associated front graph before and after the transposition remains unchanged. Thus, the front graph G_W associated with sequence W is the same as the front graph G_U associated with sequence U. □

A closed subset $\mathcal{U}$ of refinements can be applied to the base $MIG\ G_B$ in any total order U that extends the partial order, producing an $MIG\ G_U$ at an intermediate resolution. An $MMIG$ encodes a collection of all representations of Morse complexes at intermediate levels of detail which can be obtained from the base representation G_B by applying a closed set of modifications on G_B.

From an $MMIG$ it is thus possible to dynamically extract representations of the topology of an n-dimensional scalar field in terms of the Morse complexes at *uniform* and *variable* resolutions. The basic query for extracting a single-resolution representation from a multi-resolution model is known as *selective refinement*.

A *selective refinement query* on an $MMIG$ consists of extracting from it an MIG with the minimum number of nodes, satisfying some application-dependent criterion. This criterion can be formalized by defining a Boolean function τ over all nodes of an $MMIG$, such that the value of τ is $true$ for all the modifications which satisfy the criterion, and $false$ otherwise. An MIG $G = (N, A, \varphi)$ obtained from a sequence of modifications U is said to satisfy a *criterion* if all the modifications in the model that assumes the value $true$ for τ are in U. Thus, a selective refinement query consists of extracting from the $MMIG$ an intermediate graph of minimum size that satisfies τ. Equivalently, it consists of extracting a minimal closed set $\mathcal{U}$ of modifications from $\mathcal{M}$ such that the corresponding complex satisfies τ. Such closed set of modifications uniquely determines a front graph, which is obtained from the base graph $G_B = (N_B, A_B, \varphi_B)$ by applying to it all modifications from $\mathcal{U}$ in any order that is consistent with the partial order defined by the dependency relation.

In this context, the Boolean criterion τ is defined based on persistence (see Section 2.2). Thus, a persistence value is assigned to a refinement modification μ that introduces a given pair of nodes p and q. We say that a pair (p,q) satisfies Boolean criterion τ if the persistence value associated with the refinement introducing p and q is greater than some prescribed value ρ.

We can have queries at *uniform resolution*, when we extract a topological representation in which all the refinements performed have a persistence value greater than a predefined threshold value, or at *variable resolution*, when we request a value of persistence which varies in different parts of the domain. We have implemented a depth-first algorithm for performing selective refinement. The algorithm starts from the coarse $MIG\ G_B$ and recursively applies to it all refinements μ_i which are required to satisfy the error criterion. In order to apply a new modification μ, all its preceding modifications in the partial order need to be applied before μ.

It can easily be proven that the result of a selective refinement algorithm is a graph G_U with minimal number of nodes, which contains all nodes satisfying criterion τ. The three red lines shown in Figure 5.2 indicate three representations extracted from the original descending Morse complex encoded in the root of the $MMIG$.

5.1.1 Encoding a multi-resolution Morse incidence graph

In this section we describe two dimension-independent encodings for the multi-resolution Morse incidence graph [CFI12]. The data structure includes the base graph G_B, the set of refinements modifications and the dependency relations among them, represented through a Direct Acyclic Graph (DAG). The simplicial mesh Σ decomposing the domain of the scalar field is encoded through the IA data structure.

The two data structures, called *implicit* $MMIG$ and *explicit* $MMIG$, differentiate in the way the nodes involved in each refinement are encoded. In the explicit MMIG we encode all the nodes involved in each refinement explicitly. This results in a efficient reconstruction of the sets R, S and Z required by each refinement (see Section 4). On the other side, we tried to reduce the memory consumption of the whole data structure designing a more compact representation for sets of nodes. The two encodings should represent the usual tradeoff between memory consumption and runtime efficiency.

Both *implicit* and *explicit* $MMIG$ store the base graph G_B in their root encoded as an MIG (see Section 3.1). Each node of the $MMIG$ represents a refinement modification $\mu = insert(q, p, p')$; μ encodes information about the refinement as well as its dependencies from the other nodes.

The information stored for each node μ, in both the implicit and explicit representations, are:

- the index of the vertices q and p in the data structure encoding the underlying simplicial mesh Σ;
- the pointer to the $MMIG$ node corresponding to the modification μ' introducing p';
- the persistence value representing the resolution level of μ;
- a flag indicating the type of refinement (which can be $insert_{i,i+1}$, or $insert_{i,i-1}$), and the value of i (in 3D it can be equal to 1 or 2, while in 2D i is always equal to 1);
- two integers indicating where to split the list of cells (0-cells or n-cells), corresponding to the Morse top cell associated with p', to correctly initialize the cell associated with p; this only happens when p and p' are nodes corresponding to extrema.
- an array, called $ancestors$, containing the pointers to the parent of μ
- an array, called $descendants$, containing the pointers to the children of μ.

These information identify the refinement operator represented by the $MMIG$ node μ.

Both our data structures encoding the $MMIG$ are built by starting from the $MIG\, G = (G, N, \varphi)$ describing the Morse complexes at the finest resolution. A sequence of simplifications is applied on G in increasing order of persistence until there are no more feasible simplifications, or the size of the resulting MIG is below a predefined threshold. Information about all the simplifications performed are saved in a stack. Each element of the stack represents a generic $remove_{i,i+1}$(or $remove_{i,i-1}$) and stores all the nodes of G involved in the simplification. Once there are no more feasible simplifications, the base graph G_B at the coarsest resolution, is stored in the root of the DAG. To perform the refinement correctly we need to represent the set of nodes involved in the refinement (sets R, S and Z). On their representation the two data structures are different.

In the *explicit data structure*, nodes in R, S and Z are encoded explicitly for each $MMIG$ node in the structure. During the building process, a new node in the $MMIG$ is created for each element in the stack. The type of the node is initialized based on the type of the simplification operator, and also the nodes p and q deleted by $remove$ operator and the pointer to the third node p' are encoded. Then, all nodes in sets Z, R and S are explicitly referred to a pointer (4 bytes per node) and one list of pointers is stored for each set R, Z and S. The resulting structure allows for a faster application of the refinement modifications since, for each modification, the sets R, Z and S are ready to be used. The resulting storage cost for the nodes in sets R, S and Z in the explicit data structure is equal to $4|RSZ|$ bytes, where $|RSZ|$ indicates the number of nodes in sets R, S and Z.

On the contrary, the MIG nodes encoded in the implicit data structure try to represent nodes in the sets R, Z and S in a implicit manner referring to the $MMIG$ nodes introducing them or the base graph if present in G_B from the beginning. Intuitively, each node in the front graph G_U, on which the refinement μ will be performed, has been inserted in G_U from another refinement μ_i or it was in the base graph G_B.

To implicitly refer the nodes introduced by another modification, in each $MMIG$ node μ a two-bit-flag vector pq-$ancestors$ with the same size as $ancestors$ is defined. Let us consider the refinement $\mu = insert(q, p, p')$ depending from a refinement $ancestors[j] = \mu_1 = insert(q_1, p_1, p'_1)$,

- $pq - ancestors[j] = 0$ if μ depends from node p_1 introduced by μ_1
- $pq - ancestors[j] = 1$ if μ depends from node q_1 introduced by μ_1
- $pq - ancestors[j] = 2$ if μ depends from both p_1 and q_1.

Thus, vector $pq - ancestors[j]$ offers a compact way to refer nodes introduced by other $MMIG$ nodes. When the nodes to refer to are in the base graph G_B, three bit vectors are stored in $MMIG$ node μ. Let k the dimension of node q introduced by $\mu = insert(q, p, p')$:

- bit vector B_Z, has a length equal to the number of $(k-1)$-nodes in G_B,

- bit vector B_R, has a length equal to the number of k-nodes in G_B,
- bit vector B_S, has a length equal to the number of $(k-1)$-nodes in G_B.

For each bit vector (i.e., B_Z), the j-th bit flag is equal to 1 if the j-th $(i-1)$-node of G_B is in Z for modification μ. B_R and B_S are similarly defined. Vector pq-$ancestors$, bit vectors B_Z, B_R and B_S, and array $ancestor$ provide an implicit encoding of sets R, Z and S required from modification μ. These latter sets are reconstructed when applying the modification in the extraction phase.

The *implicit* data structure is built, similarly to the *explicit* one, by starting from the $MIG\ G = (G, N, \varphi)$ describing the Morse complexes at the finest resolution. A stack of simplifications is constructed iteratively by applying the $remove$ operators in a sequence. Once there no more feasible simplifications are feasible, the base graph G_B at the coarsest resolution is stored in the root of the $MMIG$.

For each element in the stack, a new node in the $MMIG$ is created, the type of the node is initialized based on the type of the simplification operator. Also nodes p and q deleted by $remove$ as well as the pointer to the third node p' are encoded. Then, the following process is repeated for all nodes in sets Z, R and S. For each node p_1, belonging to any of these sets:

- if p_1 belongs to G_B, we set its corresponding bit in the bit vector (B_Z, B_R or B_S) to 1;
- otherwise we store in $ancestors$ a pointer to the modification μ_1 in the $MMIG$ that introduces p_1 and we insert in the corresponding position in pq-$ancestors$ the value 0,1 or 2 depending on whether μ depends on first, second or both MIG nodes introduced by μ.

The storage cost for each $MMIG$ node in the implicit data structure depends on the number of nodes in sets R, S and Z and on the number of nodes in the base graph G_B. Specifically, for each $MMIG$ node, it requires:

- 1 bit for each node in N_B, with a total cost of $\frac{1}{8}|N_B|$ bytes, where $|N_B|$ is the total number of nodes of G_B,
- 2 bits for each node in $R \cup S \cup Z$, total cost $\frac{1}{4}|RSZ|$ byte, where $|RSZ|$ is the cardinality of set $R \cup S \cup Z$.

Thus, to encode such nodes in the implicit data structure $\frac{1}{8}(|G_B| + 2|RSZ|)$ bytes are required. Then, comparing the two storage costs, the implicit data structure is more efficient in terms of memory requirements when, approximating the number of nodes in the base graph, $|G_B| < 30|RSZ|$.

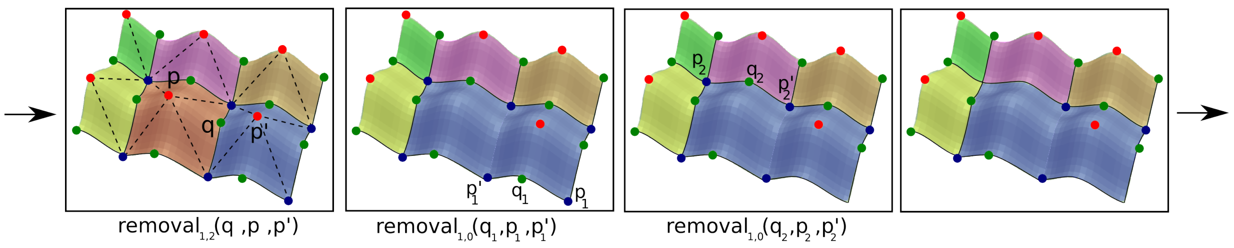


Figure 5.3: (a) 2D descending Morse complex, dotted lines delineate diamonds associated to 1-saddles; (b) After $remove_{1,2}(q, p, p')$; after $remove_{1,0}(q_1, p_1, p'_1)$ and (c) after $remove_{1,0}(q_2, p_2, p'_2)$.

5.1.2 Comparison with other approaches

As mentioned in Section 2.3, to the extent of our knowledge, the only approaches proposed in the literature for hierarchical representation of Morse or Morse-Smale complexes are for 2D scalar fields. We compare the 2D instance of the $MMIG$ with two hierarchical representations for Morse-Smale complexes in [BHEP04] and in [BPH09].

The two approaches are based on the cancellation operator [EHZ01, Mat02] which in 2D reduces to a $remove$ operator. It eliminates a saddle and a maximum ($remove_{1,2}$), or a saddle and a minimum ($remove_{1,0}$). In both [BHEP04] and [BPH09], a dependency relation between refinements is defined in terms of a *diamond*. The diamond associated with a refinement $insert_{1,2}(q, p, p')$ is a quadrangle z_1, p, z_2, p', where z_1 and z_2 are the two (not necessarily distinct) minima connected to 1-saddle q (see Figure 5.3 (a)). Dually, in the diamond associated with a refinement $insert_{1,0}(q, p, p')$, z_1 and z_2 are the two maxima connected to q. The dependency relation is defined as follows: two refinements are dependent if the associated diamonds have at least one vertex in common [BHEP04], or if they share an edge [BPH09]. The dependency relation in [BPH09] is clearly less restrictive and creates less dependency than the one in [BHEP04].

We compare the two approaches with the $MMIG$. The dependency relation defining the $MMIG$ is less restrictive than the one in [BHEP04]. If a refinement $insert(q, p, p')$ depends on a refinement $insert(s, t, t')$ in the $MMIG$, then the associated diamonds share a vertex in $\{s, t\} \cap N_{q,p} \neq \emptyset$. There is no containment relation between the dependency relation in the $MMIG$ and the one in [BPH09] in the sense that we cannot say which one is more restrictive than the other.

Figure 5.3 shows a sequence of simplifications consisting of $remove_{1,2}(q, p, p')$, $remove_{1,0}$-(q_1, p_1, p'_1) and $remove_{1,0}(q_2, p_2, p'_2)$. Let $\mu_1 = insert_{1,0}(q_1, p_1, p'_1)$, $\mu_2 = insert_{1,0}(q_2, p_2, p'_2)$ and $\mu_3 = insert_{1,2}(q, p, p')$ be the inverse refinements. The diamonds associated with saddles q (refinement μ_3) and q_2 (refinement μ_2) have one common vertex p, and the diamonds associated with saddle q (refinement μ_3) and q_1 (refinement μ_1) have two common vertices, p'_1 and p'. Refinement μ_3 directly depends on μ_1, and does not depend on μ_2, in the approach in [BPH09].

Dataset name	MIG cost	$MMIG$ cost	$MMIG$ nodes	$MMIG$ building
EGGS	0.003	0.002	38	0.08
MATTERHORN	0.17	0.38	2110	3.17
MONVISO	0.17	0.33	2236	2.8
MONT BLANC	0.21	0.41	2472	4.37
LAKE MAGGIORE	0.32	0.7	4783	9.57
BUCKY	0.05	0.07	394	4.42
HYDROGEN	0.37	0.65	1437	77.5
VISMALE	3.6	3.5	3584	791.2
ANEURISM	0.6	9.1	6362	1803.6

Table 5.1: Experimental results obtained by using implicit data structure for representing the $MMIG$ of 2D and 3D data sets. The storage costs are expressed in Megabytes and the times in seconds.

In the $MMIG$, refinement μ_3 depends on μ_2, since 1-cell q_2 is on the immediate boundary of 2-cell p' (1-node q_2 is in the set R for $\mu_3 = insert_{1,2}(q, p, p'_2)$), while μ_3 does not depend on μ_1, since none of the cells p_2 and q_2 is on the immediate boundary or the immediate co-boundary of p or q.

5.1.3 Experimental results

Experiments have been performed on the $MMIG$ to estimate the overhead of the *implicit* data structure implementing the $MMIG$ with respect to storing the MIG at full resolution, the cost of generating an $MMIG$ and the time required for extracting adaptive representations from an $MMIG$. All the experiments have been performed on a 3.2GHz processor with 2.0Gb memory. In Appendix A a description of the datasets used as well as a description of their origin can be found.

In Table 5.1 results are shown on the size of the $MMIG$ and on construction times for terrain (first five datasets) and volume data sets (last four datasets). In the first two columns (MIG cost and $MMIG$ cost), the storage costs of the MIG at full resolution and of the $MMIG$ (encoded as *implicit MMIG*) are shown, in the third column (DAG nodes) the number of nodes in the $MMIG$ is shown. The fourth column ($MMIG$ building) shows the time required for building the $MMIG$ from the MIG at full resolution. It can be noticed from these results that the $MMIG$ encoding structure requires often twice the space than the data structure encoding the MIG, but for some data sets (EGGS in 2D, VISMALE in 3D), the storage cost of the $MMIG$ is the same as the one of the MIG. The variation of these values depends on the complexity of the intermediate MIGs encountered during the simplification step. The more the degree of connectivity of the graph the more incidence relations are encoded in the $MMIG$ augmenting the storage cost.

Dataset	Implicit Data Structure		Explicit Data Structure	
name	cost	time	cost	time
Eggs	0.002	0.01	0.003	0.01
Matterhorn	0.38	2.85	0.47	2.65
Monviso	0.33	2.2	0.45	2.09
Mont Blanc	0.41	4.03	0.56	4.02
Lake Maggiore	0.7	8.68	0.81	8.05
Bucky	0.07	1.19	0.12	0.0
Hydrogen	0.65	12.2	1.46	7.0
VisMale	3.5	206.7	12.7	118
Aneurism	9.1	1472.9	32.0	1300

Table 5.2: Comparison between the implicit and explicit data structures in the 2D and 3D case. Storage costs are expressed in Megabytes and times in seconds.

We evaluate here the gain in query time and the loss in storage space when using the explicit data structure instead of the implicit one discussed (Section 5.1.1). In Table 5.2, results are shown for 2D and 3D data sets. The first two columns show the storage cost (in MB) of the $MMIG$ and the time (in seconds) required to extract the MIG at full resolution from the $MMIG$ implemented with the implicit data structure. The second two columns show the same data for the explicit one. The explicit data structure requires 20-30$\%$ more memory with respect to the implicit one in 2D, but differences are higher in the 3D case, where the explicit data structure requires three times more memory then the implicit structure. On the other hand, the direct access to the nodes in sets Z, S and R for each modification μ speeds up the navigation inside the explicit data structure, and thus we obtain lower (in 3D considerably lower) extraction times.

In Tables 5.3 and 5.4, some results are shown obtained by applying the selective refinement algorithm for extracting representations at different levels of persistence applied to 2D and 3D data sets, using the implicit data structure. We show also the ratio between the storage cost of the extracted MIG with respect to the MIG at full resolution to give some insight on the complexity of the result and on the extraction times. In our experiments, we have seen that more than 20% of the extraction time is due to associating the geometry of the descending and ascending cells to the extracted maxima and minima, respectively. As it can be noticed from Table 5.3, time needed to extract representations at variable resolutions can be relevant. We can distinguish between two main objectives when extracting a representation at different level of details:

- to reduce the complexity of the MIG, for further computations, by reducing the detail where not needed without loss of detail in the interesting area;
- to localize details on the MIG for visualization.

Considering the extraction times presented, the first point can greatly benefit from the $MMIG$ since further computations can be very challenging on the original MIG. Algorithms computing

Name	Pers.	Extracted MIG	Time
Eggs	-	100 %	0.01
	10	20%	0
	4	63%	0
	1	85%	0
MatterHorn	-	100%	2.85
	500	32%	0.8
	100	54%	1.3
	1	96%	2.02
Mont Blanc	-	100%	4.03
	200	22%	1.0
	5	46%	2.1
	1	88%	3.6
Monviso	-	100%	2.2
	200	18%	0.03
	5	50%	0.9
	1	78%	1.8
Lake Maggiore	-	100%	8.68
	1500	24%	2.3
	5	48%	4.32
	1	82%	7.02

Table 5.3: Selective refinement on 2D data sets: in the first column, we show the persistence value; in the second column, we show the ratio between the storage cost of the extracted MIG and that of the MIG at full resolution, in the third column, we show extraction times.

Dataset name	Persistence	Extracted MIG	Time (sec.)
Hydrogen	-	100%	12.2
	150	9%	0.02
	130	70%	7.1
	120	90%	8.0
Bucky	-	100%	1.19
	1000	27.8%	0.01
	250	60%	0.5
	50	98%	0.93
Aneurism	-	100%	1472.9
	250	21%	120.2
	200	79%	1325.5
	0.1	98%	1451.1
VisMale	-	100%	206.7
	150	13%	30.1
	130	40%	150.4
	1	89%	176.1

Table 5.4: Selective refinement on 3D data sets: in the first column, we show the persistence value; in the second column, we show the ratio between the storage cost of the extracted MIG and that of the MIG at full resolution; in the third column, we show extraction times.

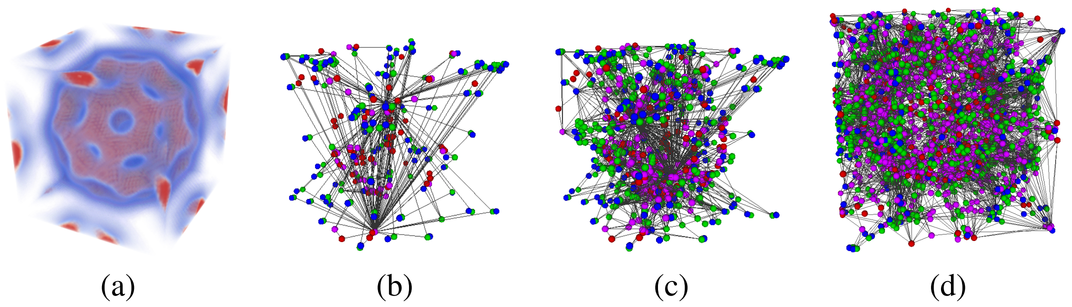


Figure 5.4: Representations extracted from the scalar field of the BUCKY dataset shown in (a). In (b) the coarse MIG is shown and in (b) and (c) the MIG extracted after 250 and 400 refinements respectively. Blue dots corresponds to minima, green dots to 1-saddles, purple dots correspond to 2-saddles and red dots to maxima.

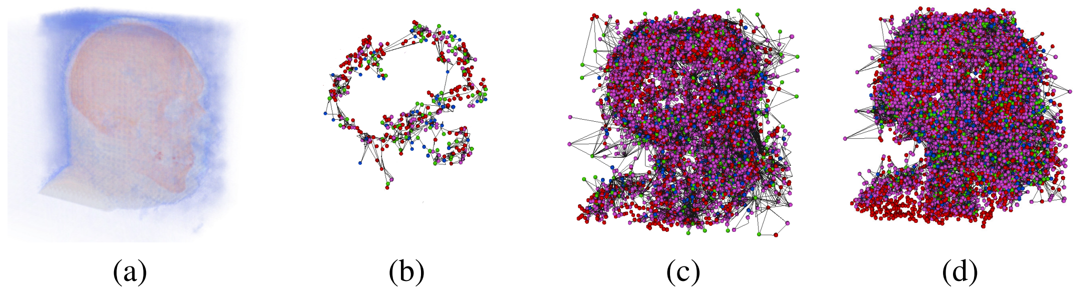


Figure 5.5: Representations extracted from the scalar field of the BONSAI dataset shown in (a). In (b) the coarse MIG is illustrated and in (b) and (c) the MIG extracted after 5K and 10K refinements respectively. Blue dots corresponds to minima, green dots to 1-saddles, purple dots correspond to 2-saddles and red dots to maxima.

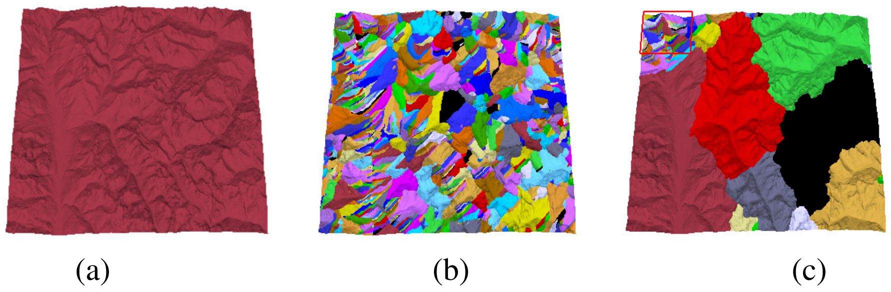


Figure 5.6: The descending Morse complexes encoded in the coarsest resolution MIG (a), the full resolution MIG (b), and the MIG at intermediate resolution (c) for the MATTERHORN 2D terrain data set.

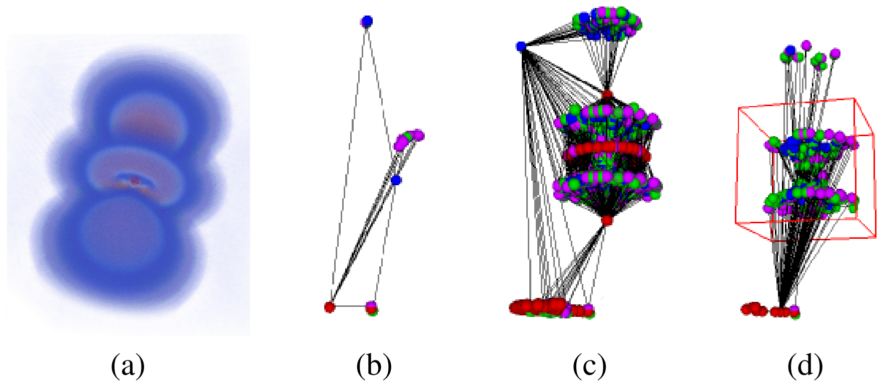


Figure 5.7: The original scalar field (a), the coarsest resolution MIG (b), the full resolution MIG (c), and the MIG at intermediate resolution (d) for the HYDROGEN 3D data set.

the Morse-Smale complex for example, as discussed in Section 2.1, can massively benefit from a reduced input. From the visualization points of view instead, further improvement should be done for reducing the extraction time. Building a spatial index on the $MMIG$ we should be able to reduce the selective refinement query using a bottom-up traversal of the $MMIG$ for selecting the interesting refinement. Based on such index, we could also develop a parallel implementation working on independent subsets of the MIG at the same time. However, we are not going to pursue any of these ideas in the immediate future since we are mainly interested in reducing also the complexity of the underlying geometry of these dataset which are generally huge.

In Figure 5.5 and 5.4 the result of a selective refinement, extracting representation of the BONSAI and BUCKY dataset at uniform levels of resolution, are shown. Results of applying the selective refinement algorithm at variable resolution are shown in Figure 5.6 and Figure 5.7 for the MATTERHORN 2D data set and the HYDROGEN 3D data set, respectively.

Chapter 6

Hierarchical Cell Complex and Homology Computation

In this Section a hierarchical model for cell complexes, called the *Hierarchical Cell Complex (HCC)*, is described. The model consists of a hierarchy of combinatorial representations of a cell complex in the form of an *Incidence Graph* (IG) based on Euler operators. We have defined and applied such model for computing homology and homology generators efficiently on a cell complex.

A *Hierarchical Cell Complex (HCC)* [CFI13a, CFI13c] is generated from an n-dimensional cell complex Γ at full resolution by iteratively applying simplification $KiC(i+1)C$ operators. A $KiC(i+1)C$, ***Kill** i-cell $(i+1)$-cell* is an Euler homology-preserving operator for cell complexes. A homology-preserving operator changes the number of cells in the complex Γ without modifying the Betti number of the complex. The updates performed by a $KiC(i+1)C$ operator on a cell complex Γ are the same as the updates performed on a Morse complex by the $remove_{i,i+1}$ operator, described in Section 4.1, and their feasibility conditions coincide. A rigorous description of such operators can be found in [CFI13a]. We will present a new set of *homology-preserving* and *homology-modifying* operators for cell complexes in Section 6.2 and we will compare them with the other Euler operators known in literature. This work has been presented in [CFI13c]

Similarly to the multi-resolution model described in Section 5 an HCC is generated from a cell complex Γ applying in a sequence the homology-preserving operators, obtaining a complex Γ_B (with fewer cells) called base complex. The set of refinements, inverse of the simplifications performed, the dependency relation among such refinements and the base complex are the three components of an homology-preserving hierarchical cell complex. We will describe this model in Section 6.3.1.

By encoding the base complex Γ_B as an incidence graph we can extend the results obtained for the Multi-resolution More complex ($MRMC$) 5.1 to the HCC. We will describe the effect of the Euler homology-preserving operators on the front complex extracted from an HCC in Section 6.2.3 comparing also the 2D instance of the HCC with another hierarchical representation, defined for images, the pyramidal model based on n-maps (see Section 6.3.1.1).

We have performed different simplification approaches for building an HCC and we have studied how different approaches affect the resulting hierarchy. Results obtained are discussed in Section 6.3.1.2. Our hierarchy has been also applied to the homology computation by defining a computation algorithm for the homology generators. Homology and homology generators are computed on the base complex and then different cell complexes are extracted from the HCC increasing the level of detail globally or in a local subset of the domain. Homology generators, computed on the base cell complex, are expanded as well during the extraction. The algorithm defined to extract generators and the results obtained in computing homology on a cell complex are presented in Section 6.3.2. These results are described in [CFI13c]

6.1 Background notions on homology

In this section we introduce some basic definitions (see [Hat01] and [For98] for a more rigorous treatment).

Given a cell complex Γ, it is possible to define the *chain complex* associated with Γ, denoted as $C_*(\Gamma) := (C_k(\Gamma), \partial_k)_{k\in\mathbb{Z}}$, where $\forall k\ C_k(\Gamma)$ is the free Abelian group generated by the k-cells of the cell complex Γ and $\partial_k : C_k(\Gamma) \to C_{k-1}(\Gamma)$ is a homomorphism called *boundary map* which encodes the boundary relations between the k-cells and the $(k-1)$-cells of Γ such that $\partial^2 = 0$. We denote as $Z_k(\Gamma) := \ker \partial_k$ the group of the k-cycles of Γ and as $B_k(\Gamma) := Im\partial_{k+1}$ the group of the k-boundaries of Γ. Then, we define the kth homology group of Γ with coefficients in $\mathbb{Z}$ as

$$H_k(\Gamma) := H_k(C_*(\Gamma)) = \frac{Z_k(\Gamma)}{B_k(\Gamma)}$$

Furthermore, given an arbitrary Abelian group A, we can define the kth homology group with coefficients in A of Γ as $H_k(\Gamma; A) := H_k(C_*(\Gamma) \otimes_{\mathbb{Z}} A)$, where $\otimes_{\mathbb{Z}}$ is the tensor product of Abelian groups. If we consider $A = \mathbb{Z}_2$, $C_*(\Gamma) \otimes_{\mathbb{Z}} \mathbb{Z}_2 := (C_k(\Gamma) \otimes_{\mathbb{Z}} \mathbb{Z}_2, \partial_k \otimes_{\mathbb{Z}} \mathbb{Z}_2)_{k\in\mathbb{Z}}$ is the chain complex whose groups $C_k(\Gamma) \otimes_{\mathbb{Z}} \mathbb{Z}_2$ are just the $\mathbb{Z}_2$-vector spaces generated by the k-cells of Γ and the homomorphisms $\partial_k \otimes_{\mathbb{Z}} \mathbb{Z}_2$ are the boundary maps ∂_k of Γ considered modulo 2.

Computing the homology groups $H_k(\Gamma; \mathbb{Z}_2)$ of a cell complex Γ with the coefficients in $\mathbb{Z}_2$, as described in [Hat01], corresponds to compute the Betti numbers of Γ with coefficients in $\mathbb{Z}_2$. Moreover, for each $k = 0, \ldots, n$, the homology generators of degree k, called *H_k generators*, can be computed. The H_k generators are the generators of the $\mathbb{Z}_2$-vector space $H_k(\Gamma; \mathbb{Z}_2)$, and they represent the independent non-bounding k-cycles in Γ. Each H_k generator of a cell complex Γ is a linear combination of k-cells in Γ with coefficients in $\mathbb{Z}_2$.

In Figure 6.9(a), two H_1 generators are shown as linear combination of 1-cells. The first generator is composed by a set of blue (bold) edges and the other one by a set of red (dotted) edges.

6.2 Homology-preserving and homology-modifying operators

In this section, we present the operators on cell complexes in arbitrary dimensions first introduced in [CFI13c]. We show here that these operators form a minimally complete basis for the set of all possible operators that modify cell complexes in a topologically consistent manner.

There have been many proposals in the literature for manipulating two- and three-dimensional cell complexes. We describe here a minimal set of Euler operators on cell complexes in arbitrary dimensions. We show here that these operators form a minimally complete basis for the set of all possible operators that modify cell complexes in a topologically consistent manner.

These operators can be classified as:

- *homology-preserving operators*: $KiC(i+1)C$ (*Kill i-Cell and (i+1)-Cell*), which remove an i-cell and an $(i+1)$-cell,
- *homology-modifying operators*: $KiCiCycle$ (*Kill i-Cell and i-Cycle*), which kill an i-cell and an i-cycle.

There are in total n homology-preserving and $n+1$ homology-modifying operators on a cell n-complex Γ.

The *Homology-preserving* operators $KiC(i+1)C$ (*Kill i-Cell and (i+1)-Cell*) (*simplification*) operators delete an i-cell and an $(i+1)$-cell from the complex Γ. We have operators of two different types. Operator $KiC(i+1)C(q,p,p')$ of the first type is *feasible* under the following conditions:

- the $(i+1)$-cell q to be deleted is bounded by exactly two i-cells (the i-cell p to be deleted and the i-cell p' which will remain), and the i-cell p appears exactly once on the boundary of the $(i+1)$-cell q;
- the i-cell q to be deleted bounds exactly two $(i+1)$-cells (the $(i+1)$-cell p to be deleted and the $(i+1)$-cell p' which will remain) and the i-cell q appears exactly once on the boundary of the $(i+1)$-cell p.

