User-controllable Polycube Map for Manifold Spline Construction

Hongyu Wang
Stony Brook University
wanghy@cs.sunysb.edu

Miao Jin
Stony Brook University
mjin@cs.sunysb.edu

Ying He
Nanyang Technological University
yhe@ntu.edu.sg

Xianfeng Gu
Stony Brook University
gu@cs.sunysb.edu

Hong Qin
Stony Brook University
qin@cs.sunysb.edu

ABSTRACT

Polycube T-spline has been formulated elegantly that can unify T-splines and manifold splines to define a new shape representation for surfaces of arbitrary topology by using polycube map as its parametric domain. Naturally, The data fitting quality using polycube T-splines hinges on the construction of underlying polycube maps. However, existing methods for polycube map construction exhibit some disadvantages. For example, existing approaches for polycube map construction either require projection of points from a 3D surface to its polycube approximation, which is therefore very difficult to handle the cases when two shapes differ significantly; or compute the map by conformally deforming the surfaces and polycubes to the common canonical domain and then construct the map using function composition, which is challenging to control the location of singularities and makes it hard for the data-fitting and hole-filling processes later on.

This paper proposes a novel framework of user-controllable polycube maps, which can overcome the disadvantages of the conventional methods and is much more efficient and accurate. The current approach allows users to directly select the corner points of the polycubes on the original 3D surfaces, then construct the polycube maps by using the new computational tool of discrete Euclidean ricci flow. We develop algorithms for computing such polycube maps, and show that the resulting user-controllable polycube map serves as an ideal parametric domain for constructing spline surfaces and other applications. The location of singularities can be interactively placed where no important geometric features exist. Experimental results demonstrate that the manifold splines built upon the proposed polycube maps can achieve the same fitting accuracy by using much fewer control points, and subsequently make the entire hole-filling process much easier to accomplish.

Keywords

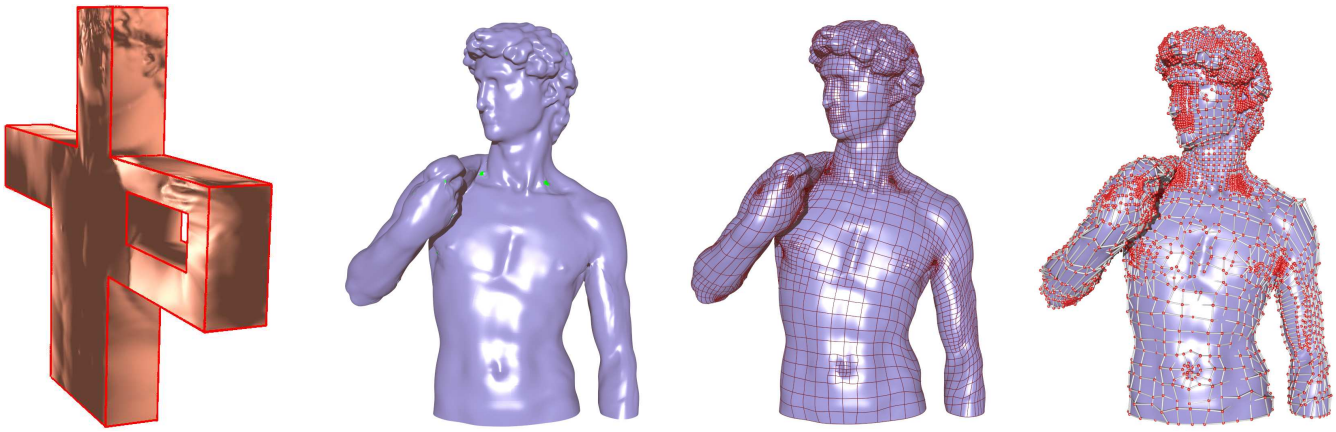
Manifold splines, polycube map, T-splines, affine atlas, solid modeling, shape computing

