Shape Approximation by Differential Properties of Scalar Functions

Silvia Biasotti[a], Giuseppe Patanè[a], Michela Spagnuolo[a], Bianca Falcidieno[a], Gill Barequet[b,c]

[a] *Istituto di Matematica Applicata e Tecnologie Informatiche - Consiglio Nazionale delle Ricerche, Genova, Italy*
[b] *Department of Computer Science, Technion - Israel Institute of Technology, Haifa, Israel*
[c] *Department of Computer Science, Tufts University, Medford, MA*

Abstract

This paper presents a method of shape chartification suitable for surface approximation. The innovation of this approach lies on the definition of an iterative refinement of the shape into a set of patches that are automatically tiled and used to approximate the original shape up to a prescribed error. The coding of the patches is supported by the Reeb graph and contains the rules to properly tile, stitch them, and reconstruct the original shape while preserving its topology, using a technique which is also exploited for reconstructing an object from non-planar contours. The method is geometry-aware by definition, as the nodes of the Reeb graph are representative of the main shape features, which belong to the approximated shape already at the initial iteration steps. The points of the reconstructed shape belong to the original surface, their total number is highly reduced, and the original connectivity is replaced by a set of patches that preserves the global topology of the input shape.

Key words: Shape chartification, Morse theory, shape approximation, shape reconstruction.

1. Introduction

Reeb graphs have been proven to be effective in shape analysis, as they provide a very compact and synthetic description of a surface embedded in $\mathbb{R}^3$ via the characterization of its main features by the critical points of a real function f. The critical points of f identify changes in the topology of the contours of the function f: saddles correspond to splitting or merging of the contours of f, while minima or maxima correspond to contour creation or termination. The topological structure coded by the Reeb graph is often associated to its geometric embedding, or topological skeleton, which is an iconic representation of the shape. Usually, the nodes are placed at the critical points of f, and the edges, which store their correspondence through topological evolution of the level sets of f, are drawn in the interior of the shape as a kind of a centerline.

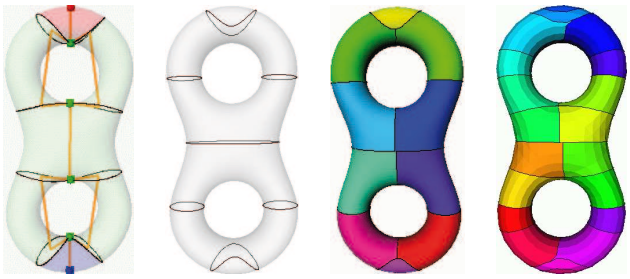
When drawing the level sets of f before and after the saddle points, the iconic representation provided by the Reeb graph sketches the original shape. Imagine further that we tile these level sets by joining their vertices while respecting the contour correspondence stored in the graph: even if approximated by a highly sparse set of contours, we immediately have a rough idea of the object shape [9].

This observation led us to code a surface by means of a set of patches that are automatically tiled and used to approximate the original shape up to a prescribed error. The way the patches are stored contains the rules to properly tile and stitch the patches while preserving the topology of the original shape. The method, therefore, can be exploited to compress the shape of 3D objects: the points in the reconstructed model belong to the original object surface, their total number is highly reduced, and the original connectivity is replaced by a tiling of the patches that preserves the global topology of the input shape. Moreover, the method is geometry-aware by definition, as the nodes of the Reeb graph are representative of relevant shape features and belong to the approximated shape already at the first iteration.

The idea of resorting to tiling for the approximation phase is motivated by the low computational cost that this technique implies. Note, however, that the contours associated to the nodes in the Reeb graph are in general neither planar nor parallel (i.e., not lying on the parallel planes). Indeed, the level sets of f are planar only for particular choices of f (e.g., the height function) and a straightforward triangulation of the level sets of f might turn out to be awkward, if not properly approached. While tiling two parallel contours is relatively trivial, to the best of our knowledge, tiling a set of non-planar contours has never been addressed in the literature in its general formulation. In our case, the Reeb graph induces correspondences among contours, which solve one of the big issues of the problem in its general settings. Moreover, we refine the contour decomposition such that the approximation phase can work automatically using any classical contour-to-contour tiling algorithms [15, 25, 26, 34].

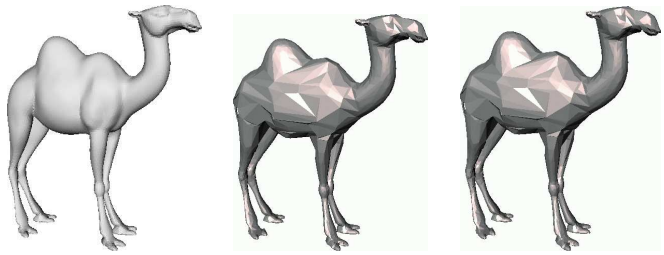
The intuition behind the refinement strategy is the following. First of all, we insert on the surface the so-called *middle contours*, which are placed between pairs of adjacent critical points (Figure 1(a,b)). The tiling of the middle contours may cause problems in the portion of the shape delimited by levels sets that include a saddle point: here, the branching of the shape might cause self-intersections and twisting effects in the recon-

Email addresses: silvia@ge.imati.cnr.it (Silvia Biasotti), patane@ge.imati.cnr.it (Giuseppe Patanè), michi@ge.imati.cnr.it (Michela Spagnuolo), bianca@ge.imati.cnr.it (Bianca Falcidieno), barequet@cs.technion.ac.il (Gill Barequet)



(a) (b) (c) (d)

Figure 1: (a) Reeb graph, (b) set of middle contours, (c) initial chartification and (d) after one refinement step. Different colors correspond to different charts.



(a) (b) (c)

Figure 2: (a) Original model and (b) its approximation. (c) The Reeb graph that led to the shape approximation.

struction. The refinement of the tiling is based on the splitting of these branching sites into two patches obtained by drawing the flow paths from the saddle point to the upper or lower contours, as depicted in Figure 1(c). The surface is decomposed into charts delimited by flow paths and level sets of f, and having in general an irregular connectivity. During the approximation phase, the patches that contain a maximum, or a minimum, of f are handled as generalized cones, while the patches that arise from the split of branching sites will be handled as generalized cylinders. At the end of the first refinement step, the shape is approximated by a highly sparse set of contours, whose cardinality is given by the number of the critical points of f. These contours are sufficient to approximate the original shape from the perspective of its global topology, and also provide a rough approximation of its geometry. To gain accuracy during the reconstruction phase, each patch is iteratively refined by the insertion of new level sets until the original shape is approximated to a given accuracy (Figure 1(d)).

Note that the decomposition is not equivalent to the one induced by the Morse-Smale decomposition, where the boundaries of the patches are all delineated by flow paths. Only at the first iteration, the decomposition may be regarded as the intersection between the Reeb graph decomposition induced by the middle contours and the Morse-Smale one, where only some of the flow paths are kept in the intersection.

Using the proposed coding, the information about the original shape can be highly reduced: for instance, in Figure 2 the original model was represented by a triangle mesh having 35K vertices and 69K triangles and whose VRML file was 2.6Mb. The file produced to store the refined Reeb graph contains just 3.8K vertices which, together with the information required in the reconstruction phase, accounts only for 138K bytes of storage needed to approximate the original shape within a Hausdorff distance of 3.7% of the diagonal of the bounding box.

Our main contributions are (i) an iterative shape chartification into a set of patches that are automatically refined, tiled, and used to approximate the original shape; (ii) the possibility to drive both the decomposition and reconstruction using different criteria such as approximation accuracy, topological consistency, geometry-awareness through the chosen functions; and (iii) the capability of concisely encoding the shape with a set of surface samples whose original connectivity is replaced by a set of contours and tiling rules that allow us to approximate the original model.

The paper is organized as follows. Section 2 presents a brief overview of previous work relevant to the proposed approach to shape coding and decoding. In Section 3 we describe the novel shape chartification, and in Section 4 we show how the chartification is used to adaptively encode and decode (i.e., approximate) the input shape. Section 5 discusses several examples, and Section 6 summarizes the main contribution of our approach and outlines a few future research directions.

2. Previous work

We briefly review the existing literature on shape encoding (chartification) and decoding (reconstruction from contours), focusing on methods that address issues that resemble the two main steps of our approach.

Shape chartification. There are several methods in the literature for partitioning an arbitrary surface into a set of charts of simpler topology and geometry, which are then used for remeshing, texture mapping, compression, and approximation. Shape segmentation generally provides a set of basic primitives (e.g., planes, spheres, cylinders, tori) or identifies relevant parts, which are delimited by lines of concave discontinuity of the tangent plane [46]. For local parameterization and texture mapping [24, 41, 48, 53], a chartification into disk-like patches is often computed by converting the input surface $\mathcal{M}$ into a base domain with the same topology of $\mathcal{M}$ through simplification. Surface partitioning into quadrilateral patches [44, 49] is used to support approximation schemes with tensor-product B-splines. Recent techniques use Morse theory and Laplacian eigenfunctions [12, 20, 30], holomorphic discrete 1-forms [28, 29], discrete harmonic functions [49], variational techniques [16, 32], and discrete exterior calculus [18]. Finally, geometry-aware maps [47], whose behavior is guided by the selection of anchor points, have been efficiently used for shape compression.

