Inspired quadrangulation

Julien Tierny, Joel Daniels, Gustavo Nonato, Valerio Pascucci, Claudio Silva

Inspired Quadrangulation

Julien Tierny *,[1] Joel Daniels II [2] Luis G. Nonato [3] Valerio Pascucci [2] Claudio T. Silva [2,4]

Abstract

This paper presents a new approach for the quadrangulation of triangular surfaces. While previous work focused on fully automatic computations or explicitly involved user control for the integration of subjective decisions, we introduce a new example-based quad-meshing paradigm, in order to easily reproduce the subjective decisions made in the design of reference examples found in a corpus. The algorithm enables to reproduce the subjective aspects of the example (extraordinary vertex layout) while minimizing the induced distortion; allowing users to leverage reference meshes considered of quality for fast prototyping. At the core of our technique, we provide the analytic gradient of as-rigid-as-possible 2D transformations. This expression is implemented in a fast solver to automatically register planar unfoldings of the geometries, yielding low-distortion cross maps of the examples onto the input in 3D. In addition, our technique provides interactive feedback for user modifications of the cross maps computed automatically. Our technique supports localized mesh composition and enables to reproduce meshing styles despite intrinsic reflective symmetry, large variation from isometry or even topological variation. Experiments demonstrate the accuracy of the mimicking process as well as its time efficiency.

Key words: Surface quadrangulation, data-driven geometric modeling, cross-paramaterization

1. Introduction

Surface quadrangulations are desirable shape representations in computer graphics as their regular structure benefits many geometry processing tasks. In many contexts, however, surfaces are represented with meshes made of arbitrary polygons. Thus, the generation of quadrangulations from raw polygonal surfaces is an important task that has received much attention by the research community in the last few years. Generating surface quadrangulations is still however a very challenging problem because many quality criteria of different nature intervene in the evaluation of the output. *Low-level* quality criteria (edge angles or element planarity) are important for texture mapping or architectural modeling. Whereas, *high-level* quality criteria (alignment to *'shape features'* and layout of the extraordinary vertices) are important subjective design decisions for application-specific tasks, i.e. producing realistic character animation. In this example, the placement of the extraordinary vertices and alignment of quadrilateral elements plays an important role in achieving visually desirable results. The subjective design decisions are often conditioned by physically plausible considerations (for instance to roughly model the character's muscles) and may be unrelated to the underlying object geometry. Consequently, in many contexts, quad-remeshing becomes not only a geometry processing task but also a subjective modeling task. While previous automatic techniques [1–3] often produce outputs with outstanding *low-level* quality metrics, they do not integrate *high-level* constraints related to the subjective aspects of the mesh structure. Therefore, in a recent past, many approaches [4,6–11] have been proposed to address this problem by allowing users to intervene in the meshing process to specify structural (extraordinary vertices) and/or high level geometrical (feature alignment) constraints. However, user-driven generation of *good* quadrangulations requires an understanding of both the application requirements and of the meshing process. As such, users need to develop expertise with the software tools and with the concepts involved in quadrangulation design.

In this work, we present a drastically different approach to surface quadrangulation, called *Inspired Quadrangulation*, which aims at reproducing the meshing style of existing quad-meshes found in a corpus of examples (considered of satisfying subjective quality by the users). Our approach supports localized com-

* Corresponding author's address: LTCI, 46 Rue Barrault, Paris, France.
Email addresses: tierny@telecom-paristech.fr (Julien Tierny), jdaniels@sci.utah.edu (Joel Daniels II), gnonato@icmc.usp.br (Luis G. Nonato), pascucci@sci.utah.edu (Valerio Pascucci), csilva@nyu.edu (Claudio T. Silva).
[1] CNRS - LTCI - Telecom ParisTech
[2] SCI Institute - University of Utah
[3] ICMC - Universidade de Sao Paulo
[4] NYU-Poly

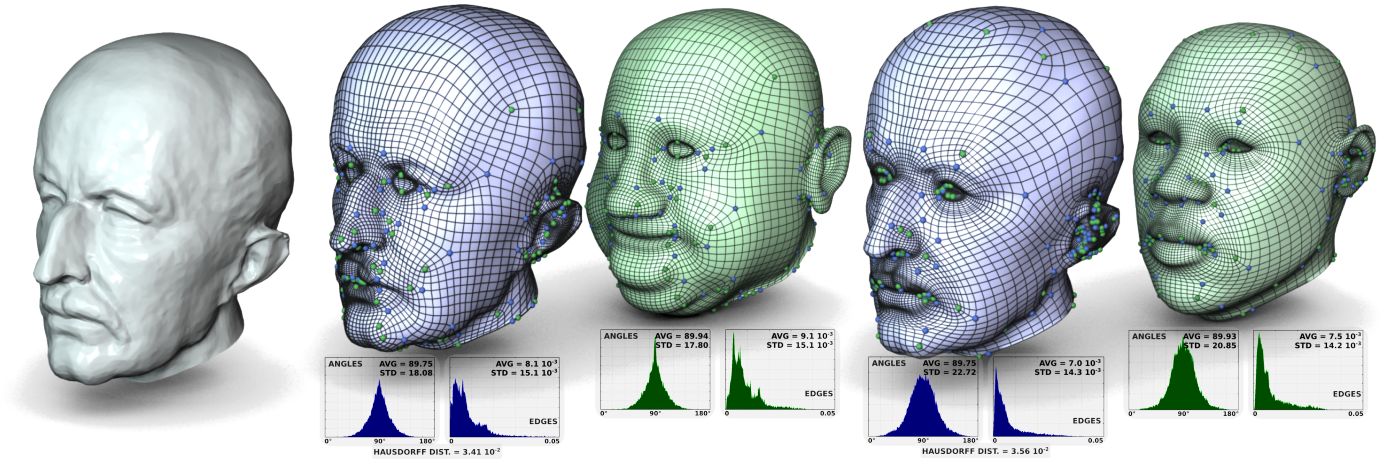


Fig. 1. Generating *inspired quadrangulations* on a scanned geometry. Given an input triangle mesh (left), the presented technique generates output quad meshes (in blue) by mimicking the meshing style of geometrically and subjectively relevant meshes (in green) found in a corpus of examples. The process reproduces the subjective aspects of the example (layout of the extraordinary vertices, blue and green spheres) while enforcing the preservation of its geometrical properties (histograms). Examples that best fit the input geometry are then presented to the user and mapped onto the input. In this example, the entire corpus ranking and mapping onto the input geometry took 4.86 seconds without user intervention.

position and enables users to remesh input surfaces with parts coming from examples with possibly different topology, while producing a manifold quad-only output. We formulate the problem of data-driven quad-remeshing as a massive, geometry-aware, cross-parameterization task and present a new automatic algorithm computing sequences of low distortion cross-maps at interactive rates. Experiments demonstrate the accuracy of the meshing style reproduction as well as its time efficiency.

Contributions This paper makes the following contributions:

(i) To the best of our knowledge, the first data-driven quad-meshing technique, enabling users to rapidly generate quadrangulations by leveraging reference meshes;

(ii) A fast and automatic cross mapping algorithm that drives the searching and ranking of candidate corpus examples to be mapped onto the input;

(iii) The analytic expression of the gradient of the rigid *Moving Least Squares Deformations* [5] (supplemental material), allowing their usage in fast iterative solvers for as-rigid-as-possible 2D registration.

In details, at the core of our technique, we describe a fast cross mapping algorithm which leverages recent as-rigid-as-possible 2D deformation techniques [5], yielding low distortion cross maps in 3D in a pose and symmetry oblivious manner. We provide the expression of their gradient, resulting in a fast optimization process. The algorithm is highly parallelizable and uses a multi-resolution strategy which allows for time execution constraints (to trade between quality and speed). Finally, since our algorithm uses a closed-form formula for candidate map evaluation, it supports at interactive rates optional user enhancements of the optimal cross map found automatically.

2. Related Work

While this paper addresses surface quadrangulation, the presented technique involves multiple research areas. In this section, following a discussion of state-of-the-art quad meshing algorithms, we discuss work related to data-driven creative modeling and cross-parameterization.

Quadrangulation Techniques Many quad-meshing techniques have been proposed in the past. While some of them are automatic [1–3], most of the recent approaches drive the quad-meshing with *geometrical* and/or *structural* constraints, in order to integrate user decisions in the process. Geometrical constraints can be given as an input direction field defined on the surface [6,7] or as sparse directional constraints later interpolated [8–11] prior to integration and mesh extraction. Structural constraints can be provided with a singularity graph [4], describing the layout of the extraordinary vertices. Singularity graphs can also be inferred and influenced by the user, from the Morse-Smale complex of user-constrained eigenfunctions of the Laplacian [8,11]. Polycubes are another intermediate representation for such user control [12]. We refer the reader to a recent survey for further discussion on the topic [13].