1. INTRODUCTION AND MOTIVATION

Manifold splines, proposed by Gu, He, and Qin [7], is a computational framework to generalize splines defined on planar domains to manifolds of arbitrary topology. Mathematically, a manifold can be equivalently treated as a set of coordinate charts in $\mathbb{R}^2$ via local parameterization, and these local charts are then glued coherently and smoothly to form a complete manifold surface. Gu *et al.* showed that there must be singularities for any closed manifold except tori [7]. Hence, for a closed manifold of $g > 1$, there has to be singularities of the atlas which can not be covered by any chart within its collection set. The existence of singularities comes from the topological obstruction, which can not be avoided within the current manifold spline framework. Given a closed domain manifold of genus g, [7] proposed a method to compute the affine structure with Euler number $|2-2g|$ extraordinary points and showed that the induced transition functions are simply the translation.

There are two research directions immediately following up Gu *et al.*'s work. One is to further reduce the number of extraordinary points. In [6], Gu *et al.* presented a method to construct manifold splines with single extraordinary point reaching their theoretic lower bound of singularity for real-world applications. They first computed a special metric of any manifold domain such that the metric becomes flat everywhere except at one point. Then, the metric naturally induces an affine atlas covering the entire manifold except this singular point. Finally, manifold splines are defined over this affine atlas. They showed that the uniformity of the metric varies drastically depending on the location of singularity.

Another direction, on the contrary, is to increase the number of extraordinary points to reduce the total area distortion in the affine atlas. In [20], Wang *et al.* proposed polycube T-splines which is a variant of manifold spline such that the metric of the affine manifold (polycube without corners) is explicitly determined by the geodesic distance on the polycube. Compared to [6], the polycube domain offers a rectangular structure which for sure facilitates geometric computing and shape analysis. Within Wang *et al.*'s framework, the user first constructs the polycube manually. Then both the 3D model and polycube are mapped to one of the canonical domains, i.e., sphere $\mathbb{S}^2$, Euclidean plane $\mathbb{E}^2$, and hyperbolic disc $\mathbb{H}^2$, depending on the topology of the input model. Next, they tried to find a map between the fundamental domains which induced the map between the input 3D shape and polycube. This method is completely different from the method introduced in [18] such that the former is intrinsic which totally avoids the projection of 3D points to the polycube domains.



(a) User-controllable Polycube map (b) Polycube T-spline (c) T-junctions on polycube spline (d) Close-up of control points

Figure 1: Polycube spline for the David Body model. (a) The user-controllable polycube map serving as the parametric domain. (b) and (c) Polycube T-splines obtained via affine structure induced by the polycube map. Note that our polycube spline is globally defined as a "one-piece" shape representation without any cutting and gluing work except at the finite number of extraordinary points (corners of the polycube). The extraordinary points are colored in green in (b). The red curves on the spline surface (see (c)) highlight the T-junctions. (d) Close-up of the spline model overlaid with the control points. The polycube T-spline contains 9781 **control points. The original model contains** $100K$ **vertices. The root-mean-square error is** 0.4% **of the diagonal of the model.**

Although the method presented in [20] can naturally compute the polycube map in an intrinsic way, it has some drawbacks: 1) there are very limited number of user-specified controls which can be used in Wang's method. For example, the user can only specify three points on the 3D model and their images on polycube for the genus zero cases. Therefore, they can not control the desired location of the extraordinary points (corners of polycube). If the extraordinary points happen to locate on the highly detailed regions, then it is difficult to fill the "holes" in the postprocessing step. 2) It is difficult to handle open surfaces in Wang *et al.*'s method. The only feasible way is to use double covering technique introduced in [9] which convert the open surfaces into closed ones. However this technique will at least double the time complexity and not practical for large scale datasets. 3) It is difficult to handle high genus models, since computing the fundamental domain of high genus model is known to be error-prone since the numerical truncated error may cause serious problems when the points are near the boundary of the Poincare disk. 4) It is difficult to control the total area distortion if the user-designed polycube differs the input 3D model too much.