The method proposed in [21] uses a network of flow paths, related to two orthogonal vector fields, and provides a quadrilateral remeshing of $\mathcal{M}$ whose number of patches is driven by the target approximation accuracy. In [20], the Morse complex of the Laplacian eigenfunctions is used to define a quadrangulation of the shape. Such spectral quadrilateral remeshing produces good results but needs Laplacian eigenfunctions with a

number of critical points sufficiently high to provide the corner vertices of the quad patches (i.e., two opposite saddles, a maximum and minimum as corners). Furthermore, a filtering of the critical points with low persistence values and a smoothing of the arcs of the Morse complex are necessary to improve the geometric quality of the patches. In fact, a small perturbation of the input scalar function makes the two ascending flow paths miss the associated saddles and approach a maximum or minimum without meeting each other [20].

Shape reconstruction from contours. A surface connecting *two* polygons contained within parallel planes can be constructed by triangulating between the contour lines and finding the optimal triangulation by using graph theory [34]. This algorithm can handle only the simple one-to-one case and implicitly assumes a high degree of resemblance between the contours. Successive methods [25, 26, 51] interpolate the surface between two contours and differ by whether a local or global "advancing rule" is used for the tiling, which tiling measure is optimized, and which algorithm is used to find this optimum.

One further step was taken in [14, 15]—the handling of simple branching cases through the use of an intermediate slice, which resembles the original slices; the insertion of "bridges" between the multiple contours of one slice; and the splitting of the single contour into several contours. More general branching cases [1, 3, 31] are solved using a Delaunay-like triangulation of the contour vertices, slice projection, partial tiling, and a straight-skeleton analysis of the symmetric difference of the slices. Finally, the Delaunay-meshing strategy in [31] has been recently extended to surface reconstruction from non-parallel planes [11, 35].

3. Reeb graph refinement

In the following, we outline the discretization we adopt for the computation of the critical points and flow paths, which are needed to define the shape decomposition. Then, we describe the chartification and its iterative refinement through the insertion of level sets, subject to accuracy requirements. Finally, we discuss degenerate cases, the computational complexity, and issues related to the selection of the function.

3.1. Theoretical background

We assume that the input surface is a 2-manifold closed triangle mesh $\mathcal{M}$; the function $f : \mathcal{M} \to \mathbb{R}$ is a piecewise linear function defined on the vertices of $\mathcal{M}$ and extended by linear interpolation across the edges and faces. Assuming that for any edge $(\mathbf{p}_i, \mathbf{p}_j)$, $f(\mathbf{p}_i) \neq f(\mathbf{p}_j)$, the gradient of f is constant, non-zero and well defined across the interiors of triangles and edges. In particular, we discretize ∇f on a triangle t with vertices $(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$ and unit normal $\overline{\mathbf{n}}$ as the solution $\nabla f|_t$ of the 3×3 linear system

$$\begin{pmatrix} \mathbf{p}_j - \mathbf{p}_i \\ \mathbf{p}_k - \mathbf{p}_j \\ \overline{\mathbf{n}} \end{pmatrix} \nabla f|_t = \begin{pmatrix} f(\mathbf{p}_j) - f(\mathbf{p}_i) \\ f(\mathbf{p}_k) - f(\mathbf{p}_j) \\ 0 \end{pmatrix}.$$

The critical points of f are located at the vertices of $\mathcal{M}$ and classified by analyzing the distribution of the function values on a neighborhood of each vertex [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. According to [38], let

$$Lk(i) := \{j_1, \ldots, j_k \in N(i) : (j_s, j_{s+1})_{s=1}^{k-1} \text{ edges of } \mathcal{P}\}$$

be the *link* of i, then the *upper link* is the set

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

and the *mixed link* is given by

$$Lk^\pm(i) := \{j_s \in Lk(i) : f(\mathbf{p}_{j_{s+1}}) > f(\mathbf{p}_i) > f(\mathbf{p}_{j_s}) \text{ or } f(\mathbf{p}_{j_{s+1}}) < f(\mathbf{p}_i) < f(\mathbf{p}_{j_s})\},$$

where $j_{k+1} := j_1$. The *lower link* $Lk^-(i)$ is defined by replacing the inequality ">" with "<" in 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 $2 + 2m$, $m \geq 1$, then $\mathbf{p}_i$ is classified as a *saddle* of *multiplicity* m.

For simplicity, we assume that each saddle has multiplicity one (i.e., it is a *Morse* saddle) and the function is *simple* (i.e., it is injective over the critical points). Degenerate cases and possible extensions of these hypotheses are discussed in Section 3.4.

An integral line of f is defined as the line of steepest ascent/descent values of ∇f, and it is discretized on a mesh as a *flow path*. Each flow path γ of f is a piecewise linear curve over the surface that follows the variation of ∇f and consists of a sequence of nodes, which are the intersection points of γ with the edges of $\mathcal{M}$. If a node of γ is a vertex $\mathbf{p}_i$ of $\mathcal{M}$, then we trace γ using the direction of $\nabla f|_t$ on the triangle t of the 1-star of $\mathbf{p}_i$ such that the intersection of γ with t is the point $\mathbf{q} \in \mathcal{M}$ with the highest persistence value $|f(\mathbf{p}_i) - f(\mathbf{q})|$. If the flow path is aligned with an edge, then we follow the edge along with the value of f that does not decrease/increase. For more details on the algorithm for tracing flow paths, we refer the reader to [20, 37]. Note that flow paths on meshes never cross, but can merge; in any case, once merged they do not separate.

3.2. Initial shape chartification

The shape chartification builds on the Reeb graph [45] as a support structure. The Reeb graph of $\mathcal{M}$ with respect to a Morse and simple map $f : \mathcal{M} \to \mathbb{R}$ is defined as the one-dimensional finite and connected simplicial complex whose nodes correspond to the critical points of f and whose arcs join pairs of critical points when the contours evolve from one critical point to the other without changing their topology type [8, 42].

Given the Reeb graph $\mathcal{R}_\mathcal{G}$ of $(\mathcal{M}, f)$, we consider its geometric embedding by associating to each node $\mathbf{n}$ the coordinates of the critical point and the iso-value $f(\mathbf{n})$. An orientation can be given to the arcs of $\mathcal{R}_\mathcal{G}$ conforming with the growing directions of the values of f.

If $e = (\mathbf{n}_1, \mathbf{n}_2)$ is an arc of $\mathcal{R}_\mathcal{G}$, then we denote $\mathcal{M}_{|e}$ the portion of $\mathcal{M}$ that corresponds to the arc e. The iso-contour of f defined by $f^{-1}\left(\frac{f(\mathbf{n}_1)+f(\mathbf{n}_2)}{2}\right) \cap \mathcal{M}_{|e}$ is called the *middle contour* of the arc e. The set of all the middle contours is indicated as $\partial\mathcal{S}$.

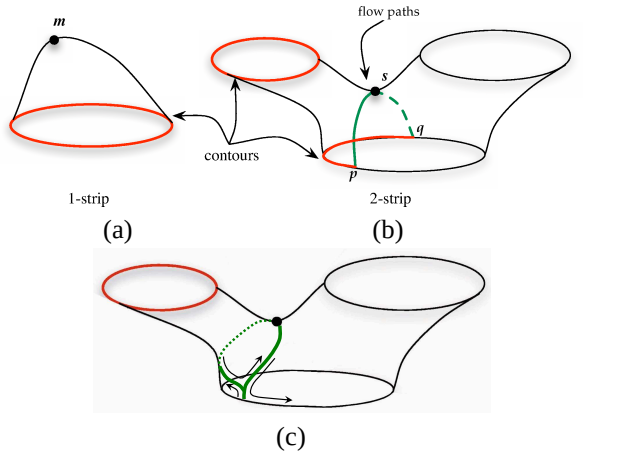


Figure 3: Types of charts in the initial decomposition: 1-strips and 2-strips. In (c), boundary of the two 2-strips when the flow paths merge.

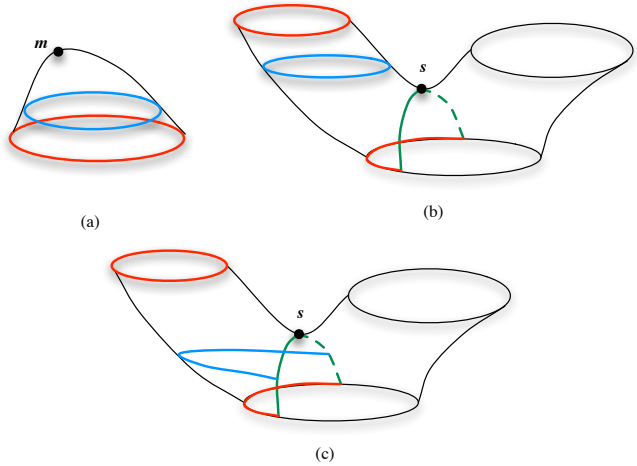


Figure 4: The refinement of a 1-strip (a), and (b,c) the two possible refinements of a 2 strip.

