Interactive Quadrangulation with Reeb Atlases and Connectivity Textures

Julien Tierny, Joel Daniels II, Luis Gustavo Nonato,
Valerio Pascucci, *Member IEEE*, and Cláudio T. Silva, *Senior Member IEEE*

Abstract—Creating high-quality quad meshes from triangulated surfaces is a highly non trivial task that necessitates consideration of various application specific metrics of quality. In our work, we follow the premise that automatic reconstruction techniques may not generate outputs meeting all the subjective quality expectations of the user. Instead, we put the user at the center of the process by providing a flexible, interactive approach to quadrangulation design. By combining scalar field topology and combinatorial connectivity techniques, we present a new framework, following a coarse to fine design philosophy, which allows for explicit control of the subjective quality criteria on the output quad mesh, at interactive rates. Our quadrangulation framework uses the new notion of *Reeb atlas* editing, to define with a small amount of interactions a coarse quadrangulation of the model, capturing the main features of the shape, with user prescribed extraordinary vertices and alignment. Fine grain tuning is easily achieved with the notion of *connectivity texturing*, which allows for additional extraordinary vertices specification and explicit feature alignment, to capture the high-frequency geometries. Experiments demonstrate the interactivity and flexibility of our approach, as well as its ability to generate quad meshes of arbitrary resolution with high quality statistics, while meeting the user's own subjective requirements.

Index Terms—Quadrangulation, Reeb graph, Connectivity operators.

1 Introduction

The generation of quad meshes from triangle meshes is a challenging task that requires the simultaneous management of many objective and subjective quality criteria, such as feature alignment, orthogonality, regularity, and adaptive sampling. Automatic optimization of multiple criteria is difficult, where global and local constraints may contradict each other. For instance, enforcing local feature alignment may induce many vertices with non-ideal valences (called *extraordinary vertices*), which affects the regularity of the mesh in a global way. Thus, the notion of an *ideal* quad mesh is application dependent but also subjective.

User assisted schemes overcome the difficulties of automated decisions by providing the user with the ability to influence the importance of the quality criteria and related constraints. Starting with the pioneering work of Krishnamurthy and Levoy [24], there has been substantial work in this area, for instance by Tarini et al. [40] and Tong et al. [44]. We design a user-centric approach that offers exhaustive capabilities and comprehensive control during quadrangulation design. This work targets knowledgable users from the diverse application domains of quadrilateral meshes, otherwise frustrated by inappropriate design decisions made by automated techniques. However, most of existing semi-automatic techniques try to approximate the user's constraints through an optimization process [21], [6], [36], which may fail in precisely reproducing the exact configuration the user had in mind (inaccurate approximation of feature corners or misalignment of the extraordinary vertices). Moreover, they do not enable tools dealing with global and local constraints simultaneously. Also they often delegate tasks to the user not directly related to quad mesh design, such as the specification of paramaterization conditions or the selection of eigenfunctions. Finally, previous work has not been specifically designed to develop editing operations at interactive rates, a necessary feature for iterative artistic design.

We address these challenges, proposing a new quadrangulation framework that supports an *explicit* global and local control during the meshing process. In addition to design and editing at interactive rates, our framework provides flexibility by enabling the user to relocate extraordinary vertices as well as to modify mesh alignment, orientation and connectivity. These tasks are achieved through the new concepts of the *Reeb atlas* and *Connectivity textures*.

Contributions We reduce the challenges of quad mesh construction to that of topology aware scalar field design, while maintaining flexible control of the output mesh, at interactive rates. An itemized list of our main contributions:

- **Flexible and interactive quadrangulation** with explicit and robust control of extraordinary vertices and mesh alignment, experiencing response times of editing operations under half a second for models with up to 400,000 triangles.
- **Local and global design flexibility**, control of the location, valence, alignment of extraordinary vertices and of the orientation of the quads, at both a global and local level.
- **Topology aware scalar field design**, a novel tech-

- *Tierny is with the CNRS at Telecom ParisTech, France. E-mail: tierny@telecom-paristech.fr.*
- *Daniels and Silva are with NYU-Poly, USA. E-mails: daniels.two@gmail.com, csilva@nyu.edu*
- *Nonato is with ICMC, Universidade de São Paulo, Brazil. E-mail: gnonato@icmc.usp.br.*
- *Pascucci is with the SCI Institute, University of Utah, USA. E-mail: pascucci@sci.utah.edu*

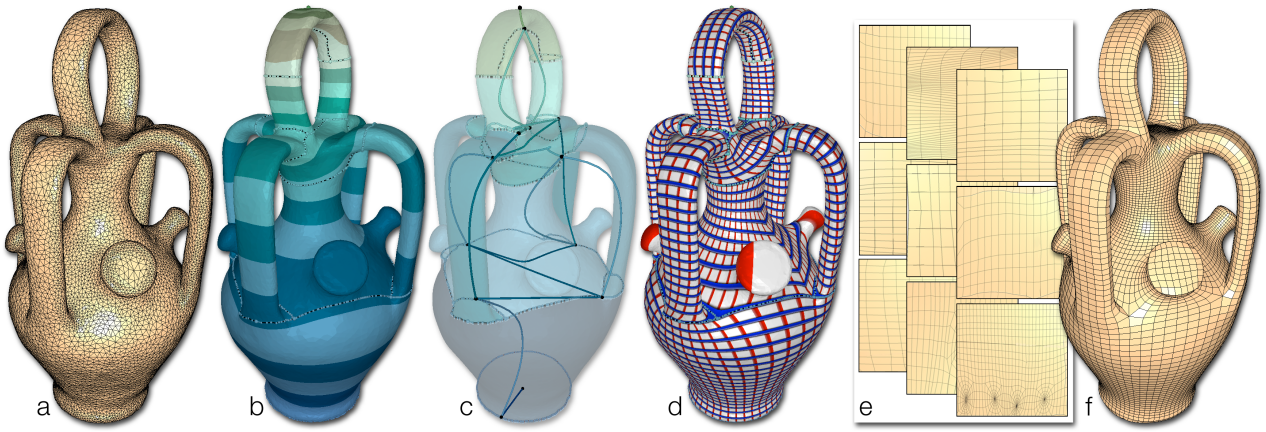


Fig. 1. For an input polygonal model (a), our interactive quadrangulation framework is driven by a user-defined scalar field (b), that guides a Reeb atlas segmentation of the model into a coarse quadrangulation capturing the dominant features of the shape (c) and allows control of the extraordinary vertices. Since the charts of the Reeb atlas have a guaranteed generic topology, they can be efficiently parameterized to the unit square (d). Designing connectivity textures of the unit square (e) enables an easy yet flexible quad-only meshing of the charts, providing a fine-grain control for the explicit capture of the high-frequency geometric details, while maintaining interactive rates of editing operations.

nique that allows the explicit control of each contour of the field, to better capture the model's geometry and to design *fractional critical contours*, while maintaing a consistent field topology.

- **Reeb atlas parameterization**, by exploiting the topological guarantees of a Reeb graph segmentation coupled with our topology aware scalar field design, we derive a robust technique for parameterization, topological and geometrical editing on surfaces of arbitrary topology.
- **Connectivity texturing**, interactive local modifications to the output mesh with interaction tools similar to on-surface texture painting.

2 Related Work

The literature on quad mesh generation has experienced considerable growth in the last few years. To better contextualize our work, this discussion organizes existing techniques as to the level of user control allowed within the varying approaches. For a more comprehensive discussion on the subject, we direct the reader to the surveys of Alliez et al. [2] and Hormann et al. [20].

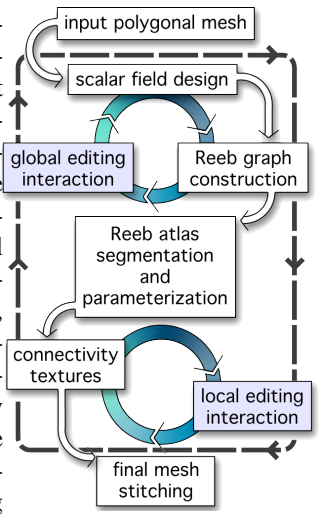
Automated techniques. Automated techniques aim at building a quadrangulation avoiding user intervention altogether. For example, connectivity-based approaches convert polygons into quad elements with local operators that are driven by advancing fronts [29], simplification to base domains and regular refinement [10], merging adjacent triangles [25], and the projections of voxel vertices [7]. The global distribution of rectangular cells facilitates the construction of quad-dominant connectivity [45]. Additionally, numerical integration of orthogonal vector fields [22] and principal curvature directions [1] is successful in automatically constructing well shaped and aligned quad elements. Global parameterization schemes [35], [3] and individual parameterization of localized charts [5],[31] generate quality meshes dominated by regular vertices (valence 4). In particular, in [31], an automatic parameterization algorithm based on a Reeb graph decomposition is proposed. In comparison, our work, also using a Reeb graph decomposition, provides a formalism to achieve interactive control over the extraordinary vertex topological and geometrical layout. While these techniques have varying success concerning feature alignment, adaptive sampling, and element quality, automated methods do not provide flexible mechanisms to handle extraordinary vertices (valence other than 4) and mesh alignment that may lead to undesirable artifacts in the final mesh. As the concept of the ideal quad-mesh is versatile, application dependent and subjective, flexibility and control are of paramount importance.