In this paper, we aim to further improve the work of [20] by proposing a novel framework of user-controllable polycube maps, which overcome the aforementioned disadvantages and chanllenging, and is much more efficient and accurate. Within this framework, the current approach allows users to directly specify the extraordinary (corner) points of the polycubes on the input 3D surfaces. The location of singularities can be interactively placed where no important geometric features exist to facilitate later hole-filling process. We then develop algorithms for computing polycube maps in an intrinsic way, and show that the resulting user-controllable polycube map is an ideal parametric domain for spline constructing and other applications. Figure 1 demonstrates an example of polycube splines construction upon proposed user-controllable polycube maps.

1.1 Contributions

The specific contributions of this paper are as follows:

1. We propose a novel framework to construct user-controllable polycube maps by using discrete ricci flow. Our method is fundamentally different from Tarini *et al.*'s technique [18] and the method proposed in [20]. The user is allowed to choose the extraordinary points directly and freely on the given 3D surfaces, thus, can avoid the high detailed regions which facilitates later hole-filling process.

2. The proposed method for polycube map construction has lower area distortion compared to traditional methods and preserves small angle distortion as well. By minimizing the size of singularities on the parametric domain, we can ensure that the corresponding holes in the resulting surfaces are also small.

3. The proposed method can construct polycube map easily for *high genus* surfaces and *open* surfaces, which are usually difficult to be handled by the traditional methods as explained above.

The remainder of this paper is organized as follows. We review the related work on splines and parametrizations in Section 2. We present the details of our algorithm to construct the user-controllable polycube map of arbitrary topology in Section 3. We then discuss our construction algorithms for polycube splines and document experimental results with statistics and performance data in Section **??**. Finally, we conclude our paper in Section 4 with future research directions.

2. PREVIOUS WORK

Global surface parameterization is critical to many applications in graphics, vision and computer-aided design, such as texture mapping, remeshing, shape matching, spline construction, etc [4, 17]. Gu and Yau pioneered global conformal parameterization using holomorphic 1-forms [9]. Jin *et al* computed optimal holomorphic 1-form to reduce the area distortion of conformal mapping [12]. Kharevych *et al.* computed the conformal parameterization using circle patterns [14]. Dong *et al.* proposed a method for quadrilateral remeshing using harmonic functions [3]. This method is

theoretically equivalent to using a holomorphic one-form as [9] except that it has at least four more zero points than Gu and Yau's method. Tong *et al.* generalized harmonic 1-forms to incorporate cone singularities and used them for quadrilateral remeshing [19]. Ray *et al.* parameterized surfaces using periodic potential functions guided by two orthogonal input vector fields [16]. Dong *et al.* stuided Laplacian eigenfunctions, whose extrema are evenly distributed on the mesh. Connecting these extrema via gradient flow led to a quadrangular base mesh which can serve as the parametric domain for quadrilateral remeshing [2]. Kälberer *et al.* computed global parameterization using branch covering and demonstrated their algorithm in high quality quadrilateral remeshing [13].

Besides the Euclidean plane, other domains can also serve as the parametric domain for surface parameterization. Spherical parametrization for genus zero surfaces are introduced in [5, 8]. Jin *et al.* computed hyperbolic surface parameterization of surfaces with negative Euler characeteristic using discrete Ricci flow [11]. Khodakovsky *et al.* parameterized the surfaces using simplicial complexes [15]. Tarini *et al.* pioneered the concept of polycube maps which aims to reduce both the angular distortion and area distortion [18]. Wang *et al.* presented an intrinsic method to construct the polycube map which avoids the projection of the vertices on 3D model to the polycube domain [20].

3. CONSTRUCTION OF POLYCUBE MAPS

In this section, we explain in details our algorithm for constructing polycube maps for surfaces with arbitrary topologies.