By definition of $\partial\mathcal{S}$, we have that $|\partial\mathcal{S}| = |E|$, where $|E|$ is the number of edges of $\mathcal{R}_\mathcal{G}$. Figure 1(b) shows an example of contours of f traced in the middle of the arcs of the Reeb graph with respect to the values assumed by f (Figure 1(a)).

The set of middle contours $\partial\mathcal{S}$ induces a mesh decomposition into regions. A region $\mathcal{S}$ of $\mathcal{M}$ delimited by $\partial\mathcal{S}$ satisfies one of the following properties (Figure 3):

1. $\mathcal{S}$ is a cap (1-*strip*) that includes only one minimum or maximum and has only one connected boundary component corresponding to an iso-value of f;
2. $\mathcal{S}$ is a branch (3-*strip*) that includes one saddle point **s** and with 3 connected boundary components corresponding to iso-values of f. Note that f assumes different values on at least two boundary components of $\mathcal{S}$; beside symmetries in f, these three values are always different.

This chart decomposition defines the initial step of the Reeb graph refinement that is used to approximate the surface. We used the middle contour of the edge e in $\mathcal{R}_\mathcal{G}$ instead of its geometric medial section to give more weight to f in the decomposition step. This choice emphasizes the role of the function f as the key to identify and measure the shape properties, which will be kept as anchors for the shape-approximation phase.

To ease the reconstruction via tiling, we split each 3-strip $\mathcal{S}$ into two 2-strips, i.e., charts with two boundary components. At the initial stage, each chart, has a number of connected boundary components that corresponds to the degree of the Reeb graph $\mathcal{R}_\mathcal{G}$ that the strip contains. For a 3-strip $\mathcal{S}$, its boundary components β_i, $i = 1, 2, 3$, can be ordered according to the value of f: since f is constant on β_i, we refer to the corresponding iso-value as $f(\beta_i)$, $i = 1, 2, 3$, and order them with respect to increasing values of f as β_1, β_2, and β_3. Reasoning on the possible evolution of the contours across the saddle **s**, we have only the following two cases: either f evolves through **s** with a contour splitting (β_1 splits into β_2 and β_3), or f evolves through **s** with a contour merging (β_1 and β_2 merge into β_3). In the first case, the node corresponding to **s** in $\mathcal{R}_\mathcal{G}$ has in-degree 1 while in the second case it has out-degree 1. To split a 3-strip $\mathcal{S}$, we draw all flow paths from the saddle $\mathbf{s} \in \mathcal{S}$ to β_1, if the corresponding node has in-degree 1, or to β_3 otherwise.

Note that the flow paths split the 3-strip into two 2-strips even if the flow paths merge. In this latter case, the two points **p** and **q** in Figure 3(b) overlap. Figure 3(c) details the two 2-strips we obtain: the boundary of one strip is made of one contour and the portions of the flow paths that do not overlap, while the boundary of the other one contains two contours and the whole flow paths, eventually duplicated in the portions merged. The small arrows in the picture highlight the boundary orientation in correspondence of the flow paths.

Applying this step to all the 3-strips of $\mathcal{M}$, we get the initial chart decomposition into 1-strips and 2-strips, which have only two maximally connected boundary components. Note that no surface patch includes saddle points in their interior.

3.3. *Refinement of the chartification*

The chart decomposition drives the approximation scheme by adding more and more contours to the initial decomposition until the approximation error is below a given value. The refinement process is independently applied to single charts and driven by the approximation error (Section 4), so that the final segmentation will be adapted locally to the shape complexity.

Let $\mathcal{S}$ be a chart selected for refinement as it does not match the target approximation error. We insert a new contour at the value $(f_{\mathcal{S}max} + f_{\mathcal{S}min})/2$, where $f_{\mathcal{S}max}$ and $f_{\mathcal{S}min}$ are the maximum and minimum values, respectively, of f over the chart $\mathcal{S}$. If $\mathcal{S}$ is a cap region that contains a critical point **m**, it is split into one new 1-strip that still contains the critical point **m** and one new 2-strip (Figure 4(a)). Similarly, a 2-strip $\mathcal{S}$ is split into two charts by the insertion of a new contour, which may give rise to one of the following cases:

- if the inserted contour does not intersect any boundary component of $\mathcal{S}$, the chart is refined into two 2-strips, the first with its two boundary components both on level sets of f, and the second with one boundary component on the

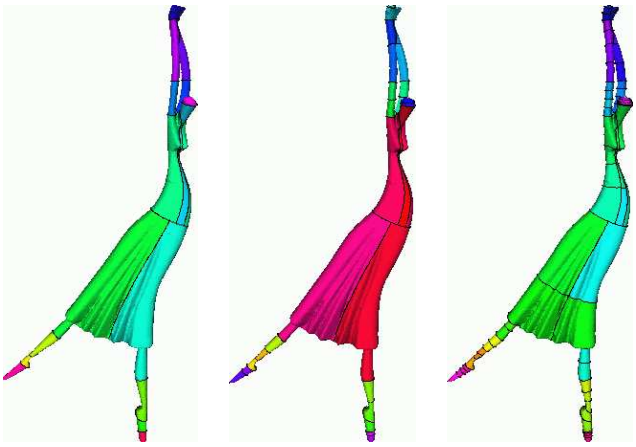


Figure 5: Three refinements of the chartification.

level set of f and one that alternates flow paths and level sets (Figure 4(b));

- if the inserted contour intersects a boundary component of $\mathcal{S}$, the chart is refined into one 2-strip and one 1-strip (Figure 4(c)).

In the second case, the 1-strip is a quadrilateral patch whose boundaries alternately composed of pieces of contours and flow paths. This kind of 1-strip does not contain any critical point in its interior and differs from the cap 1-strip because the value of f is not constant on its boundary. If we refine a region whose boundaries are parts of flow paths that partially overlap, then we again obtain either two 2 strips or one 1-strip and one 2-strip. In the latter case, the 1-strip is a triangular region bounded by two portions of flow paths without overlapping and the inserted contour.

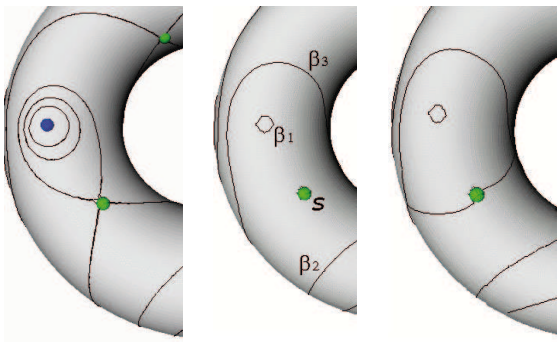
Figure 5 shows three iterations on the same model of the refinement of the charts. In this example, the function f is the harmonic map with the maxima of Gaussian curvature as Dirichlet boundary conditions. Note that not all patches are subdivided: in fact, the refinement criteria depend on the approximation error.

3.4. Degenerate cases

Degenerate case are usually associated to a mapping function f that is not simple and not Morse. If the function f is not simple, then there exist some critical points that have the same value of f. While this fact is not particularly significant for minima and maxima, if two or more saddles share the same value $f(\mathbf{s})$, then the corresponding level set induces a segmentation into a complex strip. This strip is no longer a 3-strip but a $(2 + l)$-strip, where l represents the number of saddles having the same value $f(\mathbf{s})$. However, these $(2 + l)$-strips may be split into $(l + 1)$ 2-strips by applying the technique proposed in Section 3.2. Figure 6 shows two saddles having the same value of f and how the corresponding 4-strip is split into three 2-splits.

If f is not Morse for the presence of m-fold saddles, the procedure described in [22] can be used to split the m-fold saddle into m simple saddles. The corresponding chart is split into $(m + 1)$ 2-strips by following all the flow paths that connect the saddle to the boundary component β. Finally, note that our technique allows us to split a 3-strip in case the flow paths do not intersect transversally; i.e., the scalar function is Morse but not Morse-Smale; see for example the result on a function with a *strangulation* in Figure 7.

Figure 6: A 4-strip, with two saddles having the same value of f, is split into three 2-strips. To ease the visualization, the function f is the height function.



(a) (b) (c)

Figure 7: (a) Level sets, (b) 3-strip, and (c) its split into two 2-strips.

