A Minimal Contouring Approach to the Computation of the Reeb Graph

Giuseppe Patanè, Michela Spagnuolo and Bianca Falcidieno

Abstract—Given a manifold surface $\mathcal{M}$ and a continuous scalar function $f : \mathcal{M} \to \mathbb{R}$, the Reeb graph of $(\mathcal{M}, f)$ is a widely-used high-level descriptor of $\mathcal{M}$ and its usefulness has been demonstrated for a variety of applications, which range from shape parameterization and abstraction to deformation and comparison. In this context, we propose a novel contouring algorithm for the construction of a discrete Reeb graph with a minimal number of nodes, which correspond to the critical points of f (i.e., minima, maxima, saddle points) and its level-sets passing through the saddle points. In this way, we do not need to sample, sweep, or increasingly sort the f-values. Since most of the computation uses only local information on the mesh connectivity, equipped with the f-values at the surface vertices, the proposed approach is insensitive to noise and requires a small memory footprint and temporary data structures. Furthermore, we maintain the parametric nature of the Reeb graph with respect to the input scalar function and we efficiently extract the Reeb graph of time-varying maps. Indicating with n and s the number of vertices of $\mathcal{M}$ and saddle points of f, the overall computational cost $O(sn)$ is competitive with respect to the $O(n \log n)$ cost of previous work. This cost becomes optimal if $\mathcal{M}$ is highly-sampled or $s \leq \log n$, as it happens for Laplacian eigenfunctions, harmonic maps and 1-forms.

Index Terms—Reeb graph, topological graph, Morse theory, computational topology, geometric algorithms, hierarchical segmentations, shape analysis and abstraction.

1 Introduction

Differential topology provides a suitable framework for formalizing and solving several problems related to shape understanding due to the theoretical link between critical points, their configuration, and the topological properties of the input surface. Given a scalar function $f : \mathcal{M} \to \mathbb{R}$, defined on a manifold surface $\mathcal{M}$, the topology of $\mathcal{M}$ can be effectively studied by contracting the connected components of the level-sets of f (i.e., $\gamma_\alpha := \{\mathbf{p} \in \mathcal{M} : f(\mathbf{p}) = \alpha\}$) to single points (i.e., the Reeb graph) or joining its critical points (i.e., $\{\mathbf{p} \in \mathcal{M} : \nabla f(\mathbf{p}) = \mathbf{0}\}$) with flow-lines of f (i.e., the Morse-Smale complex). The Reeb graph and the Morse-Smale complex provide an abstract representation of the surface $\mathcal{M}$. The former is based on the topological changes of the level-sets of f, the latter mainly uses the connectivity of the critical points with respect to the behavior of the gradient field of f. The most interesting aspect of this approach is its parametric nature: changing f we have different descriptions of the same surface $\mathcal{M}$ which highlight different "local" features while preserving the "global" topological structure of $\mathcal{M}$.

In this paper, we focus our attention on the computational aspect of the construction of the *Reeb graph* $\mathcal{R}_G$ of $\mathcal{M}$ with respect to a smooth scalar function $f : \mathcal{M} \to \mathbb{R}$. Formally, the Reeb graph is defined as the quotient space of $\mathcal{M} \times \mathbb{R}$ induced by the relation " $\sim$ " with $(\mathbf{p}, f(\mathbf{p})) \sim (\mathbf{q}, f(\mathbf{q})) \leftrightarrow f(\mathbf{p}) = f(\mathbf{q})$, and $\mathbf{p}, \mathbf{q}$ belong to the same connected component of the iso-contour $f^{-1}(f(\mathbf{p}))$ [39]. For scalar functions defined on simply connected domains, the Reeb graph is also simply connected and is called *contour-tree* [4], [9]. The contour-tree is a fundamental data structure for understanding and representing the behavior of phenomena in scientific visualization [32], automatically designing transfer functions [48], accelerating the extraction [47] and simplification [10] of level-sets.

A variety of functions have been used in several applications, which range from surface remeshing and parameterization to shape comparison. For instance, the *height* [18], [41] and *elevation* function [20] are the most intuitive and simple choices for analyzing 3D shapes. The *geodesic* [16], [20], [28] and *Euclidean* [18], [22] *distance* identify the surface protrusions [19] by computing the geodesic and Euclidean distance of the mesh vertices from selected feature points. *Curvature-based functions* have been frequently used to classify the local shape of 3D surfaces into planar, parabolic, and elliptic regions. Finally, their sensibility to the noise, small features, and quality of the shape discretization is reduced by applying a least-squares approximation with polynomial functions [53] or a multi-scale curvature evaluation [29]. A large number of functions are also generated by solving differential equations related to simulation problems (e.g., the Laplace and heat equation [3], [30]), decomposing the spectrum of data dependent kernels [3], and using harmonic 1-forms [21].

As guaranteed by Morse theory [27], the topological changes of the level-sets occur only at the critical points of f. We briefly remind that a critical point corresponds to a maximum, minimum, or saddle of f; otherwise, it is called *regular*. If $\mathbf{p}$ is a critical or a regular point,

The authors are with the Istituto di Matematica Applicata e Tecnologie Informatiche, Consiglio Nazionale delle Ricerche, Genova - Italy. E-mail:{patane,spagnuolo,falcidieno}@ge.imati.cnr.it.

then $f(\mathbf{p})$ is called *critical* or *regular iso-value*, respectively.

The *terminal nodes* of the graph represent the creation/deletion of contour cycles at a local extremum of f and the *branching nodes* locate the joining and/or splitting of two or more contour cycles at saddle points. Since the topology of the iso-contours changes only at the critical points, the *internal nodes* of $\mathcal{R}_G$ represent a family of contours that do not change topology.

Given a Morse function, the Reeb graph of a closed 2-manifold surface of genus g has g loops [12]. We remind that a scalar function is Morse if its critical points are not degenerate; i.e., the corresponding Hessian matrix is not singular. If $\mathcal{M}$ has b boundary components, then the Reeb graph has between g and $2g + b - 1$ loops. Therefore, this relation provides a fundamental link among the topology of $\mathcal{M}$, the differential properties of f, and the combinatorial structure of the corresponding Reeb graph.

The Reeb graph can be enriched with flow lines that create a control network for surface encoding and approximation [45]. In local parameterization, the Reeb graph has been used to decompose the input surface into a family of 0-genus patches, which can be efficiently parameterized [34], [54]. In global parameterization, the topological properties of the Reeb graph has been used to compute the generators [23], [36], [43] and cut-graphs of arbitrary 3D shapes [35], [42], [44]. The Reeb graph is also useful to identify and remove small tunnels of 3D models [50], which are considered defect with respect to the targeted description of the surface. Additional applications include surface deformations [25], [49], [51], [52], surface encoding [24], [41], [45], and shape similarity [6], [7], [22]. Skeletons enriched with topological information, and in particular the Reeb graph, has been recently used for reparing solid models [38], [50].

1.1 Previous work

Several algorithms have been proposed for the computation of the Reeb graph for surfaces or volumetric data. Most of them work by tracking the evolution of the level-sets, either with a suitable sampling of the image of f or with a complete sweeping of the image domain.

In the discrete settings, we consider a triangulated surface $\mathcal{M} := \{M, T\}$, where $M := \{\mathbf{p}_i \in \mathbb{R}^3, i = 1, \ldots, n\}$ is a set of n vertices and T is a triangular mesh. The function f on $\mathcal{M}$ is defined by the values $\{f(\mathbf{p}_i)\}_{i=1}^n$ of f at the vertices and extended by linear interpolation over $\mathcal{M}$. In the following, we will use a color-map to show the high and low function values in red and blue, respectively. Let $I := [f_{\min}, f_{\max}]$ be the interval containing the *image* of f, i.e. $\text{Image}(f) := \{f(\mathbf{p}), \mathbf{p} \in \mathcal{M}\}$, where $f_{\min} := \min_{\mathbf{p}\in\mathcal{M}}\{f(\mathbf{p})\}$ and $f_{\max} := \max_{\mathbf{p}\in\mathcal{M}}\{f(\mathbf{p})\}$ are the global extrema of f.

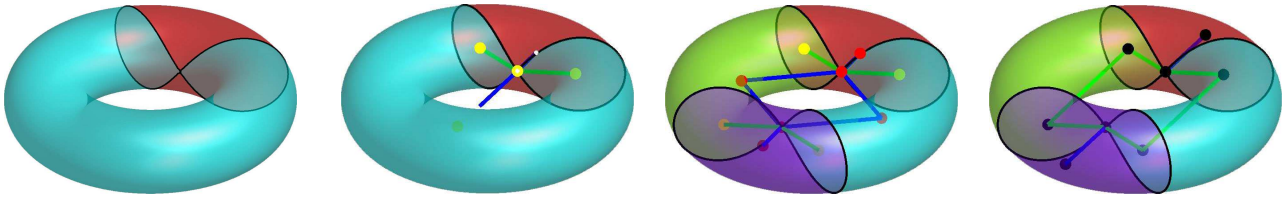
Sampling-based approaches [41] consider a partition $\mathcal{I} := \{[f_i, f_{i+1}]\}_{i=0}^k$ of I such that

$$\begin{cases} f_0 := f_{\min},\ f_i < f_{i+1},\ f_{k+1} := f_{\max}, \\ \bigcup_{i=0}^k [f_i, f_{i+1}] = [f_{\min}, f_{\max}]. \end{cases}$$

Then, the Reeb graph is built by using the ordered set $C_f(\mathcal{M}) := \{f^{-1}(f_i)\}_{i=0}^{k+1}$ of iso-contours together with the adjacency relations among the corresponding *surface strips* $\{f^{-1}([f_i, f_{i+1}])\}_{i=0}^k$. These approaches usually assume that the level-sets do not interpolate critical points. If the previous condition is not satisfied, then the corresponding iso-value is slightly perturbed in such a way that the new level-sets pass only through regular points. This technique is particularly useful to discard irrelevant and redundant critical points during the construction phase rather than *a-posteriori*. This approach has been formalized in [1], [5], where the extension of the Reeb graph equivalence to strips of triangles rather than level-sets has been introduced. More precisely, two points $\mathbf{p}, \mathbf{q} \in \mathcal{M}$ are considered equivalent if $f(\mathbf{p})$ and $f(\mathbf{q})$ belong to the same interval $t := (f_i, f_{i+1}) \in \mathcal{I}$ and $\mathbf{p}$, $\mathbf{q}$ are within the same connected component $\mathcal{R}$ of the strip $f^{-1}(t)$. In this case, all the points of $\mathcal{R}$ are equivalent in an extended sense and they identify the same equivalence class. Therefore, the nodes of the induced *Extended Reeb graph* are representative of iso-contours or regions. A node may be adjacent to another node, which identifies a surface strip, or to a set of nodes representing the boundaries of a connected component of a strip. The sampling-based approaches are intrinsically multi-resolution oriented as surface strips might include several types of critical points; e.g. saddle points together with maxima and/or minima.