The key differences between the techniques employed in [18, **?**] and ours in this paper are that Tarini *et al.*'s technique is trying to find a one-to-one mapping from the original surface to the polycube surface extrinsically, which typically requires the projection of points from the suface to the polycube. As a result, their method is usually quite difficult to handle cases where the surface and the polycube differ too much, because the point projection does not establish a one-to-one correspondence; the methods used in paper [**?**] compute such a mapping in an intrinsic way. They first conformally map the 3D shape and the polycube to the same canonical domains (e.g., sphere, Euclidean plane, or hyperbolic disk), then construct a map between these two domains, which induces a one-to-one map between the 3D shape and the polycube. The drawback of this intrinsic method is that user has very limited control on the whole mapping. For example, user can not control the positions of those points, which are mapped to the corner points of the polycube. If the neighborhoods of those points have rich geometric features, hole fillings will very challenging and error prone. In contrast, our method offers users the full control of the corner points, therefore, users can choose the corner points at regions with less geometric features to simplify the hole filling procedure. Furthermore, the method in [**?**] compute the polycube first, then construct the mapping between the surface and the polycube. If the polycube is changes, the mapping need to be recalculated; whereas, in our current method, we establish the mapping first, then we determine the polycube based on the mapping. If we modify the shape of the polycube, the correspondence between the surface and the polycube doesn't change. Therefore, we can adjust the shape of the polycube easily to obtain a better fitting for the polycube to the original surface. Our experimental results show that the new polycube method introduce less area distortion. Smaller area distortions around the corner points induce better hole filling results.

The polycube is constructed in the following way:

1) user set the positions and the curvatures of the corner points on the surface.
2) we deform the Riemannian metric of the surface by Ricci flow, such that all the corners have the prescribed Gaussian curvatures, and other points are flat.
3) We compute the straight lines connecting corners on the surface under the new metric to partition the surface to a collection of planar quadrilaterals.
4)We transform each quadrilaterial to a planar rectangle by setting the corner angles to be $\frac{\pi}{2}$'s, and running Ricci flow.
5) Assembly all the planar rectangles to the desired polycube. Then for vertices on the edges of the polycube, they might be mismatched. We enforce them to meet together on the edge, and use harmonic map to relax the interior of each rectangle.

In the above construction, the mapping between the polycube and the surface is automatically established. The shape of the polycube and the correspondence are fully determined by corner points. Therefore, the choices of the corner points are crucial. The followings are the important criteria for choosing the positions of the corners: the corners should be at regions with less geometric features for the purpose of better hole filling; the configuration of the corners should reflect the symmetry of the original surface.

Our experimental results show that current method gives users more freedom to design the polycube; it induces less area distortion between the surface and the polycube; it capable to handle surfaces with more complicated topologies, such as high genus surfaces or open surfaces, which are difficult to handle by conventional methods.

3.1 Discrete Ricci Flow

Suppose S is a surface with a Riemannian metric $\mathbf{g}$. Let $u : S \to \mathbb{R}$ be a function on the surface, then $\bar{\mathbf{g}} = e^{2u}\mathbf{g}$ is also a Riemannian metric of S, where u represents the area distortion and called the *conformal factor*. Furthermore, the angles between two tangle vectors at the same point measured by $\mathbf{g}$ equal to those measured by $\bar{\mathbf{g}}$, therefore, we say $\bar{\mathbf{g}}$ is *conformal* to $\mathbf{g}$. Gaussian curvatures are determined by Riemannian metrics. Let K and $\bar{K}$ are the Gaussian curvature functions induced by $\mathbf{g}$ and $\bar{\mathbf{g}}$ respectively. Then K, $\bar{K}$ and u are governed by the following Yamabe equation:

$$\bar{K} - e^{2u}K = \Delta u, \tag{1}$$

where $\Delta_{\mathbf{g}}$ is the Laplace-Beltrami operator determined by $\mathbf{g}$. This equation shows that given a desired Gaussian curvature $\bar{K}$, we can uniquely determine a Riemannian metric $e^{2u}\mathbf{g}$. The desired metric can be computed using *Ricci flow* method:

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

where the initial condition is $u(0) = 0$, $K(t)$ is the Gaussian curvature induced by the metric $e^{2u(t)}\mathbf{g}$. Riccif flow is proven to be convergent to the unique solution under the constraint that the surface area is preserved during the flow [10].