While these user-driven techniques can generate outputs with good low-level quality scores and user prescribed constraints, they still require an amount of user intervention directly related to the desired complexity of the output. For instance, generating a singularity graph or directional constraints for the complex meshes shown in Fig. 1 (with more than 200 extraordinary vertices) would require significant user interaction. More importantly, this process still requires advanced modeling skills from the user, who needs to be knowledgeable about application requirements as well as concepts of quadrangulation design. In contrast, our approach allows users to leverage the knowledge and expertise of previous users by mapping regions of reference meshes onto the input.

Data-Driven Modeling At the global scope, our approach follows the directions initiated by previous data-driven modeling work [14,15]. In these techniques, geometry-based searches are performed against collections of 3D shapes and compositions with retrieved objects' parts are suggested to the user. The

problem addressed in this paper is different. It requires not only to rank the collection of examples by similarity to the input, but also to provide decent cross-maps between the input and the examples of the collection. In this paper, we formulate both the corpus ranking and cross-parameterization as a single process, where the isometric error induced by the automatic cross-map is used to rank the corpus.

Cross-Parameterization Many algorithms have been proposed for cross-parameterization in the past. Given a set of corresponding landmarks between two surfaces of equivalent topology, existing techniques aim at interpolating this correspondence all over the surfaces. Some algorithms have been specialized for distinct topological classes of surfaces, for instance: for discs [16,17], for spheres [18], for non genus-0 surfaces [19]. Other algorithms, using base meshes/domains [20–22] or surface fitting techniques [23], handle surfaces of arbitrary topology; however, these surface must be homeomorphic. While they achieve cross-maps with low distortion, the main drawback of these algorithms is that they depend on landmark correspondences provided by the user; typically 10 or more for models of heads for instance (while the result presented in Fig. 1 was generated without user correspondences). While comparing to a corpus of models, ensuring identical landmarks across all models is an arduous task. Automatic computation of reliable landmark correspondences is a challenging problem [24,25]. While some techniques explore in a Monte-Carlo fashion the space of conformal canonical cross-maps from which near isometries have to be identified [24], our algorithm directly explores the *smaller* space of as-rigid-as-possible (ARAP) cross-maps. Moreover, since we introduce the analytic gradient of planar ARAP transformations, we present a fast iterative solver based on multi-resolution gradient descent, which enables computations for interactive usage (a few seconds).

3. Overview

We formulate data-driven quad-meshing as a massive, geometry-aware, cross-parameterization task. Upon initialization, our framework requires a corpus of example meshes, whose quality is deemed satisfactory by the user.

First, each example surface is unfolded to a canonical domain. Its connectivity is indexed in a binary space partition of the planar domain while its geometry is represented by a multi-resolution 2D encoding, called *quasiconformal surface encoding* (Sec. 4.2). Additionally, the user can annotate the example with a subjective quality score in order to prioritize it at query time. We study the variability of this quasiconformal encoding with regard to several types of 3D transformations (Sec. 5.1) and present a fast iterative solver that registers the 2D encodings while minimizing the distortion of the cross maps in 3D (Sec. 5.2).

Second, when querying the corpus, the quasiconformal encoding of the input is computed and registered by the solver with all those found in the corpus. Examples are then presented to the user, ranked by increasing isometric error, and the examples are instantaneously mapped on the input upon user selection, based on the registration of the encodings. Also, the user can enhance the automatic cross map at interactive rates (Sec. 5.3).

Our technique supports localized mesh composition, in order to re-use mesh patches coming from examples having distinct topology or to drive the process on a per region-of-interest basis. In that case, the examples are pre-segmented and each segment is stored in the corpus individually. The input is also segmented in a compatible manner and each segment serves as a query to the system. Then, all the example patches mapped on the input are stitched into a manifold quad-only output (Sec. 6).

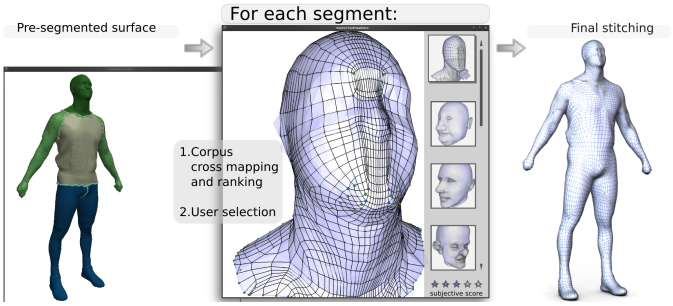


Fig. 2. Usage scenario with localized mesh composition.

4. Populating the Corpus

To ensure a fast and generic processing of the corpus, we introduce a concise canonical surface encoding. It enables to efficiently represent each entry of the corpus as a 2D picture, allowing for fast geometry aware cross map computations. The corpus examples and the input shape both undergo this unfolding process, *off-line* and *on-line* respectively.

4.1. *Topological Prerequisites*

Since two surfaces need to be homeomorphic to be cross-mapped, we require the input and the corpus examples to have a predefined topology. If this is not the case, we pre-segment them into topological primitives (as done in the case of mesh composition). In practice, we use the algorithm described in [26] since it performs semi-automatic segmentation such that the output is composed of *discs* and *cylinders* only. Then each segment can be processed individually without any further sophisticated topological considerations, which allows to handle examples or arbitrary topology.

In the remaining, we describe the algorithm in the case of topological discs and show how to extend it to other topological primitives at the end of Sec. 5.1.

4.2. *Quasiconformal surface encoding*

Let $\mathcal{S}$ be a PL 2-manifold with disc topology embedded in $\mathbb{R}^3$. We call a $k-$*quasiconformal map* an application $\phi : \mathcal{S} \rightarrow \mathcal{D}$

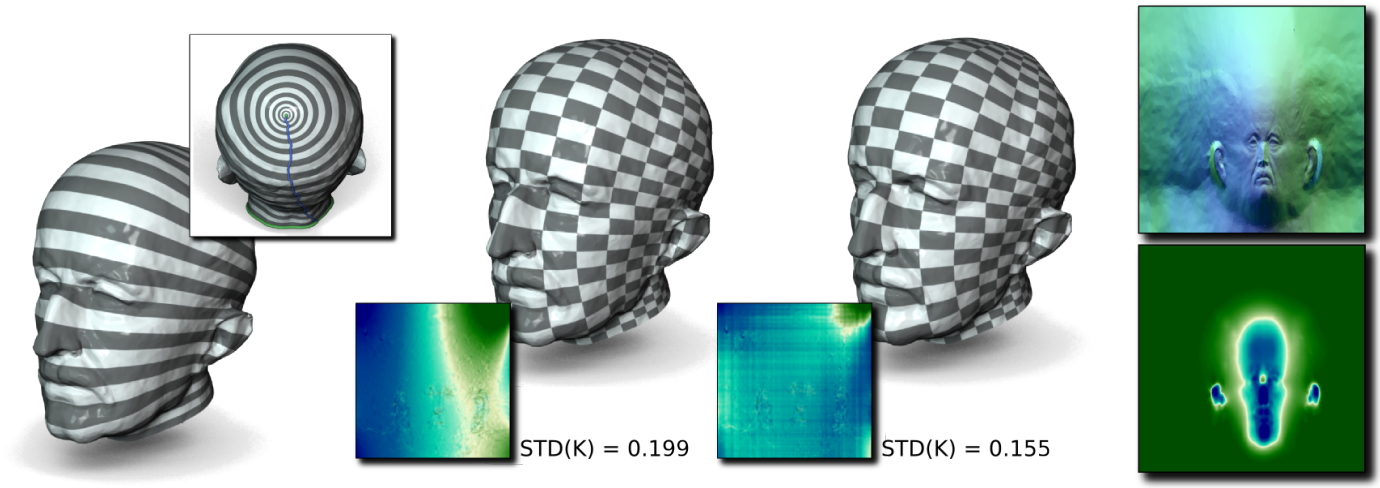


Fig. 3. Geometry unfolding procedure. From left to right: initial harmonic field (inset: pole and streamline constraints), parameterization without gradient norm adjustment (inset: k values in the planar domain), parameterization with gradient norm adjustment, unfolded normal map (top) and unfolded conformal factor (bottom). This generic unfolding procedure enables to reduce the problem of geometry aware cross parameterization to that of picture registration.

from $\mathcal{S}$ to a canonical (u, v) planar domain $\mathcal{D} \subset \mathbb{R}^2$ such that the following equalities hold for each point of $\mathcal{S}$:

$$\langle \nabla u, \nabla v \rangle = 0 \quad (1)$$

$$||\nabla u|| = k \, ||\nabla v|| \quad (2)$$

ϕ maps local circles on $\mathcal{S}$ to local ellipsoids on $\mathcal{D}$ (and reciprocally) and is conformal if $k = 1$.

For some given boundary constraints $\phi|_{\partial\mathcal{S}}$, the *quasiconformal encoding* of $\mathcal{S}$ consists of the normalized conformal factor induced by ϕ: the scalar field $\lambda(u, v) : \mathcal{D} \rightarrow \mathbb{R}$ which measures the normalized area distortion of the unfolding. The area of each triangle in 3D is first normalized by the overall surface area in 3D and the conformal factor is the ratio between this normalized area and the area of the triangle once unfolded.