Each strip $f^{-1}((f_i, f_{i+1}))$ that contains a handle is updated by iteratively adding a new iso-value α_i in the interval (f_i, f_{i+1}) and the iteration stops when the strip has 0-genus. We briefly remind that a bordered patch has 0-genus if gluing a topological disk to each boundary component we get a closed 0-genus surface. In this way, we are guaranteed that each topological handle of $\mathcal{M}$ is associated with a loop of $\mathcal{R}_G$ (i.e., *topological consistency* between $\mathcal{M}$ and $\mathcal{R}_G$). The iterative approach is time-consuming if the image of f does not smoothly vary and/or $\mathcal{M}$ has tiny handles. The choice of the values f_i and α_i is arbitrary and does not resemble the critical points distribution. Infact, the iso-value α_i is usually set equal to $(f_i + f_{i+1})/2$ and Image(f) is uniformly sampled. Finally, a multi-resolution construction of the Reeb graph through a dicotomic procedure is described in [22].

Sweeping techniques [9], [11], [12], [24], [32], [33] compute the Reeb graph by sweeping the iso-value α from the minimum $f_{\min}$ to the maximum value $f_{\max}$ of f and studying the evolution of the corresponding level-set $f^{-1}(\alpha)$. In this way, we determine when saddle points are encountered and process them. More precisely, in [9] the sweeping algorithm initially sorts the n vertices of the input triangulation by their function values. Then,



(a) f_1 $m = 1$, $M = 1$, $s = 2$ (b) (c) (d)

Fig. 1. Overview of the main steps of the proposed approach. Shape segmentation and adjacency graph of the torus achieved by cutting the input surface along the iso-contour related to the (a-b) first and (c) second saddle point of the first non-trivial Laplacian eigenfunction f_1; (d) Reeb graph of f_1.

the *join tree* is built by performing a sweep of the vertices from the smallest to the largest function value. In an analogous way, the *split tree* is achieved by sweeping the f-values from $f_{\max}$ to $f_{\min}$. Once the join and split trees have been computed in $O(n \log n)$-time, the *contour-tree* is obtained by merging the join and split tree. This last step requires linear time in the number of vertices.

In this class of techniques, the method described in [24] deals with closed surfaces equipped with the geodesic distance from a source point and [32] introduces a multi-resolution data-structure for computing and representing the Reeb graph. Specialized techniques for the computation of the contour-tree in any dimension have been proposed in [9] and [11]. Finally, [31] discusses a parallel computation of the augmented contour-tree, which stores information about the genus of the level-sets.

Recently, [33] has proposed an on-line algorithm that constructs the Reeb graph while reading the simplicial elements of the input surface or tetrahedralization. At each step, the insertion of a new element in $\mathcal{M}$ updates the current approximation of the Reeb graph following the creation/deletion of connected components of $\mathcal{M}$ and loops of the graph. The method discussed in [11] builds on the sweeping approach but does not require a global sorting of the f-values. More precisely, it identifies the critical points of f, increasingly or decreasingly sorts them by their function values, and builds the join and split tree. The join tree is computed by following monotone descending paths, which connect the critical points of f and are composed of an ordered sequence of points whose function values are monotonically decreasing. The split tree is computed in an analogous manner.

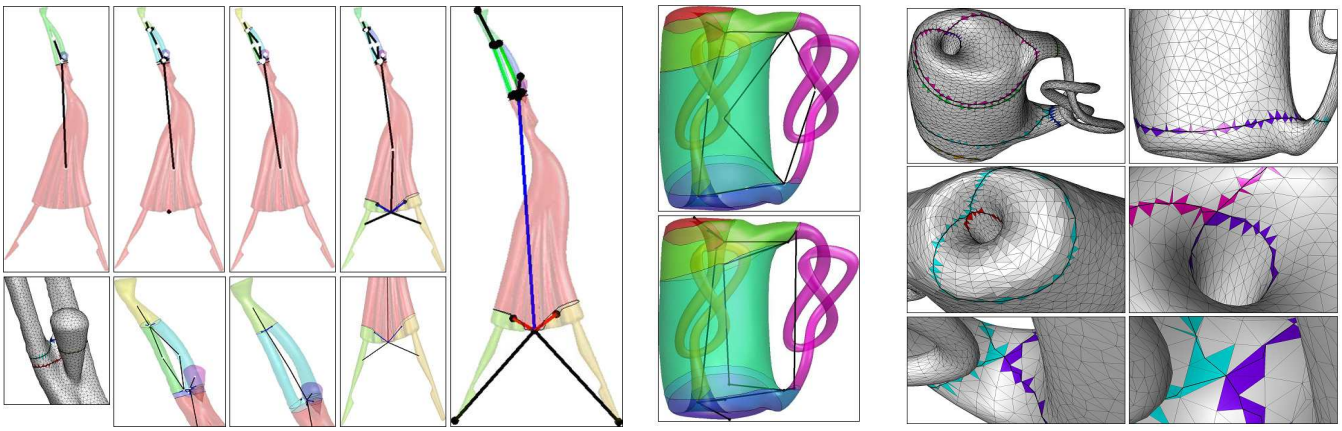
Note that [1], [9], [11], [32], [33] guarantee the topological consistency between $\mathcal{R}_G$ and $\mathcal{M}$. With the exception of [11], [33] and concerning both the sampling and sweeping approaches, the computation of the family of level-sets $\{\gamma_{f_i} := f^{-1}(f_i)\}_{i=1}^{k}$ requires to sort the f-values and initialize the iso-contour computation. The contour initialization is achieved by searching one edge $[\mathbf{p}, \mathbf{q}]$ of $\mathcal{M}$ that is intersected by γ_{f_i}, i.e. $f(\mathbf{p}) < f_i < f(\mathbf{q})$, $i = 1, \ldots, k$. Therefore, this pre-process takes $O(n \log n)$-time regardless the number of critical points.

1.2 Overview and contribution

The output of sweeping algorithms explicitly keeps track of the relations between the vertices of the input triangulation of $\mathcal{M}$ and the nodes of the Reeb graph $\mathcal{R}_G$. Infact, each regular point $\mathbf{p}_i$ of $(\mathcal{M}, f)$ is associated to a connected component γ of the level-set $f^{-1}(f(\mathbf{p}_i))$ and γ is contracted to a node of $\mathcal{R}_G$. On the one hand, the explicit link of each vertex of $\mathcal{M}$ to the corresponding arc or node of the Reeb graph is useful for applications such as skeleton-based animation [24] and the computation of seed-sets for iso-surfacing scalar functions [47]. On the other hand, several applications, which include surface parameterization, texture mapping, and shape matching, do not necessarily need an information associated to the arcs of the Reeb graph. For instance, a minimal number of nodes in the Reeb graph and the topological consistency between the surface and the graph allow us to deal with a low number of segmented patches in local parameterization [34]. Furthermore, we efficiently identify and cut the topological handles in global parameterization [36].

In this context, we present an extended and revised method for the computation of the Reeb graph based on a minimal contouring algorithm [37]. The idea behind our method is to study the evolution of the level-sets only at saddle points without sampling or sweeping the image of f. Instead of building an injective correspondence between the vertices of $\mathcal{M}$ and the nodes of the Reeb graph, our algorithm stores only a minimal information on the behavior of f on $\mathcal{M}$. Infact, each node of the computed Reeb graph is associated either with a critical point of f (i.e., maximum, minimum, saddle point) or with one of the *critical loops* of the level-set passing through a saddle point. For the definition of critical loop, we refer the reader to Section 2.1. Each arc of the Reeb graph is associated with a region $\mathcal{S}$ of $\mathcal{M}$ that is delimited by the critical loops; in $\mathcal{S}$, all the iso-contours are related to regular iso-values of f and are homeomorphic. The family of these regions provides a hierarchical shape segmentation into 0-genus patches, which are classified as generalized cones, cylinders, and pants. Fig. 1 gives an overview of the main steps of our approach.

The proposed technique does not use global sorting steps of the critical values and exploits only local infor-



(a) (b) (c)

Fig. 2. Main steps of the proposed approach. (a) Shape segmentation and adjacency graph of a scalar function with one maximum, one minimum, and two saddle points. (b) Shape segmentation derived by cutting $\mathcal{M}$ along the iso-contours related to the saddle points of f. The upper figure in (b) shows the adjacency graph and the lower part the Reeb graph. (c) Zoom-in on the *critical loops*.

mation on f; i.e., the behavior of f on the 1-ring of each vertex and the triangle-triangle adjacency. Indeed, we do not handle the function values as a whole. By combining the minimal number of nodes with the use of a small memory footprint and temporary data structures, our approach takes $O(sn)$-time, where n and s is the number of vertices of $\mathcal{M}$ and saddles of f. If $s < \log n$, then the proposed algorithm outperforms previous work and its computational cost is competitive with respect to the recent on-line computation proposed in [33]. We explicitly note that the aforementioned assumption is usually satisfied when the input surface is densely sampled or we deal with smooth scalar functions, such as Laplacian eigenfunctions [13], [40], [46], harmonic maps [30] and 1-forms [21].

To discuss the stability of the saddle iso-contouring and the computation of the Reeb graph, we distinguish the topological noise of f from the geometric and topological noise of $\mathcal{M}$. As *topological noise of* f, we refer to a high number of critical points which are close to each others, have close f-values, include multiple saddles and generally do not satisfy the Euler formula. The term *topological noise of* $\mathcal{M}$ refers to small and tiny handles; finally, the discontinuous variation of the vertex position identifies the *geometric noise of* $\mathcal{M}$. In all the aforementioned cases, we assume to deal with manifold surfaces. In case of scalar functions with topological noise, we apply a persistency-based simplification [8] before computing the critical loops. An alternative simplification scheme of the critical points of f is described in [26]. In both cases, the simplification of the critical points and the smoothing of the scalar function take $O(n \log n)$-time, where n is the number of vertices of $\mathcal{M}$. Since most of the computation uses only the mesh connectivity, with the f-values at the vertices, the proposed approach is stable with respect to topological and geometric noise of $\mathcal{M}$. The vertex position is exploited only to visualize the level-sets at saddle points and the embedding of the Reeb graph in $\mathbb{R}^3$. Finally, using only a local information on f allows us to easily extract the Reeb graph of time-depending functions.

We point out that the method discussed in [33] is best suited for the construction of Reeb graphs with respect to functions whose values have been pre-computed and streamed with the surface geometry. The authors state that the values could also be computed on the fly, but this choice does not hold for functions that require the complete shape. Among them, we cite the integral geodesic distance from source points, Laplacian eigenfunctions, harmonic maps and 1-forms. Also, the great value of Reeb graphs is their parametric nature with respect to the different choices of f, and the possibility of changing f to yield different shape descriptions. Therefore, our method is still relevant as it offers a nearly-optimal computation also in non-streaming cases. An additional motivation for using a minimal contouring algorithm instead of a sweeping or sampling strategy is that the image of f is usually oversampled with respect to the number of iso-values strictly necessary to compute the Reeb graph. A redundant number of level-sets induces a higher computational cost and an overloaded graph structure that codes sets of homeomorphic contours.

