Reeb Graph Computation Based on a Minimal Contouring

Giuseppe Patanè[*] IMATI-CNR

Michela Spagnuolo[†] IMATI-CNR

Bianca Falcidieno[‡] IMATI-CNR

Abstract

Given a manifold surface $\mathcal{M}$ and a continuous 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 computation of the Reeb graph that is based on the analysis of the iso-contours solely at saddle points and does not require sampling or sweeping the image of f. Furthermore, the proposed approach does not use global sorting steps of the function values and exploits only a local information on f, without handling it as a whole.

By combining the minimal number of nodes in the Reeb graph with the use of a small amount of memory footprint and temporary data structures, the overall computation takes $O(sn)$-time, where n is the number of vertices of the triangulation of $\mathcal{M}$ and s is the number of saddles of f.

Finally, the technique can be easily extended to compute the Reeb graphs of time-varying functions.

Index Terms: I.3.5 [Computational Geometry and Object Modeling]: Boundary representations—Geometric algorithms, languages, and systems−Object hierarchies. *Additional keywords:* Reeb graph, Morse theory, shape analysis and abstraction.

1 Introduction

Differential topology provides a suitable framework for formalizing and solving several problems related to shape understanding through the theoretical link between critical points, their configuration, and the global properties of the input surface. In this context, the *Reeb graph* $\mathcal{R}_G$ of $\mathcal{M}$ with respect to a scalar function $f : \mathcal{M} \to \mathbb{R}$, defined on a smooth surface $\mathcal{M}$, is 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 $f^{-1}(f(\mathbf{p}))$. This relation identifies each connected component of an iso-contour of f with a single class. Furthermore, Morse theory [19] guarantees that the topological changes of the iso-contours occur only at the critical points of f and the Reeb quotient space can be represented by a graph whose nodes correspond to the critical points and the arcs code the evolution of homotopic iso-contours.

We remind that the point $\mathbf{p} \in \mathcal{M}$ is called *critical* if $\nabla f(\mathbf{p}) = \mathbf{0}$ and *regular* otherwise. The scalar function f is called *Morse* if it assumes different values at distinct critical points; if $f(\mathbf{p}) \neq f(\mathbf{q})$ for any couple of distinct points $\mathbf{p}$, $\mathbf{q}$, then f is *simple.* Given a Morse and simple function f, the Reeb graph of $(\mathcal{M}, f)$ defines a canonical decomposition of $\mathcal{M}$ into cells and encodes its topology [29]. Finally, the Reeb graph is *topologically consistent* with $\mathcal{M}$, i.e. if $\mathcal{M}$ is a closed surface, then the number of loops of $\mathcal{R}_G$ is equal to the genus of $\mathcal{M}$.

In shape modeling and digital geometry processing, the usefulness of the Reeb graph has been demonstrated in applications which include local [25, 38] and global [26, 32] parameterization; surface encoding [17, 31, 33] and reconstruction [3]; shape matching [16] and topological noise removal [36].

In the following, we briefly introduce previous work and discuss the main aims and novelties of our approach.

1.1 Previous work

Several algorithms have been proposed for the computation of the Reeb graph of closed surfaces, and also for higher dimensional or time-dependent data. Most of them work by tracking the evolution of the contours, either with a suitable sampling of the image of f or with a complete sweeping process, i.e. contours are traced at each vertex. Let $I := [f_{\min}, f_{\max}]$ be the image interval, 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 extrema of f.

Sampling-based approaches, such as [31], 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 iso-contours do not interpolate critical points; if this condition is not satisfied, then the corresponding iso-value is slightly perturbed in such a way that the new iso-contour passes only through regular points.

A variation [1] of this technique builds on the extension of the Reeb graph equivalence to strips of triangles rather than iso-contours. 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 $f^{-1}(f(t))$. 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. We also note that a surface strip might include several types of critical points; e.g. saddle points together with maxima and/or minima.

Since an arbitrary partition $\mathcal{I}$ of $[f_{\min}, f_{\max}]$ does not guarantee the topological consistency between $\mathcal{M}$ and $\mathcal{R}_G$,

[*]Istituto di Matematica Applicata e Tecnologie Informatiche, Consiglio Nazionale delle Ricerche, Genova - Italy, e-mail: patane@ge.imati.cnr.it