Quasiconformal Unfolding We opt for a polar configuration on the boundary conditions for ϕ. Introducing singularities is known to reduce distortion [27], which contributes to a more uniform sampling of the surface (Fig. 4), and enables trivial extensions to other topological primitives (Sec. 5.1). The v coordinate is computed by solving a Laplace equation ($\Delta v = 0$) with the boundary conditions $v(\partial\mathcal{S}) = 0$ and $v(\mathcal{P}) = 1$ where $\mathcal{P}$ is the pole singularity. The pole can be extracted at the maximum of the conformal factor induced by a singularity-free parameterization (Fig. 4) or provided at segmentation time.

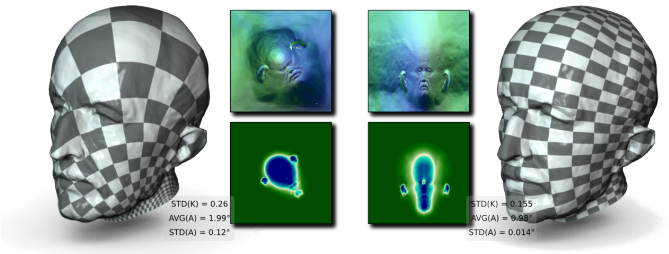


Fig. 4. Geometry unfolding without (left) and with singularity (right). The introduced singularity *absorbs* the conformal factor, improving the statistics.

Next, we cut $\mathcal{S}$ along an arbitrary streamline of v initiated at the pole and constrain each side of the cut to $u = 0$ and $u = 1$. The u coordinate is also computed by solving a Laplace equation ($\Delta u = 0$) with the additional constraint of orthogonality on the gradients ∇u and ∇v. The Laplace equation is solved by using the linear solver described in [28] (with the cotangent weight discretization of the Laplacian) while the orthogonality is integrated in the solve by minimizing the dot product of ∇u and ∇v in a least-square sense [29]. Constraining the dot product of the gradients enforces their orthogonality; however, it leaves the vector norms free. To obtain a uniform sampling on the surface (constant k value), we use the following algorithm.

Because the orthogonality of the gradients is already achieved, one now needs to *push* and *pull* the u level lines such that they become more evenly distributed on the surface. The vertices of $\mathcal{S}$ are visited by increasing u value. For each vertex V_n, its k value (noted k_{V_n}) is approximated on its triangles (weighted average based on areas). At the first visited vertex V_0, we set $\overline{u}(V_0) = u(V_0)$. For the others, we compute $\overline{u}$ as follows:

$$\overline{u}(V_n) = \overline{u}(V_{n-1}) + \frac{(u(V_n) - u(V_{n-1}))}{k_{V_n}} \quad (3)$$

The final u coordinate is computed by normalizing $\overline{u}$. In affect, this simple algorithm *pushes* the u level lines where k is low and *pulls* them where k is high. In practice, this procedure affected ϕ by changing the *values* of the level lines without drastic changes to their shape, preserving the near orthogonality of ∇u and ∇v. Next, this algorithm is repeated for the v coordinate. Note that for topological discs ∇v vanishes in theory at the pole and k is ill-defined around it. To avoid instabilities when adjusting ∇v, we artificially set $k = 0.1$ for the vertices for which $v > 0.9$. In practice, this heuristic nicely distributes the samples in the pole's immediate neighborhood. At the end of this process, k is approximately constant over the domain and we associate to $\mathcal{S}$ a unique k value, noted $k_{\mathcal{S}}$ (average of k on all the triangles).

Interpretation The advantage of using $k-$quasiconformal maps is twofold. First, since ∇u and ∇v are orthogonal, we will be able to compensate any variation in the direction of the streamline cut with a simple translation in the u component (as described in Sec. 5.1). Second, the gradient norm adjustment enables to constrain the unfolding to a perfect square with a uniform sampling, which enables to derive a generic algorithm for canonical cross map computation.

The unfolding procedure is summarized in Fig. 3 (AVG stands for average, STD for standard deviation and A represents the difference between 90° and the angle between ∇u and ∇v). We note the effects of the gradient norm adjustment in the region of the nose in Fig. 3, where the checkerboard rectangles have a more uniform width after adjustment. k_s provides an indication about the ratio between the distance from the pole to the boundary and the representative length of v level curves.

The quasiconformal encoding (i.e. the unfolded conformal factor) is represented by a color gradient from green to blue in Figs. 3 and 4. It measures the local uniform scaling stretch that one has to apply to unfold the geometry to the plane. Intuitively, it depicts the appearance of protrusions on the shape, such as the nose and the ears.

At the end of the encoding computation, the geometry of each surface is represented by the 2D picture of its conformal factor, and its mesh is stored in a binary space partition of the plane.

5. Querying the corpus

At query time, the 2D quasiconformal encoding of the input is computed. We describe in this section how to register it with those of the corpus to generate low distortion cross maps in 3D.

5.1. *Coherency of the encodings*

The quasiconformal surface encoding computation only relies on *intrinsic* surface measurements: it will not vary under global rotations or translations in 3D. Since we normalized the conformal factor, it does not vary under global uniform scaling. Moreover, the conformal factor is a smooth entity (i.e. it varies smoothly under surface transformation). As shown in Fig. 5 (top), the quasiconformal surface encoding is relatively stable under random noise on the surface.

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two surfaces in $\mathbb{R}^3$ such that there exists an isometry $\psi^* : \mathcal{S}_1 \rightarrow \mathcal{S}_2$ and that the boundary conditions of their quasiconformal maps $\phi_1 : \mathcal{S}_1 \rightarrow \mathcal{D}_1$ and $\phi_2 : \mathcal{S}_2 \rightarrow \mathcal{D}_2$ map exactly through ψ^*. Since isometries are angle-preserving *(i)* and area-preserving *(ii)*, by construction, $k_{\mathcal{S}_1}$ and $k_{\mathcal{S}_2}$ (which capture the angular distortion *(i)*) and the conformal factor fields (which capture the area distortion *(ii)*) will be identical. Then, the following equations hold ($\widehat{\psi}^* : \mathcal{D}_1 \rightarrow \mathcal{D}_2$ is the identity):

$$(k_{\mathcal{S}_1} - k_{\mathcal{S}_2})^2 = 0 \tag{4}$$

$$\int_{(u,v)\in\mathcal{D}_1} \left(\lambda_1(u,v) - \lambda_2(\widehat{\psi}^*(u,v))\right)^2 du dv = 0 \tag{5}$$

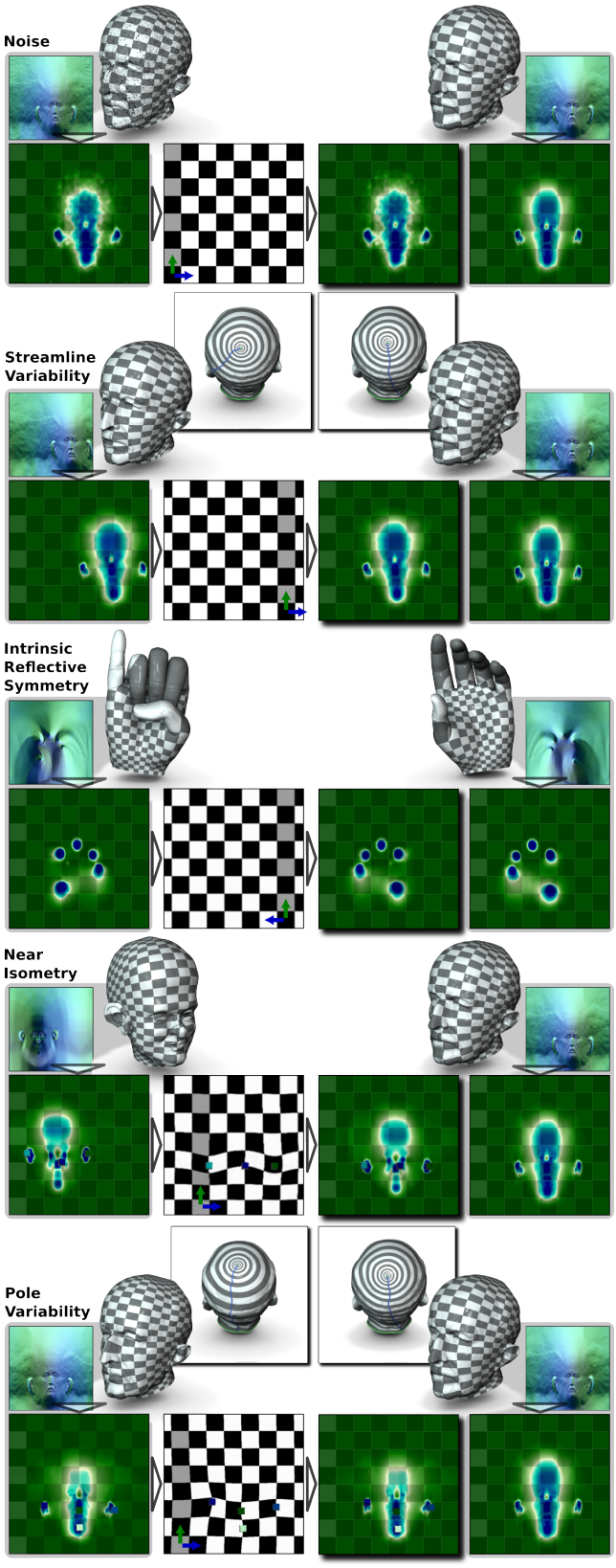


Fig. 5. Variability of the quasiconformal encoding (from top to bottom): random noise can be registered with the identity, streamline variability (pure isometry) translates into periodic translation, intrinsic reflective symmetry translates into axial symmetry, near isometries and pole variability translate into as-rigid-as possible planar maps. In the center of each row, the optimal canonical cross map $\widehat{\psi}^*$ found automatically by the solver.