User-driven techniques. We discuss user-driven techniques as those methods that provide mechanisms to allow additional user annotations of the model offering some relative control over extraordinary vertices and mesh alignment. For example, quadrangulations from scalar fields allow inputs including the specification of extrema vertices [14] to control the placement of *integer* polar singularities that correspond to extraordinary vertices in the final mesh, as well as *conductance* terms to control mesh alignment [38]. Spectral quadrangulation requires the user to select the appropriate eigenfunction [13], offering partial control over the extraordinary vertices (where each extremum of the eigenfunction has an unconstrained valence), and extended user inputs to influence alignment and importance sampling [21]. Direction field painting [6], [36] by the user influences mesh alignment while trying to determine automatically *natural* locations for extraordinary vertices. User-defined coarse quad meshes drive the global structure of a final quad representation, locally sampling each region with regular grids [24], or setting up linear system constraints for a global parameterization [44], or by adhering to spe-

cific connectivity rules to develop highly regular polycube representations [40], [26]. Existing user-driven techniques permit increased control in the quad meshing process by allowing specification of alignment and/or handling of extraordinary vertices. However, the interactions may not be straightforward (with additional inputs dictated by the nature of the algorithm and not by that of the quad design process itself) nor offer a full *exact* control over the final mesh structure at interactive rates. In this work, we propose a framework that encorporates multi-level control, local and global, of the mesh structure and alignment, with interactive response to editing operations.

3 Framework Overview

Our quadrangulation framework is motivated by several target features, including *interactivity*, *local and global editing scopes* and *design flexibility*. These goals are intended to support the user's artistic process by providing important functionalities, allowing constant modification of the design with interactive response times for any editing operation. Our local and global controls encourage a multi-level design philosophy: first, modifying the overall structure affecting the configuration of the extraordinary vertices, as well as coarse mesh alignment; then providing small-scale handling to capture the high-frequency geometrical details of the surface. The remainder of this section presents a framework overview, illustrated to the right and in Fig. 1. We reduce the challenging problem of quad mesh construction to that of topology aware scalar field design to take advantage of the efficient nature of scalar field computation. The global geometry and structure of the quad mesh is inferred from a user-defined scalar field constructed over the input triangular mesh (Fig. 1b). This first stage of our framework uses an efficient linear system solver based on fast Cholesky fatorization of the Laplace operator. We build on this in a novel way with topological constraints for explicit control of all critical level sets within the scalar field (Sec. 6).



Despite its advantages of simplicity and speed, scalar field based quadrangulation [14] can only model *integer* singularities, corresponding to the critical points of the scalar field and generating high-valence extraordinary vertices. In order to overcome this issue, while exploiting the speed and flexibility of scalar field design, we introduce the new notion of *Reeb atlases* (Sec. 5). Given a scalar field defined on the input mesh, we build the Reeb graph (Fig. 1c) to guide a chart segmentation with local parameterizations of the surface (Fig. 1d). Global editing operators of the Reeb atlas modify the scalar field by manipulating the geometry of structures derived from the Reeb graph.

After establishing a segmentation and local parameterizations that define a coarse quad mesh, we introduce the notion of *connectivity textures* (Sec. 7) that provide the user an easy and flexible localized control of the quad mesh construction. In this stage, the user defines the local connectivity as a *texture* living on top of the parameterization of the chart of interest. This connectivity texture abstraction increases the design flexibility by enabling any meshing strategy, not only restricted to parameterization contouring. Note that in our framework, the user still has the possibility to come back to the global editing of the atlas, even after local connectivity texturing. We describe in detail in Sec. 6 in which configurations the connectivity textures need to be reset. Finally, a stitching procedure (Sec. 7.2) composes the connectivity textures to construct the quad-only mesh (Fig. 1e). A typical usage scenario of our interactive approach is presented in Sec. 8 along with its performance evaluation.

4 Interactive Scalar Field Design

In our framework, the global control of the quad mesh is dependent on the design of a piecewise linear (PL) scalar field defined on the vertices of the input triangular surface and linearly interpolated over the triangles. The construction of this field drives global control mechanisms over the extraordinary vertices and high-level orientation of the mesh. To best fit our application, it is important for the scalar field to be smooth, to contain a controlled number of critical points, and to be computed and updated within interactive rates. Harmonic fields become a natural choice because their properties closely parallel these requirements. A harmonic field defined on a manifold surface is a scalar field $f : \mathcal{S} \to \mathbb{R}$ satisfying the differential equation,

$$\nabla^2 f = 0, \tag{1}$$

subject to boundary conditions (Dirichlet in our context). In the discrete case, where the surface is given by a triangular mesh $\mathcal{S}$, the Laplace-Beltrami operator ∇^2 is usually discretized using cotangent weights [33], which leads to a symmetric and positive-definite sparse matrix $L = W - D$ whose elements w_{ij} of W are defined,

$$w_{ij} = \begin{cases} -\frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij}) & \text{if edge } [i,j] \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where α_{ij} and β_{ij} are opposite angles to edge e_{ij} and D is a diagonal matrix with elements d_{ii} given by row sums of W.

We make use of the penalty method to impose constraints to the linear system derived from equation (1). Consider C,

the set of indices of constrained vertices, then the harmonic scalar field is obtained by solving the linear system,

$$(L + P)f = Pb, \tag{3}$$

where P is a diagonal matrix with non-zero entries $p_{ii} = \alpha$ only if $i \in C$ and α is the penalty weight ($\alpha = 10^8$ [46]). Constrained values are set within the vector b,

$$b_i = \begin{cases} s_i, i \in C \\ 0, \text{otherwise} \end{cases} \tag{4}$$

where s_i is the desired scalar value assigned to vertex i. The main advantage of using penalty method to impose constraints is that supernodal schemes [12] can be used to update (and downdate) the Cholesky fatorization, making it possible to include and remove constraints efficiently [46].

The initial user input to our framework is the specification of the extrema of an harmonic function. It has been observed that in general the extrema are best placed at extremities of prominant features of the shape (Fig 2). From a meshing perspective, these extrema will correspond to polar extraordinary vertices (valence editing operations will be discussed in Sec. 6). We provide the user with an automatic initial suggestion aiming at detecting prominent features by computing the integral of the geodesic distance function. This function is approximated in practice as follows (see [19] for further details). Geodesic neighborhoods (of a pre-defined maximum radius r) are iteratively built with Dijkstra's algorithm by picking randomly unvisited *source vertices* until the entire surface is covered. Then the integral geodesic distance is approximated by summing for each vertex its geodesic distance to each *source vertex*, weighted by the area of the corresponding geodesic neighborhood (for r, we used the original threshold suggested in [19]). This function is invariant to rigid transformations and will give a high value for the vertices which are the furthest away from the *geodesic barycenters* of the surface. Thus, we will identify the maxima of this function as prominent features for the initial suggestion. In practice, we sort the maxima by decreasing function value and add them to the list of constrained vertices C if their geodesic distance to previously selected constrained vertices is higher than r. The list $C = \{(v_i, h_i)\}_{i=0}^{N}$ is split into two subsets C_0 and C_1 in accordance with the height function h, where h_{min} and h_{max} are the respective minimum and maximum values of h, then $C_0 = \{v_i \mid v_i \in C \text{ and } h_i < \frac{1}{2}(h_{min} + h_{max})\}$ and $C_1 = C - C_0$. We assign initial constraint values 0 and 1 to the vertices in C_0 and C_1 respectively (Eq. 4). While the set of constrained vertices C is not dependent on global rotations of the surface, the height function is. We give the user the opportunity to re-oriente the shape if needed to obtain a more appealing initial suggestion.

5 Reeb Atlas

In this section, we introduce the notion of *Reeb atlas* which is our core abstraction for interactive quad-remeshing. We describe in the next section an important contribution of our approach, the interactive editing capabilities of the Reeb atlas, which allows the user to interactively control the global extraordinary vertex layout of the output mesh.

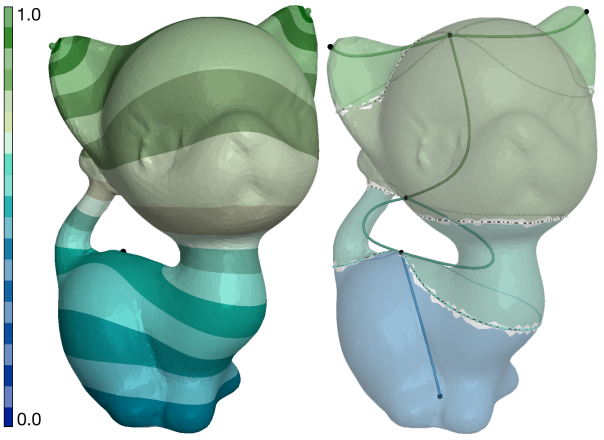


Fig. 2. The harmonic scalar field f is drafted by the user, modifying the automatically suggested extrema. Reeb charts, constructed from the arcs of the Reeb graph, segment the model into multiple regions with known topology.

Given a scalar field f over a manifold surface $\mathcal{S}$, a straightforward quadrangulation strategy consists in computing a parameterization of $\mathcal{S}$, with $U : \mathcal{S} \to [0, 1]$, whose level sets align with those f, and $V : \mathcal{S} \to [0, 1]$, whose level sets align with the gradient of f. On the sphere with two antipodal extrema, U and V respectively map to the latitude and longitude coordinate systems, and contouring regularly along both U and V constructs a quad dominant mesh with nearly orthogonal edges. While this technique has been shown to be simple and efficient [14], it is limited by the fact it can only model *integer singularities*, generating extraordinary vertices with high valence that correspond to the critical points of f. To benefit from the simplicity and speed of scalar field based quadrangulation, we extend this methodology with added structural control by leveraging topological structures inferred from the scalar field.

5.1 Morse Theory Background

For completeness, we briefly review Morse theory as well as its extension to the piecewise-linear setting.
Let $f : \mathbb{M} \to \mathbb{R}$ be a smooth function on a manifold $\mathbb{M}$. For a given scalar w, the level set $L(w)$ is defined as the inverse image of w onto $\mathbb{M}$ through f, $L(w) = f^{-1}(w)$. We call each connected component of $L(w)$ a *contour*. As w changes continuously in $\mathbb{R}$, the points at which the topology of a contour changes are called *critical points* [28] and the corresponding function values are called *critical values*. If all of the critical points of f are non-degenerate and have distinct values, then f is a Morse function. For a given critical point c, if it admits an open neighborhood on $\mathbb{M}$ such that it is the unique critical point, then c is called *isolated*. Morse functions counts finitely many critical points and all of them are isolated.