Discrete Ricci flow method is introduced in [1] and applied for solid modeling in [20]. Basically, the surface is approximated by a triangular mesh. The Riemannian metrics are approximated by the edge lengths. The Gaussian curvatures are approximated as the angle deficit from 2π at each vertex. The conformal metric is approximated by circle packing metric, where the mesh is covered by a collection of circles centered at each vertex. The circles intersect

each other. We can change the circle radii and preserve the intersection angles, then the radii and the intersection angle together determines the edge lengths, then the discrete curvatures at the vertices. Let the circle radii at vertex v_i be γ_i, u_i be $\ln \gamma_i$, then *discrete Ricci flow* has exact the same form as the smooth Ricci flow

$$\frac{du_i(t)}{dt} = \bar{K}_i - K_i(t),$$

with a normalization constraint, that during the flow the total area of the mesh is preserved. Discrete Ricci flow is a powerful tool to design edge lengths according to the user defined curvatures.

Furthermore, discrete Ricci flow is the gradient flow of the so called discrete Ricci energy. Let $\mathbf{u}$ be the vector of logarithms of radii $(u_1, u_2, \cdots, u_n)$, $\mathbf{k}$ be the vector of vertex Gaussian curvature $(K_1, K_2, \cdots, K_n$ Let $\mathbf{u}_0$ be $(0, 0, \cdots, 0)$, then the discrete Ricci energy is given by

$$E(\mathbf{u} = \int_{\mathbf{u}_0}^{\mathbf{u}} \sum_{i=1}^{n} (\bar{K}_i - K_i) du_i.$$

It is proven that the discrete Ricci energy is convex, therefore has a unique global minimum, which induces the curvature $\bar{\mathbf{k}}$. Therefore, we can use Newton's method to compute the desired metric from the user defined curvature.

3.2 Construction of Polycube Maps

Corner Selection Given a mesh M with arbitrary topology, user can design the polycube P based on the shape of the surface by directly selecting corners of P on M. The choices of the corners reflect the symmetry of M. The curvature at each corner c equals to $(2 - \frac{k}{2})\pi$, where k is the valence of c on the polycube p. Namely, protruding corners are with $\frac{\pi}{2}$, recessed corners are with $-\frac{\pi}{2}$. The total curvatures of all corners equals to $2\pi\chi(M)$, where $\chi(M)$ is the Euler-characteristic number of M. Figure 2 shows the selected corner points on Buddha model. The red corners are the protruding corners, the green corners are the recessed corners. For non-corner vertices, we set the curvature to be zeros.

Mesh Partition We use the discrete Euclidean Ricci flow to compute a new circle packing metric according to the target curvature. For any two corners c_1, c_2 on the mesh, whose correspondences are connected on the polycube, we compute the shortest path connecting them on the mesh under the new metric using Dijkstra's method. All such shortest paths segments partition the mesh to patches.Figure 4 shows the partition of the buddha mesh by this step.

Rectification Each patch is a planar quadrilateral under the new metric, but may not be a rectangle. We can use the Ricci flow method to rectify the planar quadrilateral to the rectangle by setting the target curvatures of 4 corners to be $\frac{\pi}{2}$, and all the other interior and boundary vertex curvatures to be zeros. Ricci flow can find a flat metric, the layout of the mesh under the flat metric is a rectangle. The aspect ratio of the rectangle is solely determined by original geometry of the patch. Figure **??** illustrates the rectification result.