Since the quasiconformal encoding is smooth, if $\mathcal{S}_1$ and $\mathcal{S}_2$ smoothly vary from being isometric, then the right hand side of these equations will smoothly increase and the canonical cross map $\widehat{\psi}^*$ which minimizes the dissimilarity in the encodings will smoothly vary from the identity. Then, the problem of geometry aware cross parameterization of $\mathcal{S}_1$ and $\mathcal{S}_2$ can be formulated as an optimization problem, where one seeks the canonical cross map $\widehat{\psi}^*$ which minimizes the following energy:

$$E(\widehat{\psi}) = \int_{(u,v)\in\mathcal{D}_1} \left(\lambda_1(u,v) - \lambda_2(\widehat{\psi}(u,v))\right)^2 dudv \tag{6}$$

Then the cross map in 3D can easily be reconstructed (the inverse of a quasiconformal map is quasiconformal):

$$\psi^* = \phi_2^{-1} \circ \widehat{\psi}^* \circ \phi_1 \tag{7}$$

In Fig. 5, ϕ_1 and ϕ_2 are shown with vertical arrows and $\widehat{\psi}^*$ with horizontal arrows. In the next paragraphs, we describe how to construct such canonical cross maps $\widehat{\psi}$ and describe in Sec. 5.2 how to explore their space in search of an energy minimizer.

Isometries During the boundary condition settings for the quasiconformal unfolding, an arbitrary streamline is used to cut the surface. Since it is hard in practice to find a stable heuristic for the choice of the direction of descent of the streamline, we need to consider the variability of the encoding relatively to the streamline. In particular, given two surfaces $\mathcal{S}_1$ and $\mathcal{S}_2$ mapping through an isometry ψ^*, if the poles map through ψ^*, all the level sets of v_1 on $\mathcal{S}_1$ will map by construction through ψ^* to the same level sets of v_2 on $\mathcal{S}_2$. Since $\nabla u_1.\nabla v_1 = 0$ and $\nabla u_2.\nabla v_2 = 0$, the level set $u_1 = 0$ on $\mathcal{S}_1$ necessarily corresponds to *a* level set of u_2 on $\mathcal{S}_2$ (with value $u_2 = t_u^*$). Then a periodic u-translation by t_u^* in the canonical domain will make the streamlines (and thus the encodings) correspond, as shown in Fig. 5 (second row). Then to minimize equation 6, one needs to find the optimal t_u^* (where T is a periodic translation whose parameters respectively denote the u and v components of the translation): $\widehat{\psi}^* = T(t_u^*, 0)$.

Intrinsic reflective symmetries We call two surfaces $\mathcal{S}_1$ and $\mathcal{S}_2$ intrinsically reflection-symmetric if they are isometric modulo a 3D reflection (Fig. 5, third row). If the poles map through such a transformation, then the intrinsic reflective symmetry can be recovered in the quasiconformal encoding. For topological discs, intrinsic reflective symmetries can be recovered by combining an axial symmetry (horizontal flip) to the previous expression of $\widehat{\psi}$ (as shown in Fig. 5). Then to minimize equation 6, one needs to choose the optimal value $s_u^* \in \{-1, 1\}$ (where M multiplies by its parameters the (u, v) of each point): $\widehat{\psi}^* = T(t_u^*, 0) \circ M(s_u^*, 1)$.

Near isometries The problem becomes more interesting when $\mathcal{S}_1$ and $\mathcal{S}_2$ are not strictly isometric (but with close $k_{\mathcal{S}_1}$ and $k_{\mathcal{S}_2}$ values, Fig. 5 fourth row). To register the quasiconformal encodings, we consider as-rigid-as-possible (ARAP) transformations [5] as candidate maps $\widehat{\psi}$. Since they are compositions of rotations and translations, they are as-isometric-as-possible in the canonical domain and consequently introduce as little distortion as possible. Thus, the main distortion introduced in ψ in 3D will solely come from the unfoldings ϕ_1 and ϕ_2. If $k_{\mathcal{S}_1}$ and $k_{\mathcal{S}_2}$ are equal, then the angular distortion introduced by ϕ_1 (captured by $k_{\mathcal{S}_1}$) will be symmetrically compensated by that of ϕ_2^{-1} in 3D (cf. Eq. 7). Then, by using ARAP transformations in the canonical domain, the only distortion introduced in the cross map in 3D will be area distortion (λ).

Thus, by using ARAP transformations in the construction of $\widehat{\psi}$, the minimizer of equation 6 (which precisely minimizes the difference of conformal factors over the entire domain) will be as-isometric-as-possible in 3D.

A closed-form formula for planar as-rigid-as-possible transformations is known and is described in [5]. This formulation has many nice properties: it enforces hard constraints, it is extremely fast in practice (many terms can be pre-computed) and its closed-form formula allows the design of iterative solvers based on its analytic gradient (which we introduce in appendix). We refer the reader to the original publication [5] and to the appendix for further details as well as for the exact expressions.

For some given constraints $p_i \in \mathcal{D}_1$ and their mappings $q_i \in \mathcal{D}_2$, this closed-form expression enables to compute directly the mapping of each point of $\mathcal{D}_1$ onto $\mathcal{D}_2$. Then, to minimize equation 6, one needs to choose the optimal mappings q_i of the constraints p_i. We describe in the next subsection how to automatically extract a relevant set of constraints p_i and how to optimize their mappings q_i.

Other topological primitives The reasoning developed above is valid for other topological primitives than discs, with only few adjustments. For annuli (topological cylinders), the two boundary components are respectively set to $v = 0$ and $v = 1$ and u is computed exactly as described in Sec. 4.2. However, admitting an additional boundary component extends the possibilities of intrinsic reflective symmetries. One needs to consider not only *horizontal* symmetries but also *vertical* symmetries (in the canonical domain) since the two boundary components set at $v = 0$ and $v = 1$ can be switched arbitrarily from one cylinder to another during the parameterization. Then one needs to consider the four multiplication possibilities, four *flips*, $M(1,1)$, $M(-1,1)$, $M(1,-1)$ and $M(-1,-1)$ respectively corresponding to no symmetry, horizontal symmetry, vertical symmetry, horizontal *and* vertical symmetry in the canonical

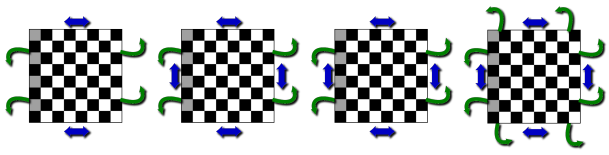


Fig. 6. Canonical domain diagrams for (from left to right) discs, annuli, spheres and tori. Green arrows indicate boundary periodicity. Blue arrows indicate potential reflective symmetries.

domain, as shown in Fig. 6. Spheres with two antipodal poles are processed the exact same way. For tori (not supported by our prototype implementation), a *handle loop* [30] needs to be extracted (with v set to 0 and 1 on both sides) and periodicity in the vertical direction needs to be considered.

Subjective coherency and final ranking score The quasi-conformal encodings are smooth and so are as-rigid-as-possible transformations. Thus the energy (Eq. 6) should vary smoothly when $\mathcal{S}_1$ and $\mathcal{S}_2$ vary smoothly from an isometry. In other words, after minimization, the resulting energy provides a good evaluation on how far $\mathcal{S}_1$ and $\mathcal{S}_2$ are from being isometric. Then we use the energy after optimization as a coherency score, in order to rank the corpus examples when presented to the user. We extend this score in two ways. First, we penalize the difference in k values (which need to be close to guarantee little distortion). Second, we prioritize examples for which a good subjective evaluation has been given a priori. The example corpus is then finally ranked by *increasing* values of:

$$E'(\widehat{\psi}^*) = \frac{r_{max}}{r(S_1) + r_{max}} \left(1 + (k_{\mathcal{S}_1} - k_{\mathcal{S}_2})^2\right) E(\widehat{\psi}^*) \qquad (8)$$

where r_{max} is the maximum relevance score (typically 5) and $r(S_1)$ is the relevance score of the considered corpus example (cross maps are computed from the example $\mathcal{S}_1$ to the input $\mathcal{S}_2$).