In the first instance, denoted as $KiC(i+1)C_{co}(q,p,p')$ (*contract*), the effect of the operator is as follows:

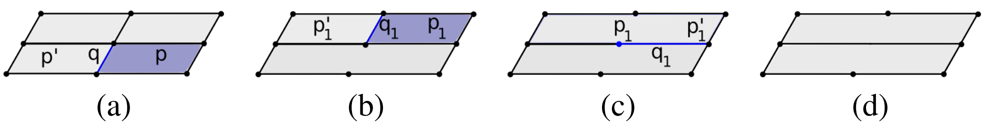


Figure 6.1: Homology-preserving operators on a 2-complex: $K1C2C_{re}(q, p, p')$ (*Kill 1-Cell and 2-Cell*) (a); $K1C2C_{re}(q_1, p_1, p'_1)$ (*Kill 1-Cell and 2-Cell*) (b); $K0C1C_{co}(q_2, p_2, p'_2)$ (*Kill 0-Cell and 1-Cell*) (c).

- the i-cell p is replaced with the i-cell p' on the boundary of each $(i + 1)$-cell r in the co-boundary of the i-cell p.
- if the i-cell p appears k times on the boundary of the $(i + 1)$-cell r, then k copies of the $(i + 1)$-cell q are merged into each $(i + 1)$-cell r.

Second instance, denoted as $KiC(i + 1)C_{re}(q, p, p')$ (*remove*), is completely dual. The effect of the operator is as follows:

- the $(i + 1)$-cell p is replaced with the $(i + 1)$-cell p' on the co-boundary of each i-cell r in the boundary of the $(i + 1)$-cell p.
- if the $(i + 1)$-cell p appears k times on the co-boundary of the i-cell r, then k copies of the i-cell q are merged into each i-cell r.

We give an alternative definition for $KiC(i+1)C(q, p)$ operator when it involves only two cells. $KiC(i + 1)C(q, p)$ is *feasible* under the following conditions:

- the $(i + 1)$-cell q (to be deleted) is bounded only by the i-cell p, which will be deleted as well ($KiC(i + 1)C_{co}(q, p)$);
- the i-cell q (to be deleted) bounds only the $(i + 1)$-cell p which will be deleted as well ($KiC(i + 1)C_{re}(q, p)$).

In both cases, the deleted i-cell appears exactly once on the boundary of the deleted $(i + 1)$-cell. The effect of the operator is to delete both cells from the complex.

Figure 6.1 illustrates a sequence consisting of operators $K1C2C_{re}(q, p, p')$, $K1C2C_{re}$ (q_1, p_1, p'_1) and $K0C1C_{co}$ (q_2, p_2, p'_2) in 2D. $K1C2C_{re}(q, p, p')$ removes 1-cell q and 2-cell p similarly to $K1C2C_{re}(q_1, p_1, p'_1)$, which removes 1-cell q_1 and 2-cell p_1, while $K0C1C_{co}(q_2, p_2, p'_2)$ collapses 1-cell q_2 and 0-cell p_2 into 0-cell p'_2. Figure 6.2 illustrates a sequence consisting of operators $K2C3C_{re}$ (q, p, p') and $K1C2C_{re}$ (q_1, p_1, p'_1) in 3D. $K2C3C_{re}$ (q, p, p') removes 2-cell q and 3-cell p, while $K1C2C_{re}$ (q_1, p_1, p'_1) removes 1-cell q_1 and 2-cell p_1.

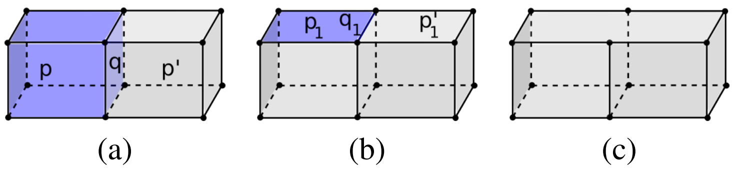


Figure 6.2: Homology-preserving operators on a 3-complex: $K2C3C_{re}(q, p, p')$ (*Kill 2-Cell and 3-Cell*) (a); $K1C2C_{re}(q_1, p_1, p'_1)$ (*Kill 1-Cell and 2-Cell*).

Inverse (*refinement*) operators $MiC(i+1)C$ change the number of cells in cell complex Γ by increasing the number $|\Gamma_i|$ of i-cells and the number $|\Gamma_{i+1}|$ of $(i+1)$-cells by a unit. We have proven, by using discrete Morse theory [For98], that these homology-preserving operators do not change the Euler characteristic, or the Betti numbers of the cell complex with respect to any Abelian group. The proof can be found in [CFI13c]. There are two types of homology-preserving operators, each of which has two distinct instances.

Operator $MiC(i+1)C$ of the *first type* has the following two instances:

- the first instance, that we denote as $MiC(i+1)C_{ex}(q, p, p')$ (*expand*), subdivides the existing i-cell p' into two by splitting its co-boundary, and creates $(i+1)$-cell q bounded by the two i-cells p and p';
- the second instance, that we denote as $MiC(i+1)C_{in}(q, p, p')$ (*insert*), subdivides the existing $(i+1)$-cell p' into two by splitting its boundary, and creates the i-cell q separating the two $(i+1)$-cells p and p'.

In both cases, the new i-cell appears exactly once on the boundary of the new $(i+1)$-cell.

For a 2-complex Γ embedded in $\mathbb{E}^3$, the homology-preserving operators are usually called MEV (*Make Edge and Vertex*) and MEF (*Make Edge and Face*), which correspond to $M0C1C$ and $M1C2C$, respectively. For a 3-complex Γ embedded in $\mathbb{E}^3$, there is an additional homology-preserving operator, $MFVl$ (*Make Face and Volume (3-Cell)*) which creates a new 2-cell and a new 3-cell. It is the same as $M2C3C$.

The alternative definition for the $MiC(i+1)C$ operator works only on two distinct cells ($MiC(i+1)C(q, p)$). It creates an i-cell q and an $(i+1)$-cell p bounded only by the i-cell q ($MiC(i+1)C_{in}(q, p)$), or dually, it creates an $(i+1)$-cell q and an i-cell p bounding only the $(i+1)$-cell q ($MiC(i+1)C_{ex}(q, p)$).

Homology-modifying operators change the number of cells in a complex Γ plus its Betti number and Euler characteristic. Homology-modifying operator $MiCiCycle$ increases the number n_i of i-cells and the number β_i of non-bounding i-cycles by one. It is feasible on a complex Γ if all the cells on the boundary of the cell to be created are in Γ. The inverse $KiCiCycle$ (*Kill i-Cell and*

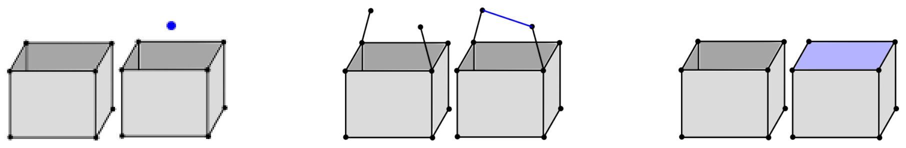


Figure 6.3: Homology-modifying operators on a 2-complex in $\mathbb{E}^3$: $MV0Cycle$ (*Make Vertex and 0-Cycle*) (a); $ME1Cycle$ (*Make Edge and 1-Cycle*) (b); $MF2Cycle$ (*Make Face and 2-Cycle*) (c).

i-Cycle) operator deletes an i-cell and destroys an i-cycle, thus decreasing the numbers n_i and β_i by one. It is feasible on a cell complex Γ if the co-boundary of the cell to be deleted is empty.

For a 2-complex Γ embedded in $\mathbb{E}^3$, the homology-modifying operators (illustrated in Figure 6.3) are also called $MV0Cycle$ (*Make Vertex and 0-Cycle*), $ME1Cycle$ (*Make Edge and 1-Cycle*) and $MF2Cycle$ (*Make Face and 2-Cycle*). Operator $MV0Cycle$ creates a new vertex and a new connected component, it increases by one the number of vertices (0-cells) and the zeroth Betti number β_0. It is used as an initialization operator to create a new complex Γ. Operator $ME1Cycle$ creates a new edge and a new 1-cycle, thus increasing both the number of edges (1-cells) and the first Betti number β_1 by a unit. Operator $MF2Cycle$ creates a new 2-cell (face) and a new 2-cycle, thus increasing the number of 2-cells and the second Betti number β_2 by a unit. When considering a 3-complex Γ embedded in $\mathbb{E}^3$, the homology-modifying operators will be the same as for 2-complexes, since in this case the third Betti number β_3 is null.

6.2.1 Minimality and completeness

The operators described in Subsection 6.2 form a minimally complete set of basis operators for creating and updating cell n-complexes. To prove this fact, we interpret such operators as ordered $(2n+2)$-tuples $(x_0, x_1, .., x_n, c_0, c_1, .., c_n)$ in an integer grid, belonging to the hyperplane Π: $\sum_{i=0}^{n}(-1)^i x_i = \sum_{i=0}^{n}(-1)^i c_i$ defined by the Euler-Poincaré formula. The first $n+1$ coordinates denote the number of i-cells created or deleted by the operator, depending on the sign of the coordinate, while the last $n+1$ coordinates denote the change in the Betti numbers of the complex induced by the operator. Operator $MiC(i+1)C$, $0 \leq i \leq n-1$, has coordinates

$$x_i = x_{i+1} = 1, x_j = 0, j \in \{0, 1, ..., n\} \backslash \{i, i+1\}, c_j = 0, j \in \{0, 1, .., n\}.$$

Operator $MiCiCycle$, $0 \leq i \leq n$, has coordinates

$$x_i = c_i = 1, x_j = c_j = 0, j \in \{0, 1, ..., n\} \backslash \{i\}.$$

We will show that:

(i) the $2n+1$ $(2n+2)$-tuples corresponding to our operators are linearly independent, and

(ii) any $(2n+2)$-tuple in the hyperplane Π can be expressed as a linear combination of $2n+1$ $(2n+2)$-tuples corresponding to our operators,

which will imply the claim.

A linear combination

$$\sum_{i=0}^{n-1} \alpha_i MiC(i+1)C + \sum_{i=0}^{n} \beta_i MiCiCycle$$

vanishes if and only if

$$(\alpha_0, \alpha_0, 0, .., 0) + (0, \alpha_1, \alpha_1, .., 0) + .. + (0, ..0, \alpha_{n-1}, \alpha_{n-1}, 0, .., 0) +$$
$$(\beta_0, 0, .., 0, \beta_0, 0, .., 0) + (0, \beta_1, 0, .., 0, \beta_1, 0, .., 0) + .. + (0, .., 0, \beta_n, 0, .., 0, \beta_n) = 0,$$

which is equivalent to

$$(\alpha_0 + \beta_0, \alpha_0 + \alpha_1 + \beta_1, \alpha_1 + \alpha_2 + \beta_2, .., \alpha_{n-2} + \alpha_{n-1} + \beta_{n-1}, \alpha_{n-1} + \beta_n, \beta_0, \beta_1, .., \beta_n) = 0.$$

It follows that $\alpha_i = 0, 0 \leq i \leq n-1$, $\beta_i = 0, 0 \leq i \leq n$, implying that the tuples corresponding to our operators are linearly independent.

A tuple $(a_0, a_1, .., a_n, b_0, b_1, .., b_n)$ in the hyperplane Π (i. e., such that $\sum_{i=0}^{n}(-1)^i a_i = \sum_{i=0}^{n}(-1)^i b_i$) can be expressed through the $2n+1$ independent $(2n+2)$-tuples corresponding to our operators as

$$\sum_{i=0}^{n-1} \alpha_i MiC(i+1)C + \sum_{i=0}^{n} \beta_i MiCiCycle$$

if

$$(\alpha_0 + \beta_0, \alpha_0 + \alpha_1 + \beta_1, \alpha_1 + \alpha_2 + \beta_2, .., \alpha_{n-2} + \alpha_{n-1} + \beta_{n-1}, \alpha_{n-1} + \beta_n, \beta_0, \beta_1, .., \beta_n)$$
$$= (a_0, a_1, .., a_n, b_0, b_1, .., b_n).$$

It follows that

$$\beta_i = b_i, 0 \leq i \leq n,$$

and

$$\begin{aligned}
&\alpha_0 = a_0 - b_0,\\
&\alpha_1 = a_1 - b_1 - \alpha_0 = (a_1 - a_0) - (b_1 - b_0),\\
&\alpha_2 = a_2 - b_2 - \alpha_1 = (a_2 - a_1 + a_0) - (b_2 - b_1 + b_0), \cdots,\\
&\alpha_{n-1} = (a_{n-1} - a_{n-2} + .. + (-1)^n a_0) - (b_{n-1} - b_{n-2} + .. + (-1)^n b_0) = a_n - b_n.
\end{aligned}$$

In short, $\alpha_i = \sum_{j=0}^{i}(-1)^{i-j}a_j - \sum_{j=0}^{i}(-1)^{i-j}b_j$, $0 \leq i \leq n-1$ ($\alpha_{n-1} = a_n - b_n$) and $\beta_i = b_i$, $0 \leq i \leq n$. Thus, each operator satisfying Euler-Poincaré formula on a cell complex Γ can be expressed as a linear combination of our operators.

In the space (hyperplane) of dimension $2n+1$, a generating set consisting of $2n+1$ independent tuples forms a basis for the hyperplane.

6.2.2 Comparison with homology-preserving operators and handle operators

Virtually all proposed sets of basis Euler operators on 2D and 3D cell complexes contain MEV (Make Edge and Vertex) and MEF (Make Edge and Face) operators, which are the same as our $M0C1C$ (Make 0-cell and 1-cell) and $M1C2C$ (Make 1-cell and 2-cell) homology-preserving operators, respectively.

Several homology-modifying operators have been proposed for manifold 2-complexes bounding a solid in $\mathbb{E}^3$, called *boundary models*. In these models, there is only one implicitly represented volumetric cell (corresponding to the cavity determined by the complex), which is not necessarily a topological cell.

The *glue* operator in [EW79] merges two faces and deletes both of them. It corresponds to the connected sum operator on manifold surfaces. If the two glued faces belong to two different shells, one shell is deleted (β_0 is decreased by one). If the two glued faces belong to the same shell, a handle (genus) is created (β_1 is increased by two).

In [BHS80, MS82, Man88], the topology-modifying operator is called $MRKF$ (*Make Ring, Kill Face*). It is similar to the *glue* operator in [EW79], but it imposes less restrictive conditions on the glued faces, and it deletes only one of the faces.

Homology-modifying operators defined for non-manifold 2-complexes in $\mathbb{E}^3$ [LL01] are called $MECh$ (*Make Edge and Complex Hole*), $MFKCh$ (*Make Face, Kill Complex Hole*) and $MFCc$ (*Make Face and Complex Cavity*). They are the same as our operators $M1C1Cycle$, $M2CK1Cycle$ (*Make 2-Cell Kill 1-Cycle*, which can be expressed as $K1C1Cycle$, $M1C2C$) and $M2C2Cycle$, respectively. For 3-complexes in $\mathbb{E}^3$ [MSNK89, Mas93], an additional homology-modifying operator is defined, called $MVlKCc$ (*Make Volume, Kill Complex Cavity*). It is the same as $M3CK2Cycle$ (*Make 3-Cell, Kill 2-Cycle*) operator, and can be expressed as $K2C2Cycle$, $M2C3C$.

In [Gom04], the operators in [MSNK89] have been extended to complexes called *stratifications*, in which cells, called *strata*, are defined by analytic equalities and inequalities. The cells are not necessarily homeomorphic to a ball, and they may have incomplete boundaries. Among the operators on stratifications proposed in [Gom04], operators on topological cells with complete boundaries can be classified as homology-preserving operators (called *cell subdividers*) and homology-modifying ones (called *global hole shapers*). Both types of operators are instances of the operators we proposed here.

A cell subdivider subdivides an i-cell by inserting into it an $(i-1)$-cell. This operator is equal to the $M(i-1)CiC$ operator.

A global hole shaper either attaches or detaches a cell, thus creating a hole. There are two instances of this operator: the attached topological i-cell creates an i-hole or the detached topological i-cell creates an $(i-1)$-hole. The first instance of this operator corresponds to $MiCiCycle$. The second instance corresponds to $KiCM(i-1)Cycle$ (*Kill i-Cell, Make (i-1)-Cycle*), and can be expressed as $M(i-1)C(i-1)Cycle$, $K(i-1)CiC$.

The inverse homology-modifying operators attach or detach a cell, thus deleting a hole. They correspond to $KiCiCycle$ and $MiCK(i-1)Cycle$ (inverse to $KiCM(i-1)Cycle$), respectively.

(Homology-modifying) *handle operators* on a manifold cell 2-complex Γ triangulating a surface S have been introduced in [LPT$^+$03]. They are based on handlebody theory for surfaces [Mat02], stating that any surface S can be obtained from a 2-ball by iteratively attaching handles (0-, 1- and 2-handles).

Attachment of a 0-handle is also an initialization operator. It creates a new surface with one face, three edges and three vertices. It can be expressed as $M0C0Cycle$ operator, two $M0C1C$ operators and one $M1C2C$ operator, which together create a triangle. The operator that corresponds to attaching a 1-handle identifies two boundary edges of Γ (incident in exactly one face) with no vertices in common. The operator that corresponds to the attachment of a 2-handle identifies two edges on the boundary of Γ with two vertices in common. They can be expressed through our operators in a similar manner.

Handle operators have been extended to 3D in [LT97]. The operator that creates a new 3-ball (initialization operator) corresponds to the attachment of a 0-handle. Other operators identify two boundary faces (incident in exactly one 3-cell) of a cell 3-complex Γ triangulating a solid S. The attachment of a 1-handle (2-handle or 3-handle) can be applied if the two identified boundary faces have no edges (some edges or all edges) in common. The handle operators in 3D generalize the glue operator in [EW79]. They can be expressed in terms of our operators in a similar manner.

6.2.3 Topological operators on the incidence graph

It can be noticed that the homology-preserving operators are equivalent to the updated operator defined in Section 4.1 for Morse complexes. In particular, $KiC(i+1)C_{co}$ operator is equivalent to $remove_{(i,i-1)}$ operator while $KiC(i+1)C_{re}$ operator is equivalent to $remove_{(i,i+1)}$ operator. Similarly to what we have done in Section 4.2, we have considered the effect of homology-preserving and homology-modifying operators on the incidence graph structure $G = (N, A, \varphi)$. $KiC(i+1)C$ operator deletes an i-node and an $(i+1)$-node from N, and suitably reconnects the remaining nodes.

The *contract* operator, $KiC(i+1)C_{co}(q, p, p')$, is feasible on incidence graph $G = (N, A)$ if $(i+1)$-node q is connected to exactly two different i-nodes p and p', and there is exactly one arc in A connecting $(i+1)$-node q and i-node p ($\varphi((q, p)) = 1$).

- It deletes nodes p and q, all the arcs incident in $(i+1)$-node q and all the arcs incident in i-node p and connecting p to $(i-1)$-nodes, and it replaces all the arcs incident in i-node p and connecting p to $(i+1)$-nodes with arcs connecting i-node p' to the same $(i+1)$-nodes.
- The label $\varphi'((p', r))$ after $KiC(i+1)C_{co}(q, p, p')$ is increased by the product of the label of the arc connecting nodes p' and q and the label of the arc connecting nodes p and r.

The removal operator $KiC(i+1)C_{re}(q, p, p')$ is dual.

The inverse $MiC(i+1)C$ operators on graph $G = (N, A, \varphi)$ insert two nodes in N and arcs incident in them in A. $MiC(i+1)C_{ex}(q, p, p')$ is specified by

- the two nodes ($(i+1)$-node q and i-node p) to be inserted;
- i-node p' that is the only i-node (except for i-node p); that will be connected to $(i+1)$-node q;
- the $(i+2)$-nodes that will be connected to $(i+1)$-node q;
- the $(i-1)$-nodes and $(i+1)$-nodes that will be connected to i-node p, together with the multiplicity (labels φ') of all the inserted arcs.

$MiC(i+1)C_{ex}(q, p, p')$ is feasible if all the specified nodes are in N, and the label $\varphi((p', r))$ before the refinement for each $(i+1)$-node r that will be connected to i-node p is greater than or equal to $\varphi'((p', q)) \cdot \varphi'((p, r))$. Its effect is to add nodes p and q in N and all the specified arcs in A and to set $\varphi'((p', r)) = \varphi((p', r)) - \varphi'((p', q)) \cdot \varphi'((p, r))$.

The second instance $MiC(i+1)C_{in}(q, p, p')$ has a completely dual effect.

The effect of homology-modifying operators on the incidence graph $G = (N, A)$ is as follows. Operator $KiCiCycle(p)$ deletes i-node p and all the incident arcs from the graph. It is feasible if i-node p is not connected to any $(i + 1)$-node in N. The inverse $MiCiCycle(p)$ operator creates i-node p and arcs connecting it to $(i - 1)$-nodes. It is specified by $(i - 1)$-nodes that will be connected to i-node p, together with the multiplicity (φ) of the corresponding arcs. It is feasible on the incidence graph $G = (N, A)$ if all the $(i - 1)$-nodes to be connected to p are in N.

6.3 The Hierarchical Cell Complex (HCC)

In this Section, we introduce a hierarchical model for a cell complex, that we call a *Hierarchical Cell Complex (HCC)*, and we discuss its major properties. We define an HCC in terms of the refinement operators introduced in Section 6.2.

A *Hierarchical Cell Complex (HCC)* is generated from a n-complex Γ at full resolution by iteratively applying simplification operators $KiC(i + 1)C$ and $KiCiCycle$. By applying first the homology-preserving operators, we obtain a complex Γ' having the same homology as the original complex Γ but with fewer cells and such that no homology-preserving operator is feasible on Γ'. By applying next the homology-modifying operators to iteratively remove the cells of Γ', the homology is affected at each step and the process is repeated until all the cells in the complex but one have been deleted. At each step, when we apply a homology-modifying operator, we remove a top cell from the complex. After each application of a homology-modifying operator, we perform feasible homology-preserving ones.

We call the application of a simplification operator a *simplification modification*. The complex obtained as the result of the simplification sequence (see Figure 6.4), is the coarsest representation of the original cell complex Γ. We denote such coarsest complex as Γ_B. It represents the first component of an HCC.

The second component of an HCC is the set $\mathcal{M}$ of the *refinement modifications* which are inverse with respect to the simplification modifications in that have produced Γ_B from Γ. Each refinement introduces new cells (two cells if it is homology-preserving, one if it is homology-modifying).

The third component of an HCC is the dependency relation between the modifications in the set $\mathcal{M}$ of all refinement modifications. We consider, for simplicity, the creation of the coarse complex Γ_B as a dummy refinement modification that we denote as μ_0 (i.e. μ_0 generates Γ_B). We define the dependency relation between the refinement modifications in $\mathcal{M}$ as follows:

- a homology-preserving refinement $\mu = MiC(i + 1)C$, which creates cells p and q and is defined by the cells that will appear on the immediate boundary or co-boundary of either p or q, *directly depends* on a refinement μ^*, if μ^* creates a cell that will be on the immediate

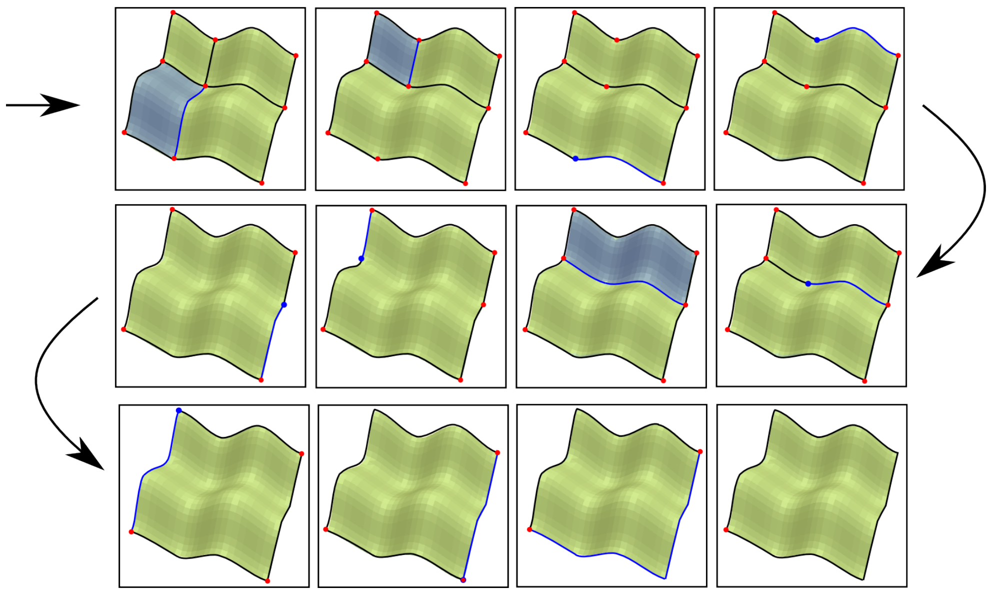


Figure 6.4: A sequence of simplifications applied on a cell complex to produce a coarsest representation. 2-cells are illustrated in yellow, 1-cells are shown as black lines and 0-cells are the red points. Cells deleted at each simplification are shown in blue.

boundary or co-boundary of p or q.

- a homology-modifying refinement $\mu = MiCiCycle$, which creates i-cell p and defined by the $(i-1)$-cells that will be on the immediate boundary of p, *directly depends* on a refinement μ^*, if μ^* creates a cell that will be on the immediate boundary of p.

An HCC is thus a triple $(\Gamma_B, \mathcal{M}, \mathcal{R})$, where $\mathcal{R}$ denotes the direct dependency relation defined above. The dependency relation between refinement modifications is the transitive closure of the direct dependency relation. It is a partial order relation, since a cell is never introduced twice by the modifications in $\mathcal{M}$. An HCC can be naturally encoded as a Directed Acyclic Graph (DAG), in which the nodes encode the modifications in $\mathcal{M}$, the root encodes the creation of the base complex Γ_B (modification μ_0), and the arcs describe the direct dependency relation $\mathcal{R}$.

In Figure 6.5 an example of the HCC built from the sequence of simplifications shown in Figure 6.4 is illustrated. The base complex Γ_B is indicated as the root of the model while the remaining nodes store the refinement operations. An arrow connects two nodes when there is a dependency relation among them.

Figure 6.6 shows the dependency relations among three different refinements extrapolated from the hierarchical model shown in Figure 6.5. To perform refinement 7 a specific set of cells are required in the boundary and co-boundary of the newly introduced cells p and q. The dependency relations with refinements 4 and 5 assure that such cells will be present in the complex once

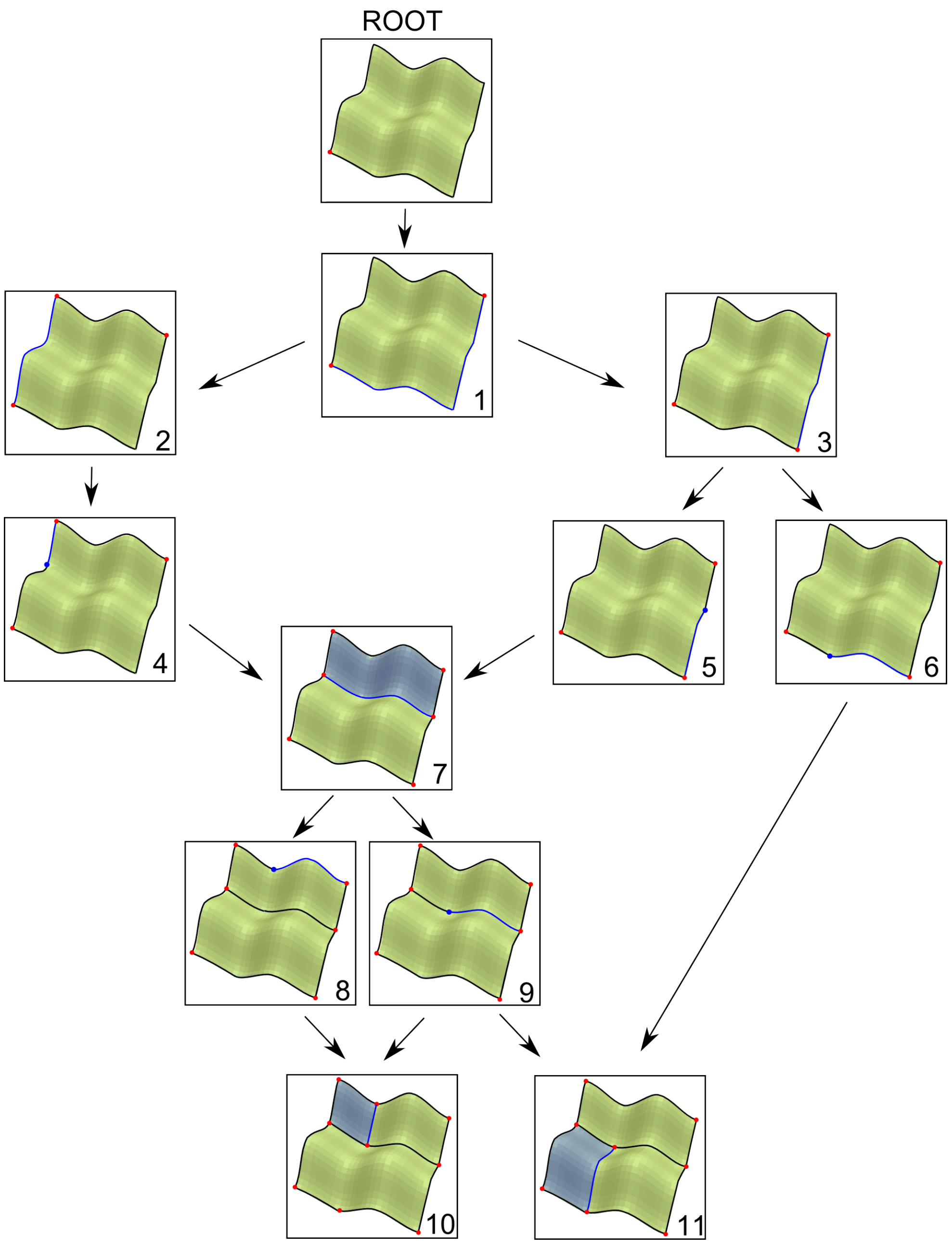


Figure 6.5: HCC model built from a sequence of simplifications. The root refinement correspond to the base complex Γ_B. Remaining nodes indicate a refinement operation each one, with the introduced cells colored in blue. Arrows represent the dependency relation.

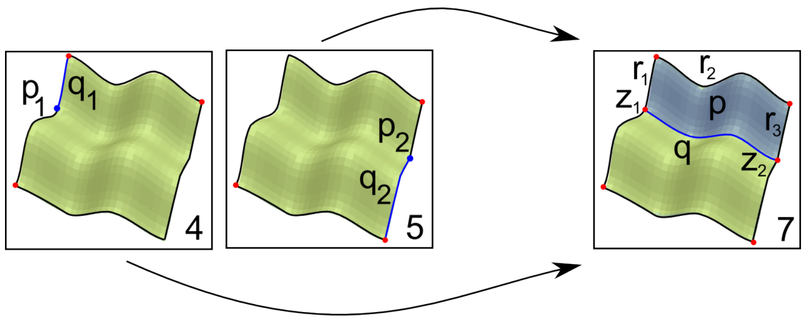


Figure 6.6: Dependency relation between the refinement 4,5 and 7 from the multi-resolution model shown in Figure 6.5.

refinement 7 is performed. Refinement 4 reintroduces the 1-cell r_1 and 0-cell z_1 required by refinement 7 while refinement 5 reintroduces the 0-cell z_2. Remaining cells required by refinement 7 are in the base complex.

From an HCC, a large number of complexes at intermediate resolution can be obtained by applying sequences of refinement modifications in $\mathcal{M}$ to the base complex Γ_B. A sequence $U = [\mu_0, \mu_1, .., \mu_k]$ is said to be *feasible* if each refinement μ_i in U is feasible on the complex obtained from the base complex Γ_B by applying all the refinements preceding μ_i in U. For a feasible sequence $U = [\mu_0, \mu_1, \mu_2, ..., \mu_m]$ of refinement modifications in $\mathcal{M}$, the complex obtained from the base complex Γ_B by applying U is called the *front complex* associated with U, and we denote it as Γ_U. A front complex is a complex at an intermediate resolution.

The refinement modification μ, which creates the cells p and q (if μ is homology-preserving), or the cell p (if μ is homology-modifying) is feasible on a front complex Γ_U (at some intermediate resolution) if and only if all the cells on the immediate boundary and co-boundary of the cells p and q (if μ is homology-preserving) or all the cells on the immediate boundary of the cell p (if μ is homology-modifying) are in the complex Γ_U, i.e., if the sequence U that creates Γ_U from Γ_B contains all refinement modifications on which μ depends.

A large number of adaptive morphological representations can thus be extracted from an HCC defined by the triple $(\Gamma_B, \mathcal{M}, \mathcal{R})$ by considering the closed sets of refinement modifications in $\mathcal{M}$ plus μ_0 under the dependency relation $\mathcal{R}$. Recall that the dependency relation $\mathcal{R}$ is a partial order relation, and thus it defines a closure operator on the set $\mathcal{M}$ of refinement modifications. We denote a *closed set* of such refinement modifications as $\mathcal{U}$. Set $\mathcal{U}$ implicitly defines complex Γ_U representing an approximation of the original complex.

The set $\mathcal{U} = \{\mu_0, \mu_1, \mu_2, ..., \mu_m\}$ of refinement modifications in $\mathcal{M}$ is *closed* with respect to the dependency relation $\mathcal{R}$ if for each i, $1 \leq i \leq m$, each refinement modification on which the refinement μ_i depends is in $\mathcal{U}$. If the sequence $U = [\mu_0, \mu_1, \mu_2, ..., \mu_m]$ of refinement modifications in $\mathcal{M}$ is feasible, then the set $\mathcal{U} = \{\mu_0, \mu_1, \mu_2, ..., \mu_m\}$ is *closed* with respect to the dependency relation $\mathcal{R}$.