[†]e-mail: spagnuolo@ge.imati.cnr.it

[‡]e-mail: falcidieno@ge.imati.cnr.it

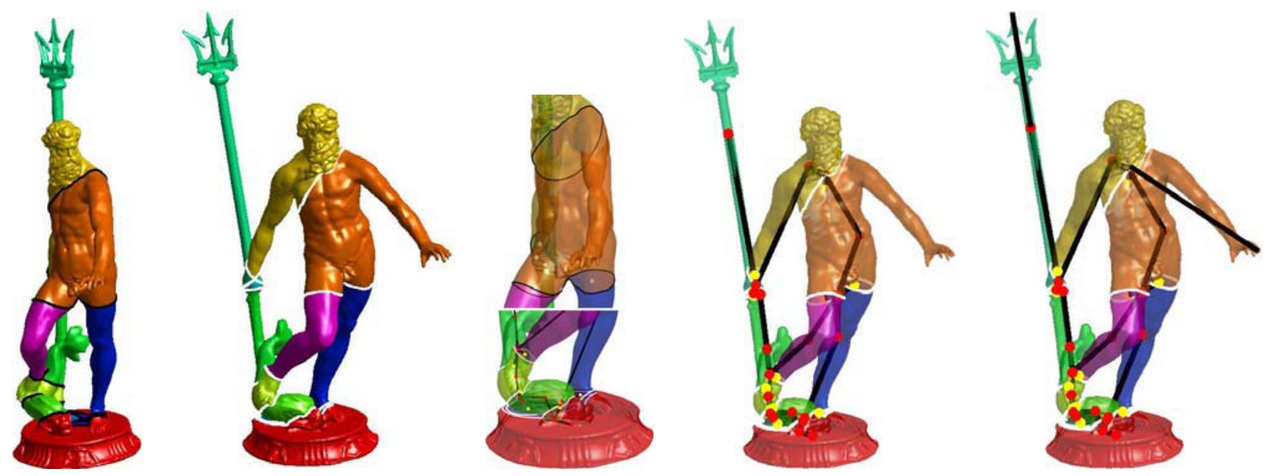


Figure 1: (a-b) Shape segmentation achieved by cutting $\mathcal{M}$ along the iso-contours of $s = 6$ saddle points of a function f with $m = 1$ minimum, $M = 1$ maximum. (c) Zoom-in on the iso-contours (also called *critical loops*) related to the saddle points located in the bottom and body part of the input surface. (d) Adjacency graph among surface patches: red (resp., yellow) points locate the barycenters of the surface patches (resp., critical loops). (e) Conversion of $\mathcal{A}$ to the Reeb graph $\mathcal{R}_G$ of $(\mathcal{M}, f)$ and achieved by joining the maxima and minima of f to the nodes of $\mathcal{A}$ associated to the surface patches they belong to. In (e), the two terminal arcs of the Reeb graph are related to the extrema of f and they are located in the upper part of the trident and the finger of the right hand (viewer point of view).

the initial sampling of Image(f) is updated by iteratively adding a new iso-value α_i in each interval (f_i, f_{i+1}) whose strip $f^{-1}((f_i, f_{i+1}))$ has at least one connected component of genus equal or greater than one. The iteration stops when each strip has 0-genus, thus recovering the 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. Furthermore, the choice of the values f_i and α_i is arbitrary and does not resemble the critical points distribution. The iso-level 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 [16].

Sweeping techniques [6, 7, 8, 17, 23, 24] 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 iso-contour $f^{-1}(f(\alpha))$ to determine when saddle points are encountered and process them. More precisely, in [6] the sweeping algorithm initially sorts the n vertices of the input triangulation by their function values. Then, the *join tree* (resp., *split tree*) is built by performing a sweep of the vertices from the smallest (resp., largest) to the largest (resp., smallest) function value. Finally, the *contour tree* is obtained by merging the join and split tree; this last step requires linear time in the number of vertices.

Since the iso-contours change whenever the iso-value passes through the function value at a vertex, the iso-values are chosen equal to $\{f(\mathbf{p}_i)\}_{i=1}^n$ or set by discretizing the interval $[f_{\min}, f_{\max}]$. In this class of techniques, the method described in [17] deals with closed surfaces equipped with the geodesic distance from a source point and [23] introduces a multi-resolution data-structure for computing and representing the contour tree. Recently, [24] 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 on the base of the creation/deletion of connected components of $\mathcal{M}$ and loops of the graph. Finally, we note that [1, 6, 23, 24] guarantee the topological consistency between $\mathcal{R}_G$ and $\mathcal{M}$.

The method discussed in [7] builds on the sweeping approach but does not sort all the vertices. 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 a symmetric manner.

With the exception of [7, 24] and concerning both the sampling and sweeping approaches, the computation of the set of iso-contours $\{\gamma_{f_i} := f^{-1}(f_i)\}_{i=1}^k$ requires to increasingly (or decreasingly) reordering the values of f at the vertices and initializing the iso-contour computation by searching one edge $[\mathbf{p}, \mathbf{q}]$ of $\mathcal{M}$ that is intersected by γ_{f_i}, i.e. $f(\mathbf{p}) < f_i < f(\mathbf{q})$, $i = 1, \dots, k$. Therefore, this preprocess 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 regular vertices of the input triangulation of $\mathcal{M}$ and the nodes of the Reeb graph $\mathcal{R}_G$; in fact, each regular point $\mathbf{p}_i$ of $(\mathcal{M}, f)$ is associated to a connected component γ of the iso-contour $f^{-1}(f(\mathbf{p}_i))$ and γ is contracted to a node of $\mathcal{R}_G$.

On the one hand, coding each vertex of $\mathcal{M}$ in the Reeb graph is mandatory for the construction of a complete skeleton [17] and the computation of seed-sets for iso-surfacing scalar functions [35]. On the other hand, several applications, which include surface parameterization, texture mapping, and shape matching, do not need a complete information associated to the arcs of the Reeb graph. For instance, a minimal number of nodes and the topological consistency between the surface and the Reeb graph enable us to deal with a low number of segmented patches in local

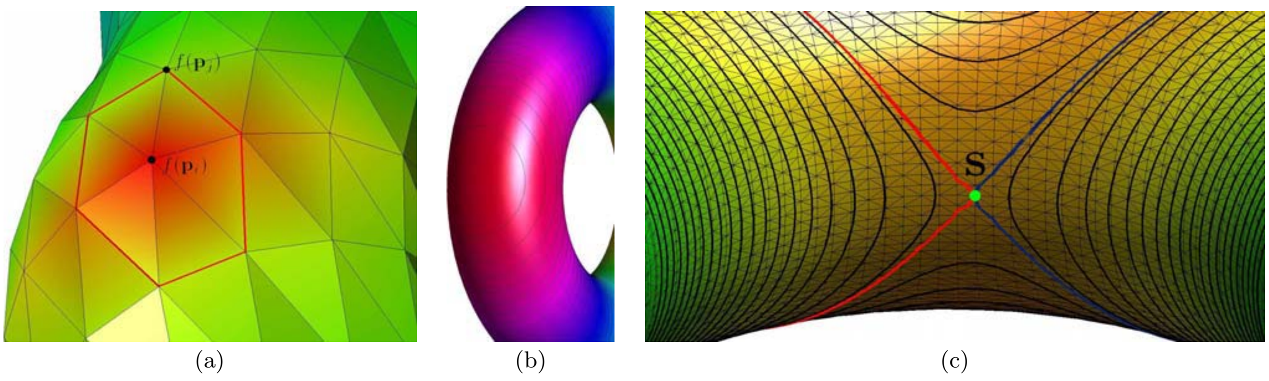


Figure 2: In (a), 1-star of the vertex $\mathbf{p}_i$, (b) iso-contours close to a local maximum, and (c) zoom-in on the two loops (blue and red curve) related to a saddle point (green point) of multiplicity one.

parameterization [25], reduce the computation time of the minimal common subgraph [18] in shape comparison [4], and minimize the number of steps to identify and cut the topological handles in global parameterization [27].