In the case of triangulated surfaces, the domain of $f : \mathcal{S} \rightarrow \mathbb{R}$ is a manifold 2D simplicial complex. Within each simplex of $\mathcal{S}$, f is the linear interpolation of its values at the vertices: f is called a piecewise-linear (PL) function. The following operations on a simplicial complex are used to identify critical points. The *star* of a simplex v is the set of simplices that contain v as a face: $St(v) = \{\sigma \in \mathcal{S} \mid v \leq \sigma\}$, where $v \leq \sigma$ denotes that v is a face of σ. The *link* $Lk(v)$ of a simplex v is the set of simplices in the closure of the star of v that are not also in its star: $Lk(v) = \overline{St(v)} - St(v)$, where $\overline{St}$ denotes the closure of the star. In PL functions, critical points can only occur at vertices. The *lower link* of v is the subset of the link containing only simplices with all their vertices lower in function value than v: $Lk^-(v) = \{\sigma \in Lk(v) | \ u \leq \sigma \rightarrow f(u) < f(v)\}$. Symmetrically, the *upper link* is $Lk^+(v) = \{\sigma \in Lk(v) | \ u \leq \sigma \rightarrow f(v) < f(u)\}$. A vertex v in $\mathcal{S}$ is *regular* iff both $Lk^-(v)$ and $Lk^+(v)$ are simply connected, otherwise v is a critical point of f. If $Lk^-(v)$ is empty, v is a *minimum*. Symmetrically, if $Lk^+(v)$ is empty, v is a *maximum*. Otherwise, if v is not regular, v is a *saddle*. We call a *critical contour* a contour containing a critical point. A sufficient condition for the above critical point classification to succeed is that all the vertices of $\mathcal{S}$ admit distinct values. If this is not case, we symbolically perturbate f with *Simulation of Simplicity* [17], which will also implicitly resolve non-isolated critical points.

5.2 Reeb Graph

Reeb Graphs are traditionally defined through an equivalence relation [37]. In this discussion, we will use an alternate, but equivalent, formalism [42] since it better reflects the implemented data structure and it raises a straightforward definition of the *Reeb charts*. Given a smooth manifold $\mathbb{M}$, a *retraction* is defined as a continuous map such that the image is a subset of its domain $\mathbb{M}$ and the restriction of the map to the image is the identity [18]. A *contour retraction* of $\mathbb{M}$ under a Morse function f is defined as a continuous map that retracts each contour (connected component of a level set) of f to a single point (see [42] for further details). By continuity, adjacent contours are retracted to adjacent points and distinct contours are retracted to distinct points. Then the Reeb graph $\mathcal{R}(f)$ is the *contour retract* of $\mathbb{M}$ under f. It consists of *arcs* and *nodes*, where branching only occurs at critical points of f. The field f can be decomposed into $f = \psi \circ \phi$, where $\phi : \mathbb{M} \rightarrow \mathcal{R}(f)$ is a contour retraction and $\psi : \mathcal{R}(f) \rightarrow \mathbb{R}$ is a continuous function that maps points in $\mathcal{R}(f)$ to the real line $\mathbb{R}$. Several algorithms have been proposed to compute a Reeb graph from a piecewise-linear scalar field defined on a triangular surface $\mathcal{S}$ [9], [30]. Our experiments showed that the saddle contouring algorithm [32] presented the best performances in our context. Its complexity depends on the number of simplices in $\mathcal{S}$ and the number of saddles of f. Typically the user designed scalar fields generate few saddles, leading to virtually linear computation. Moreover, our implementation explicitly stores the regular vertices of

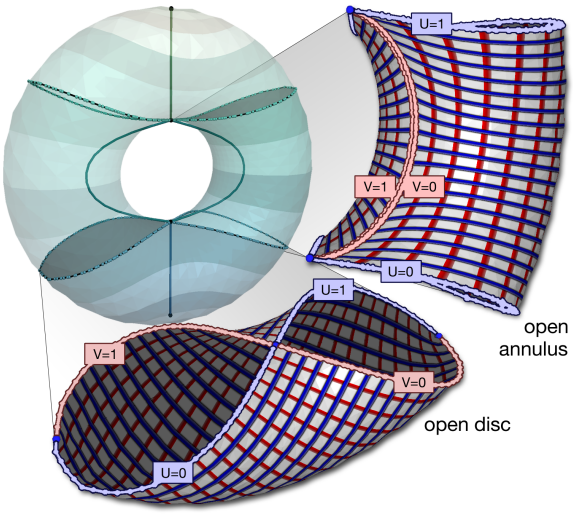


Fig. 3. Our parameterization strategy maps the boundaries of the Reeb chart (open annulus or open disc) to the unit square by defining UV Dirichlet boundary conditions within the Laplace system. Open annuli are cut into discs by a streamline guided by the scalar field gradient.

f along the arcs of the Reeb graph. Although the Reeb graph is defined for manifolds of arbitrary dimension (see [4] for a survey), we restrict the remaining discussion to closed 2-manifolds of arbitrary genus, denoted $\mathcal{S}$ (surfaces with boundaries are discussed in Sec. 8.1).

Reeb chart. Given the contour retraction $\phi : \mathbb{M} \rightarrow \mathcal{R}(f)$, a Reeb chart $\mathcal{S}_i$ is the preimage by ϕ of the *interior* of an arc $\mathcal{A}_i$ of $\mathcal{R}(f)$ [43]. By construction, Reeb charts are continuous pilings of closed 1-dimensional contours. Since they are the preimage of the *interior* of arcs, Reeb charts do not include critical contours and are thus open sets with the topology of an open annulus (a connected genus zero surface, with two boundary components excluded). Note that a boundary component collapses to a point if an arc is linked to an extremum. Because Reeb charts are constructed from the regular contours of f, their definition does not require f to be strictly Morse (i.e. degenerate saddles). Given the segmentation of the surface into Reeb charts, the *Reeb atlas* is defined as the union of the charts with respective local parameterizations. Because Reeb charts have a controlled topology, they are robustly, easily and efficiently parameterized with a generic strategy. The remainder of this section discusses the parameterization of a Reeb chart.

5.3 Reeb Chart Parameterization

Each Reeb chart $\mathcal{S}_i$ of $\mathcal{S}$ is built by duplicating the triangles of $\mathcal{S}$ that fully map to the *interior* of the arc $\mathcal{A}_i$ via ϕ. *Boundary triangles*, intersected by the critical contours adjacent to $\mathcal{A}_i$, are also inserted into $\mathcal{S}_i$, illustrated as grey triangles in Fig. 2. In order to obtain smooth boundary components for the charts, the *boundary triangles* are shrunk such that their vertices being outside of the chart

get snapped along the boundary critical contour (by sliding them along an incoming edge crossing the contour).

A parameterization maps the open annulus $\mathcal{S}_i$ to the unit square by solving two harmonic functions with Dirichlet boundary conditions (Fig. 3) using the solver presented in Sec. 4. The field $U : \mathcal{S}_i \rightarrow [0, 1]$ is computed to align with the level lines of f by constraining the boundary vertices of $\mathcal{S}_i$, projected to the two critical contours, to either $U = 0$ or $U = 1$ (Fig. 3). The orthogonal field $V : \mathcal{S}_i \rightarrow [0, 1]$ is computed by tracing a cutting streamline along the mesh edges of $\mathcal{S}_i$ guided by the gradient of U, turning the annulus into a disc. The vertices of the cutting edges are duplicated and assigned values, $V = 0$ and $V = 1$, to map the boundary of $\mathcal{S}_i$ to the unit square. Each Reeb chart $\mathcal{S}_i$ mapping through ϕ to an arc of $\mathcal{R}(f)$ adjacent to an extremum of f (Fig. 3) are parameterized differently, in the purpose of generating quad-only outputs (as explained later in Sec. 6.3). The boundary triangles that neighbor the extremum are included within $\mathcal{S}_i$ so that $\mathcal{S}_i$ has a single boundary component and is homeomorphic to a disc. The boundary vertices are segmented into four contiguous polylines and assigned values mapping the boundary to the unit square. This parameterization scheme splits polar singularities into four fractional components (Sec. 6.3).

At this stage, the Reeb atlas represents a coarse quadrangulation of the surface. Each Reeb chart is equipped with its own local parameterization to the unit square and may be represented by a single quad in the coarse mesh. Note that saddles of f correspond to extraordinary vertices.

6 Global Editing of the Reeb Atlas

In the previous section, we introduced the Reeb atlas abstraction. We now describe the interactive control capabilities of this framework, with which the user will interact in a feedback loop process. We propose a set of editing operations to allow the user to control both the geometry and the topology of the Reeb atlas. To ensure interactive times in the design process, the Reeb atlas editing operations are based on modifications of the underlying scalar field f through fast updates provided by the supernodal schemes of the penalization solver (Sec. 4). These functionalities provide the user global control on the orientation of the final mesh by editing the geometry of Reeb chart boundaries as well as global control on the valence, location and alignment of extraordinary vertices.

The Reeb chart boundaries are defined by critical contours of f. While the relocation of minima and maxima is well understood, consisting of removing the original constraint and replacing it with a new one at a different location, moving saddle contours requires a bit more machinery. This section discusses the constraints we associate with the saddle contours (Sec. 6.1) and, consequently, our ability to control them (Sec. 6.2).