The main differences of [11] with respect to our approach are that we use only the saddle points and do not sort the corresponding function values. Furthermore, we compute the links among saddle points by exploiting the adjacency information among regions of $\mathcal{M}$. On the contrary, using monotone descending paths might be time-consuming and slowly convergent [11]. Since [11] deals with 3D scalar functions, it is more general than our technique. However, we mention that our approach can be extended to 3D scalar functions by studying the adjacency relations, the topology, and the number of

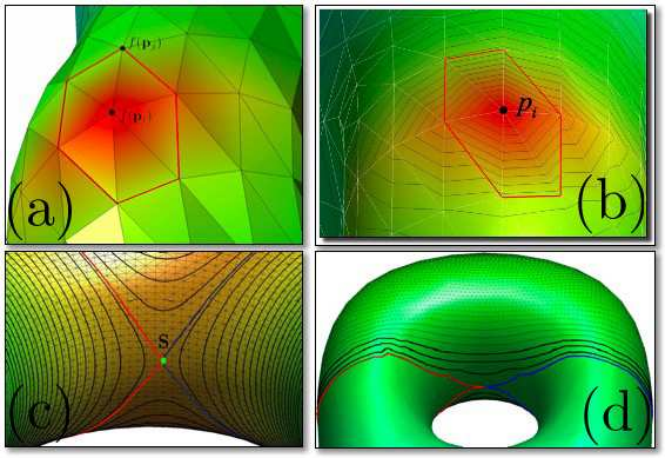


Fig. 3. In (a), 1-star of the vertex $\mathbf{p}_i$, (b) iso-contours close to a local maximum, and (c,d) zoom-in on the two loops (blue and red curve) related to a saddle point (green point) of multiplicity one.

shells of the volumes in-between the iso-surfaces of two consecutive function values at critical points.

The paper is organized as follows. In Section 2, we describe the construction of the Reeb graph and in Section 3 we discuss the main novelties and applications of the proposed approach. Finally, Section 4 outlines possible extensions and concludes the paper.

2 Building the Reeb Graph

The minimal information that we use for the construction of the Reeb graph is the set of critical points of f and iso-contours traced at saddle points, thus showing that they provide enough information to compute the Reeb graph of $(\mathcal{M}, f)$. The proposed approach is sketched by the following steps:

- extraction and classification of the critical points of f as maxima, minima, and saddles (see Section 2.1);
- computation of the iso-contours at saddle points; for each saddle $\mathbf{s} \in \mathcal{M}$, we cut $\mathcal{M}$ along the loops of $f^{-1}(f(\mathbf{s}))$ that intersect at $\mathbf{s}$ (see Section 2.2). Therefore, $\mathcal{M}$ is decomposed into a set of regions that we will call *shells*;
- coding of the adjacency relations among the shells of $\mathcal{M}$ into an *adjacency graph* $\mathcal{A}$ (see Section 2.3);
- conversion of $\mathcal{A}$ in the Reeb graph $\mathcal{R}_G$ of $(\mathcal{M}, f)$ by joining the maxima and minima of f with the nodes of $\mathcal{R}_G$ associated to the shells they belong to (see Section 2.4). Finally, in Section 2.5, we discuss the topological consistency of the computed Reeb graph.

In the following, we detail the aforementioned steps; Fig. 2 depicts the complete framework.

2.1 Critical points classification

First of all, we assume that f is general (i.e., $f(\mathbf{p}_i) \neq f(\mathbf{p}_j)$, $i \neq j$). This assumption guarantees that

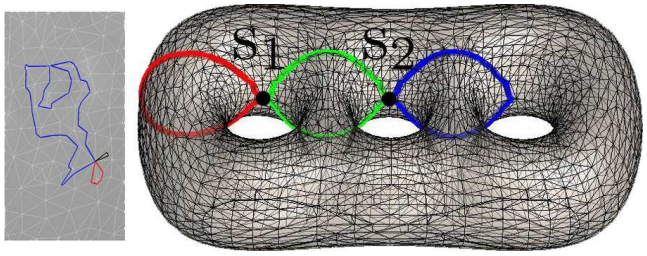


Fig. 4. (*Left*) Critical loops related to a saddle points of multiplicity two; each critical loop is shown with a different color. (*Right*) Critical loops related to two saddle points $\mathbf{s}_1$ and $\mathbf{s}_2$ of multiplicity two and with the same function value.

the level-sets of f related to regular iso-values are not degenerate and the Euler formula applies (c.f., Equation (1)). The critical points of a scalar function defined on a triangle mesh can be located and classified by analyzing for each vertex $\mathbf{p}_i$ the distribution of the f-values on the neighborhoods of $\mathbf{p}_i$ [2]. More precisely, let $N(i) := \{j : (i,j) \text{ edge}\}$ be the 1-*star* of i, i.e. the set of vertices incident to i. The *link* $Lk(i)$ of i is achieved by reordering the indices of $N(i)$ in an anti-clockwise manner; assuming for simplicity that the indices of $Lk(i)$ are $\{1, \ldots, k\}$, $Lk(i)$ is defined as

$$Lk(i) := \{j \in N(i) : (j, j+1)_{j=1}^{k-1} \text{ edge of } \mathcal{M}\}.$$

Then, the *upper link* is defined as

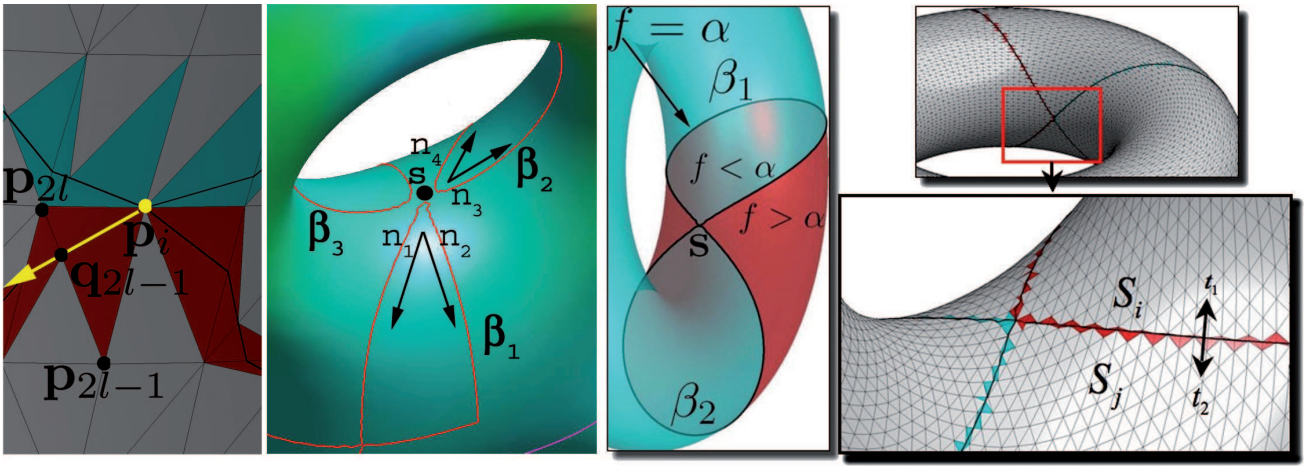
$$Lk^+(i) := \{j \in Lk(i) : f(\mathbf{p}_j) > f(\mathbf{p}_i)\},$$

and the *mixed link* as

$$Lk^\pm(i) := \{j \in Lk(i) : f(\mathbf{p}_{j+1}) > f(\mathbf{p}_i) > f(\mathbf{p}_j)\},$$

where j is intended as mod $(k+1)$. The *lower link* $Lk^-(i)$, is defined by replacing the inequality "$>$" with "$<$" in the definition of the upper link. If $Lk^+(i) = \emptyset$ or $Lk^-(i) = \emptyset$, then $\mathbf{p}_i$ is a *maximum* or a *minimum*, respectively. If the cardinality of the set $Lk^\pm(i)$ is $2 + 2m_i$, $m_i \geq 1$, then $\mathbf{p}_i$ is classified as a *saddle* of *multiplicity* m_i. Once the vertex-vertex relation has been extracted, the classification procedure requires $O(n)$-time. Fig. 3 shows the behavior of the iso-contours of f when they become close to a critical point. A vertex that does not fall in the previous classification is called *regular*.

We are now ready to clarify the relation among the critical points, the evolution of the iso-contours, and the Reeb graph $\mathcal{R}_G$ of $\mathcal{M}$ with respect to f. Assuming that $f(\mathbf{p}_i) < f(\mathbf{p}_j)$, $i < j$, the topology of the level-sets remains the same as long as t belongs to the open interval $(f(\mathbf{p}_i), f(\mathbf{p}_{i+1}))$. If $\mathbf{p}_{i+1}$ is a maximum, then the level-set $f^{-1}(t)$, $t \in [f(\mathbf{p}_{i+1}) - \epsilon, f(\mathbf{p}_i))$, $\epsilon > 0$, degenerates to $\mathbf{p}_i$. If $\mathbf{p}_{i+1}$ is a minimum, then the level set $f^{-1}(t)$, $t \in (f(\mathbf{p}_i), f(\mathbf{p}_{i+1}) + \epsilon]$, develops a set of homeomorphic contours. If $\mathbf{p}_{i+1}$ is a saddle point, then two or more contours cycles of the level-sets $f^{-1}(t)$, $t \in (f(\mathbf{p}_{i+1}) - \epsilon, f(\mathbf{p}_{i+1}) + \epsilon)$, are joined into a new cycle or an existing cycle is split into two or more cycles.



(a) (b) (c)

Fig. 5. In (a), 1-star of the saddle $\mathbf{p}_i$; (b) iso-contour $\beta := \cup_{i=1}^{3}\beta_i$ of a saddle point $\mathbf{s}$ of multiplicity two. For the critical loop β_1 and β_2, the couples of outgoing directions $(\mathbf{n}_1, \mathbf{n}_2)$ and $(\mathbf{n}_3, \mathbf{n}_4)$ are shown. (c) Evaluation of the sign of $f - \alpha$ around $f^{-1}(f(\alpha))$ and extraction of the adjacency relations among the surface patches.

The number of critical points is related to the genus of $\mathcal{M}$ by the Euler formula. If $\mathcal{M}$ has e edges, t faces, and b boundary components, then the genus g of $\mathcal{M}$ is given by $g = \frac{1}{2}(2 - \chi(\mathcal{M}) - b)$, where $\chi(\mathcal{M}) := n - e + t$ is the *Euler characteristic*. For a closed surface $\mathcal{M}$, the identity

$$\chi(\mathcal{M}) = \text{minima} - \text{saddles} + \text{maxima} \qquad (1)$$

gives the relation between the critical points of $(\mathcal{M}, f)$ and the genus of $\mathcal{M}$ [2], [27]. We explicitly note that the number $s := \sum_{\mathbf{s}_i \text{ saddle}} m_i$ of saddles considers the multiplicity m_i of each saddle.