3.5. Computational cost

In the worst case, the combinatorial complexity of the Reeb graph extraction is $O(n \log n)$, where n is the number of vertices of $\mathcal{M}$; efficient algorithms for its computation were proposed in [17, 39]. Denoting $|E|$ the number of edges of the Reeb graph $\mathcal{R}_\mathcal{G}$, $\partial\mathcal{S}$ is computed by inserting $|E|$ contours in $\mathcal{M}$ with $O(|E|n)$ operations. During this phase, the complexity of the model increases with the possible insertion of new vertices that belong to contours in $\partial\mathcal{S}$. Since each 3-strip split operation acts only on a single chart $\mathcal{S}_i$, it takes $O(|\mathcal{S}_i|)$ operations, where $|\mathcal{S}_i|$ denotes the number of elements of $\mathcal{S}_i$. Assuming that the insertion of $\partial\mathcal{S}_i$ into $\mathcal{M}$ adds w new elements to $\mathcal{M}$, the overall cost of the 3-strip splits is $O(n + w)$, which is $O(|E|n)$. Therefore, the combinatorial complexity of the initial shape chartification is $O(\max(n \log n, |E|n))$.

For every step of the adaptive refinement, the operations needed to split a single chart are limited by the number of elements of the chart even if the refinement might involve all charts. Since each chart is split into at most two new charts, the spatial complexity of the charts increases and the number of elements of the single chart is at most duplicated. In summary, the worst case complexity of the adaptive chart refinement after k steps is $O(2^k|E|n)$. However, even if the spatial complexity of the model increases with the insertion of the contours, we have experimented that the number of steps required to obtain a satisfactory chartification is generally lower than seven.

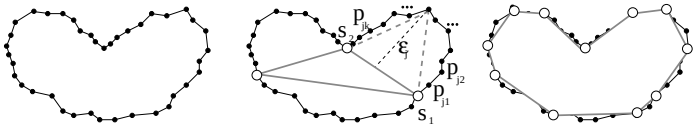


Figure 8: Iterative approximation of a contour. The white circles highlight the contour samples.

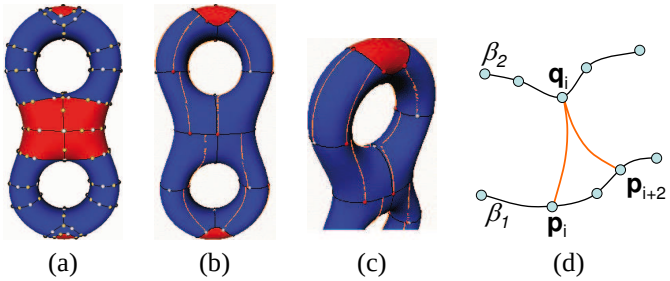


Figure 9: (a) Contour sampling and (b,c) some correspondences among boundary components. The 1-strips are colored in red while blue regions represent 2-strips. Red points represent the samples the closest to the correspondence. (d) Example of correspondences that flow towards the same point and generate a generalized triangular region.

4. Surface approximation

This section describes the procedure that allowed us to use the chart refinement as a surface approximation technique, which also enables simple yet effective surface coding. Intuitively, the approximation is devised to work as a surface reconstruction from non-planar and non-parallel contours, where the contours correspond now to the boundaries of the patches. We proceed first by approximating the boundaries of the charts by a sampling process. Then, we extract correspondences between vertices of the contours to be tiled to improve the quality of the reconstruction and minimize self-intersections and twisting effects. In this process, each chart is tiled separately, and all the approximated patches are concatenated so as to form the final surface. Since the configurations of the boundaries of our charts do not include branching cases, the approximation strategy is easily turned into a surface coding and decoding process. Then, the decoding works as an automatic tiling of the chart boundaries, or contours, coded appropriately with their implicit reconstruction rules.

Contour sampling. During this step, the boundaries of each patch are approximated using a progressive sampling, which stops when the local approximation error is below a given threshold. We proceed by first sampling the boundary of the 1-strip caps, and then sampling the other kind of strips. Given a boundary component β of a k-strip $\mathcal{S}$, $1 \leq k \leq 2$, since β is shared by another strip $\mathcal{S}_1$, we distinguish between two situations: (i) β has not yet been sampled and (ii) β was sampled for approximating $\mathcal{S}_1$. In the first case, the initial sampling of the boundary β consists of three points that are regularly spaced over the contour. In the second case, we count the number of existing samples (note that $\mathcal{S}$ and $\mathcal{S}_1$ may share only a portion of a boundary component), and start to approximate β by using the existing samples (i.e., those induced by the approximation of $\mathcal{S}_1$) until we identify the required three points. If the initial sampling already has more than three points, these points are set as the initial boundary sampling.

At each step, we add new samples to the boundary β until the approximation error ϵ falls below a given threshold t_1, or it becomes stationary (ceasing to change significantly). Then, the error induced over the segment $(\mathbf{s}_1, \mathbf{s}_2)$ that approximates the portion $(\mathbf{s}_1, \mathbf{p}_{j_1}, \ldots, \mathbf{p}_{j_k}, \mathbf{s}_2)$ of β is computed as $\epsilon = \max_{i=1,\ldots,k}\{|d((\mathbf{s}_1, \mathbf{s}_2), \mathbf{p}_{j_i})|\}$, where d is the distance in space between a segment and a point. In practice, t_1 is specified by the user as a percentage error of the lengths of β, for example, 0.02% means that $t_1 = 0.02|\beta|$, where $|\beta|$ is the length of the curve β (Figures 8 and 9(a)).

Correspondences between boundary components. Given a strip $\mathcal{S}$ with two boundary components β_1 and β_2, we determine a set of correspondences among points of β_1 and β_2, so-called *attachments*. The attachment $(\mathbf{p}_i, \mathbf{q}_i)$, $\mathbf{p}_i \in \beta_1$ and $\mathbf{q}_i \in \beta_2$ is extracted by computing the flow path γ_i that flows from $\mathbf{p}_i$ to β_2. During the first iterations, the correspondences are computed for all points in the initial sampling of the contours. Due to the sparseness of the sampling at the first stage, the flow paths drawn from points in β_1 will not flow, in general, to a point belonging to the samples of β_2. Therefore, the attachment will be defined by selecting the sample $\mathbf{q}_i$ of β_2 that is the nearest to γ_i. For this reason or because two flow paths merge, it is possible that more than one point on β_2 is attached to the same point in β_1.

The union of all the attachments between every pair of contours β_1 and β_2 forms a set of guiding lines for the tiling, meaning that they are constraints for approximating the strip $\mathcal{S}$. Their role is to prevent self-intersecting triangles as much as possible. The attachments are re-computed every time the strip is refined. In our experiments, every boundary component is initially sampled with three points, which are uniformly sampled. However, the bigger the number of attachments between β_1 and β_2 is, the more accurate the reconstruction of the flow paths in the approximation is. Figures 9(b,c) show some examples of contour correspondence between the boundary components of 2-strips: orange lines represent the flow paths that connect points on β_1 and β_2.

Contour tiling. At each step of the chart refinement, the resulting approximation is matched against the required error. In order to compute this error, the approximation is applied locally. This process resembles the reconstruction of a surface from non-planar and non-parallel contours, where the contours correspond now to the boundaries of the patches, sampled as explained above. Therefore, we linearly and locally approximate $\mathcal{M}$ through the creation of a triangle mesh between the sampling of the contours enclosing each chart.

Concerning the tiling phase, triangulating a single non-planar contour was used in the context of repairing defective mesh descriptions (by means of triangulating the gaps in the meshes) or as a tool in the reconstruction of an object from a series of cross sections. Some methods, e.g. [10, 33], triangulate the

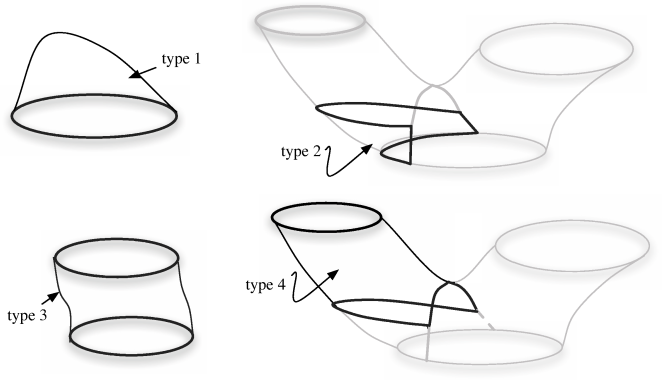


Figure 10: The four types of charts.

contour in a greedy manner, while other methods, e.g., [4, 5], seek a triangulation optimal with respect to some merit function, e.g., the total area of the triangles. In the following, we use any of the classical contour-to-contour tiling algorithms (e.g., [15, 25, 26, 34]), bearing the remote chance to obtain self-intersecting tiling in cases in which the two original contours are extremely twisted and different in shape. For more details on the best triangulation of non-planar contours, we refer the reader to [4, 10].

We recall that the charts can be only of the following types (Figure 10).

