Computers & Graphics 33 (2009) 369–380

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

A divide-and-conquer approach for automatic polycube map construction

Ying He [a,*], Hongyu Wang [b], Chi-Wing Fu [a], Hong Qin [b]

[a] School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue, Blk N4, Singapore 639798, Singapore
[b] Computer Science Department, Stony Brook University, NY, USA

ARTICLE INFO

Article history:
Received 8 December 2008
Received in revised form
25 February 2009
Accepted 3 March 2009

Keywords:
Shape modeling
Polycube map
Discrete Ricci flow
Uniform flat metric
Computational geometry
Object modeling
Geometric algorithms

ABSTRACT

Polycube map is a global cross-surface parameterization technique, where the polycube shape can roughly approximate the geometry of modeled objects while retaining the same topology. The large variation of shape geometry and its complex topological type in real-world applications make it difficult to effectively construct a high-quality polycube that can serve as a good global parametric domain for a given object. In practice, existing polycube map construction algorithms typically require a large amount of user interaction for either pre-constructing the polycubes with great care or interactively specifying the geometric constraints to arrive at the user-satisfied maps. Hence, it is tedious and labor intensive to construct polycube maps for surfaces of complicated geometry and topology. This paper aims to develop an effective method to construct polycube maps for surfaces with complicated topology and geometry. Using our method, users can simply specify how close the target polycube mimics a given shape in a quantitative way. Our algorithm can both construct a similar polycube of high geometric fidelity and compute a high-quality polycube map in an automatic fashion. In addition, our method is theoretically guaranteed to output a one-to-one map. To demonstrate the efficacy of our method, we apply the automatically-constructed polycube maps in a number of computer graphics applications, such as seamless texture tiling, T-spline construction, and quadrilateral mesh generation.

1. Introduction

Polycube map is a novel cross-surface parameterization technique where the parametric domain is a polycube (a.k.a. cubical complex). Compared with other global parameterization techniques, the quality of a polycube map (in terms of angle and area distortion) can be quantitatively controlled by designing the polycube which resembles the geometry of the input shape and shares the same topology [1]. Because of their highly regular structure (i.e., each face is a square or poly-square) and the nature of the "one-piece" global parametric domain (i.e., no cutting and abutting), polycube maps have shown great promise in texture mapping and synthesis [1,2], shape morphing [3], spline constructions [5,4] and harmonic volumetric mapping [17].

Despite many promising properties and great modeling potentials of polycube maps, polycube maps have not yet been widely applied to real-world applications. The underlying reasons are two-fold: (1) Polycubes are usually constructed manually with great care and specific domain knowledge. Designing polycubes for shapes of complicated geometry and topology remains to be very tedious and labor intensive. (2) Once the polycube is devised, the existing techniques to construct the map between the given 3D shape and polycube require either projection of the vertices from 3D shape to the polycube (e.g., [1] which is an extrinsic method) or computing a global surface parameterization (e.g., [4] which is an intrinsic method). As a result, many technical challenges cannot be easily overcome with all the existing methods. For example, the extrinsic method may not produce a valid one-to-one map if the polycube differs from the modeled shape significantly. The intrinsic method, though theoretically sound to guarantee a bijection, may not be practically useful for a topologically complicated surface since the rounding error will cause serious numerical problems in computing the hyperbolic parameterization and the fundamental domain. In [6,4], the 3D surfaces with negative Euler characteristics are required to reduce the number of faces significantly before computing the hyperbolic parameterization. Therefore, many geometry details are lost in the resulting polycube maps. In order to arrive at a high-fidelity polycube map, particularly for a complicated real-world object, our goal is to develop more efficient and accurate methods for producing polycube maps for shapes of complicated geometry and arbitrary topology with far less user intervention towards a full automation.

In particular, this paper tackles the aforementioned technical challenges and develops a novel method to construct polycube maps of arbitrary topology. Compared with the existing methods which usually require tremendous amount of efforts from users to

* Corresponding author. Tel.: +65 65141008; fax: +65 67926559.
E-mail addresses: yhe@ntu.edu.sg (Y. He), wanghy@cs.sunysb.edu (H. Wang), cwfu@ntu.edu.sg (C.-W. Fu), qin@cs.sunysb.edu (H. Qin).

0097-8493/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2009.03.024

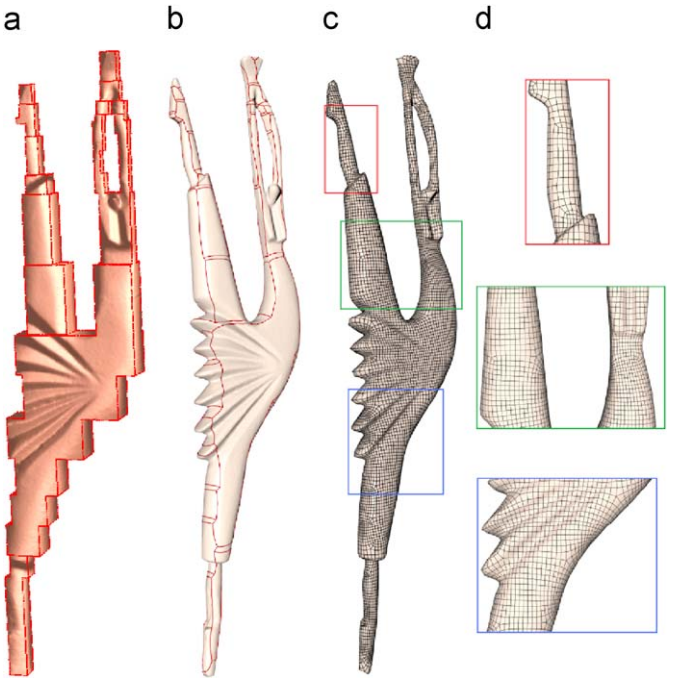

Fig. 1. Polycube map for the genus-1 Dancer model. (a) shows the constructed polycube map. The red curves in (b) illustrate the polycube structure. (c) shows the quadrilateral mesh generated using the polycube map. (d) shows some close-up views of the quadrilateral mesh. Note that both the polycube and polycube map are constructed automatically.

design and build the polycubes, our method, in sharp contrast, is automatic. The user may choose to specify two parameters to control how close the polycube mimics the geometry of the input shape, then our algorithm can construct both the polycube and the one-to-one map between the polycube and input shape automatically. As an example, Fig. 1 shows the automatically-constructed polycube map for the genus-1 Dancer model, as well as the quadrilateral remeshing using polycube maps.

The contributions of this paper include:

(i) We develop an automatic method to construct polycube maps of complicated topology and geometry. The polycube map is theoretically sound to guarantee a bijection between the 3D model surface and the polycube domain.
(ii) We compare our method with the existing polycube map construction techniques and show that the constructed polycube maps have lower angle and area distortions, and hence, are of high-quality.
(iii) We apply the constructed polycube maps to various applications, such as polycube T-splines, seamless texture synthesis, and quadrilateral and hexahedral mesh generation, and demonstrate the efficacy of our method in real-world examples.

2. Previous work

Tarini et al. pioneered the concept of polycube maps for seamless texture mapping with low angle and area distortion [1]. Wang et al. presented an intrinsic method to construct the polycube map which avoids the projection of the vertices on a 3D model to the polycube domain [4]. Wang et al. presented a technique where the user can interactively control the desired locations and the number of singularities of the polycube map (i.e., the corners in polycubes) which facilitates the manifold spline construction [5]. Lin et al. used Reeb graph to segment the surface and then developed an automatic method to construct polycube map [10]. However, their segmentation method may not work for shapes with complicated topology and geometry and does not guarantee a bijection between the polycube and the 3D model. (The detailed comparisons among different polycube map construction methods are documented in Section 5.) Because of the highly regular structure, polycube maps are very promising in seamless texture mapping and synthesis [11,2], level-of-detail [12], morphing [3], and T-spline construction [4,5].

3. Automatic polycube map construction

This section details the theory and algorithmic pipeline of automatic polycube map construction. As mentioned earlier, our method is intrinsic in that it avoids the projection of vertices from a 3D surface to the polycube domain. Therefore, the major goal is to map the input model and polycube to the canonical domains and then find the map between the canonical domains. The existing intrinsic method proposed by Wang et al. [4] requires the global parameterization, i.e., mapping the models with positive Euler characteristic $\chi>0$ to sphere $\mathbb{S}^2$, models with $\chi=0$ to Euclidean plane $\mathbb{E}^2$ and models with $\chi<0$ to hyperbolic disk $\mathbb{H}^2$. It is known that embedding models with negative Euler characteristic is error-prone when the point is very close to the boundary of the Poincaré disk due to the numerical rounding error. Therefore, Wang et al.'s method is not practical and much less numerically stable to construct polycube maps of large-scale models with negative Euler characteristics.

To construct intrinsic polycube maps in a more robust and practical way, we use a divide-and-conquer strategy, i.e., segmenting the polycube and the given 3D surface into multiple disjoint components, then constructing the piecewise polycube map for each component, and finally smoothing the map for the entire polycube domain. The key reason that we use the divide-and-conquer approach is to avoid the time consuming and error-prone global parameterization since parameterizing each segmented component (of genus-0) to the planar domain is relatively easier, more efficient, and more robust than working directly on the global shape in its entirety. Note that the straightforward gluing of the individual polycube maps have only C^0 continuity across the cutting boundaries. Therefore, we must apply a global smoothing algorithm to the entire shape to improve the quality of the polycube map. Our polycube map construction algorithm consists of five steps:

(i) Given a 3D mesh M, construct a harmonic function $f: M \to \mathbb{R}$ and extract all the critical points of f which reveal the topological structure of M (Section 3.1).
(ii) Progressively construct the polycube P using the scan-line like algorithm (Section 3.2).
(iii) Slice M and P into disjoint components M_i and P_i, i.e., $M=\bigcup M_i$, $P=\bigcup P_i$, where M_i and P_i are genus-0 open surfaces. Compute the *uniform flat metric* for M_i and P_i, and embed them to the multi-hole disks (Section 3.3).
(iv) Compute a one-to-one map between the M_i and P_i by solving a harmonic map between the multi-hole disks (Section 3.4).
(v) Smooth the polycube map by solving the harmonic map for the individual face, edge and corner charts (Section 3.5).