5.2. *Multi-resolution cross map solver*

To explore the space of candidate planar cross maps, we use a gradient descent formulation. Given an initial cross map, the algorithm will progressively transform it to minimize Eq. 6. Energy local minima can impair the finding of decent cross maps. To overcome this issue, we derive the following multi-resolution strategy. Each quasiconformal encoding (originally computed at a resolution of 512x512 pixels) is stored in a hierarchy of pictures with decreasing resolutions (decreased at each level by 2, typically until 32x32 pixels). The advantage of this strategy is twofold. First, reducing the resolution will act as a global filter on the images. Thus, only the most important features will guide the optimization at the beginning at the coarse level, while more fine-scale features will appear as the optimization refines the result in the finer levels of the hierarchy. Moreover, the multi-resolution will help the solver to avoid local minima by bringing it closer to the global solution at the coarser levels of the hierarchy. Second, proceeding in a coarse to fine fashion enables to focus the computing resources on the corpus examples which look more promising. The process is illustrated in Fig. 7, where the input appears in gray and the corpus examples in color. At the first level of the hierarchy,

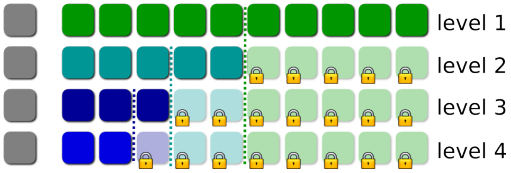


Fig. 7. Multi-resolution processing of the example corpus.

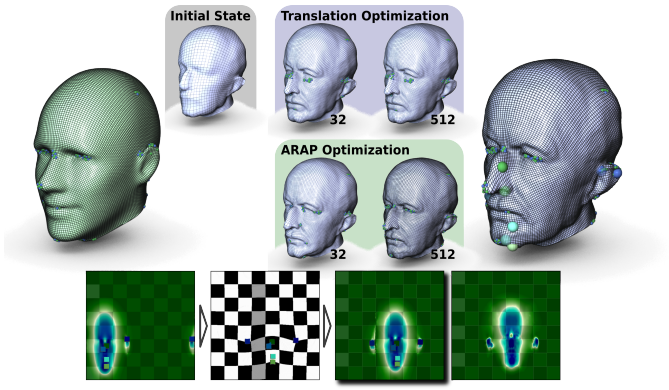


Fig. 8. Evolution of the optimization process. Small blue and green spheres are extraordinary vertices. The canonical cross map found automatically by the solver is shown at the bottom. Colored squares are the ARAP constraints. In 3D (right), transparent spheres show the locations of the corresponding constraints after translation optimization; non-transparent spheres show their location after ARAP optimization. Most of the periodic translation component is compensated at a resolution of 32x32 and further refined until 512x512, while the ARAP optimization aligns the features.

the entire corpus is cross mapped at low resolution and ranked by increasing energy ($E'(\widehat{\psi}^*)$). Each optimization is resumed at higher and higher resolutions at the next levels, where only the top half of the corpus of the previous level is used and re-ranked while the remaining examples are locked in the list. The process stops when the highest resolution is completed or on user demand. In practice, we still enforce the computation at full resolution for the top-n entries (typically $n = 4$). The advantage of this strategy is that when two surfaces are far from being isometric, their quasiconformal encoding will be significantly different, resulting in high cross map energy. Then our solver will spend less computation resources on those.

Notice that this strategy aims at quickly presenting to the user the most relevant examples at the top of the ranked list. Once this process is finished, the incomplete computation of poorly ranked examples is resumed if they are selected by the user for cross mapping. Now, we describe the optimization process on a per-entry basis, where $\mathcal{S}_1$ represents an example from the corpus and $\mathcal{S}_2$ the input.

Intrinsic reflective symmetries Since switching from an axial symmetry to another in the canonical domain is not a smooth transition (complete horizontal or vertical flip, or both), the solver starts by duplicating the quasiconformal encoding hierarchy of $\mathcal{S}_1$ and flipping it, in order to optimize individually each potential symmetry. For instance, the quasiconformal encoding hierarchy of a disc is duplicated once with an extra horizontal flip while that of an annulus is duplicated three times (one with horizontal flip, one with vertical flip, one with both). Then, these duplicated entries undergo the rest of the optimization algorithm individually and the symmetry which minimizes $E(\widehat{\psi}^*)$ is considered to be the optimal one for $\mathcal{S}_1$.

Isometries Periodic translations are needed to compensate for streamline variability (Sec. 5.1). At the coarsest level of the hierarchy, the solver evaluates greedily each possible periodic

translation (translating by one pixel at a time). The translation which minimizes $E(\widehat{\psi}^*)$ is considered to be the optimal translation for the current resolution. At the next hierarchy level, the optimal translation of the previous level is re-evaluated and refined by also evaluating the immediate neighbor translations: one pixel to the left and one to the right (corresponding to half-pixel translations in the previous resolution). This process is then iterated from one level of the hierarchy to the next until completion, resulting in a progressive adjustment of the translation component of the optimal canonical cross map (Fig. 8).

Near isometries We now consider that at reach resolution the periodic translation component of $\widehat{\psi}^*$ is resolved. As-isometric-as-possible 3D cross maps can be computed through ARAP canonical cross maps (cf. Sec. 5.1). Those can be computed with the closed-form formula given in appendix, where p_i is the location of the i-th control feature point in $\mathcal{D}_1$ and $q_i = \widehat{\psi}(p_i)$. We now describe how to compute both p_i and q_i automatically.

Since the solver tries to minimize the difference of conformal factors (cf. Eq. 6), local maxima of λ_1 are valid candidates to drive the construction of $\widehat{\psi}$. Starting at the coarsest level, the solver extracts a handful of maxima of λ_1. The pixels of the encoding are sorted by decreasing λ_1 and maxima are added to the set of p_i points if they are sufficiently distant in $\mathcal{D}_1$ from previously selected feature points (at least 10 pixels) and if their λ_1 exceeds a reasonable threshold (typically 4). When this process is completed, the extracted feature points are projected at the next level of the hierarchy and the extraction process is iterated to extend if possible the set of feature points in the new resolution. As suggested in [5], to speed up the optimization, the set of p_i and their related terms in the closed-form formula are computed for each entry of the corpus *once for all offline*, when the example is added to the corpus. Then, *online*, to optimize $\widehat{\psi}$, one needs to move each point $q_i = \widehat{\psi}(p_i)$ on $\mathcal{D}_2$ such that Eq. 6 gets minimized, yielding a low distortion cross map in 3D. Since $\widehat{\psi}$ has a closed-form formula, we can directly predict the local move of each q_i that improves the energy by deriving Eq. 6:

$$\nabla^i E(\widehat{\psi}) = 2 \sum_{\forall x \in \mathcal{D}_1} \left[\left(\lambda_2(\widehat{\psi}(x)) - \lambda_1(x)\right) \left(\nabla\lambda_2(\widehat{\psi}(x))\ \nabla^i\widehat{\psi}(x)\right) \right]$$

where $\nabla\lambda_2(\widehat{\psi}(x))$ is a vector in row notation (i.e. the gradient of the conformal factor picture) and $\nabla^i\widehat{\psi}(x)$ a 2x2 matrix:

$$\nabla^i\widehat{\psi}(x) = \begin{pmatrix} \nabla^i_u\widehat{\psi}_u(x) & \nabla^i_v\widehat{\psi}_u(x) \\ \nabla^i_u\widehat{\psi}_v(x) & \nabla^i_v\widehat{\psi}_v(x) \end{pmatrix} \quad (9)$$

∇^i_u denotes the u component of gradient vectors and $\widehat{\psi}_u$ denotes the u component of $\widehat{\psi}$. The derivation of the as-rigid-as-possible canonical cross map $\widehat{\psi}$ is a bit involved and detailed in appendix.

For each control point q_i, we compute $\nabla^i E(\widehat{\psi})$, revert it and scale it. The obtained vector points to the next pixel which q_i should visit to minimize Eq. 6. Once the gradient is computed for all the control points q_i, those are actually displaced to their optimal neighbor, $\widehat{\psi}$ is reconstructed and the energy is re-evaluated. This process is repeated until the energy stops decreasing. The optimization is iterated at the next resolutions, after projection of the optimized q_i in the new resolution, until completion at the highest resolution. We found in practice that only very few automatic constraints are needed to achieve visually appealing 3D cross maps. Notice that the formalism presented in [5] provides no mechanism to avoid fold-overs. After each control point displacement, we verify that $\widehat{\psi}$ is monotonic for each of its component (i.e. one-to-one and onto). If this test fails, the optimized locations of the control point(s) q_i located on the edge of the fold-over(s) are rolled back until the fold-over is resolved and q_i is locked from any further displacement.