- *Type 1*: patches with only one boundary component that corresponds to an iso-contour of f. In this case, the surface is approximated by a generalized cone having the inner critical point as the apex and the boundary sampling as the base.
- *Type 2*: patches with only one boundary component composed by iso-contour segments alternating with flow paths. In general, the patch is approximated by a generalized open cylinder. Since the boundary of $\mathcal{S}$ alternates pieces of contours and flow paths, we connect the two portions of contours as in the tiling algorithm, and close the strip with the vertices that belong to the flow paths. If the flow paths merge, then the region becomes a triangoloid, as discussed in Section 3.3. In this case, we connect the points of the iso-contour to those ones in the flow paths and create an open cone that has the first point of the merge between the flow paths as vertex.
- *Type 3, 4*: patches with two boundary components. Each boundary of a patch of type 3 corresponds to an iso-contour of f. Otherwise (type 4), one boundary corresponds to an iso-contour and the other one is composed by iso-contour segments alternating with flow paths. In both cases, the surface is approximated by a generalized closed cylinder: we invoke any classical algorithm (e.g., [15, 25, 26, 34]) for tiling between two contours, subject to the constraints imposed by the attachments.

Starting from any attachment between two contour samplings, the triangulation algorithm tiles β_1 and β_2 locally, adopting any suitable edge optimization criterion, until another attachment is reached or β_1 and β_2 are completely traversed. In our approach, the contours are progressively visited according to the method described in [5]. More precisely, denoting $\mathbf{u}$ and $\mathbf{v}$ the two current vertices on β_1 and β_2, a new triangle $t := (\mathbf{u}, \mathbf{v}, \mathbf{w})$ is added if it satisfies an optimization criterion given as a combination of the following criteria: edge length; edge dihedral angle; minimal area; and percentage of visited contour. In the current implementation, the weights of these criteria are chosen as 0.5, 0.25, 0.125, and 0.125 (which sum up to 1), and normalized so that they provide values in the unit interval. Figures 11(c,d) shows an example of this method.

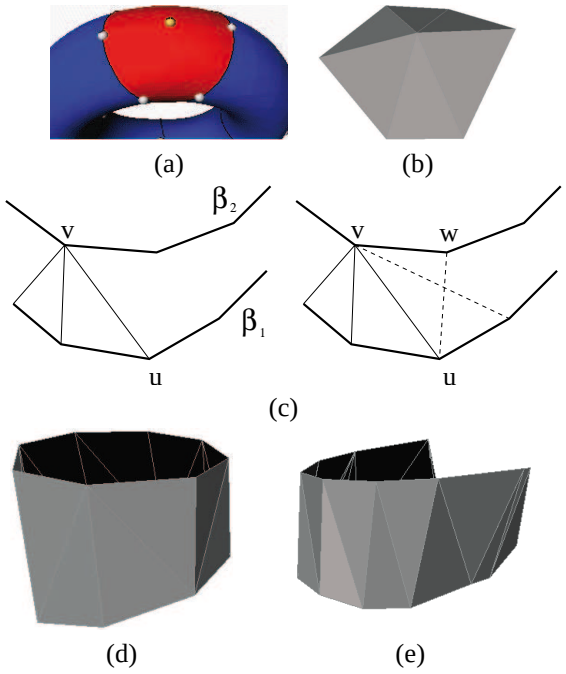


Figure 11: Reconstruction of a pseudo-cone: (a) sampling on the original model and (b) approximated patch. (c) Tiling algorithm and (d) reconstruction of a 2-strip. (e) Tiling of a 1-strip whose boundary is made up of segments of two contours and two flow paths. This case corresponds to the regions depicted in Figure 4(c). .

Error evaluation. The symmetric Hausdorff distance $d_{\mathcal{H}}(\mathcal{S}, \mathcal{S}') := \max\{d(\mathcal{S}, \mathcal{S}'), d(\mathcal{S}', \mathcal{S})\}$, with $d(\mathcal{S}, \mathcal{S}') := \sup_{\mathbf{p}\in\mathcal{S}}\{\inf_{\mathbf{q}\in\mathcal{S}'} \|\mathbf{p} - \mathbf{q}\|_2\}$ measures the error between a chart $\mathcal{S}$ on $\mathcal{M}$ and its approximation $\mathcal{S}'$. This error is evaluated by computing the k-nearest neighbor graph $\mathcal{T}$ of the vertices of $\mathcal{S}$ in $O(|\mathcal{S}| \log |\mathcal{S}|)$ time, and using $\mathcal{T}$ to compute $d_{\mathcal{H}}(\mathcal{S}, \mathcal{S}')$ for each updated approximation $\mathcal{S}'$. In case $d_{\mathcal{H}}(\mathcal{S}, \mathcal{S}')$ is greater than a user-defined threshold t_2, the patch $\mathcal{S}$ is refined until $d(\mathcal{S}, \mathcal{S}')$ becomes smaller than t_2.

Approximation coding and decoding. All the information (contour samples, attachments between strip boundary components, and tiling rules) needed to compute the approximated surface is opportunely stored in a file. First of all, we list the Cartesian coordinates of the contour samples and the critical points of f: generally, these data are largely less than the original vertex count. Then, every chart is stored in the file as a set of two

or four lines each coding the sequence of the indices of samples that form the chart boundary, interleaved with the symbols ";" that indicate attachments and "." to denote the end of the boundary.

We distinguish between three tiling rules, which also implicitly encode the rules themselves.

1. *Rule 1*: The first line of the chart coding contains only the indices of the vertex corresponding to the inner critical point; the second line contains the sequence of indices to the vertices of its boundary samples.
2. *Rule 2*: The chart is stored using four lines, the first and third lines contains the indices of the samples of the boundary segments lying on the contours, while the second and fourth lines code the boundary segment on the flow paths. The sequences of indices may be interleaved with the attachment indicators. If the boundary segments on the flow paths or on the contours do not contain any sample, the corresponding lines are left empty.
3. *Rule 3*: The first and second lines of the chart coding contain the indices of the samples of the two boundary components β_1 and β_2; the sequences of indices may be interleaved with the attachment indicators. The two lines contain the same number of attachments and in case a point on one contour is attached to more points on the other contour, the symbol ";" is repeated as many times as the points.

The orientation of the boundaries of each chart $\mathcal{S}$ is stored consistently with the original surface $\mathcal{M}$. Choosing a *type 1* strip $\mathcal{S}$ and indicating with $\mathbf{m}$ the critical point contained in $\mathcal{S}$, we orient its boundary β in a counterclockwise or clockwise manner if $f(\beta) < f(\mathbf{m})$ or $f(\beta) > f(\mathbf{m})$, respectively. In case $\mathcal{S}$ is a *type 3 and 4* strip with boundary components β_1 and β_2, such that $f(\beta_1) < f(\beta_2)$, we code β_1 (resp., β_2) in a counterclockwise (resp., clockwise) orientation. Similarly, we order in counterclockwise or clockwise manner the portions of boundary of a *type 2* strip lying on contours while the flow paths are ordered according to the increasing values of f.

The role of the decoder is to load the data and to reconstruct the tiles according to the rules embedded in the file. Once the tiles have been built, they are glued together in a unique mesh $\mathcal{M}'$ by identifying the edges that are shared by two patches. Once the surface has been reconstructed, a Laplacian fairing [35] and a curvature optimization along the edges may be applied to obtain a smoother continuity along the boundary patches and guarantee that $\mathcal{M}'$ interpolates all contours $\partial\mathcal{S}$ with $\mathcal{G}^1$ regularity.

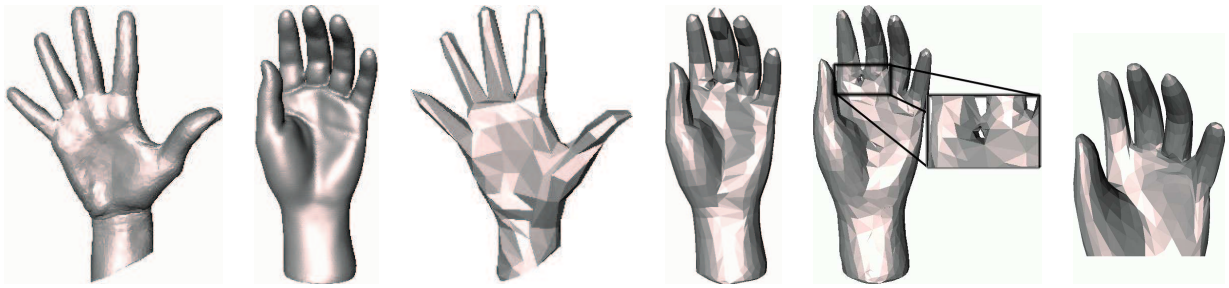
5. Examples and discussion

We tested our method on several models with different types of features and scalar functions. In general, we notice that the number of vertices of $\mathcal{M}'$ is quite independent of the number of elements of $\mathcal{M}$ while it is influenced by the number of shape characteristics, i.e., number of critical points of f. In the experiments, the sampling threshold t_1 varies on the contours between 0.01 and 0.05, and the threshold t_2 for the patches is a percentage of the diagonal of the bounding box of $\mathcal{M}$. This implies that both the contour sampling and the error evaluation are independent of the original tesselation. In Figure 12(c), a first rough yet effective approximation of the original model in Figure 12(a) is obtained using only 254 vertices and a harmonic map whose Dirichlet boundary conditions are placed on the finger tips. Similarly, the model in Figure 12(d) approximates with 1287 vertices the model in Figure 12(b).