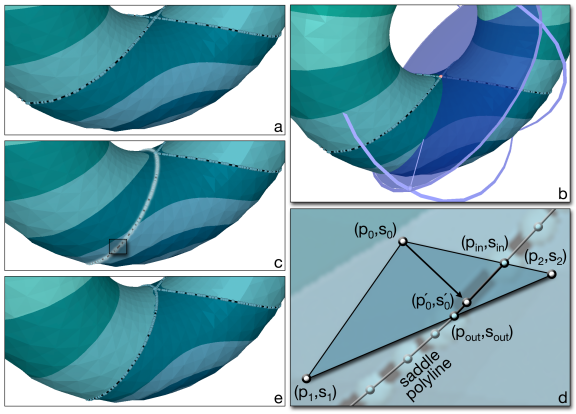


Fig. 4. Reorienting a saddle contour: the original levelset curve (a) is modified through the critical contour widget (b) where the mesh-plane intersection describes the new contour geometry (c). Saddle triangles are constrained (d) to ensure the scalar field respects the critical contours (e).

6.1 Enforcing the Geometry of Critical Contours

For each saddle contour, additional constraints are added to the Laplace system at the vertices of *saddle triangles* (triangles intersected by the critical contour) to ensure that the scalar field levelsets respect the user designed geometry. Assume that the user modified the geometry of a saddle contour with a scalar value s_c (Sec. 6.2) so as to intersect the triangle $t = \{p_0, p_1, p_2\}$ on edges $e_0 = \overline{p_0 p_1}$ and $e_2 = \overline{p_2 p_0}$ (Fig. 4). The intersection points $p_{in} = p_0 + \alpha_0(p_1 - p_0)$ and $p_{out} = p_2 + \alpha_1(p_0 - p_2)$ and scalar values $s_{in} = s_0 + \alpha_0(s_1 - s_0)$ and $s_{out} = s_2 + \alpha_1(s_0 - s_2)$, where s_0, s_1, s_2 are associated with the vertices of t, assist in the definition of vertex constraints. The vertex p_i of t is projected onto the segment $\overline{p_{in} p_{out}}$ yielding the point p'_i with a scalar value s'_i. The scalar constraint assigned to p_i for t is $\tilde{s}_i = s_c + (s_i - s'_i)$. The final constraint of each vertex is averaged with values of adjacent saddle triangles. This novel constraint computation enables strict control of the contour of f, aligning to the user's designed polyline. Introducing Dirichlet boundary conditions in this way might generate unintended critical points in highly constrained configurations. To remove this potential noise in f, we use a combinatorial cleanup procedure. The extra critical points are identified without ambiguity (not belonging to any constraints) and removed from the Reeb graph [15]. To reflect the changes induced by the simplification within the scalar field, we used the algorithm described in [16]. This procedure guarantees the topological correctness of the designed scalar field.

Note that initially, and also after each editing operation, *all* the saddle contours of f are constrained using the above scheme (even if the contours are not displaced). Then, the effects of the editing operations are localized to the charts of interest. For instance, when adding an extrema within a Reeb chart (Fig. 5, top), the level sets of f remain unchanged in the other charts since the geometry and the f value of their boundaries are enforced in the solve (see

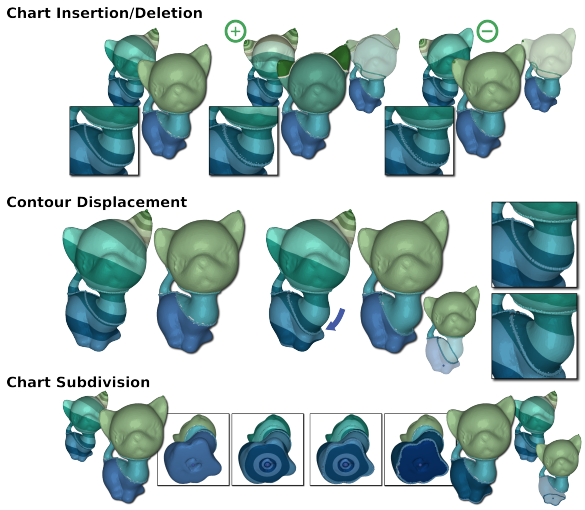


Fig. 5. Global impact of Reeb atlas editing operations. Reeb charts whose triangle list changes after the operation are transparent. Top: chart insertion and deletion (through insertion and deletion of extrema). Middle: contour displacement. Bottom: chart subdivision.

Fig. 5, top insets). Similarly, when displacing a saddle contour at the boundary of a chart of interest, the other boundaries remain in position and conserve their f value (Fig. 5, middle). Then to guarantee that the saddle contour displacement does not alter the topology of f, our interface discards any interaction that generates a contour which is not a closed loop, or that makes the contour overlap another saddle contour or sweep an extremum. Finally, note that after each editing operation, the Reeb graph is recomputed globally. Then, the list of Reeb charts which need an update of their individual triangle list is tracked based on the lists of regular vertices of the arcs of the Reeb graph (Fig. 5). From the user's perspective, every time an extremum is added to or removed from a chart of interest, the chart needs a reset of its triangle list. Also, every time a chart boundary is edited (including fractional singularity design, cf. Sec. 6.3), the adjacent charts need an update of their triangle lists.

6.2 Manipulating the Critical Contours

The modification of saddle curves is achieved via a 3-dimensional *critical contour widget* (Fig. 4). This widget consists of rotational handles that allow the user to orient a cutting plane whose intersection with the surface defines the new saddle curve geometry. Anchors can be defined on the saddle curves to behave as endpoints for the widget, localizing the effects of the manipulation. This widget and its design interactions are further demonstrated in the accompanying video. We further describe the use of this widget for multiple important and novel controls in scalar field design.

Aligning Multiple Saddles. When the scalar field f admits a succession of nearby saddles (Fig. 6), it may be desireable to align the associated critical contours. In effect, this

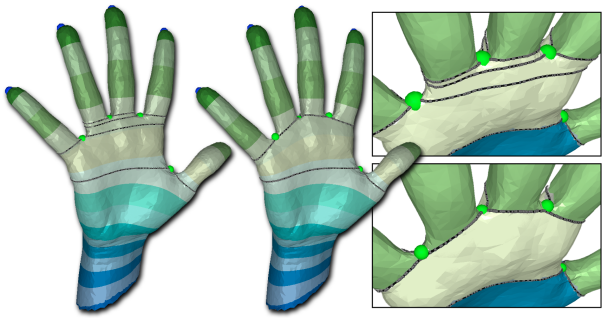


Fig. 6. *Thin Reeb charts* (left and top) result where multiple saddles have nearly equivalent scalar values. Our global editing operations support the geometric contol of the contours, linking the saddle vertices and removing *thin Reeb charts* (right and bottom).

functionality coarsens the Reeb atlas by removing *thin Reeb charts* to align extraordinary vertices in the final quad mesh. The atlas coarsening maintains the total number of saddle vertices while decreasing the number of saddle contours, yet the Reeb charts remain well defined.

Aligning multiple saddles (Fig. 6) is achieved interactively by first deleting the critical contours to align. Next, the user clicks on pairs of saddles to be connected with automated mesh traversals (shortest paths) providing initial curve segments. Then, the user can further re-orient them with the critical contour widget (the aligned curves are then constrained as discussed in Sec. 6.1).

Subdividing Reeb Charts. In addition to merging Reeb charts, we support their splitting as well. Because the Reeb chart is defined as a collection of contours, splitting a chart into two can be achieved by flagging a particular contour (i.e. clicking on a vertex, see Fig. 5, bottom) and by construction each child chart maintains the topological guarantees of the Reeb atlas segmentation. Reeb chart splitting facilitates alignment of the scalar field, the charts' parameterizations and consequently the final quad mesh, to surface features. This functionality is demonstrated with the L-shape (Fig. 13) and Moai (Fig. 14) models.

6.3 Fractional Singularities

In the following, we assume that the final quad mesh is extracted by contouring the local chart parameterizations. As such, we are able to offer a formalization linking scalar field critical points and the extraordinary vertices of the related quadrilateral mesh. We introduce a novel mechanism for controlling the extraordinary vertices, managed at a global level during the scalar field design, through the new notion of fractional critical points (an extension of concepts from direction field design [36] and surface parameterization [35], [44] to scalar fields).

Fractional Polar Singularities. When a boundary component of a Reeb chart is an extremum vertex, parameterizing

Operation	Add Chart	Del Chart	Move Bound.	Add Bound.	Saddle Align.	Frac. Poles	Frac. Saddles
Figure	Fig. 5 (top)	Fig. 5 (top)	Fig. 5 (mid.)	Fig. 5 (bot.)	Fig. 6	Fig. 7	Fig. 8
Interaction	Click	Click	Drag	Click	Clicks	Clicks	Clicks + Drag

TABLE 1
Summary list of the Reeb atlas editing operations.

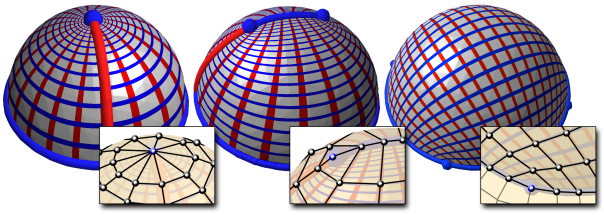


Fig. 7. Fractional poles: a polar vertex (left) split into 2 half-poles (val. 2, middle) and 4 quarter-poles (val. 3, right).

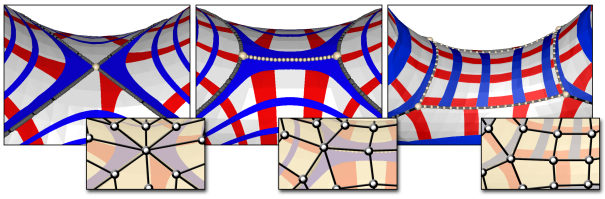


Fig. 8. Fractional saddles: a saddle vertex (val. 8, left) split into 2 half-saddles (val. 6, middle) and 4 quarter-saddles (val. 5, right).

the chart with a cutting streamline (as an open annulus) generates a polar singularity that leads to triangular elements around a high valence extraordinary vertex (Fig. 7, left). To guarantee the generation of a quad-only output, we use the notion of *fractional singularity*. In particular, our default parameterization strategy for disc charts (Sec. 5.3) splits a polar singularity into quarter poles, where the resulting quad mesh containts 4 valence 3 vertices (Fig. 7, right).