In this context, we propose a novel computation of the Reeb graph that is based on a minimal contouring algorithm. The idea behind our method is to exploit the analysis of the evolution of the iso-contours 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 $\mathcal{M}$. In fact, each node of the computed Reeb graph is associated either with a maximum/minimum of f or with an iso-contour that corresponds to a saddle point. Each arc is associated with a region of $\mathcal{M}$ that is delimited by the connected components of the iso-contours corresponding to saddle points. In this region, the iso-contours related to the regular iso-values of f are homotopic.

The proposed technique does not use global sorting steps and exploits only a local information on f (i.e., the behavior of f on the 1-ring of each vertex, the triangle-triangle adjacency) without handling it as a whole. By combining the minimal number of nodes with the use of a small amount of memory footprint and temporary data structures, our approach takes $O(sn)$-time, where n (resp., s) is the number of vertices (resp., saddles) of $\mathcal{M}$ (resp., f). If $s < \log n$ (e.g., f is a harmonic function [22] or a Laplacian eigenfunction related to a small eigenvalue [10, 30, 34]), then the proposed algorithm outperforms previous work and its computational cost is competitive with respect to the recent on-line computation proposed in [24].

While building the Reeb graph, we also provide a hierarchical shape segmentation into 0-genus patches (i.e., generalized cones, cylinders, and shape junctions) and iso-contours, which code the behavior of f at saddle points. If necessary, we use a persistency-based simplification to pruning the Reeb graph by eliminating clustered saddles before their processing. Exploiting only a local information on f also allows us to easily extract the Reeb graph of time-depending functions. Finally, we discuss the stability of the proposed approach with respect to topological and geometric noise (i.e., tiny handles and perturbation of the surface shape).

We point out that the method discussed in [24] is best suited for the construction of Reeb graphs with respect to functions whose values have been pre-computed and streamed with the mesh geometry. The authors state that the values could be also computed on the fly, but this does not hold for functions that require the complete shape such as the integral geodesic distance from source points, harmonic functions, and Laplacian eigenfunctions. 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 is that the oversampling of the image of f, usually much larger than the number of iso-values strictly necessary to compute the Reeb graph, induces a higher computational cost and an overloaded graph structure with unnecessary regular nodes (i.e., a set of nodes related to a sequence of homotopic contours).

The main differences of [7] 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, as done in [7], might be time-consuming and slowly convergent. Since [7] 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 and the topology (i.e., genus, number of shells) of the volumes in-between the iso-surfaces of two consecutive function values at saddle points.

The paper is organized as follows. In Section 2, we describe the construction of the Reeb graph and in Section 3 we identify a class of scalar functions whose Reeb graphs are commonly used to address several shape modeling problems. Section 4 discusses the main novelties and applications of the proposed approach. Finally, Section 5 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 iso-contours traced at saddle points, that 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

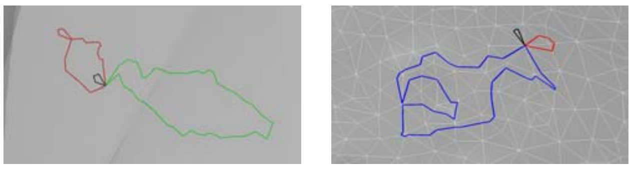


Figure 3: Critical loops related to two saddle points of multiplicity two; each critical loop is shown with a different color.

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).

In the following, we detail the aforementioned steps; Figure 1 depicts the complete framework.

2.1 Critical points classification

In the discrete setting, we consider a triangular mesh $\mathcal{M} := \{M, T\}$, where $M := \{\mathbf{p}_i,\, i = 1, \dots, n\}$ is a set of n vertices and T is an *abstract simplicial complex*, which contains the adjacency information. The function f on $\mathcal{M}$ is defined by linearly interpolating the values $\{f(\mathbf{p}_i)\}_{i=1}^{n}$ of f at the vertices. In the following, we assume that f is general (i.e., $f(\mathbf{p}_i) \neq f(\mathbf{p}_j)$, $i \neq j$); this assumption guarantees that the iso-contours of f are not degenerate, the classifications of the critical points is not ambiguous, and the Euler formula applies (c.f., Equation (1)). To this end, we define the 1-star $N(i)$ of the vertex i as the set of vertices incident to i, i.e. $N(i) := \{j : (i, j)\,\text{edge}\}$.

Critical points on triangle meshes have been defined by Banchoff [2] and the classification of each vertex is done according to the values of f on its neighborhood. The vertex $\mathbf{p}_i$ is a *maximum* (resp., *minimum*) if its function value is higher (resp., lower) than those of its neighborhood. Let

$$Lk(i) := \{j_1, \dots, j_k \in N(i) : \ (j_l, j_{l+1})_{l=1}^{k-1} \text{ edges of } \mathcal{M}\}$$

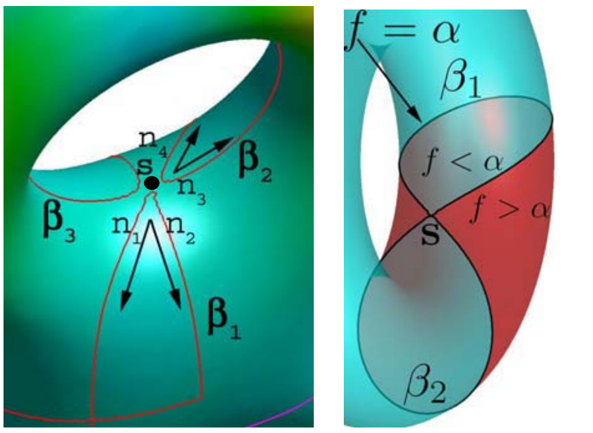
be the *link* (i.e., a clockwise or anticlockwise reordering of the vertices of $N(i)$) and

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

the *mixed link* of i [22], where l is intended as mod $(k + 1)$. If the cardinality of $Lk^{\pm}(i)$ is $2 + 2m_i$, then $\mathbf{p}_i$ is classified as a *saddle* of *multiplicity* $m_i \geq 1$. Once the vertex-vertex relation has been extracted, the classification procedure requires $O(n)$-time. Figure 2 shows the behavior of the iso-contours of f when they approach a critical point.