	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{M}'\rvert$	$d_{\mathcal{H}}$	Time (seconds) $\mathcal{R}_{\mathcal{G}}$	*1-step*	*all*	*size %*
	12,286	762	0.033	0.029	0.032	0.273	4.2
	9,111	620	0.039	0.087	0.034	0.289	4.6
	1,515	254	0.035	0.035	0.028	0.13	10.8
	1,515	571	0.027	0.035	0.029	0.28	26.2
	5,510	647	0.027	0.047	0.037	0.55	7.8
	136,650	723	0.032	0.857	0.62	3.76	0.3
	26,789	680	0.034	0.209	0.17	0.71	1.7
	26,789	1,287	0.029	0.209	0.17	1.298	3.2
	26,358	1,243	0.031	0.31	0.15	1.42	3.15
	45,705	1,238	0.086	0.08	0.12	1.281	1.8
	10,005	885	0.031	0.078	0.13	1.29	5.9
	41,160	1,816	0.054	0.129	0.095	0.78	3
	34,025	4,461	0.016	0.315	0.74	5.296	8.7
	91,863	10,448	0.062	0.62	0.92	16.3	7.8
	31,994	7,932	0.018	0.29	0.22	3.78	17.3
	12,386	3,966	0.014	0.11	0.13	0.883	22.4
	35,474	8,477	0.013	0.32	0.27	4.86	16.4
	5,201	958	0.035	0.045	0.029	1.776	12.3

Table 1: Statistics of the approximation. The Hausdorff error $d_{\mathcal{H}}$ is expressed as the percentage of the diagonal of the bounding box of the original model. Time is evaluated on an AMD Atlon, 64 Processor 3500+ having 2GB of RAM. The rightmost column represents the size of our coding system with respect to the original size of the triangle mesh (in VRML 1.0 format).

Table 1 shows the performance of our method over a set of models; in particular, we emphasize that the achieved simplification is almost independent of the number of vertices of the original mesh. In particular, Figure 13 represents the evolution of the maximum error $d_{\mathcal{H}}$ over all the approximated charts when the strips are adaptively refined with thresholds $t_1 = 0.07$ and $t_2 = 0.01$. The time of the decoding phase for the camel



(a) (b) (c) (d) (e) (f)

Figure 12: (a,b) Input and (c,d) reconstructed models of two hand models using harmonic maps, whose Dirichlet boundary conditions are placed on the finger tips. Reconstructed model (e) with and (f) without a tiny handle.

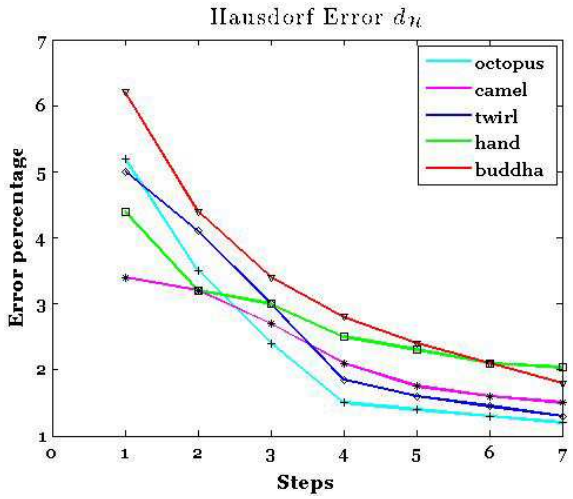
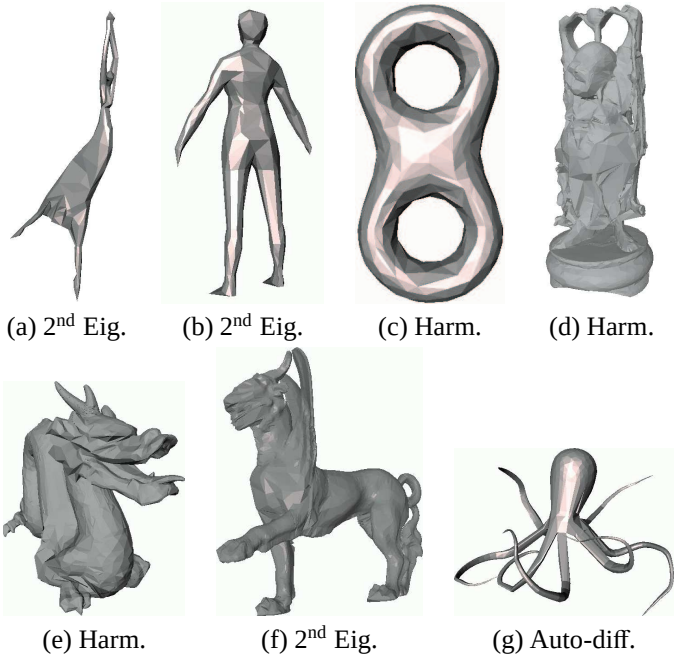


Figure 13: Error evolution with respect to the strip refinements. Considering only the initial chartification, the reconstructed octopus, camel, twirl, hand, and Buddha models contain 393, 208, 127, 254, and 700 vertices, respectively. After seven steps the models are respectively made of 6732, 7028, 1776, 3123, and 10356 vertices.

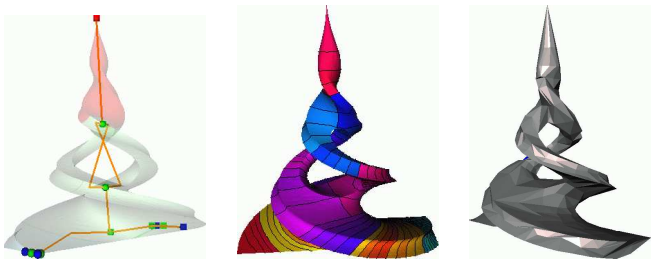


(a) 2nd Eig. (b) 2nd Eig. (c) Harm. (d) Harm.

(e) Harm. (f) 2nd Eig. (g) Auto-diff.

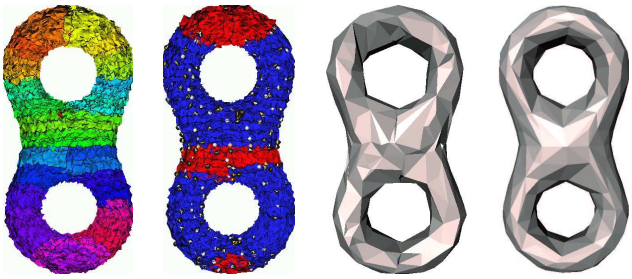
Figure 14: (a,b) Surfaces approximated using two refinement steps. (c) Reconstruction of the bitorus model using three refinement steps. Approximation of the (d) Buddha, (e) dragon, (f) feline, and (g) octopus models after five iterations.

model varies from 0.0013 seconds (208 vertices) to 0.057 seconds (7028 vertices), while for the Buddha model it varies from 0.012 seconds (700 vertices) to 0.085 seconds (10,356 vertices). The approximation rate obtained in our experiments is comparable with the analogous test shown in [47], see also the approximation of the feline model in Figure 14(f). Finally, Figure 15 depicts the Reeb graph of a twisted model obtained with respect to the first Laplace-Beltrami eigenfunction and the shape approximation that we obtained after four refinements of the strips. In this case, as shown in the last row of Table 1 the reconstruction error is 0.035. As shown in Figure 12(e), our contouring detects and preserves the tiny handles of the original model. However, tiny handles can be easily detected and discarded in the construction of the approximation thanks again to the use of the Reeb graph. This is guaranteed by the use of the Reeb graph as driving topological structure for the chartification: the Reeb graph indeed preserves the topology of the original surface. However, the reconstruction process is not guaranteed to be self-intersection free and watertight, even if the refinement process highly reduces the occurrence of these artifacts as demonstrated by the examples. An a posteriori polishing of the reconstructed mesh would be the best way to handle these situations. If the measure of the handle, which is defined as the persistence of the corresponding minimum non-separating graph cut of the Reeb graph $\mathcal{R}_\mathcal{G}$, is smaller than a user-chosen threshold, we apply a topology simplification strategy to $\mathcal{R}_\mathcal{G}$ [52]. In this latter case, our system provides a simplified model (in term of through holes) by discarding the contours related to the loops in $\mathcal{R}_\mathcal{G}$ that we want to eliminate. For instance, in Figure 12(f) we have reconstructed the triangle mesh without considering the contours related to the saddles on the small handle of the hand model. In this example, the approximated shape $|\mathcal{M}'|$ has 786 vertices and the error with respect to the original model is 0.08. Our approach is robust with respect to perturbations of the vertex coordinates when the scalar function f is robust to noise. The chartification of the noisy surface in Figure 16(a) was obtained in two refinement steps fixing $t_1 = 0.05$ and $t_2 = 10\%$. Using a harmonic func-



(a) (b) (c)

Figure 15: (a) Reeb graph with respect to the first non-trivial Laplacian eigenfunction, (b) strips after four refinements and (c) shape approximation with 958 vertices.