Two feasible refinement modifications μ_1 and μ_2 on a complex are *interchangeable* if the sequence (μ_1, μ_2) of refinements (consisting of μ_1 followed by μ_2) produces the same complex as the sequence (μ_2, μ_1) (consisting of μ_2 followed by μ_1). Any two independent refinement modifications μ_1 and μ_2 are interchangeable. Thus, a closed subset $\mathcal{U}$ of refinement modifications can be applied to the base complex Γ_B in any total order U that extends the partial order, producing a complex Γ_U at an intermediate resolution. An HCC encodes a collection of all complexes at intermediate level of detail which can be obtained from the base complex Γ_B by applying a closed set of modifications on Γ_B.

From an HCC it is thus possible to dynamically extract representations of the original cell complex Γ at uniform and variable resolutions. The basic query for extracting a single-resolution representation from a multi-resolution model is known as selective refinement. A *selective refinement query* consists of extracting from an HCC a complex with the *minimum* number of cells, satisfying some application-dependent criterion. This criterion can be formalized by defining a Boolean function τ over all nodes of an HCC, such that the value of τ is $true$ on nodes which satisfy the criterion, and $false$ otherwise. The same value of τ is associated with the cells created by the modification encoded in the node of the HCC (p and q for homology-preserving modification, p for homology-modifying modification). The selective refinement query consists of extracting from the HCC an intermediate complex of minimum size among the complexes encoded in the HCC that satisfies τ. Equivalently, it consists of extracting a minimal closed set $\mathcal{U}$ of modifications in $\mathcal{M}$, such that the corresponding complex satisfies τ. Such closed set of modifications uniquely determines a front complex, which is obtained from the base complex Γ_B, by applying to it all modifications from $\mathcal{U}$ in any order that is consistent with the partial order defined by the dependency relation. Criterion τ can be defined based on various conditions posed on the cells in the extracted complex, like the size of the cell (which may be expressed as the maximum distance between its vertices or the diameter of its bounding box) or the portion of the complex in which full resolution is required (while in the rest of the complex, the resolution may be arbitrarily low).

6.3.1 Homology-preserving HCC

We have implemented a version of the HCC based on the homology-preserving operators and on the incidence graph. The motivation for this choice is in the computation of homology and homology generators at multiple resolutions. A sequence of homology-preserving operators can be applied on a complex at full resolution, until no such operator is feasible, producing a simplified complex with the same homology as the initial one. Homology and its generators can be computed on the simplified complex, the root of the HCC, using existing techniques based on boundary matrices. Homology generators could then be extracted on any complex implicitly encoded in the HCC.

We encode the initial full-resolution complex as an incidence graph. Thus, we show how the operators defined in Section 6.2 are defined on the incidence graph and we discuss how we encode the HCC based on such representation. Note however that our definition of the HCC is independent of the specific implementation data structure. We have chosen the incidence graph as the basis of our implementation since it can effectively and efficiently encode arbitrary cell complexes in any dimension.

The HCC is encoded as a Direct Acyclic Graph (DAG). Its root represents modification μ_0 that creates the base graph G_B and the complex Γ_B at coarsest resolution. The other nodes represent refinement modifications in $\mathcal{M}$ and arcs represent direct dependency relations. The root stores G_B as an incidence graph and the pointers to its child modifications that depend on μ_0. Each node in the DAG encodes the two nodes p and q created, a pointer to the node p' (it can be NULL) with the multiplicity of arc (q, p'), a list of pointers to the nodes that will be connected to p or q with the multiplicity of the connecting arcs, dimension i of q, the type of refinement, and two sets of pointers (to parent and child nodes) encoding the direct dependency relation.

6.3.1.1 Comparison with pyramids on 2-maps

An alternative approach to hierarchical representation of cell complexes is a pyramidal model, defined on quasi-manifolds represented as 2-maps [BK00] and n-G-maps [SDL05]. The n-maps [BK00] and n-G-maps [Lie94] are data structures proposed for modeling cell complexes subdividing quasi-manifolds in arbitrary dimensions. Quasi-manifolds are a subclass of pseudo-manifolds. n-maps and n-G-maps are both implicit representations, in which the cells are represented through sets of incident abstract elements, called *darts*. The data structure implementing n-G-maps occupies twice as much space as the n-maps data structure. In [DH07], it has been shown that the IG is more compact than the n-G-map. The storage cost of the IG (in the case of manifold domain) and n-map encoding a simplicial complex Σ with n_i i-cells is $\sum_{i=1}^{d}(i+1)n_i$ and $n_d(d+1)(d+1)! + n_d(d+1)!$, respectively.

Each level in a map pyramid is a map. The first level describes the initial full-resolution complex, while the other levels describe successive reductions of the previous levels. The reduction is obtained by applying the removal and contraction operators [DL03]. In 2D, these operators merge regions (2-cells) in the lower level into one region in the successive level (contraction operators) and simplify the boundaries between the new merged regions (removal operators).

Compared with pyramidal models, the HCC has a wider representation domain, since it can represent arbitrary cell complexes and this also is entirely supported by the encoding of the HCC as an incidence graph. This is true also if we restrict our consideration to homology-preserving HCCs. Another important advantage of the HCC is its greater adaptivity, and thus the larger number of complexes that can be extracted from it. In a map pyramid, a contraction kernel at level k describes a set of cells that are merged into one cell from one level of the pyramid to the next one. This merging (a set of contractions) can be seen as a macro-operator, consisting of a sequence of contractions (i.e., $KiC(i+1)C_{co}$ operators). All the operators in this sequence are either performed (at levels greater than k), or neither of them is performed (at levels less than k). The removal kernel at level k describes the simplification of the boundaries of the merged cells. We build the HCC using the same removal ($KIC(i+1)C_{re}$) and contraction ($KIC(i+1)C_{co}$) simplification operators, but we consider each of them separately and we do not group them into macro-operators.

6.3.1.2 Experimental evaluation

The purpose of our experiments is twofold. In the first set of experiments, two simplification strategies to build the HCC have been tested: one approach is based on performing simplifications one at the time, and the other one on performing a set of independent simplifications. Successively the capabilities of the HCC to extract adaptive representations at variable resolutions have been tested and the timings have been compared for the two approaches.

The experiments have been performed on 2D and 3D simplicial complexes available on the Web and encoded as an IG, that become cell complexes after undergoing some simplifications. The initial size of these complexes is between 400K and 953K triangles for 2D data sets, and between 68K and 577K tetrahedra for 3D data sets. Experiments have been performed on a desktop computer with a 3.2Ghz processor and 16Gb of memory.

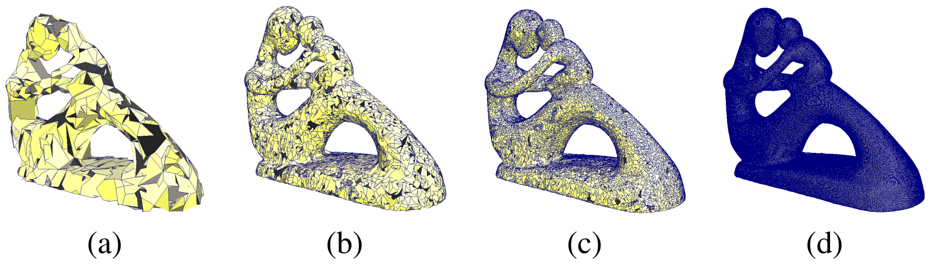


Figure 6.7: The base complex (a) of the FERTILITY data set, and complexes obtained from it after 300K (b), 600K (c) and 700K (d) refinements equally distributed on the domain.

To build the HCC, we start from the complex at full resolution and perform all feasible homology preserving simplifications, according to some predefined error criterion, until the coarsest representation is reached. Incidence graphs, similarly to the majority of data structures proposed in the literature for representing cell complexes, can encode only complexes in which each cell has a non-empty immediate boundary and if it has a non-empty co-boundary (if it is not a top cell), then it has a non-empty immediate co-boundary. Thus, we introduce one additional condition for the feasibility of simplification operators on the incidence graphs, namely that after the application of the operator each i-node is connected to at least one $(i-1)$-node ($1 \leq i \leq n$), and (if it does not correspond to a top cell) to at least one $(i+1)$-node ($0 \leq i \leq n-1$).

The implementation of the simplification algorithm is independent of the error criterion. We have used a geometric criterion computed on the vertices of the cells to be deleted. The simplification is guided by a priority queue in which feasible simplifications are sorted according to the selected criterion. We have implemented two different simplification approaches. In the first one, called *simplification step by step*, simplifications are taken from the priority queue in ascending order and performed if still feasible. After each simplification, the local connectivity of the nodes involved in it changes and each new feasible simplification is enqueued. The algorithm ends when there are no more feasible simplifications.

The second approach, called *batch simplification*, tries to execute at each step a large number of feasible independent simplifications (that involve nodes not involved by any other already selected simplification). At each step, we build a priority queue with all the feasible simplifications sorted in ascending order. We select a set of simplifications from the queue, we perform all of them, and we initialize a new priority queue. In all the complexes we used in the experiments, the DAG produced with the batch method has approximately half the depth of the DAG produced with the step by step method. A shallower DAG guarantees a faster traversal and, thus, a faster refinement process. Refinements on the HCC built on 3D data sets with the batch method are between 2 and 10 times faster than those on the HCC built with the step by step method.

Our experiments also show that the storage cost of the HCC encoding structure is around 25% lower than the storage cost of the incidence graph representing the complex at full resolution. The storage cost of the incidence graph at full resolution is between 4.8MB and 398MB for 2D data sets, and between 118MB and 980MB for 3D data sets. The storage cost of the corresponding HCC is between 3.3MB and 273MB, and between 4.8MB and 398MB, respectively. This behavior can be explained by the fact that the representation of a deleted arc in each node in the HCC is more compact than the arc encoding in the incidence graph (one reference instead of four). Thus, even if the total number of arcs represented in the HCC is larger than the number of arcs in the incidence graph, it is usually less than four times larger.

In Figure 6.7, we show examples of extractions at uniform resolution performed on the FERTILITY data set. The initial complex has approximately 2500K cells.

In Table 6.1, a summary of the results obtained with the two approaches are shown. The columns show, from left to right, data set name ($Data\ set$), total number of cells ($Cells$), number of simplifications ($Simpl.\ Num.$), time needed to perform them ($Simpl.\ Time$), time needed to build the HCC ($HCC\ Time$), storage cost of the HCC ($HCC\ storage$), time needed to perform all the refinements in the HCC ($Ref.\ Time$), storage cost of the cell complex at full resolution ($Full\ complex$) and storage cost of the base complex ($Base\ complex$).

We notice that the time needed to perform all the refinements is always much less than the time needed to perform all the simplifications (refinement is 5 to 10 times faster than simplification). An important aspect is that the storage cost of the HCC structure plus the base graph is less

	Data set	Cells	Simpl. Num.	Simpl. Time	HCC Time	HCC storage	Ref Time	Full complex	Base complex
				Step-by-step simplification					
2D	EROS	2859566	1429781	74.4	5.3	254.9	18.1	349.0	0.0002
	HAND	1287532	643694	35.4	2.3	117.2	7.58	157.1	0.01
	VASELION	1200002	599999	26.7	2.1	105.8	6.8	146.4	0.00028
				Batch simplification					
	EROS	2859566	1429781	218.8	6.4	241.0	18.7	349	0.0002
	HAND	1287532	643741	99	2.6	120.7	7.6	157.1	0.004
	VASELION	1200002	599999	90.7	2.3	110.5	7.7	146.4	0.00028

	Data set	Cells	Simpl. Num.	Simpl. Time.	MCC Time.	MCC storage	Ref Time	Full complex	Base complex
				Step-by-step simplification					
3D	VISMALE	297901	147594	45.1	0.6	40.4	5.1	48	0.46
	BONSAI	1008357	498790	380.6	2.7	146.9	27.2	162.5	1.8
	HYDROGEN	2523927	1248743	8643.8	7.8	395.7	419.5	407.4	4.4
				Batch simplification					
	VISMALE	297901	148116	69.2	0.7	37.6	2.5	48	0.28
	BONSAI	1008357	501524	305.8	2.69	126.4	10.4	162.5	0.89
	HYDROGEN	2523927	1253913	1412.9	7.4	321.3	33.9	407.4	2.7

Table 6.1: Experimental results for the DAG construction. The storage cost is expressed in Megabytes and the computation time in seconds.

2D				3D			
Data set	Perc.	Refinement Time step-by-step	batch	Data set	Perc.	Refinement Time step-by-step	batch
	50%	0.80	0.92		50%	3.45	0.12
EROS	80%	1.42	1.01	*VisMale*	80%	3.77	0.15
	100%	2.63	2.60		100%	4.01	0.53
	50%	0.31	0.57		50%	15.3	0.65
HAND	80%	0.45	0.65	*Bonsai*	80%	17.4	0.69
	100%	1.20	1.19		100%	19.1	1.88
	50%	0.73	0.69		50%	106.3	8.1
VASELION	80%	1.01	0.99	*Hydrogen*	80%	127.7	8.7
	100%	1.10	1.06		100%	172.1	11.3

Table 6.2: Experimental timing results (in seconds) for extraction at variable resolution.

than the storage cost of the graph at full resolution, with the exception of the largest tested (HYDROGEN) data set using the step-by-step method. Although the total number of simplifications is slightly higher for the batch simplification approach, the time required to perform all simplifications that lead to the base complex is less in the case of step-by-step simplification, since it requires fewer computations. On the other hand, the HCC generated through batch simplification uses less memory and consequently can be visited in less time. Also, the DAGs produced by the batch approach have less dependency relations compared to the ones produced through step-by-step simplification.

In Table 6.2, timing results for performing extractions at variable resolution are shown. Column $Perc.$ indicates the desired percentage of operations performed inside a query box. $Ref.Time$ indicates the time needed to perform the required number of refinements with the step-by-step method ($step$) or the batch ($batch$) simplification methods. The query box has been chosen by hand to cover a relevant part for each data set and with size between 15 and 30 percent of the whole data set at full resolution. The extraction times for refinements are similar for the two methods in the 2D case, while they differ considerably in the 3D case. The explanation is that in 2D each 1-node in the graph is connected with at most two different 0-nodes and two different 2-nodes, while in 3D the number of connections between 1-nodes and 2-nodes is larger, and thus the simplification method has a significant impact.

In Figure 6.8, we show examples of refinement queries at uniform and variable resolution performed on the VASELION data set. The holes that seem to appear in the crown of the lion are rendering artifacts.

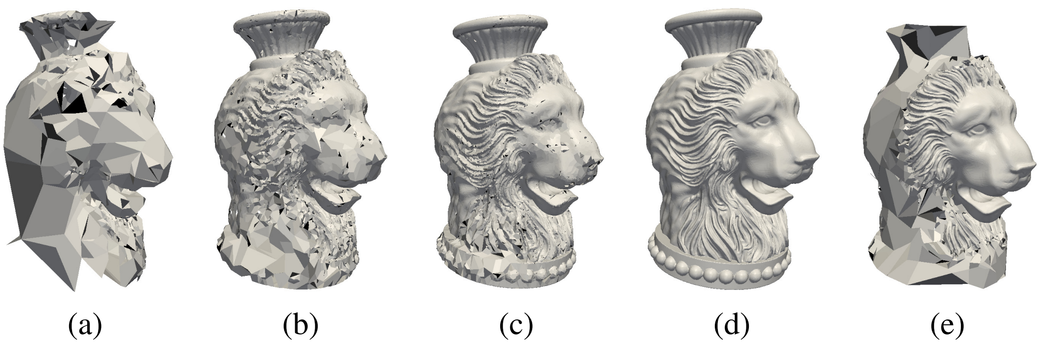


Figure 6.8: The representations obtained from the MCC after (a) 10000, (b) 50000 and (c) 2000000 refinements. (d) The complex at full resolution of the VASELION data set. In (e) the representation obtained with a query at variable resolution.

6.3.2 Homology computation on HCC

In this section, we describe a method exploiting the HCC representation for computing homology and homology generators efficiently. We address the problem of computing homology and homology generators on a cell complex Γ using the Smith Normal Form reduction (SNF) [Ago05]. Computation of the SNF reduction is highly dependent from the number of cells in Γ and is generally slow. We plan to compute the SNF reduction on the base complex Γ_B stored in the root of an homology-preserving HCC obtained simplifying the original complex Γ.

In a homology-preserving HCC, the homology of the base complex is the same as the homology of any other complex implicitly encoded in the HCC (included Γ). However, the Smith Normal Form (SNF) reduction algorithm on the base complex Γ_B is much faster than the SNF computed on Γ (or any other front complex Γ_U extracted from the HCC). Then, we are interested in extracting representations, increasing the resolution of Γ_B. as well as the resolution of the homology generators, computed on Γ_B, to agree with the resolution of the front complex Γ_U.

6.3.2.1 Computation of homology generators

Let consider a base complex Γ_B, on which the homology generators have been computed, and a front complex Γ_U extracted from Γ_B. For each refinement in the feasible sequence $U = [\mu_1, \ldots, \mu_n]$ applied on G_B, the homology generators are modified according to algorithm *ExpandGenerators* described below. In [CDFI13] has been shown that, when applying $MiC(i+1)C$ only the H_{i+1} generators are affected.

Let us consider refinement modification Make i-cell $(i+1)$-cell, undo of a $KiC(i+1)C$, denoted $MiC(i+1)C_{ex}(q,p)$, which creates an i-cell p and an $(i+1)$-cell q (the case of a refinement $MiC(i+1)C_{in}$ is entirely dual). Operator $MiC(i+1)C_{ex}(q,p)$ is applied on a complex Γ

producing a refined complex Γ'.

Algorithm *ExpandGenerators* checks in Γ' if the $(i+1)$-cell introduced q breaks a $(i+1)$-cycle corresponding to an H_{i+1} generator in Γ. This is done by considering the number of $(i+1)$-cells in the co-boundary of i-cell p that are involved in H_{i+1} generators. This idea is illustrated in Figures 6.9(b) and 6.9(c), where

- operator $M0C1C_{ex}(q_1, p_1, p')$, illustrated in Figure 6.9(b), modifies one of the two H_1 generators in the torus. It can be noticed that the new 0-cell p_1 has exactly one incident 1-cell belonging to the blue (bold) 1-chain. Thus the 1-cycle has been broken by the refinement and 1-cell q_1 is added to the 1-chain to reconstruct the cycle;
- operator $M0C1C_{ex}(q_2, p_2, p')$, illustrated in Figure 6.9(c), does not affect the generators. Note that 0-cell p_2 has no incident 1-cell belonging to some H_1 generator.

Algorithm 3 $ExpandGenerators(p, q, G)$

```
Require: $p$ is an $i$-cell
Require: $q$ is an $(i+1)$-cell
Require: $G$ is the set of $H_{i+1}$ generators
 1: // $C$ is a map from a generator $g$ to an integer $m$
 2: $C$ = empty map
 3: // Extract the $(i+1)$-cells on the co-boundary of $p$
 4: for all cofaces $r$ of $p$ do
 5:    // $G_r$ is the set of generators involving $r$
 6:    $G_r$ = getGeneratorsOn($r, G$)
 7:    // Consider the number of incidences between $p$ and $r$
 8:    for all generators $g$ in $G_r$ do
 9:       $C[g]$=getIncidence($p$,$r$)+$C[g]$
10: // Expand the generators on $q$ if necessary
11: for all pairs $(g, m)$ in $C$ do
12:    if $m$ is odd then
13:       addGenerator($g, q, G$)
14: return $G$
```

In the description of algorithm *ExpandGenerators(p,q,G)*, p and q denote, respectively, the i-cell and the $(i+1)$-cell introduced by the refinement operator, and G represents the generators of Γ. The algorithm makes use of a map C from a generator g to an integer m, that, for each generator g, stores the number of $(i+1)$-cells in the co-boundary of i-cell p which also belong to g.

Algorithm *ExpandGenerators(p,q,G)* uses the following three functions:

- $getGeneratorsOn(r, G)$, which returns the set of generators G_r containing cell r in their chain,
- $getIncidence(p, r)$, which returns the number of times i-cell p appears on the boundary of $(i + 1)$-cell r,
- $addToGenerator(g, q, G)$, which updates the generators in G by adding $(i + 1)$-cell q to the $(i + 1)$-chain corresponding to g.

Algorithm *ExpandGenerators(p,q,G)* considers only the $(i + 1)$-cells, in the co-boundary of p, that are part of one or more H_{i+1} generators. For each such $(i + 1)$-cell r, G_r is the set of generators involving r ($getGeneratorsOn(r, G)$). In map C, for each generator $g \in G_r$, the number of times the i-cell p appears in the boundary of r is updated ($getIncidence(p, r)$). Once all the cells in the co-boundary of p have been examined, cell q is added to generator g only if the number m of incidences for g is odd ($addGenerator(g, q, G)$).

In Figure 6.9 (b), the 1-cell q_1 will be included in the blue generator since p_1 is incident to an odd number (one) of 1-cells in the blue generator. On the contrary, In Figure 6.9 (c), the 1-cell q_2 is not refining any generator and indeed the 0-cell p_2 is incident to an even number (zero) of 1-cells in the blue generator.

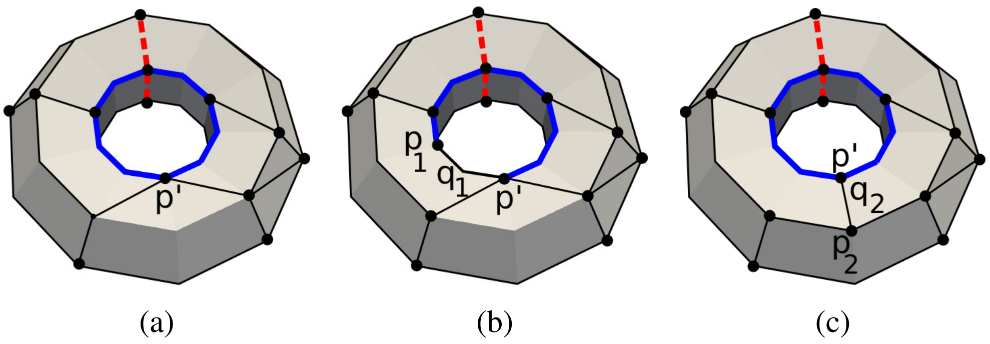


Figure 6.9: (a) A cell complex representing a torus. Black dots represent 0-cells. Red (dotted) and blue (bold) edges correspond to the two H_1 generators. (b) Application of operator $M0C1C_{ex}(q_1, p_1, p')$, which affects one of the homology generators. (c) Application of generator $M0C1C_{ex}(q_2, p_2, p')$, which does not affect the homology generators.

6.3.2.2 Experimental results

Using the HCC and the *expandGenerator* algorithm we have performed tests on the 2D and 3D complexes described in Table 6.3, extracting different representations of the homology generators computed on them. All complexes are simplicial complexes, that become cell complexes

Dataset	Cells	HCC cost(MB)	Homology	SNF Time	Tot Ref Time	Uniform Ref.	Uniform Time	Generators Ref.	Generators Time
GENUS3	40K	3.3	(1,6,1)	9.2×10^{-5}s	0.15s	4K 10K 16K	0.03s 0.07s 0.12s	5K	0.03s
FERTILITY	1.4M	122	(1,8,1)	8.3×10^{-5}s	9.31s	144K 362K 579K	1.8s 4.6s 7.52s	68K	1.48s
HAND	2.1M	177	(1,2,0)	9.8×10^{-5}s	14.9s	200K 500K 800K	2.6s 6.8s 11.2s	19K	1.6s
BUDDHA	3.2M	273	(1,208,1)	0.02s	23.7s	320K 800K 1.2M	0.5s 4.3s 19.2s	162K	3.6s
SKULL	748K	84	(1,2,1,0)	0.007s	6.4s	75K 187K 299K	1.0s 2.9s 5.0s	191K	2.6s
FERT-SOLID	6.2M	720	(1,4,0,0)	8.8s	74.5s	1.2M 3.1M 4.9M	7.5s 29.1s 69.3s	267K	10.9s

Table 6.3: Experimental results obtained by refining four 2D shapes and two volumetric datasets and by computing homology generators on them through the Smith Normal Form (SNF) reduction. The columns from left to right indicate: the name of the dataset (*Dataset*) , the number of top cells (*Cells*) in the datasets, the storage cost of the HCC (*HCC cost*), the Betti numbers (*Homology*), time required to compute them on the base complex (*SNF Time*), the time needed to extract the complex at full resolution and to expand all the generators (*Tot Ref Time*), the number of refinements and the time needed to extract the complex and the geometry of the generators at uniform level of detail (*Uniform Ref.* and *Uniform Time*) and the number of refinements and the time needed to extract the complex and the generators concentrating the resolution only in the neighborhood of the generators (*Generators Ref.*) and (*Generators Time*). The time is expressed in seconds.

after undergoing some simplification.

First the time required to compute the homology and its generators has been evaluated by using the HCC. To this aim, the homology generators have been computed on the base complex, encoded in the root of the HCC. This computation requires between 8.3×10^{-5} and 8.8 seconds depending on the dataset (column *SNF Time* in Table 6.3). Then, all the refinements in the HCC are performed, by applying, when necessary, the refinement of the generators, as described in Section 6.1. This produces the representation of the complex at full resolution together with the homology generators. The total cost of this computation is the sum of the time required to compute the homology of the base complex (column *SNF Time*) and the time needed to fully refine the complex and its generators (column *Tot. Ref. Time*). This takes from a minimum of 0.15 to a

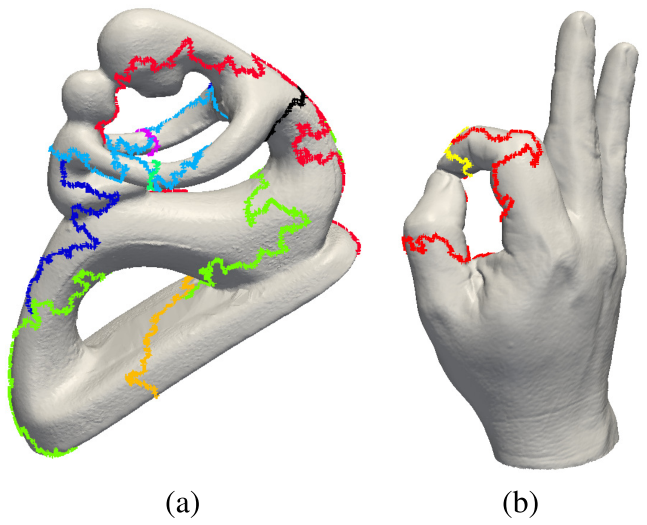


Figure 6.10: The H_1 generators computed on the FERTILITY dataset (a) and on the HAND dataset (b) by fully refining the cell complex.

maximum of 83.3 seconds. Applying the same SNF reduction directly on the original complex, requires about 2.6 hours on a relatively small complex (the dataset GENUS3), while it results in very high computation times for the other datasets.

Figure 6.10 shows the H_1 generators computed on two 2D shapes FERTILITY and HAND and, in Figures 6.12(b) and 6.12(c), we show the H_1 and H_2 generators computed on the 3D SKULL dataset.

Then, an attempt in extracting different representations of the complex by expanding the computed generators at different resolutions has been done. First, the extraction of representations at uniform resolution has been considered: representations obtained from the base complex are extracted by applying approximatively 20%, 50% and 80% of the total possible refinements (column *Uniform Ref.* in Table 6.3). Refinements are forced to be evenly distributed inside the complex in order to obtain a uniformly refined complex. We can notice (see column *Uniform Time*) that the time required depends on the number of refinements performed and is between 0.03 and 7.5 seconds for extraction at 20% resolution and between 0.12 and 69.3 seconds for extraction at 80% resolution.

Then, representations of the complexes have been extracted varying the resolution inside the domain. The objective has been to obtain a cell complex, and the corresponding homology generators, with a maximum resolution only in a neighborhood of a specific homology class, as explained below. This corresponds to computing the generators (H_i) of degree i on the base complex and, by traversing the HCC, to perform only those refinements that involve some i-cell

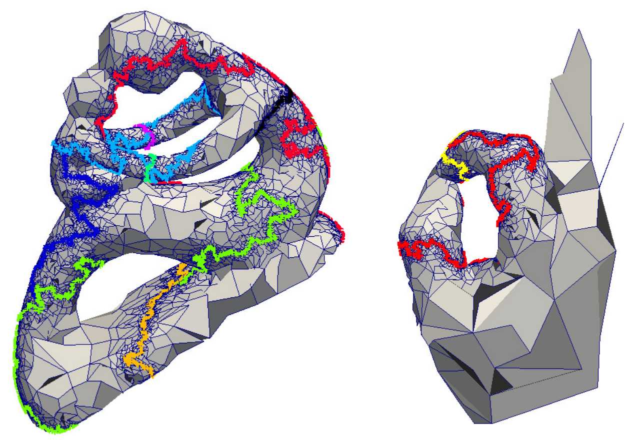

(a) (b)

Figure 6.11: The H_1 generators computed on the FERTILITY dataset and on the HAND dataset. In (a) and (b) the generators obtained by refining the cell complex only in a neighborhood of the generators (selective refinement at variable resolution).

belonging to any generator of degree i and those on which they depend. This kind of selective refinement produces cell complexes with a lower number of cells outside the area around the generators and thus leads to a further saving (15-30%) with respect to extracting generators at maximum resolution. Note that the extraction at variable resolution is a distinctive feature of the HCC which cannot be performed on other hierarchical models. Examples of variable resolution extractions are shown in Figure 6.11 and in Figure 6.12(d).

6.4 Future developments

We have defined a hierarchical model, called *Hierarchical Cell Complex (HCC)*, based on a set of homology-preserving and homology-modifying operators. We have defined the HCC in a dimension-independent way on arbitrary cell complexes. We have implemented a version of the HCC based on the homology-preserving operators and we have demonstrated experimentally its properties. We have also applied the HCC to homology computation implementing an algorithm for computing homology and homology generators with coefficients in $\mathbb{Z}_2$ on the base complex (the coarsest one) and using our model to extract the homology generators at uniform or variable resolution levels.

Our current and future work moves in the same direction. We are interested in enhancing the expressive power of our model by enabling the extraction of shapes at variable resolutions and with variable homology. To do this we are defining a multi-resolution model based on both the homology-preserving and homology-modifying operators. We recall that an homology-

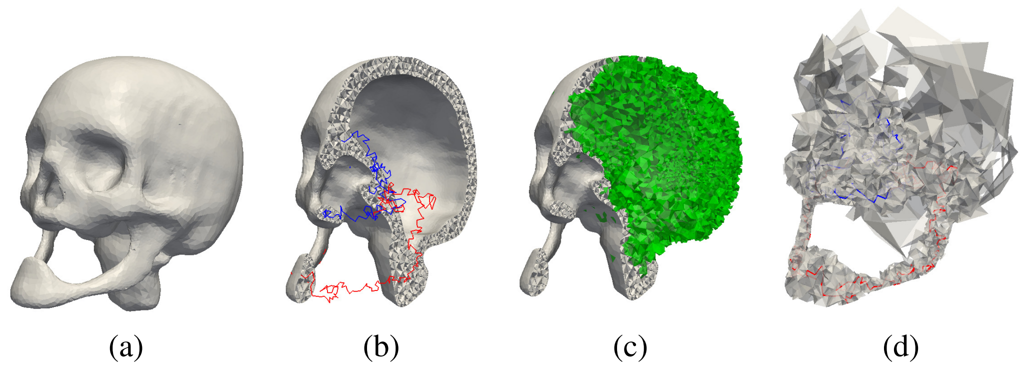


Figure 6.12: The H_1 and H_2 generators computed on the SKULL dataset. In (a) the original dataset, in (b) and (c) the H_1 and H_2 generators computed at full resolution and in (d) the H_1 generators extracted at variable resolution and visualized inside the extracted cell complex.

modifying operator changes the number of cells in a complex Γ plus its Betti number and Euler characteristic. Specifically, a $KiCiCycle$ (***Kill i-Cell and i-Cycle***) operator deletes an i-cell and destroys an i-cycle, thus decreasing the numbers n_i and β_i by one. It is feasible on a cell complex Γ if the co-boundary of the cell to be deleted is empty. Thus, the set of homology-modifying operators deleting on a i-cell from a cell complex are always performed from the cell of higher dimension to the cell of lower dimension.

6.4.1 A proposal for a homology-modifying HCC

The homology-modifying HCC is built from a sequence of simplifications. All the homology-preserving operators are performed first, in order to reduce the complexity of the cell complex Γ as much as possible. Once Γ reaches its coarsest representation Γ', we compute homology and homology generators as usual with the Smith Normal Form (SNF) reduction algorithm [Ago05]. From this point, Γ' can be further reduced applying a homology-modifying operator for each homology generator computed on Γ'.