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

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



(a) (b)

Figure 4: (a) Iso-contour $\beta := \cup_{i=1}^{3} \beta_i$ of a saddle point $\mathbf{s}$ of multiplicity two. For the critical loop β_1 (resp., β_2), the outgoing directions $(\mathbf{n}_1,\, \mathbf{n}_2)$ (resp., $(\mathbf{n}_3,\, \mathbf{n}_4)$) are shown. (b) Evaluation of the sign of $f - \alpha$ around $f^{-1}(f(\alpha))$.

gives the relation between the critical points of $(\mathcal{M}, f)$ and the genus of $\mathcal{M}$ [2, 19]. We explicitly note that the number of saddles is counted with their multiplicity, i.e. $\sum_{\mathbf{s}_i \text{ saddle}} m_i$, where m_i is the multiplicity of the saddle $\mathbf{s}_i$.

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 [25] and the computation of the topological generators of arbitrary 3D shapes [27]. As added value, in this paper we show its capability of handling multiple saddles (see Figure 3) and noisy surfaces.

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 Figure 4(a)) is the union of $m+1$ closed curves $\beta_1, \dots, \beta_{m+1}$ that intersect at $\mathbf{p}_i$, i.e. $\beta := \cup_{l=1}^{m+1} \beta_l \ni \mathbf{p}_i$. In the following of the paper, we refer to β_l as a *critical loop* related to the saddle $\mathbf{p}_i$. For $l = 1, \dots, m+1$ and $j_{2l-1},\ j_{2l} \in Lk^{\pm}(i)$, the vectors $\mathbf{n}_{2l-1} := \mathbf{p}_{j_{2l-1}} - \mathbf{p}_i$ and $\mathbf{n}_{2l} := \mathbf{p}_{j_{2l}} - \mathbf{p}_i$ give the outgoing directions that originate at $\mathbf{p}_i$ and 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 Figure 5(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 Figure 5(d-f)). We 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 [1, 31] and sweeping [6, 17] approaches perform this search on the whole triangle mesh because the location of the intersected edges is not known *a-priori*.

Slicing $\mathcal{M}$ along a given loop β_l requires to duplicate its points and therefore to determine which parts of the triangles intersected by β_l are below and above the iso-contour

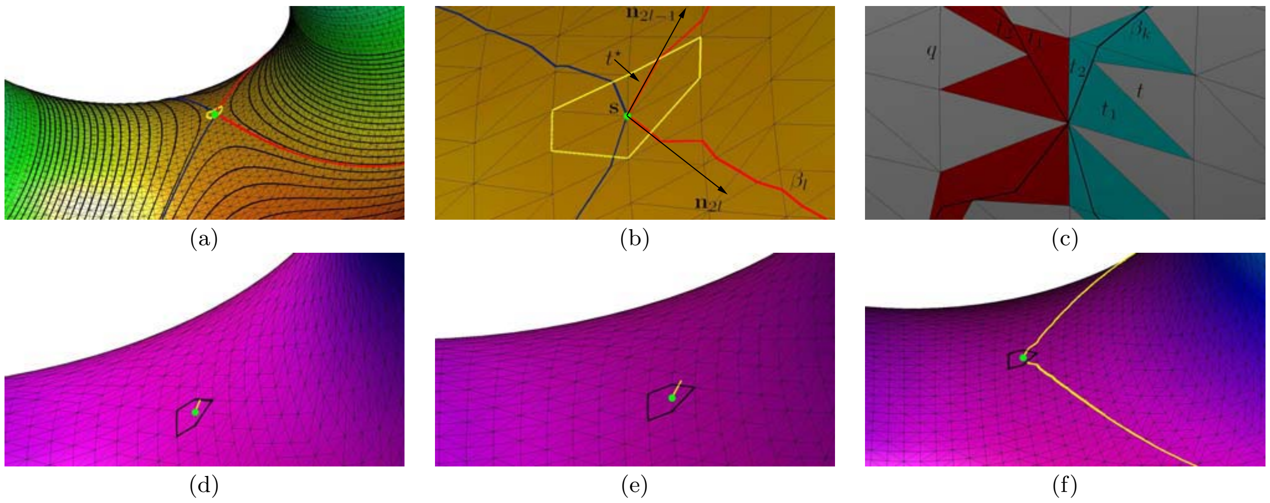


Figure 5: (a) Iso-contours close to a saddle point s; (b) 1-star of s and related critical loops. In (b), the edge e of the triangle $t^\star$ belonging to the 1-star of s is intersected by the loop β_l along the outgoing (resp., incoming) direction $\mathbf{n}_{2l-1}$ (resp., $-\mathbf{n}_{2l}$). The picture also shows the search of the intersection between β_l and the triangle adjacent to $t^\star$. (c) Cut of the triangle through a vertex (blue triangles) and re-triangulation of a quadrilateral face (red triangles): in both cases, the triangles t_1 and t_2 are used to code the adjacency relations between the corresponding patches and the iso-contour β_l. (d-f) Computation of one of the two critical loops of the iso-contour related to s.

(see Figure 4(b)). Since β_l might have a clockwise or an anti-clockwise orientation, we infer the value around β_l by evaluating 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 $\mathbf{p}_{j_{2l-1}}$ to $\mathbf{p}_{j_{2l}}$ along the 1-star of i. If l does not exist, the third vertex $\mathbf{p}_j$ of the triangle with edge $\mathbf{p}_i\mathbf{p}_{j_{2l-1}}$ is external to β_l; then, 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 $\mathbf{s}_i$, we extract at last $s + \sum_{\mathbf{s}_i \text{saddle}} m_i$ critical loops in linear time without using a global sorting of the function values. Figure 3 shows the critical loops related to two saddle points of multiplicity two. To simplify the discussion, in the following we assume to have simple saddles (i.e., $m_i = 1$) where f has different values; in this case, we trace $2s$ critical loops.