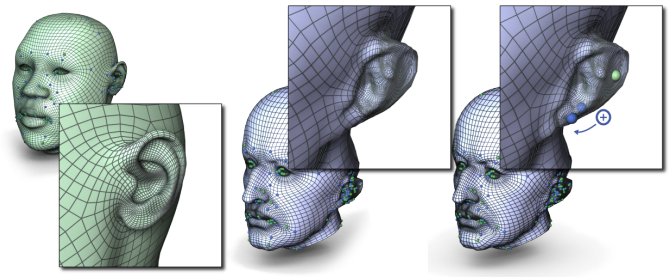


Fig. 9. User editing of the inspired quadrangulation. Selected example (left). Suggested mapping (middle). Subtle user enhancements of the suggested mapping (right). The addition of the new constraint (in blue on the right), the cross map update and the re-projection of the mesh took respectively 0.03s., 0.023s. and 0.18s. (with 10,384 quads in the example).

Concurrent execution An appealing aspect of this optimization algorithm is that much of the work can be done concurrently. The optimization of each example of the corpus is run in an independent thread. Each symmetry is also run in an independent thread. Finally, each component $\nabla^i E(\widehat{\psi})$ of the gradient is run in an independent thread. In practice, we implemented a thread pool pattern, in order to control the maximum number of concurrently running threads.

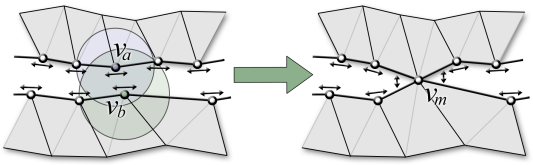
5.3. *Interactive editing of the cross maps*

When the corpus processing is completed, the examples are presented to the user who can select them to trigger their mapping on the input. Since some features of the example quad-meshes can recover more *semantic* than purely geometrical aspects, we allow the user to interact with the cross map. This is shown in Fig. 9 where there is no well-defined earlobe on the Max Planck model, while there is one on the example. The interaction with the cross map happens at interactive rates. The user can modify the constraints automatically optimized or add new constraints to the system. In both cases, since we use a closed-form formula for the canonical cross map computation, this is achieved very efficiently (less than half a second). We used this interaction feature for instance in Fig. 14 to improve the alignment of the faces (in particular the eyes of the cat and the camel, corresponding in both cases to flat regions).

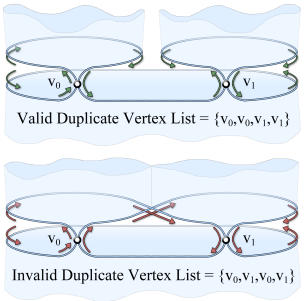
6. Mesh Structure and Composition

Internally, the mesh connectivity that populates the corpus of example mesh components is stored as a triangle mesh. The non-triangular elements of the mesh are tesselated and the inserted edges are flagged as *phantom* edges. While this paper is aimed at recycling quad-meshes, the *phantom* edge implementation supports the construction of arbitrary polygonal meshes. More importantly, the ability to encode arbitrary polygonal elements allows for a flexible segmentation scheme, such that boundary curves are not restricted to mesh edges.

The final phase of our mesh generation paradigm involves a stitching scheme aimed at composing the multiple mesh components mapped to the input geometry. The final composition handles the resolution of the phantom edges and any triangles resulting along the boundary curves. The following stitching seamlessly composes the multiple mesh components, resolving mis-matched boundary elements and phantom edges to generate quad-only results. The method begins by identifying the multiple double-linked lists describing the boundary components of the different mesh components. The stitching process manipulates these structures, while (1) merging nearby vertices, (2) filling holes, (3) resolving triangles, (4) removing quadrilateral doublets and (5) smoothing the composed elements.



Vertex merging occurs in a greedy fashion. The nearest pairs of boundary curve vertices are merged iteratively, updating the double-linked list that maintains the boundary curve structures. Two vertices v_a and v_b are merged if $v_b \notin \mathcal{N}_a$ (the 1-ring neighborhood of v_a) and the distance between them is less than or equal to the boundary edge lengths to which both v_a and v_b are neighbors, as illustrated above. The double-linked list that describes the boundary curve orientation plays a critical role during the vertex mergers.



First, the linked lists temporarily allow non-manifold vertices in the mesh representation without loss of the neighborhood ordering and boundary components. Second, the double-linked lists facilitate the validation of vertex mergers by tracking the duplicate vertices encountered while marching the boundary curve. As illustrated to the left, duplicate vertices must be encountered in pairs, i.e., $\{v_0, v_0, v_1, v_1, ...\}$. In contrast, when the duplicate vertices alternate, i.e., $\{v_0, v_1, v_0, v_1, ...\}$, the figure-8 boundary curve will result in a non-manifold edge construction during the hole filling.

Following the completion of the greedy merging process, the automated composition resolves the mesh holes that arise due to t-junctions, as illustrated in Fig 10. This is similar to challenges addressed by geometry clipmaps [31] and multi-chart geometry images [32]. The hole filling begins by identifying and tesselating the multiple connected components (loops) within the linked list of boundary edges. In the simple case a loop consists of two edges, in which case the edges are in fact neighboring half-edges, updating the neighborhood information in the mesh (2-edge termination case). A boundary loop with three edges defines a triangle inserted into the mesh unflagging any phantom edges it shares (3-edge termination case). Lastly, when more edges describe the loop, the algorithm inserts a quadrilateral, tesselated as two triangles with a phantom edge between them, along the longest edge in the boundary loop. The loop is updated with two fewer edges, and the process is repeated until one of the terminating cases is reached. If during the vertex merging process all boundary vertices are paired, then all boundary loops will contain only two edges and no additional face elements are required to fill the seams.

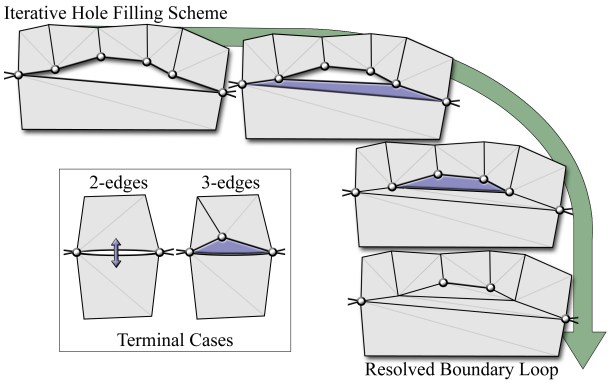


Fig. 10. The quad-dominant hole filling scheme iteratively inserts quadrilaterals (tesselated as two triangles) adjacent to the boundary loop's longest edge until a terminal case is reached (a 2 or 3 edge loop).

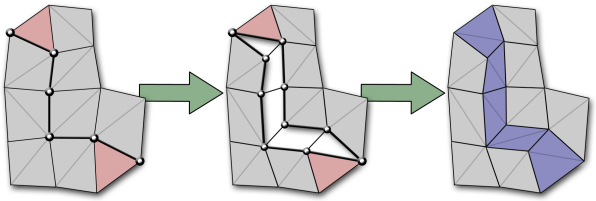


Fig. 11. Triangles are resolved (a) by cutting the mesh along the list of edges forming the shortest path between them (b), then inserting quads by connecting the duplicated vertices (c).

Next, our algorithm resolves possible triangles at the boundary of the patches by performing mesh surgery along the sequence of non-phantom mesh edges that forms the shortest path between them. Connecting edges between the duplicated vertices constructs quad elements, Fig. 11, while preserving much of the original mesh alignment and dual structure. Note that following the hole filling method, the mesh describes a connected closed manifold, and will always contain an even number of triangle elements (then triangles can always be paired). If a boundary component is desired in the final mesh (closed by the hole filling procedure), then the inserted elements can be removed during a post-process to re-open the mesh.

The final two stages of the stitching procedure involve mesh

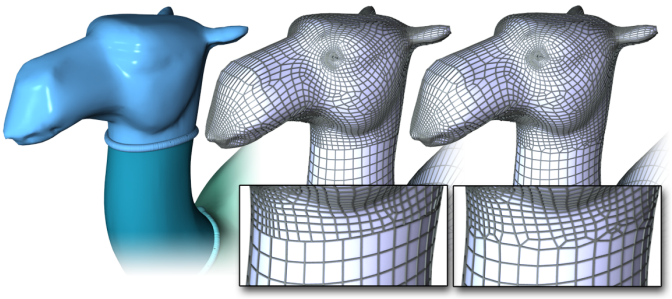


Fig. 12. Boundary stitching: surface segmentation (left), quad-mesh patches before (middle) and after (right) the overall stitching procedure.

