Topology-driven Shape Chartification

Tommaso Sorgente[a], Silvia Biasotti[a], Marco Livesu[a], Michela Spagnuolo[a]

[a]*Istituto di Matematica Applicata e Tecnologie Informatiche - Consiglio Nazionale delle Ricerche, Genova, Italy*

Abstract

We propose a novel algorithm to decompose a 3D object into an atlas of disk-like charts. Decomposition into charts with controlled shape and topology is relevant in many engineering areas, such as spline fitting, compression and re-meshing. We produce our chartifications by jointly exploiting the Reeb graph of a guiding function and its gradient aligned flow paths. The key advancements of our method with respect to similar approaches are: (i) a novel strategy to provably remove all T-junctions; (ii) a stable system to trace flow paths starting far from critical points; (iii) the exploitation of the regularity of certain functions under isometries (e.g., harmonic ones) to produce structurally equivalent chartifications for families of objects posed differently. The charts produced by our system can be of two types: topological quads and topological octagons. Both of them can be easily gridded to produce full quadrilateral meshes, as we demonstrate in the second part of the article.

Keywords: Shape analysis, shape chartification, Morse theory.

1. Introduction

Shape chartification is the process of partitioning an arbitrary surface into a set of charts having simpler topology and geometry [52]. As demonstrated by recent research in the field [51, 10, 24, 23], in the CAD/CAE community it is often convenient to have charts that can be easily gridded, and also to make sure that the chartification does not contain T-junctions [36]. Furthermore, chartifications are beneficial in a whole variety of applications, including remeshing [39, 40, 47, 12], spline fitting [11], texturing [41, 50], compression [13], shape approximation [14] and fabrication [26].

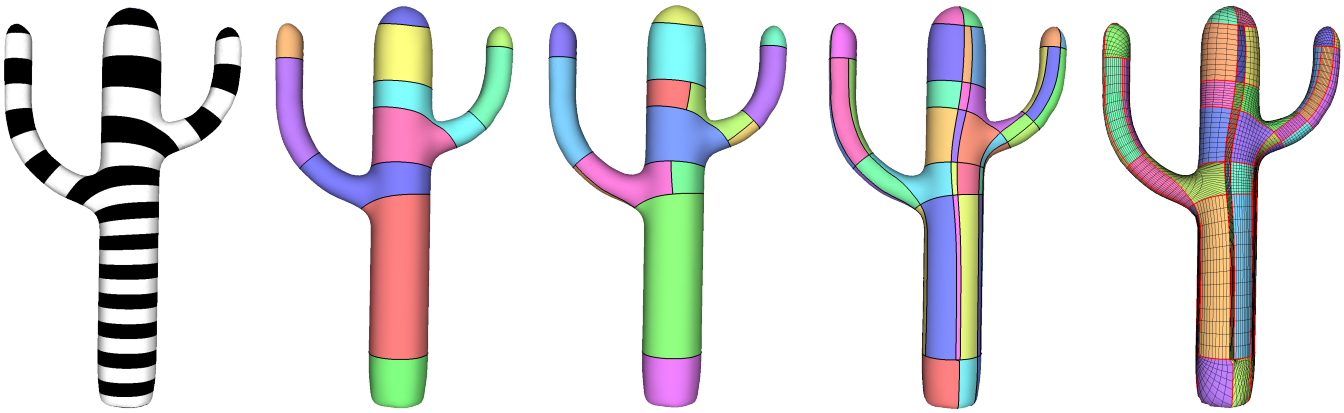
Our goal is to produce a chartification that remains consistent if the surface undergoes deformations that, at least to some extent, preserve geodesic distances (i.e. isometries). Moreover, we target a chartification that is T-junction free and keeps the valence of the chart vertices as regular as possible. The solution presented in this paper adopts a topology-driven approach whose ingredients are Reeb graphs and discrete gradient flow paths. The core novelty relies in the way they are combined together to produce the final decomposition.

The Reeb graph of a guiding scalar field is used to extract a coarse chartification of the surface, isolating cylindrical and saddle areas, and caps. These initial charts are then refined by tracing piece-wise linear curves aligned with the function gradient, ensuring the elimination of all T-junctions and the construction of only 4-sided and *rhombus* charts. Rombus charts are 8-sided charts enclosing saddles. The strategy adopted for refining the coarse Reeb atlas distributes the boundaries of the charts evenly with respect to the behaviour of the field over the surface, and sufficiently far away from the critical points to avoid instability of the chart boundary positioning, a known critical issue in previous similar approaches [20, 25]. An overview of the chartification approach is shown in Figure 1.

While this machinery is agnostic to the function being used and could potentially be adopted for any function which admits a Reeb Graph (i.e., Morse-Smale), the choice of functions which are invariant under isometries is the key to ensure consistent chartifications across different poses of the same shape. There is practical evidence that harmonic functions exhibit a consistent behaviour when boundary conditions are well placed [2] (e.g., at the extrema of protuberances). In our experiments we mostly rely on harmonic functions, producing consistent decompositions for shapes belonging to the same class. For completeness, in Section 6 we show a few chartifications obtained with alternative functions.

Methods that adopt similar chartification approaches have been presented in literature, but either they are restricted to a small class of shapes (e.g., tubular [50]) or suffer from presence of T-junctions or instability in the quad layout (see Section 2 for a detailed discussion). The main achievements of our method with respect to existing ones can be summarized as follows:

Email addresses: tommysorgente@gmail.com (Tommaso Sorgente), silvia.biasotti@ge.imati.cnr.it (Silvia Biasotti), marco.livesu@ge.imati.cnr.it (Marco Livesu), michela.spagnuolo@ge.imati.cnr.it (Michela Spagnuolo)



Harmonic field | (a) Coarse Reeb chartification | (b) Rhombus charts | (c) Final chartification | (d) Quadrilateral remeshing

Figure 1: Our chartification algorithm in a nutshell. Left: we start from a triangle mesh and a guiding function; middle left: we extract the Reeb regions (caps, saddles, cylinders); middle: we refine saddle charts, creating rhombus domains; middle right: we propagate and eventually delete the T-junctions generated at the previous step; right: we grid each domain to produce a quadrilateral remeshing of the input shape.

- *consistency across deformations*: the chartification is primarily guided by the topological structure induced by the critical points of a harmonic field defined over the surface, whose critical points are well behaving with respect to deformations of the surface and therefore ensuring consistency of the topological structure across isometric deformations (Figure 1);
- *stability of the chart structure*: we propose a method to trace separatrices of our coarse Reeb charts as flow paths of the underlying scalar function. The key point of our tracing system is that we start tracing them far from critical points (Figure 1(a-b). Other approaches, for instance [20, 25], trace separatrices starting from saddle points. We observe that this may lead to unstable behaviour, because the gradient is not defined in critical points and, due to discretization issues, is also usually unstable nearby;
- *T-junction free chartification*: our surface charts are topological quads with no T-junctions. The most recent approach that is able to obtain such a result [48] deletes T-junctions by using a geometric greedy stitching algorithm that is not always able to remove all of the them. Our T-junction removal system is more general and robust, and always guarantees a T-junction free chartification (Figure 1(c).

We demonstrate our chartification algorithm in the context of quadrilateral remeshing (Figure 1(d)). In Section 5 we discuss a simple yet effective method to map both 4- and 8-sided charts onto proper quadrangular domains, producing a quadrilateral tessellation of the input surface. By exploiting the regularity of harmonic functions we also show how to generate consistent quadrilateral meshes of similar shapes having different discretization and no cross-parameterization. Differently from the recently published method [1] which produces quad meshes with *similar* structure, our topological approach produces *exactly* the same structure (same number of singular vertices and separatrices).

2. Previous work

The chartification process is composed by an earlier decomposition and (optionally) a per chart parameterization. General segmentation algorithms [44] may not fulfill all the necessary requirements. Typical requirements are the generation of charts with disk-like topology [39] (e.g. for texturing), or the generation of charts where all the charts are topological quads. The latter are often called *quad layouts* [7] and play a fundamental role in quadrilateral remeshing [47, 50, 12] and spline fitting [36], where the tensor product structure of the domains is exploited.

2.1. Function-driven Decompositions

More relevant to our work are chartification processes driven by some continuous function defined on the boundary of a three-dimensional shape.