An alternative proposed to the user is to split the polar singularity into 2 half-poles, constraining a sequence of mesh edges with constant min/max f values (0 or 1); then, the chart is parameterized with a cutting streamline (Fig. 7, middle). This configuration corresponds to the concept of *non-isolated critical points* in the smooth setting. We use *Simulation of Simplicity* (*SoS*) [17] in the PL setting to maintain a consistent combinatorial representation of f. The resulting quad mesh has 2 valence-2 extraordinary vertices at the endpoints of the extremum segment.

Fractional Saddle Singularities. In the spirit of handling fractional polar singularities, we design fractional saddle singularities within the scalar field design. Saddle vertices correspond to extraordinary vertices within the final quad mesh (Fig. 8, left). We provide a set of atomic editing operations that enable the user to redistribute easily the high valency of saddles with the new notions of *half-* and *quarter*-saddles. While there exists multiple possible combinations of adjacent Reeb chart parameterization configurations, we abrieviate this discussion to the example shown in Fig. 8. A non-degenerate saddle contour is a set of two closed curves admitting exactly one common point. Half-saddle splitting is supported by modifying the geometry of the saddle contour to be described, for example, with two closed curves linked by a *middle segment* that is aligned to the edges of the mesh. The half-saddles are defined at the intersection of the middle segment and the two closed curves (Fig. 8, middle).

Due to our default parameterization method for discs, splitting polar vertices into quarter-poles, the construction of half-saddles can lead to the removal of pairs of extraordinary vertices in the quad mesh. In particular, the singularities of *min/split-saddle* and *merge-saddle/max* Reeb charts are removed. On the torus model, all singularities can be classified as these types, resulting in a completely regular quadrangulation (Fig. 9).

To design half-saddle configurations (Fig. 8, middle), the user deletes the original saddle contours and vertex, then initiates the tracing of two contours from manually chosen vertices. The middle segment is automatically computed as the shortest path defined along mesh edges between the two points. User-defined half-saddle contours can be geometrically edited via the critical contour widget to align to surface features. The network of critical contours defining the half-saddle is assigned a single constraint isovalue. Note, the middle segment relates to the notion of non-isolated critical point in the smooth setting, handled in the PL setting with *SoS*. Splitting a saddle reduces the valence of the related vertex by redistributing it among the multiple, created extraordinary vertices. The quarter-saddle configuration (Fig. 8, right) further supports this observation, later exemplified on the Blade model (Fig. 14). Quarter saddles are designed by first deleting the original saddle contour and vertex. Then the user clicks on a reference vertex to extract its isocontour. Three other reference vertices are selected along this isocontour and pairs of reference vertices are connected through shortest path computations (Fig. 8, right). Finally, an extremum is inserted at the location of the original saddle to maintain a valid field topology. The user can use the critical contour widget to further align the contour (constrained as discussed in Sec. 6.1).

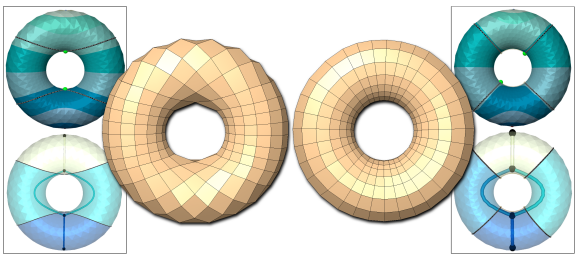


Fig. 9. The torus remeshed with fractional half-saddles (right) does not contain any extraordinary vertices.

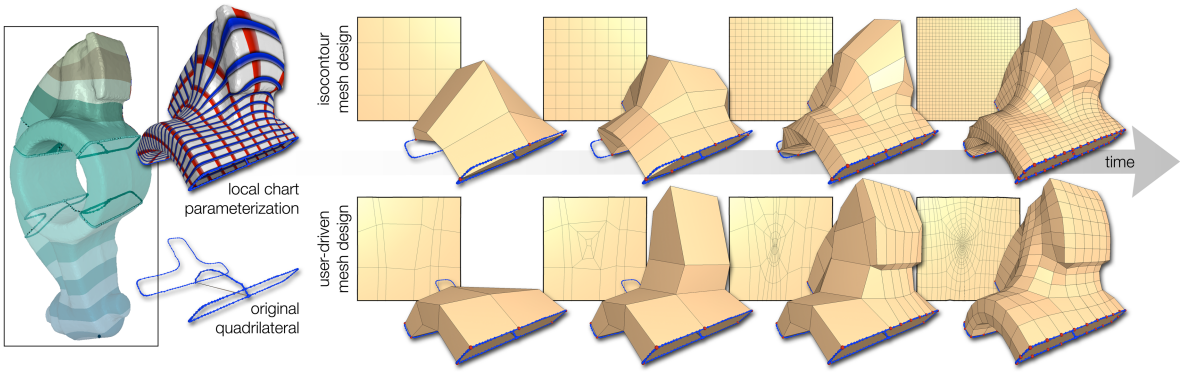


Fig. 10. Connectivity texturing of a challenging geometry (multiple sharp features) on a purposely coarse segmentation. For a given Reeb chart (left), global subdivisions reproduce meshing results from isocontouring (top). In contrast, connectivity texturing maps user designed quadrangulations of the unit square to the parameterized chart (bottom) for improved flexibility and control. Above, we illustrate snapshots of the design process over time (bottom): in this example, the user triggered a few polychord insertions, followed by cube subdivisions to capture the feature corners at the top of the shape, and finally subdivided the texture to obtain the desired sampling density.

7 Local Editing & Connectivity Textures

To this point the user-defined scalar field guides a Reeb atlas segmentation of the model, resulting in multiple individually parameterized mesh regions. In practice, quadrangulation techniques based on parameterizations derive a final quad mesh by uniformly contouring each parameter. This gridded sampling approach generates all ideal valence vertices internal to each segmented region; however, it lacks flexibility in the local design of mesh connectivity. We introduce *connectivity textures* that decouple the alignment of the final mesh elements from the underlying parameterizations to improve flexibility in the mesh design (Fig. 10).

Connectivity textures. Similar to texture images, a connectivity texture is a user designed quadrangulation of the unit square that is *mapped* to a Reeb chart region based on its parameterization. This abstraction improves flexibility within the design process by allowing the user to explicitly insert additional extraordinary vertices and modify the orientation of the quads. Additionally, internally representing the quadrangulations within the plane, then projecting the points onto the Reeb chart improves robustness and speed, discussed throughout the remainder of this section.

Planar-based Projections. The vertices of a connectivity texture are efficiently projected to the Reeb chart mesh $\mathcal{S}_i$. We maintain a planar representation $\mathcal{S}_i'$ based on the parameterization (Sec. 5.3) stored within a binary space partitioning (BSP) tree. The BSP tree allows efficient lookups while the mapping of $\mathcal{S}_i$ to the unit square ensures robustness. Given a quadrangulation $\mathcal{Q}$ of the unit square, the connectivity texture is projected onto $\mathcal{S}_i$ in $\mathcal{O}(mlog(n))$ time, where m is the number of quad vertices and n is the number of triangles in $\mathcal{S}_i$. The triangle $t' \in \mathcal{S}_i'$ containing a vertex $v \in \mathcal{Q}$ is found in $\mathcal{O}(log(n))$ time by virtue of the BSP tree. The projection of v to $\mathcal{S}_i$ is obtained based on its barycentric coordinates within t', computed on t.

While interactions described in the following section are performed in 3D-space, where the texture is mapped to $\mathcal{S}_i$, the underlying computations are performed on the unit square. The connectivity and vertex locations of the final quad mesh are stored as a texture. With the described projection methods, we are able to maintain interactive rates and guarantee smoothly interpolating projections during vertex movement and mesh subdivision operations.

7.1 Local User Interface

We support a collection of connectivity-based operators to interactively design and edit the quad elements as desired by the user's meshing paradigm. Initially the connectivity texture assigned to each Reeb chart region is the unit square (Fig. 10). The user designs a quadrangulation for each region by applying refinement, coarsening and improvement operations [23], [39], [11] to the quad elements.

Connectivity Operators. The user interacts with the connectivity texture design by simply selecting an element (or pairs of elements) for refinement and coarsening, illustrated by the time lapse in Fig. 10 and showcased in the accompanying video. We support global subdivision of the texture, user selected edge subdivision that initiates a polychord insertion, polychord deletion, cube-based subdivision for polycube-like meshing [40], [26], quad-open and -close operators, as well as quad-edge and vertex-edge flipping. Meanwhile, we maintain a history stack to undo/redo the specified operations. By inserting additional extraordinary vertices, the mesh can be designed to precisely adapt sample densities to complex geometry and better align with mesh features as compared to isocontouring (Fig. 10).

Vertex Movement. Via connectivity textures, the ability to locally modify the location of vertices of the final quad mesh is straightforward and robust. In a comparable manner to [34], the mouse movement is used to perturb the uv-mapping of selected vertices. The vertex re-projection onto the Reeb chart $\mathcal{S}_i$ is efficiently computed, and the small processing required to ensure that the reprojected point

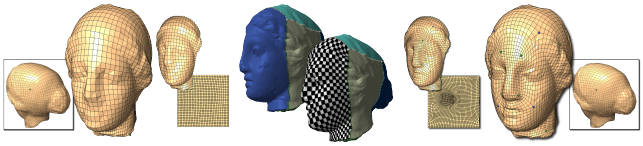


Fig. 11. A coarse base domain is extracted from an existing quad-mesh and parameterized (middle). The face is locally improved by connectivity texturing to better capture the nose (right) with no impact on the rest of the mesh (insets).

moves in the same screen space direction as the mouse is negligible, maintaining interactive rates. It is important to note that vertex movement simultaneously executes local relaxation to allow easy displacements of groups of vertices in a single mouse move, while ensuring orthogonality of the quads, identifying flags differentiate anchor vertices that remain unaffected by the smoothing (shown in the video).