2.2 Iso-contours of f at saddle points

This section details the saddle iso-contouring, which has already shown accurate and efficient results in different contexts such as shape segmentation for local parameterization [34] and the computation of the topological generators of arbitrary 3D shapes [36]. As added value, in this paper we will show also its capability of handling multiple saddles, non-simple scalar functions, and noisy surfaces (see Fig. 4).

Let $f : \mathcal{M} \to \mathbb{R}$ be a general function and $\mathbf{p}_i$ a saddle point of multiplicity m such that $f(\mathbf{p}_i) = \alpha$. The connected component β of $f^{-1}(\alpha)$ that contains $\mathbf{p}_i$ (see Fig. 5(a,b)) is the union of $m + 1$ closed curves $\beta_1, \ldots, \beta_{m+1}$ that intersect at $\mathbf{p}_i$, i.e. $\beta := \cup_{l=1}^{m+1}\beta_l \ni \mathbf{p}_i$. In the following, we refer to β_l as a *critical loop* related to the saddle $\mathbf{p}_i$. From the definition of saddle points of multiplicity m, it follows that the cardinality of the mixed link is equal to $2 + 2m$. For simplicity, let us assume that $Lk(i) = \{1, \ldots, 2(m + 1)\}$. For the m odd indices $l = 1, 3, \ldots, 2m - 1$ (see Fig. 5(a)), the level-set $f^{-1}(f(\mathbf{p}_i))$ intersects the edges $(\mathbf{p}_{2l-1}, \mathbf{p}_{2l})$ and $(\mathbf{p}_{2l+1}, \mathbf{p}_{2(l+1)})$ at the points

$$\mathbf{q}_{2l-1} := (1-t)\mathbf{p}_{2l-1} + t\mathbf{p}_{2l}, \quad t := \tfrac{f(\mathbf{p}_i) - f(\mathbf{p}_{2l-1})}{f(\mathbf{p}_{2l}) - f(\mathbf{p}_{2l-1})},$$
$$\mathbf{q}_{2l} := (1-t)\mathbf{p}_{2l+1} + t\mathbf{p}_{2(l+1)}, \quad t := \tfrac{f(\mathbf{p}_i) - f(\mathbf{p}_{2l+1})}{f(\mathbf{p}_{2(l+1)}) - f(\mathbf{p}_{2l+1})}.$$

Therefore, the vectors $\mathbf{n}_{2l-1} := \mathbf{q}_{2l-1} - \mathbf{p}_i$ and $\mathbf{n}_{2l} := \mathbf{q}_{2l} - \mathbf{p}_i$ give the outgoing directions that originate at $\mathbf{p}_i$ and that are used to trace β_l. These vectors are computed during the classification of the vertices of $\mathcal{M}$ as critical points of f.

Starting from $\mathbf{p}_i$ toward the direction $\mathbf{n}_{2l-1}$, we trace β_l by following the gradient field of f until we come back to $\mathbf{p}_i$ along $-\mathbf{n}_{2l}$. At the first step, we consider the edge e of the triangle $t^\star$ in the 1-star of $\mathbf{p}_i$ that is intersected by β_l along the direction $\mathbf{n}_{2l-1}$ (see Fig. 6(a,b)). Then, we search the next intersection between β_l and the two edges of the triangle adjacent to $t^\star$ along e. The iteration proceeds by using the triangle-triangle adjacencies until we draw the critical loop (see Fig. 6(c,d)). Note that we initialize the iso-contour by searching the intersected edges in the 1-star of $\mathbf{p}_i$. On the contrary, sampling-based and sweeping approaches perform this search on the whole triangle mesh because the location of the intersected edges is not known *a-priori*. Then, these methods need to increasingly (or decreasingly) sort the function values to identify the first intersected edge, which initializes the iso-contouring algorithm.

Slicing $\mathcal{M}$ along a given loop β_l requires to duplicate its points and determine those parts of the triangles that are intersected by β_l (see Fig. 5(b,c)). Since the points of β_l might have a clockwise or an anti-clockwise ordering with respect to the triangle and surface orientation, we infer the value around β_l. To this end, we evaluate the sign of $f(\mathbf{p}_k) - \alpha$, where $k \in Lk(i)$ and $\mathbf{p}_k$ is the vertex that we meet by walking from $\mathbf{p}_{2l}$ to $\mathbf{p}_{2l+1}$ along the 1-star of i. If l does not exist, then the third vertex $\mathbf{p}_j$ of the triangle with edge $\mathbf{p}_i\mathbf{p}_{2l-1}$ is external to β_l; therefore,

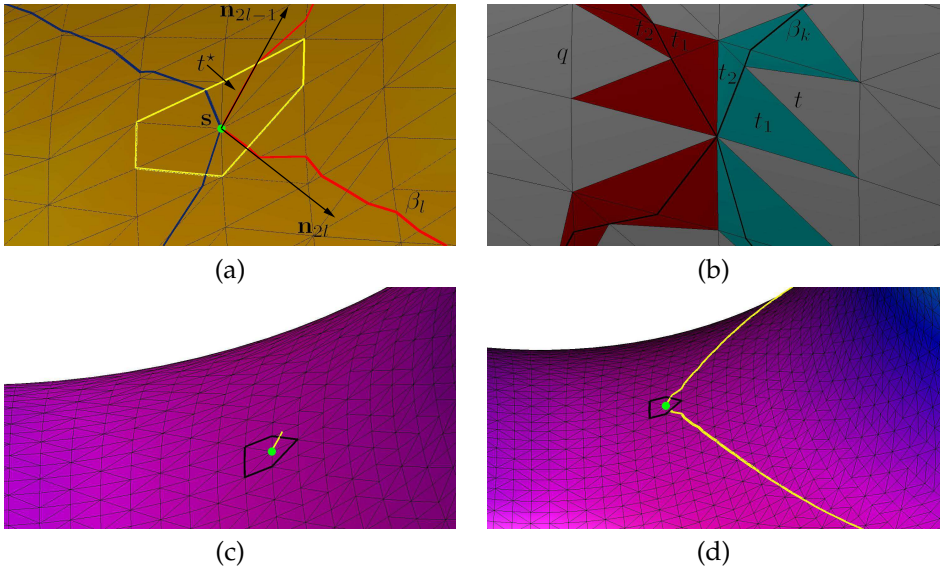


Fig. 6. In (a), 1-star of a saddle $\mathbf{s}$; (b) related critical loops. In (a), the edge e of the triangle $t^\star$ belonging to the 1-star of $\mathbf{s}$ is intersected by the loop β_l along the outgoing and incoming direction $\mathbf{n}_{2l-1}$ and $-\mathbf{n}_{2l}$. The picture shows the search of the intersection between β_l and the triangle adjacent to $t^\star$. (b) Cut of the triangle through a vertex (blue triangles) and re-triangulation of a quadrilateral face (red triangles). The triangles t_1 and t_2 are used to code the adjacency relations between the corresponding patches and the iso-contour β_l. (c,d) Computation of a critical loop.

the sign of $f-\alpha$ inside β_l is opposite to that of $f(\mathbf{p}_j)-\alpha$.

If f has s saddles and m_i is the multiplicity of the saddle $\mathbf{s}_i$, then we extract $s+\sum_{\mathbf{s}_i \text{saddle}} m_i$ critical loops in linear time without using a global sorting of the function values. Fig. 4(*left*) shows the critical loops related to a saddle point of multiplicity two. As shown in Fig. 4(*right*), the iso-contouring algorithm is also able to trace the critical loops of saddle points where f is not simple. To simplify the discussion, we omit these technical details. In the following of the paper, we assume that f is Morse; therefore, it has only simple saddles (i.e., $m_i = 1$) and we trace $2s$ critical loops.

2.3 Adjacency graph

Let s be the number of saddles of the input scalar function f and suppose that we processed k, $k < s$, saddles of $(\mathcal{M}, f)$. Then, the slice of $\mathcal{M}$ along the corresponding critical loops has generated c connected components of $\mathcal{M}$. Let $\mathbf{s} \in \mathcal{M}$ be a simple saddle point that has not been processed and consider the connected component β of $f^{-1}(f(\mathbf{s}))$ that contains the saddle $\mathbf{s}$. Since $\mathbf{s}$ is simple and f is general, β is the union of the two sub-loops β_1, β_2; i.e., $\mathbf{s} \in \beta_1 \cup \beta_2 = \beta \subseteq f^{-1}(f(\mathbf{s}))$.

For $k = 1, 2$, we slice $\mathcal{M}$ along β_k and store the adjacency relations among the triangles intersected by β_k (see Fig. 5(c)). Two situations can happen (see Fig. 6(b)): if β_k intersects a triangle t passing through one of its vertices, then t is split into two new faces t_1, t_2 that share a part of β_k. Otherwise, β_k splits t into one triangle and one quadrilateral q; then, q is re-triangulated by subdividing it along its shortest diagonal and we consider as t_2 the triangle of q that is adjacent to β_k. In both cases, we store the adjacency among β_k, t_1, and t_2. Once we have sliced $\mathcal{M}$ along β_1 by duplicating its vertices and edges, we apply the same procedure to β_2 and update the number of connected components of $\mathcal{M}$ in $O(n)$-time. Note that after this step β_1 and β_2 are two disjoint curves (see Fig. 7(c)). To simplify the notation, in the following $\mathcal{M}$ refers to the cut surface.

To count and update the number of connected components of the sliced surface, we mark each triangle as belonging to a shell of $\mathcal{M}$. To this end, starting from an unmarked triangle t we visit all the faces that are reachable from t by using the triangle-triangle adjacencies. The set of visited triangles gives the shell of $\mathcal{M}$ that contains t. The selection of a non-visited triangle (if any) initializes a new shell whose construction follows the aforementioned process. Finally, the counting of the number of connected components stops when all the triangles of $\mathcal{M}$ have been visited.

During the iterations, the connected component of $\mathcal{M}$ that contains t_2 will change on the base of the shells generated by cutting $\mathcal{M}$ along β_k, $k = 1, 2$, and the loops related to the saddle points already visited. It is worth to mention that if t_1 belongs to the shell S_i and it is paired, with respect to the same critical loop β_k of $\mathbf{s}$, to the triangle t_2 that belongs to the connected component S_j, then S_i and S_j are adjacent. This relation will be used to build the arcs of the adjacency graph that code the evolution of the level-sets related to the iso-values close to $f(\mathbf{s})$ (see Fig. 5(c)).