2.3 Adjacency graph

Let s be the number of saddles of the input 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 saddle point that has not been processed and consider the two sub-loops β_1, β_2 of the connected component of the iso-contour $\beta := f^{-1}(f(\mathbf{s})) = \beta_1 \cup \beta_2 \ni \mathbf{s}$ that contains $\mathbf{s}$.

For $k = 1, 2$, we slice $\mathcal{M}$ along β_k and store the adjacency relations among the triangles intersected by β_k (see Figure 6(a)). Two situations can happen (see Figure 5(c)): 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. We note that after this step β_1 and β_2 are two disjoint curves; hereinafter, $\mathcal{M}$ refers to the cut surface and its shells.

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 count 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 iso-contours related to the iso-values close to $f(\mathbf{s})$ (see Figure 6(b)).

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 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 Figure 6(b-c), 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 Figure 6(c)).

The iterations proceed until all the saddles of f have been visited. Since the approach processes one saddle point at

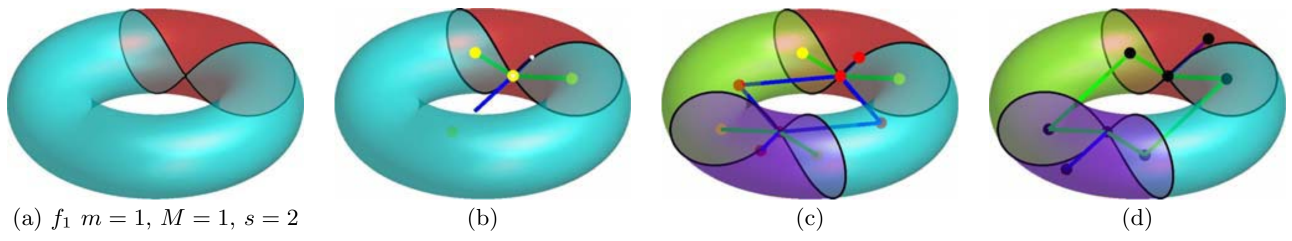


Figure 6: Shape segmentation and adjacency graph of the torus achieved by cutting the input surface along the iso-contour related to the (a-b) first and (c) second saddle point of the first non-trivial Laplacian eigenfunction f_1; (d) Reeb graph of f_1.

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. Then, the Reeb graph is constructed from the adjacency graph as described in Section 2.4 (see Figure 6(d)). Another example is given in Figure 7(a-e).

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 $\mathcal{A}$ (see Figure 6 and 7) and construct the Reeb graph as follows.

- 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 Figure 7(e), yellow region in the middle part of the bitorus).
- 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 Figure 7(e), tubular regions in the up and bottom part of the bitorus).
- 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.

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 $\mathbf{p}_i$. The barycenters of the connected components of an iso-contour are used only for embedding the Reeb graph of $(\mathcal{M}, f)$ in the 3D space and visualizing it; however, we can select any other point as representative of the each connected component.

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

- $\bigcup_{i=1}^{r_k} R_i^k = \mathcal{M}$;
- R_i^k is a connected region, $i = 1, \dots, r_k$;
- $R_i^{k\,\circ} \cap R_j^{k\,\circ} = \emptyset, i \neq j$, with X° internal part of X;
- $R_i^k \neq \emptyset$ has l_i boundary components $\{\gamma_j\}_{j=1}^{l_i}$.

We note that at the iteration k the genus of each patch might be greater than zero and the number of boundary components of each patch is arbitrary. However, at the last iteration $k = s$ we get that each surface patch has 0-genus.

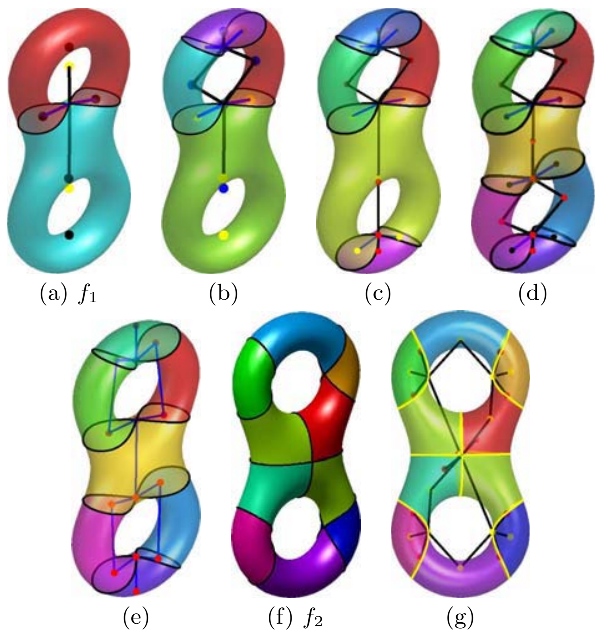
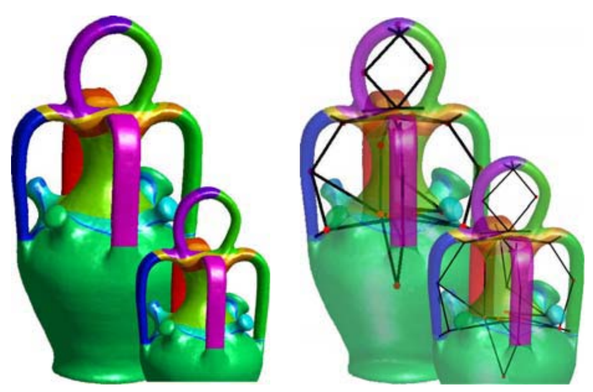


Figure 7: Adjacency graph and segmentation achieved by processing (a) one, (b) two, (c) three, and (d) four saddle points of a function f_1 with $m = 1$ minima, $M = 1$ maxima, and $s = 4$ saddles. (e) Reeb graph of $(\mathcal{M}, f_1)$. (f-g) Shape segmentation and adjacency graph of a function f_2 with $m = 2$ (resp., $M = 2$ and $s = 6$) minima (resp., maxima and saddles).