3.1. Extracting the topological structure

The divide-and-conquer approach requires segmentation of the input mesh to a set of genus-0 shapes. To develop a general

and automatic segmentation method, we should extract the topological structure of the input mesh M. To achieve this goal, we construct a harmonic function [13], $f : M \to \mathbb{R}$, such that

$$\Delta f = 0, \tag{1}$$

with the boundary condition

$$f(v_0) = 0 \quad \text{and} \quad f(v_1) = 1,$$

where Δ is the Laplace–Beltrami operator under the Euclidean metric (edge length) of M, whereas v_0 and v_1 are the bottom-most and top-most points on M, respectively. Note if multiple bottom(-top)-most points exist, we just pick arbitrary one.

We then find all the critical points of f whose partial derivatives vanish. These critical points can be classified into four categories (see Fig. 2):

- Maximal point where a new component starts.
- Saddle point where the handle splits.
- Saddle point where the handle merges.
- Minimal point where the current component ends.

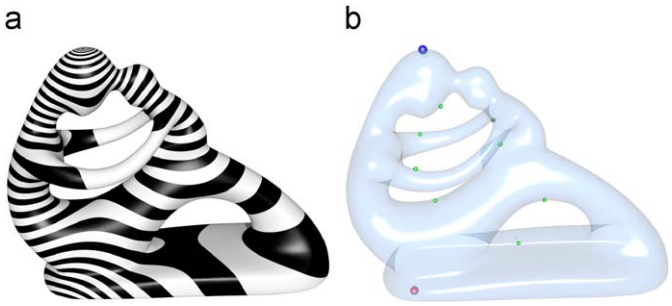

Fig. 2. Critical points of a harmonic function on a closed surface. (a) shows the harmonic function $\Delta f = 0, f(v_0) = 0$ and $f(v_1) = 1$ where v_0 and v_1 are the bottom-most and top-most points on the model, respectively. (b) The saddle, the global minimal, and the global maximal points are colored in green, red, and blue, respectively.

For a closed surface M of genus g, the number of critical points satisfies the following equation:

$$\#minimal - \#saddle + \#maximal = 2 - 2g.$$

Since the maximal and minimal points are specified by the boundary condition, the number of saddle points is always $2g$.

3.2. Constructing the polycube

We sort all the critical points in an ascending order by their z values. Let $v_0, c_1, c_2, \ldots, c_{2g}, v_1$ denote the sorted critical points including the bottom-most and top-most points which are the global minimal and maximal points, respectively. Let $z(p)$ denote the z-coordinate of point p. Then we construct $2g + 1$ horizontal cutting planes, such that

$$z_0 = \frac{z(v_0) + z(c_1)}{2},$$
$$z_1 = \frac{z(c_1) + z(c_2)}{2},$$
$$\ldots$$
$$z_{2g} = \frac{z(c_{2g}) + z(v_1)}{2}.$$

Note that because of shape symmetry, two or more critical points may have the same (or nearly the same) z-coordinate. In such a case, only one representative point is selected.

Let d_z be the user-specified parameter for the maximal distance between two adjacent cutting planes. This parameter controls how close the resulting polycube mimics the given shape. Intuitively speaking, the smaller the value of d_z, the larger the number of cutting planes, and thus, the more similar to the given shape the polycube approximation is. Note that if the distance between two consecutive cutting planes, say, z_i and z_{i+1} is greater

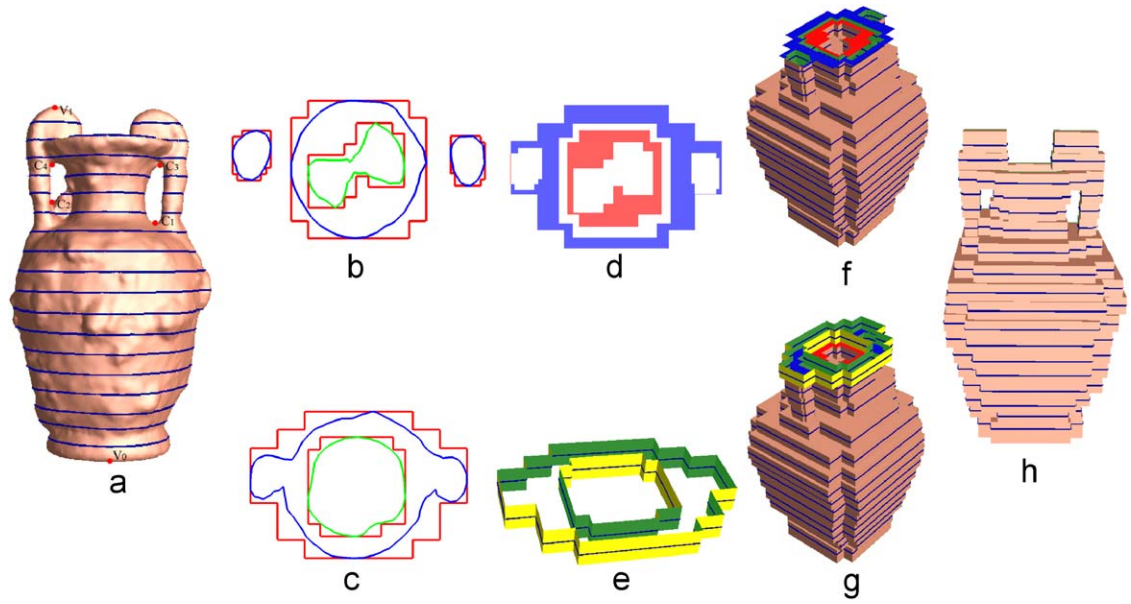

Fig. 3. Automatic polycube construction for the genus-2 Amphora model (with parameters $d_z = 0.05$ and $d_a = 0.3$). (a) shows the genus-2 Amphora model marked with saddle points $c_1, \ldots, c_4$, and the global minimum/maximum points v_0, v_1 in red. A total of 18 horizontal cutting planes slice the given shape M into 29 components, each of which is a genus-0 open surface. The blue curves show the cross-section contours of the horizontal scanning planes. (b) and (c) shows the cross-section contours (blue lines) for scanning plane 15 (with z value z_{15}) and 16 (with z value z_{16}), respectively. The inner boundaries (holes) are drawn in green. Then axis-aligned polygons Q_i (red polygons in (b) and (c)) are used to approximate the curved cross-section contours (blue and green curves in (b) and (c)). (d) shows the polygonal face at z value $(z_{15} + z_{16})/2$, which is the union of $Q_{15} - (Q_{16} \bigcap Q_{15})$ (red) and $Q_{16} - (Q_{16} \bigcap Q_{15})$ (blue). (e) shows the side face by projecting the boundaries of Q_{16} along the z-axis from $(z_{15} + z_{16})/2$ to $(z_{16} + z_{17})/2$. (f) shows the result by adding (d) to the partial polycube after processing the first 15 intersections. (g) shows the result by adding the side face (e) to the partial polycube in (f). (h) shows the final polycube. The blue lines in (h) correspond to the intersection contours in (a).

than d_z, we uniformly insert $\lfloor (z_{i+1}-z_i)/d_z \rfloor$ cutting planes in-between. Since there is at least one cutting plane between two adjacent critical points, the given shape M is sliced into multiple disjoint components, each of which is a genus-0 open surface. Then the polycube can be constructed automatically using the scan-line like algorithm as detailed below.

Let us use the genus-2 Amphora model to illustrate our idea and the key algorithmic components. There are four saddle points c_1, c_2, c_3, and c_4 shown in Fig. 3(a), whereas v_0 and v_1 are the bottom-most and top-most points, respectively. Note that c_3 and c_4 have similar z-coordinate, therefore it is not necessarily to differentiate these two points by inserting a cutting plane in-between. In our implementation, two or more critical points are considered on the same level if the difference of their z-coordinates is less than 0.01 of the height of the model. Then the z range of the given model is split into four intervals: $[v_0, c_1]$, $[c_1, c_2]$, $[c_2, c_3]$, $[c_3, v_1]$. Next, we uniformly segment the shape by several cutting planes perpendicular to z-axis as shown by the blue lines in Fig. 3(a).

The intersection between each horizontal cutting plane and M is a set of planar curves as shown in Fig. 3(b) and (c). Then we approximate these intersection curves by a set of axis-aligned polygons using a quad-tree method, i.e., starting from the bounding rectangle of this polygon, and keep subdividing it until the given approximation accuracy threshold is satisfied or the maximal subdivision level is reached. The approximation accuracy of the axis-aligned polygons p to the input curved contours c is quantitatively measured by the normalized area difference $d_a = area(p \backslash c)/area(c)$. Note that d_a is a user-specified threshold. In general, the smaller the value of d_a, the more accurate the approximated axis-aligned polygons to the curved contours, and thus, the more detailed the axis-aligned polygons are (Fig. 4).

After we get the axis-aligned polygon approximation of the curved intersection contours, we can readily construct the 3D polycube by extruding the axis-aligned polygons along the z-axis and by performing necessary CSG operations. Suppose there are n scanning planes with z values $z_1 < z_2 < \cdots < z_n$ and Q_i is the set of axis-aligned polygons for scanning plane i with z value z_i, all the boundaries of the polygons in Q_i are extruded along the z-axis from $(z_i + z_{i-1})/2$ ($z(v_0)$ for the first scanning plane) to $(z_i + z_{i+1})/2$ ($z(v_1)$ for the last scanning plane) to form the side face. The polygonal face with the z value $(z_i + z_{i-1})/2$ is computed as the union of $Q_{i-1} - (Q_i \bigcap Q_{i-1})$ and $Q_i - (Q_i \bigcap Q_{i-1})$. The polygon face will be Q_1 at the z value $z(v_0)$, and Q_n at the z value $z(v_1)$.

Fig. 3(d) and (e) show the polygon face at $(z_{15} + z_{16})/2$ and the side face for the 16-th scanning plane, respectively. Fig. 3(f) and (g) show the partial polycube after combining the polygon face and side face for 16-th scanning plane. Fig. 3(h) shows the final polycube; the contours for the axis-aligned polygon approximations are colored in blue.