Let $\mathcal{A}_{i-1}$ be the adjacency graph related to the $(i-1)^{\text{th}}$-step. When we process the saddle $\mathbf{s}$, we split the surface patch it belongs to into two or more regions by slicing $\mathcal{M}$ along the critical loops related to $\mathbf{s}$. Then, the adjacency relations among these new regions and

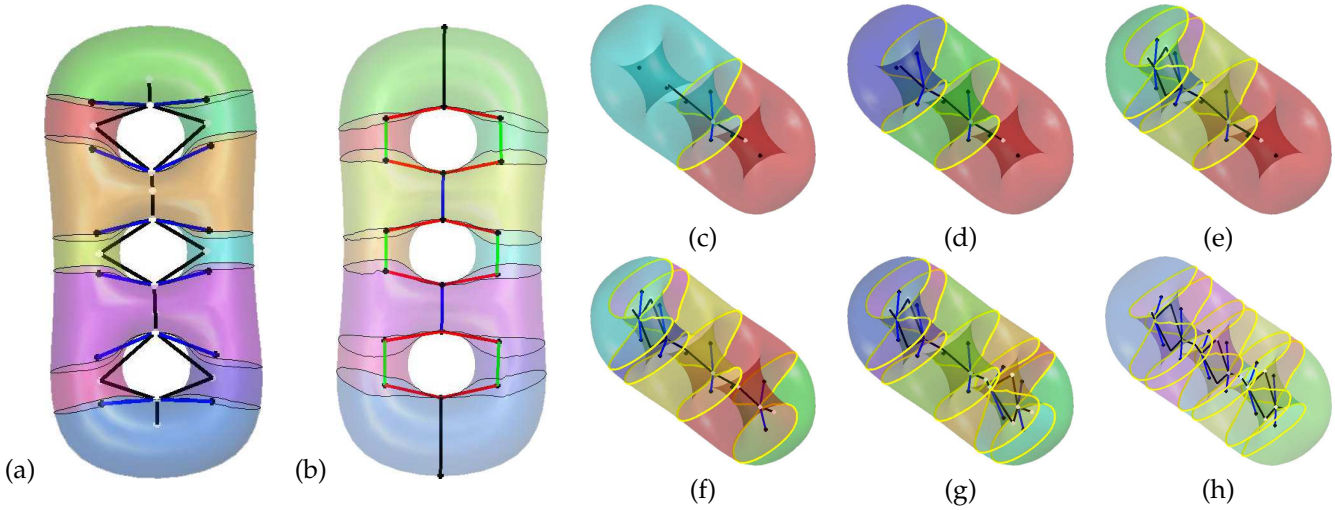


Fig. 7. (a) Adjacency and (b) Reeb graph achieved by cutting (c-h) the 3-torus along the critical loops at its saddles.

the previous ones are coded in $\mathcal{A}_i$ by using the triangle-triangle adjacency with respect to β_k. To this end, we update the arcs of $\mathcal{A}_{i-1}$ that are incident to the critical points that belong to the region $\mathcal{S}$ that includes β_1 and β_2. As shown in Fig. 7(d), we join the barycenters of β_1 and β_2 with the saddle $\mathbf{s}$ and each visited saddle in $\mathcal{S}$ is connected to the barycenter of the adjacent shells. Clearly, a new shell creates a set of arcs in the adjacency graph; a loop is generated when all the saddle points which belong to a topological handle have been processed (see Fig. 7(e)).

The iterations proceed until all the saddles of f have been visited (see Fig. 7(c-h)). Since the approach processes one saddle point at each step, it provides a hierarchical family $\{\mathcal{A}_i\}_{i=1}^{s}$ of adjacency graphs whose construction is guided by the saddles location together with the related iso-contours. Note that each node of the adjacency graph corresponds to a saddle, a connected component of the sliced surface, or a critical loop. To visualize the adjacency graph of $(\mathcal{M}, f)$, we create an embedding that provides a skeleton representation of $\mathcal{M}$. To this end, each node of $\mathcal{A}_i$, $i = 1, \ldots, s$, corresponds to a saddle point, the barycenter of a critical loop, or the barycenter of a surface patch generated by slicing $\mathcal{M}$ along a critical loop (see Fig. 7(a)). Then, the Reeb graph is computed from the adjacency graph as described in Section 2.4 (see Fig. 7(b)). Other tests are shown in Fig. 8 and Fig. 9. Note that the examples in Fig. 2 and Fig. 9 confirm the capability of handling inner topological handles without ambiguities in the extraction of the adjacency information among the surface patches.

We now analyze in detail the computational cost of the steps necessary to compute the adjacency graph. The computation of the link of each vertex is linear in the number of vertices and it is independent of the input scalar function. Since the averaged cardinality of the link is six, we compute and store the link of each vertex using $O(n)$-time and memory allocation. The computation of the iso-contour at a saddle pont $\mathbf{s} \in \mathcal{M}$ takes linear time in the number of intersected edges and $O(n)$-time for the worst case complexity. In this case, we initialize the iso-contouring algorithm by searching an edge of $\mathcal{M}$ intersected by $\gamma := f^{-1}(f(\mathbf{s}))$ in the link of $\mathbf{s}$. Finally, the counting and update of the number of connected components take $O(st)$-time, where t is the number of triangles. For time-varying scalar functions $f_t : \mathcal{M} \to \mathbb{R}$, we need only to classify the critical points of f_t corresponding to a new time step t and without re-computing the link structure, which is computed once and then stored.

2.4 Reeb graph construction and hierarchal segmentation

Once the adjacency graph $\mathcal{A} := \mathcal{A}_s$ of $(\mathcal{M}, f)$ has been extracted, we modify the three types of arcs of $\mathcal{A}$, i.e. *saddle-saddle*, *saddle-patch*, and *saddle-loop*, and construct the Reeb graph using the following rules.

- If both the two critical loops of two saddle points $\mathbf{s}_i$ and $\mathbf{s}_j$ are the boundary components of the same shell, then $\mathbf{s}_i$ and $\mathbf{s}_j$ are connected by an arc (see Fig. 7(a), yellow and violet regions of the 3-torus).
- If the two boundary components of a shell are the critical loops of two saddle points, then the barycenters of the boundaries are joined together and each one of them is connected to the corresponding saddle (see Fig. 7(a), tubular regions in the bottom, middle, and up part of the 3-torus).
- If the two critical loops of a saddle $\mathbf{s}_i$ and one critical loop β of a saddle $\mathbf{s}_j$ belong to the same shell, then $\mathbf{s}_i$ is connected to the barycenter of β by an arc (see Fig. 10).

Finally, we need to code in $\mathcal{A}$ the minima and maxima of f that have not been considered by the previous process. To this end, if $\mathbf{p}_i$ is an extremum of f, regardless to its classification as maximum or minimum, the Reeb graph of $(\mathcal{M}, f)$ is achieved by adding one arc from $\mathbf{p}_i$ to the saddle which belongs to the same shell of $\mathbf{p}_i$.

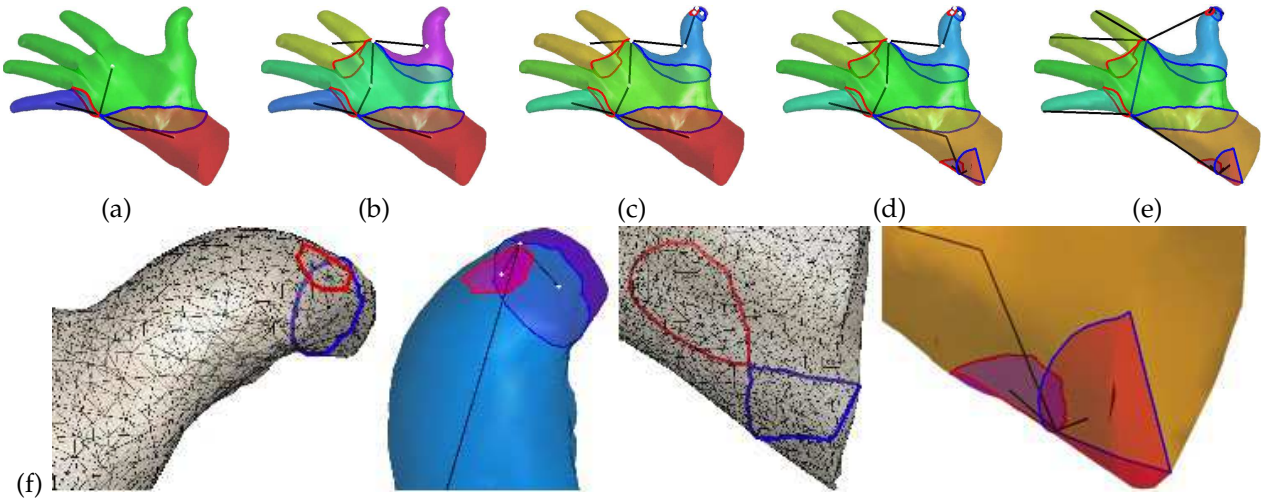


Fig. 8. (a-d) Shape segmentation at each step of the construction of the adjacency graph of a 0-genus surface. (e) Reeb graph and (f) zoom-in related to the steps (a-d).

The barycenters of the critical loops are used only for embedding the Reeb graph of $(\mathcal{M}, f)$ in the 3D space; however, we can select any other representative point.

At each iteration k, $k \leq s$, we partition the input surface into a family of r_k patches $\mathcal{S}^k := \{S_i^k\}_{i=1}^{r_k}$ such that:

- $\bigcup_{i=1}^{r_k} S_i^k = \mathcal{M}$;
- S_i^k is a connected region, $i = 1, \ldots, r_k$;
- $int(S_i^k) \cap int(S_j^k) = \emptyset, i \neq j$, with $int(X)$ internal part of X;
- $S_i^k \neq \emptyset$ has l_i boundary components $\{\gamma_j\}_{j=1}^{l_i}$.

Note that at the iteration k the number of boundary components of each patch is arbitrary. Therefore, we decompose $\mathcal{M}$ into only three types of primitives, i.e. generalized cones, cylinders, and pants. More precisely, a *generalized cone* is defined as a patch with one boundary component; in $\mathcal{R}_G$, it is associated to a terminal arc which corresponds to a maximum or a minimum. A *generalized cylinder* is identified by a region with two boundary components; in $\mathcal{R}_G$, each of these boundaries corresponds to a critical loop of a saddle. The *generalized pants* have three or four boundary components; since we assumed that f is general, each patch cannot have more than four boundaries. In Fig. 7(a,b), the yellow and violet patches S are two generalized pants with four boundary components. Infact, the two critical loops of each one of the two saddles are disjoint boundary components of S.