After editing the connectivity textures of the different charts, the user still has the possibility to modify globally the Reeb atlas. However, it implies a reset of the connectivity textures of the updated Reeb charts (Sec. 6.1). Finally, note that connectivity texturing can be used in principle on top of any quad base domain, as shown in figure 11 for the purpose of localized interactive improvement.

7.2 Final Mesh Stitching

Because the quadrangulations are constructed individually for each Reeb chart region, the final composition of the mesh is handled with a post process stitching and localized vertex relaxation. The stitching algorithm greedily merges pairs of nearest boundary vertices. Two vertices, v_a and v_b, are merged if the distance between them, $d_{ab} = |v_a - v_b|$, is smaller than a ratio of the minimum distance to the neighboring boundary vertices, $d < \alpha d_a$ and $d < \alpha d_b$ (in practice $\alpha = 0.25$), where $d_a = min(|v_i - v_a|, |v_j - v_a|)$ and $d_b = min(|v_m - v_b|, |v_n - v_b|)$. The vertices $v_{i,j}$ and $v_{m,n}$ are the neighboring boundary vertices of v_a and v_b respectively. T-junctions may be present after the greedy stitching algorithm exhausts the possible vertex mergers by associated the non-merged vertices with nearby boundary edges as demonstrated in Fig. 12. Similarly addressed in

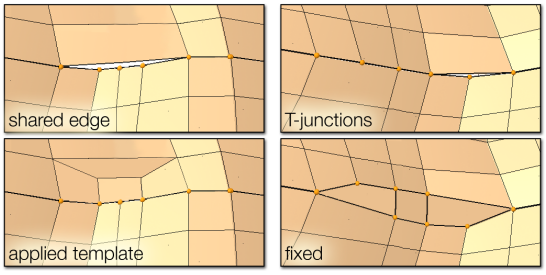


Fig. 12. T-junctions are resolved by the stitching process: inserting new quads where multiple T-junctions share a common edge (a), or performing mesh surgery along the path between two nearby triangles and inserting quads along the cut (b).

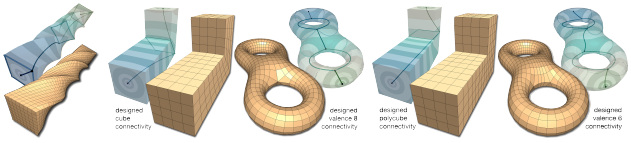


Fig. 13. Remeshing examples on primitive shapes. Our global and local approach is flexible, making it possible to design different quad connectivities and extraordinary vertex types over the same model.

geometry clipmaps [27] and rectangular multi-chart geometry images [8], we resolve such regions to develop a watertight mesh. Where clipmaps insert zero-area triangles and multi-chart geometry images use local remeshing of boundary triangles, we must couple T-junctions to ensure quad-only connectivity. Edge flips have been applied to merge nearby zero-area triangles [41], but this causes a twist in the mesh elements that negatively affects the alignment of mesh edges. Instead, we implement a greedy algorithm to resolve pairs of nearby zero-area triangles by inserting new quads between them. First, multiple T-junctions on a shared mesh edge are resolved by recursively applying the illustrated template (Fig. 12), refining the element whose edge the T-junctions share, such that any mesh edge contains at most one T-junction. The T-junctions then describe zero-area triangles on the mesh, and breadth-first traversals compute the set of shortest paths between mesh vertices belonging to pairs of these triangles. The mesh is cut along the shortest of these candidate paths, and new edges are inserted between the duplicated vertices to form a quad-only connectivity (Fig. 12). The breadth-first traversal and subsequent mesh surgery is repeated until all pairs of triangles are removed from the model in a greedy fashion. Given a closed manifold, it is guaranteed that there will be an even number of T-junctions (equivalently thought of as zero-area triangles). As such, it is possible to pair all T-junctions by quadrangulating the region between them, and construct a quad-only watertight mesh.

8 Discussion

Typical usage scenario. As described in the accompanying video, user interaction is required at two levels.

First, in the *global view*, the user places sparse segmentation inputs (corresponding to extrema of f) typically at the extremity of prominent features if he/she is unsatisfied with the automatic suggestion. The Reeb atlas is then automatically completed while guaranteeing the generation of an atlas made of charts with controlled topology. The user may decompose further the atlas by the addition or the subdivision of charts with click interactions. Also, a 3D widget is provided to edit the alignment of the boundaries of the chart, and consequently of their parameterization. Finally, the intersection points of the chart boundaries (corresponding to critical points of f) will correspond to extraordinary vertices in the final mesh. The user can edit the valence and location of those extraordinary vertices

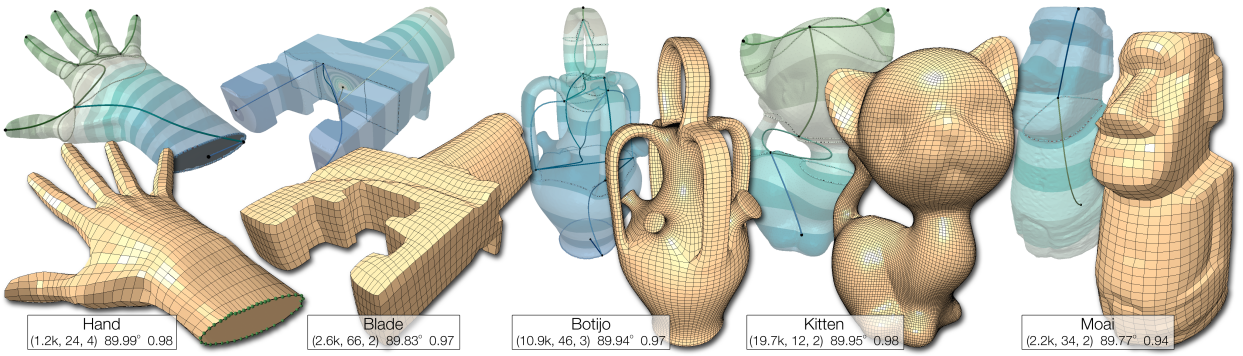


Fig. 14. Examples of user designed quad meshes generated with our framework accompanied by quality statistics (vertex count, extraordinary vertex count, max difference from the valence-4, the average mesh angle and scaled Jacobian).

with a set of curve editing operations applied on the chart boundaries as demonstrated in the accompanying video.

Second, when the user is satisfied with the Reeb atlas segmentation, *local views* of each Reeb chart are opened to design connectivity textures via subdivisions, deletions and element movements. Typically, our experiments showed that connectivity texturing was often achieved through a sequence of global subdivisions, possibly with intermediate cube subdivisions (Fig. 10). Finally, an automated stitching algorithm generates the final output mesh by composing the connectivity textures.

Notice that although topology aware scalar field design is a central technique in our approach, this aspect is totally hidden to the user who does not have to be knowledgeable about topology or scalar field design. The system inputs are solely focused on the users' quality criteria, determining the exact placement and valence of extraordinary vertices as well as localized edge alignment. These controls are specific and exact as the user does not abstractly affect the mesh through the modification of algorithmic parameters, i.e. eigenfunction selection nor boundary parameterization specifications. Because the Reeb graph can be a close approximation of the medial axis of the shape [26], it aids in the creation of a coarse quadrangulation for the model that captures dominant features with minimal amount of interaction. Also, the related theory of the Reeb graph provides important topological guarantees for our chart segmentation, enabling generic connectivity texture mapping.

8.1 Experimental Results

We implemented our interactive quadrangulation framework in C++ using the CHOLMOD libraries for our linear system solver [12]. The timings reported in this section are the results collected from experiments run on commodity desktop computers with Linux and MacOS. The duration of a quadrangulation session is variable (from a few seconds to several minutes for the models presented in this paper, cf. accompanying video), depending on the complexity of the input shape as well as the user skill and design exigence. Regarding the Reeb atlas editing, while the number of final extrema is fairly low in the presented outputs (from 2 to 5), the number of editing operations of the chart boundaries is related to the topology of the surface. For instance, the genus-5 Botijo atlas editing involved one reeb chart subdivision, 2 half-saddle splitting followed by 3 saddle alignment. More interestingly, and quantifiable, are the response timings for computations imposed by our system. The boxplots (Fig. 16), illustrating the median, minimum, maximum, as well as lower and upper quartiles, highlight the timings acquired from our experiments for the scalar field update, Reeb graph computation, Reeb chart parameterization, and connectivity texture subdivision and coarsening operations. The median timings are well below 0.5s, allowing for real time interaction. We showcase several models generated using our approach in Figs. 13, 14, and 15, illustrating important features of our framework. The L-shaped meshes, with cube-like and polycube-like connectivity, and the bitori meshes, illustrating a single saddle vertex with valence 8 versus two half-saddles with valence 6, spotlight the design flexibility (Fig. 13). The twisted bar (Fig. 13) illustrates the orientation and alignment control of the Reeb atlas. The Moai and Bimba models showcase the removal of polar vertices from genus-0 models. The Botijo model highlights the advantage of the Reeb atlas abstraction, capable of managing a user-driven segmentation of a model with complex topology, while providing topological guarantees necessary for parameterization. Our framework handles mesh boundaries by either constraining a contour of f along the boundary, i.e., the Hand, or by filling the boundary then removing quads from the texture.