The use of a scalar function to drive a mesh tesselation has been addressed in [20, 19] and further optimized in [25]. In [20] the patch boundaries are generated by both by iso-contours of the function and orthogonal lines (in practice authors combine iso-contours of the function f with the flow paths of a scalar function whose gradient is orthogonal to ∇f). The approach is further developed in [19]; there the authors proposed to choose the eigenfunctions of the Laplace operator as possible scalar functions. In this case the same method generated a family of quadrangulations, one per each eigenfunction. In this latter approach, the use of one scalar function, instead of two orthogonal maps, directly relates the number of quadrangles and the

number of critical points of f. The approach in [19] has been further extended in [25] to provide explicit controls of the orientation and alignment of the quad elements. In this kind of methods the patch boundaries are generally made by the flow paths that cross in the critical points of the eigenfunction (the corners of the quadrangles). Since an eigenfunction can be regarded as a periodic function, its period is prescribed by choosing an appropriate eigenvalue. However, there is no guarantee that the period determined in this way is compatible with the orientation and alignment control. In practice, where the flow paths do not intersect transversally (i.e. the eigenfunction is not Morse-Smale) several adjustments are necessary to deal with particular cases (either partial or total overlaps of the paths, strangulations, etc.). Thus, the spectral quadrangulation method may fail to generate high quality quads when complex alignment controls are imposed. The idea of considering Morse-Smale complexes is further addressed in [53]; in this case instead of an eigenfunctions and its gradient field, the authors adopt the principal curvature directions [15]. The resulting tessellation is anisotropic, well aligned with the principal curvatures but might fails with the tessellation of handles if the input mesh contains edge whose length is larger than the size of these handles.

A small number of articles are addressing the problem of generating tessellations with large tiles. By reducing the number of tiles, the needs of a topological guide is necessary, and the existing methods are exploiting Morse functions to address this challenge. Branch *et al.* [9] used Morse-Smale complexes, by connecting critical points using flow paths). Lu *et al.* [33] described a method to build a quadrangulation with the aim to drive the number of tiles only by the topology of the mesh, with good combinatorial properties, *e.g.* the degree of each vertex of the quadrangulation is four or five. On the contrary, the n-loop framework [21] handles the question of tessellating a surface with large quadrangles, adjusting the location of the paths with respect to the geometry, but with the limitation that paths are edge-based, and the computation it is not computationally fast (being not driven by a scalar function).

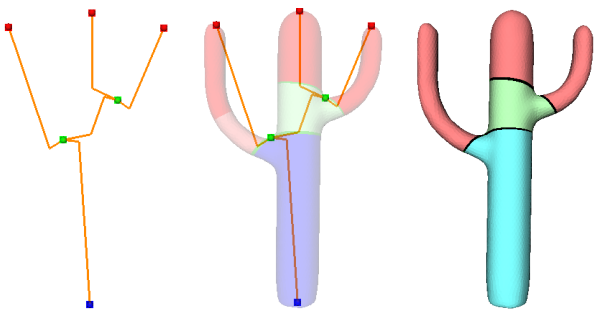
Tierny *et al.* [48] described a method to drive a quadrangulation using a scalar function, exploiting the associated Reeb graph to generate cylindrical and disc tiles, as a structure for a small quadrangulation. In their work, the scalar function is exploited as a parameter to adjust the quadrangulation and to avoid multiple intersections in correspondence of critical points or small slices among to critical level sets. Analogously, Bærentzen *et al.* [2] proposed a surface manifold partition into topologically disk-like and annular regions driven by a mesh-skeleton co-representation. Such a dual representation coupled the mesh vertices with a Reeb graph-like skeleton computed with respect to an harmonic function. Then, a template-driven quad refitting was applied to each region. The practical robustness of the harmonic function with respect to intrinsic model deformations made the skeleton suitable for character animation and sculpting. Similarly to [48] and [2], we also adopt the Reeb graph to drive an initial mesh subdivision, whose elements are guaranteed to be topological discs and cylinders. However, in our approach, discs are used to isolate critical points of the function, contours and flow paths are computed far from criticalities and cylinders are adopted to decompose the rest of the surface.

2.2. Quad Meshing

We review here quadrilateral meshing processes that start from a coarse decomposition of the input object into charts. We point the reader to [7] for a more comprehensive discussion on quad meshing and alternative techniques. The majority of algorithms start from a decomposition into quadrilateral charts [49, 25, 17, 48, 27] (possibly containing T-junctions) and employ a local, per-chart, parameterization to project vertices on the target surface. Notable exceptions to this rule are [45, 34], where quadrangulation starting from general polygonal decompositions is applied in the context of animation. When charts are topological quads, the so generated quadrilateral meshes will replicate the singular structure if the decomposition, having as irregular vertices (i.e. not valence four) all and only the points where more (or less) than four charts meet. The quadmesh connectivity is generated by gridding each chart. Attention must be paid to the transitions along boundaries shared between adjacent charts, especially in presence of T-junctions. Boundaries should be split in the same number of quads on both sides. This is a global problem that can be solved at the cost of two linear equations per chart, imposing that opposite boundaries of the same chart are be subdivided in the same number of quads [49, 8]. Usai and colleagues [50] observed that such a formulation would fail when multiple charts appear on both sides of the same boundary, and proposed a more general and efficient formulation. Notice that if charts contain a small number of boundaries (e.g. less than 7), they can be meshed right away by exploring the space of quad tilings [46], as done in [43]. Chart boundaries can also be relaxed, in order to better follow the geometry or align to sharp creases. Both [47, 50] illustrate how to extend the abstract domain technique [40] to the quadmesh case. In Section 5 we apply a combination of the techniques discussed in [49, 50] to produce quadrilateral meshes starting from our chartification process. Differently from them, our approach guarantees provably correct vertex-domain assignments (Section 7).

3. Theoretical background

The theoretical tools which allow us to build our chartification are Reeb graphs and flow paths. We recall in this Section the key definitions we will use later on.



(a) (b) (c)

Figure 2: Geometric representation of the Reeb graph of the cactus model (a), its overlay onto the model (b), and the corresponding decomposition (c) obtained using an harmonic function with boundary conditions set on the protrusion tips of a cactus model.

3.1. Reeb graph

The Reeb graph of a manifold M with respect to real-valued Morse function f, $\mathcal{R}_{\mathcal{G}}(M, f)$, 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 topology. The original definition of Reeb graph can be found in [42], while for detailed discussions of its properties and application we refer to [4, 3]. The Reeb graph definition requires f to be Morse and simple, that is, a function whose critical points are non degenerate and injective on the critical points. The Reeb graph has been widely used in the literature to analyse 3D shapes, and its extension to non-Morse and non-simple functions have been presented in [5]. The Reeb graph naturally induces a decomposition of M, or of its discrete representation $\mathcal{M}$, into connected regions, each corresponding to an arc the simplicial complex. Figure 2 shows an example of the relation between the Reeb graph and the corresponding surface decomposition into regions. In this work, we base our method on the algorithm for the extraction of the Extended Reeb Graph, as described in [6], where we detailed how to deal with degenerate and non-simple saddles.

3.2. Gradients and integral lines on a triangle mesh

Gradients and integral lines are the main ingredients to build our chartification. We assume M to be represented by a triangle mesh $\mathcal{M}$, and f such as to guarantee discrete differentiability conditions, that is, for any edge (v_i, v_j) of $\mathcal{M}$ we assume $f(v_i) \neq f(v_j)$.

Gradient. In literature there are several methods to estimate gradient vector fields [18] on each mesh element (i.e., vertices, edges or triangles). Being the function defined on vertices and linearly interpolated within each triangle, the gradient field is piece-wise constant. To compute per-triangle gradients we rely on the discretization proposed in [6], which straightforwardly applies the definition of the gradient of a function of being perpendicular to its level sets. Alternative definitions, such as the one derived from the Green-Gauss method [30] could be used instead. More in detail, we define the gradient of f on a triangle $t(v_i, v_j, v_k)$, $\nabla f|_t$, as the solution of the 3×3 linear system:

$$\begin{pmatrix} v_j - v_i \\ v_k - v_j \\ \mathbf{n}_t \end{pmatrix} \nabla f|_t = \begin{pmatrix} f(v_j) - f(v_i) \\ f(v_k) - f(v_j) \\ 0 \end{pmatrix}. \qquad (1)$$

where $\mathbf{n}_t$ is the unit length normal of t. In practice, we impose that the projection of the gradient of f on the plane induced by a triangle is equal to the vector $(f(v_j) - f(v_i), f(v_k) - f(v_j), 0)$. When necessary, we approximate the gradient at the vertices and edges of the mesh as the weighted average of all the gradients of the triangles that are incident to an element (2 triangles per edge, $n \geq 3$ triangles per vertex). Weights are proportional to the area of the corresponding triangles.

Integral lines. They allow us to move on the manifold following the gradient direction. Formally, an integral line $\gamma : \mathbb{R} \to M$ of f is defined as the maximal path on M whose velocity vectors, or tangent vectors, agree with the gradient of f, meaning that $\frac{\partial \gamma}{\partial s} = \nabla f(\gamma(s))$ for all s in $\mathbb{R}$. Each integral line is open at both ends, having its origin (i.e., $\lim_{s \to -\infty} \gamma(s)$) and its destination (i.e., $\lim_{s \to +\infty} \gamma(s)$) at critical points of f [37].