Therefore, we decompose $\mathcal{M}$ into only three types of primitives, i.e. generalized cones, cylinders, and junctions. 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 $\mathcal{R}_G$, these boundaries correspond to the critical loops of two distinct saddle points. The *shape junctions* have three or more boundary components. In Figure 7(e), the yellow patch R, in the middle part of the bitorus, is a shape junction with four boundary components; in fact, the two critical loops of each one of the two saddles are disjoint boundary components of R. We also note that each saddle point must be duplicated; otherwise, the patch is not manifold. Finally, Figure 6 and 7(a-e) show the hierarchy of adjacency graphs and shape segmentations.

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



(a) $n = 150K$ (b)

Figure 8: (a) Shape segmentation and (b) Reeb graph of a 5-genus surface with a set of non-manifold triangles in the body parts. This example shows that we are able to deal with non-manifold triangles if they are not intersected by the iso-contours related to saddle points.

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.

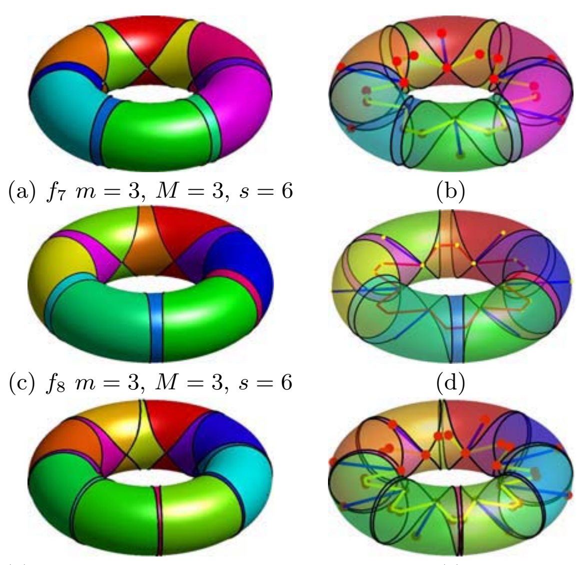
3 Hierarchical shape segmentation and choice of f

Even though the proposed computation of the Reeb graph is independent of f, specific choices require a low computational cost for the construction of the Reeb graph, provide better segmentations of $\mathcal{M}$, and can be used to target specific applications such as quadrilateral remeshing [10], visualization [23], and shape comparison [16]. In fact, each f provides:

- 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. $O(sn)$.

A variety of functions have been used in several applications, which range from surface remeshing and parameterization to shape comparison. For instance, the *height* [13, 31] and *elevation* function [15] are the most intuitive and simple choices for analyzing 3D shapes. The *geodesic* (resp., *Euclidean*) *distance* identifies the surface protrusions [14] by computing the geodesic (resp., Euclidean) distance of the mesh vertices from selected feature points [11, 15, 20] (resp., [13, 16]). *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 [37]) or a multi-scale curvature evaluation [21].

From the perspective of the segmentation, it follows that the best choice of f is a function which 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 review common choices of f and we focus our attention on the shape segmentation provided by harmonic functions and Laplacian eigenfunctions.



(a) f_7 $m = 3$, $M = 3$, $s = 6$ (b)
(c) f_8 $m = 3$, $M = 3$, $s = 6$ (d)
(e) f_{13} $m = 4$, $M = 4$, $s = 8$ (f)

Figure 9: (a, c, e) Shape segmentation and (b, d, f) Reeb graph of the torus induced by three Laplacian eigenfunctions with a different number s (resp., m, M) of saddle points (resp., minima, maxima). The black lines show the iso-contours related to the saddle points.

The *harmonic function* f solves the Laplace equation with Dirichlet boundary conditions, that is,

$$\textbf{Pb1}: \quad \text{find } f : \mathcal{M} \to \mathbb{R} \text{ such that } \begin{cases} \Delta f(\mathbf{p}_i) = 0, & i \in \mathcal{B}^C \\ f(\mathbf{p}_i) = \alpha_i, & i \in \mathcal{B}, \end{cases}$$

where $\mathcal{B} \subset \{1, \dots, n\}$, $\mathcal{B}^C$ is the complementary set of $\mathcal{B}$, and $\alpha_i \in \mathbb{R}$, $i \in \mathcal{B}$. The equations $\Delta f(\mathbf{p}_i) = 0$, $i = 1, \dots, n$, can be written in matrix form as $L_{un}\mathbf{f} = \mathbf{0}$, where

$$L_{un}(i,j) := \begin{cases} \sum_{k \in N(i)} w(i,k) & \text{if } i = j \\ -w(i,j) & \text{if } (i,j) \text{ is an edge of } \mathcal{M} \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

is the *un-normalized graph Laplacian matrix*, $w(i,j)$ is the weight associated to the directed edge (i,j), $j \in N(i)$, and $N(i)$ is the 1-star of i. As coefficients $w(i,j)$ we can select the *mean-value* [12], *cotangent* [28], and *normalized cotangent weights* [9]. Regardless the choice of the weights, (Pb1) is equivalent to the sparse linear system $L^\star \mathbf{f}^\star = \mathbf{b}$, where $\mathbf{f}^\star := (f(\mathbf{p}_i))_{i \in \{1,\dots,n\} \setminus \mathcal{B}}$ is the vector of unknowns, $\mathbf{b}$ is a constant vector related to the boundary conditions, and $L^\star$ is achieved by removing the i^{th}-row and i^{th}-column of L_{un}, $i \in \mathcal{B}$.

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 by applying 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

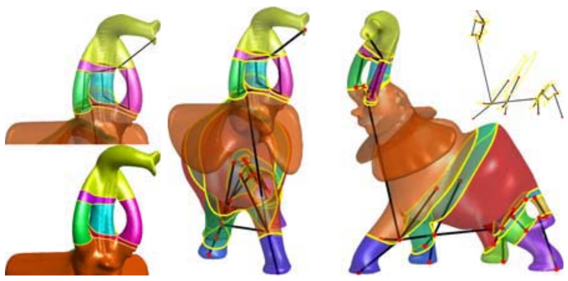


Figure 10: Shape segmentation and Reeb graph of a 3-genus surface with respect to the third Laplacian eigenfunction.

motivation to use harmonic functions for shape segmentation and the Reeb graph construction; in fact, 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 Figure 8.