3.3. Uniform flat metric and multi-hole disk

After a polycube is constructed automatically and then segmented into multiple disjoint components, we are ready for the parameterization step. Note that each segmented component is of a genus-0 open surface, therefore, the ideal parametric domain is the Euclidean disc. A common technique for the planar parameterization is solving a harmonic map with the user-specified boundary condition. However, the harmonic map is not suitable for this step since it is very hard to specify the position of boundary points for surface with multiple boundaries. Therefore, we use discrete Ricci flow [14,9,15] for the parameterization step since we only need to specify the target curvatures (rather than their positions) of the boundary points.

Suppose S is a surface with a Riemannian metric $\mathbf{g}$. Let $u : S \to \mathbb{R}$ be a scalar function on S, then $\bar{\mathbf{g}} = e^{2u}\mathbf{g}$ is also a Riemannian metric which is conformal to $\mathbf{g}$. Let K and $\bar{K}$ denote the Gaussian curvature induced by $\mathbf{g}$ and $\bar{\mathbf{g}}$, then the desired metric $\bar{\mathbf{g}}$ can be computed using

$$\frac{du(t)}{dt} = \bar{K} - K(t), \tag{2}$$

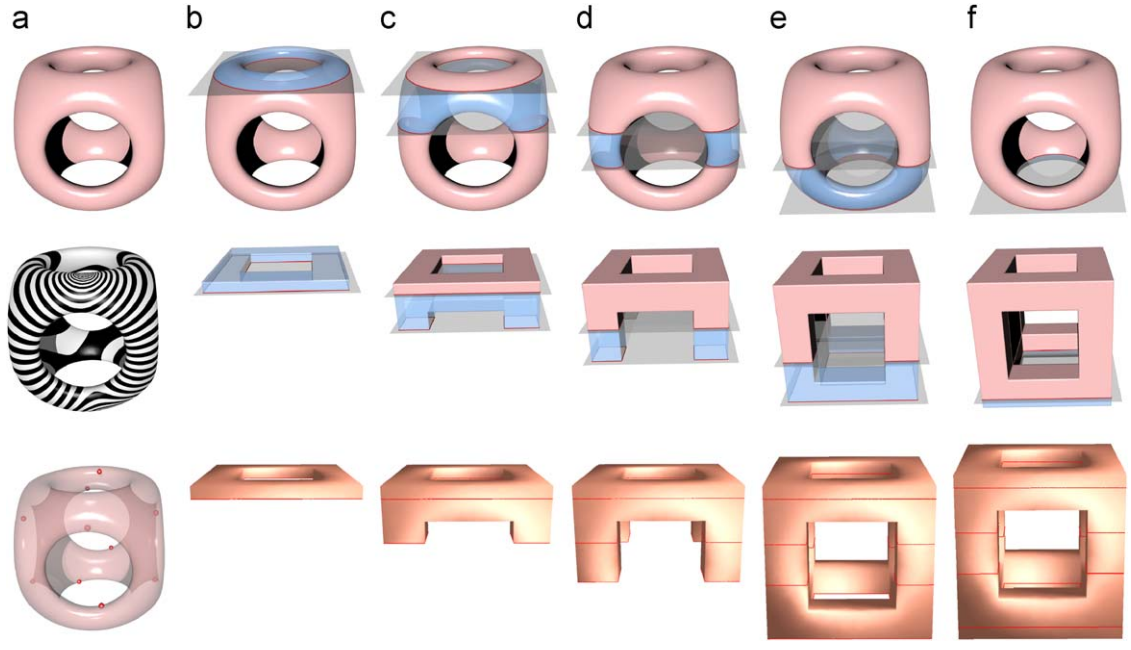

Fig. 4. Automatic polycube map construction for the genus-5 Decocube model. Because of symmetry, there are only five distinct z-values among the critical points (see (a)), so four cutting planes are used to slice the model M and P into eight components, each of which is a genus-0 open surface. We construct the one-to-one map ϕ between P_i and M_i, $i = 1, \ldots, 8$, respectively (see (b)–(f)). Note that each cutting boundary (instead of the boundary in the original shape) appears in two adjacent components. Since we use the consistent parameterization between ∂P_i and ∂M_i (see Eq. (6)), the boundary conditions of the harmonic map of two adjacent components are consistent. As a result, the piecewise polycube maps are C^0 continuous across the cutting boundaries (red curves), i.e., they are seamless.

where the initial condition is $u(0)=0$ and $K(t)$ is the Gaussian curvature induced by the metric $e^{2u(t)}\mathbf{g}$. During this curvature deformation, the metric $\mathbf{g}(t)$ is conformal to the original metric $\mathbf{g}(0)$ at any time t. To map the genus-0 open surface to Euclidean plane, we compute the uniform flat metric of S, namely, a metric $\mathbf{g}(\infty)$ which is flat everywhere inside the surface and the geodesic curvature is constant on the boundary

$$\bar{K}=0, \quad v\notin\partial S, \tag{3}$$

$$\bar{k}_g=const, \quad v\in\partial S, \tag{4}$$

where $\bar{K}_v$ and $\bar{k}_v$ are the target Gaussian and geodesic curvatures. If the total geodesic curvature on each boundary is given, such a uniform flat metric exists and is unique. Using uniform flat metric, we can map genus-0 open surface to a multi-hole disk and the map is guaranteed to be a diffeomorphism.

Note that both the given mesh M and polycube P have been segmented into multiple disjoint components M_i, P_i, $i=1,2,\ldots$, each of which is a genus-0 open surface with b ($b\geqslant 1$) boundaries $C_0\cup C_1\cup\cdots\cup C_{b-1}$. For $b\geqslant 2$, C_0 is the boundary with the longest length. We set the target curvature of interior vertices to zero, the total geodesic curvature of the first boundary $C_0-2\pi$ and the total geodesic curvature to be -2π for each of the remaining boundaries, C_i, $i=1,\ldots,b-1$. Then the total target Gaussian and geodesic curvatures satisfy the Gauss–Bonnet theorem

$$\int_S K+\int_{\partial S}k_g=\int_S\bar{K}+\int_{\partial S}\bar{k}_g=2\pi(2-2g-b), \tag{5}$$

where $g=0$. Once the target Gaussian curvatures are given, we can compute the uniform flat metric by solving the discrete Ricci flow. Then, we embed the shape to the Euclidean plane using uniform flat metric and obtain a $(b-1)$-hole disc as shown in Fig. 5.

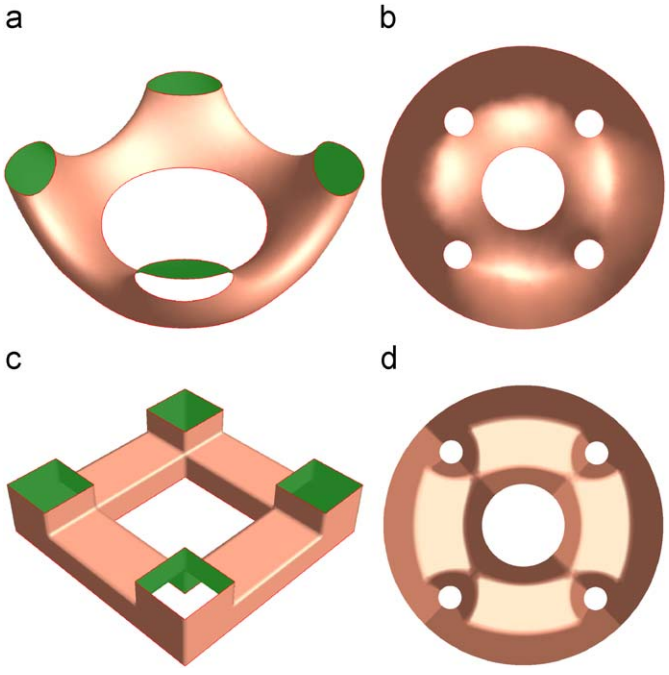

Fig. 5. We map each segmented component P_i and M_i to a multi-hole disk using uniform flat metric, where the Gaussian curvature of the interior vertices is zero and the total geodesic curvature of the boundary is constant, i.e., 2π for the outer boundary, and -2π for each hole. Since the geometry of P_i and M_i are similar, their embeddings of the uniform flat metric are consistent and stable. (a) M_i. (b) D_{M_i}. (c) P_i. (d) D_{P_i}.

3.4. Computing the piecewise map

As explained earlier, we take a "divide-and-conquer" approach in that we segment the topologically complicated shape M and polycube P into multiple disjoint components, M_i, P_i, $i=1,\ldots,$ each of which has simple topology. Then we construct a harmonic map between P_i and M_i. Note that the map between P_i and M_i is smooth for the interior vertices of M_i and P_i. In general, the map of two adjacent components (components share one common boundary) may not be continuous across the boundary. In order to ensure that the two adjacent polycube maps have C^0 continuity across the boundary (otherwise, we cannot smooth the polycube map in the next step), we must impose the boundary condition of adjacent components in a consistent way.