It can be shown that integral lines are pairwise disjoint, that is, if their images share a point, then they are the same line. The images of integral lines cover the whole M, but if we consider the integral lines associated to the critical points of f, their images define a partition of M. This partition decomposes M into regions of uniform flow. Indeed the concepts of the *descending manifold* and the *ascending manifold* of a critical point p are introduced respectively as the set $D(p)$ of points that flow towards p or the set $A(p)$ of points that originate from p. When the function f is a *Morse-Smale* function over M, an ascending 1-manifold intersects a descending 1-manifold at exactly one point [37]. This condition is also known as transversality and it is a stable and generic condition, which is independent of small perturbations of the function f and M.

On a triangle mesh, moving according to ∇f means following the flows defined as the steepest ascending or descending paths with respect to the gradient. Therefore, flow paths are meant to approximate for meshes the concept of *integral lines* and a flow path φ is a piecewise linear curve over the mesh faces constituted by a list of nodes that we find intersecting the gradient vector with the edges of the mesh. From a critical saddle s, at least four flow paths start (more if the saddle is degenerate), two ascending and two descending paths. For this reason, for each saddle, we follow the first four (or more) steepest ascending and descending edges incident in it as the first edge of an flow paths moving from that. Note that flow paths on meshes never cross, but can merge (for instance, due

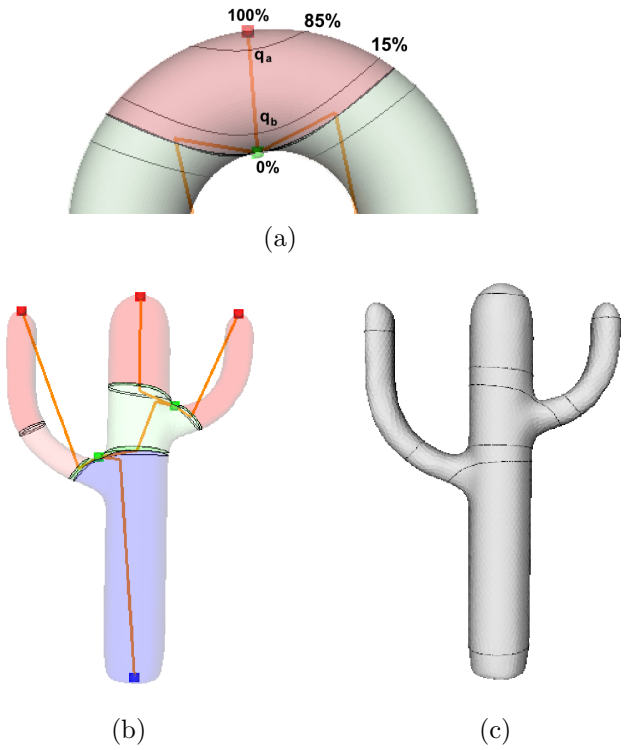


Figure 3: (a): Details of the self-adaptive chartification with a=15 and b=85. (b) Reeb graph and (c) contour-based decomposition of a cactus model.

to numerical approximations). In any case, once merged they cannot separate any more.

4. Shape chartification

The chartification process works in three steps. Given $\mathcal{R}_{\mathcal{G}}(M, f)$, we start with the initial coarse chartification induced by the Reeb graph on $\mathcal{M}$, using the algorithm [6]. This initial subdivision separates cylindrical areas from saddle areas and caps, that is, the regions containing the maxima and minima of f (Figure 1a). In the second step, a rhombus chart for each saddle point is generated (Figure 1b). The chartification is finalized by removing all the T-junctions generated at the second step, producing a chartification containing only 4- and 8-sided regions, and having no T-junctions (Figure 1c). In the remainder of the section we will detail each of these three steps.

4.1. Coarse Reeb Chartification

Based on $\mathcal{R}_{\mathcal{G}}(\mathcal{M}, f)$, we extract a coarse chart decomposition by cutting the surface along the isocontours of f, in the range spanned by arcs. Given the arc e of $\mathcal{R}_{\mathcal{G}}(\mathcal{M}, f)$ and given a value $f_a \in \mathrm{Im}(f)$, we call a *contour* the curve γ_a defined by

$$\gamma_a = f^{-1}(f_a) \cap \mathcal{M}_{|e},$$

where $\mathcal{M}_{|e}$ is the region of $\mathcal{M}$ corresponding to the arc e (see Fig. 4).

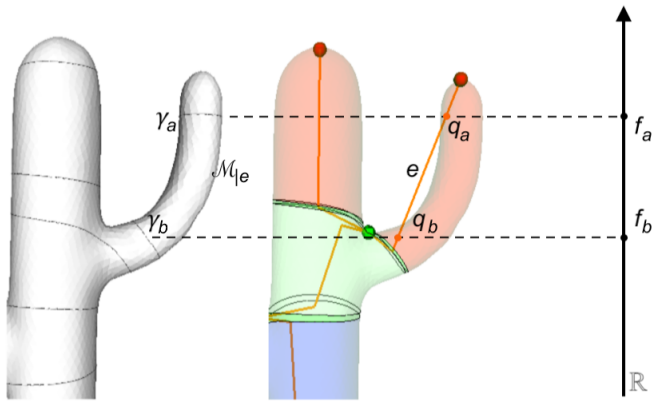


Figure 4: Contours on a cactus model.

We may consider the process of drawing contours as cutting the arcs of $\mathcal{R}_{\mathcal{G}}(\mathcal{M}, f)$, which is equivalent to changing the size of the Reeb charts and the position of their boundaries, while keeping their adjacency consistent with the Reeb graph.

Our goal is to produce a chart decomposition whose boundaries are far enough from the critical points of f, and possibly with uniform size. To reach this goal, a key point is the choice of the real values f_a and f_b, which are used to trace the iso-contours γ_a, γ_b (Figure 4). Indeed, they control the distance between the contours and the critical points and, therefore, the size of the charts.

We noticed that, especially for harmonic functions, the variation of f tends to concentrate around maxima and minima. This means that if we drive the cutting process by a uniform split of the image of e, the contours could localize quite close to critical points and could split the area spanned by e into two patches with unbalanced vertex distribution. Instead, we use an adaptive process. Given an arc of the $\mathcal{R}_{\mathcal{G}}(\mathcal{M}, f)$, whose nodes are two critical points v_1 and v_2, we split the range $(f(v_1), f(v_2))$ of $\mathrm{Im}(f)$ in intervals of size δ defined as follows:

$$\delta = \frac{|f(v_2) - f(v_1)|}{n},$$

and consider $f_a = f(v_1) + \delta\, i_a$, with

$$i_a = \min_{j \in \{1,..,n\}} \{\#\{v \in \mathcal{M}_{|e} \mid f(v) \leq j\delta\} \geq a\% \text{ of } \#\mathcal{M}_{|e}\}$$

where $\#A$ is the cardinality of the set A. The choice of i_a corresponds to the minimum number of intervals of $Im(e)$ that guarantees the interval $[f(v_1), \delta i_a] \in Im(e)$ contains an appropriate number of mesh vertices (the appropriateness is driven by the parameter a). We proceed similarly for f_b, i_b. In all our tests we set $n = 30$. Note that a and b are free parameters that we can tweak to control the size of the charts. In simpler words, for each region corresponding to an arc e, we count the total number of vertices belonging to the region and, moving away from the contours corresponding to nodes of e, we insert

two contours containing respectively the $a\%$ and $b\%$ of the vertices (Figure 3). In this way, we defined a method which adapts not only to the variation of f but also to the distribution of points on the mesh.

4.2. Rhombus charts

The first step has produced charts that nicely cover the areas identified by ERG nodes. The next important step is the construction of well behaved patches around saddle points, which are handled by the rhombus charts. Rhombus charts are special 8-sided charts built around saddle vertices and bounded by four flow paths and three iso-contours .

To describe the generation of a rhombus chart we refer to the example in Figure 5. Starting from a saddle point s, we follow the two flow paths opposed to the orientation of the saddle, until we reach the border of another chart. This operation identifies two points, p_1, p_2, located at opposite sides of the same iso-contour. Let us now focus on point p_1 (the same process applies to p_2). To define two lateral boundaries of the rhombus chart we move laterally from p_1 along its iso-contour, finding two points a, b. From a and b we follow the flow paths up to the higher contours, finding the points q_a and q_b. Applying the same process to p_2, we define the other side of the rhombus chart.

The shape and size of the chart depends on the distance between p_1 and a, b. To fix this degree of freedom, we center a sphere at p_1, using as radius the value $r = ld$, where l is the average edge length of the mesh and d is a parameter exposed to the user to control the chart size. In all our experiments we used $d \in [1, 10]$.

For each rhombus chart, eight new T-junctions are generated. In the subsequent step of the algorithm we will remove them, producing the output chartification composed of only 4- and 8-sided charts, without any T-junction.

4.3. T-junctions removal

We detail here how we propagate and eventually delete the eight T-junctions per rhombus chart introduced at the previous step, producing the final chartification.