If there is not a predefined choice of the Dirichlet boundary conditions, then the Laplacian eigenfunctions provide an alternative to harmonic functions and they still guarantee a low number of critical points and a smooth behavior on $\mathcal{M}$. More precisely, we now consider the *eigenvalue problem*

$$\textbf{Pb2}: \quad \text{find } f : \mathcal{M} \rightarrow \mathbb{R} \text{ such that } \Delta f = \lambda f, \quad \lambda \in \mathbb{R}. \quad (3)$$

As shown in [30, 34], in the discrete setting (3) is equivalent to the *generalized eigenvalue problem*

$$L_{\text{cot}}\mathbf{f} = \lambda B\mathbf{f}, \quad \mathbf{f} := (f(\mathbf{p}_i))_{i=1}^{n},$$

where L_{cot} is the un-normalized graph Laplacian matrix with cotangent weights and B codes the geometry of $\mathcal{M}$ in terms of the triangles areas. Finally, we mention that this approximation can be improved by using higher degree finite elements; for more details on the cubic case, we refer the reader to [30]. Examples are shown in Figure 9 and 10. Finally, another discretization of (Pb2) is to consider as L the un-normalized Laplacian matrix (2) and compute the eigensystem associated to the *standard eigenvalue problem* $L\mathbf{f} = \lambda\mathbf{f}$ [10, 34].

4 Discussion

This section discusses the computational cost and stability of the proposed approach; we also analyze its degrees of freedom and the computation of the Reeb graph of time-varying function.

4.1 Computational cost and stability issues

We note that slicing $\mathcal{M}$ along β_k, $k = 1, 2$, (resp., counting the number of shells) takes linear time in the number of triangles intersected by β_k (resp., of $\mathcal{M}$) and the overall cost for processing the saddle $\mathbf{s}$ is $O(k)$, with k number of intersected edges. Therefore, 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 $\mathcal{E} := \{(e_k, t_k)\}_k$,

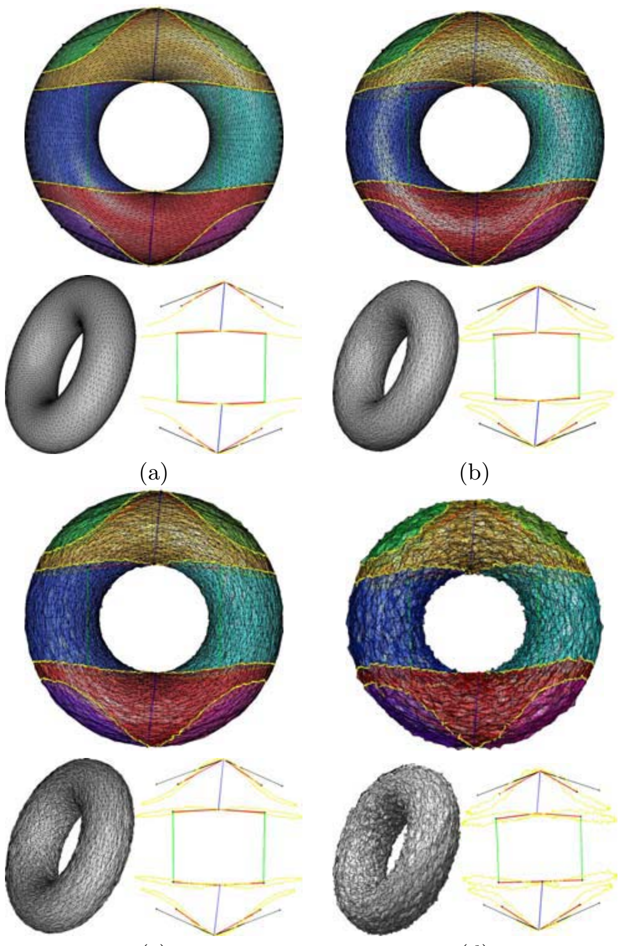


Figure 11: From (a) to (d): stability of the saddle iso-contouring and computation of the Reeb graph with respect to an increasing noise on the surface shape.

where $e := (j_k, j_{k+1})$ 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, $t_k \in [0, 1]$, 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 count 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. Figure 11 shows the stability of the saddle iso-contouring and the computation of the Reeb graph with respect to a different noise of the surface shape.

4.2 Degrees of freedom

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

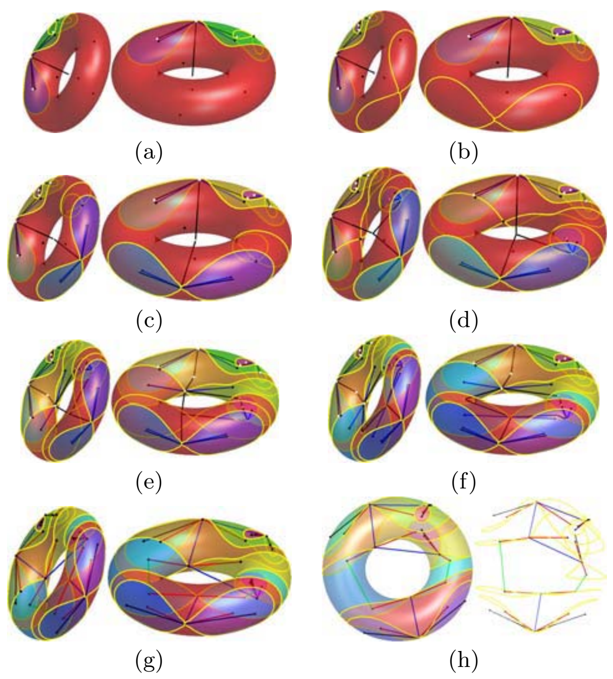


Figure 12: (a-g) Adjacency graphs of a scalar function on the torus with $m = 3$ (resp., $M = 3$, $s = 6$) minima (resp., maxima, saddles). (h) Shape decomposition and Reeb graph.

Table 1: The table shows the computational cost of the main steps of the proposed framework, where n (resp., s and r_k) is the number of vertices (resp., saddle points and shells at the iteration k) of $\mathcal{M}$.