(a) (b) (c) (d)

Figure 16: (a) Chartification of a noisy model induced by a harmonic map and after two iterations; (b) sampled points. Reconstruction of (c) the input shape with 247 vertices and (d) the analogous result on the original model.

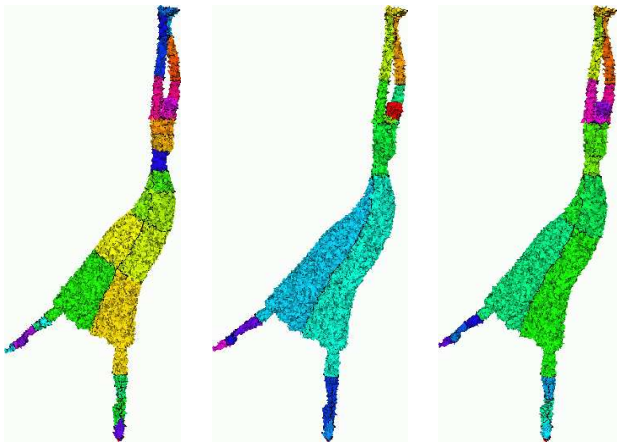
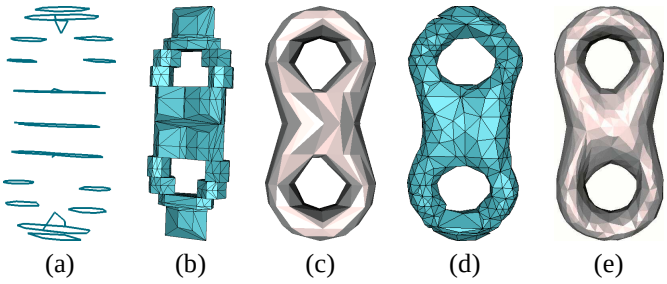


Figure 17: Stability of the chartification against noise; the corresponding chartifications on the smooth surface are shown in Figure 5.



(a) (b) (c) (d) (e)

Figure 18: (a) Isocontours and flow paths, (b) reconstructed model using [35] with 458 vertices, (c) our reconstruction with 192 vertices. (d) Surface refinement and smoothing using [35] with 536 vertices and (e) our surface refinement and smoothing with 873 vertices.

tion, the number and types of charts remain stable in spite of the perturbations to the surface geometry. Finally, the reconstructed model (c) was obtained by decoding the points selected during the coding; differently from the approximated model in (d) obtained from the original bitorus model with comparable thresholds, we note that the perturbation of vertices marginally affects the reconstruction. Also, the chartifications in Figures 17 are obtained with a harmonic function computed on a noisy model, while Figure 5 depict their analogues on the original model.

The previous examples show that the spirit of our chartification resembles the quadrilateral remeshing discussed in [20, 21] even in the presence of the substantial difference that we admit 2-strips. In fact, the boundaries of our charts are composed of iso-contours and flow paths of the input f. In [21] the boundaries are instead the iso-contours of f and the flow paths of a scalar function whose gradient is orthogonal to ∇f. In [20], the patch boundaries are the flow paths of f but they do not include the corresponding level sets. Furthermore, using only one scalar function, instead of two orthogonal maps, provides a relation between the number of patches and the critical points of f. Combining level sets and flow paths provides a number of patches which is lower than that of [20].

Our tiling method is able to approximate shapes by using the level sets of generic functions, therefore it is not limited to the use of planar contours. In this sense, our method generalizes the work proposed in [35] that is able to process only planar contours. In this latter case, both the iso-contours of f and flow paths lie on planes, and therefore it is possible to reconstruct the shape by applying the Delaunay triangulation on the plane as described [35]. A comparison of the two techniques, therefore, can be done using our framework with f as the height function, which produces planar contours. An example of reconstruction using our method and [35] is shown in Figure 18.

Choice of the scalar function f. The scheme used to compute the chartification can be applied to any scalar function, thus providing different options that mainly depend on the number of critical points and their significance for shape characterization [6, 7]. If the scalar function f is not given as input, then we can select f among several classes of functions, each of them providing an approximation that emphasizes different characteristics of the input shape. For instance, if we aim at approximating the surface with a minimal number of patches, then we adopt a harmonic map with a minimal number of critical points (i.e., 1 minimum, 1 maximum, and $2g$ saddles) [38]. The harmonic maps may be also forced to explicitly code geometric feature points (e.g., vertices with high Gaussian curvature) by selecting them as Dirichlet boundary conditions [19, 24, 43]. In a more general setting, geometric features are well identified by Laplacian eigenfunctions [20, 50], and the auto-diffusion maps [27]. In fact, these functions automatically locate the tips of the shape features as maxima and minima and provide an initial chartification with smooth boundaries and a low number of patches.

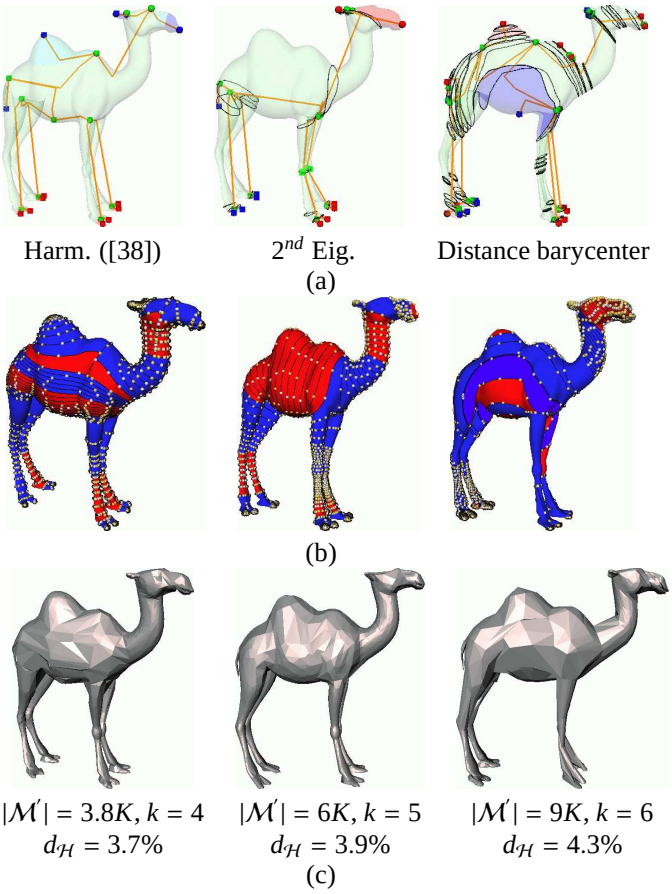


Figure 19: (a) Reeb graphs, (b) contour samples, and (c) final reconstruction of the same model with respect to (from left to right) a harmonic map with Dirichlet boundary conditions placed at the maxima of the Gaussian curvature, the first non-trivial Laplacian eigenfunction, and the Euclidean distance from the barycenter. Here, $|\mathcal{M}'|$, k, and $d_{\mathcal{H}}$ represent the number of vertices of $\mathcal{M}'$, the refinement steps, and approximation error, respectively.

Figure 19 *(last row)* shows the reconstruction of the same surface using three different scalar functions. In all these cases, the reconstruction error is lower than the 4.5% of the maximal diagonal of the bounding box. We observe that the approximation using the Euclidean distance from the barycenter *(rightmost column)* converges more slowly than the harmonic map *(leftmost column)* and the second Laplacian eigenfunction *(middle column)*. In particular, this happens on the camel body and the anterior legs, where the shape is almost spherical and sparsely sampled. The choice of this Laplacian eigenfunction correctly recognizes the protrusions of the model and forgets only the camel's hump: in this case, we need one more refinement step to achieve an approximation rate similar to the one provided by the harmonic map.

Since a high number of irrelevant critical points would make the framework less effective in terms of compression, noisy scalar functions are smoothed by removing critical points with low persistence values. Any approach for canceling pairs of critical points can be used [13, 23]. An alternative solution is to work in the function space and apply isotropic Laplacian filters [20, 38] or bilateral smoothing operators to the function itself [36], eventually with constraints [40]. Even for geometric noise, the number and configuration of the charts depends only on the smoothness of the function f used to decompose $\mathcal{M}$.