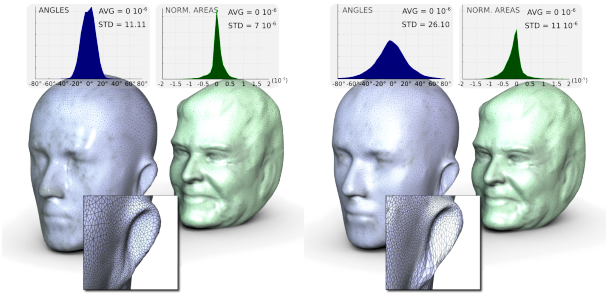


Fig. 13. Comparison between the inspired quadrangulation algorithm (left) and the template fitting algorithm (right). The histograms represent the angle and normalized area distortions induced by the cross map in 3D.

cleanup and vertex smoothing. The cleanup phase replaces quadrilateral doublets, the neighbor elements of a valence-2 vertex, with a single quadrilateral element, removing the valence-2 vertex. To this point the connectivity describes a quad-only mesh. We finalize the stitching procedure with a smoothing operation, iteratively performing Laplacian smoothing and backwards projecting to the original model until desirable element qualities are obtained.

7. Experimental results

We implemented our approach in C++ and ran our experiments on an i7 CPU (with 4 cores at 2.93 GHz and 8 GB of RAM). In this section, we first evaluate the performance of the cross mapping algorithm and next evaluate the efficiency of the quad-meshing style mimicking process.

Cross parameterization performance In Fig. 13 we compare our algorithm to the template fitting algorithm [23], on a data-set kindly provided by the authors. Head data-sets are challenging test cases for cross map algorithms since inaccuracies are rapidly identified by human observers. Thus, *interpolation-only* cross map algorithms need a large set of user defined correspondences (24 in Fig. 13, right) while our approach extracted the cross map without any user defined correspondence and still with less distortion, which is critical to faithfully reproduce the subjective layout of the examples.

Computational aspects An important aspect of our approach is response time. First, only the unfolding step is conditioned in response time by the complexity of the input mesh. As for the corpus processing (with 45 example segments in our experiments), since we use a multi-resolution strategy, most of the computation happens for the top-ranked entries only. In detail, for each entry, the time execution of the cross map optimization can be very variable. It first depends on the employed resolution of the quasiconformal encodings. Also, it depends on the number of automatically extracted ARAP constraints. Finally, the number of iterations required by the solver directly depends on the similarity between the input and the examples. In practice, we stop the optimization after a maximum number of iterations, typically 10. Other aspects contribute to the time performance: first, the optimization is based on an analytic expression of the gradient of the candidate maps (which is faster than naive all-neighbor exploration); then the algorithm permits parallelization. Query timings are provided in table 1, where some satisfying cross maps were already obtained at a resolution of 128x128 or 256x256. We applied the default value of 512x512 for all the other experiments.

Inspired quadrangulations Figure 14 shows some representative quadrangulations obtained with our approach, where each surface segment has been processed individually by the automatic cross mapping algorithm (showing 36 cross maps total). As shown in Fig. 12, quad-mesh parts are stitched seamlessly on the input. Our approach is oblivious to intrinsic reflective symmetries, as shown at the third row. A noticeable advantage of the pre-segmentation of the input is that it enables quad-meshing mimicking even if the example varies importantly from an isometry or does not have the same topology as the input (the camel is genus-1 while the dog is genus-0). In particular, at the fourth row, an extra piece of geometry (regular quadrangulation of a cylinder) has been mapped onto the neck of the camel (dark blue segment), in between the head and the torso (where dog parts were mapped). The same process was used to remesh the knees of the camel where a topological handle occurs. While our approach mimics the edge alignment and the configuration of extraordinary vertices of the example, it also enforces the preservation of its low-level geometrical statistics thanks to the low distortion cross map algorithm. As shown in the histograms, the overall shape of these distributions are preserved, with variations if the input is far from isometric to the example (see cat/dog against camel/dog). In comparison to previous quadrangulation techniques, our approach enables a novice user to rapidly generate quadrangulations with professional looking, by mimicking the quadrangulations of models produced by artists (Fig. 1). In contrast, user-driven quadrangulation techniques would not only require many interactions, but also a deep understanding by the user of the meshing concepts

Input	#Tri.	G.U. (s.)	C. P. (s.)	Res.
Fig. 1	12k	1.06	3.80	128x128
Fig. 13	92k	6.16	1.09	256x256
Fig. 14, Cat*	7k	0.59	2.37	512x512
Fig. 14, Human*	13k	1.17	9.40	512x512
Fig. 14, Hand*	6k	0.64	1.88	512x512
Fig. 14, Camel*	19k	1.65	6.33	512x512
Fig. 15	12k	1.06	7.54	512x512

Table 1 Query times for the outputs presented in this paper. $\#Tri.$, $G.U.$, $C.P.$ and $Res.$ respectively correspond to the number of triangles in the input, the time of the geometry unfolding step, the time of the corpus processing step and the employed maximal encoding resolution. For the composition results (*), these numbers are averages over the set of surface segments.

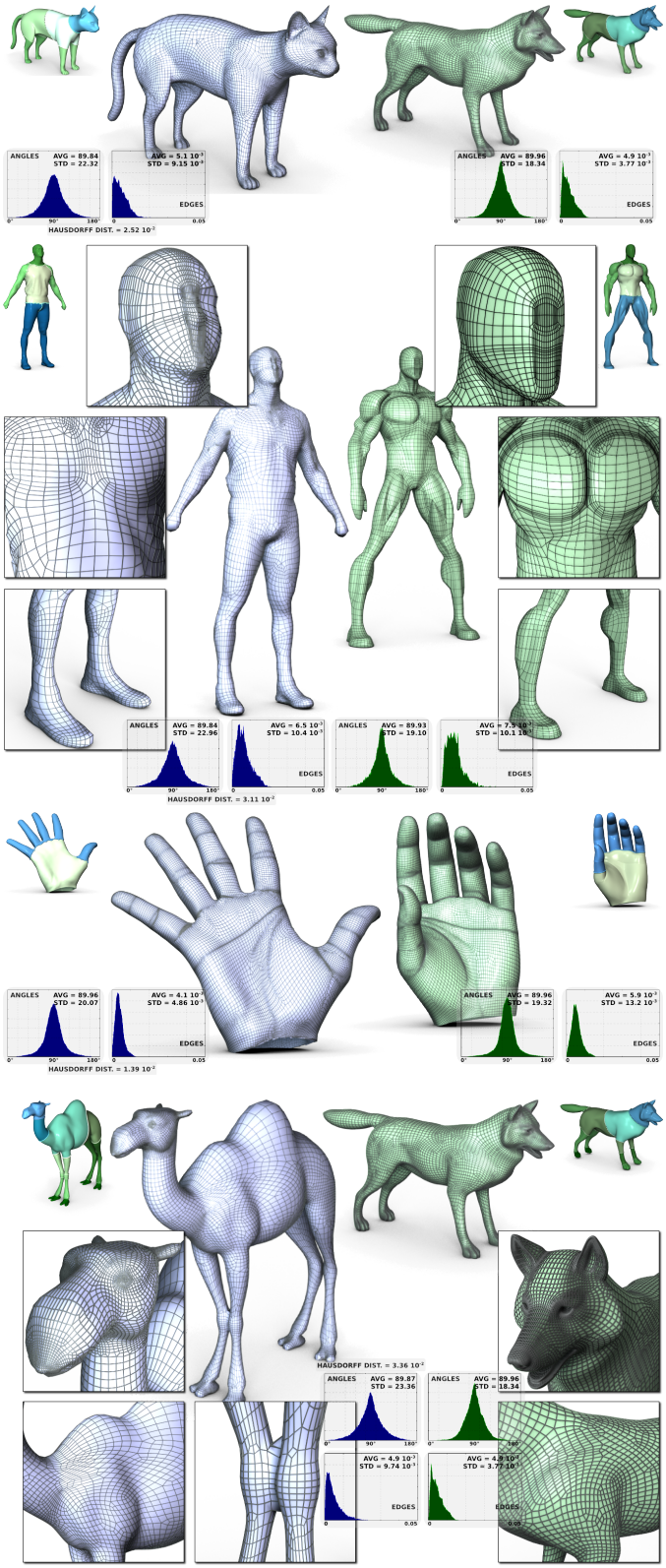


Fig. 14. Inspired quadrangulations generated with our prototype (examples appear in green on the right, segmentations in the top corners of each row). The histograms respectively show the distributions of all quad angles and edge lengths (distances are normalized by the bounding box diagonal). Our approach enables to quad-mesh scanned geometries with the style of cartoon characters (second row), to mimic meshing style despite intrinsic reflective symmetry (third row) or even despite important variation from isometry and topological changes (fourth row).

involved in the production of such outputs. Our algorithm can also be used as a complementary tool to previous quadrangulation algorithms. This is illustrated in Fig. 15, where the example has been generated with the *Geometry Aware Direction Field* (GADF) design algorithm [10].