All the homology-modifying operators are performed reaching the base complex Γ_B (note that some homology-preserving operators could be triggered during this step). Then the homology-modifying HCC structure is reconstructed from the sequence of simplification operators performed. The homology-modifying operators are encoded in DAG nodes. For each $MiCiCycle$ operator, undo of a $KiCiCycle$, introducing an i-cell p and augmenting the i-Betti numbers by one, a new DAG node μ is instantiated; μ will be connected, through a dependency relation, to all the DAG nodes introducing an i-cell in the H_i generator restored by $MiCiCycle$ operator. DAG nodes representing the homology-preserving refinements are encoded as described in Section 6.3.1. In Figure 6.13(a), we show an example of a $M1C1Cycle$ operator applied on a 1-dimensional cell complex. The 1-cells in colored in blue after the refinement are part of the new

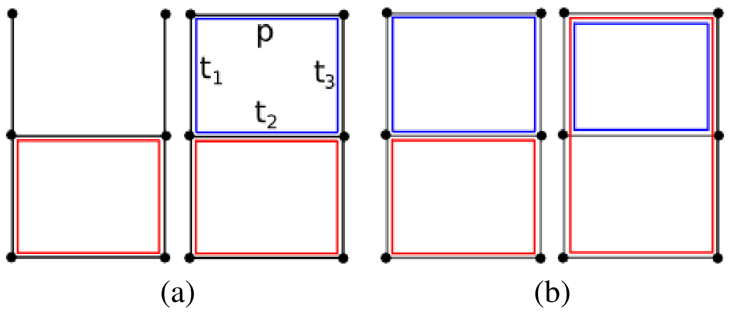


Figure 6.13: (a) $M1C1Cycle(p)$ operator and (b) example of a good ”independent” homology generators (on the left) and of bad generators (on the right).

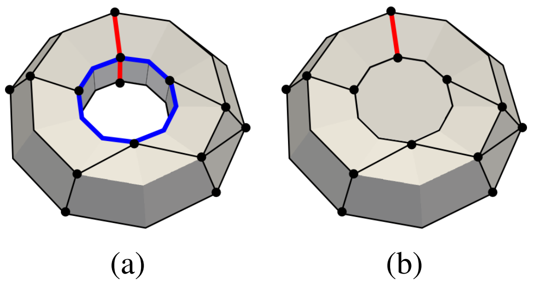


Figure 6.14: (a) example of the original torus extracted with all its homology generators; (b) example of the same torus extracted killing one of its H_1 generators

H_1 generator restored by $M1C1Cycle$ and thus the DAG node (encoding $M1C1Cycle$ operator) will be dependent from all the DAG nodes introducing such cells (t_1, t_2 and t_3).

From the homology-modifying HCC obtained different representations can be extracted refining the base graph G_B. As for the homology-preserving HCC the resolution level is increased by successive refinements based on a geometrical criterion. However, in this model the homology of the extracted cell complex is chosen a priori. For each homology-modifying refinement we chose if enable the update of the corresponding Betti number.

In Figure 6.14(b), we show an example of a torus, extracted from a homology-modifying HCC built from the torus shown in Figure 6.14(a), with a 2-cell killing one of its H_1 generators.

6.4.2 Current and future work

We would like to adapt the HCC model to a class of complexes more suitable for a practical use, the simplicial meshes. We will consider two simplification operators for generating the hierarchical model: *simplex collapse* [KMŚ98], which is an instance of simplification operator $KiC(i+1)C(q,p)$, and *edge contraction*, a widely used operator in mesh processing which has been proven to be homology-preserving [ALS12]. We will evaluate the expressive power of a model based on such operators. Will be fundamental to have a compact representation to overcome the storage costs required encoding massively all the incidence relations among the simplexes of the simplicial complex. In other words, since the homology computations algorithms are well suited to be adapted on simplicial complexes, our effort will concentrate initially on the data structures used to represent such complexes.

We are interested in extracting ”good looking” homology generators during the refinement process. This task can be intuitively described as the computation of homology *local* properties (see [ZC08] for a formal definition). We want to compute a ”good” set of generators representing the homology we have on Γ and we want the smallest cycles as possible to represent each generator. In the left part of Figure 6.13(b) the notion of *good generator* is shown from an intuitive point of view. The generators, depicted in red and blue, are good since they are independent (not self contained). Conversely, the H_1 generators shown in the right part of Figure 6.13(b) are not acceptable from this point of view since the blue generator is contained in the green generator even if the smallest 1-cycles on the cell complex are not. A description for the quality of the homology generators can be found in [CF08] with a method for localizing the homology generators producing cycles as smallest as possible. We are planning to implement such method on our homology-modifying HCC for computing generators on the base complex Γ_B. The set of generators would be refined afterward during the extraction of a refined representation and we are planning to define a method for maintain the good quality during the process.

Chapter 7

Multi-resolution Model for Triangulated Scalar Fields

By using the multi-resolution model for scalar fields described in Chapter 5, we can inspect a scalar field from the point of view of its morphology, but a huge problem in the analysis of a scalar field is the complexity of real datasets and their dimension. In this chapter we present a new multi-resolution model able to extract representations at different level of resolution both the geometry and the morphology of a scalar field. A first attempt of coupling the geometric and morphological part of a scalar field in a multi-resolution model has been done in [DDFMV10]. The model in [DDFMV10] is composed by three interconnected graphs, one representing the modifications of the geometry and the others representing modifications which involve also morphology (see Section 2).

The new model, called *Multi-Tessellation based on Forman Gradient* ($MTFG$), is a multi-resolution model for scalar fields $\mathbb{M}_\Sigma = (\Sigma, f)$ defined on a triangle mesh Σ on which a Forman gradient V has been computed. The structure of an $MTFG$ is a single hierarchy described as a DAG with two types of nodes:

- geometric nodes, denoted $Node_g$, representing modifications on the mesh of Σ
- topological nodes, denoted $Node_t$, representing modifications on V (or, alternatively, on the topology of the ascending/descending Morse complexes that can be computed on V)

The $MTFG$ nodes represent the available refinement modifications for the geometry or the morphology of $\mathbb{M}$, which are undo operators with respect to simplifications performed during the construction of $MTFG$.

In Section 7.1, we describe the operators used for simplifying $\mathbb{M}$ both from the geometric and topological point of view and required to build an $MTFG$. For the geometric simplification we have been inspired by the geometric multi-resolution model defined in [MDDF$^+$08] called the Multi-Tessellation; We will use a well known simplification operation for simplicial complexes which deletes an edge, collapsing one of its vertices in the other one, the *edge-collapse*. For the topological simplifications, we use the same $remove$ operators used for the MMC (see Section 5) . We define a $remove$ operator on a Forman gradient V and we will describe the resulting updates on V in Section 7.1.2. In Section 7.2 we define the $MTFG$ as the model, obtained from a sequence of simplifications (geometric and topological), composed of a set of refinement modifications and a dependency relation among such modifications. Moreover we describe the encoding structure defined for the $MTFG$ and we will compare our model to the one presented in [DDFMV10] highlighting similarities and differences.

7.1 Geometric and topological update operators

In this section, we describe the update operators we use to modify the geometry and the morphology of a scalar field $\mathbb{M}_\Sigma = (\Sigma, f)$ defined on a triangle mesh Σ. In Subsection 7.1.1, we describe the modification operators used to simplify and refine the triangle mesh, namely *edge-collapse* and *vertex split* operators, respectively. In Subsection 7.1.2, we present the modification operators for simplifying and refining the Morse complexes computable on V. Such operators are equivalent to $remove$ and $insert$, described in Section 4. We will describe them in terms of updates on a Forman gradient.

7.1.1 Geometric update operators

We use two operators to modify the geometry of the triangle mesh Σ: the *edge-collapse* operator is used during the simplification step to coarsen Σ, while the *vertex-split* operator is encoded in the $MTFG$ and used to refine Σ.

7.1.1.1 Edge-collapse operator

An *edge-collapse* is a well known simplification operator for simplicial complexes. It is a local update that acts on a triangle meshes by contracting an edge e, with endpoints v_1 and v_2, to one of its endpoints (i.e., v_1), sometime called half-edge collapse.

According to the notation illustrated in Figure 7.1(a) we denote as:

- v_1 the vertex deleted by the edge-collapse,
- v_2 the vertex surviving after the edge-collapse,

- $e = (v_1, v_2)$ the edge collapsed,
- $t^{(l)}$ the triangle on the left of edge e by considering e as directed from v_1 to v_2,
- $t^{(r)}$ the triangle on the right of e,
- $v_3^{(l)}$ and $v_3^{(r)}$ the vertices of $t^{(l)}$ and $t^{(r)}$, respectively, different from v_1 and v_2,
- $t_1^{(l)}$ and $t_1^{(r)}$ the triangles adjacent to $t^{(l)}$ and $t^{(r)}$, respectively, on the edge opposite to v_2,
- $t_2^{(l)}$ and $t_2^{(r)}$ the triangles adjacent to $t^{(l)}$ and $t^{(r)}$, respectively, on the edge opposite to v_1.

The edge-collapse, denoted $collapse(v_1, v_2)$, applied on the edge $e = (v_1, v_2)$, removes edge e, vertex v_2 and triangles $t^{(l)}$ and $t^{(r)}$ from Σ (see Figures 7.1(a) and 7.1(b)).

As a consequence,

- $t_1^{(l)}$ become adjacent to $t_2^{(l)}$;
- $t_1^{(r)}$ become adjacent to $t_2^{(r)}$;
- all the triangles incident in v_1 become incident in v_2;

Since edge-collapse turned to be useful in various applications involving topology, it was observed that contracting edges in a simplicial mesh could change its homology groups. In [DEGN99] the so called *link condition* is defined as condition for an edge-collapse to preserve homology.

Definition 7.1.1. *An edge* $e = (v_1, v_2) \in \Sigma$ *satisfies the* **link condition** *if and only if* $Lk(v_1) \cap Lk(v_2) = Lk(e)$.

In [DHKS13] a weaker and more local variant of the *link condition* is defined, called *p-link condition*. Since our model works on a triangle mesh, we require an edge e, involved in a $collapse$ operator, to satisfy the *link condition* only.

7.1.1.2 Vertex-split operator

A *vertex-split* is defined as the undo of an edge-collapse simplification. It is a local update that acts on triangle meshes by expanding a vertex v_1 into an edge e having its endpoints at v_1 and v_2.

The *vertex-split*, denoted $split(v_2)$, is *feasible* on the vertex v_1 if:

- vertex v_2 is in Σ;

- all the vertices, adjacent to v_2 when the inverse $collapse(v_2, v_1)$ was applied, are adjacent to v_1 in Σ.

Vertex-split operator introduces vertex v_2 in Σ as edge $e = (v_2, v_1)$ and the two triangles $t^{(l)}$ and $t^{(r)}$ incident in e. As a consequence,

- $t_1^{(l)}$ is not adjacent to $t_2^{(l)}$ anymore and both $t_1^{(l)}$ and $t_2^{(l)}$ become adjacent to $t^{(l)}$;
- $t_1^{(r)}$ is not adjacent to $t_2^{(r)}$ anymore and both $t_1^{(r)}$ and $t_2^{(r)}$ become adjacent to $t^{(r)}$;
- all the triangles incident in v_1 become incident in v_2.

7.1.1.3 Edge-collapse and vertex-split on a Forman gradient

Edge-collapse and Vertex-split operators performed on the triangle mesh Σ reduce or augment the number of simplexes of Σ. Then, we have to modify also the pairings in the Forman gradient, computed on $\mathbb{M}$, accordingly. The key idea of a gradient update is to locally modify V without modifying the number of critical simplexes and maintaining the gradient flow.

Intuitively, the number of critical simplexes is maintained if the edge-collapse does not remove simplexes that are critical and if all the paired simplexes involved are still paired after the *edge-collapse*. Let Σ_{vp_i} be the set of simplexes in a V-path vp_i, and let $vp_{(i+1)}$ be the V-path, corresponding to vp_i, after the *edge-collapse*. Let Σ_{rem} the set of simplexes removed by the *edge-collapse*. Them, the gradient flow is maintained if $\Sigma_{vp_{(i+1)}} = \Sigma_{vp_i} - \Sigma_{rem}$ for each V-path vp_i. In other words, the gradient flow is maintained if all the simplexes, inside a V-path involved in the *edge-collapse*, are still in the V-path after, excepting the simplexes removed by the *edge-collapse*. Specifying some preconditions for V in the neighborhood of an edge collapsed we can obtain these properties.

An edge-collapse, $collapse(v_1, v_2)$, is feasible if:

- $t^{(l)}$ and $t^{(r)}$ are not critical,
- all the edges on $t^{(l)}$ and $t^{(r)}$ are not critical,
- v_1 has more than three incident triangles,
- v_1 is paired with edge e in V,
- $v_3^{(l)}$ is not paired in V with edge $(v_3^{(l)}, v_1)$,
- $v_3^{(r)}$ is not paired in V with edge $(v_3^{(r)}, v_1)$.

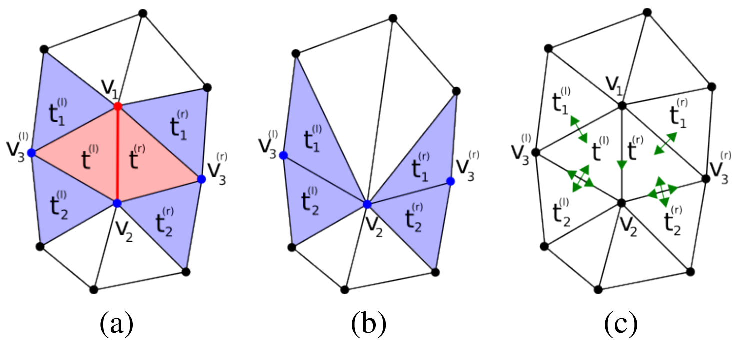


Figure 7.1: (a) the simplexes involved in a edge-collapse. (b) result of the edge collapse and in (c) the gradient arrows (green) that are valid for the edge-collapse.

In Figure 7.1(c) we show all the possible gradient arrows that can appear in the neighborhood of a feasible edge-collapse. Gradient arrows not showed in the figure are pairing simplexes not involved in the edge collapse or are not valid for the collapse operator. The allowed gradient configurations guarantee another nice property during a simplification/refinement operation. When an edge-collapse (or a vertex split) is performed, the modifications on the local frame of the Forman gradient V are entirely and unambiguously determined by the gradient arrows present in V before the operation. In Section 7.2.2 we will show that no information are stored about how to update the Forman gradient when performing a vertex-split encoded in the multi-resolution model.

The edge-collapse applied on edge e simplifying the simplicial complex locally, as described in Subsection 7.1.1.1. We have to update the gradient encoded in $t_1^{(l)}$ and $t_1^{(r)}$ in the worst case. Since the updates required are symmetrical on the left and on the right part, with respect to the collapsed edge, we will describe them for the left side only.

The updates on V depend on the edges $(v_3^{(l)}, v2)$ and $(v_3^{(l)}, v1)$. If these edges are both paired with a triangle then no updates are required on V, (see Figure 7.2(a) and 7.2(b)). Otherwise, if the edge $(v_3^{(l)}, v1)$ is paired with a vertex (v_1 or $v_3^{(l)}$) this gradient pair must be updated in the *local frame* codified in $t_1^{(l)}$, (see Figure 7.2(c) and 7.2(d)).

The updates on the Forman gradient V during a vertex-split are handled in a similar fashion not changing the number of critical simplexes and maintaining the gradient flow. A vertex-split will introduce a vertex v_2 as well as a new edge e and the two incident triangles $t^{(l)}$ and $t^{(r)}$ (see Figure 7.3(b)).

The Forman gradient V has to be modified in the neighborhood of the simplexes introduced. Similarly to the collapse, we can unambiguously determine how to extend the gradient on the new simplexes by simply knowing the gradient defined on the simplexes of simplicial mesh Σ.

Updates required on the Forman gradient V to obtain V' are the following (as before we will

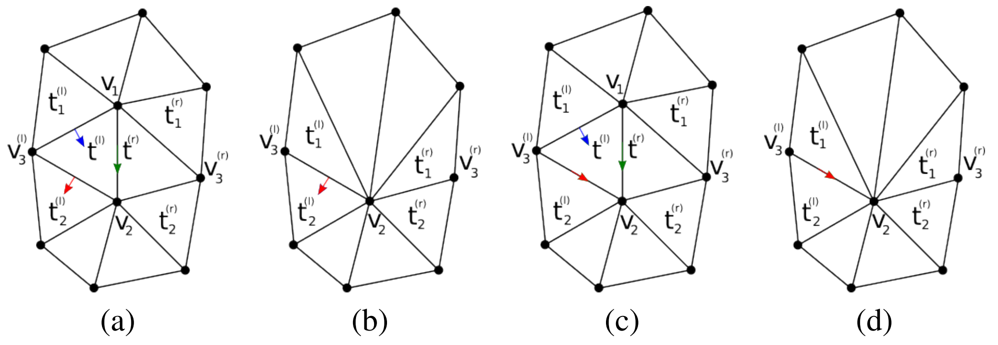


Figure 7.2: In (a) and (b) a gradient configuration that does not require an update of V during an edge collapse is shown. Triangle t_1^{left} has an arrow pointing the adjacent triangle before (blue arrow in (a)) and after (red arrow in (b)) the edge collapse. On the contrary, if the gradient configuration is different (in (c)) it loses the arrow pointing the adjacent triangle after the edge collapse (see (d))

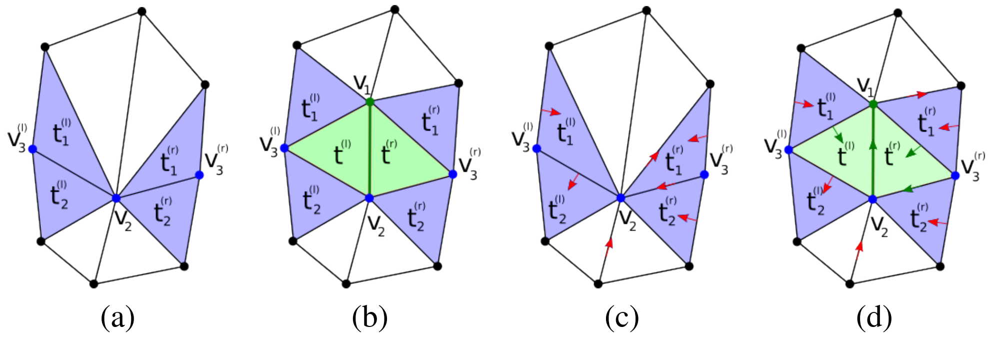


Figure 7.3: In (a) is shown a set of simplexes before the edge expansion showed in (b). In (c) and (d) the same edge expansion is illustrated showing the updates required on the Forman gradient with two different configurations on the left and on the right of the introduced edge.

describe the updates only for the left side):

- new edge edge e is paired in V' with v_1 or v_2. If v_2 is paired with an edge that will be redirected to v_1, than e is paired with v_2 otherwise it is paired with v_1. (See Figure 7.3(c) and (d));
- if edge $(v_3^{(l)}, v_2)$ is paired with v_2 or $v_3^{(l)}$, then edge $(v_3^{(l)}, v_1)$ is paired with $t^{(l)}$;
- if edge $(v_3^{(l)}, v_2)$ is paired with $t_2^{(l)}$, then edge $(v_3^{(l)}, v_1)$ is paired with $t^{(l)}$;
- if edge $(v_3^{(l)}, v_2)$ is paired with $t_1^{(l)}$, then edge $(v_3^{(l)}, v_2)$ is paired with $t^{(l)}$ and edge $(v_3^{(l)}, v_1)$ will be paired with $t_1^{(l)}$.

7.1.2 Update operators for the Forman gradient

In [For98] the notion of cancellation has been extended to discrete Morse theory. If p and q are two critical cells, in a cell complex Γ, with $dim(p) = dim(q) + 1$ and if there is exactly one gradient path from the boundary of p to q, then p and q can be canceled. We will describe simplifications as performed on a pair of critical simplexes of a simplicial mesh Σ even if all the following results hold for cell complexes.

We use two operators to modify the Forman gradient V defined on Σ, the *remove* operator is used during the simplification step to reduce the number of critical simplexes in V while the *insert* operator is encoded in the $MTFG$ and used to refine V. Both operators have been defined for the modification of Morse complexes and described in Section 4. In the following we will describe them in terms of modifications on a Forman gradient V computed on a triangle mesh.

7.1.2.1 Simplification operators on a Forman gradient

A $remove_{i,i+1}(q, p, p')$ applied to a discrete gradient field V removes a critical i-simplex q and a critical $(i + 1)$-simplex p if and only if

- p and q are connected through a unique separatrix V-path,
- q is connected through a separatrix V-path at most to another $(i + 1)$-saddle p' different from p.

The effect of a removal $removal_{i,i+1}(q, p, p')$ on V is to remove p and q from the set of critical simplexes and to reverse the gradient arrows on the path between p and q. As a consequence, the separatrix V-path starting from p' and ending in q before the simplification will be extended to

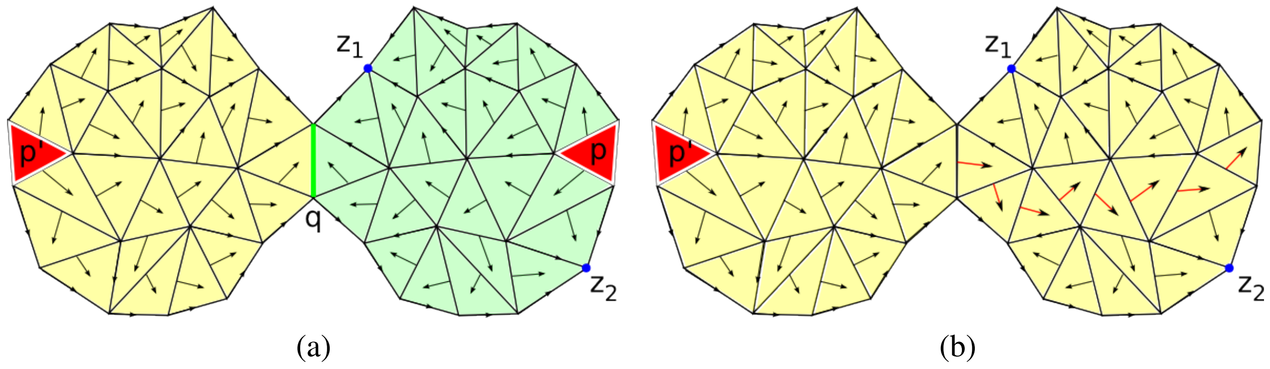


Figure 7.4: Effect of a $removal_{1,2}(q, p, p')$ on a Forman gradient V defined on a 2D simplicial complex. (a) the original V with the two descending 2-cells, defined in green and yellow, corresponding to maximum p and p' respectively; (b) V is updated reversing the V-path (red arrows). The descending 2-cells change accordingly and the green cell is merged in the yellow one.

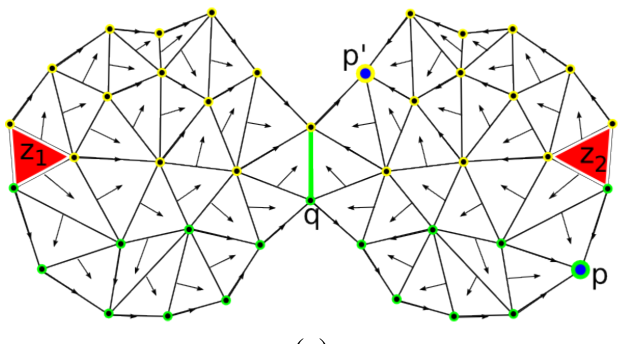
critical i-simplexes, indicated with r_j, previously connected with p. Removing the two critical simplexes p and q the separatrix V-path connecting p with the z_k critical $(i-1)$-simplexes become normal V-paths.

Let us consider for example a $removal_{1,2}(q, p, p')$ on a triangle mesh; q is a critical edge and p and p' are two critical triangles. Starting from q the separatrix V_2-path connecting q to p is reversed. As a consequence p and q are no more critical simplexes. In Figure 7.4 is presented the effect of a $remove_{1,2}(q, p, p')$ on a Forman gradient V. In Figure 7.4(a) the original Forman gradient F is shown overlayed with the two 2-cells forming the descending Morse complex colored in yellow and green for p' and p, respectively. In Figure 7.4(b) the Forman gradient obtained after the $remove_{1,2}(q, p, p')$ is shown. The critical simplexes p and q are removed from V and separatrix V_2-path is reversed (red arrows). As a consequence the two 2-cells are merged. It can be noticed how, by reversing the V_2-path, the V-paths starting from p' fully cover the mesh.

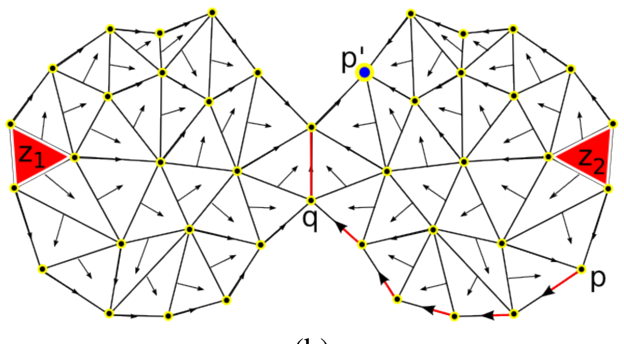
A $remove_{i,i-1}(q, p, p')$, applied on a discrete gradient field V, removes a critical i-simplex q and a critical $(i-1)$-simplex p if and only if

- p and q are connected through a unique separatrix V-path
- q is connected through a separatrix V-path with at most another $(i-1)$-saddle p' different from p.

The effect of a removal $remove_{i,i-1}(q, p, p')$ on V is to remove p and q from the set of critical simplexes and to reverse the gradient arrows on the path between p and q. As consequence of these updates, the separatrix V-path starting from p' and ending into q before the simplification



(a)



(b)

Figure 7.5: Effect of a $removal_{1,2}(q, p, p')$ on a Forman gradient V defined on a 2D simplicial complex. In (a) the original gradient V with the two ascending 2-cells, defined in green and yellow, corresponding to the minimum p' and p respectively. In (b) V is updated reversing the V-path (red arrows). The ascending 2-cells change accordingly and the green cell is merged in the yellow one.

will be extended to the r_j critical $(i-1)$-simplexes previously connected with p. Removing the two critical simplexes p and q the separatrix V-path connecting p with the z_k critical $(i+1)$-simplexes become normal V-paths.

In Figure 7.5, an example of a $remove_{1,0}(q, p, p')$ on a Forman gradient F is illustrated. In (a) the original Forman gradient F is shown overlayed to the two 2-cells forming the ascending Morse complex colored in yellow and green for p' and p, respectively. In (b) is shown the Forman gradient obtained after the $remove_{1,0}(q, p, p')$. The critical simplexes p and q are removed from F and the separatrix V_1-path is reversed (red arrows). As a consequence the two 2-cells are merged. It can be noticed how, reversing the V_1-path, the V-paths starting from p' fully cover the mesh.

7.1.2.2 Refinement operators on a Forman gradient

An $insert_{i,i+1}(q, p, p')$, undo of a $remove_{i,i+1}(q, p, p')$, introduces in the Forman gradient two critical simplexes. The feasibility condition for an $insert_{i,i+1}(q, p, p')$ operator is the presence of the critical simplexes $p' \cup R \cup Z \cup S$ in F,

- critical simplexes $r_j \in R$ must be connected with a separatrix V_{i+1}-path with p',
- critical simplexes $z_h \in Z$ must be connected with a V_i-path with the simplex q that will be critical,

- critical simplexes $s_k \in S$ must be connected with a V_{i+2}-path with the simplex p that will be critical.

The updates required by the operator will restore the two critical simplexes p and q by reversing only the arrows in the V-path between them.

Dually, an $insert_{i,i-1}(q, p, p')$, undo of a $remove_{i,i-1}(q, p, p')$, introduce in the Forman gradient two critical simplexes. The feasibility condition for an $insert_{i,i-1}(q, p, p')$ operator is the presence of the critical simplexes $p' \cup R \cup Z \cup S$ in F,

- critical simplexes $r_j \in R$ must be connected with a separatrix V_i-path with p',
- critical simplexes $z_h \in Z$ must be connected with a normal V_{i+1}-path with the simplex q that will be critical,
- critical simplexes $s_k \in S$ must be connected with a normal V_{i-1}-path with the simplex p that will be critical.

The updates required by the operator will restore the two critical simplexes p and q by reversing only the arrows in the V-path between them.

To successfully apply an $insert_{a,b}(q, p, p')$ operator in any dimension, the V-path between q and p must be known. This path is always a sub-path of the separatrix V-paths connecting p' with one of the nodes $r_j \in R$ passing by q and p. Thus, starting from p and traversing the separatrix V-path vp_σ, between p' and r_j, until reaching q, the separatrix V-path between q and p can be reconstructed in linear time in the number of simplexes composing vp_σ.

In Figure 7.6 an example of an $insert_{1,2}(q, p, p')$ is shown. In Figure 7.6(a) the original Forman gradient V is illustrated. All the critical simplexes required for the insert operator (namely p', z_1 and z_2) are in V. Critical 2-simplex p' is connected through a V-path, passing through q, to p and the two critical 0-simplexes z_1 and z_2 are connected through a V-path to q; thus, the feasibility conditions are satisfied. In Figure 7.6(b) Forman gradient V obtained and the corresponding descending Morse complex Γ_d are shown. The two critical simplexes p and q are introduced in V, and the V-path connecting them is reversed (red arrows) by restoring the 2-cell corresponding to p.

7.2 Multi-tessellation based on Forman gradient

A *Multi-Tessellation based on Forman Gradient* ($MTFG$), computed from a scalar field $\mathbb{M}_\Sigma = (\Sigma, f)$ and a Forman gradient V on it, is defined as a quadruple $(\Sigma_B, V_B, \mathcal{M}, \mathcal{R})$ where:

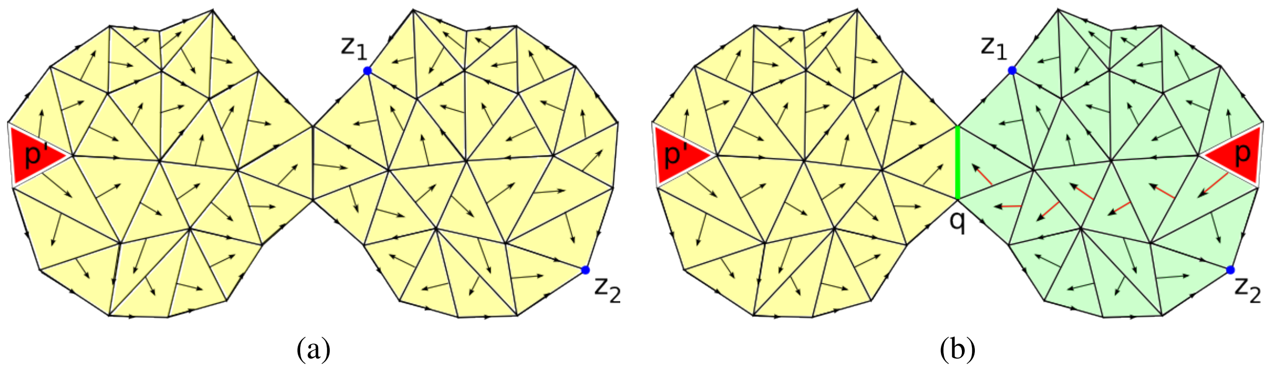


Figure 7.6: Effect of an $insert_{1,2}(q, p, p')$ on a Forman gradient V define on a two-dimensional simplicial complex. In (a) the original V with the ascending 2-cell corresponding to p' colored in yellow. In (b) V is updated restoring the separatrix V-path (red arrows) and reintroducing p and q as critical simplexes.

- Σ_B, called base mesh, is the triangle mesh obtained by simplifying Σ;
- V_B, called base Forman gradient, is the coarsest Forman gradient obtained by simplifying V;
- $\mathcal{M} = (\mathcal{M}_\Sigma, \mathcal{M}_V)$ is the set of refinement modifications which are inverse of the simplifications performed on the triangle mesh ($\mathcal{M}_\Sigma$) and on the Forman gradient ($\mathcal{M}_V$);
- $\mathcal{R}$ is the dependency relation between the modifications in $\mathcal{M}$.

The dependency relation between two modifications is defined differently if they are both in $\mathcal{M}_\Sigma$, both in $\mathcal{M}_V$ or mixed. Thus, we defined the set of dependency relations $\mathcal{R}$ as a triple $(\mathcal{R}_g, \mathcal{R}_t, \mathcal{R}_m)$ where:

- $\mathcal{R}_g$ are the (geometric) dependency relations involving two modifications in $\mathcal{M}_\Sigma$;
- $\mathcal{R}_t$ are the (topological) dependency relations involving two modifications in $\mathcal{M}_V$;
- $\mathcal{R}_m$ are the (mixed) dependency relations involving a modification in $\mathcal{M}_\Sigma$ and one in $\mathcal{M}_V$.

We recall that a vertex-split in $\mathcal{M}_\Sigma$ expands a vertex v_1 into an edge e having its endpoints at v_1 and v_2 and it is *feasible* on vertex v_1 if vertex v_2 is in Σ all the vertices, adjacent to v_2 when the dual $collapse(v_2, v_1)$ was applied, are adjacent to v_1 in Σ (see Section 7.1.1.1). An $insert$ operator, undo of a $remove$ operator, introduce into the Forman gradient V two critical simplexes. The feasibility condition for an $insert_{a,b}(q, p, p')$ operator is the presence of the critical simplexes $p' \cup R \cup Z \cup S$ in V (see Section 7.1.2).