We start from the observation that every corner of a rhombus chart is defined as the intersection between an iso-contour of f and a flow path. For each such vertex, we therefore continue tracking the steepest ascending (or descending) direction until we hit the boundary of a cap region (i.e. a region containing either a maximum or a minimum of f). By definition, flow paths will converge to one of these regions and will never intersect to each other, therefore no new T-junctions will appear during this process. In practice, due to numerical errors, merging between two adjacent paths may occur. Should this be the case, we use the parameter d (which determines the distance between adjacent flow paths) to impose a bigger distance between such paths, avoid intersections.

At this point, the chartification is composed of as many rhombus charts as the number of saddle points in f, as many caps as the number of maxima and minima in f, and all 4-sided charts in between (Figure 6). In order to complete the chartification process we need to refine the caps, also deleting the T-junctions arose along their boundaries during the aforementioned propagation process.

For caps having only two T-junctions generated by two incoming flow path lines, we simply add two new vertices on the boundary of the chart, and trace two flow paths starting from them. By definition, these paths will terminate in another cap region, and never in a saddle region.

After this refinement step, we process all the cap charts having even number of sides (at this point guaranteed to be more than two), and we split the cap into topological quads. To do so, we consider all the vertices at the boundary of the chart and alternatively connect them with the critical point centered at the cap (see Figure 7 for an example).

The algorithm described so far can provably provide a chartification for any case in which cap regions with odd number of sides do not occur. In case such caps arise, a valid chartification can always be produced by refining the chartification with a step of Catmull-Clark subdivision (i.e. halving all the chart sides and adding one new vertex per chart). Nevertheless, such refinement was never necessary in our experiments. To the best of our understanding, this unlucky configurations appear only when the function f does not observe the transversality condition of Morse-Smale complexes [37].

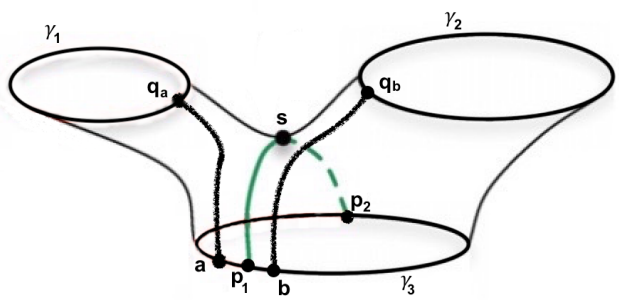


Figure 5: Construction of rhombus charts

5. Quad Meshing

The chartifications produced so far are characterized by a coarse yet simple structure almost entirely made of regular vertices (i.e. vertices having four incident charts). The only irregular vertices occur nearby local maxima and minima, where half of the integral lines used to suppress T-junctions produce valence three vertices, and the other half meet at a vertex with higher valence (Figure 7). This is a fundamental property to ensure quality meshes with well-shaped quads having angles close to 90° (Figure 8). In the remainder of the section we discuss how to process such a chartification to produce a dense quadrilateral mesh. The goal is to grid each patch, making sure that

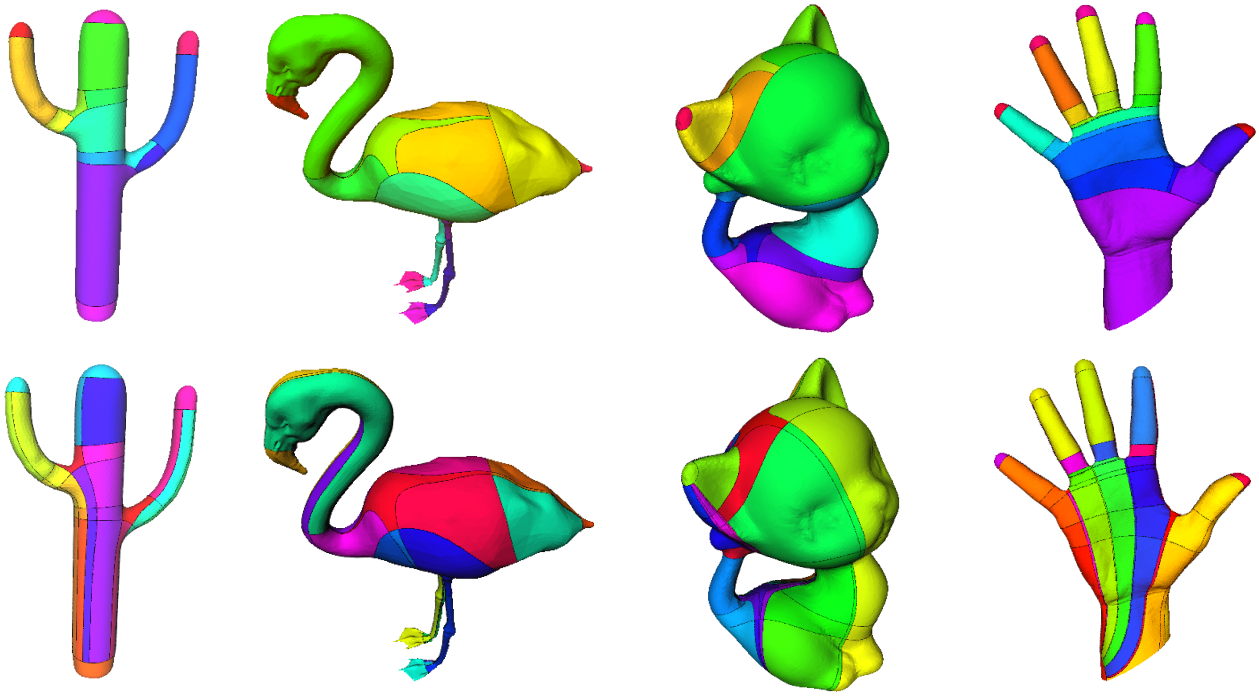


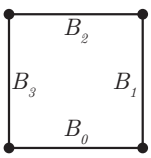
Figure 6: An intermediate step of our chartification process. Rhombus charts have been created, and the newly created T-junctions propagated along the flow lines until they reached the boundary of cap regions. Refining the caps will generate the final chartifications.

chart boundaries are tessellated with the same number of elements at both sides to achieve mesh conformity.

Similarly to previous approaches [50, 47], we proceed with a two steps approach: we first map each chart into a $m \times n$ parametric square; then, we use the integer iso-lines of the parametric space to design the mesh connectivity. We remind the reader that our decompositions is hybrid, as it contains 8-sided charts around saddle points, and 4-sided charts everywhere else. Previous methods do not support octagonal charts, thus cannot be used as-is. We extend [50, 47] by providing a novel map from 8-sided charts to quadrilateral parametric domains, and also by replacing their heuristic vertex/domain assignment with a provably robust intrinsic assignment that exploits our field-driven tracing system (Section 7).

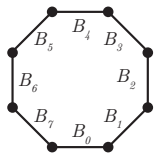
Map generation. Here we detail how to map each chart to the parametric space. We distinguish between maps that bring a quadrilateral chart to a $[0, m] \times [0, n]$ parametric square (Φ_{quad}) to maps that bring an octagonal chart to a $[0, a+b+c] \times [0, d]$ parametric square ($\Phi_{octagon}$). Maps are computed through the well known Tutte embedding, implemented using cotangent weights [35] to discretize the Laplace operator (Δ) so as to produce as-conformal-as-possible (i.e., angle preserving) maps. For a visual example of how these maps are realized, please refer to Figure 9.

Let B_0, B_1, B_2, B_3 be the counter clock-wise list of piece-wise linear boundaries of a given patch. We denote the length of the i-th boundary as $|B_i|$, and the length of the same boundary up to the point $p \in B_i$ as $|p|$. We realize the map Φ_{quad} and compute the uv coordinates of the parametric square by solving the following Laplace equation, subject to Dirichlet boundary conditions:



$$\begin{cases} u = 0 & \forall p \in B_0 \\ u = n & \forall p \in B_2 \\ u = n \cdot \frac{|p|}{|B_1|} & \forall p \in B_1 \\ u = n \cdot \frac{|p|}{|B_3|} & \forall p \in B_3 \\ \Delta u = 0 & \text{otherwise} \end{cases} \quad \begin{cases} v = 0 & \forall p \in B_1 \\ v = m & \forall p \in B_3 \\ v = m \cdot \frac{|p|}{|B_0|} & \forall p \in B_0 \\ v = m \cdot \frac{|p|}{|B_2|} & \forall p \in B_2 \\ \Delta v = 0 & \text{otherwise} \end{cases}$$

Similarly, given the ordered list $B_0, B_1, ..., B_7$ representing the piece-wise linear boundaries of an octagonal patch, we realize the map $\Phi_{octagon}$ and compute the uv coordinates of the parametric square as:



$$\begin{cases} u = 0 & \forall p \in B_0, B_1, B_7 \\ u = d & \forall p \in B_3, B_4, B_5 \\ u = d \cdot \frac{|p|}{|B_2|} & \forall p \in B_2 \\ u = d \cdot \frac{|p|}{|B_6|} & \forall p \in B_6 \\ \Delta u = 0 & \text{otherwise} \end{cases} \quad \begin{cases} v = 0 & \forall p \in B_6 \\ v = a + b + c & \forall p \in B_2 \\ v = a \cdot \frac{|p|}{|B_5|} & \forall p \in B_5 \\ v = a \cdot \frac{|p|}{|B_7|} & \forall p \in B_7 \\ v = a + b \cdot \frac{|p|}{|B_4|} & \forall p \in B_4 \\ v = a + b \cdot \frac{|p|}{|B_0|} & \forall p \in B_0 \\ v = a + b + c \cdot \frac{|p|}{|B_3|} & \forall p \in B_3 \\ v = a + b + c \cdot \frac{|p|}{|B_1|} & \forall p \in B_1 \\ \Delta v = 0 & \text{otherwise} \end{cases}$$

Notice that the so generated maps cannot be truly conformal, as we are constraining the boundary of the parametric domain to a fixed square, but in general will have little angle distortion and produce good tessellations (Figure 8, bottom).