Polycube Assembly Assemble all the rectangles to a polycube, scale each rectangle along x and y directions when it is necessary. First, we build the dual graph of the polycube, each node represents a face of the polycube, each edge corresponds to an edge. Then we use breadth first searching method to traverse the dual graph. We first embed the root face, each time we access a new face, we determine the coordinates of its corners. In this way, we can embed the whole polycube in $\mathbb{R}^3$.

If two rectangles on the polycube share one edge, make the corresponding vertices to align each other. Then we use a discrete harmonic map to relax the positions of the interior vertices of each rectangle with the fixed boundary condition.

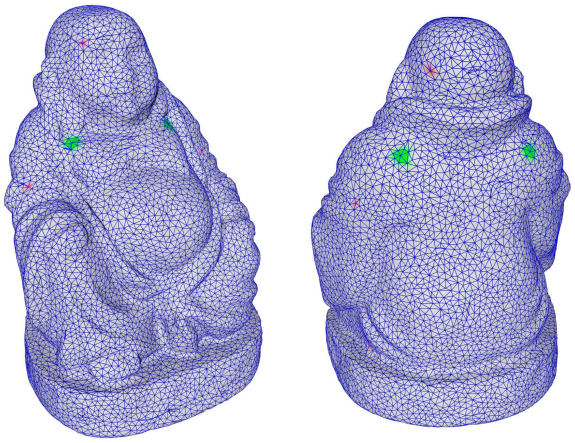


Figure 2: Corner points are marked on Buddha model, red ones with $\pi/2$ target Gaussian curvature, and green ones with $-\pi/2$ target Gaussian curvatures.

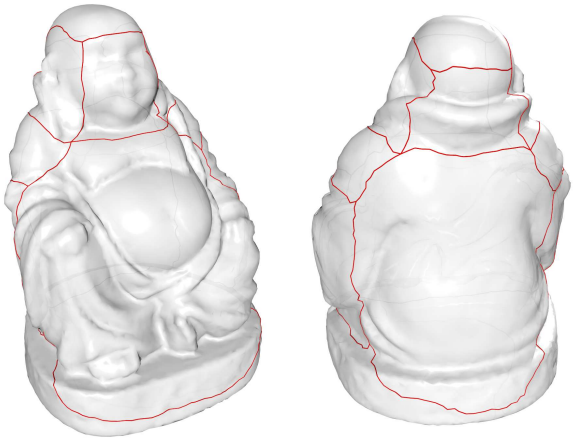


Figure 3: Geodesics between corner points are marked with sharp edges, which are computed using Dijkstra's algorithm with computed conformal metric as edge lengths.

4. CONCLUSIONS

We proposes a novel framework of user-controllable polycube maps, which can overcome the disadvantages of the conventional methods and can be generalized to complicated surfaces of arbitrary topology. The proposed method allows users to directly select the corner points of the polycubes on the original 3D surfaces, then construct the polycube maps by using the new computational tool of discrete Euclidean ricci flow. The resulting polycube map usually has lower area distortion and small angle distortion which are pleasing for spline construction. We develop algorithms for computing such polycube maps, and show that the resulting user-controllable polycube map serves as an ideal parametric domain for constructing spline surfaces. The location of singularities can be interactively