We first construct a one-to-one map $\psi:\partial P_i\to\partial M_i$ between the boundaries of P_i and M_i in a piecewise fashion. Note that each boundary is a closed planar curve (on the cutting plane). For each vertex $v\in\partial P_i$, let $\psi(v)\in\partial M_i$ denote its image, then we require that the map ψ_i minimizes the following distance functional:

$$\min\int_{\partial P_i}\|\psi_i(v)-v\|^2. \tag{6}$$

Solving the above optimization problem gives rise to a parameterization between the boundaries of P_i and M_i with least distortion. Note that each cutting boundary (not the boundary in the original shape) connects two adjacent components, say, P_i and P_{i+1}. Let $v\in\partial P_i$ and $v\in\partial P_{i+1}$, then the above map can guarantee that the images of v under ψ_i and ψ_{i+1} are consistent, $\psi_i(v)=\psi_{i+1}(v)$. Therefore, the resulting polycube maps of two adjacent components are C^0 continuous across the boundary, i.e., they are seamless. Fig. 6 shows an example of such a map between the boundary curves of P_i and M_i. Let D_{M_i} and D_{P_i} denote the embedding of M_i and P_i in the Euclidean plane using uniform flat metric, respectively. Similar to the intrinsic method proposed in [4], we want to construct the one-to-one correspondence between P_i and M_i by the composite map $\phi_{P_i\to M_i}=\phi^{-1}_{M_i\to D_{M_i}}\circ\phi_{D_{P_i}\to D_{M_i}}\circ\phi_{P_i\to D_{P_i}}$ as shown in the following commutative diagram:

$$\begin{array}{ccc} P_i & \xrightarrow{\phi_{P_i\to M_i}} & M_i \\ \phi_{P_i\to D_{P_i}}\downarrow & & \downarrow\phi_{M_i\to D_{M_i}} \\ D_{P_i} & \xrightarrow{\phi_{D_{P_i}\to D_{M_i}}} & D_{M_i} \end{array}$$

Harmonic map is a widely used technique to compute the mapping between two 2D regions. It is well known that a harmonic map $f:A\subseteq\mathbb{R}^2\to B\subseteq\mathbb{R}^2$ is a diffeomorphism if ∂B is convex and the boundary mapping $f(\partial A)=\partial B$ is a homeomorphism. Unfortunately, both D_{M_i} and D_{P_i} are multi-hole discs, i.e., concave shape. Thus, solving a harmonic map between D_{P_i} and D_{M_i}, i.e., $\Delta\phi=0$ and $\phi(\partial D_{P_i})=\partial D_{M_i}$, cannot guarantee a bijection in general.

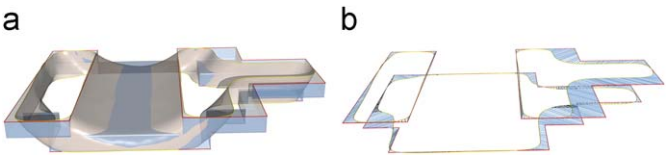

Fig. 6. Constructing the mapping between boundary curves ∂P_i to ∂M_i. (a) The boundaries of P_i and M_i are planar curves. (b) The blue lines show the map ψ_i between the vertices on ∂P_i and ∂M_i (see Eq. (6)).

To address this problem, we decompose the multi-hole discs to topological discs and then compute the harmonic map between two topological disks.

(i) We modify the topology of D_{M_i} and D_{P_i} by introducing the cuts to connect the inner circles and the outer circle such that $\bar{D}_{M_i}$ and $\bar{D}_{P_i}$ are topologically equivalent to a disk. The cuts are constructed as follows: for each inner circle, we find the shortest line to the outer circle. If the line does not pass through any other inner circles, we simply use it as cut locus; otherwise we cut through the shortest line between two inner circles to connect them. We repeat this cutting until the final shape is a topological disk.

(ii) We compute the harmonic maps $f_M : \bar{D}_{M_i} \to D$ and $f_P : \bar{D}_{P_i} \to D$ where $D \subseteq \mathbb{R}^2$ is a unit disk. Note that f_M and f_P map theboundaries $\partial\bar{D}_{M_i}$ and $\partial\bar{D}_{P_i}$ homeomorphically into the boundary of unit disk ∂D. Thus, f_M and f_P are diffeomorphisms.

(iii) We compute the harmonic map $g : D \to D$ between the two unit disks. The boundary condition is specified such that the cutting loci are mapped to each other consistently.

(iv) The composite map $\phi^{-1}_{M_i \to D_{M_i}} \circ f_M^{-1} \circ g \circ f_P \circ \phi_{P_i \to D_{P_i}}$ induces the bijection from P_i to M_i. The commutative diagram is shown as follows:

$$\begin{array}{ccccc} & P_i & \overset{\phi_{P_i \to M_i}}{\longrightarrow} & M_i & \\ \phi_{P_i \to D_{P_i}} & \downarrow & & \downarrow & \phi_{M_i \to D_{M_i}} \\ & D_{P_i} & & D_{M_i} & \\ f_p & \downarrow & & \downarrow & f_M \\ & \bar{D}_{P_i} & \overset{g}{\longrightarrow} & \bar{D}_{M_i} & \end{array} \tag{7}$$

Fig. 7 illustrates the idea to compute the diffeomorphism between P_i and M_i. Note that a direct harmonic map between D_{P_i} and D_{M_i} is not one-to-one (see the flipover in the close-up view Fig. 7(c)). We modify the topology of D_{P_i} and D_{M_i} and then construct the bijection between $\bar{D}_{P_i}$ and $\bar{D}_{M_i}$ (see Fig. 7(i)).

Fig. 4 shows the piecewise polycube map construction for the genus-5 Decocube model, which is decomposed into eight components. A bijective map is constructed for each component, and finally, the whole map is obtained by gluing all components together. Note that the piecewise polycube map is smooth for interior vertices and C^0 continuous across the cutting boundaries.

3.5. Smoothing the polycube map

The polycube map constructed by the aforementioned steps is C^∞ inside each segmented component, however, only has C^0 continuity across the cutting boundaries. Now, we further improve the quality of the polycube map by solving a harmonic

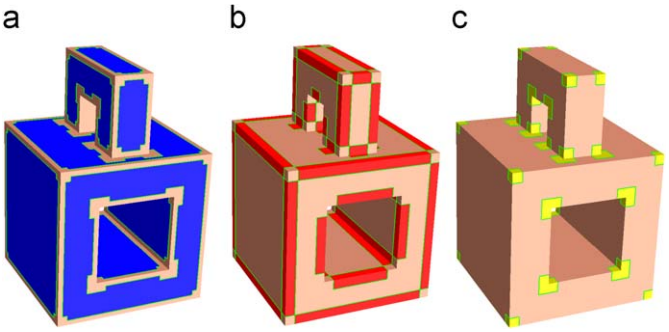

Fig. 8. A polycube is covered by face, edge, and corner charts. Each face chart (drawn in blue) covers only the interior points of the corresponding face and leaves off all the boundary edges of the face. Each edge chart (drawn in red) covers the interior points of the edges but leaves off corner vertices. Each corner chart (drawn in yellow) covers the corner. (a) Face charts. (b) Edge charts. (c) Corner charts.

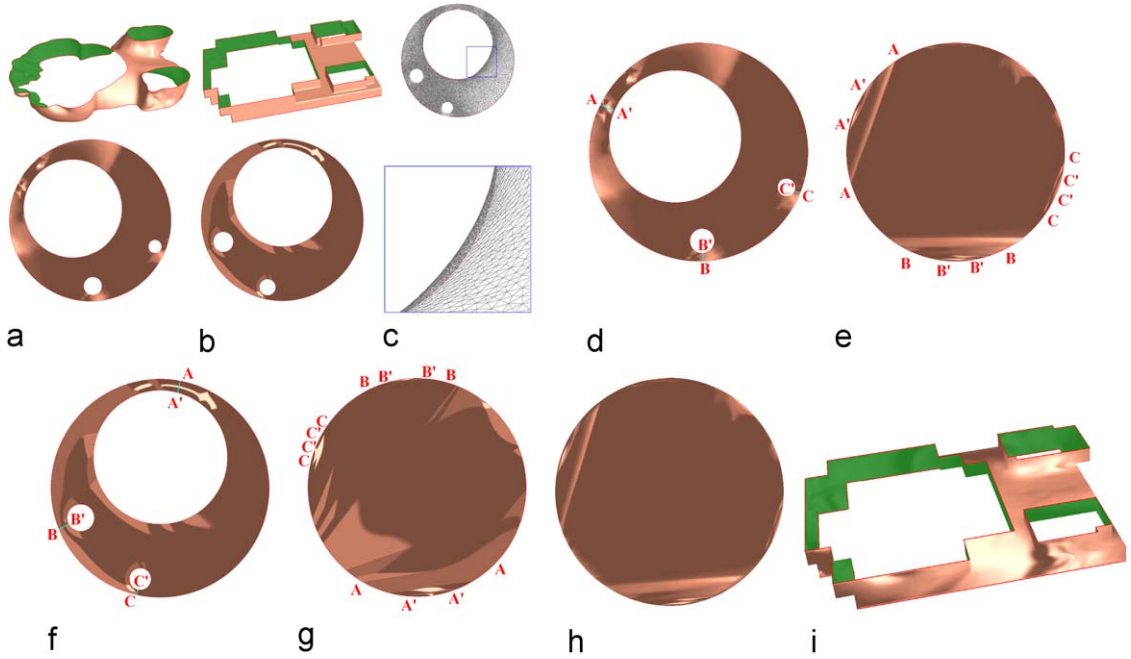

Fig. 7. Constructing a diffeomorphism between P_i and M_i (see (a) and (b)). Since D_{M_i} and D_{P_i} are not convex, the harmonic map $\phi : D_{P_i} \to D_{M_i}$ is not one-to-one. Pay attention to the flipover in the close-up view in (c). To correct this problem, we first modify the topology of D_{P_i} and D_{M_i} by introducing three cuts to connect the three inner boundaries with the outer boundary (see (d) and (e)). Then we map the modified shape $\bar{D}_{P_i}$ and $\bar{D}_{M_i}$ to unit disk. Next we compute the harmonic map between two unit disks. The boundary condition is specified that the cutting loci are mapped to each other consistently, e.g., the arc AA' in $\bar{D}_{M_i}$ to the corresponding arc AA' in $\bar{D}_{P_i}$. Finally, the polycube map from P_i to M_i is the composite map $\phi^{-1}_{M_i \to D_{M_i}} \circ f_M^{-1} \circ g \circ f_P \circ \phi_{P_i \to D_{P_i}}$. (a) $M_i \to D_{M_i}$, (b) $P_i \to D_{P_i}$, (c) $\phi_{D_{P_i} \to D_{M_i}}$, (d) $\bar{D}_{M_i}$, (e) $f_M : \bar{D}_{M_i} \to D$, (f) $\bar{D}_{P_i}$, (g) $f_P : \bar{D}_{P_i} \to D$, (h) $g : D \to D$ and (i) $\phi^{-1}_{M_i \to D_{M_i}} \circ f_M^{-1} \circ g \circ f_P \circ \phi_{P_i \to D_{P_i}}$.