Grid conformity. The map Φ_{quad} associated to each quadrilateral domain has two degrees of freedom (m, n) which control the integer size of the parametric space. Similarly, each octagonal map $\Phi_{octagon}$ has four degrees of freedom (a, b, c, d). To produce a *globally conforming* quadrilateral mesh (i.e. a mesh without T-junctions [36]) these values cannot be fixed locally, but rather ensure that side-adjacent patches sample the shared boundary with the same frequency. Several papers have shown that these degrees of freedom can be fixed by solving an integer linear programming problem, where each unknown represents the number of intervals in which a boundary will be split and is restricted to be strictly positive [31, 50, 8, 49]. We implemented the technique described in [50], which optimizes for as equiareal as possible quads. In Figure 8 we show various quadrilateral meshes computed with this technique.

6. Results

We implemented our chartification algorithm in C++, extending the library developed for [6] and using Eigen [22] and CinoLib [29] for numerics and geometry processing (harmonic functions, quad mesh generation). All tests were run on a MacBook Pro equipped with a 2,3 GHz Intel Core i7 processor on which we installed an Ubuntu 16.04 virtual machine with three CPU and 5GB of RAM dedicated. Running times vary from fractions of a second for moderate size models to a few seconds for high resolution models (see Table 1). In Figures 8 and 13 we showcase a number of decompositions produced with our method.

6.1. Driving functions

Our chartifications are fully driven by the underlying function f. We only require such function to be Morse-Smale. For all the results shown throughout the paper we used harmonic functions, obtained solving the Laplace problem $\triangle f = 0$ subject to manually prescribed Dirichlet boundary conditions on maxima and minima. In Figure 11

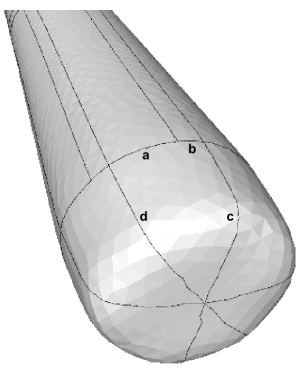


Figure 7: Type 1 minimum patch of a cactus turned into 4-sided charts.

Model	Size	t_{field}	t_{reeb}	t_{chart}	$t_{\mathsf{T-rem}}$	t_{tot}
Cactus	5K	0.04	0.06	0.07	0.15	0.29
Dancer #1	16K	0.10	0.19	0.25	0.5	0.98
Dancer #2	26K	0.13	0.33	0.43	0.74	1.56
Flamingo	26K	0.13	0.26	0.3	0.44	1.05
Hand (Fig 12)	180K	1.43	1.13	1.79	2.62	5.97
Hands (Fig 8)	14K	0.07	0.18	0.18	0.34	0.72
Horse	46K	0.25	0.47	0.66	1	2.22
Kitten	19K	0.19	0.21	0.27	0.36	0.89
Mug	11K	0.05	0.15	0.18	0.24	0.6
Rocker Arm	10K	0.07	0.12	0.15	0.23	0.52
Sphere	8K	0.08	0.14	0.15	0.19	0.5
Tori (Fig 11)	4K	0.03	0.06	0.10	0.12	0.29
Twirl	5K	0.04	0.05	0.07	0.13	0.31

Table 1: Performances of our method. For each model we report its size (number of vertices), the time necessary to produce the input driving field (t_{field}), and running times for each algorithmic step: generation of the ERG (t_{reeb}); initial chartification (t_{chart}); T-junctions removal ($t_{\mathsf{T-rem}}$); and total (t_{tot}). All times are expressed in seconds. The line corresponding to the hands in Figure 8 and the tori in Figure 11 report average data.

we show two chartifications obtained with alternative functions. Precisely, we used the height function (i.e. the per vertex z coordinates), and the bi-harmonic function (similar to the harmonic, but obtained solving $\triangle^2 f = 0$).

6.2. Structural properties

Our chartifications are characterized by a coarse yet simple structure almost entirely made of regular vertices. The very same structure is inherited by the subsequent quadrilateral meshes, of which the initial chartification represents the coarse layout [12]. The only irregular vertices occur nearby local maxima and minima, where half of the integral lines used to suppress T-junctions produce valence three vertices, and the other half meet at a vertex with higher valence (Figure 7). This is a fundamental property to ensure quality meshes with well-shaped quads having angles close to 90° (Figure 8). Gridding the 8-sided charts for quadrilateral remeshing introduces four additional valence three vertices per chart, located at the corners of the parametric space (Figure 9).

6.3. Structural coherency

An interesting outcome of our research is that we can exploit the ability of certain functions (e.g. the harmonic ones) to be stable under isometries, thus permitting the generation of topologically coherent chartifications of the same object in different poses. We demonstrate this property in Figure 8, where consistent chartifications of the same hand in four different poses are provided. Notice that each decomposition contains exactly the same number of quadrilateral and romboidal charts, and thus produced quadrilateral meshes that embed the same quad layout (i.e. number and connectivity between irregular vertices) [7]. To this end, we differ from the recently published [1], which produces quad meshes with only *similar* structure.

6.4. Computational cost

The combinatorial complexity of the Reeb graph extraction is $O(n \log n)$, where n is the number of vertices