Despite the generality of the choice of the function f, in our experiments we have found that the harmonics maps, eventually selecting points of high Gaussian curvature as Dirichlet boundary conditions, the first Laplacian eigenfunctions and the auto-diffusion maps usually give better results. Although this calls for further investigation, these functions generally provide a shape chartification suitable for shape approximation because they are able to identify significant shape characteristics.

6. Conclusions and future work

This paper has proposed a method to approximate the shape of a 3D object surface $\mathcal{M}$ that readily defines a shape coding and decoding mechanism. The method is based on a shape chartification and reconstruction from the level sets and flow paths of a scalar function defined on $\mathcal{M}$. In practice, the encoding provided by our approach is efficient (memory-wise) and accurate, thus allowing the approximate reconstruction of the original object with a relatively small error. We highlight that our approximation approach is driven by error and not vertex count and, differently from [35] where the topology of the reconstructed surface is sensitive to changes in the configuration of the cross-sections, our use of the Reeb graph during the coding process enables us to dominate the topology of the shape approximation. Moreover, one strength of our chartification is the fact that the contours of the initial chartification are placed far from critical points (which can be unstable).

The proposed approach is suitable for progressive simplification and reconstruction: in fact, the reconstruction accuracy is directly guided by the tolerances used to approximate the boundaries of the charts. Future work mainly concerns the optimization of the details needed for the encoding and decoding, so that the method can be fully extended to a lossy compression technique. Also, we are evaluating the adoption of a tiling strategy that builds directly on the computation of all the attachments for all the vertices during thes sampling of the contours.

Acknowledgments

The authors would like to thank all the members of the Shape Modeling Group at CNR-IMATI for their valuable support. Special thanks are also given to the anonymous reviewers for their comments and suggestions.

This work has been partially supported by the FOCUS K3D Coordination Action, EU Contract ICT-2007.4.2 contract number 214993, the Italian National Project SHALOM funded by the Italian Ministry of Research under contract number RBIN04HWR8 and the CNR research activities (ICT-P10.007 and ICT-P10.009).

References

[1] C.L. Bajaj, E.J. Coyle, and K.-N. Lin. Arbitrary topology shape reconstruction from planar cross sections. *Graphical Models and Image Processing*, 58(6):524–543, 1996.

[2] T. F. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *American Mathematical Monthly*, 77:475–485, 1970.

[3] G. Barequet, M.T. Goodrich, A. Levi-Steiner, and D. Steiner. Contour interpolation by straight skeletons. *Graphical Models*, 66(4):245–260, 2004.

[4] G. Barequet and M. Sharir. Filling gaps in the boundary of a polyhedron. *Comput. Aided Geom. Des.*, 12(2):207–229, 1995.

[5] G. Barequet and M. Sharir. Piecewise-linear interpolation between polygonal slices. *Computer Vision and Image Understanding*, 63(2):251–272, 1996.

[6] S. Biasotti, L. De Floriani, B. Falcidieno, P. Frosini, D. Giorgi, C. Landi, L. Papaleo, and M. Spagnuolo. Describing shapes by geometrical-topological properties of real functions. *ACM Computing Surveys*, 40(4):1–87, 2008.

[7] S. Biasotti, B. Falcidieno, P. Frosini, D. Giorgi, C. Landi, G. Patanè, and M. Spagnuolo. 3D shape description and matching based on properties of real functions. In *Eurographics 2007 Tutorial Proc.*, pages 949–998, 2007.

[8] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(1-3):5 – 22, 2008.

[9] S. Biasotti, M. Mortara, and M. Spagnuolo. Surface compression and reconstruction using **R**eeb graphs and shape analysis. In *Spring Conference on Computer Graphics*, pages 174–185, Bratislava, 2000.

[10] J. H. Bohn and Wozny M.J. Automatic CAD-model repair: shell-closure. In *Proc. Solid Free form Fabrication Symposium*. Dept. of Mech. Eng., Univ. of Texas at Austin, 1993.

[11] J.-D. Boissonnat and P. Memari. Shape reconstruction from unorganized cross-sections. In *Proc. of Symposium on Geometry Processing*, pages 89–98. Eurographics Association, 2007.

[12] D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. *ACM Trans. on Graphics*, 28(3):1–10, 2009.

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

[14] Y.-K. Choi and K.H. Park. A heuristic triangulation algorithm for multiple planar contours using an extended double branching procedure. *The Visual Computer*, 10(7):372–387, 1994.

[15] H.N. Christiansen and T.W. Sederberg. Conversion of complex contour line definitions into polygonal element mosaics. *Computer Graphics*, 13:187–192, 1978.

[16] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. In *ACM Siggraph*, pages 905–914, 2004.

[17] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. In *Proc. of the Symposium on Computational Geometry*, pages 344–350, 2003.

[18] M. Desbrun, A. N. Hirani, and J. E. Marsden. Discrete exterior calculus for variational problems in computer vision and graphics. In *Proc. of Decision and Control, 2003*, volume 5, pages 4902–4907, Dec. 2003.

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

[20] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Spectral surface quadrangulation. *ACM Trans. on Graphics*, 25(3):1057–1066, 2006.

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

[22] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry*, 30:87–107, 2003.

[23] H. Edelsbrunner, D. Morozov, and V. Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *Proc. of the Symposium on Computational Geometry*, pages 127–134. ACM, 2006.

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

[25] H. Fuchs, Z.M. Kedem, and S.P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, 1977.

[26] S. Ganapathy and T.G. Dennehy. A new general triangulation method for planar contours. *ACM SIGGRAPH*, 16(3):69–75, 1982.

[27] K. Gebal, J. Andreas Bærentzen, H. Aanæs, and R. Larsen. Shape analysis using the auto diffusion function. *Computer Graphics Forum*, 28(5):1405–1413, 2009.

[28] X. Gu, S. J. Gortler, and H. Hoppe. Geometry images. *ACM Trans. on Graphics*, 21(3):355–361, 2002.

[29] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proc. of Symp. on Geometry Processing*, pages 127–137, 2003.

[30] J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, and H. Bao. Spectral quadrangulation with orientation and alignment control. *ACM Trans. on Graphics*, 27(5):1–9, 2008.

[31] J.-D.Boissonnat. Shape reconstruction from planar cross sections. *Computer Vision, Graphics, and Image Processing*, 44(1):1–29, 1988.

[32] D. Julius, V. Kraevoy, and A. Sheffer. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum*, 24(3):581–590, 2005.

[33] M. Kel and A. Dolenc. Some efficient procedures for correcting triangulated models. *Proc. Solid Free form Fabrication Symposium, Dept. of Mech. Eng., Univ. of Texas at Austin*, 1993.

[34] E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research and Development*, 19(1):2–11, 1975.

[35] L. Liu, C. Bajaj, J. Deasy and. D. A. Low, and T. Ju. Surface reconstruction from non-parallel curve networks. *Comput. Graph. Forum*, 27(2):155–163, 2008.

[36] Rong Liu and Hao Zhang. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum*, 26:385–394, 2007.

[37] A. Mebarki, P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. In *Visualization*, pages 479–486, 2005.

[38] X. Ni, M. Garland, and J. C. Hart. Fair Morse functions for extracting the topological structure of a surface mesh. *ACM ToG*, 23(3):613–622, 2004.

[39] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust online computation of reeb graphs: simplicity and speed. *ACM Trans. on Graphics*, 26(3):58–67, 2007.

[40] G. Patané and B. Falcidieno. Computing smooth approximations of scalar functions with constraints. *Computer & Graphics*, 33(33):399–413, 2009.

[41] 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.

[42] G. Patanè, M. Spagnuolo, and B. Falcidieno. A minimal contouring approach to the computation of the Reeb Graph. *IEEE Trans. on Visualization and Computer Graphics*, 2009.

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

[44] N. Ray, W.C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Trans. on Graphics*, 25(4):1460–1485, 2006.

[45] G. Reeb. Sur les points singulièrs d'une forme de Pfaff complètement intégrable ou d'une fonction numèrique. *Comptes Rendu Hebdomadaires des Séances de l'Académie des Sciences*, 222:847–849, 1946.

[46] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 37(6):1539–1556, 2008.

[47] O. Sorkine, D. Cohen-Or, D. Irony, and S. Toledo. Geometry-aware bases for shape approximation. *IEEE Trans. on Visualization and Computer Graphics*, 11(2):171–180, 2005.

[48] J. Tierny, J.P. Vandeborre, and M. Daoudi. Enhancing 3D mesh topological skeletons with discrete contour constrictions. *The Visual Computer*, 24(3):155–172, March 2008.

[49] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Designing quadrangulations with discrete harmonic forms. In *Proc. of Symposium on Geometry Processing*, pages 201–210, 2006.

[50] B. Vallet and B. Levy. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum 27(2)*, 2008.

[51] Y. F. Wang and J. K. Aggarwal. Surface reconstruction and representation of 3-D scenes. *Pattern Recognition*, 19(3):197 – 207, 1986.

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

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