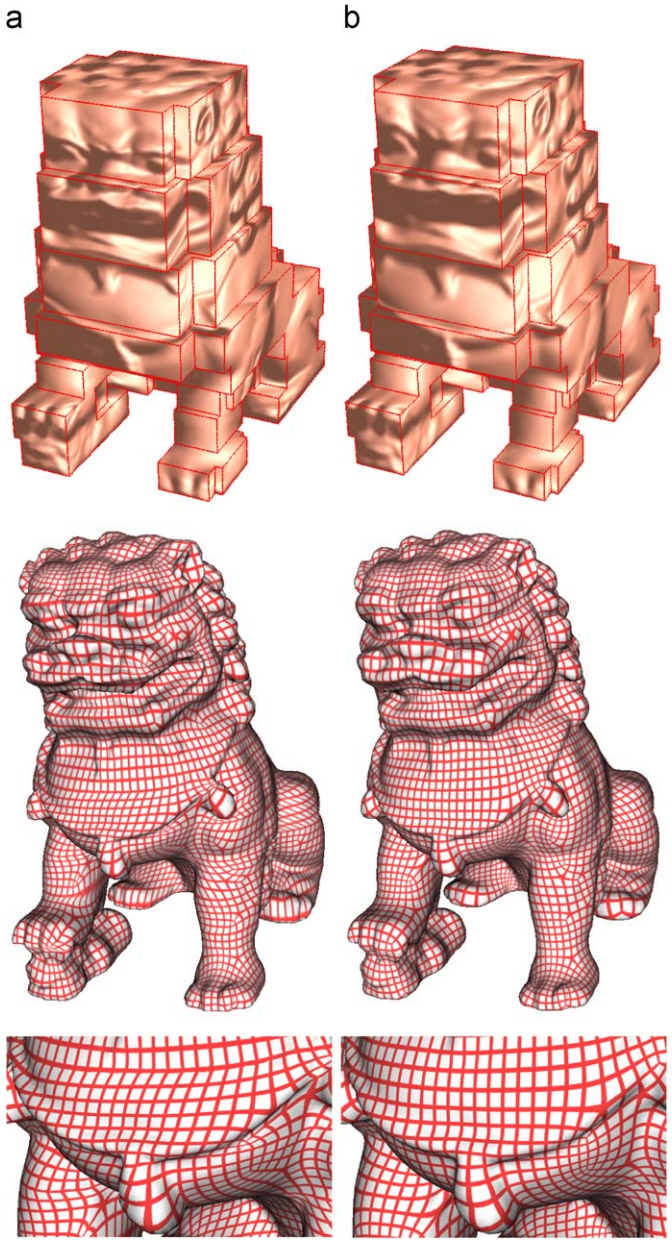

Fig. 9. Improving the polycube map by computing the harmonic map for the entire shape. The continuities across the cutting boundaries before and after improvement are C^0 and C^∞, respectively. The angle distortions before and after the improvement are 1.141 and 1.028, respectively. Please pay attention to the quality improvement on the conformality of the checkerboard texture mapping. (a) Before improvement. (b) After improvement.

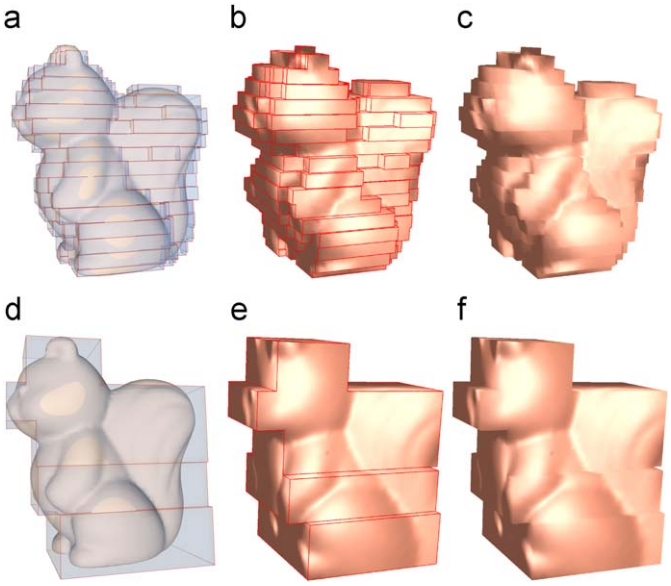

Fig. 10. The user can easily control the shape of the polycube by specifying two parameters, d_z, the maximal distance between two consecutive cutting planes, and d_a, the area difference between the axis-aligned contours and the curved intersection contours. The parameters for the Squirrel model are $d_z = 0.06$, $d_a = 0.2$ ((a)–(c)) and $d_z = 0.16$, $d_a = 0.3$ ((d)–(f)). The model is scaled to a unit cube.

Table 1
Statistics of the experimental results.

Model	g	d_z	d_a	# Δ (K)	ε_{angle}	ε_{area}	T
Amphora	2	0.06	0.3	125	1.015	1.128	12
Bimba	0	0.18	0.3	200	1.014	1.143	9
Buddha	6	N/A	N/A	300	1.051	1.316	28
Bunny	0	0.08	0.3	34	1.026	1.127	5
Dancer	1	0.05	0.25	186	1.032	1.119	19
Dragon	0	0.1	0.25	200	1.028	1.118	20
Decocube	5	0.25	0.3	60	1.026	1.089	3
Fertility	4	0.08	0.3	100	1.020	1.148	19
Gargoyle	0	0.08	0.3	75	1.021	1.155	11
Greek	4	0.05	0.3	200	1.034	1.082	23
Kitten	1	0.08	0.2	134	1.045	1.153	12
Laurana	0	0.25	0.3	125	1.003	1.122	10
Rabbit	0	0.2	0.2	27	1.032	1.142	2
Sheep	0	0.14	0.3	200	1.004	1.191	18
Squirrel	0	0.06	0.2	144	1.004	1.125	14
Totem	0	0.08	0.25	217	1.025	1.055	23

Test models are scaled to a unit cube. g, genus; # Δ, number of triangles in the given shape; d_a, the area difference between the axis-aligned contours and the curved intersection contours; d_z, the maximal distance between two adjacent cutting planes; ε_{angle}, angle distortion; ε_{area}, area distortion; T, execution time measured in minutes.

map for the entire shape. Let $\{U_c, \psi_c\}$, $\{U_e, \psi_e\}$ and $\{U_f, \psi_f\}$ denote the set of corner, edge, and face charts, respectively. As shown in Fig. 8, the corner set U_c covers the polycube corners; the edge set U_e covers the interior points of the polycube edge but leaves off corner vertices; the face set U_f covers the interior points of the polycube face but leaves off corner and edge vertices.

For any vertex $v \in U_f$ on the polycube face, v and its neighbors are co-planar. Function $\psi_f : U_f \to \mathbb{R}^2$ is defined by an orthogonal projection along the normal of the polycube face.

For any vertex $v \in U_e$ on the polycube edge, its neighbors are on two different polycube faces. Function $\psi_e : U_e \to \mathbb{R}^2$ is defined by rotating one attached polycube face 90 degrees (i.e., making v and its neighbors co-planar) followed by a projection along the normal of the un-rotated polycube face.

For any vertex $v \in U_c$ on the polycube corner, its one-ring neighbors are on three or five different polycube edges. Function $\psi_c : U_c \to \mathbb{R}^2$ maps v to the origin and its one-ring neighbors to uniformly distributed points on a unit circle.

Let $\phi : P \to M$ denote the constructed piecewise polycube map and $\phi^{-1} : M \to P$ the inverse map. Then we solvea harmonic map for the face, edge and corner charts, respectively. We consider the corner chart in the following, and the edge and face charts can be handled in a similar fashion.

Given a point $v \in U_c$ on the polycube corner, let $p = \phi(v) \in M$ denote the point on the 3D model M. The composite map $\psi \circ \phi^{-1}$:

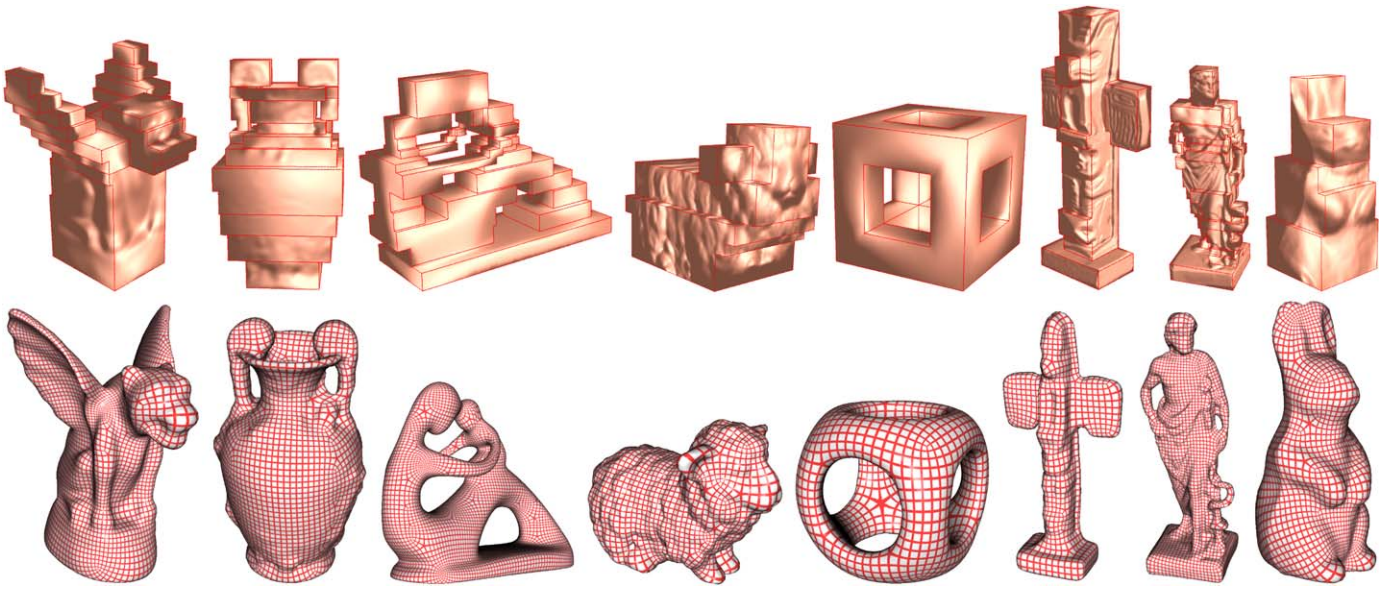

Fig. 11. Automatically-constructed polycube maps of complicated topology and geometry.