Special attention has to be payed to handling the saddle points at the boundaries of the patches. If we cut the surface along the critical loops at a saddle points and keep exactly these cuts as boundaries, then the patches S_i will be non-manifold at the saddle points. Their 1-stars, indeed, belong to two different patches by definition. While the 1-star is necessary to keep track of the correct adjacency among patches, the homotopy equivalence between the boundaries of patches has to be demonstrated ideally removing the 1-star: this operation topologically corresponds to the gluing of the cells corresponding to the patches (see Section 2.5). Note that removing the 1-star of the two saddle points of a generalized pant converts this region to a generalized cylinder, i.e. a patch with two boundary components instead of four boundaries. Finally, the boundary components of each surface primitive correspond to the critical loops of the saddle points of f. Fig. 8, Fig. 9, and Fig. 11 show the hierarchy of adjacency graphs and shape segmentations on surfaces with different topological and geometric complexity.

2.5 Topological consistency of the computed Reeb graph

To prove the consistency of the extracted Reeb graph with the topology of $\mathcal{M}$, we show that the level-sets at saddle points are correctly computed, all the critical points of f are coded in the graph, and the regular level-sets in the interior part of each patch are homeomorphic. The theoretical result which is at the basis of our demonstration is the following [4], [18]: *the interval $[a, b]$, $a < b$, contains no critical values of f if and only if the level-sets $f^{-1}(\alpha_1)$ and $f^{-1}(\alpha_2)$ are homeomorphic for all $\alpha_1 \neq \alpha_2 \in [a, b]$*. With this premise, the assumption that f is general guarantees that starting from the saddle $\mathbf{p}_i$ we always come back to $\mathbf{p}_i$ along the critical loop β_l. Infact, if β_l has a self-intersection at a point $\mathbf{q}$, $\mathbf{q} \neq \mathbf{p}_i$, then $\mathbf{q}$ is necessarily a critical point of f. Therefore, $f(\mathbf{p}_i) \equiv f(\mathbf{q})$ and f is not general; we conclude that the saddle isocontouring exactly works.

From the construction of the partition of $\mathcal{M}$, each patch S_i has saddle points only on the boundary and may contain maxima or minima in its interior. Therefore, on the basis of the aforementioned property, S_i does not contain saddles and the regular level-sets $\{f^{-1}(\alpha)\}_{\alpha \in \mathcal{S}}$, $\mathcal{S} := \{f(\mathbf{p}), \mathbf{p} \in int(S_i)\}$, are homeomorphic. As mentioned in the previous section, we note that considering only the interior of the patch allows us to actually disconnect th 1-star of the saddles at the boundaries of

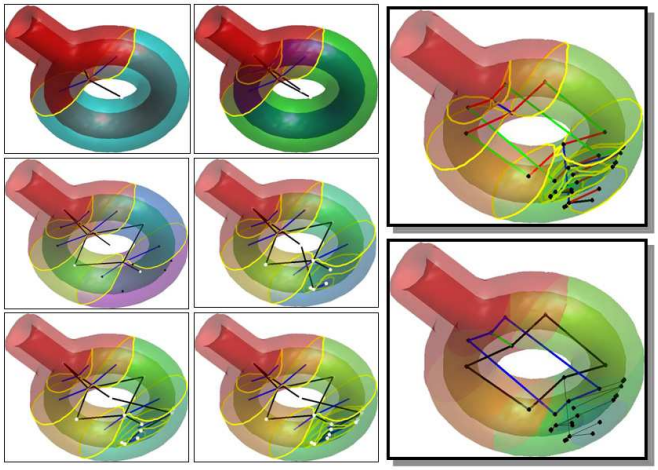


Fig. 9. (*Left*) Shape segmentation at each step of the construction of the adjacency graph of a 2-genus surface. (*Right*) The up and bottom picture show the adjacency and Reeb graph of the input surface, respectively.

the patch. From the previous property, it also follows that it is sufficient to code only the maxima and minima of f and the level-sets at saddle points in the Reeb graph. Since we have sliced the surface along the level-sets at each saddle point and each extremum of f is linked in the graph to the saddle of the patch it belongs to, we conclude that all the critical points of f have been coded in the Reeb graph.

3 Discussion

This section discusses the degrees of freedom of the proposed approach (see Section 3.1), the computational cost, and the stability to surface noise and sampling (see Section 3.2). Then, we identify a class of scalar functions whose Reeb graphs are commonly used to address shape segmentation and abstraction (see Section 3.3). Finally (see Section 3.4), we show how the computation of the Reeb graph can be easily adapted to time-varying scalar functions.

3.1 Degrees of freedom for the Reeb graph computation

The proposed method does not assume that the critical points of f are ordered in a specific manner; on the contrary, previous work usually requires a global sorting step of the function values at the mesh vertices and/or at the critical points. If we are interested in the hierarchy of the segmented surfaces, then several reordering criteria of the critical values are possible. A simple choice is to increasingly (or decreasingly) sort the function values only at saddle points; this step requires $O(s \log s)$-time and provides a hierarchy which follows the variation of f on $\mathcal{M}$.

Noise scalar functions are generally characterized by a large number of extremum-saddle pairs and an irregular variation of f-values on the input surface. In

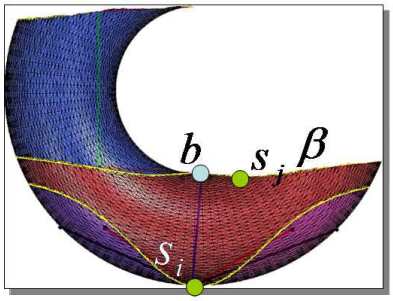


Fig. 10. The two critical loops of the saddle $\mathbf{s}_i$ and one critical loop β of a saddle $\mathbf{s}_j$ belong to the same shell; then, $\mathbf{s}_i$ is connected to the barycenter $\mathbf{b}$ of β by an arc.

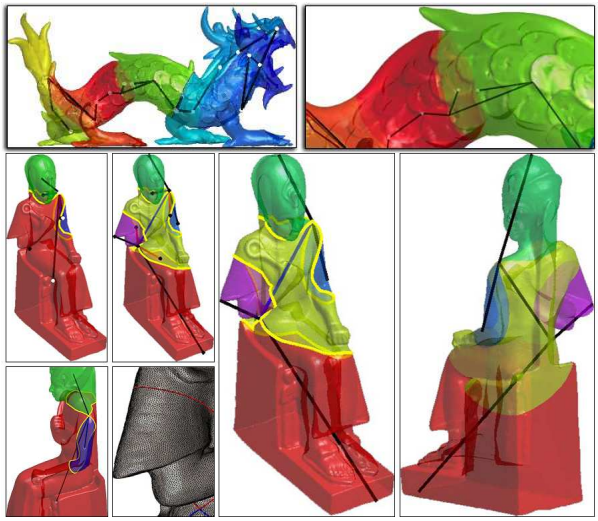


Fig. 11. *First row*: Reeb graph on a 0-genus surface. *Second row*: Adjacency graph (left side) on the Ramses model and Reeb graph (right side).

this case, the choice of the iso-values of sweeping and sampling techniques do not reflect the variation of the f-values on $\mathcal{M}$ and might result in a large number of regular nodes of the Reeb graph. Therefore, regular nodes are removed *a-posteriori* and redundant extremum-saddle pairs $(\mathbf{e}, \mathbf{s})$ are simplified on the base of their persistency [15] $\epsilon := |f(\mathbf{e}) - f(\mathbf{s})|$, defined as the maximum difference in their function values. Multi-resolution simplification techniques are also used to incrementally reduce the complexity of these high-level structures, in terms of number of nodes, arcs, and loops, thus highlighting their global behavior. In [22], a multi-resolution representation of the Reeb graph is computed by hierarchically sampling the image of f and concurrently updating the Reeb graph. In [32], a multi-resolution contour-tree is built using the join and split trees [9] and collapsing proper pairs of critical points of f. This simplification defines a hierarchy of topological cancellations, which do not disconnect the graph and are guided by several metrics such as the persistency, the geometric location and properties of the critical points (e.g., curvature values, distribution as clusters). Thereby, the

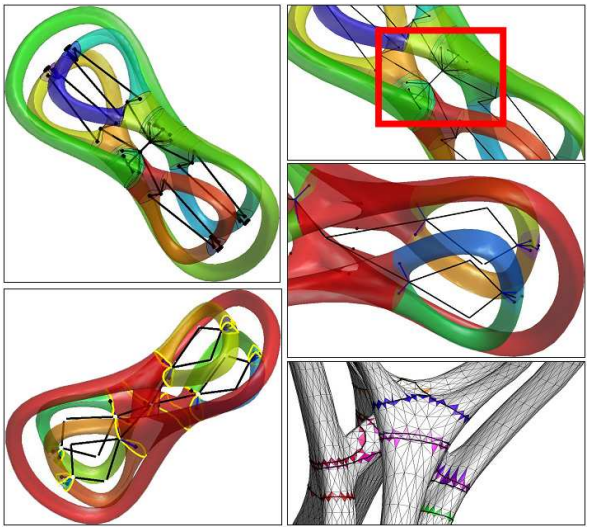


Fig. 12. Shape segmentation and Reeb graph of a 5-genus surface with respect to a harmonic function. One of the five loops of the Reeb graph is represented by the double arc in the red square.

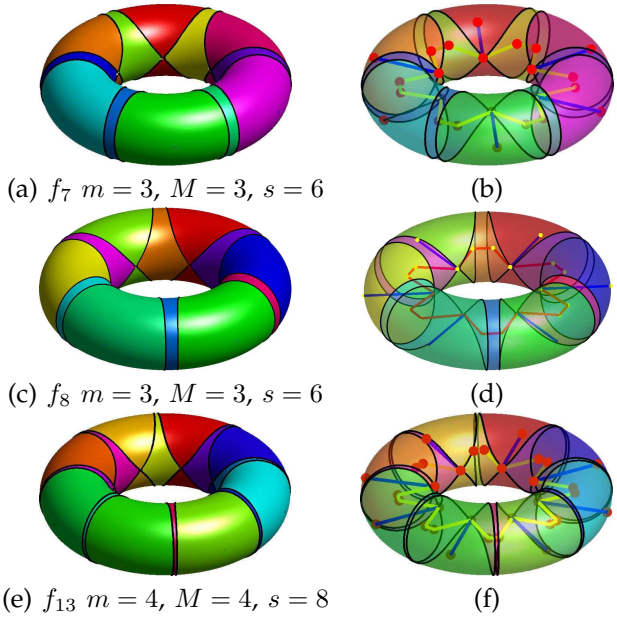


Fig. 13. (a, c, e) Shape segmentation and (b, d, f) Reeb graph of the torus induced by three Laplacian eigenfunctions. The value s, m, and M are the number of minima, maxima, and saddle points, respectively. The black lines show the iso-contours related to the saddle points.

level of persistency represents the scale of the topological features that are preserved and/or simplified.