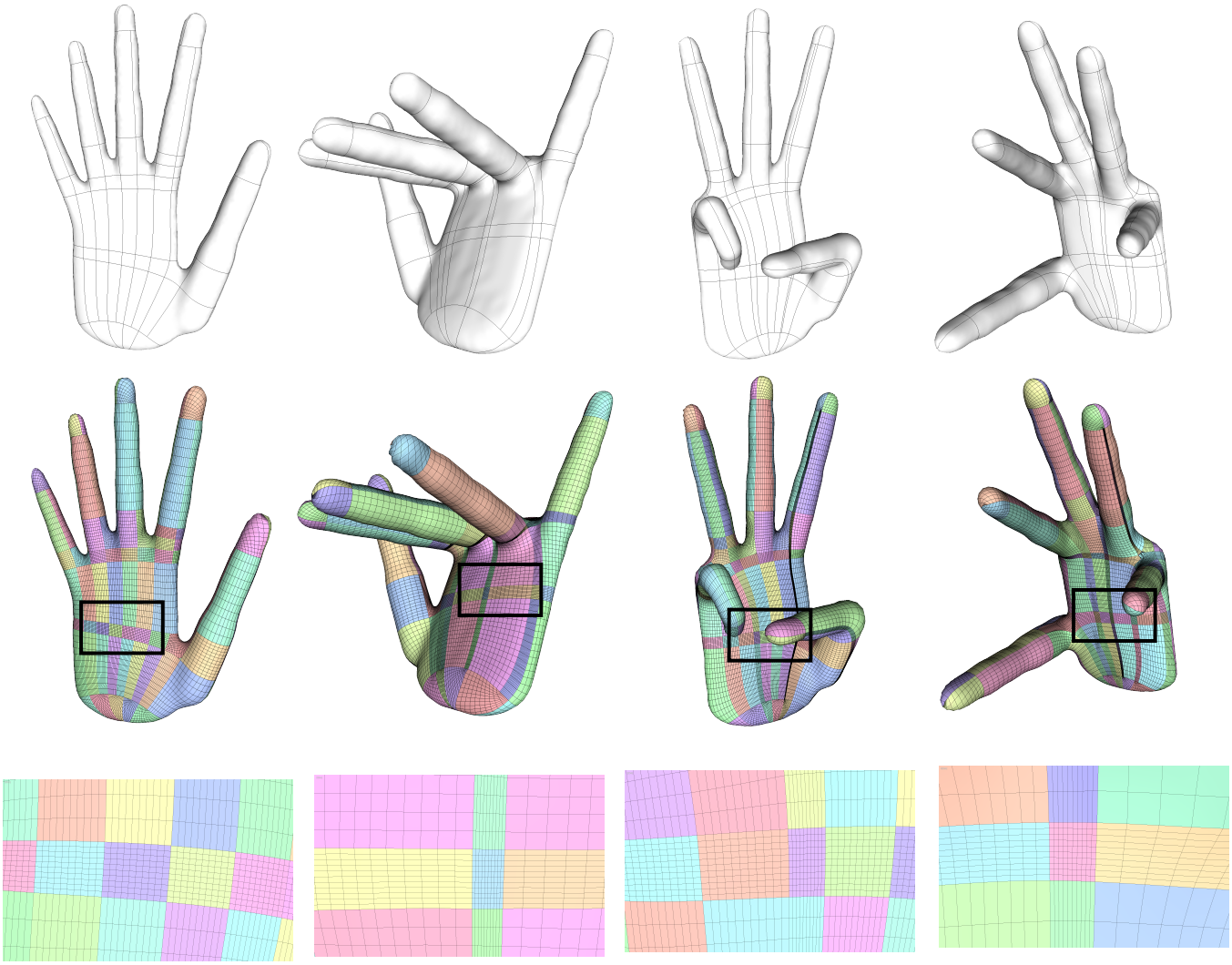


Figure 8: Top: four different poses for the hand model and their associated chartifications. All the meshes have different vertex count and connectivity. Middle: quadrilateral meshes obtained by gridding each domain in a conforming way (see Section 5 for details). Bottom: mesh close ups; our as-conforming-as-possible map generates well shaped quads with angles close to 90°.

of $\mathcal{M}$; efficient algorithms for its computation were proposed in [16, 38]. Denoting $|E|$ the number of edges of the Reeb graph $\mathcal{R}_{\mathcal{G}}$, the set of all the middle contours $\partial\mathcal{S}$ is computed by inserting $2|E|$ contours in $\mathcal{M}$ (two per arc) with $O(|E|n)$ operations. During this phase, the complexity of the model may increase with the insertion of new vertices that belong to contours in $\partial\mathcal{S}$. Since each rhombus definition 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 $\mathcal{S}_i$. If that the insertion of $\partial\mathcal{S}_i$ into $\mathcal{M}$ adds w new elements to $\mathcal{M}$, the overall cost of the rhombus charts is $O(n + w)$, which is $O(|E|n)$. Therefore, the combinatorial complexity of the essential shape chartification is $O(\max(n \log n, |E|n))$.

For the uniformation of the mesh, we need to connect every vertex of each rhombus chart to the closest maximum or minimum region. Again, only one region at time is involved, so it will be $O(|\mathcal{S}_i|)$ operations per each chart $\mathcal{S}_i$ that we cross. Being I the set of indices such that $\mathcal{S}_i$ is *not* a max/min region, we have $O(\sum_{i\in I} |\mathcal{S}_i|)$ operations for every vertex, and eight vertices per saddle. We can add to this calculation the cost of inserting two vertices and lines on the $k = 2$ max/min regions, being basically the same operation, hence getting a uniform mesh with four or more vertices in the max/min regions needs $O((8s + 2m) \cdot \sum_{i\in I} |\mathcal{S}_i|)$, with s the number of saddles and m the number of $k = 2$ max/min regions. Obviously, we will generally be summing in a much smaller set than I since we do not have to cross every region but just to follow the steepest ascending/descending direction.

T-junctions elimination is computed with $\frac{1}{2}(8s+2m) = 4s + m$ operations for every max/min region because we connect half of the vertices to the critical point. Together with the previous result, this gives $(8s+2m) \cdot \sum_{i\in I} |\mathcal{S}_i| + (4s + m) \cdot \sum_{i\notin I} |\mathcal{S}_i| = 6(2s + m) \cdot n$, so in conclusion we pass from the essential chartification to a coarse quad tes-

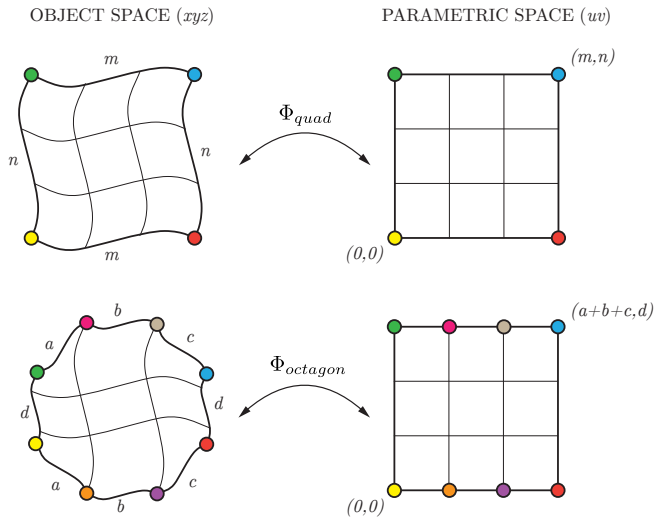


Figure 9: To produce dense quadrilateral meshes we map each patch into a $[0, m] \times [0, n]$ square, and use the integer isolines of the parametric space to design the quad mesh connectivity. Letters a, b, c, d, m, n represent the number of intervals in which each boundary will be split. Notice that deciding not to split any boundary leads to a 1×1 grid for the quadrilateral domain, and to a 3×1 grid for the octagonal domain.

sellation with $O((2s + m)n)$.

7. Comparisons

We compare here against the skeleton-driven coarse chartification algorithms presented in [50, 31]. As often happens there is no clear winner, but rather pros and cons for both methods. Skeleton-driven approaches give their best with tubular shapes that are well described by a curve-skeleton, producing extremely coarse chartifications with well-shaped domains. On the same class of shapes, our method is not able to be as coarse and regular, mainly because saddle points of the guiding function tend to arise where different branches of the skeleton meet, generating a number of long and tiny quadrilateral charts that emanate from each octagonal chart containing a saddle (Figure 12). On the other hand, our approach is far more general and poses almost no limitation on the class of shapes that can be chartified, including objects such as a mug, which could not be processed with [50, 31] because its axis is external with respect to the shape, and therefore it does not admit a skeletal representation [32].

Considering more technical aspects, we also observe that [50, 31] and similar approaches (e.g. [28]) project the domain decomposition on the surface with simple ray-casting. While this procedure performs well on nice and smooth shapes, it may easily fail on detailed or noisy shapes, producing wrong vertex/domain assignments or creating ill-defined domains with intersecting boundaries. Heuristics can often cure wrong vertex/domain assignments [47] but no guarantees of success can be provided. In our method, boundaries between adjacent domains are defined as integral curves of a scalar field. As a result, boundaries are traced directly *on* the surface of the object, and vertex/domain assignment is implicit. This allows us to provably produce a correct chartification and subsequent per chart parameterization, regardless the complexity of the shape being processed or the amount of surface details (Figure 13).

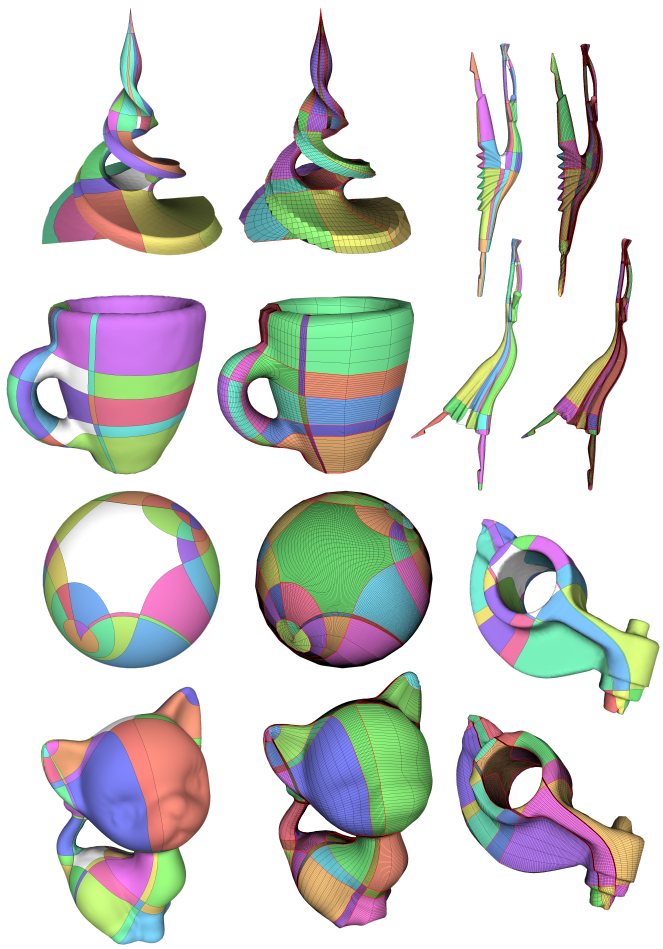


Figure 10: A mosaic of different chartifications and their associated quadrilateral remeshing. A variety of different scenarios have been considered, ranging from complex shapes embedded with simple functions (e.g., the twist at the top-left corner) to simple shapes and complex functions (e.g. the sphere in the middle).