$M \to \mathbb{R}^2$ maps a 3D point p and its neighborhood to the planar domain. We can solve a harmonic map $h : \phi(U_c) \to \mathbb{R}^2$

$$\Delta h(p) = \sum_{q_i \in Nb(p)} \omega_i (h(p) - h(q_i)) = 0, \tag{8}$$

where $Nb(p)$ is the set of one-ring neighbors of p and ω_i is the *cotan* weights induced by the metric of the given mesh M. The vertices on the boundary of corner chart ∂U_c are fixed, i.e., $h(\phi(\partial U_c)) = \psi(\partial U_c)$.

Solving harmonic map for each individual face, edge and corner chart significantly improves the conformality for each chart and the charts cover the whole polycube domain, thus, the quality of the polycube map can be improved significantly. Note that all the cutting boundaries are entirely covered by the face charts, as a result, the resulting polycube map has C^∞ continuity along the cutting locus as demonstrated in Fig. 9. We should also point out that the boundaries of face, edge and corner charts are only C^0 continuous since they serve as the boundary constraints in the harmonic map. Thus, the resultant polycube map is only guaranteed to be a bijection rather than a diffeomorphism. However, based on our experiences, the entire polycube map, i.e., the interior and the boundaries of the charts, looks very smooth after solving the harmonic map for face, edge and corner charts several times (see Fig. 11).

4. Experimental results

We conducted extensive tests of our algorithm over a large variety of models ranging from genus-0 to genus-6. Computation time were measured in minutes on a workstation with 3.0 GHz CPU and 3 GB memory. Among all of the five steps in Section 3, computing the uniform flat metric takes nearly 80% of the entire time. Within our framework, the user may choose to simply specify two parameters, d_z, the maximal distance between two adjacent scanning planes, and d_a, the threshold of the normalized area difference between the axis-aligned polygons on P and the curved intersection contours on M (see Section 3.2 for the details). Fig. 10 shows how the user can easily control the shape of the polycube by specifying the above two parameters. The quality of the polycube map is measured by the angle distortion ε_{angle} and area distortion ε_{area} [16]

$$\varepsilon_{angle} = \sum_i \frac{\cot\alpha a^2 + \cot\beta b^2 + \cot\gamma c^2}{4A(\Delta_i)} A(\phi(\Delta_i)), \tag{9}$$

$$\varepsilon_{area} = \frac{1}{2}\sum_i \left(\frac{A(\Delta_i)}{A(\phi(\Delta_i))} + \frac{A(\phi(\Delta_i))}{A(\Delta_i)}\right) A(\phi(\Delta_i)), \tag{10}$$

where $\Delta_i \in P$, $\phi(\Delta_i) \in M$, $a, b, c, \alpha, \beta, \gamma$ are the side length and angles of Δ_i, and $A(\cdot)$ denotes the area. In the isometric map, $\varepsilon_{angle} = 1$ and $\varepsilon_{area} = 1$. Therefore, the closer the values of ε_{angle} and ε_{area} to 1, the better the quality of the constructed polycube maps. The statistics and performance of test cases are reported in Table 1, whereas the corresponding constructed polycube maps are shown in Fig. 11. Note that our method can produce polycube maps with very small area and angle distortions.

We have applied the constructed polycube maps to a wide range of applications, such as quadrilateral mesh generation, T-spline construction, seamless texture synthesis, and volumetric parameterization, as demonstrated in Figs. 12 and 13.

5. Comparisons

In this section, we compare our method with the existing approaches and show its advantages and disadvantages. Table 2 summarizes the key differences between our new approach and the existing methods.

Comparison with [1]. In [1], Tarini et al. first constructed the polycube manually and then warped the polycube close to the given mesh. Next, the vertices on the given mesh are projected onto the warped polycube. Finally, the polycube is warped back. This method is *extrinsic*, since it requires the projection of the vertices of the input shape M to the polycube domain P. Therefore, this method requires the user to design the polycube P manually and carefully such that it closely resembles the geometry of the input shape M, otherwise, it is difficult to warp the polycube close to M and the resulting polycube map may not be bijective (Fig. 14). Our method is *intrinsic* in that it guarantees the bijection between the given shape and the polycube. Fig. 15 shows the comparison between Tarini et al.'s method and our method on the Laurana model.

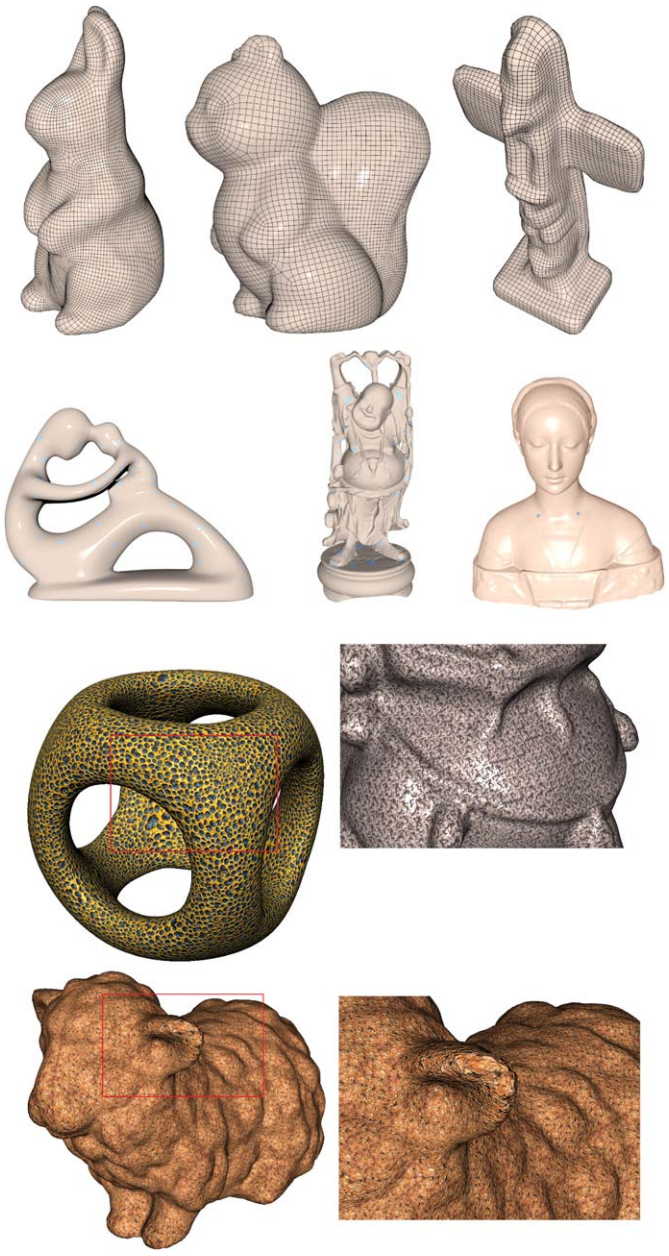

Fig. 12. Polycube maps applied to quadrilateral remeshing, T-splines and tile-based texture synthesis.

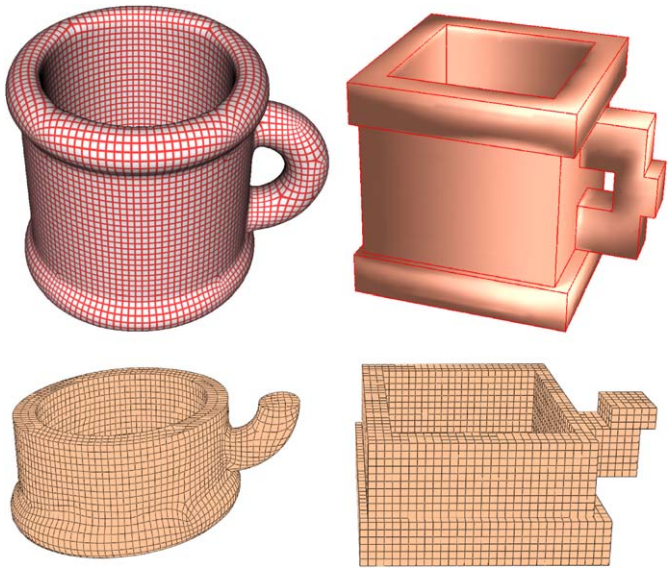

Fig. 13. Polycube serves a natural parametric domain for volumetric parameterization. Therefore, we can generate all-hexahedral meshes without any extraordinary points and T-junctions.

Comparison with [4]. Following Tarini et al.'s pioneering work, Wang et al. proposed an *intrinsic* method to construct a polycube map [4]. Instead of computing the map between the polycube P and input shape M directly, both P and M are first embedded into one of the three canonical domains, $\mathbb{S}^2$, $\mathbb{E}^2$, or $\mathbb{H}^2$, depending on the topology of M, i.e., $\pi_M : M \to D_M$ and $\pi_P : P \to D_P$ using uniformization metric, i.e., the Gaussian curvature is constant everywhere. Then by seeking the one-to-one map between the two domains $\phi_{D_M \to D_P} : D_M \to D_P$, the composition $\phi_{M \to P} = \pi_P^{-1} \circ \phi_{D_M \to D_P} \circ \pi_M$ is the desirable polycube map from M to P. This method is intrinsic in that it avoids the vertex projection from M to P. However, it is known that embedding a surface with negative Euler characteristic into $\mathbb{H}^2$ is error-prone when the point is very close to the boundary of the Poincaré disk due to the numerical rounding error. Therefore, Wang et al.'s method is not practical and much less numerically stable to construct polycube maps of large-scale models with negative Euler characteristics. Our method uses a divide-and-conquer approach which avoids computing the uniformization metric. Note that from the point of view of numerical computation and its robustness and stability, embedding the genus-0 open surface into $\mathbb{R}^2$ using *uniform flat metric* is much more robust than embedding a surface with negative Euler characteristic into hyperbolic space $\mathbb{H}^2$ using uniformization metric. Fig. 16 compares Wang et al.'s method [4] and our method on the Bimba model.