Definition 7.2.1. *A modification* $split(v_1) \in \mathcal{M}_\Sigma$ directly depends *from another modification* $split(v_1') \in \mathcal{M}_\Sigma$ *if the vertex introduced by* $split(v_1')$ *is in the set of vertices required by* $split(v_1)$*. The resulting dependency relation is in* $\mathcal{R}_g$*.*

Definition 7.2.2. *A modification* $insert(q, p, p') \in \mathcal{M}_V$ directly depends *from another modification* $insert(q_1, p_1, p_1') \in \mathcal{M}_V$ *if* $insert(q_1, p_1, p_1')$ *introduces a critical simplex included in the sets of critical simplexes* $p' \cup R \cup Z \cup S$ *of* $insert(q, p, p')$*. The resulting dependency relation is in* $\mathcal{R}_t$*.*

Definition 7.2.3. *A modification* $insert(q, p, p') \in \mathcal{M}_V$ directly depends *from a modification* $split(v_1) \in \mathcal{M}_\Sigma$ *if* $split(v_1)$ *introduce a simplex that will be transformed in critical by* $insert$*-*(q, p, p')*. The resulting dependency relation is in* $\mathcal{R}_m$*.*

Note that modifications in $\mathcal{M}_\Sigma$ never depend from modifications in $\mathcal{M}_V$. Intuitively, this means that the model is always able to refine the geometry of Σ independently of the critical simplexes in V. Dependency relation $\mathcal{R}$ is a partial order relations since a simplex is never introduced twice in Σ and a critical simplex is never introduced twice in V. The dependency relation between refinement modifications is the transitive closure of the direct dependency relation.

Similarly to what we have described in Sections 5 and 6, a large number of meshes at intermediate resolution can be obtained by applying a sequence of refinement modifications in $\mathcal{M}$ to the base mesh Σ_B or to the base gradient V_B. The notion of *feasible sequence* of modifications, *front complex*, *interchangeable* modifications and *closed set* of modifications are the same as for the MMC and HCC multi-resolution models.

From an $MTFG$ it is thus possible to dynamically extract representations of the original triangle mesh Σ and of the corresponding Forman gradient V at uniform and variable resolutions. A *selective refinement query* on the $MTFG$ consists of extracting from a mesh with the *minimum* number of simplexes and a gradient with the *minimum* number of critical simplexes, satisfying some application-dependent criterion. Differently from the MMC and HCC, two different criteria, τ_Σ and τ_V, are defined on the modifications in $\mathcal{M}_\Sigma$ and $\mathcal{M}_V$ respectively,. The selective refinement query consists of extracting from the $MTFG$ an intermediate mesh of minimum size among the meshes encoded in the $MTFG$ that satisfies τ_Σ and extracting an intermediate Forman gradient V that satisfies criterion τ_V.

7.2.1 Building an $MTFG$

An $MTFG$ is built from a triangle mesh Σ and a Forman gradient V defined on Σ by alternating sequences of geometric and topological simplifications.

All the feasible edge-collapse operators are performed (see Section 7.1.1.1). Edge-collapse simplifications are the first tool we use to reduce the resolution of Σ and, consequently, its storage cost. The definition of a simplification algorithm is a fundamental step for the construction of a multi-resolution model. As discussed in Section 6.3.1.2, different simplification algorithms lead to different hierarchical structure of the models. For the $MTFG$, we selected a batch simplification, maximizing the number of independent simplifications performed (and thus of independent refinements) in order to maximize the number of representations that can be extracted from the $MTFG$.

When all feasible edge-collapses have been performed, the queue of the topological simplifications (see Section 7.1.2) is constructed by only a percentage of all the feasible topological simplifications. This percentage can be a user-defined parameter. We have fixed this parameter based on the persistence range. At each iteration all the simplifications having a persistence value lower than the 10% of the maximum persistence value are performed. Performing some topological simplifications will unlock new edge-collapses (since some critical simplexes are removed and some arrows are flipped). Similarly to the geometric simplifications, we want to perform as much topological simplifications as possible by distributing them throughout the dataset. To achieve this result, we use a support data structure encoding the incidence graph of the critical points of the scalar field (the MIG described in Section 3.1). The MIG is extracted from V and kept up to date during the undergoing of topological simplifications as described in Section 4.2. Note that the MIG is used here only for the seek of efficiency. The simplification process could be performed by computing, after each simplification, all the possible pairs of critical simplexes and choosing among them the best candidate for the simplification. This interchange of geometric and topological simplifications continues until no more simplifications are available.

Once all the simplifications have been performed the base mesh Σ_B and the base Forman gradient V_B are obtained as well as to all the modifications $\mathcal{M}_\Sigma$ and $\mathcal{M}_V$, undo of simplifications performed on Σ and V respectively. Σ_B and V_B are stored in the root of a DAG structure. Then the $MTFG$ is built as a DAG where the nodes corresponds to performable modifications and the arcs to the dependency relation between two nodes.

7.2.2 Encoding an $MTFG$

We distinguish among two types of $MTFG$ nodes, *geometric nodes* ($Node_g$) and *topological nodes* ($Node_t$).

A *geometric nodes* ($Node_g$) encodes the vertex-split refinement, undo of an edge-collapse $collapse$-(v_2, v_1). $Node_g$ representing a vertex-split $split(v_1)$ encodes:

- a vector vv of vertices adjacent to v_2 when the $collapse(v_2, v_1)$ was applied;
- the coordinates of the new vertex inserted v_2;
- the index v of the vertex v_1 that will be on the boundary of the new edge with v_2;

- the set of $Node_g$ from which this operation depends;
- a Boolean value indicating whether the split has been applied or not.

Vertices in vv are stored in a specific order such that in the first two positions of the vector will be stored the two vertices $v_3^{(l)}$ and $v_3^{(r)}$ (see Section 7.1.1.1). Memory occupied by vertex v_2 instead is used differently before and after the expansion is executed. If the expansion is not yet applied integer v indicates the index of vertex v_1. After the expansion, its value will be replaced with the index of the new vertex v_2. This is an implementation trick to have access in constant time to a vertex during the extraction of the front complex Σ'.

All the indices of a vertex in the $MTFG$ are encoded as signed integers. We have to distinguish between vertices encoded in the base mesh Σ_B and vertices that will be introduced by some $Node_g$. To do this, we store all $Node_g$ in consecutive memory cells. Then, a vertex is represented in the $MTFG$ with:

- a negative integer $-i$, if the index correspond to the i-th vertex in Σ_B,
- a positive integer i, if the index corresponds to the vertex introduced by the i-th $Node_g$. If the expansion encoded in $Node_g$ has been applied, v represent the actual index of the vertex in the front complex Σ'.

In this way we can reorganize the indexes of the vertices in the Indexed data structure with adjacency (IA data structure), encoding the triangle mesh (see Section 3.3), only at the end of the selective refinement.

A *topological DAG node* ($Node_t$) encodes the refinement modification, undo of a $remove_{a,b}$-(q, p, p'), applied on V during the simplification step. $Node_t$ representing an $insert_{a,b}(q, p, p')$ operator encodes:

- the set of vertex indices indicating the simplex p that will be critical (critical triangle if $b = 2$, critical point if $b = 1$);
- the pair of vertex indices indicating the edge q that will be critical;
- a set of $Node_t$ from which this operation depends;
- a set of $Node_g$ from which this operation depends;

We can estimate the storage cost for the $MTFG$ by considering the storage cost of the IA data structure for the base mesh Σ_B, the storage cost of the base Forman gradient V_B, the storage cost for each single node and arc of the DAG encoding the $MTFG$. We are considering the storage cost of a pointer as 4 $bytes$.

A geometric $MTFG$ node requires 4 bytes for each vertex in vv, 32 bytes for the coordinates of v_2 (3+1 floats), 4 bytes for vertex v_1 and 4 bytes for each pointer to a $Node_g$ and 1 byte for the boolean. In total $37 + 4|vv| + 4|Node_g|$ bytes.

A topological $MTFG$ node occupies 4 bytes for each vertex used to indicate the critical simplexes (depending from the operation we need 6 or 4 vertices), 4 bytes for each $Node_t$ and 4 bytes for each $Node_g$. Thus, overestimating the amount of vertices required, $24+4|Node_t|+4|Node_g|$, we can overestimate the number of $Node_g$ (one for each vertex) and thus it will cost in total $48 + 4|Node_t|$ bytes.

Σ_B, as described in [Nie97], occupies $35|V| + 24|T|$ where $|V|$ and $|T|$ represent the number of vertices and triangles, respectively, in Σ_B.

7.2.3 Selective refinement on an $MTFG$

The purpose of combining geometry and morphology in a multi-resolution is the extraction of geometrical representations that conforms with the level of detail required by the simplified morphology.

On the $MTFG$ hierarchy we can perform selective refinement queries setting the desired resolution for the topology of $\mathbb{M}$ and for Σ. Performing a selective refinement query requires a *Level-Of-Detail (LOD)* criterion τ for each component of $\mathcal{M} = (\mathcal{M}_\Sigma, \mathcal{M}_V)$.

We consider a criterion τ_Σ satisfied by triangle mesh Σ_B and a criterion τ_V for the base Forman gradient V_B. By varying τ_Σ, the extracted scalar field will have as Forman gradient V_B (thus the same morphology of the scalar field encoded on the base triangle mesh) and as triangle mesh the mesh Σ', extracted from Σ_B, with resolution τ_Σ. By varying τ_V instead, the extracted scalar field will have a Forman gradient V', extracted from V_B, with resolution τ_V and, as underlying geometry, the triangle mesh Σ' extracted from Σ_B on which the modifications in $\mathcal{M}_\Sigma$ triggered by the modifications in $\mathcal{M}_V$ are applied.

When both τ_Σ and τ_V vary, the extraction is performed on the geometry of $\mathbb{M}$ first. Once the triangle mesh Σ' satisfying τ_Σ has been extracted the morphological modifications are considered and the Forman gradient V' satisfying τ_V is extracted. Note that during the extraction of V', Σ' could be refined further if some morphological modification requires a higher resolution level on Σ'. Both extractions start from the root of the DAG structure. All the $MTFG$ nodes satisfying the criteria are applied as well as the DAG nodes from which they depend.

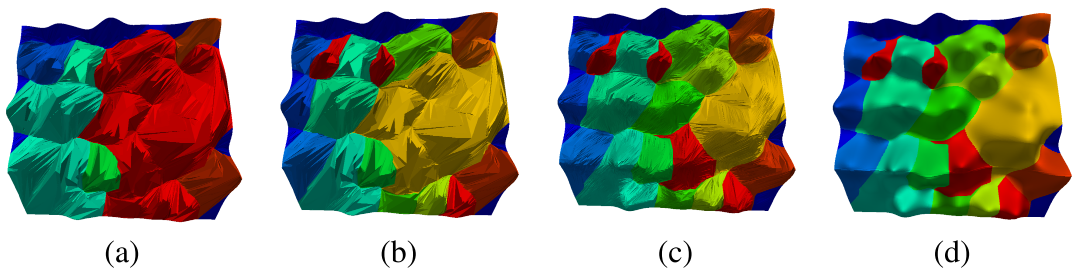


Figure 7.7: Extraction of representations of the EGGS dataset at different resolution levels. (a-b) Topological refinement are executed without changing the underlying geometry. (c) Also the geometric representation is refined until the top mesh is reached (d). Figures illustrate the top cells of the descending Morse complex with different colors.

Thanks to the DAG structure of the multi-resolution model also *extractions at variable resolution* can be performed. These type of queries, for both τ_Σ and τ_V, consider a Region Of Interest (ROI), where different resolution levels are imposed with respect to the remaining parts of the dataset. The selective refinement query is performed on the $MTFG$ similarly to extractions at uniform resolution. The DAG is traversed starting from the root and only the $MTFG$ nodes, representing modifications inside the window query, are performed. Thus, we need a spatial representation for both the modifications in $\mathcal{M}_\Sigma$ and $\mathcal{M}_V$. A geometrical modification in $\mathcal{M}_\Sigma$ is localized considering the coordinates of the vertices involved. A topological modification in $\mathcal{M}_V$ is localized considering the coordinates of the vertices forming the two critical simplexes involved. The region of interest can be equivalent for both the modification types or not.

7.2.3.1 Some preliminary results

We have implemented the $MTFG$ and we are currently testing our implementation working with synthetic and real datasets. We collect here only preliminary results obtained working with the synthetic EGGS datasets. In Figure 7.7, we show the results obtained from a selective refinement query executed on the EGGS dataset. In Figures 7.7 (a) and (b), we show the results obtained by augmenting, progressively, the topological resolution of the dataset. In Figure 7.7 (c) and (d), we have augmented the topological resolution further as well as the geometric resolution until the original complex (Figure 7.7(d)) is extracted. Note that we are showing only one of the possible features that can be extracted from the Forman gradient. For all these queries the topological extraction is performed at first. When the Forman gradient V has been extracted at the desired resolution the geometric refinements are executed. Note that in Figure 7.7, the boundaries of the terrain are excluded from the geometric simplification to overcome the collapse of the terrain mesh.

In Figures 7.8 (a) and (b), we show the results obtained from a selective refinement query concentrating the geometric resolution only in a subset of the domain. In Figure 7.8 (a), the geometric

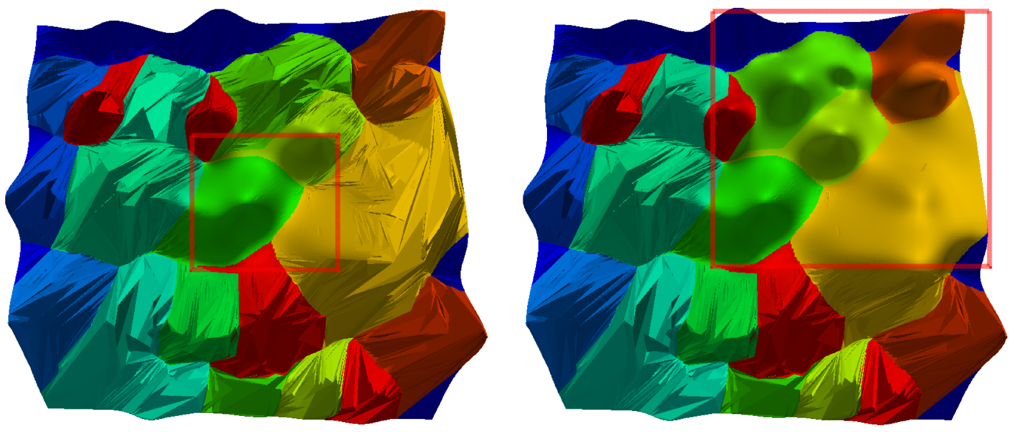


Figure 7.8: (a-b) Extraction of different representations obtained through a selective refinement query at variable resolution. The triangle mesh Σ is extracted at full resolution only inside the red box. Figures illustrate the top cells of the descending Morse complex with different colors.

resolution is incremented inside the red box, and in Figure 7.8 (b) the dimensions of the query box are augmented. Also in these kind of queries the extraction of the Forman gradient V is performed, followed by increasing in the geometric resolution. Working with real datasets we will aim more to the topological extraction, with respect to what we have done in these preliminary experiments, since the presence of noisy critical simplexes is the best environment to study the expressive power of such queries.

A first attempt, for the experimental evaluation of our model, has been done computing the ascending and descending Morse complexes on the extracted representations. We have noticed a considerable reduction in the timings required to extract such complexes in particular when the resolution is localized. Referring to the figures, the extraction for both the Morse complexes on the fully refined scalar field (Figure 7.7 (d)) takes 2.2 seconds (1.7 seconds for the 2-cells and 0.5 for the 1-cells). Reducing the geometric resolution (Figure 7.7 (c)) the timings decrease to 1.3 seconds for the Morse 2-cells and 0.4 seconds for the 1-cells. Localizing the resolution only inside a small subset (Figure 7.8 (a)) it takes 0.13 seconds for computing the Morse 2-cells and 0.05 seconds for the 1-cells.

7.2.4 Comparison with the Multi-resolution Morse triangulation

As far as we know, only one model addressed the problem of extracting representations the inspection of both the topology and the geometry of a terrain, the *Multi-resolution Morse triangulation* (MMT) presented in [DDFMV10].

As described in Section 2.3, an MMT encodes the Morse complexes, computed on a two-dimensional scalar field $\mathbb{M}_\Sigma = (\Sigma, f)$ defined on a triangulated mesh Σ encoding multiple representations of the critical net.

Three types of simplification are identified on such model and this represent the main difference with the $MTFG$, where we identify only two types of modifications. The MMT includes three models, one for the triangle mesh Σ, one for the 1-skeleton of the MS complex encoded as collection of edges from Σ and one for the combinatorial representation of the 1-skeleton of the MS complex.

Consequently in an MMT the model structure is represented through three DAGs interconnected where an $MTFG$ provides one DAG structure having two types of modifications encoded.

For both $MTFG$ and MMT, the simplification operator used for reducing geometrical resolution is the *edge-collapse*. In an MMT any valid *edge-collapse* is performed and, based on the ascending/descending Morse cells it affects (2-cells only, 1-cells only or 1-cells and 0-cells), it is related to a modification on the geometry or the topology of the 1-skeleton of the Morse-Smale complex, encoded as a graph G. The main drawback of an MMT is that the edge-collapse may create new critical points.

On the contrary, building an $MTFG$, a valid *edge-collapse* can only affect the geometry of Σ. Changes in the topology of the Morse complexes are caused only by a topological simplification operator. The number of *edge-collapse* operators selected during the building phase of an MMT is higher than on the $MTFG$ since, as described in Section 7.1.1.3, an *edge-collapse* in an $MTFG$ is considered as valid if it respects also the local Forman gradient configuration. The MMT model stores in the root of the DAGs three structures, the base triangle mesh, the edges composing the 1-skeleton of the MS complex and the graph structure of this latter. Similarly, the $MTFG$ stores the triangle mesh Σ_B but only the base Forman gradient is stored in addition.

Considering expansion to higher dimensions, the edge-collapse operator is defined also for tetrahedral meshes thus the $MTFG$ can be extracted to 3D and higher dimensions. The MMT can not be extended in higher dimensions; it is based on the 1-skeleton of the Morse-Smale complex. As described in Section 2.1.3, this feature is impossible to compute in general but considering the 1-skeleton of the Quasi Morse-Smale complex (see Section 2.1.3), it could be computed in the 3D case at most. However edge contraction may affect the boundary of the cells in the MS complex and thus the 1-skeleton encoded is not enough (the 2-skeleton of the MS complex have to be encoded at different level of details).

7.3 Current and future work

Our work is still concentrated in improving the experimental evaluation of the *Multi-Tessellation based on Forman gradient* model performing various selective refinement queries at uniform and variable resolution levels, in particular on real dataset.

A first effort will be the improvement of geometry simplification. We are planning to improve our simplification algorithm base on edge-collapse in order to enhance the quality of the meshes extracted. Error estimators such as Quadric Error Metrics (QEM) [GH97] are well established tools for the simplification of meshes in order to obtain representations with good quality at each step of our process.

The Forman gradient V is also a suitable tool for homology computation. In particular, an ascending/descending Morse complex computed on a scalar field $\mathbb{M}_\Sigma = (\Sigma, f)$ is a cell complex with the same homology as Σ. Moreover, homology computation can be performed on the ascending/descending Morse complex knowing only the incidence relations among its cells. This correspond to compute all the separatrix V-paths of V. Again, we could perform these computations on the base Forman gradient improving the computational speed considerably. Note the Morse complexes computed from any function f defined on Σ have the same homology as Σ. Thus, we could use the model also for the homology computation on shapes assigning an arbitrary function value to all the vertices of Σ (i.e. their index).

Another aspect we want to consider is the agreement between the geometric representation extracted from an $MTFG$ and the function values defined on its vertices. Since the model modifies the number of simplexes in Σ, but not the function values defined on its vertices, we could obtain geometric representations that disagree with the ascending and descending paths described by the Forman gradient. This problem has been studied in [BHEP04] and [WGS10] and described in Section 2. We are planning to relax the problem addressed in [BHEP04] modifying the function f, at the end of a selective refinement query, computing the new function f' based on the gradient paths in the extracted gradient V'. In other words, we want to use the Forman gradient V as constrain to modify the vertices values. Since all the vertices are in some V-path in V ending in a minimum, we are planning to rescale the function value of every single V-path such that any vertex in the path will have a function value higher than its predecessor. The V-paths defined on the triangle mesh will be used as well to prevent the creation of new critical simplexes.

We are also considering to extend the model to the 3D case. The edge-collapse is defined for tetrahedral meshes as it is for triangles. We should study carefully a gradient configuration, defined for all the tetrahedra incident in the collapse edge, in order to maintain the gradient flow after edge-collapse and in order to not introduce new critical simplexes; in other words a local gradient configuration having the same good properties of the configuration described in Section 7. Moreover the simplification of V, in dimensions higher then two is complicated. As described in [GRSW13] the i-cancellation operator in higher dimensions, can lead to a change in the connectivity of the critical points remaining if a subset of the V-path is shared by multiple separatrices. Problems occur when a single separatrix can merge and split and, in the 3D case, this occurs at saddle-saddle simplifications only. During 1-cancellation(p, q) of 1-saddle p and 2-saddle q, q can be connected to an arbitrary number of 1-saddles different from p. Changing the direction of a V-path, connecting p and q, can lead to a change in the connectivity of the critical points remaining if a subset of the V-path is shared by multiple separatrices. To overcome this problem we are planning to investigate the behavior of our simplification operator $remove$. In particular, by considering the two macro-operators defined in Section 4.3.2, we hope to able to enforce a connectivity, in the neighborhood of each critical point pair, compatible with the Forman gradient update.

Chapter 8

Curvature-based Segmentation of Tetrahedral Manifolds

The aim of morphological analysis is to provide a tool for understanding the structure of a scalar field through structural representations of the field so that its basic features can be easily recognized. In our work we used curvature to support morphological analysis.

The local curvature of a surface is very descriptive and it is an important tool in geometry processing. The applications of curvature estimation to the analysis of discrete surfaces have been extensively studied, and these include mesh simplification [HG99, She01], alignment [RL01], ridge-valley line detection [OBS04], non-photorealistic rendering [Rus04], segmentation [LPRM02, LDB05], partial shape matching [GG06], symmetry detection [MGP06], denoising [MDSB03, KSNS07], and remeshing [KNSS09].

Curvature estimation methods can be broadly classified into two categories: *fitting methods* and *discrete methods*. The former use local regression to estimate the parameters of continuous models and evaluate curvature using its continuous definition. The latter find discrete analogues to the continuous elements involved in defining curvature so that the notion can be evaluated directly in the discrete domain.

This chapter will focus on discrete methods. While fitting methods are more tolerant to noise and tessellation artifacts, they are computationally more intensive. This is a drawback in applications that require curvature to be estimated in real time, *e.g.*, physical simulation, non-photorealistic rendering, or real-time shape analysis for robotics applications. Discrete methods are simple to implement, require fewer computations, and are trivially parallelizable.

A volume dataset can be seen as a 3-manifold embedded in 4D space (*hypersurface*) and, as such, is amenable to 3D curvature analysis. Similarly, such analysis can be conceived for *hypersurfaces* in 4D space which are not the graph of a 3D scalar field, such as isosurfaces of time-varying scalar fields or tetrahedral meshes defined by animation sequences [PH08]. However, the application of curvature and its discrete variants to volumetric shape analysis remains comparatively unexplored. One exception is the concept of *discrete distortion* introduced in [MDFP08] as a discrete approach to curvature for three-dimensional tetrahedralized shapes embedded in 4D space.

Considering 3D scalar fields, these values of the field can be viewed as constraints on the vertices of a tetrahedral mesh. From this perspective, the values induce a distortion of the geometry of the mesh. As for surface curvature, discrete distortion highlights the local curvature of the constrained shape (the graph of the 3D scalar field) which cannot be perceived in the three-dimensional domain. As curvature gives interesting insights in terrain analysis, we show that distortion provides additional information to analyze the behavior of the intensity field. A null distortion value highlights a linear behavior of the intensity field, while a constant distortion corresponds to a uniform non-linear behavior. We observe that directions in which distortion changes indicate interesting directions in which the intensity field varies its growth speed.

Our contribution here concentrate on the study of segmentations based on the distortion values computed on a tetrahedral mesh. We will apply our approach to the analysis of the morphology of scalar fields through examples on synthetic, biological and medical datasets. We have studied the interaction between the resolution of the tetrahedral mesh approximating the field and the distortion values, showing that we can reasonably approximate the 3D image at fairly low resolutions [FIM$^+$12]. We will show also results on Morse segmentations based on the intensity and on the distortion values. Moreover we will discuss the existing definitions of distortion for discrete surfaces, classifying them into two categories, *intrinsic* and *extrinsic*, and we will present a generalization of extrinsic distortion to nD, deriving a weighting that can be used to compute mean curvature. We have analyzed the behavior of the operator on 3-manifolds in 4D and compared it to the well known Laplace-Beltrami operator [SFIM13]. We will show results obtained comparing the behavior on a suite of dataset sampled under varying conditions of resolution, distribution of samples, and noise.

In Section 8.1 we present a small overview on the many methods proposed in literature for computing curvature in the context of discrete representations. In Section 8.2 the classification in *intrinsic* and *extrinsic* curvature definition is given in the 2D and 3D case. In Section 8.3 we apply the notion of discrete distortion to the analysis of the morphology of scalar fields through examples on synthetic, biological and medical datasets. In Section 8.4 we present the generalization for the *extrinsic* distortion to n dimensions and we validate such generalization experimentally applying also the obtained operator to the segmentation of 3-manifold hypersurfaces in 4D.

8.1 Discrete curvature

Curvature is an important notion in mathematics that found a great interest in the last century. Curvature is also used to study the local geometry and topology of surfaces from the metric point of view.

Let us consider a plane π which contains the unit normal vector n_p at a point p on a surface patch S. The plane π intersects S in a curve C containing p with a curvature k at the point p. We recall that the curvature k at a point $p = c(t_0)$ of a parametric curve $(c(t))_{t \in R}$ is given by

$$k(p) = \frac{1}{\phi} = \frac{|c'(t)| \wedge c''(t)}{|c'(t)|^3}$$

where ϕ is the curvature radius.

When varying plane π around p also C varies. We can obtain two extremal curvatures $k_1 \leq k_2$ corresponding to the two orthogonal curves C_1 and C_2 at the point p.

Definition 8.1.1. ***Gaussian Curvature*** *- At a point p of a surface the Gaussian curvature K_p is defined as $K_m = k_1 k_2$.*

Definition 8.1.2. ***Mean Curvature*** *- At a point p of a surface the Mean curvature H_p is defined as $K_p = \frac{k_1 + k_2}{2}$.*

From these two definitions we can see that both the Gaussian and mean curvature depend on the local geometrical shape of a surface. Curvature is a mathematical tool defined for smooth, at least C^2-continuous, surfaces.

Definition 8.1.3. ***Gauss-Bonnet Theorem*** *- Given a compact surface S with possible boundary components ∂S we have $\iint_S K_m ds + \int_{\partial S} k_g(m) dl = 2\pi\chi(S)$ where χ is the Euler characteristic of surface S and k_g indicates the geodesic curvature at boundary points.*

The importance of this theorem is the connection between the geometrical and topological properties of a surface S defined, respectively, by Gaussian curvature and by the Euler characteristics.

With the development of discrete geometry, many authors tried to define a discrete counterpart of curvature based on the properties observed in the continuum [GG06, HBB$^+$09, Sha06, SMS$^+$03].

Here, we will assume familiarity with the fundamental notions of curvature in the continuum and refer the interested reader to the relevant texts [Car76]. In the context of discrete representations of 2D surfaces such as meshes and point clouds, curvature is a well-studied area within geometry processing due to its numerous applications to shape [GG06, HBB$^+$09]. There are a plethora of methods for curvature evaluation, each with advantages and disadvantages. These methods can be broadly classified into *fitting* methods and *discrete* methods.

Concentrated curvature [Ale57, Tro86] is a simple and efficient method to define a discrete curvature. Dyn et al. discuss how to optimize the triangulation of the boundary of a 3D object based on discrete curvature [DHSJL01].

Fitting methods use local regression to fit a continuous function, such as a polynomial, to the surface data near a point of interest. Once the parameters of such a function are determined, a curvature estimate can be computed analytically or using finite element methods [GG06, HBB$^+$09, Ham92, SW92, PWHY09, CS92]. Other approaches, rather than fitting a model to surface points, fit the curvature tensor to normal variations in a local neighborhood [Rus04, KSNS07]. In contrast, discrete approaches [SW92, Tau95, MDSB03, PSKA02, LBS07, CSM03] compute quantities that approximate curvature values directly on the discrete surface without explicitly fitting a continuous model.

In the 3D case, the Ricci tensor is used to define the curvature notion for three-dimensional shapes [And05], and, in the discrete case, the Laplace operator is generally used to define a discrete approach to curvature [RWP06].

In the 4D case, curvature is one of the most important mathematical notions on which general relativity is based. Curvature of the space-time gave an important contribution to understand many phenomena in physics (black holes, gravitational lenses, light trajectories, interaction between planets, ...). Based on Aleksandrov's concentrated curvature, Regge introduced a discrete version of curvature for the four dimensional space-time [Reg61].

Within the context of 3-manifolds in 4D, there is much less work within the geometry processing community. Hamann introduces a generalization of the polynomial fitting approach to extend to such data [Ham94a], but this is based on a continuous method rather than a discrete one. Within the discrete setting, the notion of discrete distortion arises as a purely discrete analog to continuous curvature [MDFP08]. It has been successfully used in several applications, including morphological analysis [MDFM09], guiding multi-resolution simplification [WDFM10], and medical visualization [FIM$^+$12].

8.2 Discrete intrinsic and extrinsic distortion

In this Section we present our categorization for the notion of distortion in *intrinsic* and *extrinsic* distortion. A first notion of discrete distortion derives, as described in [MDFP08], as discrete analog to continuous curvature. This particular notion of discrete distortion is an *intrinsic* measure based on a generalization of concentrated curvature [Ale57, Tro86] and angle deficit [MDSB03]. In 2D, however, the previously-introduced notion of distortion is based on dihedral angles and is related to mean curvature [MDFM12] and, as such, is an *extrinsic measure*.

8.2.1 Intrinsic distortion in 2D

In the 2D case a notion of discrete distortion has been introduce in [MDFM08] as discrete counterpart of the concentrated curvature.

Given a triangulated surface, and a vertex v in its interior, we consider the local neighborhood of v. If v lies inside a triangle, then this is a whole disc; if v is located on an edge then it is the union of two half discs; finally, if v is a vertex, then its local neighborhood is the union of as many angular sectors as the number of faces incident in v. In this last case, the sum of angles of all these sectors may not be equal to 2π. In case it is not, v is called a singular conical point. We consider the total angle ω_v at v given by the sum of angles at v of all triangles incident in v. The *concentrated Gaussian curvature* at a vertex v is defined as

$$K(v) = 2\pi - \omega_v$$

if v is an internal vertex

$$K(v) = \pi - \omega_v$$

if v lies on the boundary of the surface.

8.2.2 Extrinsic distortion in 2D

Distortion has been previously defined at a vertex [MDFM12] as follows. The idea is to compute the sum of the angle deficits of the dihedral angles at the edges incident on v, with respect to the flat angle. Vertex distortion at an internal vertex v of the triangulation is defined as

$$D(p) = \sum_{i=1}^{N}(\pi - \Theta_{e_i}),$$

where $e_1, \ldots, e_N$ are the edges incident on v, and Θ_{e_i} is the dihedral angle formed between the two triangles incident at edge e_i.

A weighted version of this formulation can be used to estimate mean curvature. This weighted form coincides with the cylindrical approximation method [DHSJL01]. For each edge, the integral form is obtained by weighting the angle by half of the edge length for each vertex in the edge. Another factor of one half is introduced by the fact that the one of the principal curvatures of a cylinder is null, thus causing the mean curvature to be one half that of the non-null principal curvature. The final punctual form of the mean curvature estimate is obtained by dividing by the area A_p associated with the vertex p, computed as the sum of fractional areas of all triangular faces incident on p. This fraction can be taken to be a fixed $1/3$, leading to the barycentric formulation, or the *Voronoi region* can be used instead. The final expression of the punctual form is thus:

$$\hat{H}(p) = \frac{1}{4A_p} \sum_{i=1}^{N} \|e_i\|(\pi - \Theta_{e_i}).$$

8.2.3 Intrinsic distortion in 3D

For tetrahedral meshes embedded in four-dimensional Euclidean space a separate notion of distortion has been introduced [MDFP08], in this case, as a generalization of Aleksandrov's concentrated curvature [Ale57] to higher dimensions, which can be considered a discrete counterpart to the scalar Ricci curvature [JKG07]. A similar approach has been independently proposed [YJLG08]. This is an *intrinsic* measure based on angle deficits and, given a tetrahedralized manifold embedded in 4D space, the intrinsic distortion at an internal vertex p can be defined as

The graphical representation of a scalar field f defined on a tetrahedral mesh Σ is a *hypersurface* $(\Sigma; f)$ in R^4, namely, a tetrahedral mesh embedded in R^4. Hypersurface $(\Sigma; f)$ is generally curved due to the effects of the scalar field values. As for concentrated curvature, one may compare the defect solid angle at the vertices of Σ, when applying the scalar field.

The *discrete distortion* at a vertex v of Σ is defined as the quantity

$$D(p) = 4\pi - \sum_{i=1}^{n} \omega_i,$$

where ω_i is the solid angle at v of the i-th tetrahedron incident at said vertex.

Intrinsic distortion for 3D scalar fields has similar properties as concentrated curvature for 2D fields. Concentrated curvature gives positive values to locally convex, or concave, areas of the surface, negative values to saddles, and null values to flat areas. Similarly, positive values of distortion correspond to locally convex, or concave, portions of the hypersurface which is the graph of the field. Negative values correspond to saddle and degenerate saddle configurations.

Constant scalar fields are distortion-free (i.e., their distortion is null). This can easily be understood since, for a constant scalar field, mesh $(\Sigma; f)$ is only a translation, in the fourth dimension, of the mesh Σ decomposing the domain of the field. Hence, the Euclidean geometric structure of the mesh is preserved. More generally, affine scalar fields are distortion-free, since they combine rotations and translations of the whole mesh. Hence, the geometrical structure is not subject to any distortion.