8. Conclusions, limitations and future work

We introduced a novel topological method to decompose a 3D object into an atlas of 4- and 8-sided surface charts. These decompositions are important in a number of applications, ranging from reverse engineering to computer animation. The main contributions of our approach consist in two robust and easy to implement methods to trace chart boundaries that align to the gradient of a guiding field, and remove all T-junctions. While special purpose approaches (e.g. skeleton-driven chartification [50]) may

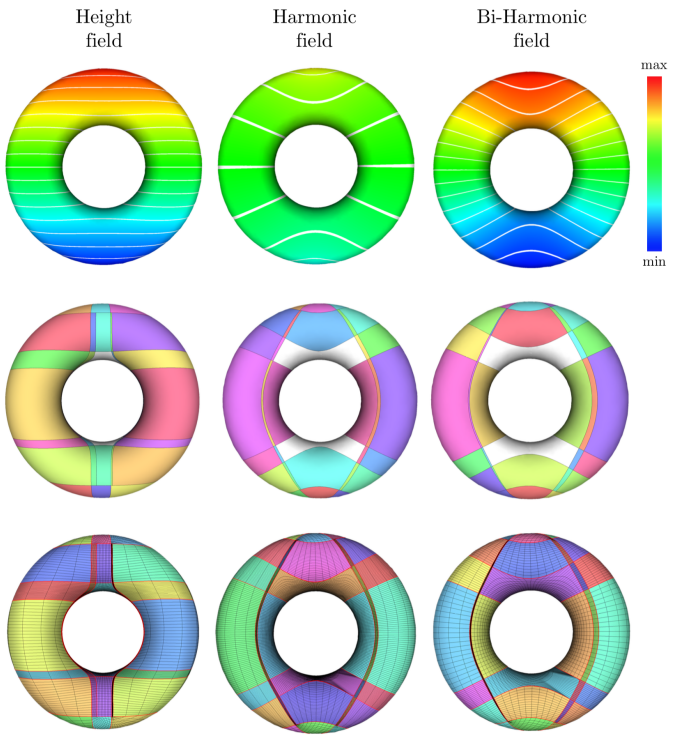


Figure 11: Our algorithm is agnostic about the function being used. Any univariate Morse-Smale function defined on the object can be used to drive the chartification process. Here we show a torus chartified using three different functions: height (left column), harmonic (middle column) and bi-harmonc (right column). For each example we show the function (top line), the resulting chartification (middle line) and the associated quadrilateral remeshing (bottom line).

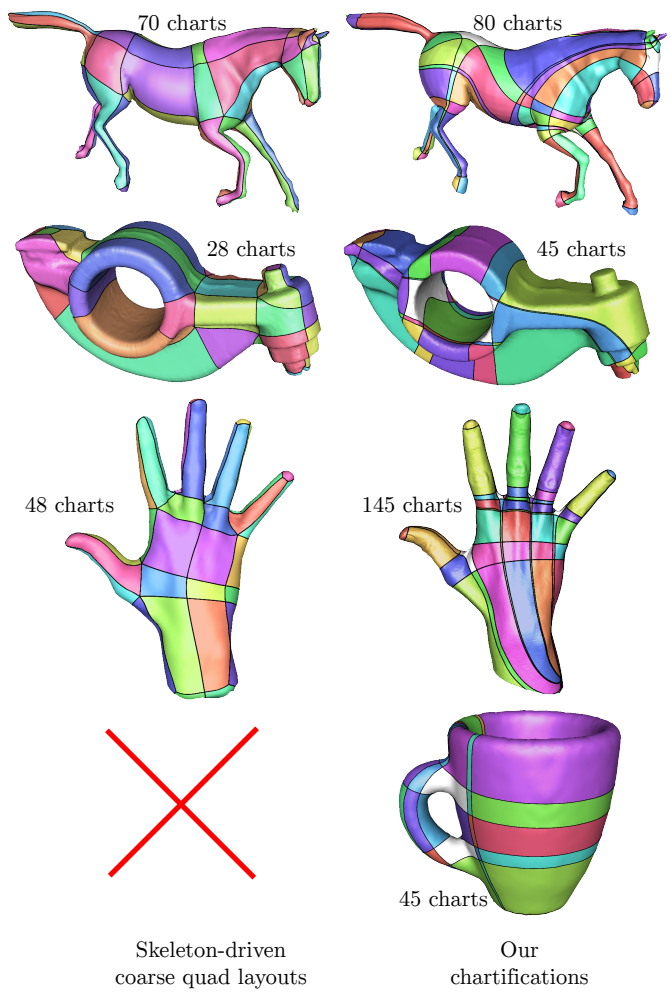


Figure 12: Visual comparisons between our chartification method (right) and the skeleton-driven coarse quad decomposition described in [50] (left). While skeleton-driven methods tend to produce coarser decompositions, they can only process tubular shapes that admit a skeletal representation, and are not suited for objects like a mug.

be superior in terms on number and shape of each chart, our method is far more general as it applies to any closed 3D surface, and it does not rely on heuristics to assign vertices to each domain. Indeed, tracing chart boundaries directly on the shape our method is guaranteed to always produce a valid chartification and accompanying parameterization.

On the downsides, the chartifications we obtain tend to contain extremely anisotropic charts nearby saddle points, and triangular (or nearly triangular) charts around polar regions (Figure 7). While these charts are harmless in the context of quadrilateral remeshing, in applications such as reverse engineering they may produce dramatically degenerate patches around critical points; a clearly undesired configuration in CAD applications. Nevertheless, this behaviour can be alleviated by locally relaxing the corners of the offending charts while fixing the global topology, for example iteratively re-positioning such corners at the center of the surface patch containing all the charts incident to it. Similar heuristics could also be used to globally relax the chartification and align it to sharp creases or other relevant features. We believe this type of post processing is beyond the scope of the current paper, and we leave it as a future work.

Finally, we plan to lift this machinery one dimension up to create volumetric decompositions. To this end, we observe that most of the ingredients we rely on (e.g. the Reeb graph and iso-contours) naturally extend to volumes. What is currently missing, is a method to trace the equivalent of integral curves on surfaces, which should complement iso-surfaces to bound each volumetric domain. Research on this topic is currently ongoing within our research group.

Acknowledgments

The authors would like to thank all the members of the Shape Modeling Group at CNR-IMATI for their valuable support, and to professor Mauro Beltrametti from University of Genoa for his precious help. Special thanks are also given to the anonymous reviewers for their comments and suggestions. This work has been supported by the internship program IMATI/TIROCINIO/1/2016/GE

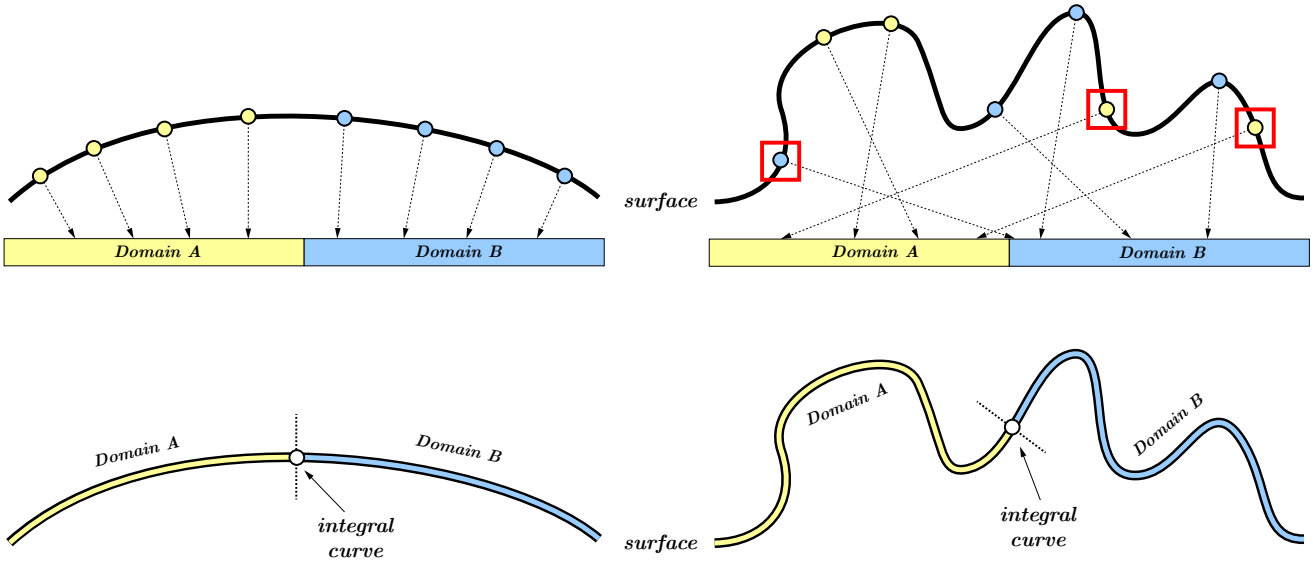


Figure 13: Previous methods for chart parameterization rely on heuristics to assign each mesh vertex to the correct domain. In [50] and ray casting along surface anti-normal is used (top). While this strategy works if the surface is nice and fair (top left), wrong vertex-domain assignments may occur for detailed surfaces (top right). In our approach separatrices between charts are defined directly on the surface and embedded in the tessellation, therefore vertex-domain assignment is implicit (bottom).

by CNR-IMATI and partially supposed by the ERC Advanced Grant CHANGE contract N.694515.