Comparison with [5]. Wang et al. proposed an interactive method to improve the polycube map for high genus surfaces [5]. The key difference between this user-controllable method and [4] is that users have full freedom to specify the number and locations of the singularities (the pre-images of polycube corners) on M and their connectivity, i.e., which pair of corners forms a polycube edge, which set of polycube edges form the polycube face, etc. This method avoids the global parameterization and can nicely produce polycube maps. However, manually specifying the polycube structure on the given mesh M is tedious and sometimes not feasible even for expert users with deep geometric insight and broad topological knowledge. For example, the minimal number of singularities for the genus-5 Decocube model is 48. It is rather time consuming and error-prone to specify both the locations and connectivity of the singularities on the input model of genus-5 for 48 points. Our method is more flexible in that the user plays with the parameters to specify how the polycube mimics the given shape and then produces the polycube map with low area and angle distortion. As demonstrated in Fig. 11, our method is capable of computing high-quality polycube maps for surfaces of complicated geometry and topology.

Comparison with [10]. Most recently, Lin et al. proposed an automatic method to construct polycube maps [10]. They first segmented the 3D model using Reeb graph and then approximate the polycube into several polycube primitives, i.e., cube, L-, O-, and U-shapes. They demonstrated their approach on bunny, 3-hole torus and horse models. However, Lin et al.'s approach may not work for the surfaces with complicated topology and geometry. Note that our approach can generate a polycube for this case, but may have large number of extraordinary points, which will be discussed in the next section. According to the report in [10], the

Table 2
Comparison with existing polycube map construction techniques.

Methods	Polycube construction	Bijective	Performance	Limitation
Tarini et al. [1]	Manual (the polycube should mimic the given shape)	No	Efficient	Difficult for surfaces with complicated topology and geometry due to vertex projection from the given shape to the polycube
Wang et al. [4]	Manual (the polycube can differ from the given shape significantly)	Yes	Computationally expensive	Difficult for surfaces with complicated topology and geometry due to the numerical unstableness in hyperbolic embedding
Wang et al. [5]	Manual (the user directly specifies the polycube structure on the given shape)	Yes	Many user interactions	Not practical for surfaces with complicated topology
Lin et al. [10]	Automatic (the user specifies several parameters to control the Reeb graph embedding and surface segmentation)	N/A	Efficient	Not practical for surfaces with complicated topology and geometry
Our method	Automatic (the user may set two parameters to specify how close the polycube mimics the given shape)	Yes	Efficient	Non-axis-aligned branches or handles will usually result in a geometrically complicated polycube

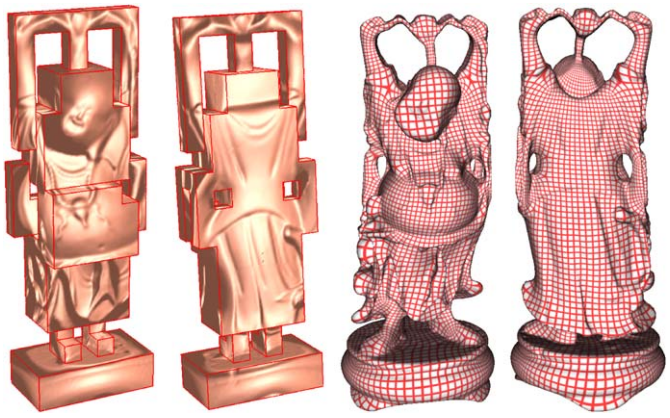

Fig. 14. Our method also applies to manually constructed polycubes.

angle and area distortions of Bunny model are 1.12 and 1.15, respectively. Our approach results in polycube map with smaller angle and area distortions 1.026 and 1.127 but with large number of extraordinary points (see Fig. 17).

6. Discussions

Manually vs. automatically-constructed polycubes. In the existing techniques of constructing polycube maps [1,4,5], the polycube maps are constructed manually. Although manual constructions work well for the models with simple topology, it is extremely tedious and time consuming to construct polycube with complicated topology. The proposed approach (Section 3.2) can generate polycubes for complicated topology and geometry. However, it usually generates polycubes which are more complicated (based on the number of corners and faces) than the manually-built ones. We should also point out that the current polycube construction stage can be simplified/replaced by any alternative method (either automatic or manual approach) in order to produce a polycube with less complexity. Fig. 14 shows the genus-6 happy Buddha model, whose polycube approximation is constructed manually. Note that using our new method we can still construct the high-quality polycube map automatically and efficiently, while accommodating the varying complexity.

Limitations. Our proposed method has certain limitations and demands further improvement in the future. First, the constructed polycube depends on the orientation of the 3D model. Different orientations may result in very different polycubes. In our implementation, we require the user to align the model before the polycube construction. Second, the proposed method will generate a geometrically complicated polycube for non-axis-aligned long branches or handles, such as the ears of the bunny model (see Fig. 17). As a result, it may cause difficulty in some applications, e.g., spline construction, since each corner of

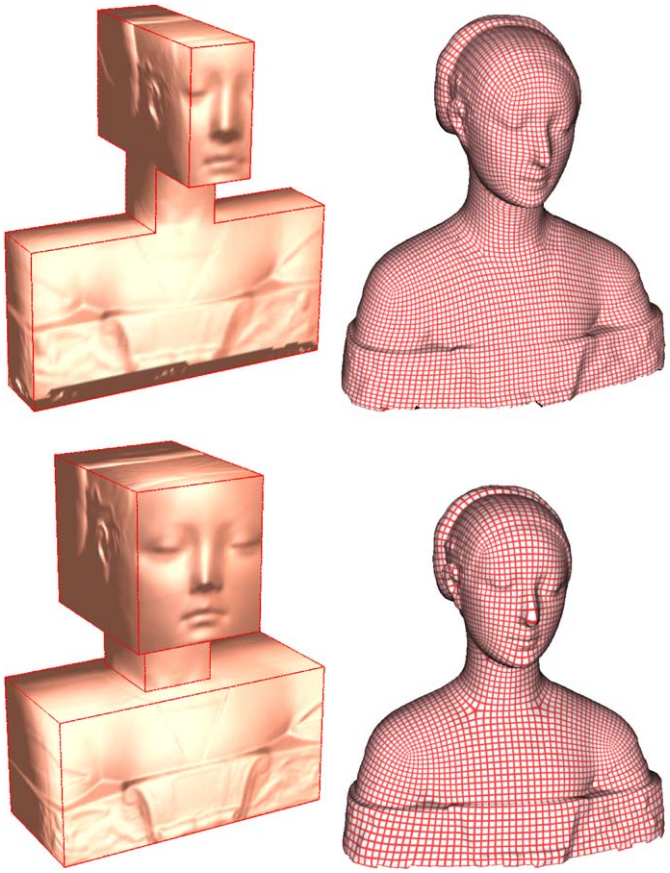

Fig. 15. Comparison with Tarini et al.'s method [1], where the polycube is constructed manually to mimic the given shape (data courtesy of Dr. Marco Tarini). The angle and area distortions of the polycube map are $\varepsilon_{angle} = 1.102$, $\varepsilon_{area} = 1.140$ (Tarini et al.'s method [1], top row) and $\varepsilon_{angle} = 1.003$, $\varepsilon_{area} = 1.122$ (our method, bottom row), respectively. Note that our new method is more flexible in that the user can easily control the shape of polycube and reduce the area and angle distortion.

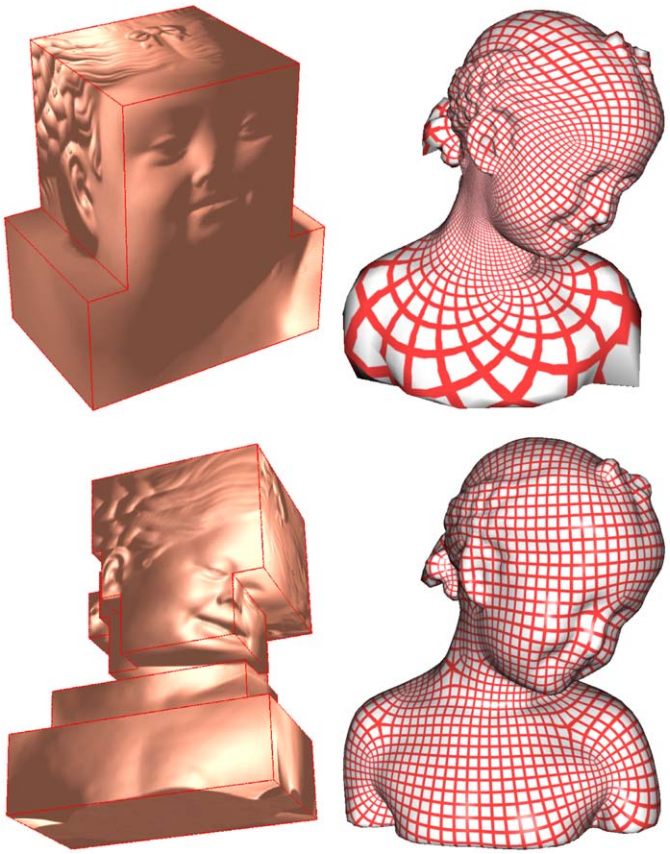

Fig. 16. The top row shows Wang et al.'s method [4] on Bima model with distortions $\varepsilon_{angle} = 1.052$ and $\varepsilon_{area} = 5.145$. The bottom row shows the results using our automatic method, where $\varepsilon_{angle} = 1.014$ and $\varepsilon_{area} = 1.143$. Note that the checkerboard texture mapping of our method is much more uniform than that of Wang et al.'s approach.