In many cases, especially when the Reeb chart describes a cylindrical mesh component, regular subdivision provides fast and easy high-quality quadrangulations of the Reeb charts, i.e., the Botijo handles, the Hand's fingers, the torso's of the Moai and Bimba, as well as pieces

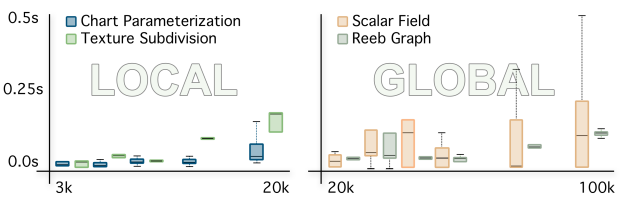


Fig. 16. Boxplot response timings of our system computations (in seconds, wrt the number of input triangles).

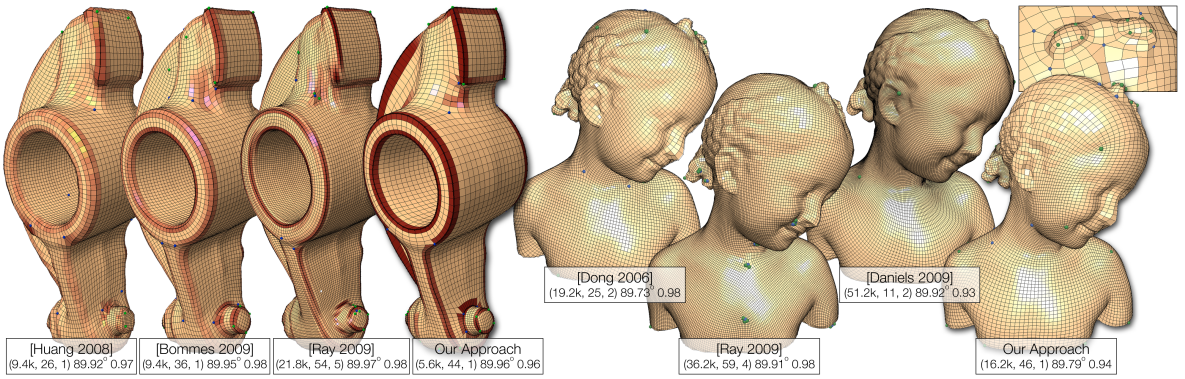
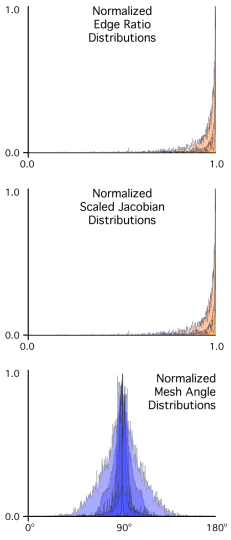


Fig. 15. The comparison of our technique against quad meshes from multiple algorithms illustrates our improved control of extraordinary vertices, i.e. location and valence, as well as element alignment, while producing quality output models for the Rocker Arm and Bimba models (reporting the same quality statistics as Fig. 14). Extraordinary vertices with a valence greater or less than four are respectively reported with blue and green spheres. On the Rocker Arm model, each quadrilateral is color mapped based on the max angle between its normal and its neighbors' normals, from yellow ($0°$) to dark red ($90°$). Notice the alignment of the extraordinary vertices and the alignment to sharp features.

of the Blade and Rocker Arm models. The added flexibility of connectivity texturing improves the alignment of the final mesh to surface features (sharp features on the the Rocker arm, Botijo and Blade models) as well as user designed adaptivity to better sample high frequency geometry (the Bimba's bow, zoom inset in Fig. 15). The number of extraordinary vertices is strictly controlled by the user, and can be maintained to a desired value. The quality of the meshes used throughout this paper is measured in the inset histograms. The quality of the output is dependent on the user's diligence, but, as demonstrated by the histograms, high quality meshes that are numerically stable for finite element simulations can be generated using our system.



8.2 Comparison

In this section, we compare our *interactive* approach to automatic [10] and semi-automatic techniques [13], [21], [6], [36] for a mechanical and organic models (Figs. 15 and 17). Included in these visual comparisons are quality statistics of the assorted models, demonstrating that our approach generates quad meshes with objective quality scores that are on par with other state-of-the-art techniques. More interestingly, the visual comparison highlights advantages of our approach which are not directly reflected by objective quality measurements, such as: sharp feature preservation, extraordinary vertex alignment and localized adaptative sampling. A key advantage of our approach is its ability to robustly control the number, location, valence and alignment of the extraordinary vertices. In contrast, other techniques (Figs. 15 and 17) provide output meshes that *approximate* the user's input constraints. As a result, these methods may produce undesireable effects related to extraordinary vertices, such as inaccurate approximation of feature corners, misalignment of the emminating mesh edges, and extraordinary vertex clustering. By strictly designing our meshes to align extraordinary vertices through straight edge paths, in contrast to [6], [36] but similar to [13], [21], [10], we design quad models that are conducive to texturing, coarsening and smooth surface fitting via subdivision and spline-based surfaces.

As illustrated in Figs. 15 and 17, the extraordinary vertices of our Rocker Arm, Bimba and Botijo models are well aligned, placed at strategic locations, and the mesh edges between them are aligned to the sharp features of the models. In particular, we focus the reader's attention to the protruding corners on the hammer component of the Rocker Arm where our approach exactly captures these elements; as well as the bottom of the Bimba model to which our edges are aligned and extraordinary vertices placed within corner regions. In contrast, note the quad strip *twisting* that occurs across sharp features with the other techniques.

Another major distinction between our approach and the semi-automated techniques is our ability to design a configuration that captures the symmetry of the model that may be difficult to quantify by (semi-) automatic algorithms (Fig. 17). For example, the Bimba model contains symmetric

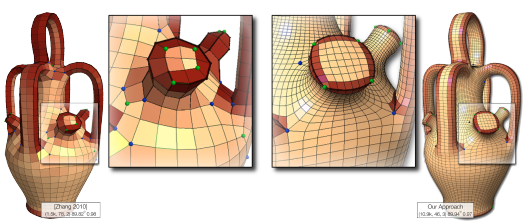


Fig. 17. Comparison to [47]. Our approach enables to generate a mesh with fewer extraordinary vertices, aligned and placed at specific locations, respecting local symmetries.

extraordinary vertices, i.e., the face and shoulders, with edges aligned to the model's curvature, i.e., the face, neck and chest. Further, the design of the connectivity textures on a local scale allows the user to adapt the sample density, capturing high frequency geometric detail and symmetry of the bow in the Bimba's hair (zoom inset in Fig. 15).

8.3 Limitations

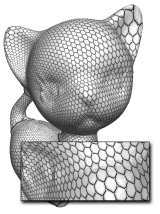
An important observation of this work is the link between extraordinary vertices and critical points of the underlying scalar field, handled by the Reeb atlas segmentation algorithm. The system requires training of users to understand the connections between the interactions on the Reeb atlas and the resulting extraordinary vertices, as well as becoming familiar with the interaction tools. Our current interface can benefit from more straightforward control mechanisms to facilitate user interactions. However, as these aspects of the user interface do not affect the discussed underlying technologies, we view this as out of the scope of this research. Also, although our approach enables to split scalar field critical points into many types of extraordinary vertices (with valence 2, 3, 5, 6 and more on monkey saddles), a formal investigation of the exact output space of scalar-field based meshing may provide concrete insights about the theoretical expressiveness of the Reeb atlas framework. However, we believe such a theoretical study goes beyond the scope of this paper.

Finally, optimal alignment of quad meshes often requires adaptive sampling, introducing many extraordinary vertices. While this is completely feasible within our framework (demonstrated in multiple models in this paper), it can become time consuming to design and manipulate such models.

9 Conclusion and Future Work

Our technique bridges the gap between purely geometrical approaches and combinatorial connectivity techiques to leverage advantages of the two distinct worlds within one coherent system. This study provides interesting insights, linking scalar field topology to extraordinary vertices and their global alignment. Our Reeb atlas, the mechanism by which we induce the alignment and construct a coarse quadrangulation of the model, enriches scalar field design by providing topological structure and awareness. We develop a multi-level methodology that, in addition to global Reeb atlas updates, supports local editing operations via connectivity textures to explicitly define the final mesh structure. Reeb atlas and connectivity textures are two complementary tools, with partial overlapping scopes, that uniquely provide global and local controls (respectively). Designing a complex Reeb atlas will tend to allow simple connectivity textures (the Hand, Bitorus and Botijo, Fig.14); whereas, designing a simple Reeb atlas may require complex connectivity textures (RockerArm, Fig. 10). Our connectivity textures completely localize the global effects of quadrangulation design, limited to a single Reeb chart. Relying on the topological information provided by the Reeb atlas, our framework is able to resolve conflicts between regions meshed with different sample densities. The local operations are performed over the unit square for efficient and robust computation and projection. We demonstrate the interactive (response times below 0.5s) and flexible nature of our approach throughout the paper.

This paper focuses on developing underlying technologies that provide the flexibility, interactivity and robustness required by a user-centric meshing process. Based on the generality and flexibility of our framework, in future work we intend to enrich our system's interface with additional automated user assistances to augment the designer's productivity. At a global scale, improved heuristics may suggest better initial Reeb atlases, possibly providing hints that contain aligned and/or fractional saddles. At a fine scale, geometrical analysis in the parametric domain of individual Reeb charts can lead to automated initial geometry-aware connectivity textures, on which a user may interactively edit. Finally, it will be interesting to further explore the full potential of connectivity textures to design meshes with arbitrary polygons, i.e., hexagons (right), as well as extensions to volumetric shape representations.



Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. We thank Chems Touati for his help for the accompanying video. We also thank the authors of the following papers [21], [6], [36], [13], [47], who kindly provided their data-sets. This research has been funded by Fapesp-Brazil (#2008/03349-6), CNPq-NSF (#491034/2008-3), NSF (grants IIS-0905385, IIS-0844546, CNS-0855167, IIS-0746500, ATM-0835821, CNS-0751152, IIS-0713637, OCE-0424602, IIS- 0534628, CNS-0514485, IIS-0513692, CNS-0524096), DoE, and IBM Faculty Awards.