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	–	$O(sn)$

are possible. A simple choice is to increasingly (or decreasingly) reorder 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 $\mathcal{M}$. Alternatively, we sort the saddle points according to their persistency value; as proposed in [5], critical points are paired by visiting $\mathcal{M}$ with respect to the increasing reorder of the values of f and the importance weight associated to the pair $(\mathbf{p}_i, \mathbf{p}_j)$ is measured as the *persistence* of $\mathbf{p}_i$, $\mathbf{p}_j$, that is, $|f(\mathbf{p}_i) - f(\mathbf{p}_j)|$. Given a threshold $\epsilon > 0$ we can also consider only the saddle points whose persistency value is grater than ϵ and use the corresponding iso-contours to build the Reeb graph. In this way, we prune the Reeb graph by eliminating clustered saddle points and irrelevant topological features before their processing. Other examples are shown in Figure 12 and 13.

4.3 Extension to time-depending functions

We note that our approach easily handles time-depending functions $f_t : \mathcal{M} \to \mathbb{R}$; in fact, 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 Figure 14.

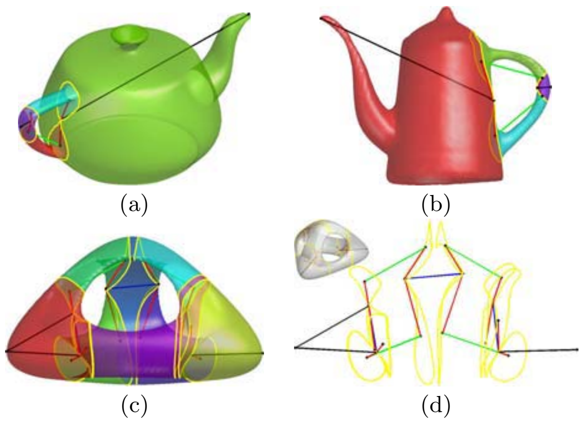


Figure 13: Shape segmentation, iso-contours at saddle points, and Reeb graph of a surface of genus (a-b) one and (c-d) three.

5 Future work

The common approach for computing the Reeb graph of a scalar function $f : \mathcal{M} \to \mathbb{R}$, 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 (resp., n) of the saddle points (resp., vertices 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 [5]. As future work, we plan to extend the proposed approach to computing the Reeb graph of 3D scalar function by studying the adjacency relations and the topology (i.e., genus, number of shells) of the volumes in-between the iso-surfaces of two consecutive critical function values.

Acknowledgments Special thanks are given to the anonymous reviewers for their comments and suggestions. This work has been partially supported by the FP6 IST AIM@SHAPE NoE.

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] S. Biasotti. *Computational Topology Methods for Shape Modelling Applications*. PhD thesis, Università degli Studi di Genova, May 2004.

[4] S. Biasotti, S. Marini, M. Mortara, G. Patanè, M. Spagnuolo, and B. Falcidieno. 3D shape matching through topological structures. In *DGCI*, pages 194–203, 2003.

[5] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A topological hierarchy for functions on triangulated

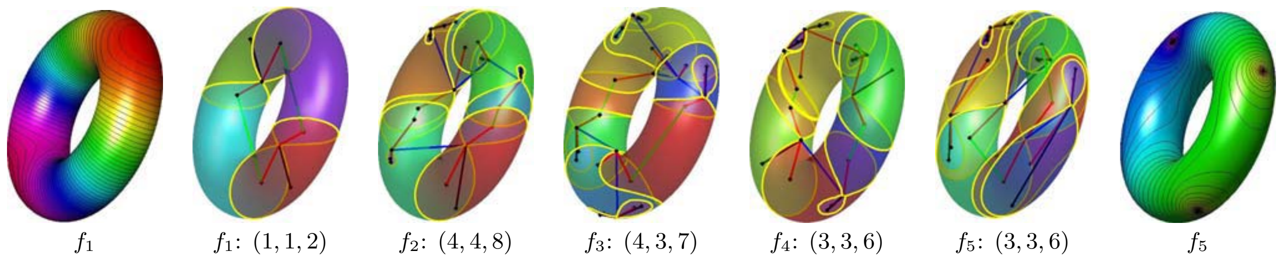


Figure 14: Iso-contours and Reeb graphs of a time-depending scalar function at different time steps. Each row also shows the number (m, M, s) of m (resp., M, s) minima (resp., maxima, saddle points).

surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385–396, 2004.

[6] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Comput. Geom. Theory Appl.*, 24(2):75–94, 2003.

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

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

[9] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *ACM SIGGRAPH 1999*, pages 317–324, 1999.

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

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

[12] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, pages 157–186. 2005.

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

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

[15] R. Gal, A. Shamir, and D. Cohen-Or. Pose-oblivious shape signature. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):261–271, 2007.

[16] 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.

[17] 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.

[18] B. T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–504, 1998.

[19] J. Milnor. *Morse Theory*, volume 51 of *Annals of mathematics studies*. Princeton University Press, 1963.

[20] M. Mortara and G. Patanè. Shape-covering for skeleton extraction. *International Journal of Shape Modelling*, 8(2):245–252, 2002.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] G. Patanè, M. Spagnuolo, and B. Falcidieno. Families of cut-graphs for bordered meshes with arbitrary genus. *Graphical Models*, 69(2):119–138, 2007.

[27] 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.

[28] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[29] G. Reeb. Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction numerique. In *Comptes Rendu Acad. Sciences*, pages 847–849. Sciences Park, 1946.

[30] 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.

[31] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosian. Surface coding based on Morse theory. *IEEE Computer Graphics and Applications*, 11:66–78, 1991.

[32] 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.

[33] 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. ACM, 1997.

[34] B. Vallet and B. Levy. Manifold harmonics. *To appear in EUROGRAPHICS*, 2008.

[35] 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 Symposium on Computational geometry*, pages 212–220. ACM, 1997.

[36] Z. Wood, H. Hoppe, M. Desbrun, and P. Schröder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics*, 23(2):190–208, 2004.

[37] 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.

[38] E. Zhang, K. Mischaikow, and G. Turk. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics*, 24(1):1–27, 2005.