Since we consider only critical iso-values, we do not code the level-sets as regular nodes of the Reeb graph $\mathcal{R}_G$. To avoid creating large graphs, which have redundant arcs/loops and are induced by clustered critical points, we prefer to simplify the un-significant extremum-saddle points during the classification of the critical points and before computing the adjacency graph. Therefore, the Reeb graph includes only the preserved critical points without tracing the iso-contours related to the removed saddle points. Note that removing an extremum $\mathbf{p}_i$ of f and the associated saddle $\mathbf{s}$ corresponds to cancelling the arc of the Reeb graph that joins $\mathbf{p}_i$ to $\mathbf{s}$. From the point of view of the surface segmentation, this is equivalent to merge the surface patch that includes both $\mathbf{p}_i$ and $\mathbf{s}$ with the patch whose boundary is $\gamma_{\mathbf{s}} \subseteq f^{-1}(f(\mathbf{s}))$, $\mathbf{s} \in \gamma_{\mathbf{s}}$. Indeed, we glue the surface patches that share $\gamma_{\mathbf{s}}$ and the relations stored in the adjacency graph are updated and concurrently propagated to the nodes of $\mathcal{R}_G$ that are incident to $\mathbf{s}$.

3.2 Computational cost and stability issues

As discussed in Section 2.3, slicing $\mathcal{M}$ along β_k, $k = 1, 2$ takes linear time in the number of triangles intersected by β_k. Counting the number of shells takes $O(t)$-time, where t is the number of triangles of $\mathcal{M}$; indeed, the overall cost for processing the saddle $\mathbf{s}$ is $O(t)$. We conclude that the classification of the critical points of f, the *coarse-to-fine structure* of surface patches, the corresponding adjacency graph, and indeed the Reeb graph, are built in $O(sn)$-time. Table 1 summarizes the computational cost of the main steps of the proposed approach.

Each critical loop $\gamma := f^{-1}(f(\mathbf{p}_i))$ of a saddle vertex with index i can be abstracted as the ordered list $\mathcal{E} := \{(e_k, t_k)\}_k$, where $e_k := (j_k, j_{k+1})$ is an edge of the input triangulation intersected by γ (i.e., $f(\mathbf{p}_{j_k}) < f(\mathbf{p}_i) < f(\mathbf{p}_{j_{k+1}})$ or $f(\mathbf{p}_{j_{k+1}}) < f(\mathbf{p}_i) < f(\mathbf{p}_{j_k})$) and $t_k \in [0, 1]$, is the parameter that identifies the intersection point $t_k\mathbf{p}_{j_k} + (1-t_k)\mathbf{p}_{j_{k+1}}$ between e_k and γ. Since the computation of $\mathcal{E}$, the slicing of the input surface, and the counting of the connected components uses only the mesh connectivity, equipped with the function values at the mesh vertices, the overall scheme is independent of the geometry of $\mathcal{M}$.

Finally, the position of the vertices is used only to compute and visualize the iso-contours of f from $\mathcal{M}$ and the embedding of the Reeb graph. Fig. 14 and Fig. 15 show the stability of the saddle iso-contouring and the computation of the Reeb graph with respect to a different noise of the surface shape. The decreasing behavior of the graph in Fig. 15 confirms that the number of triangles intersected by the iso-contours at saddle points is much lower than the number of input vertices, which is the upper bound for the worst case complexity.

3.3 Hierarchical shape segmentation and choice of f

Even though the proposed computation of the Reeb graph only assumes that the input scalar function is general, specific choices require a low computational cost for the construction of the Reeb graph. Additionally, specific scalar functions provide better segmentations of $\mathcal{M}$ and can be used to target specific applications such

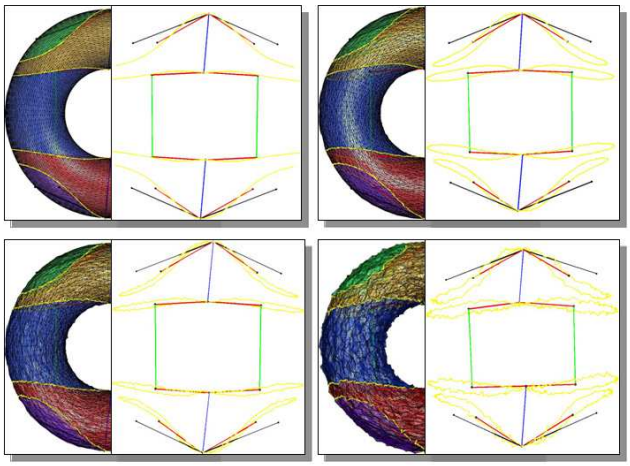


Fig. 14. Stability of the saddle iso-contouring and computation of the Reeb graph with respect to an increasing geometric noise.

as quadrilateral remeshing [13], visualization [32], and shape comparison [22]. Infact, each f is associated with

- a different set of saddle points. Then, each saddle is characterized by a different location on $\mathcal{M}$ and shape of the related iso-contours, which become the boundary components of the segmentation patches;
- a different number s of saddle points, $s \geq 2g$, which affects the number of patches of the segmentation and the overall computational cost, i.e. $O(sn)$.

From the perspective of the segmentation, it follows that the best choice of f is a function that takes into account the shape of $\mathcal{M}$ (e.g., symmetries, protrusions) and has a low number of critical points. In the following, we briefly motivate the use of the harmonic functions and Laplacian eigenfunctions as f.

The *harmonic function* f solves the Laplace equations with Dirichlet boundary conditions and it is computed by solving a sparse linear system [14], [17]. The *maximum principle* states that f has no local extrema other than at constrained vertices. In the case that all constrained minima are assigned the same global minimum value and all constrained maxima are assigned the same global maximum value, all the constraints will be guaranteed to be extrema in the resulting field. If the closed surface $\mathcal{M}$ has genus g and we select m minima and M maxima as boundary conditions, then from the Euler formula (1) we get $s = m + M + 2g - 2$ saddles and the corresponding Reeb graph has $2(m + M + g - 1)$ critical points.

The maximum principle provides the main motivation to use harmonic functions for shape segmentation and the Reeb graph construction. Infact, it allows us to build harmonic functions with a minimal (i.e., one maximum, one minimum, and $2g$ saddles) or pre-defined number of critical points once we have fixed the Dirichlet boundary conditions. An example is shown in Fig. 12. If f is a harmonic function with only one maximum and one minimum, then f has $2g$ saddle points. In this case, we have one patch for each extremum, two cylinder-like patches associated to each topological handle of $\mathcal{M}$ and $(g - 1)$ regions which join them. Therefore, the segmentation provided by such a function has $3g + 1$ patches, which provide a decomposition of $\mathcal{M}$ in a *minimal number* of 0-genus regions. Also, the harmonic 1-forms [21] provide a good set of functions for computing the Reeb graph with few nodes and a shape segmentation with a low number of patches. If there is not a predefined choice of the Dirichlet boundary conditions, then the Laplacian eigenfunctions [40], [46] provide an alternative to harmonic functions and they still guarantee a low number of critical points and a smooth behavior on $\mathcal{M}$.

TABLE 1
Computational cost of the steps of the proposed framework, where n, s and r_k is the number of vertices, saddles, and shells of the sliced surface at the iteration k.

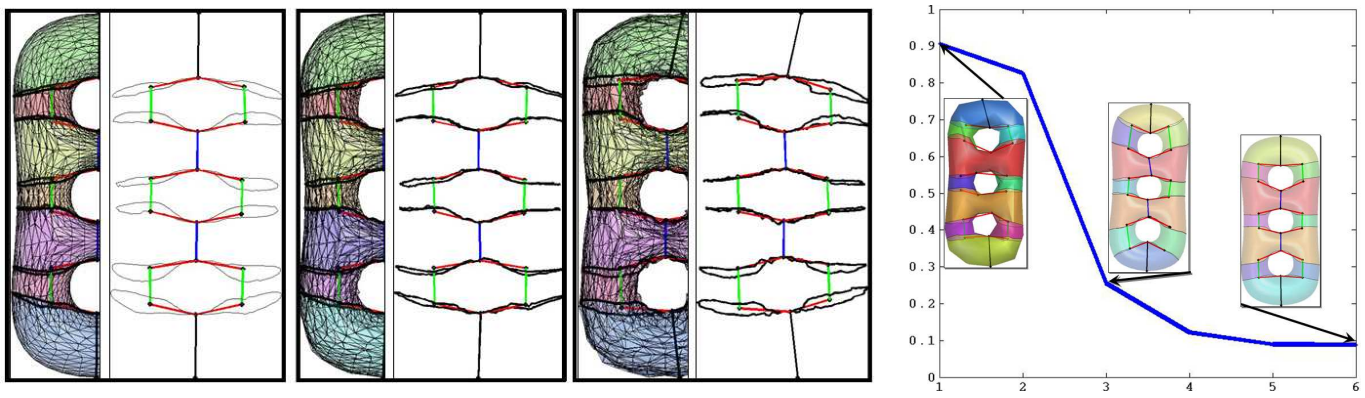
Task	At each iterat. k	Overall
Critical point classif.	$O(n)$	–
Saddle iso-contouring	$O(n)$	$O(sn)$
Adjacency graph	$O(r_k)$	$O(sn)$
Reeb graph	–	$O(sn)$

3.4 Extension to time-depending functions

Our approach easily handles time-depending functions $f_t : \mathcal{M} \rightarrow \mathbb{R}$; infact, the 1-star of each vertex is computed once and used to classify the critical points of f_t in linear time, for each time value t. Then, the computation of the iso-contours and the extraction of the Reeb graph of $(\mathcal{M}, f_t)$ follows the procedure previously discussed. An example is shown in Fig. 16.

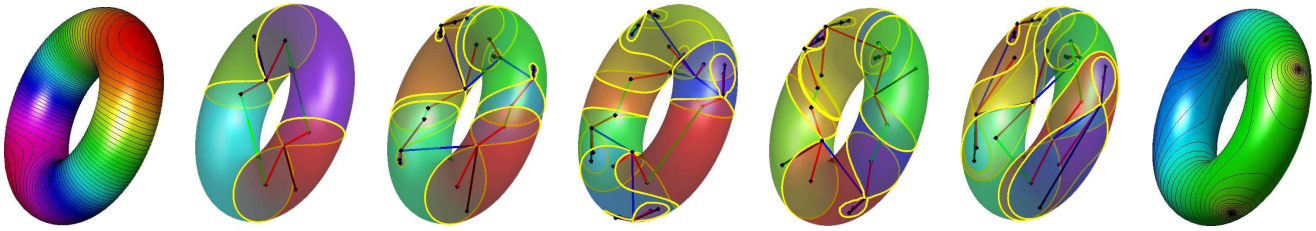
4 Future Work

The common approach for computing the Reeb graph of a scalar function $f : \mathcal{M} \rightarrow \mathbb{R}$, defined on a surface $\mathcal{M}$, is to trace the iso-contour of each regular vertex of f. This choice makes the computational cost of sweeping techniques proportional to the number of input vertices. By working directly with the critical points, we proposed an algorithm whose computational cost $O(sn)$ depends only on the complexity of f, in terms of the number s of saddle points and vertices n of $\mathcal{M}$. In all those cases where $s < \log n$ and it is not necessary to provide a complete coding of all the surface vertices in the Reeb graph, the computational cost of the proposed algorithm is lower than the cost of the state-of-the-art techniques. Finally, the assumption that s is lower than $\log n$ is commonly fulfilled (e.g., f is a harmonic function or a Laplacian eigenfunction) and can be induced by simplifying clustered critical points or highly noisy scalar functions [8]. As future work, we plan to extend the proposed approach to computing the Reeb graph of harmonic 1-forms and 3D scalar functions. We can target this last aim by studying the adjacency relations and the



(a) (b)

Fig. 15. In (a), from left to right: stability of the saddle iso-contouring and computation of the Reeb graph with respect to an increasing geometric noise. (b) Choosing six different samplings of the 3-torus surface, the plot shows the ratio between the number of intersected triangles and input vertices (y-axis).



f_1 f_1: $(1,1,2)$ f_2: $(4,4,8)$ f_3: $(4,3,7)$ f_4: $(3,3,6)$ f_5: $(3,3,6)$ f_5

Fig. 16. Iso-contours and Reeb graphs of a time-depending scalar function at different time steps. Each row also shows the number (m, M, s) of minima, maxima, and saddle points.

topology of the volumes in-between the iso-surfaces of two consecutive critical function values.