As a consequence, piecewise linear scalar fields are distortion-free at the interior vertices of regions where the field is linear, as they act affinely within such regions. Another relevant property is that distortion is mesh-dependent. This means that the distortion value at a vertex depends on the way in which the neighborhood of such vertex is triangulated.

8.2.4 Extrinsic distortion in 3D

We can extend to the 3D case the extrinsic distortion previously defined for the 2D case. Extending the idea to compute the sum of the angle deficits of the dihedral angles at the edges incident on v, and considering the dihedral angle at the triangles incident in a vertex v we can defined an extrinsic notion of distortion for the 3D case as

$$D(p) = \sum_{\tau_{ij} \in \mathrm{St}^2(p)} (\pi - \Theta_{f_i}),$$

where $f_1, \ldots, f_N$ are the edges incident on v, and Θ_{f_i} is the dihedral angle formed between the two tetrahedra incident at face f_i.

In [SFIM13] we have proposed a generalization of the extrinsic distortion to nD, deriving also a weighting that can be used to compute mean curvature on tessellated hypersurfaces. We will present such generalization in Section 8.4. We have studied the behavior of such operator on 3-manifolds in 4D and compared to the well-known Laplace-Beltrami operator (described in Section 8.4.1.1) using ground truth hypersurfaces defined by functions of three variables, and a segmentation application, showing it to behave as well or better while being intuitively simple and easy to implement.

8.3 Intrinsic distortion for 3D data analysis

In this Section are illustrated some experimental results which show the behavior of *discrete distortion* as a tool for analysis of 3D scalar fields. Because of the large size of current data sets, it is also important to perform accurate analysis on low-resolution representations of the field. Thus, the influence of mesh resolution on distortion has been studied by considering variable-resolution conforming tetrahedral meshes extracted from a hierarchy of diamonds according to a user-defined threshold on the approximation error. In this case, resolution can be coarsened locally in less interesting regions, without affecting the quality of the approximation.

In [FIM$^+$12] we have applied the notion of *intrinsic distortion* to the analysis of the morphology of 3D scalar fields through examples on synthetic, biological and medical datasets. In particular, the interaction between the resolution of the tetrahedral mesh approximating the field and the distortion values has been studied, showing that is possible to reasonably approximate the 3D scalar

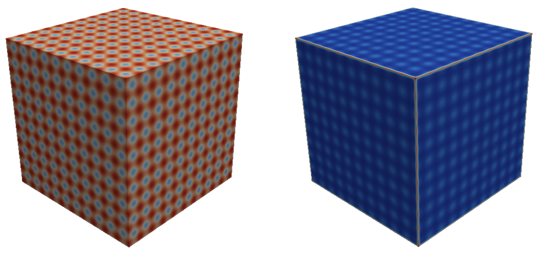


Figure 8.1: Intensity field (left) and distortion field (right) for synthetic data set sampling function $f(x, y, z) = sin(x) + sin(y) + sin(z)$ over a 65^3 grid.

field at fairly low resolutions. Moreover, results obtained by computing Morse decompositions based on the intensity and on the distortion values are shown.

One way to perform morphological analysis is to automatically decompose the domain of the field into meaningful parts in order to support understanding and semantic annotation. Segmentation has been the basic tool to support reasoning on terrains and 3D shapes. The segmentations proposed here, are based on the values of the scalar field or on discrete distortion, in a similar way as done for terrains where segmentations are computed based on elevations and/or on curvature values. We will show results on comparing segmentations of the 3D scalar fields, obtained through Morse decompositions of the intensity and of the distortion fields. To this aim, results on both synthetic and real data set are shown.

8.3.1 3D datasets and distortion

Tetrahedral meshes considered are all extracted from a multi-resolution representation of 3D scalar fields provided by a regular tetrahedral hierarchy. The efficient representation of a regular tetrahedral hierarchy developed is called a hierarchy of diamonds, and has been discussed in [WD09].

First example is a synthetic dataset defined over a regularly sampled domain of 65^3 vertices. The intensity field is obtained by sampling the ANALYTIC function $f(x, y, z) = sin(x) + sin(y) + sin(z)$. The relationship between the intensity field and the induced distortion field over this domain is illustrated in Figure 8.1 along the boundary of the cubic domain using a blue-red color scale to indicate the low and high scalar and distortion values.

The second dataset, the NEGHIP, is a simulation of the spatial probability distribution of electrons in a high potential protein molecule. The knowledge of electron distribution within such molecules is important in pharmacology to understand the interactions between molecules and an organism. The inhibition of some protein molecules can reduce complications in diseases such as cataracts and neuropathies for diabetic subjects. The understanding of the catalytic mechanism and the electrostatic potential of the molecule plays a relevant role here. It may help to study, at the atomic scale, the transfer of electrons and protons in complex biological processes such as oxidation/reduction in relation to metallic ions by considering the reaction between hemoglobin (containing iron ions) and the oxygen molecule.

In Figure 8.2, the intensity and distortion fields are shown for the tetrahedral mesh extracted from the hierarchy computed on the NEGHIP dataset at variable resolution corresponding to 0% approximation error. The range of colors used for visualization goes from blue for low values to red for high values, with gray indicating mean values.

Intrinsic 3D distortion highlights the growth behavior of the density scalar field, which is maximal around the atoms. The density field grows quickly around atoms within small regions and then stabilizes its growth. Distortion becomes nearly constant in such case. It was also observed that, within regions where the electron density has low values, many small regions have high distortion values. This indicates changes in the electron density and may be due to the interference between adjacent atoms or to some artifacts in the processing of the data. Regions in blue (for distortion) indicate that the scalar field grows differently in different directions. This corresponds to saddle regions where the convexity of the electron density field changes.

The third dataset, the CTA-BRAIN, is a CTA-scan of a human brain with an aneurysm. Computed Tomographic Angiography (CTA) is a minimally invasive technique that uses imaging technologies (e.g., X-rays) to explore the structure of vessels and tissues. A contrast agent is generally used to produce clear images.The original dataset has 512x512x120 vertices and measure the intensity of the contrast agent. To show the behavior of the intensity field and of distortion, a variable-resolution mesh from the diamond hierarchy has been extracted, which has 1.74 million vertices and 9.52 million tetrahedra.

Figure 8.3 illustrates the dataset, where the scalar field corresponds to the intensity of the contrast agent, and its distortion, through equally spaced horizontal slices. The geometric structure of the scanned region is well represented by distortion. Most regions have gray or light blue color, which indicates a uniform distribution of the contrast agent within the brain. The regions with high distortion correspond to changes in the intensity of the contrast product.

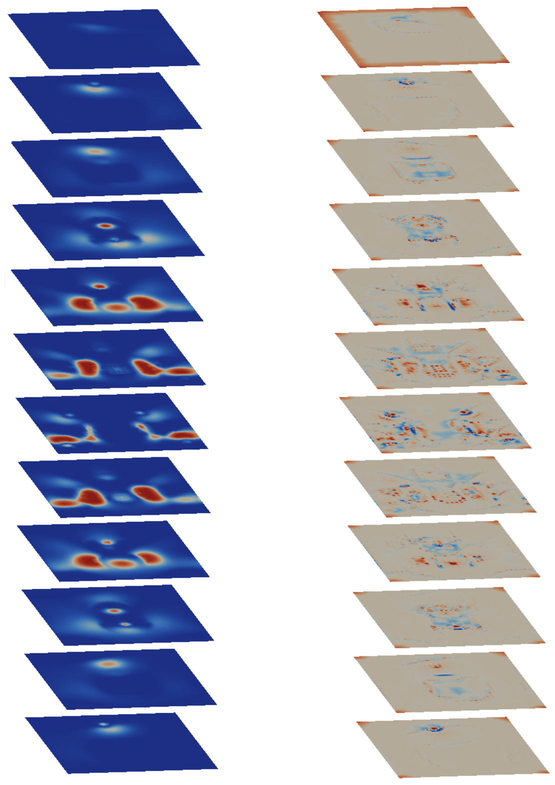


Figure 8.2: Twelve equally spaced slices (along the z-axis) of the intensity field (left), and the distortion field (right) of the NEGHIP dataset at 0% error. The colors of the distortion field are scaled to highlight the extreme values.

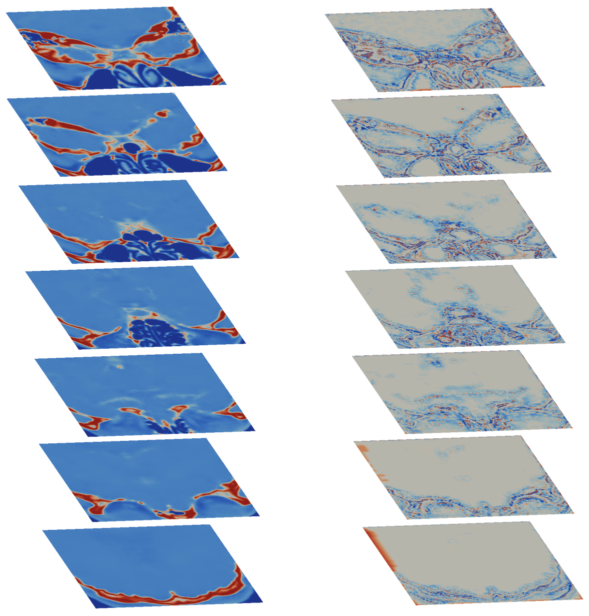


Figure 8.3: Seven equally spaced slices (along the z-axis) of the CTA-BRAIN dataset at 10% error illustrating the scalar field (left), and the distortion field (right). The colors of the distortion field are scaled to highlight the extreme values.

8.3.2 Distortion and mesh resolution

The validity of distortion analysis has been demonstrated on lower resolution approximations by considering the distribution of distortion values over a set of extracted meshes with increasingly fine resolution. Results are shown on the two real data sets only, NEGHIP and CT-BRAIN datasets. The error associated to a diamond δ is computed as the maximum difference between the intensity values of all grid points within the domain of δ, and the values obtained by linear interpolation over the vertices of the tetrahedra of δ. Extracted a series of meshes Σ_{ϵ_i} of uniform approximation error ϵ_i from Δ_H, the distortion of the vertices of these meshes is evaluated using threshold values of $\epsilon_i \in \{30\%, 10\%, 5\%, 2\%, 1\%, 0\%\}$ of the total error.

Figure 8.4 shows the *Cumulative Distribution Function (CDF)* of the discrete distortion (horizontal axis) of the vertices of each mesh. The sharp spike in the CDF of all datasets around a null distortion value indicates that the vast majority of vertices have (nearly) null distortion. As the resolution increases, this spike becomes steeper, indicating that the increased resolution is distributed among regions with nearly null distortion. Thus, the distortion is concentrated in relatively few vertices within the mesh, and appears prominently in lower resolution approximations. For example, when $\epsilon = 0$, more than 94% of the 129 K vertices in $\Sigma_{0\%}$ have distortion $D(v) \leq |1|$, and for $\epsilon = 2\%$, more than 83% of the 33 K vertices in $\Sigma_{2\%}$ have distortion $D(v) \leq |1|$.

Similarly, Figure 8.5 shows the CDF of meshes Σ_{ϵ_i} using threshold values of $\epsilon_i \in \{99\%, 75\%,$-$50\%, 30\%, 10\%, 5\%\}$ extracted from the CTA-BRAIN dataset. These meshes illustrate the same general trend as the Neghip approximations, although they are a bit noisier since they are scanned images.

These experiments indicate that a fairly accurate understanding of the behavior of scalar field can be obtained via its discrete distortion even at lower resolutions, without the need to compute the distortion on the field at full resolution.

8.3.3 Morse Decompositions

In this Subsection, Morse decompositions of the synthetic and real data sets are computed using the intensity and the distortion fields. The experiments have been performed by using the watershed by simulated immersion implementation (see Section 2.1.4).

Lets consider the distribution of the intensity and distortion values for the synthetic data set shown in Figure 8.1. Figure 8.6 shows the ascending and descending Morse 3-cells computed based on the intensity and on the distortion fields. It is clear how the distribution of the intensity and of the distortion values influences the corresponding segmentations. Both ascending and descending Morse 3-cells obtained from the intensity field consists of 1,331 cells and have a

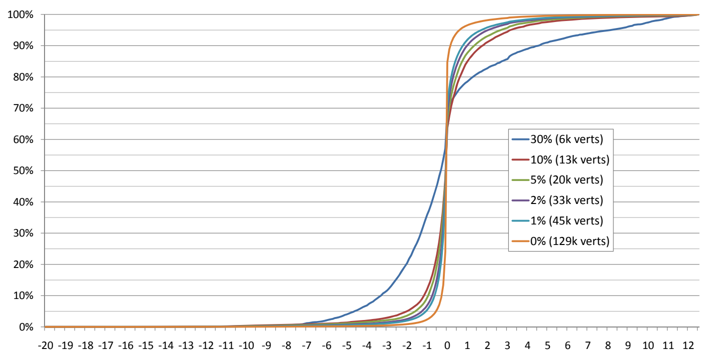


Figure 8.4: Cumulative distribution functions of distortion values (horizontal axis) over increasingly fine meshes extracted from the NEGHIP dataset.

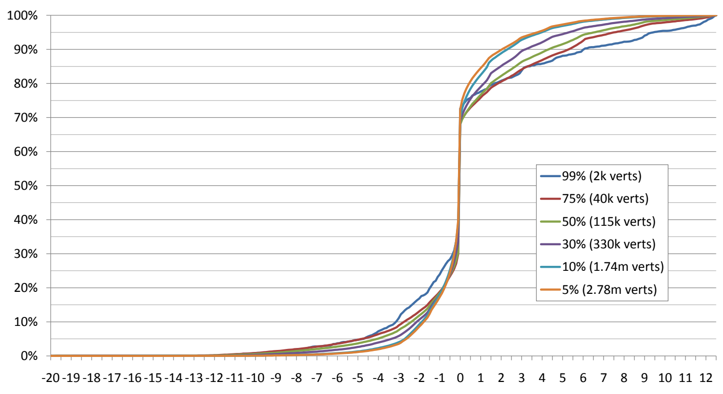
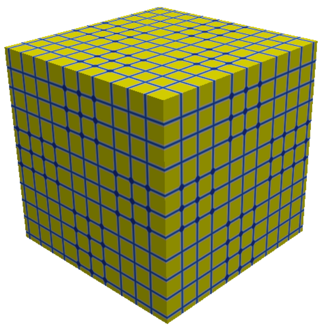


Figure 8.5: Cumulative distribution functions of distortion values (horizontal axis) over increasingly fine meshes extracted from the CTA BRAIN dataset.

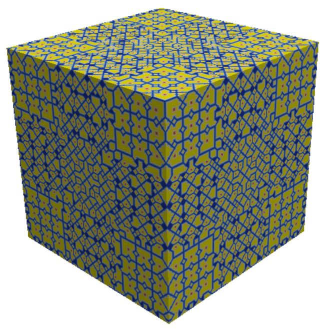
regular structure. The ascending decomposition obtained from the distortion field consists of 12,972 cells, while the descending one consists of 3,738 cells. The decomposition pattern in the ascending and descending distortion-based complex varies in different portions of the mesh. This is due to the function sampling that is different from its period.

Figure 8.7 shows Morse 3-cells built from the full-resolution tetrahedral mesh discretizing the NEGHIP dataset. Visualization is thresholded along an isovalue to better illustrate the structure of the molecules. The ascending and descending Morse 3-cells obtained from the intensity field consist of 104 cells and of 41 cells, respectively. The ascending and descending Morse 3-cells obtained from the distortion field consist of 3,654 ascending cells and 23,334 descending cells, respectively.

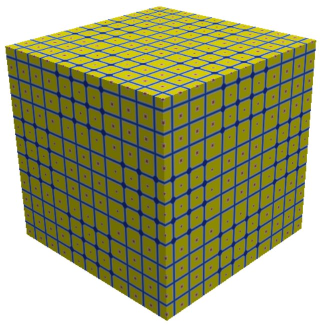
Some components of the descending decomposition represent the location of atoms (i.e. maxima of the density) and the proper space in which electrons revolve around. Due to the interference



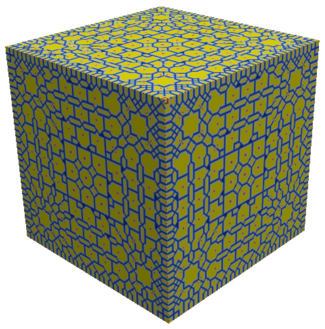
Ascending decomposition of intensity field (1331 cells)



Ascending decomposition of distortion field (12972 cells)



Descending decomposition of intensity field (1331 cells)



Descending decomposition of distortion field (3738 cells)

Figure 8.6: Morse 3-cells for the ANALYTIC data set defined by intensity function $f(x, y, z) = sin(x)+sin(y)+sin(z)$. Minima or maxima vertices are colored in red, vertices on the boundary of several regions in blue and vertices within a region in yellow.

of electron density of adjacent atoms, some components are created and correspond to some maxima of the density field. These components do not properly contain atoms.

Figure 8.8 illustrates the intensity field, the corresponding distortion values and the segmentations obtained from a uniform resolution mesh $\Sigma_{10\%}$ extracted from the CTA-BRAIN dataset (see also a view as set of slices in Figure 8.3).

The decomposition obtained from the intensity field consists of 37,631 ascending 3-cells and of 23,835 descending 3-cells, while the decomposition obtained from the distortion one consists of 136,641 ascending 3-cells and 128,687 descending 3-cells. Figure 8.9 shows the largest segments from the segmentations. Observe that, while the descending regions are more structured and follow the field values, the ascending regions are much more influenced by the boundary and by the less relevant regions of the original scalar field. The former therefore seem to provide a more meaningful decomposition. The large number of cells in the descending decomposition computed on the basis of distortion is due to the fact that there is a large number of small areas in which the concentration of the contrast agent changes abruptly (i.e., distortion has a maximum).

8.4 Generalizing extrinsic distortion to nD

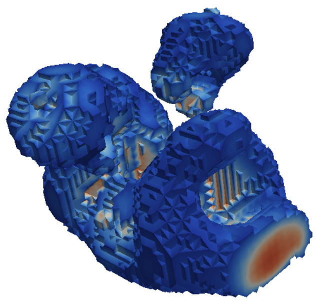
In [SFIM13] we have proposed a generalization of the extrinsic distortion to nD, deriving also a weighting that can be used to compute mean curvature on tessellated hypersurfaces.

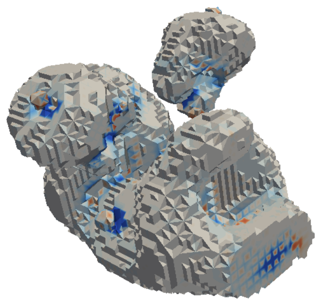
The generalization can be done by considering the dihedral angle at adjacent simplexes. On a discrete n-dimensional manifold, embedded in $(n+1)$D, represented by a simplicial complex Σ, pairs of adjacent n-simplexes form a dihedral angle determined by the two hyperplanes containing each of them. Assuming the manifold is orientable, the signed dihedral angle formed by these hyperplanes can be determined in a straightforward manner, leading to formulate the general expression for extrinsic distortion:

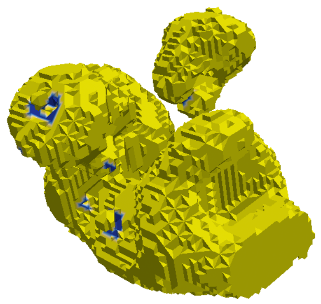
$$D(v) = \sum_{\tau_{ij} \in \mathrm{St}^2(v)} (\pi - \Theta_{ij}),$$

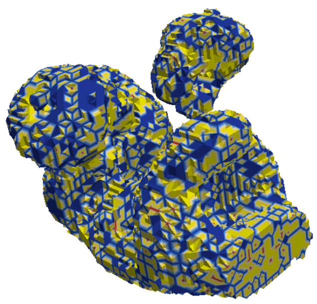
where Θ_{ij} represents the signed dihedral angle between the simplexes σ_i and σ_j, and $\tau_{ij} \in \mathrm{St}^2(v)$ is defined as true if $\sigma_i, \sigma_j \in \mathrm{St}(v)$ and $\tau_{ij} = \sigma_i \cap \sigma_j$, where τ_{ij} is an $(n-1)$-simplex. This is to say σ_i and σ_j are adjacent and their union has disk topology.

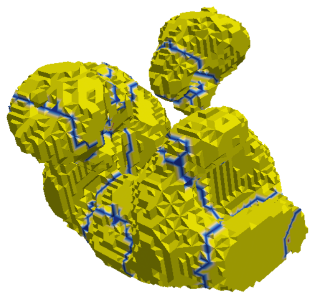
The weighting that leads to a mean curvature approximation can also be generalized, inferring it from Dyn's cylindrical approximation in 2D [DHSJL01]. Two adjacent n-simplexes will meet at an $(n-1)$-simplex τ_{ij}, and the edge length used in the 2D case can be generalized to the volume of τ_{ij}. Just as in the 2D case one half of the weighted angle went to each vertex on the edge, in the general case $1/n$ goes to each of the vertices at the simplicial intersection. Finally, the


NEGHIP field


NEGHIP distortion


Ascending decomposition
of original field (104 cells)


Ascending decomposition
of distortion field (3,654 cells)


Descending decomposition
of original field
(41 cells)

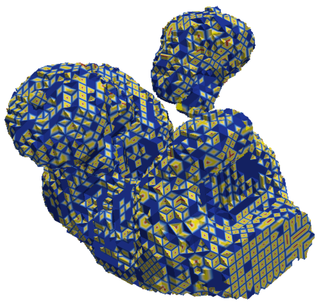
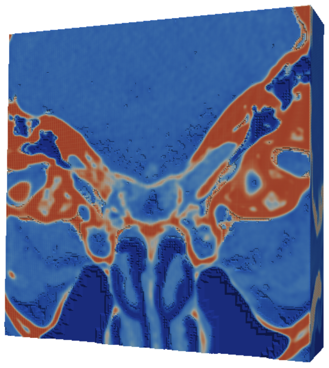
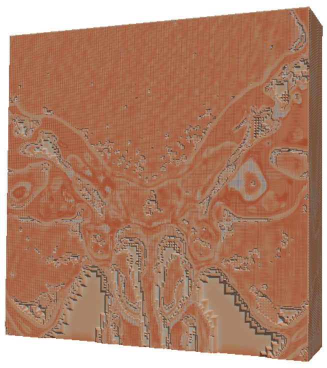

Descending decomposition
of distortion field
(23,334 cells)

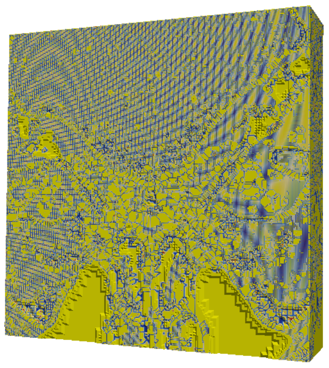
Figure 8.7: Original field and distortion field, and segmentations, for variable resolution NEGHIP data set at 0% approximation error. Segmentations are shown with minima (ascending) or maxima (descending) vertices in red, vertices on the boundary of more than one region in blue and vertices within a region in yellow



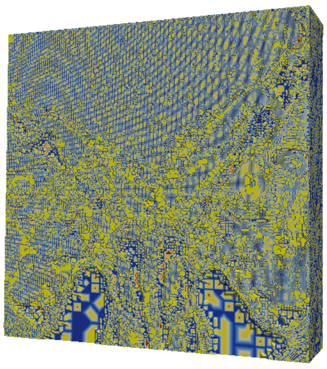
BRAIN field



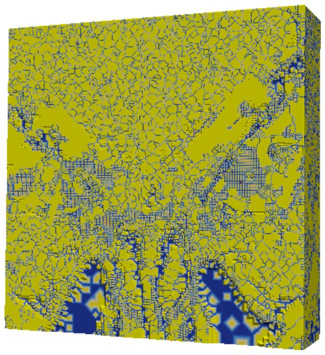
BRAIN distortion



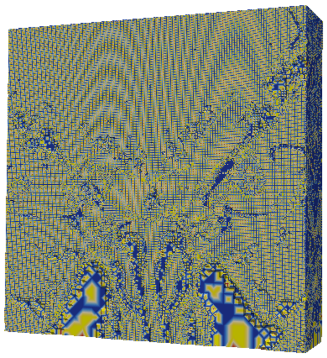
Ascending decomposition
of original field (37,631 cells)



Ascending decomposition
of distortion field (136641 cells)

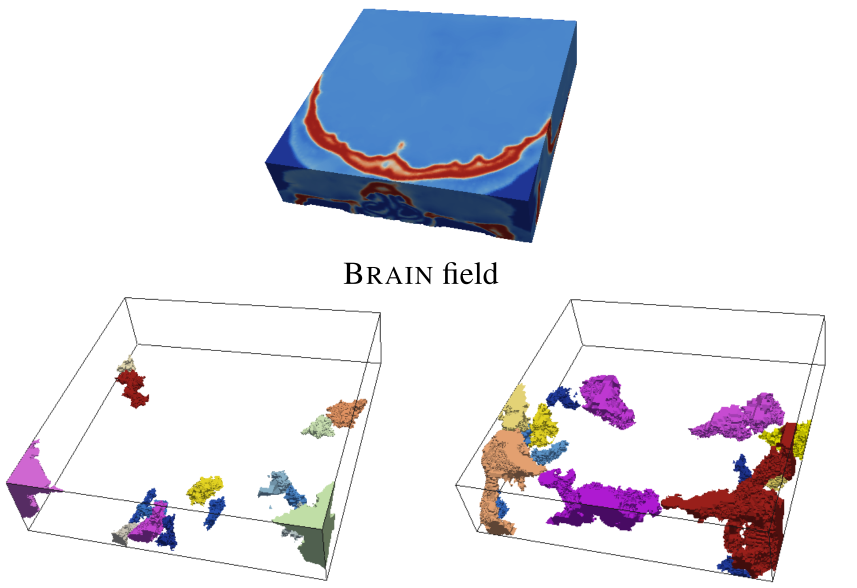


Descending decomposition
of original field
(23,835 cells)



Descending decomposition
of distortion field
(128,687 cells)

Figure 8.8: Original field and distortion field, and segmentations, for CTA-BRAIN data set at 10% resolution.



BRAIN field

Ascending regions with number of tetra between 13,000 and 100,000

Descending regions with number of tetra more than 20,000

Figure 8.9: Ascending and descending 3-cells using a threshold to visualize distinct regions formed by a large number of tetrahedra.

cylindrical generalization has $n-1$ null principal curvatures and thus its mean curvature is given by $1/n$-th of the non-null value. This all leads to the final weighted expression:

$$\hat{H}(v) = \frac{1}{n^2\|v\|} \sum_{\tau_{ij}\in \mathrm{St}^2(v)} (\pi - \Theta_{ij})\|\tau_{ij}\|,$$

where $\|v\|$ is the n-dimensional barycentric volume associated with vertex v and $\|\tau_{ij}\|$ is the $(n-1)$-dimensional volume associated with the simplex τ_{ij} at the intersection of the adjacent simplexes σ_i and σ_j.

Derivation The derivation of the above weighting is provide here. Note that this is not intended as a proof of convergence. Remarking that when two hyperplanes in R^{n+1} intersect, the intersection is an $(n-1)$-affine plane $\mathcal{P}_{n-1}$, lets approximate smoothly the piecewise linear hypersurface through a cap of the curved nD hypersurface $\mathcal{C}_n(r) = S^1_r \times \mathcal{P}_{n-1}$, where r is a positive real number. The cap is obtained from an arc of the circle S^1_r. The hypersurface $\mathcal{C}_n(r)$ is isometric to the hypersurface defined by

$$\mathcal{C'}_n(r) := \{(x_1, \ldots, x_{n+1}) : x_n^2 + x_{n+1}^2 = r^2\},$$

which can be seen as the image of two functions $f_{\pm}$:

$$f_{\pm}(x_1,\dots,x_n) = \pm\sqrt{r^2 - x_n^2}.$$

This hypersurface is also isometric to the graph $\mathcal{C}"_n(r)$ of the translated functions $r - f_{\pm}$. Let us define g as

$$g(x_1,\dots,x_n) := r - \sqrt{r^2 - x_n^2}.$$

where $g(0,...,0) = 0$ and $\frac{\partial g}{\partial x_i}(0,\dots,0) = 0$ for all i. Then, the second fundamental form at the origin of $\mathcal{C}"_n(r)$ reduces to the Hessian matrix of g at the origin whose coefficients are all 0 except the last diagonal one, which is equal to $1/r$. Consequently, the mean curvature of $\mathcal{C}"_n(r)$ at the origin is simply $\frac{1}{nr}$. Thus the mean curvature of $\mathcal{C}_n(r)$ at any of its points is equal to $\frac{1}{nr}$.

The total curvature of the cap approximating the piecewise linear hypersurface is thus equal to the integral over the cap of $\frac{1}{nr}$. Since the cap is tangent to the piecewise linear hypersurface, then, at the contact point, the cap and the piecewise linear hypersurface have the same normal vectors. This means that the angle defining the arc of S^1_r of the cap is equal to π minus the angle Θ between the two normal vectors at the contact points, which are simply the normal vectors of the hyperplanes whose intersection is approximated by the cap. Thus the total mean curvature over the cap is

$$\|\tau_{ij} \cap v\| r(\pi - \Theta_{ij})\frac{1}{nr}$$

where $\tau_{ij} \cap v$ represents the intersection of τ_{ij} with the neighborhood of v. Now, supposing that in the neighborhood on the hypersurface around the vertex v the mean curvature H_v is constant, then the total mean curvature over the neighborhood is equal to $H_v\|v\|$. Hence

$$H_v\|v\| = \sum_{\tau_{ij}\in \mathrm{St}^2(v)} \|\tau_{ij} \cap v\|\frac{(\pi - \Theta_{ij})}{n}$$

From this is obtained

$$H_v = \frac{1}{n\|v\|}\sum_{\tau_{ij}\in \mathrm{St}^2(v)} (\pi - \Theta_{ij})\|\tau_{ij} \cap v\|$$

Supposing, when computing the mean curvature at all vertices v, the volume neighborhoods around v divide every $(n-1)$-simplex into n subsimplexes of the same volume (*e.g.* as in a barycentric configuration), then it holds that $\|\tau_{ij} \cap v\| = \frac{1}{n}\|\tau_{ij}\|$ and thus:

$$\hat{H}(v) = \frac{1}{n^2\|v\|}\sum_{\tau_{ij}\in \mathrm{St}^2(v)} (\pi - \Theta_{ij})\|\tau_{ij}\|.$$

8.4.1 Experimental comparison on analytic surfaces

In order to evaluate the proposed method, a set of analytic 3D surfaces embedded in 4D as been used with ground-truth mean curvature values as well as a segmentation application based on Morse decomposition [CFI12]. In all cases the results are compared with those obtained using the Laplace-Beltrami operator. Laplace-Beltrami operator is one of the best known tools for the computation of mean curvature on discrete surfaces [MDSB03]. It can be readily generalized to 3D manifolds and, as a well-known discrete operator and estimator of mean curvature, we use it as a basis for comparison of our operator.

8.4.1.1 The Laplace-Beltrami operator

We consider the Laplace-Beltrami operator because of how well established it is in the literature as a discrete estimator of mean curvature. A thorough comparison of a large set of curvature operators is beyond the scope of our work and, for such a comparison, we refer the reader to published works [GG06, KSNS07]. Our main goal is to compare our approach to a well known one, establishing it to behave as well as said approach with some advantages, and thus placing it in context.

Value Laplace-Beltrami operator at a vertex v is given by

$$\mathbf{K}(v) = \frac{1}{V(v)} \sum_{i \in N_1(v)} w_i(v - x_i)$$

In 3-manifolds, the mean curvature value is given by $\frac{1}{3}||\mathbf{K}||$. Here, $V(v)$ denotes the tetrahedral volume assigned to vertex v and w_i denotes the weight associated with the edge (v, x_j). In 3D, this weight is given by

$$w_i = \frac{1}{6} \sum_j \ell_i^j \cot \alpha_i^j$$

where ℓ_i^j is the length of the edge *opposite* to edge (v, x_i) within the j-th tetrahedron incident on (v, x_i), and α_i^j is the dihedral angle at this opposite edge.

For the value of the volume $V(v)$ we simply use barycentric volumes, obtained as $1/4$th of the sum volume of all tetrahedra incident on v. Alternatively, it is possible to use Voronoi volumes, though we have found that, in the 3D case, it does not reliably improve the accuracy of the mean curvature estimate.

The scalar value of the operator as defined above would always be a positive quantity, given that it is a fraction of the norm of the $\mathbf{K}$ vector. However, we can set the sign of the scalar value by setting it to match the sign of the dot product between $\mathbf{K}$ and the manifold normal (positive when they are in agreement, negative when opposite). In the case of graphs of scalar fields, as

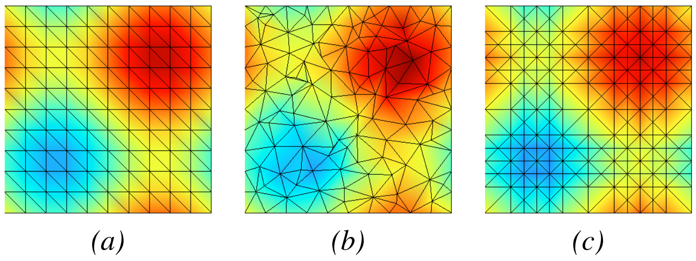


Figure 8.10: A $z = 0$ slice of the different tessellations used (lowest resolution for each shown for illustrative purposes) illustrated on our sixth analytic function; (a) regular grid, (b) irregularly sampled, (c) diamond mesh.

we will be examining, we can simply use the sign of the last component of **K** for efficiency and simplicity.