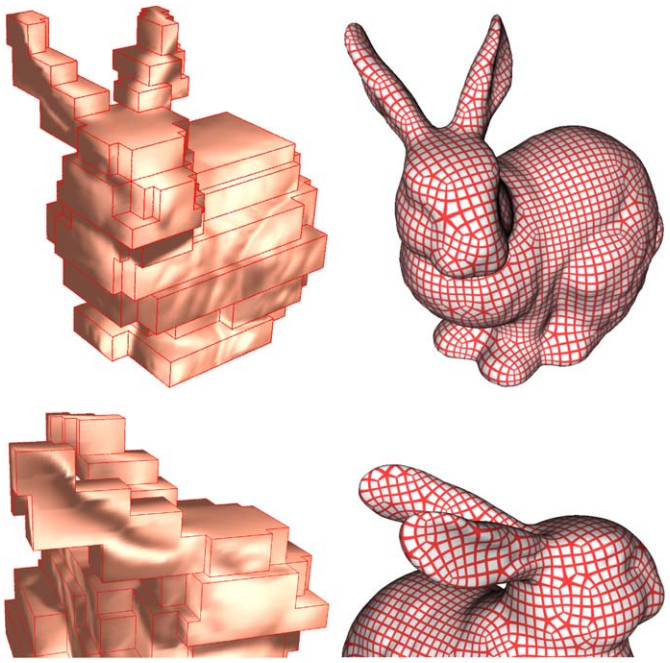

Fig. 17. The proposed method may generate a geometrically complicated polycube for non-axis-aligned long branches or handles, such as the Bunny ears. As a result, the polycube has a large number of extraordinary points (polycube corners).

polycube is an extraordinary point. Third, the polycube construction relies on the user inputs, i.e., d_a and d_z. For a shape with complicated geometry and topology, the global parameters may not generate a valid polycube. Thus, the local adaptive parameters must be used, which will result in a much more complicated implementation.

7. Conclusions

We have developed an automatic method to construct a polycube map for surfaces of arbitrary topology. The underlying theory and the entire algorithmic pipeline have been clearly documented. Within our framework, the users only need to control how close the polycube resembles the given shape by using two intuitive parameters. With no user intervention after the initial parameter setup, our new method can automatically construct a high-quality polycube map. Furthermore, our method is theoretically sound and numerically robust and stable to guarantee a one-to-one map between the constructed polycube and the given 3D model. We applied the constructed polycube maps to various graphics applications, such as seamless texture synthesis and tiling, T-spline construction, and quad mesh generation. Our experimental results have demonstrated the great promise of our new method over existing techniques. Extensive comparisons have been conducted to highlight all the advantages of our algorithm. Comprehensive discussions also pinpoint certain limitations that will lead to future research and broader application scopes.

Acknowledgment

This work was supported by the Singapore National Research Foundation Interactive Digital Media R&D Program, under research Grant NRF2008IDM-IDM004-006.

Appendix A

In the appendix, we show that our method generates a bijection between the polycube and 3D model. Here we assume that given the user-specified parameters d_a and d_z, a *valid* polycube P is constructed in step 2.

Given a close surface of genus g, we solve Laplace's equation using the z-coordinate of the top-most and bottom-most points as the boundary condition in step 1. Laplace's equation results in $2g$ saddle points. For each handle, one saddle point corresponds to the case in which the handle splits, and the other saddle point corresponds to the case in which the handle merges.

In step 2, the saddle points are sorted in the ascending order of z-coordinate. Then we construct $2g+1$ horizontal cutting planes

$$z_0 = \frac{z(v_0) + z(c_1)}{2} \quad \dots \quad z_{2g} = \frac{z(c_{2g}) + z(v_1)}{2}.$$

Note that using the top and bottom cutting planes z_0 and z_{2g}, the model will be segmented into three parts, the top part (a genus-0 open surface with 1 boundary), the middle part (a genus-g open surface with two boundaries), and the bottom part (a genus-0 open surface with one boundary). Next, $2g-1$ cutting planes, $z_1, \dots, z_{2g-1}$, are used to slice the middle part into $2g$ layers, each of which contains a set of disjoint genus-0 surfaces with at least two boundaries. Note that if two or more saddle points are on the same cutting plane, the number of layers decreases.

In step 3, we compute the uniform flat metric of each segmented component P_i or M_i using discrete Ricci flow. Ricci

flow is theoretically sound to guarantee the diffeomorphism between the genus-0 surface and the multi-hole disk.

In step 4, we construct the harmonic map between P_i and M_i using the following commutative diagram:

$$\begin{array}{cccc} & P_i & \overset{\phi_{P_i\to M_i}}{\longrightarrow} & M_i & \\ \phi_{P_i\to D_{P_i}} & \downarrow & & \downarrow & \phi_{M_i\to D_{M_i}} \\ & D_{P_i} & & D_{M_i} & \\ \phi_{D_{P_i}\to \bar{D}_{P_i}} & \downarrow & & \downarrow & \phi_{D_{M_i}\to \bar{D}_{M_i}} \\ & \bar{D}_{P_i} & \underset{\phi_{\bar{D}_{P_i}\to \bar{D}_{M_i}}}{\longrightarrow} & \bar{D}_{M_i} & \end{array} \tag{11}$$

The uniform flat metric computed using discrete Ricci flow is guaranteed to induce a diffeomorphism between P_i and D_{P_i} (and M_i and D_{M_i}) [14]. Note that a harmonic map $f : A \subset \mathbb{R}^2 \to B \subset \mathbb{R}^2$ is a diffeomorphism if ∂B is convex and the boundary condition $f(\partial A) = \partial B$ is a homeomorphism. Since $\partial \bar{D}_{P_i}$ and $\partial \bar{D}_{M_i}$ are circular, $\phi_{D_{P_i}\to \bar{D}_{P_i}}$, $\phi_{D_{M_i}\to \bar{D}_{M_i}}$, and $\phi_{\bar{D}_{P_i}\to \bar{D}_{M_i}}$ are diffeomorphism. Then the piecewise polycube map $\phi : P \to M$ is given by $\phi = \bigcup_i \phi_{P_i\to M_i}$.

In step 5, we further improve the polycube map quality by solving the harmonic maps for face, edge, and corner charts respectively. The polycube P is covered by face, edge and corner charts, $\{U, \psi\}$, where $U \in P$ is anopen set of P and $\psi : U \to \mathbb{R}^2$ maps U to the planar domain (see Fig. 8). The definition of ψ is given in Section 3.5.

Given a point $v \in U$ on a chart, let $p = \phi(v) \in M$ denote the point on the 3D model M. Then the composite map $\psi \circ \phi^{-1} : M \to \mathbb{R}^2$ maps a 3D point p to the planar domain. We solve a Laplace's equation $h : \phi(U) \to \mathbb{R}^2$ such that

$$\Delta h(p) = \sum_{q_i \in Nb(p)} \omega_i (h(p) - h(q_i)) = 0,$$

where $Nb(p)$ is the set of one-ring neighbors of p and ω_i is the cotan weights induced by the metric of the given mesh M. The boundary conditions are given by

$$h(\phi(\partial U)) = \psi(\partial U).$$

For the edge and corner charts, ψ maps ∂U to a rectangle and a unit circle, respectively. Note that $\psi(\partial U)$ is a homeomorphism and the boundary of $\psi(\partial U)$ is convex, and thus, h is a diffeomorphism.

For the face charts, ψ is defined as an orthogonal projection of the polycube face along the normal direction. Thus, ψ maps ∂U to itself and h is also a diffeomorphism.

Since the improved map h is a diffeomorphism for every point inside the face, edge and corner charts, and a homeomorphism for the points on the boundary of face, edge and corner charts. Thus, h induces a bijection between the polycube P and the 3D model M.

Appendix B. Supplementary material

Supplementary data associated with this article can be found in the online version of 10.1016/j.cag.2009.03.024.

References

[1] Tarini M, Hormann K, Cignoni P, Montani C. Polycube-maps. TOG 2004;23(3):853–60.
[2] Li H, Lo K-Y, Leung M-K, Fu C-W. Dual Poisson-disk tiling: an efficient method for distributing features on arbitrary surfaces. IEEE Transactions on Visualization and Computer Graphics 2008;14(5):982–98.
[3] Fan Z, Jin X, Feng J, Sun H. Mesh morphing using polycube-based cross-parameterization. Computer Animation and Virtual Worlds 2005;16(3–4): 499–508.
[4] Wang H, He Y, Li X, Gu X, Qin H. Polycube splines. Computer-Aided Design 2008;40(6):721–33.
[5] Wang H, Jin M, He Y, Gu X, Qin H. User-controllable polycube map for manifold spline construction. In: ACM symposium on solid and physical modeling (SPM '08); 2008. p. 397–404.
[6] Li X, Bao Y, Guo X, Jin M, Gu X, Qin H. Global optimal surface mapping for shapes of arbitrary topology. IEEE Transactions on Visualization and Computer Graphics 2008;14(4):805–19.
[9] Jin M, Kim J, Gu XD. Discrete surface Ricci flow: theory and applications. In: IMA conference on the mathematics of surfaces; 2007. p. 209–32.
[10] Lin J, Jin X, Fan Z, Wang CCL. Automatic polycube-maps. In: GMP; 2008. p. 3–16.
[11] Leung M-K, Pang W-M, Fu C-W, Wong T-T, Heng P-A. Tileable BTF. IEEE Transactions on Visualization and Computer Graphics 2007;13:953–65.
[12] Ji J, Wu E, Li S, Liu X. Dynamic LOD on GPU. In: Computer graphics international; 2005. p. 108–14.
[13] Ni X, Garland M, Hart JC. Fair Morse functions for extracting the topological structure of a surface mesh. ACM Transactions on Graphics 2004;23(3): 613–22.
[14] Chow B, Luo F. Combinatorial Ricci flows on surfaces. Journal of Differential Geometry 2003; 97–129.
[15] Gu X, Wang S, Kim J, Zeng Y, Wang Y, Qin H, et al. Ricci flow for 3D shape analysis. In: ICCV; 2007.
[16] Degener P, Meseth J, Klein R. An adaptable surface parameterization method. In: IMR; 2003. p. 201–13.
[17] Li X, Guo X, Wang H, He Y, Gu X, Qin H. Harmonic volumetric mapping for solid modeling applications. In: Symposium on solid and physical modeling; 2007. p. 109–20.