References

[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Trans. Graph. (Proc of SIGGRAPH)*, 22(3):485–493, 2003.

[2] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. In L. D. Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*, Mathematics and Visualization. Springer Berlin Heidelberg, 2008.

[3] M. Ben-chen, C. Gotsman, and G. Bunin. Conformal flattening by curvature prescription and metric scaling. *Comp. Graph. Forum (Proc. of Eurographics)*, 27(2):449–458, 2008.

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

[5] I. Boier-Martin, H. Rushmeier, and J. Jin. Parameterization of triangle meshes over quadrilateral domains. *Symp. on Geom. Proc.*, pp. 193–203, 2004.

[6] D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. *ACM Trans. Graph. (Proc of SIGGRAPH)*, 28(3):1–10, 2009.

[7] P. Bremer, S. Porumbescu, B. Hamann, and K. Joy. Automatic semi-regular mesh construction from adaptive distance fields. In *Curve and Surface Fitting*, 2002.

[8] N. Carr, J. Hoberock, K. Crane, and J. Hart. Rectangular multi-chart geometry images. In *Symp. on Geometry Proc.*, pp. 181–190, 2006.

[9] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. In *ACM Symp. on Comp. Geom.*, pp. 344–350, 2003.

[10] J. Daniels, C. Silva, and E. Cohen. Semi-regular quadrilateral-only mesh generation from simplified base domains. *Comp. Graph. Forum.*, 28(5):1427–1435, 2009.

[11] J. Daniels, C. T. Silva, J. Shepherd, and E. Cohen. Quadrilateral mesh simplification. *ACM Trans. Graph. (Proc of SIGGRAPH Asia)*, 27(5):1–9, 2008.

[12] T. Davis and W. Hager. Dynamic supernodes in sparse cholesky update/downdate and triangular solves. *ACM Transactions Mathematical Software*, 35(4):1–23, 2009.

[13] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. Hart. Spectral surface quadrangulation. *ACM Trans. Graph. (Proc of SIGGRAPH)*, 25(3):1057–1066, 2006.

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

[15] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28:511–533, 2002.

[16] H. Edelsbrunner, D. Morozov, and V. Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *ACM Symp. on Comp. Geom.*, pp. 127–134, 2006.

[17] H. Edelsbrunner and E. P. Mucke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. on Graph.*, 9:66–104, 1990.

[18] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[19] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *SIGGRAPH*, pp. 203–212, 2001.

[20] K. Hormann, K. Polthier, and A. Sheffer. Mesh parameterization: theory and practice. In *ACM SIGGRAPH ASIA 2008 courses*, pp. 1–87, 2008.

[21] J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, and H. Bao. Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph. (Proc of SIGGRAPH Asia)*, 27(5):1–9, 2008.

[22] F. Kalberer, M. Nieser, and K. Polthier. Quadcover: Surface parameterization using branched coverings. *Comp. Graph. Forum (Proc. of Eurographics)*, 26:375–384, 2007.

[23] P. Kinney. Cleanup: Improving quadrilateral finite element meshes. In *Proc. of the 6th International Meshing Roundtable*, pp. 437–447, 1997.

[24] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *ACM SIGGRAPH*, pp. 313–324, 1996.

[25] Y.-K. Lai, L. Kobbelt, and S.-M. Hu. An incremental approach to feature aligned quad dominant remeshing. In *ACM Symp. on Solid and Phys. Modeling*, pp. 137–145, 2008.

[26] J. Lin, X. Jin, Z. Fan, and C. Wang. Automatic polycube maps. In *Proc of Geom Model and Process*, pp. 3–16, 2008.

[27] F. Losasso and H. Hoppe. Geometry Clipmaps: Terrain rendering using nested regular grids. *ACM Trans. Graph. (Proc of SIGGRAPH)*, 23:769–776, 2004.

[28] J. Milnor. *Morse Theory*. Princeton University Press, 1963.

[29] S. Owen, M. Staten, S. Canann, and S. Saigal. Q-morph: An indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering*, pp. 1317–1340, 1999.

[30] V. Pascucci, G. Scorzelli, P. T. Bremer, and A. Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, 26:58.1–58.9, 2007.

[31] G. Patanè, M. Spagnuolo, and B. Falcidieno. Para-graph: Graph-based parameterization of triangle meshes with arbitrary genus. *Comp. Graph. Forum.*, 23:789–797, 2004.

[32] G. Patanè, M. Spagnuolo, and B. Falcidieno. A minimal contouring approach to the computation of the reeb graph. *IEEE Trans. on Vis. and Comp. Graph.*, 15:583–595, 2009.

[33] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Exp. Math.*, 2:15–36, 1993.

[34] D. Pinskiy. Sliding deformation: shape preserving per-vertex displacement. In *Eurographics (short papers)*, 2010.

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

[36] N. Ray, B. Vallet, L. Alonso, and B. Lévy. Geometry aware direction field design. *ACM Trans. on Graph.*, 2009.

[37] G. Reeb. Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes-rend. de l'Acad. des Scien.*, 222:847–849, 1946.

[38] O. Schall, R. Zayer, and H.-P. Seidel. Controlled field generation for quad-remeshing. In *ACM Symp. on Solid and Phys. Modeling*, pp. 295–300, 2008.

[39] M. Staten and S. Canann. Post refinement element shape improvement for quadrilateral meshes. In *ASME AMD: Trends in Unstructured Mesh Generation*, pp. 9–16, 1997.

[40] M. Tarini, K. Hormann, P. Cignoni, and C. Montani. Polycube-maps. pp. 853–860, 2004.

[41] M. Tarini, N. Pietroni, P. Cignoni, D. Panozzo, and E. Puppo. Practical Quad Mesh Simplification. *Comp. Graph. Forum (Proc. of Eurographics)*, 2010.

[42] J. Tierny, A. Gyulassy, E. Simon, and V. Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Trans. on Vis. and Comp. Graph. (Proc. of VIS)*, 15:1177–1184, 2009.

[43] J. Tierny, J.-P. Vandeborre, and M. Daoudi. Partial 3D shape retrieval by reeb pattern unfolding. *Comp. Graph. Forum.*, 28:41–55, 2009.

[44] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Designing quadrangulations with discrete harmonic forms. In *Symp. on Geom. Proc.*, pp. 201–210, 2006.

[45] N. Viswanath, K. Shimada, and T. Itoh. Quadrilateral meshing with anisotropy and directionality control via close packing of rectangular cells. In *Proceedings of 9th International Meshing Roundtable*, pp. 227–238, 2000.

[46] K. Xu, H. Zhang, D. Cohen-Or, and Y. Xiong. Dynamic harmonic fields for surface processing. *Comput. Graph.*, 33(3):391–398, 2009.

[47] M. Zhang, J. Huang, X. Liu, and H. Bao. A wave-based anisotropic quadrangulation method. *ACM Trans. Graph. (Proc of SIGGRAPH)*, 29, 2010.

Julien Tierny received the Ph.D. degree in computer science from Lille 1 University in October 2008. He is currently a CNRS permanent researcher at Telecom ParisTech, Paris. Prior to his CNRS tenure, he held a Fulbright fellowship (U.S. Department of State) and was a postdoctoral research associate at the Scientific Computing and Imaging Institute at the University of Utah. His research interests include topological and geometrical spatial data analysis for scientific visualization and computer graphics.

Joel Daniels II received a B.S. in Computer Science from the University of New Hampshire in 2003 graduating summa cum laude, and a Ph.D. in Computer Science from the University of Utah in 2010. He is currently working as a post-doctoral researcher at the NYU Polytechnic Institute where his research interests and activities have focused on user-guided geometry modeling, scientific visualization and data browsing.

Luis Gustavo Nonato received the Ph.D. degree from the Pontifcia Universidade Catlica do Rio de Janeiro, Rio de Janeiro, in 1998. He joined the University of So Paulo in 1999 and has been an associate professor since 2006. His research interests include visualization and geometry processing. His teaching activities include geometry processing, numerical analysis, and geometric modeling.

Valerio Pascucci is an Associate Professor of the School of Computing and Scientific Computing and Imaging Institute at the University of Utah since 2008. Before joining SCI, Dr. Pascucci served as the Data Analysis Group Leader and as a Project Leader at the Lawrence Livermore National Laboratory, Center for Applied Scientific Computing (from May 2000) and Adjunct Professor at the Computer Science Department of the University of California Davis (from July 2005). Prior to his CASC tenure, he was a senior research associate at the University of Texas at Austin, Center for Computational Visualization, CS and TICAM Departments. Dr. Pascucci's research interests include: large data management and analysis, topological methods for image segmentation, progressive and multi-resolution techniques for scientific visualization, combinatorial topology, geometric compression, computer graphics, applied computational geometry, and solid modeling.

Claudio T. Silva is Professor of computer science and engineering at NYU's Polytechnic Institute. Previously, he was a full professor of computer science and a faculty member of the Scientific Computing and Imaging Institute at the University of Utah. He received his Ph.D. in computer science at SUNY-Stony Brook in 1996. He coauthored more than 160 technical papers and eight US patents, primarily in visualization, geometric computing, scientific data management, and related areas. He has served on more than 80 program committees, and he is currently in the editorial board of Computing in Science and Engineering, Computer and Graphics, The Visual Computer, and Graphical Models. He was general co-chair of IEEE Visualization 2010, and papers co-chair of VIS 2005 and 2006. He received IBM Faculty Awards in 2005, 2006, and 2007, and best paper awards at IEEE Visualization 2007, IEEE Shape Modeling International 2008, the 2010 Eurographics Educator Program, and the 11th Eurographics Symposium on Parallel Graphics and Visualization 2011. His work is funded by grants from the NSF, NIH, DOE, IBM, and ExxonMobil.