8.4.1.2 Operator evaluation

The proposed operator is evaluated by considering analytic functions and comparing the result to the corresponding known analytic mean curvature values. The functions are those suggested by Hamann [Ham94b] as well as a second trigonometric function. They are as follows:

1. Quadratic polynomial: $0.4(x^2 + y^2 + z^2)$
2. Quadratic polynomial: $0.4(x^2 - y^2 - z^2)$
3. Cubic polynomial: $0.15(x^3 + 2x^2y - xz^2 + 2y^2)$
4. Exponential: $\exp(-0.5(x^2 + y^2 + z^2))$
5. Trigonometric: $0.1(\cos(\pi x) + \cos(\pi y) + \cos(\pi z))$
6. Trigonometric: $\sin(\pi x) + \sin(\pi y) + sin(\pi z)$

The functions are sampled in the $[-1, 1]^3$ real interval using three different approaches. In the first, a uniform grid of samples is used which is then tessellated using a stencil Voronoi approach. That is to say, a cube is Voronoi-tessellated in the 3D domain and then repeated over the entire grid. This is equivalent to a Voronoi tessellation of the grid samples in the domain, but clearly more efficient. In the second approach, we create irregular tessellations. For a given number of vertices, we randomly sample the interior of the interval and Delaunay-tessellate said samples. In order to discourage poorly-shaped tetrahedra, the boundary is uniformly sampled, a minimum distance constraint during the sampling is enforced, and the final mesh is relaxed with 100 iterations of Laplacian smoothing. Finally, also diamond meshes are used [WDF11]. Figure 8.10 illustrates these approaches by showing a 2D slice of the sixth function above.

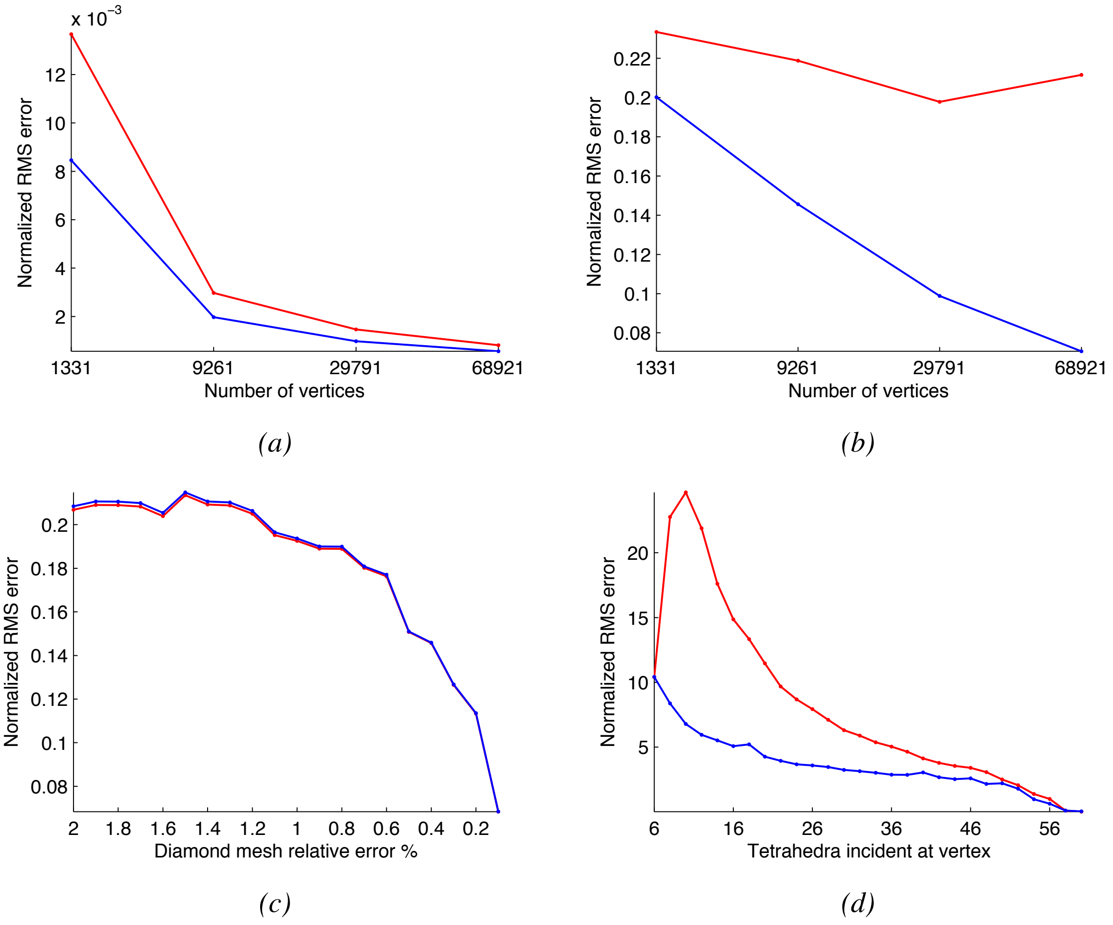
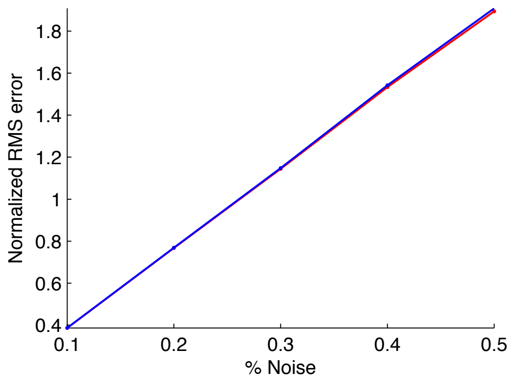


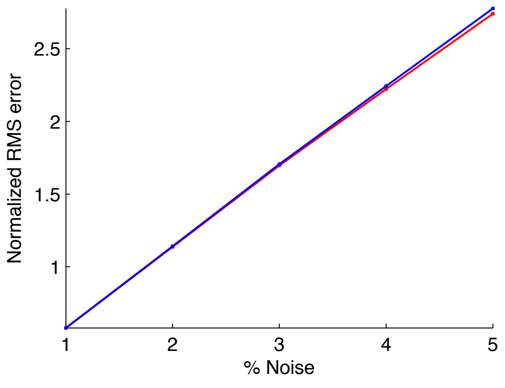
Figure 8.11: Average normalized RMS error of weighted distortion (blue) vs the Laplace-Beltrami operator (red) as a function of increasing resolution on (a) uniform grid (b) irregular and (c) diamond meshes, and also (d) as a function of the number of tetrahedra incident per vertex on irregular tessellations.

The first experiments show how the error with respect to ground truth values of mean curvature changes as a function of increasing resolution. Using the Root Mean Square (RMS) error normalized in each case by the range (maximum minus minimum) of analytic values taken by each function in the interval and averaged over the six functions. Figure 8.11 illustrates these results.

Figure 8.11(a) shows the result on uniform grids. It can be seen, while both operators converge, the weighted distortion does so more quickly, achieving an average reduction in error of 29% compared to the Laplace-Beltrami operator. On the non-grid tessellations, both operators are less well-behaved. This is mitigated by smoothing the estimates using 50 iterations of local averaging. Figures 8.11(b) and 8.11(c) show these results on the irregular tessellation and diamond meshes.



(a)



(b)

Figure 8.12: Average normalized RMS error of weighted distortion (blue) vs the Laplace-Beltrami operator (red) on regular grid tessellations of fixed resolution and increasing Gaussian noise. (a) Noise is added in the vertical direction with standard deviation as a percentage of the field range. (b) Noise is added in the surface normal direction with standard deviation as a percentage of the average edge length.

A further comparison, in the irregular tessellations, express the estimation error as a function of the number of tetrahedra incident on each vertex. For fixed incidence number, the normalized RMS error is evaluated over the vertices with this valence and average the results over all irregular tessellations and all functions. While both operators converge to the analytic values of mean curvature as vertex valence increases, the weighted distortion error is found to be much lower than that of the Laplace-Beltrami operator. This is illustrated in Figures 8.11(d).

Finally, the behavior of the operators under increasing noise is evaluated. For each uniformly sampled mesh, two forms of noise are added. In the first case, to simulate image noise, Gaussian noise is added to the field component of the vertex coordinates as a proportion of the range of analytic values. In the second, the Gaussian noise is added in the normal direction as a proportion of the average edge length. As Figure 8.12 illustrates, the operators behave very similarly under these conditions.

8.4.1.3 Application to segmentation

To further evaluate the proposed operator, an application to the segmentation of hypersurfaces in 4D space has been explored. The idea is to extend to 3D the intuition that shape boundaries often perceptually align with concavities, which correspond to regions of negative mean curvature. While this intuition is common, it should be noted that if one scales the function by a factor, the Euclidean curvature values may change in a relatively complicated way. Pottmann and Opitz argue that it may be more natural to use isotropic curvatures [PO94]. Based on the concavity intuition, a hierarchical Morse decomposition of the mean curvature field defined at the vertices of the tetrahedral mesh is applied. To finds segment centers at locations of high (positive) curvature and boundaries at areas of low (negative) curvatur, the descending Morse complex is used. Over-segmentation is countered applying hierarchical region-merging based on the notion of persistence, which relates to the "height" difference between adjacent segments.

Data sets considered are FUEL, NEGHIP and SILICIUM. They and their segmentation results using Laplace-Beltrami, weighted and unweighted distortion are shown in Figure 8.13. For each data set, we empirically chose a merging threshold that results in a number of segments between 20 and 40.

For each dataset, increasing artificial noise is added to the field, the Laplace-Beltrami and distortion fields is recomputed, and the segmentations re-obtained using the originally chosen threshold for each set. After that the similarity to the original segmentation is measured. For this last step is used the Hamming distance metric proposed by Huang and Dom [HD95, CGF09b]. It performs a volumetric analysis of the 3-cells of both the decompositions. The original metric defines the similarity between two 3-cells searching, by starting from a region s_1 in the first decomposition, the region s_2 in the second decomposition with the largest common volume. Thus, the value of such metric corresponds to the ratio between the number of tetrahedra assigned to the same 3-cell in both the decompositions and the total number of tetrahedra. It is also interesting to compare the ratio of the number of segments in each segmentation n/m, where $m > n$. While this "region number" metric is much less discriminative, it gives an intuitive sense of how the number of segments grows as a result of noise. Given that these images are originally captured on a regular grid, image noise is simulated by adding Gaussian noise to the field component of the data with standard deviation set as a percentage of the field range. The results are illustrated in Figure 8.14. As the curves show, the distortion operator in both its weighted and unweighted forms, maintains higher similarity to the noiseless segmentation under increasing noise.

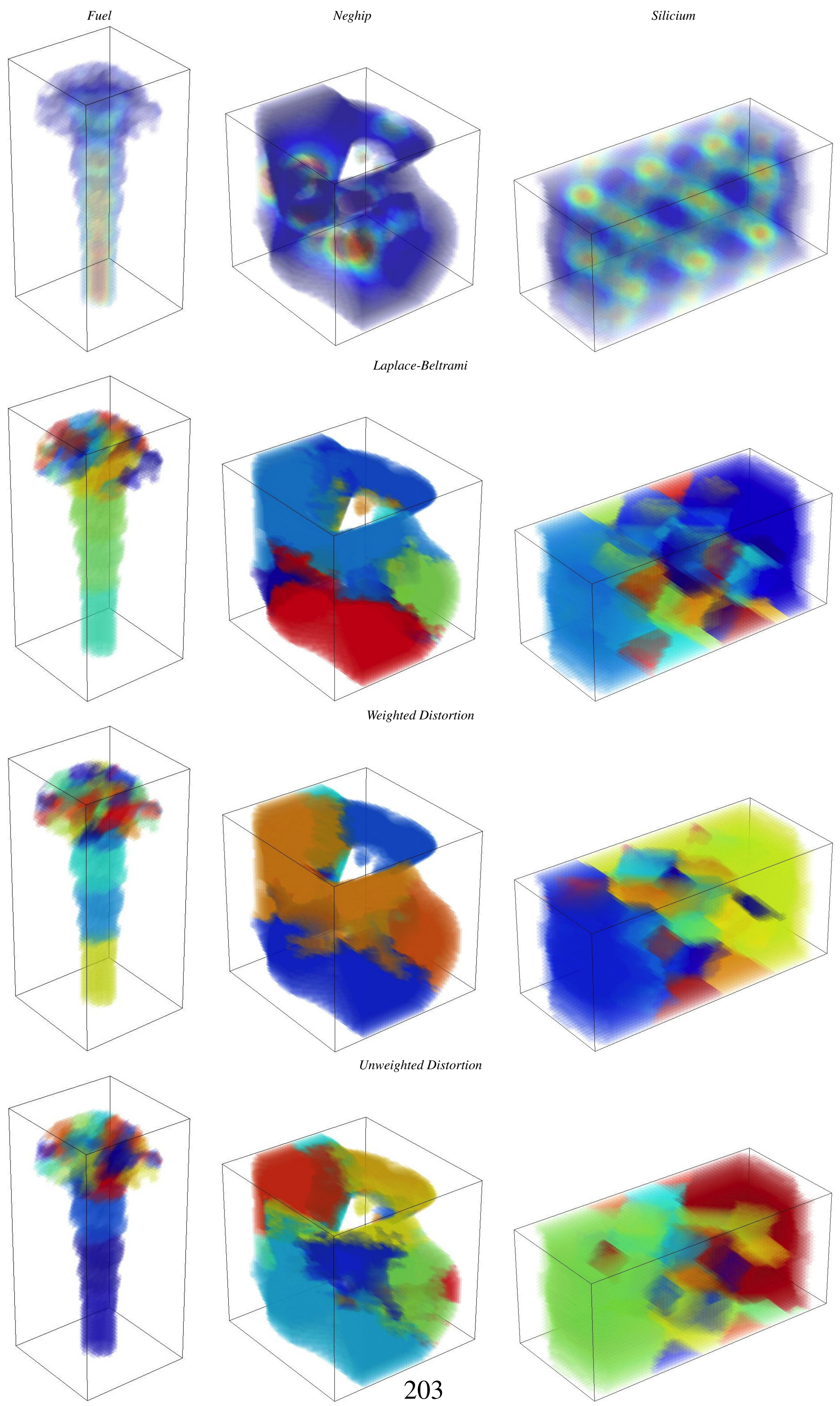
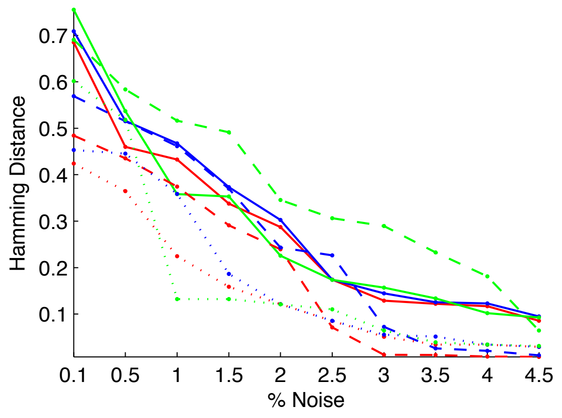


Figure 8.13: Segmentations of FUEL, NEGHIP, and SILICIUM volume datasets obtained using a hierarchical descending Morse complex on the Laplace-Beltrami, weighted and unweighted distortion fields.



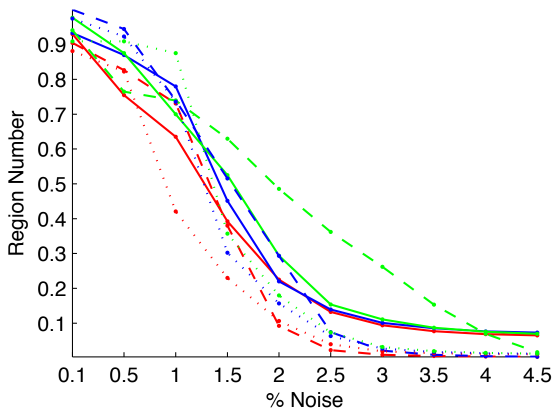


Figure 8.14: Comparison of the stability of the segmentations under increasing noise. Left: Hamming Distance, right: Region Number. Red: Laplace-Beltrami, blue: weighted distortion, green: unweighted distortion. Solid: FUEL, dashed: NEGHIP, dotted: SILICIUM. Note that, since we are measuring similarity to the noiseless case, higher is better. Also note the rate discontinuity in the x axis.

Chapter 9

Concluding remarks

In this thesis we have investigated the analysis of scalar fields focusing our attention on methods rooted in Morse theory. The main contributions of this thesis are in these directions:

- a dimension-independent representation, called the **Morse Incidence Graph** (MIG), for the ascending/descending Morse complexes and for the 1-skeleton of the Morse-Smale complex [CFI10];
- **primal/dual interpretation** of the descending and ascending Morse cells in terms of simplexes of the primal simplicial mesh and of the cells of its dual mesh [WIFF13];
- a compact encoding of the Forman gradient field on triangle and tetrahedral meshes s using **local frame** representation [WIFF13];
- the evaluation and comparison of the Forman-bases and watershed methods for computing Morse complexes on unstructured triangular and tetrahedral meshes [FIMS13];
- the definition and implementation of two dimension-independent simplification operators, called $remove_{i,i+1}$ and $remove_{i,i-1}$, on the MIG [CFI11];
- a theoretical and experimental comparison between the **remove** operators we have defined and i-$cancellation$ operator [CFI13b];
- the definition of a multi-resolution model for the morphology of scalar fields in any dimension, called the **Multi-resolution Morse complex**, and its implementation for the 2D and 3D cases [CFI12];
- the definition of a multi-resolution model for both the morphology and the geometry of two-dimensional scalar fields, called **Multi-Tessellation based on Forman Gradient**;

- the definition and implementation of the multi-resolution model for cell complexes, called the **Hierarchical cell complex**, and its application to the efficient computation of homology and homology generators [CFI13a, CFI13c];

- a first attempt to generalize the notion of **extrinsic distortion** in three and higher dimensions and investigation of new approaches to the analysis of 3D scalar fields through segmentation and analysis of 3-manifold hypersurfaces that are not graphs of 3D scalar fields [FIM$^+$12, SFIM13].

In Chapter 3, we have presented a primal/dual interpretation of the descending and ascending Morse cells in terms of simplexes of the primal simplicial mesh associated with a 3D scalar field and of the cells of its dual mesh. The MS complex has been described combinatorially as collections of cells from the dually subdivided mesh which is the intersection of the primal and dual ones. This led to simple descriptions of morphological features in terms of only the vertices and top simplexes of the primal mesh. We have also proposed a compact encoding of discrete vector fields using the *local frame* representation, which associates information with the top simplexes in the primal mesh, and which we apply to the discrete Morse gradient field.

In Chapter 4, we have described a simplification operator ($remove$) on the dimension-independent graph-based representation for Morse complexes described in Chapter 3.1, the MIG. The $remove$ operator has two important properties. First, it forms a basis for the set of topologically consistent operators on Morse complexes: any such operator can be expressed as a sequence of $remove$ operators. Second, it reduces the size of the MIG at each step, both in terms of the number of nodes and of the number of arcs. The i-$cancellation$ operator is guaranteed to reduce only the number of nodes in the MIG, but, in general, it increases the number of its arcs, thus increasing the total size of the MIG. We have presented the results of the experiments evaluating the $remove$ operator. From our experiments it follows that the majority of simplification operators evoked at the beginning of the simplification process are saddle-saddle operators, implying the importance of saddle-saddle operators that reduce the size of the Morse incidence graph at each step. We have compared the $remove$ operator with the existing simplification i-$cancellation$ operator defined in the literature. We have shown that the number of arcs in the graph simplified using i-$cancellation$ operator always exceeds the number of arcs in the graph simplified using $remove$ operator, thus influencing both the storage requirements and the time complexity of the simplification algorithm. We have applied the simplification algorithm using $remove$ operator on several 2D and 3D data sets, and we have evaluated the time complexity and memory requirements of the algorithm at different persistence thresholds. We have also defined and implemented a set of dimension-independent refinement operators, that are the inverse of the atomic and macro simplification operators defined in Chapter 4.

In Chapter 5, we have defined a multi-resolution representation for the two Morse complexes, based on the $remove$ operator and its undo refinement operator, called the *Multi-resolution Morse complex*. The $MRMC$ is dimension-independent. We have developed a graph-based model to encode an $MRMC$ and we have called this model *Multi-Resolution Morse Incidence Graph* ($MMIG$). An $MMIG$ is generated from the iterative generalization of the MIG at full resolution, and consists of the MIG representing the complexes at the coarsest resolution, of a set of refinement modifications reversing the simplification ones applied in the generalization phase, and of a dependency relation among such modifications. We have implemented two data structures for encoding an $MMIG$, in 2D and 3D, and we have compared their performances both from the time efficiency and storage cost points of view. To overcome the limitations of the $MRMC$ we have presented, in Chapter 7, a new multi-resolution model for scalar fields $\mathbb{M}_{\Sigma} = (\Sigma, f)$ defined on a triangle mesh Σ on which a Forman gradient V has been computed. The *Multi-Tessellation based on Forman Gradient ($MTFG$)* is able to support the extraction of representations of both Σ and V at different level of details. We have described the model as well as its encoding used in its implementation. We have compared the $MTFG$ with the only comparable model known in literature, the Multi-resolution Morse Triangulation (MMT) [DDFMV10]. Here, we have just present some preliminary results on synthetic data sets. We are currently performing extensive experiments on such model.

In Chapter 6, we have defined a hierarchical model, called the *Hierarchical Cell Complex (HCC)*, based on a set of homology-preserving operators (called $KiC(i+1)C$ *Kill i-cell (i+1)-cell*) and a set of homology-modifying operators (called $KiCiCycle$ *Kill i-cell and i-cycle*). We have defined the HCC in a dimension-independent way on arbitrary cell complexes and we have compared its 2D instance with a pyramidal model based on n-maps. The first advantage of the MCC over pyramidal models is its space efficiency. This is a consequence of the fact that incidence graph, which is the basis for the HCC, occupies less memory than an n-map representing the same complex [DH05]. The second advantage is a wider representation domain. Incidence graph can represent arbitrary cell complexes, while n-maps can represent (closed orientable) quasi-manifolds, which are a class of pseudo-manifolds. The third advantage of the HCC is its greater adaptivity, and thus a larger number of topological representations of the complex that can be extracted from it. In a pyramid based on a 2-map, a contraction kernel at level k describes a set of regions that are merged into one region from one level of the pyramid to the next one. This merging (a set of edge contractions) can be seen as a macro-operator, consisting of a sequence of edge contractions (i.e., $K0C1C$ operators). All the operators in this sequence are either performed (at levels greater than k), or neither of them is performed (at levels less than k). The removal kernel at level k describes the simplification of the boundaries of the merged regions. In the 2D instance of our multi-resolution model, we built the MCC using the same edge removal ($K1C2C$) and edge contraction ($K0C1C$) simplification operators, but we have considered each of them separately and we do not group them into macro-operators. Moreover, we have studied the performances of the MCC varying the simplification sequence used to built it. We have developed an implementation of the HCC, based on our homology-preserving operators, and we have demonstrated experimentally that distributing the simplifications throughout the cell complex, during the simplification sequence, results in fewer dependency relations among the nodes; thus their expressive power is higher and the storage cost is reduced. We used the homology-preserving HCC as the basis for computing the homology and its generators for a cell complex at various resolutions in an efficient and effective way. We have developed an algorithm for computing homology and homology generators with coefficients in $\mathbb{Z}_2$ on the base complex (the coarsest one) using existing methods. The advantages of our approach are that a homology-preserving HCC concentrates the challenging computation on the base complex (the one with fewer cells) reducing the complexity of the input. Homology generators are extracted at uniform or variable resolution levels performing, for each refinement μ, local updates in the neighborhood of μ.

In Chapter 8, we have presented an innovative approach to the analysis of 3D scalar fields based on the notion of discrete distortion, which generalizes discrete curvature to triangulated hypersurfaces in 4D space, and on Morse decomposition. We have proposed the use of a multi-resolution model based on clusters of tetrahedra, called diamonds, which enables the analysis of a the graph of 2D scalar fields through crack-free approximations encoded as tetrahedral meshes. One important aspect of using mesh-based multi-resolution models is that the scalar filed can be analyzed by using much fewer samples than in the full-resolution one. This facilitates our analysis of large 3D volume datasets by using significantly fewer resources. The other aspect that we have shown through our experiments is the utility of discrete distortion in analyzing approximated representation of the 3D scalar field, thus giving good insights about the field behavior already at low resolutions. We have also examined the previously-existing notions of distortion and note that they can be divided into *intrinsic* and *extrinsic* categories depending on whether they are defined using the interior angles or the dihedral angles of the tessellation. We have a presented a new discrete operator generalizing the notion of *extrinsic distortion* to nD. We have analyzed the behavior of the operator on 3-manifolds in 4D, comparing it to the well known Laplace-Beltrami operator, using ground-truth analytic surfaces with varying conditions of resolution, sampling distribution, and noise. We have also investigated it in the context of an application that uses the mean curvature field to obtain a volumetric segmentation, examining the stability of the segmentations under increasing image noise. In each case we showed that extrinsic distortion behaves similarly or better than the Laplace-Beltrami operator while being intuitively simple and easy to implement.

Current and future developments

Also thanks to the active community working in scalar fields analysis and since the promising results obtained during these three years, we are planning to study further many of the topics addressed during the thesis work.

As discussed in Section 3.2, a further analysis of the **representation** through the lens of our primal/dual interpretation reveals that the discrete vector field is a labeling of the 1-skeleton of the dually-subdivided mesh Σ_S. In this interpretation, there is a one-to-one correspondence between the nodes of Σ_S's 1-skeleton and the simplices of Σ, and between the edges of Σ_S's 1-skeleton and the incidence relations of Σ whose dimensions differ by one (i.e. the relations encoded in the Incidence Graph). The Forman gradient restrictions discussed in Section 1.4 imply that regular nodes in the 1-skeleton have valence one and critical nodes have valence zero. This interpretation opens the possibility of extending our representation in the immediate future to more general cell complexes, such as irregular hexahedral meshes. Both our Forman gradient encoding and our formulation of morphological features are independent of the topological data structure used to encode the mesh (as long as it encodes the vertices and the top simplexes), of the method in which the Forman gradient is computed and of the algorithms used for feature extraction. Thus, we are planning to investigate the application of such methods in higher dimensions. However, since the quick growth of possible valid cases for a Forman gradient V (97 in 2D and 51,030 in 3D), we are considering other encodings for the gradient. The encoding will be still based on the top simplexes of the representation, but it hould represent only the gradient arrows actually present in V, exploiting the restrictions in the pairings imposed by discrete Morse theory (i.e. if a vertex v_1 is paired with edge (v_1, v_2), (v_1, v_2) will be not paired with any other triangle or vertex in the triangle mesh Σ).

Regarding the **multi-resolution analysis**, we are currently working in improving the experimental evaluation of the *Multi-Tessellation based on Forman gradient* model performing various selective refinement queries at uniform and variable resolution levels. We are planning to improve considerably the geometry simplification, implementing tools well established in literature for mesh simplification (such as Quadric Error Metrics (QEM) [GH97]) in order to obtain better shapes during the simplification (and consequently refinement) step. Moving forward to real-time computation we would consider, as long term goal, multi-resolution structures alternative to the classic *Multi-Triangulation* in order to reduce the DAG traversal cost. Following the approach described in [CGG$^+$05] we could move the granularity, of our refinement steps, from triangles to precomputed patches including multiple vertex-split operators in one refinement macro-operator. A similar idea could be also applied to the topological refinements. Since the Forman gradient is a suitable tool for the homology computation we will adapt our model to this purpose computing homology and homology generators on the root of the model and extracting representations of the homology generators. This can be achieved with a minor effort since the boundary matrices used for the computation of homology are easily extracted from a Forman gradient V. As for the HHC complex 6, computing the boundary matrices on the base Forman gradient will improve the computation speed considerably. We are interested also that the geometrical representations extracted from an $MTFG$ would agree with the function f defined on its vertices. In other words, we would like that for each descending/ascending path in V, the function values in f would be descendent/ascendent. This problem has been studied in [BHEP04] and [WGS10]. We are planning to relax the problem addressed in [BHEP04] modifying the function f, at the end of a selective refinement query, computing the new function f' based on the gradient paths in the extracted gradient V'. As long term goal, we are considering to extend the model to the 3D case. The edge-collapse is defined also the tetrahedral meshes as it is for triangles. We should study carefully a gradient configuration, for a neighborhood of an edge-collapse, having the same good properties of the configuration described in Section 7. Moreover the simplification of V, in dimensions higher then two, presents a wider set of critical pairs that can be removed and, as discussed in [GRSW13] some of them could result in an invalid change in the topology of the critical points of the scalar field.

We are also performing experiments on a hierarchical cell complex based on both homology-preserving and homology-modifying operators. This version of the HCC is built from a sequence of homology-preserving operators until no more simplifications can be performed. Then, a homology-modifying operator is applied removing cells forming cycles and killing homologies from the higher to the lower one. The HCC obtained allows the extraction of cell complexes, from the base complex Γ_B, at different resolution and with different homologies with respect to the initial cell complex Γ. We are currently evaluating the likelihood of our model working with two-dimensional and three-dimensional cell complexes only. Moreover, we are considering the model as tool for the computation of the, so called, **localized homology** [ZC08]. The problem addressed in localized homology is the computation of homology generators, on a geometrical object, which are independent and as small as possible. Performing the homology computations on the base complex we could benefit, from the low number of cells in Γ_B, for retrieving the set of independent homology generators and we could exploit the refinement process for extracting homology generators with good quality. Independently from localized homology, we are also planning to extend the previous approaches to the computation of homology and homology generators with coefficients in $\mathbb{Z}_2$ adapting the HCC to simplicial complexes. We will consider two simplification operators for generating an HCC: *simplex collapse* [KMŚ98], which is an instance of simplification operator $KiC(i+1)C(q,p)$, and *edge contraction*, a widely used operator in mesh processing which has been proven to be homology-preserving [ALS12]. Since these operators are homology-preserving, we are able to built the corresponding multi-resolution model based on them. However, simplicial complexes have more restrictive topological constraints with respect to cell complexes, constraints that have to be maintained during the whole simplification process. Thus, the challenging part would be to guarantee a sufficient reduction of the mesh geometry in order to compute, fast enough, homology and homology generators of the base mesh.

In the end, regarding the analysis of 3D scalar fields based on **discrete distortion**, we are interested into increasing the robustness of the operator under conditions of irregular tessellation, such as using intelligent smoothing schemes based on vertex valence. Our method could also be applied to the segmentation and analysis of 3-manifold hypersurfaces that are not graphs of 3D scalar fields, as is fully permitted by the current formulation and implementation. Lastly, other applications of mean curvature in higher dimensions are open to investigation, including visualization, registration, matching, alignment, and simplification of volumetric datasets.

These new analysis techniques based on discrete distortion, the efficient computation of Morse and Morse-Smale complexes, and the efficient computation of homology and homology generators are different aspects that could be aggregated in a monolithic tool for the multi-resolution analysis of scalar fields. The extraction of the features of the Morse and Morse-Smale as well as homology generators complexes could give good insight on the structural characteristics of the scalar field, while distortion values computed could be treated as an additional scalar function g from which extract further information on the scalar field.

Appendix A - Datasets

In this appendix we collect some information about all the datasets we have used for the experimental evaluation of our work. We divide the datasets in three sets, 2D scalar fields (terrains), 3D scalar fields (volumetric datasets) and simplicial meshes (triangular and tetrahedral meshes). All the datasets we have used are courtesy of Aim@shape [`www.aimatshape.net`], Volvis [`www.volvis.org`] and VTerrain [`www.vterrain.org`] repositories.

The 2D scalar fields are all triangular meshes where a scalar function has been defined on its vertices. All the 2D scalar fields used are terrain data sets where the scalar function is the height function of the vertices.

- EGGS dataset is an analytic dataset deriving from the intersection of many Gaussians with a plane. It is composed by 103041 vertices and 204800 triangles
- MATTERHORN dataset is a terrain dataset representing the Matterhorn mountain. It is composed by 263169 vertices and 524288 triangles
- MONVISO is a terrain dataset representing the Monviso mountain. It is composed by 263169 vertices and 524288 triangles
- MONT BLANC is a terrain dataset representing the neighborhood near the Mont Blanc . It is composed by 263169 vertices and 524288 triangles
- LAKE MAGGIORE is a terrain dataset representing the area around the Lake Major. It is composed by 810000 vertices and 1616402 triangles
- MARCY is a subset of the bigger MOUNT MARCY dataset characterized by an irregular triangulation. It is composed by 928 vertices and 1738 triangles.
- MALLORCA TERRAIN is a portion of Mallorca island discretized on a regular grid. It is composed by 1825 vertices and 3454 triangles.
- USTICA dataset is a portion of the Ustica island. It is composed by 1128 vertices and 2111 triangles.

The 3D scalar fields are all tetrahedral meshes where a scalar function has been defined on its vertices. All the 3D scalar fields used are volume data sets generally obtained from scientific or medical analysis.