Limitations A limitation of our cross mapping algorithm (although our multi-resolution strategy overcomes this difficulty) is that the optimization is not global. However, global optimization is recognized to be much less computable than gradient descent approaches. We believe this limitation is the price to pay for interactive usage capabilities. Also, since we use a pixel-based representation for the cross map optimization, it is possible that some 3D features on the surface go in 2D below pixel precision. For instance, if the shape exhibits many important protrusions, those will undergo important distortion once unfolded and the exact registration of their extremities will require high resolution quasiconformal encodings. While the resolution of the encodings could be simply increased, we believe the ultimate solution is to introduce extra singularities in the unfolding (hence improving the distortion), which is done in our overall framework through segmentation (one pole per disc segment). The fact that we use in the framework of composition pre-segmentations that need to be compatible is also a limitation. However, this user intervention is limited (the segmentations are coarse and intuitive). Also automatic algorithms could be used [33]. Moreover, we believe that it is in fact a strength of our approach. It enables users to refine their example search on a per region-of-interest basis. Also, as demonstrated in Fig. 14 (fourth row), it enables to exploit examples having different topology. However, if adjacent quad-patches exhibit a high difference in resolution, new extraordinary vertices and small edges are necessarily inserted by the stitching procedure in order to generate a quad-only output (Fig. 12). Finally, if the user selects an example mesh that is far from isometric to the input, distortion inevitably occurs due to the local difference in conformal factors. Then, once mapped on the input, the example mesh may be locally scaled down or up, leading respectively to local quad shrinking or approximation errors. For instance, if a feature is present on the input mesh but not on the example, the output mesh may not be fine enough to capture finely the extra feature.

8. Conclusion and future work

In this paper, we presented *Inspired Quadrangulation*, a new approach to produce quality quadrangulations from input triangular surfaces, by reproducing the meshing style of reference examples provided by experts. At the core of our technique, we provided the analytic gradient of 2D as-rigid-as-possible transformations [5], that we used in a fast multi-resolution solver which registers unfolded geometries, yielding low distortion cross maps in 3D. Experiments showed that our algorithm could be trained with the output of state-of-the-art algorithms or by meshes designed by professional artists. Our approach enables to mimic accurately the meshing style of examples despite intrinsic reflective symmetry, high variation from isometry or

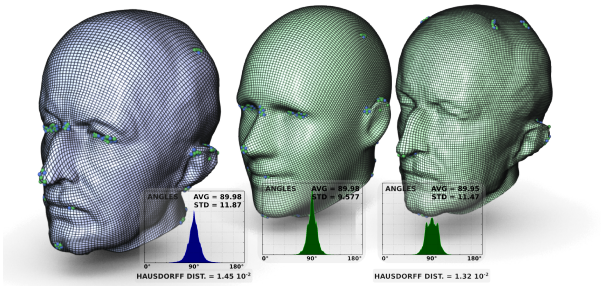


Fig. 15. Comparison between the inspired quadrangulation algorithm (cross map computed automatically, left) inspired by *GADF* (middle) and *GADF* itself (right). By preserving the overall shape of the angle distribution of an example with better element quality (middle), the inspired quadrangulation algorithm generated a mesh with a sharp and centered angle distribution (left).

even topological changes. In future work, we would like to continue our investigation of data-driven geometry processing. We believe that many challenging problems can be addressed by exploiting examples recognized as correct solutions, allowing users to increase their productivity by re-exploiting their own creations or by leveraging valid examples provided by others. While our algorithm mimics meshing styles by cross mapping examples, it would be interesting to develop in the future further algorithms aiming at actually *learning* the process of quadrangulation design (capable of taking design decisions). The solution of such a problem would probably require a user-centric formalization of quadrangulation design as well as the consideration of training data sets capturing the entire history of decisions made by human experts during their meshing sessions.

Acknowledgments We thank Stanley Durrleman and Tobias Ritschel for inspiring discussions. This work was supported in part by NSF, DoE, IBM Faculty Awards, US ARO, ExxonMobil and Fapesp-Brazil.

References

[1] I. Boier-Martin, H. Rushmeier, J. Jin, Parameterization of triangle meshes over quadrilateral domains, Symp. on Geom. Proc. (2004) 193–203.

[2] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, J. Hart, Spectral surface quadrangulation, ACM Trans. Graph. (SIGGRAPH) 25 (2006) 1057–1066.

[3] D. Kovacs, A. Myles, D. Zorin, Anisotropic quadrangulation, in: Proc. of ACM SPM, 2010, pp. 137–146.

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

[5] S. Schaefer, T. McPhail, J. Warren, Image deformation using moving least squares, ACM Trans. Graph. (SIGGRAPH) 25 (2006) 533–540.

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

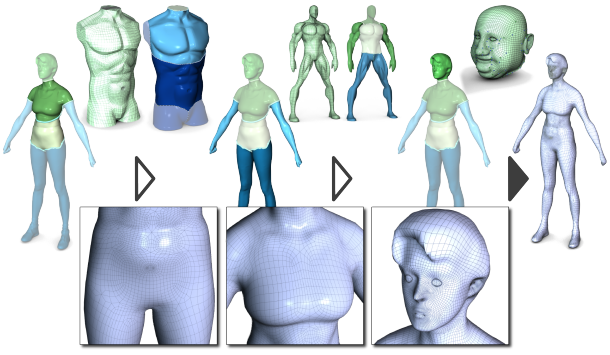


Fig. 16. Inspired quadrangulation by composition of several models.

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

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

[9] D. Bommes, H. Zimmer, L. Kobbelt, Mixed-integer quadrangulation, ACM Trans. Graph. (SIGGRAPH) 28 (2009) 1–10.

[10] N. Ray, B. Vallet, L. Alonso, B. Lévy, Geometry aware direction field design, ACM Trans. on Graph. 29.

[11] M. Zhang, J. Huang, X. Liu, H. Bao, A Wave-based Anistropic Quadrangulation Method, ACM Trans. Graph. (SIGGRAPH) 29.

[12] M. Tarini, K. Hormann, P. Cignoni, C. Montani, Polycube-maps, ACM Trans. on Graph. (SIGGRAPH) (2004) 853–860.

[13] P. Alliez, G. Ucelli, C. Gotsman, M. Attene, Recent advances in remeshing of surfaces, in: Shape Analysis and Structuring, Mathematics and Visualization, Springer Berlin Heidelberg, 2008.

[14] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, Modeling by Example, ACM Trans. Graph. (SIGGRAPH) 23 (2004) 652–663.

[15] S. Chaudhuri, V. Koltun, Data-driven suggestions for creativity support in 3D modeling, ACM Trans. Graph. (SIGGRAPH Asia) 29.

[16] T. Kanai, H. Suzuki, F. Kimura, Three-dimensional geometric metamorphosis based on harmonic maps, The Visual Computer 14 (1998) 166–176.

[17] N. Litke, M. Droske, M. Rumpf, P. Schroder, An image processing approach to surface matching, in: Symp. on Geom. Proc., 2005, pp. 207–216.

[18] M. Alexa, Merging polyhedral shapes with scattered features, in: Shape Modeling International (SMI), 1999, pp. 202–210.

[19] X. Li, Y. Bao, X. Guo, M. Jin, X. Gu, H. Qin, Globally optimal surface mapping for surfaces with arbitrary topology, IEEE Trans. on Vis. and Comp. Graph. 14 (2008) 805–819.

[20] A. W. F. Lee, D. Dobkin, W. Sweldens, P. Schroder, Multiresolution mesh morphing, in: ACM SIGGRAPH, 1999, pp. 343–350.

[21] V. Kraevoy, A. Sheffer, Cross-parameterization and compatible remeshing of 3D models, ACM Trans. Graph. (SIGGRAPH) 23 (2004) 861–869.

[22] J. Schreiner, A. Asirvathan, E. Praun, H. Hoppe, Inter-surfae mapping, ACM Trans. Graph. (SIGGRAPH) 23 (2004) 870–877.

[23] I. Cheng-Yeh, C.-H. Lin, O. Sorkine, T.-Y. Lee, Template-based 3D model fitting using dual domain relaxation, IEEE Trans. on Vis. and Comp. Graph. Accepted.

[24] Y. Lipman, T. Funkhouser, Mobius voting for surface correspondence, ACM Trans. Graph. (SIGGRAPH) 28.

[25] O. K.-C. Au, C.-L. Tai, D. Cohen-Or, Y. Zheng, H. Fu, Electors voting for fast automatic shape correspondence, Comp. Graph. Forum (Eurographics) 29 (2010) 645–654.

[26] J. Tierny, J. Daniels, L. G. Nonato, V. Pascucci, C. Silva, Interactive quadrangulation with Reeb atlases and connectivity textures, Tech. rep., SCI Institute, University of Utah (2010).

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

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

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

[30] T. K. Dey, K. Li, Persistence-based handle and tunnel loops computation revisited for speed up, Computers and Graphics 33 (2009) 351–358.

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

[32] N. Carr, J. Hoberock, K. Crane, J. Hart, Rectangular multi-chart geometry images, in: Symp. on Geometry Proc., 2006, pp. 181–190.

[33] J. Erickson, K. Whittlesey, Greedy optimal homotopy and homology generators, in: Proc. of ACM Symp. on Discrete Algorithms, 2005, pp. 1038–1046.