Acknowledgments. This work has been supported by the FOCUS K3D C.A., the FP6 IST AIM@SHAPE NoE, and the Italy-Israel project SHALOM. Special thanks are given to the reviewers for their comments. Models are courtesy of the AIM@SHAPE repository, the Stanford 3D Scanning Repository, Y. Liu and B. Levy (INRIA), T. Ju and C. Grimm (Washington Univ. in St. Louis).

REFERENCES

[1] M. Attene, S. Biasotti, and M. Spagnuolo. Shape understanding by contour-driven retiling. *The Visual Computer*, 19(2-3):127–138, 2003.

[2] T. Banchoff. Critical points and curvature for embedded polyhedra. *Journal of Differential Geometry*, 1:245–256, 1967.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computations*, 15(6):1373–1396, 2003.

[4] S. Biasotti, B. Falcidieno, L. De Floriani, P. Frosini, D. Giorgi, C. Landi, L. Papaleo, and M. Spagnuolo. Describing shapes by geometric-topological properties of real functions. *ACM Computing Surveys*, 40(4).

[5] S. Biasotti, B. Falcidieno, and M. Spagnuolo. Surface shape understanding based on extended reeb graphss. *Topological Data Structures for Surfaces: An Introduction for Geographical Information Science*, pages 87–103, 2004.

[6] S. Biasotti, S. Marini, M. Mortara, G. Patanè, M. Spagnuolo, and B. Falcidieno. 3D shape matching through topological structures. In *Discrete Gemetry for Computer Imagery*, pages 194–203, 2003.

[7] S. Biasotti, S. Marini, M. Spagnuolo, and B. Falcidieno. Sub-part correspondence by structural descriptors of 3D shapes. *Computer-Aided Design*, 38(9):1002–1019, 2006.

[8] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385–396, 2004.

[9] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry Theory and Applications*, 24(2):75–94, 2003.

[10] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *Proc. of IEEE Visualization*, pages 497–504, 2004.

[11] Y.-J. Chiang, T. Lenz, X. Lu, and G. Rote. Simple and optimal output-sensitive construction of contour trees using monotone paths. *Computational Geometry Theory and Applications*, 30(2):165–195, 2005.

[12] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. *Discrete Computational Geometry*, 32(2):231–244, 2004.

[13] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Spectral surface quadrangulation. *ACM Siggraph 2006*, pages 1057–1066, 2006.

[14] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Geometric Design*, 22(5):392–423, 2005.

[15] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *ACM Symp. on Computational Geometry*, pages 70–79, 2001.

[16] A. Elad and R. Kimmel. On bending invariant signatures for surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1285–1295, 2003.

[17] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[18] A. Fomenko and T. L. Kunii. *Topological Modelling for Visualization*. Springer Verlag, 1997.

[19] R. Gal and D. Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics*, 25(1):130–150, 2006.

[20] R. Gal, A. Shamir, and D. Cohen-Or. Pose-oblivious shape signature. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):261–271, 2007.
[21] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proc. of Symp. on Geometry Processing*, pages 127–137, 2003.
[22] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *ACM Siggraph 2001*, pages 203–212, 2001.
[23] E. Kartasheva. Reduction of h-genus polyhedrons topology. *International Journal of Shape Modeling*, 5(2):179–194, 1999.
[24] F. Lazarus and A. Verroust. Level set diagrams of polyhedral objects. In *Proc. of the Symp. on Solid Modeling and Applications*, pages 130–140. ACM, 1999.
[25] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *ACM Siggraph*, pages 165–172, 2000.
[26] Y.-S. Liu, M. Liu, D. Kihara, and K. Ramani. Salient critical points for meshes. In *Proc. of Symp. on Solid and physical modeling*, pages 277–282, 2007.
[27] J. Milnor. *Morse Theory*, volume 51 of *Annals of mathematics studies*. Princeton University Press, 1963.
[28] M. Mortara and G. Patanè. Shape-covering for skeleton extraction. *International Journal of Shape Modelling*, 8(2):245–252, 2002.
[29] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica*, 38(1):227–248, 2004.
[30] X. Ni, M. Garland, and J. C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. In *ACM Siggraph 2004*, pages 613–622, 2004.
[31] V. Pascucci and K. Cole-McLaughlin. Parallel computation of the topology of level sets. *Algorithmica*, 38(1):249–268, 2003.
[32] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. In *IASTED Conference on Visualization, Imaging, and Image Processing*, pages 452–290, 2004.
[33] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. In *ACM Siggraph 2007*, pages 58.1–58.9, 2007.
[34] G. Patanè, M. Spagnuolo, and B. Falcidieno. Para-graph: graph-based parameterization of triangle meshes with arbitrary genus. *Computer Graphics Forum*, 23(4):783–797, 2004.
[35] G. Patanè, M. Spagnuolo, and B. Falcidieno. Families of cut-graphs for bordered meshes with arbitrary genus. *Graphical Models*, 69(2):119–138, 2007.
[36] G. Patanè, M. Spagnuolo, and B. Falcidieno. Topological generators and cut-graphs of arbitrary triangle meshes. In *Proc. of Shape Modeling and Applications*, pages 113–122, 2007.
[37] G. Patanè, M. Spagnuolo, and B. Falcidieno. Reeb graph computation based on a minimal contouring. In *Proc. of Shape Modeling and Applications*, pages 73 – 82, 2008.
[38] Z. Qian.yi, T. Ju, and H. Shimin. Topology repair of solid models using skeletons. *IEEE Transactions on Visualization and Computer Graphics*, 13:657–685, 2007.
[39] G. Reeb. Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction numerique. In *Comptes rendus de l'Académie des sciences*, pages 847–849. Sciences Park, 1946.
[40] M. Reuter, F.-E. Wolter, and N. Peinecke. Laplace-Beltrami spectra as Shape-DNA of surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006.
[41] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosian. Surface coding based on Morse theory. *IEEE Computer Graphics and Applications*, 11:66–78, 1991.
[42] D. Steiner and A. Fischer. Cutting 3D freeform objects with genus-n into single boundary surfaces using topological graphs. In *Proc. of the Symp. on Solid Modeling and Applications*, pages 336–343, 2002.
[43] D. Steiner and A. Fischer. Finding and defining the generators of genus-n objects for constructing topological and cut graphs. *The Visual Computer*, 20(4):266–278, 2004.
[44] D. Steiner and A. Fischer. Planar parameterization for closed manifolds genus-1 meshes. In *ACM Symp. on Solid Modeling and Applications*, pages 83–92, 2004.
[45] S. Takahashi, Y. Shinagawa, and T. L. Kunii. A feature-based approach for smooth surfaces. In *Proc. of the Symp. on Solid Modeling and Applications*, pages 97–110, 1997.
[46] B. Vallet and B. Levy. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum 27(2)*, 2008.
[47] M. Van Kreveld, R. Van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proc. of the Symp. on Computational Geometry*, pages 212–220. ACM, 1997.
[48] G. H. Weber and G. Scheuermann. Topology-based transfer function design. In J. J. Villanueva, editor, *Proc. of IASTED International Conference on Visualization, Imaging, and Image Processing*, pages 527–532, 2002.
[49] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman. Context-aware skeletal shape deformation. *Computer Graphics Forum*, 26(3), 2007.
[50] Z. Wood, H. Hoppe, M. Desbrun, and P. Schröder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics*, 23(2):190–208, 2004.
[51] H.-B. Yan, S. Hu, R. R. Martin, and Y.-L. Yang. Shape deformation using a skeleton to drive simplex transformations. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):693–706, 2008.
[52] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Skeleton-based variational mesh deformations. *Computer Graphics Forum*, 26(3):255–264, 2007.
[53] T. Zaharia and F. J. Preteux. 3D-shape-based retrieval within the MPEG-7 framework. In *Nonlinear Image Processing and Pattern Analysis*, volume 4304, pages 133–145, 2001.
[54] E. Zhang, K. Mischaikow, and G. Turk. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics*, 24(1):1–27, 2005.

Giuseppe Patanè is researcher at IMATI-CNR and member of the Shape Modelling Group. He received a Ph.D. in "Mathematics and Applications" from the University of Genova (2005), a Post-Lauream Degree Master in "Applications of Mathematics to Industry" from the "F. Severi National Institute for Advanced Mathematics", University of Milan (2000). His research interests include modelling of discrete data, computational geometry, and numerical analysis. During the period 2004-2008, he has been mainly involved in the NoE FP6-IST AIM@SHAPE.

Michela Spagnuolo is Senior Researcher at IMATI-CNR, where she is member of the Shape Modeling Group. She received the Doctorate degree in Computer Science Engineering from the INSA Lyon, France. Her research interests are related to shape-based approaches for modelling 3D data and geometric reasoning for the extraction of shape features from discrete surface models. During the period 2004-2008, she has been mainly involved in the NoE FP6 - IST AIM@SHAPE, as leader of the workpackage "Analysis and Structuring". From March 2008, she is involved in the FP7 Coordination Action FOCUS K3D.

Bianca Falcidieno is a Research Director of the CNR. She has been leading and coordinating in CNR research at international level in advanced and interdisciplinary fields (such as computational mathematics, computer graphics, multidimensional media and knowledge technologies), strongly interacting with outstanding industrial and social application fields: from industrial design to geographic information systems, from manufacturing to semantic web. During the period 2004-2008, she has been the Coordinator of the NoE AIM@SHAPE, in the frame of IST activity "Semantic-based Knowledge Systems" of the European 6th Framework Programme. From March 2008, she is the coordinator of the FP7 C. A. FOCUS K3D.