- ANALYTIC is, a synthetic dataset originated from the function $f(x, y, z) = sinx(x) + sin(y) + sin(z)$. It is composed by 274625 vertices and 1250235 tetrahedra.
- BUCKY is a carbon molecule having 60 atoms arranged in the form of a truncated icosahedron. It is composed by 262144 vertices and 1250235 tetrahedra.
- HYDROGEN consists of the high density region around the nucleus, two regions of high density on either side, and a torus of high density around the nucleus. It is composed by 544525 vertices and 3057196 tetrahedra.
- VISMALE is rotational C-arm x-ray scan of a human visage. It is composed by 619785 vertices and 3405126 tetrahedra.
- ANEURYSM is a rotational angiography scan of a head with an aneurysm. It is composed by 644114 vertices and 3616984 tetrahedra.
- SILICIUM is the simulation of a silicium grid. It is composed by 113288 vertices and 633798 tetrahedra.
- FUEL is the simulation of fuel injection into a combustion chamber. It is composed by 262144 vertices and 1500282 tetrahedra.
- NEGHIP is a simulation of the spatial probability distribution of electrons in a high potential protein molecule. It is composed by 262144 vertices and 1500282 tetrahedra.

The cell complexes used are all triangle and tetrahedral meshes that become cell complexes after undergoing simplifications.

- GENUS3 represents a 3-torus with homology (1,6,1). It is composed 40K cells.
- FERTILITY is a digital representation of the Fertility statue with homology (1,8,1). It is composed by 1.4M cells.
- VASELION is a digital representation of a vase with homology (1,0,1). It is composed by 1.2M cells.
- EROS is a digital representation of the Eros statue with homology (1,0,1). It is composed by 2.8M cells.

- HAND is a digital representation of the human and with homology (1,2,0). It is composed by 2.1M cells.

- BUDDHA is a digital representation of the Buddha statue with homology (1,208,1). It is composed by 3.2M cells.

- SKULL is a digital representation of a human skull with homology (1,2,1,0). It is composed by 748K cells.

- FERT-SOLID is a tetrahedralization of the fertility shape with homology (1,4,0,0). It is composed by 6.2M cells.

List of Figures

List of Tables

Bibliography

[AE98] U. Axen and H. Edelsbrunner. Auditory morse analysis of triangulated manifolds. In *Mathematical Visualization*, pages 223–236. Springer, 1998.

[Ago05] Max K. Agoston. *Computer Graphics and Geometric Modeling: Mathematics - ISBN:1-85233-817-2*. Springer-Verlag London Ltd., 2005.

[Ale57] P.S. Aleksandrov. *Topologia Combinatoria.* Torino, 1957.

[ALS12] D. Attali, A. Lieutier, and D. Salinas. Efficient data structure for representing and simplifying simplicial complexes in high dimensions. *Int. J. Comput. Geometry Appl.*, 22(4):279–304, 2012.

[And05] M. T. Anderson. Géométrisation des Variétés de Dimension 3 via le Flot de Ricci. *Société Mathématique de France, Gazette,*, 103:25–40, 2005.

[Ban67] T. Banchoff. Critical points and curvature for embedded polyhedra. *J. Diff. Geom*, 1(245-256):226, 1967.

[Ban70] T. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *American Mathematical Monthly*, 77(5):475–485, 1970.

[BCH$^+$07] M. Berg, O. Cheong, H. Haverkort, J.G. Lim, and L. Toma. I/o-efficient flow modeling on fat terrains. In F. Dehne, J.R. Sack, and N. Zeh, editors, *Algorithms and Data Structures*, volume 4619 of *Lecture Notes in Computer Science*, pages 239–250. Springer Berlin Heidelberg, 2007.

[Beu94] S. Beucher. Watershed, hierarchical segmentation and waterfall algorithm. In Jean Serra and Pierre Soille, editors, *Mathematical Morphology and Its Applications to Image Processing*, volume 2 of *Computational Imaging and Vision*, pages 69–76. Springer Netherlands, 1994.

[BFF$^+$08] S. Biasotti, L. De Floriani, B. Falcidieno, P. Frosini, D. Giorgi, C. Landi, L. Papaleo, and M. Spagnuolo. Describing shapes by geometrical-topological properties of real functions. *ACM Comput. Surv.*, 40(4), 2008.

[BFS02] S. Biasotti, B. Falcidieno, and M. Spagnuolo. Shape abstraction using computational topology techniques. In *From geometric modeling to shape modeling*, pages 209–222. Springer, 2002.

[BHEP04] P. T. Bremer, B. Hamann, H. Edelsbrunner, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385–396, July 2004.

[BHS80] I. C. Braid, R. C. Hillyard, and I. A. Stroud. Stepwise construction of polyhedra in geometric modelling. In K.W.Brodlie, editor, *Mathematical Methods in Computer Graphics and Design*, pages 123–141. Academic Press, 1980.

[BK00] L. Brun and W. G. Kropatsch. Introduction to Combinatorial Pyramids. In *Digital and Image Geometry*, volume 2243 of *Lecture Notes in Computer Science*, pages 108–128. Springer, 2000.

[BPH09] P. T. Bremer, V. Pascucci, and B. Hamann. Maximizing adaptivity in hierarchical topological models using cancellation trees. In T. Möller, B. Hamann, and R. D. Russell, editors, *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, Mathematics and Visualization, pages 1–18. Springer Berlin Heidelberg, 2009.

[BPS98] C. L. Bajaj, V. Pascucci, and D. R. Schikore. Visualization of scalar topology for structural enhancement. In *Proceedings of the conference on Visualization'98*, pages 51–58. IEEE Computer Society Press, 1998.

[Car76] M. P. Do Carno. *Differential Geometry of Curves and Surfaces*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1976.

[CDFI13] L. Comic, L. De Floriani, and F. Iuricich. Modeling three-dimensional morse and morse-smale complexes. In Michael Breuß, Alfred Bruckstein, and Petros Maragos, editors, *Innovations for Shape Analysis*, Mathematics and Visualization, pages 3–34. Springer Berlin Heidelberg, 2013.

[CF08] C. Chen and D. Freedman. Quantifying Homology Classes. In Susanne Albers and Pascal Weil, editors, *25th International Symposium on Theoretical Aspects of Computer Science*, volume 1 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 169–180, Dagstuhl, Germany, 2008. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[CF11] L. Comic and L. De Floriani. Dimension-independent simplification and refinement of morse complexes. *Graphical Models*, 73(5):261–285, 2011.

[CFI10] L. Comic, L. De Floriani, and F. Iuricich. Building morphological representations for 2d and 3d scalar fields. In Enrico Puppo, Andrea Brogni, and Leila De Floriani, editors, *Eurographics Italian Chapter Conference*, pages 103–110. Eurographics, 2010.

[CFI11] L. Comic, L. De Floriani, and F. Iuricich. Simplifying morphological representations of 2d and 3d scalar fields. In Isabel F. Cruz, Divyakant Agrawal, Christian S. Jensen, Eyal Ofek, and Egemen Tanin, editors, *GIS*, pages 437–440. ACM, 2011.

[CFI12] L. Comic, L. De Floriani, and F. Iuricich. Dimension-independent multi-resolution morse complexes. *Computers & Graphics*, 36(5):541–547, 2012.

[CFI13a] L. Comic, L. De Floriani, and F. Iuricich. Multi-resolution cell complexes based on homology-preserving euler operators. In Rocío González-Díaz, María José Jiménez, and Belén Medrano, editors, *DGCI*, volume 7749 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2013.

[CFI13b] L. Comic, L. De Floriani, and F. Iuricich. Simplification operators on a dimension-independent graph-based representation of morse complexes. In Cris L. Luengo Hendriks, Gunilla Borgefors, and Robin Strand, editors, *ISMM*, volume 7883 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2013.

[CFI13c] L. Comic, L. De Floriani, and F. Iuricich. Topological modifications and hierarchical representation of cell complexes in arbitrary dimensions. In *Computer Vision and Image Understanding*, page to appear, 2013.

[CFW11] D. Canino, L. De Floriani, and K. Weiss. Ia*: An adjacency-based representation for non-manifold simplicial shapes in arbitrary dimensions. *Computers & Graphics*, 35(3):747–753, 2011.

[CGF09a] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3D mesh segmentation. In *Proceedings of ACM SIGGRAPH*, page 73. ACM, 2009.

[CGF09b] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3D mesh segmentation. In *Proceedings of ACM SIGGRAPH*, page 73. ACM, 2009.

[CGG$^+$05] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. Batched multi triangulation. In *IEEE Visualization*, page 27. IEEE Computer Society, 2005.

[CKF03] J. Cox, D.B. Karron, and N. Ferdous. Topological zone organization of scalar volume data. *Journal of Mathematical Imaging and Vision*, 18(2):95–117, 2003.

[CMF11] L. Comic, M. M. Mesmoudi, and L. De Floriani. Smale-like decomposition and forman theory for discrete scalar fields. In Isabelle Debled-Rennesson, Eric Domenjoud, Bertrand Kerautret, and Philippe Even, editors, *DGCI*, volume 6607 of *Lecture Notes in Computer Science*, pages 477–488. Springer, 2011.

[CS92] X. Chen and F. Schmitt. Intrinsic surface properties from surface triangulation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 739–743, 1992.

[CSM03] D. Cohen-Steiner and J.-M. Morvan. Restricted Delaunay triangulations and normal cycle. In *Symposium on Computational Geometry*, pages 312–321, 2003.

[DDFM03] E. Danovaro, L. De Floriani, and M. M. Mesmoudi. Topological analysis and characterization of discrete scalar fields. In *Proceedings of the 11th International Conference on Theoretical Foundations of Computer Vision*, pages 386–402, Berlin, Heidelberg, 2003. Springer-Verlag.

[DDFMV10] E. Danovaro, L. De Floriani, P. Magillo, and M. Vitali. Multiresolution morse triangulations. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, SPM '10, pages 183–188, New York, NY, USA, 2010. ACM.

[DDFVM07] E. Danovaro, L. De Floriani, M. Vitali, and P. Magillo. Multi-scale dual morse complexes for representing terrain morphology. In *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*, GIS '07, pages 29:1–29:8, New York, NY, USA, 2007. ACM.

[DEGN99] T. K. Dey, H. Edelsbrunner, S. Guha, and D. V. Nekhayev. Topology preserving edge contraction. *Publ. Inst. Math.(Beograd)(NS)*, 66(80):23–45, 1999.

[DFKP05] L. De Floriani, L. Kobbelt, and E. Puppo. A survey on data structures for level-of-detail models. In NeilA. Dodgson, MichaelS. Floater, and MalcolmA. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 49–74. Springer Berlin Heidelberg, 2005.

[DGDP12] G. Damiand, R. Gonzalez-Diaz, and S. Peltier. Removal Operations in nD Generalized Maps for Efficient Homology Computation. In *Proc. of 4th International Workshop on Computational Topology in Image Context (CTIC)*, volume 7309 of *Lecture Notes in Computer Science*, pages 20–29. Springer Berlin/Heidelberg, 2012.

[DH05] L. De Floriani and A. Hui. Data Structures for Simplicial Complexes: An Analysis and a Comparison. In Mathieu Desbrun and Helmut Pottmann, editors, *Proceedings of the 3rd Eurographics Symposium on Geometry Processing*, volume 255 of *ACM International Conference Proceeding Series*, pages 119–128, Aire-la-Ville, Switzerland, 2005. Eurographics Association.

[DH07] L. De Floriani and A. Hui. Shape Representations Based on Cell and Simplicial Complexes. In J. Bittner D. Schmalstieg, editor, *Eurographics 2007, State-of-the-art Report*, pages 63–87. Eurographics Association, September 2007.

[DHKS13] T. K. Dey, A.l N. Hirani, B. Krishnamoorthy, and G. Smith. Edge contractions and simplicial homology. *CoRR*, abs/1304.0664, 2013.

[DHSJL01] N. Dyn, K. Hormann, K. Sun-Jeong, and D. Levin. Optimizing 3D triangulations using discrete curvature analysis. In *Mathematical Methods for Curves and Surfaces: Oslo 2000*, pages 135–146. Vanderbilt University, 2001.

[DL03] G. Damiand and P. Lienhardt. Removal and Contraction for n-Dimensional Generalized Maps. In *DGCI*, volume 2886 of *Lecture Notes in Computer Science*, pages 408–419. Springer, 2003.

[DSNW13] H. Doraiswamy, N. Shivashankar, V. Natarajan, and Y. Wang. Topological saliency. *Computers & Graphics*, 37(7):787 – 799, 2013.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[EHNP03] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-smale complexes for piecewise linear 3-manifolds. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, pages 361–370, New York, NY, USA, 2003. ACM.

[EHZ01] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical morse complexes for piecewise linear 2-manifolds. In *Proceedings of the Seventeenth Annual Symposium on Computational Geometry*, SCG '01, pages 70–79, New York, NY, USA, 2001. ACM.

[ELZ02] H. Edelsbrunner., D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

[EM90] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics (TOG)*, 9(1):66–104, 1990.

[EW79] C. M. Eastman and K. Weiler. Geometric Modeling Using the Euler Operators. In *1st Annual Conference on Computer Graphics in CAD/CAM Systems*, MIT, May 1979.

[FGH04] L. De Floriani, D. Greenfieldboyce, and A. Hui. A data structure for non-manifold simplicial d-complexes. In Jean-Daniel Boissonnat and Pierre Alliez, editors, *Symposium on Geometry Processing*, volume 71 of *ACM International Conference Proceeding Series*, pages 83–92. Eurographics Association, 2004.

[FIM$^+$12] L. De Floriani, F. Iuricich, P. Magillo, M. M. Mesmoudi, and K. Weiss. Discrete distortion for 3d data analysis. In Lars Linsen, Hans Hagen, Bernd Hamann, and Hans-Christian Hege, editors, *Visualization in Medicine and Life Sciences II*, Mathematics and visualization, pages 3–25. Springer, 2012.

[FIMS13] L. De Floriani, F. Iuricich, P. Magillo, and P. D. Simari. Discrete morse versus watershed decompositions of tessellated manifolds. In Alfredo Petrosino, editor, *ICIAP (2)*, volume 8157 of *Lecture Notes in Computer Science*, pages 339–348. Springer, 2013.

[FMP97] L. De Floriani, P. Magillo, and E. Puppo. Building and traversing a surface at variable resolution. In *IEEE Visualization*, pages 103–110, 1997.

[FMP99] L. De Floriani, P. Magillo, and E. Puppo. Multiresolution representation of shapes based on cell complexes (invited paper). In Gilles Bertrand, Michel Couprie, and Laurent Perroton, editors, *DGCI*, volume 1568 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 1999.

[For98] R. Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134:90–145, 1998.

[For02] R. Forman. A user's guide to discrete morse theory. *Sém. Lothar. Combin*, 48:B48c, 2002.

[GBHP11] A. Gyulassy, P.T. Bremer, B. Hamann, and V. Pascucci. Practical considerations in morse-smale complex computation. In Valerio Pascucci, Xavier Tricoche, Hans Hagen, and Julien Tierny, editors, *Topological Methods in Data Analysis and Visualization*, Mathematics and Visualization, pages 67–78. Springer Berlin Heidelberg, 2011.

[GG06] T. Gatzke and C. Grimm. Estimating curvature on triangular meshes. *Int Journal of Shape Modeling*, 12(1):1–29, 2006.

[GH97] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[GN05] A. Gyulassy and V. Natarajan. Topology-based simplification for feature extraction from 3d scalar fields. In *Visualization, 2005. VIS 05. IEEE*, pages 535–542, 2005.

[GNP$^+$06] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. A topological approach to simplification of three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):474–484, 2006.

[GNPH07] A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann. Efficient computation of morse-smale complexes for three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1440–1447, November 2007.

[Gom04] A. J. P. Gomes. Euler operators for stratified objects with incomplete boundaries. In *Proceedings of the ninth ACM symposium on Solid modeling and applications*, SM '04, pages 315–320, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[GP12] A. Gyulassy and V. Pascucci. Computing Simply-Connected Cells in Three-DimensionalMorse-Smale Complexes. In R. Peikert, H. Hauser, H. Carr, and R. Fuchs, editors, *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications*, Mathematics and Visualization, pages 31–46. Springer Verlag, Heidelberg, 2012.

[GR10] T. Gurung and J. Rossignac. SOT: Compact Representation for Triangle and Tetrahedral Meshes. Technical Report GT-IC-10-01, College of Computing, Georgia Institute of Technology, Atlanta, GA, USA, 2010.

[GRSW13] D. Günther, J. Reininghaus, H.-P. Seidel, and T. Weinkauf. Notes on the simplification of the morse-smale complex. In *Proc. TopoInVis*, Davis, U.S.A., March 2013.

[GRWH12] D. Günther, J. Reininghaus, H. Wagner, and I. Hotz. Efficient computation of 3d morse-smale complexes and persistent homology using discrete morse theory. *The Visual Computer*, 28(10):959–969, 2012.

[Ham92] B. Hamann. Curvature approximation for triangulated surfaces. In Gerald E. Farin, Hans Hagen, Hartmut Noltemeier, and Walter Knödel, editors, *Geometric Modelling*, volume 8 of *Computing Supplement*, pages 139–153. Springer, 1992.

[Ham94a] B. Hamann. Curvature approximation of 3D manifolds in 4D space. *Computer Aided Geometric Design*, 11(6):621–633, 1994.

[Ham94b] B. Hamann. Curvature approximation of 3D manifolds in 4D space. *Computer Aided Geometric Design*, 11(6):621–633, 1994.

[Hat01] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2001.

[HBB$^+$09] S. Hahmann, A. Belayev, L. Busé, G. Elber, B. Mourrain, and C. Rössl. Shape interrogation. In L. De Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*, Series on Mathematics and Visualization, pages 1–51. Springer-Verlag, 2009.

[HD95] Q. Huang and B. Dom. Quantitative methods of evaluating image segmentation. *Image Processing, 1995. Proceedings., International Conference on*, 3:53–56, 1995.

[HG99] P. S. Heckbert and M. Garland. Optimal triangulation and quadric-based surface simplification. *Comput. Geom. Theory Appl.*, 14(1-3):49–65, 1999.

[HMM$^+$10] S. Harker, K. Mischaikow, M. Mrozek, V. Nanda, H. Wagner, M. Juda, and P. Dlotko. The efficiency of a homology algorithm based on discrete morse theory and coreductions. In *Proceedings 3rd International Workshop on Computational Topology in Image Context (CTIC 2010). Image A*, volume 1, pages 41–47, 2010.

[HMMN14] S. Harker, K. Mischaikow, M. Mrozek, and V. Nanda. Discrete morse theoretic algorithms for computing homology of complexes and maps. *Foundations of Computational Mathematics*, 14(1):151–184, 2014.

[JKG07] M. Jin, J. Kim, and X.D. Gu. Discrete surface Ricci flow: Theory and applications. In *Proceedings 12th IMA international conference on Mathematics of surfaces XII*, volume 4647 of *Lecture Notes in Computer Science*, pages 209–232, 2007.

[KKM05] H. King, K. Knudson, and N. Mramor. Generating Discrete Morse Functions from Point Data. *Experimental Mathematics*, 14(4):435–444, 2005.

[KMŚ98] T. Kaczyński, M. Mrozek, and M. Ślusarek. Homology computation by reduction of chain complexes. *Computers and Mathematics with Applications*, 35(4):59 – 70, 1998.

[KNSS09] E. Kalogerakis, D. Nowrouzezahrai, P. Simari, and K. Singh. Extracting lines of curvature from noisy point clouds. *Comput. Aided Des.*, 41(4):282–292, April 2009.

[KSNS07] E. Kalogerakis, P. Simari, D. Nowrouzezahrai, and K. Singh. Robust statistical estimation of curvature on discretized surfaces. In *Proceedings of the Symposium on Geometry Processing*, pages 13–22, 2007.

[LBS07] T. Langer, A. G. Belyaev, and H.-P. Seidel. Exact and interpolatory quadratures for curvature tensor estimation. *Computer Aided Geometric Design*, pages 443–463, 2007.

[LDB05] G. Lavoue, F. Dupont, and A. Baskurt. A new cad mesh segmentation method, based on curvature tensor analysis. *Computer-Aided Design (CAD)*, 37(10):975–987, 2005.

[Lie94] P. Lienhardt. N-Dimensional Generalized Combinatorial Maps and Cellular Quasi-Manifolds. *International Journal of Computational Geometry and Applications*, 4(3):275–324, 1994.

[LL01] S.H. Lee and K. Lee. Partial entity structure: a fast and compact non-manifold boundary representation based on partial topological entities. In *Proceedings Sixth ACM Symposium on Solid Modeling and Applications*, pages 159–170. Ann Arbor, Michigan, June 2001.

[LPRM02] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *ACM SIGGRAPH*, pages 362–371. ACM Press, 2002.

[LPT$^+$03] H. Lopes, S. Pesco, G. Tavares, M. Maia, and A. Xavier. Handlebody Representation for Surfaces and Its Applications to Terrain Modeling. *International Journal of Shape Modeling*, 9(1):61–77, 2003.

[LT97] H. Lopes and G. Tavares. Structural Operators for Modeling 3-Manifolds. In *Proceedings Fourth ACM Symposium on Solid Modeling and Applications*, pages 10–18. ACM Press, May 1997.

[LW69] A. T. Lundell and S. Weingram. *The topology of CW complexes*. Van Nostrand Reinhold Company, 1969.

[Mag98] P. Magillo. *Spatial Operations on Multiresolution Cell Complexes*. PhD thesis, PhD thesis, Department of Computer Science-University of Genova, 1998.

[Man88] M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, 1988.

[Mas93] H. Masuda. Topological Operators and Boolean Operations for Complex-Based Non-Manifold Geometric Models. *Computer-Aided Design*, 25(2):119–129, feb 1993.

[Mat02] Y. Matsumoto. An introduction to morse theory. translations of mathematical monographs, vol. 208. *American Mathematical Society*, 2002.

[MB09] M. Mrozek and B. Batko. Coreduction homology algorithm. *Discrete & Computational Geometry*, 41(1):96–118, 2009.

[McA99] M. McAllister. A watershed algorithm for triangulated terrains. In *CCCG*, 1999.

[MDDF$^+$08] P. Magillo, E. Danovaro, L. De Floriani, L. Papaleo, and M. Vitali. A discrete approach to compute terrain morphology. In José Braz, Alpesh Ranchordas, HélderJ. Araújo, and JoãoMadeiras Pereira, editors, *Computer Vision and Computer Graphics. Theory and Applications*, volume 21 of *Communications in Computer and Information Science*, pages 13–26. Springer Berlin Heidelberg, 2008.

[MDFM08] M. M. Mesmoudi, L. De Floriani, and P. Magillo. Morphological analysis of terrains based on discrete curvature and distortion. In *Proceedings of International Conference on Advances in Geographic Information Systems (ACMGIS 2008), Irvine, California, USA*, 2008.

[MDFM09] M. M. Mesmoudi, L. De Floriani, and P. Magillo. Morphology analysis of 3D scalar fields based on Morse theory and discrete distortion. In *ACM International Symposium on Advances in Geographic Information Systems*, Seattle, November 2009.

[MDFM12] M. M. Mesmoudi, L. De Floriani, and P. Magillo. Concentrated Curvature for Mean Curvature Estimation in Triangulated Surfaces. In *CTIC*, pages 79–87, 2012.

[MDFP08] M. M. Mesmoudi, L. De Floriani, and U. Port. Discrete distortion in triangulated 3-manifolds. *Computer Graphics Forum (Proceedings SGP 2008)*, 27(5):1333–1340, 2008.

[MDSB03] M. Meyer, M. Desbrun, M. Schroder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In H.-C. Hege and K. Polthier, editors, *Proceedings VisMath 2002*, pages 35–57, 2003.

[Mey94] F. Meyer. Topographic distance and watershed lines. *Signal Process.*, 38(1):113–125, July 1994.

[MFI13] P. Magillo, L. De Floriani, and F. Iuricich. Morphologically-aware elimination of flat edges from a tin. In Craig A. Knoblock, Markus Schneider, Peer Kröger, John Krumm, and Peter Widmayer, editors, *SIGSPATIAL/GIS*, pages 244–253. ACM, 2013.

[MGP06] N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3D geometry. *ACM SIGGRAPH*, 25(3):560–568, 2006.

[Mil63] J. Milnor. *Morse Theory*. Princeton University Press, New Jersey, 1963.

[MS82] M. Mantyla and R. Sulonen. Gwb: A Solid Modeler with Euler Operators. *IEEE Computer Graphics and Applications*, 2:17–31, 1982.

[MSNK89] H. Masuda, K. Shimada, M. Numao, and S. Kawabe. A Mathematical Theory and Applications of Non-Manifold Geometric Modeling. In *IFIP WG 5.2/GI International Symposium on Advanced Geometric Modeling for Engineering Applications*, pages 89–103, Berlin, Germany, nov 1989. North-Holland.

[Mun84] J. R. Munkres. *Elements of algebraic topology*, volume 2. Addison-Wesley Reading, 1984.

[MW99] A. P. Mangan and R. T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, October 1999.

[NC03] L. Najman and M. Couprie. Watershed algorithms and contrast preservation. In Ingela Nyström, Gabriella Sanniti di Baja, and Stina Svensson, editors, *Discrete Geometry for Computer Imagery*, volume 2886 of *Lecture Notes in Computer Science*, pages 62–71. Springer Berlin Heidelberg, 2003.

[NGH04] X. Ni, M. Garland, and J. C. Hart. Fair Morse Functions for Extracting the Topological Structure of a Surface Mesh. In *International Conference on Computer Graphics and Interactive Techniques ACM SIGGRAPH*, pages 613–622, 2004.

[Nie97] G. M. Nielson. Tools for triangulations and tetrahedralizations and constructing functions defined over them. In G. M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization: overviews, Methodologies and Techniques*, chapter 20, pages 429–525. IEEE Computer Society, Silver Spring, MD, 1997.

[NP05] V. Natarajan and V. Pascucci. Volumetric data analysis using morse-smale complexes. In *SMI*, pages 322–327. IEEE Computer Society, 2005.

[OBS04] Y. Ohtake, A. Belyaev, and H. P. Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on Graphics*, 23(3):609–612, August 2004.

[Pap04] L. Papaleo. *Surface reconstruction: online mosaicing and modeling with uncertainty*. PhD thesis, PhD thesis, Department of Computer Science-University of Genova, 2004.

[PBCF93] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1):56–102, January 1993.

[PH08] F. Ponchio and K. Hormann. Interactive Rendering of Dynamic Geometry. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):914–925, jul 2008.

[PIH$^+$07] S. Peltier, A. Ion, Y. Haxhimusa, W. G. Kropatsch, and G. Damiand. Computing Homology Group Generators of Images Using Irregular Graph Pyramids. In *GbRPR*, volume 4538 of *Lecture Notes in Computer Science*, pages 283–294. Springer, 2007.

[PO94] H. Pottmann and K. Opitz. Curvature analysis and visualization for functions defined on euclidean spaces or surfaces. *Computer Aided Geometric Design*, 11(6):655–674, 1994.

[PSKA02] D. L. Page, Y. Sun, A.F. Koschan, and M.A. Abidi. Normal vector voting: crease detection and curvature estimation on large, noisy meshes. *Graphical Models*, 64(3/4), 2002.

[PWHY09] H. Pottmann, J. Wallner, Q. Huang, and Y.-L. Yang. Integral invariants for robust geometry processing. *Comput. Aided Geom. Design*, 26:37–60, 2009.

[Ran71] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[Reg61] T. Regge. General relativity without coordinates. *Nuovo Cimento*, 19(3):558–571, 1961.

[RL01] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proceedings of 3D Digital Imaging and Modeling*, pages 145–152, 2001.

[RM00] Jos B.T.M. Roerdink and A. Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundam. Inf.*, 41(1,2):187–228, April 2000.

[Rus04] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Proceedings of the Symposium on 3D Data Processing, Visualization, and Transmission*, September 2004.

[RWP06] M. Reuter, F-E Wolter, and N. Peinecke. Laplace-Beltrami spectra as "Shape-DNA" of surfaces and solids. *Computer-Aided Design*, 38:342–366, 2006.

[RWS11] V. Robins, P. J. Wood, and A. P. Sheppard. Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1646–1658, 2011.

[Sam06] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. The Morgan Kaufmann series in computer graphics and geometric modeling. Morgan K., 2006.

[SDL05] C. Simon, G. Damiand, and P. Lienhardt. Pyramids of n-Dimensional Generalized Maps. In *Proc. of 5th Workshop on Graph-Based Representations in Pattern Recognition (GBR)*, volume 3434 of *Lecture Notes in Computer Science*, pages 142–152. Springer Berlin/Heidelberg, 2005.

[SFIM13] P. D. Simari, L. De Floriani, F. Iuricich, and M. M. Mesmoudi. Generalized extrinsic distortion and applications. *Computers & Graphics*, 37(6):582–588, 2013.

[Sha06] A. Shamir. Segmentation and shape extraction of 3d boundary meshes. In *EG 2006 - State of the Art Reports*, Vienna, 2006.

[She01] A. Sheffer. Model simplification for meshing using face clustering. *Computer-Aided Design*, 33(13):925–934, 2001.

[SMN12] N. Shivashankar, S. Maadasamy, and V. Natarajan. Parallel computation of 2d morse-smale complexes. *IEEE Trans. Vis. Comput. Graph.*, 18(10):1757–1770, 2012.

[SMS$^+$03] T. Surazhsky, E. Magid, O. Soldea, G. Elber, and E. Rivlin. A comparison of gaussian and mean curvatures estimation methods on triangular meshes. In *Proceedings of Conference on Robotics and Automation, Proceedings. ICRA '03. IEEE International*, volume 1, pages 739–743, 2003.

[SN12] N. Shivashankar and V. Natarajan. Parallel computation of 3d morse-smale complexes. *Comput. Graph. Forum*, 31(3):965–974, 2012.

[Soi03] P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2 edition, 2003.

[SS00] S. L. Stoev and W. Straßer. Extracting regions of interest applying a local watershed transformation. In *Proceedings of the Conference on Visualization '00*, VIS '00, pages 21–28, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[SW92] E.M. Stokely and S.Y.N.A. Wu. Surface parametrization and curvature measurement of arbitrary 3-D objects: five practical methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 833–839, 1992.

[SW04] B. Schneider and J. Wood. Construction of metric surface networks from raster-based dems. *Topological Data Structures for Surfaces: An Introduction to Geographical Information Science*, pages 53–70, 2004.

[Tau95] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings ICCV'95*, pages 902–907, 1995.

[TIS$^+$95] S. Takahashi, T. Ikeda, Y. Shinagawa, T. L. Kunii, and M. Ueda. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. *Computer Graphics Forum*, 14(3):181–192, 1995.

[TP12] J. Tierny and V. Pascucci. Generalized topological simplification of scalar fields on surfaces. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2005–2013, 2012.

[Tro86] M. Troyanov. Les surfaces Euclidiennes à singularités coniques. *L'enseignement Mathématique*, 32:79–94, 1986.

[Vit10] M. Vitali. *Morse Decomposition of Geometric Meshes with Applications*. PhD thesis, PhD thesis, Department of Computer Science-University of Genova, 2010.

[VS91] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(6):583–598, June 1991.

[WD09] K. Weiss and L. De Floriani. Supercubes: A high-level primitive for diamond hierarchies. *IEEE Transactions on Visualization and Computer Graphics (Proceedings IEEE Visualization 2009)*, 15(6):1603–1610, 2009.

[WDF11] K. Weiss and L. De Floriani. Simplex and Diamond Hierarchies: Models and Applications. *Computer Graphics Forum*, 30(8):2127–2155, 2011.

[WDFM10] K. Weiss, L. De Floriani, and M. M. Mesmoudi. Multiresolution Analysis of 3D Images Based on Discrete Distortion. In *Proceedings of the International Conference on Pattern Recognition*, pages 4093–4096, 2010.

[WGS10] T. Weinkauf, Y. Gingold, and O. Sorkine. Topology-based smoothing of 2d scalar fields with c1-continuity. In *Proceedings of the 12th Eurographics / IEEE - VGTC Conference on Visualization*, EuroVis'10, pages 1221–1230, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[WIFF13] K. Weiss, F. Iuricich, R. Fellegara, and L. Floriani. A primal/dual representation for discrete morse complexes on tetrahedral meshes. *Comput. Graph. Forum*, 32(3):361–370, 2013.

[WLH85] L. T. Watson, T. J. Laffey, and R. M. Haralick. Topographic classification of digital image intensity surfaces using generalized splines and the discrete cosine transformation. *Computer Vision, Graphics, and Image Processing*, 29(2):143–167, 1985.

[WS04] G. H Weber and G. Scheuermann. Automating transfer function design based on topology analysis. In *Geometric Modeling for Scientific Visualization*, pages 293–305. Springer, 2004.

[WSHH02] G. H. Weber, G. Scheuermann, H. Hagen, and B. Hamann. Exploring scalar fields using critical isovalues. In *Visualization, 2002. VIS 2002. IEEE*, pages 171–178. IEEE, 2002.

[YJLG08] X. Yin, M. Jin, F. Luo, and X. D. Gu. Discrete curvature flows for surfaces and 3-manifolds. In F. Nielsen, editor, *ETVC*, volume 5416 of *Lecture Notes in Computer Science*, pages 38–74. Springer, 2008.

[YVKS96] S. Yu, M. Van Kreveld, and J. Snoeyink. Drainage queries in tins: from local to global and back again. In *Proc. 7th Int. Symp. on Spatial Data Handling, pages A*, volume 13. Citeseer, 1996.

[ZC08] A. Zomorodian and G. E. Carlsson. Localized homology. *Comput. Geom.*, 41(3):126–148, 2008.