Bibliography

[1] Omri Azencot, Etienne Corman, Mirela Ben-Chen, and Maks Ovsjanikov. Consistent functional cross field design for mesh quadrangulation. *ACM Trans. Graph.*, 36(4):92:1–92:13, 2017.

[2] J. Andreas Bærentzen, Rinat Abdrashitov, and Karan Singh. Interactive shape modeling using a skeleton-mesh co-representation. *ACM Trans. Graph.*, 33(4):132:1–132:10, July 2014.

[3] 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 Survey*, 40(4):1–87, 2008.

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

[5] Silvia Biasotti, Bianca Falcidieno, and Michela Spagnuolo. *Extended Reeb Graphs for Surface Understanding and Description*, pages 185–197. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[6] Silvia Biasotti, Giuseppe Patanè, Michela Spagnuolo, Bianca Falcidieno, and Gill Barequet. Shape approximation by differential properties of scalar functions. *Comput. Graph.*, 34(3):252–262, June 2010.

[7] David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. Quad-mesh generation and processing: A survey. *Computer Graphics Forum*, 32(6):51–76, 2013.

[8] David Bommes, Tobias Vossemer, and Leif Kobbelt. Quadrangular parameterization for reverse engineering. In *International Conference on Mathematical Methods for Curves and Surfaces*, pages 55–69. Springer, 2008.

[9] J. W. Branch, F. Prieto, and P. Boulanger. Fitting surface of free form objects using optimized nurbs patches network with evolutionary strategies (mu + lambda) - es. In *Proceedings of the British Machine Vision Conference*, pages 99.1–99.10. BMVA Press, 2007.

[10] Florian Buchegger and Bert Jüttler. Planar multi-patch domain parameterization via patch adjacency graphs. *Computer-Aided Design*, 82:2–12, 2017.

[11] Marcel Campen and Denis Zorin. Similarity maps and field-guided t-splines: a perfect couple. *ACM Trans. Graph*, 36(4), 2017.

[12] Gianmarco Cherchi, Marco Livesu, and Riccardo Scateni. Polycube simplification for coarse layouts of surfaces and volumes. *Computer Graphics Forum*, 35(5):11–20, 2016.

[13] Sungyul Choe, Junho Kim, Haeyoung Lee, and Seungyong Lee. Random accessible mesh compression using mesh chartification. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):160–173, 2009.

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

[15] David Cohen-Steiner and Jean-Marie Morvan. Restricted Delaunay triangulations and normal cycle. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, pages 312–321, New York, NY, USA, 2003. ACM.

[16] 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.

[17] Joel Daniels, Claudio T Silva, and Elaine Cohen. Semi-regular quadrilateral-only remeshing from simplified base domains. *Computer Graphics Forum*, 28(5):1427–1435, 2009.

[18] Fernando do Goes, Mathieu Desbrun, and Yiying Tong. Vector field processing on triangle meshes. In *SIGGRAPH Asia 2015 Courses*, page 17. ACM, 2015.

[19] 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.

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

[21] J.-M. Favreau and V. Barra. Tiling surfaces with cylinders using n-loops. *Computers & Graphics*, 35, 02/2011 2011.

[22] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[23] Kangkang Hu and Yongjie Jessica Zhang. Centroidal voronoi tessellation based polycube construction for adaptive all-hexahedral mesh generation. *Computer Methods in Applied Mechanics and Engineering*, 305:405–421, 2016.

[24] Kangkang Hu, Yongjie Jessica Zhang, and Tao Liao. Surface segmentation for polycube construction based on generalized centroidal voronoi tessellation. *Computer Methods in Applied Mechanics and Engineering*, 316:280–296, 2017.

[25] 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.

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

[27] Ruotian Ling, Jin Huang, Bert Jüttler, Feng Sun, Hujun Bao, and Wenping Wang. Spectral quadrangulation with feature curve alignment and element size control. *ACM Transactions on Graphics (TOG)*, 34(1):11, 2014.

[28] Lei Liu, Yongjie Zhang, Yang Liu, and Wenping Wang. Feature-preserving t-mesh construction using skeleton-based polycubes. *Computer-Aided Design*, 58:162–172, 2015.

[29] Marco Livesu. cinolib: a generic programming header only C++ library for processing polygonal and polyhedral meshes., 2017. https://github.com/maxicino/cinolib/.

[30] Marco Livesu. A heat flow relaxation scheme for n dimensional discrete hyper surfaces. *Computers & Graphics*, 71:124 – 131, 2018.

[31] Marco Livesu, Alessandro Muntoni, Enrico Puppo, and Riccardo Scateni. Skeleton-driven adaptive hexahedral meshing of tubular shapes. *Computer Graphics Forum*, 35(7):237–246, 2016.

[32] Marco Livesu and Riccardo Scateni. Extracting curve-skeletons from digital shapes using occluding contours. *The Visual Computer*, 29(9):907–916, 2013.

[33] Linfa Lu, Xiaoyuan Qian, Xiquan Shi, and Fengshan Liu. Quading triangular meshes with certain topological constraints. *J. Comput. Appl. Math.*, 236:916–923, October 2011.

[34] Giorgio Marcias, Kenshi Takayama, Nico Pietroni, Daniele Panozzo, Olga Sorkine-Hornung, Enrico Puppo, and Paolo Cignoni. Data-driven interactive quadrangulation. *ACM Transactions on Graphics (TOG)*, 34(4):65, 2015.

[35] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, pages 35–57. Springer, 2003.

[36] Ashish Myles, Nico Pietroni, Denis Kovacs, and Denis Zorin. Feature-aligned t-meshes. *ACM Transactions on Graphics (TOG)*, 29(4):117, 2010.

[37] J. Palis and W. De Melo. *Geometric Theory of Dynamical Systems: An Introduction*. Springer-Verlag, 1982.

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

[39] 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.

[40] Nico Pietroni, Marco Tarini, and Paolo Cignoni. Almost isometric mesh parameterization through abstract domains. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):621–635, 2010.

[41] Budirijanto Purnomo, Jonathan D Cohen, and Subodh Kumar. Seamless texture atlases. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 65–74. ACM, 2004.

[42] 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.

[43] Viertel Ryan, Staten Matt, and Osting Braxton. Toward a paver replacement. In *26th International Meshing Roundtable*. Elsevier, 2017.

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

[45] Kenshi Takayama, Daniele Panozzo, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. Sketch-based generation and editing of quad meshes. *ACM Trans. Graph.*, 32(4):97–1, 2013.

[46] Kenshi Takayama, Daniele Panozzo, and Olga Sorkine-Hornung. Pattern-based quadrangulation for N-sided patches. *Computer Graphics Forum (proceedings of EUROGRAPHICS Symposium on Geometry Processing)*, 33(5):177–184, 2014.

[47] Marco Tarini, Enrico Puppo, Daniele Panozzo, Nico Pietroni, and Paolo Cignoni. Simple quad domains for field aligned mesh parametrization. *ACM Transactions on Graphics (TOG)*, 30(6):142, 2011.

[48] Julien Tierny, Joel Daniels II, Luis Gustavo Nonato, Valerio Pascucci, and Claudio T Silva. Interactive quadrangulation with Reeb atlases and connectivity textures. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1650–1663, 2012.

[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] Francesco Usai, Marco Livesu, Enrico Puppo, Marco Tarini, and Riccardo Scateni. Extraction of the quad layout of a triangle mesh guided by its curve skeleton. *ACM Transactions on Graphics*, 35(1):6:1–6:13, 2015.

[51] Shiwei Xiao, Hongmei Kang, Xiao-Ming Fu, and Falai Chen. Computing iga-suitable planar parameterizations by polysquare-enhanced domain partition. *Computer Aided Geometric Design*, 2018.

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

[53] Muyang Zhang, Jin Huang, Xinguo Liu, and Hujun Bao. A wave-based anisotropic quadrangulation method. *ACM Trans. Graph.*, 29(4):118:1–118:8, July 2010.