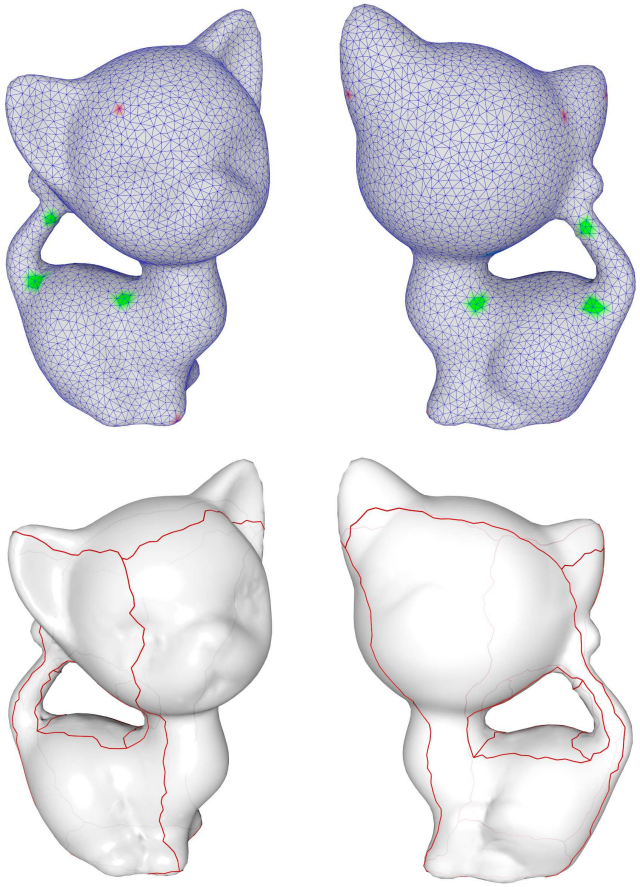


Figure 4: Geodesics between corner points are marked with sharp edges, which are computed using Dijkstra's algorithm with computed conformal metric as edge lengths.

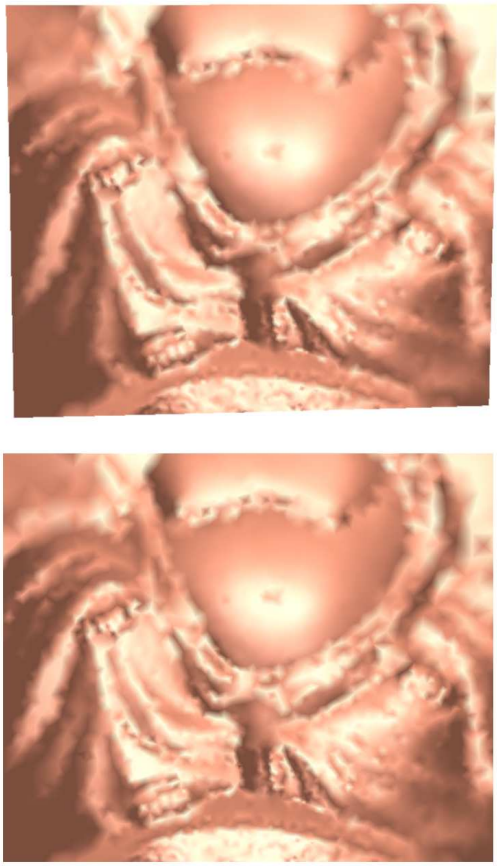


Figure 5: One patch from Buddha model, also one face for the polycube. (a) irregular, before adjusting. (b) regular, after adjusting.)

placed where no important geometric features exist. Experimental results demonstrate that the manifold splines built upon the proposed polycube maps can achieve the same fitting accuracy by using much fewer control points, and subsequently make the entire hole-filling process much easier to accomplish. Through extensive experiments on various models, we demonstrate that proposed user-controllable polycube maps are well suited for spline construction of complicated geometric models of arbitrarily complicated topology.

Acknowledgement

The Buddha, venus models are provided courtesy of INRIA by the AIM@SHAPE Shape Repository. The third author is partially supported by NTU SUG 19/06.

5. REFERENCES

[1] B. Chow and F. Luo. Combinatorial Ricci flows on surfaces. *J. Differential Geom.*, 63(1):97–129, 2003.

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

[3] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Comput. Aided Geom. Des.*, 22(5):392–423, 2005.

[4] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 157–186. Springer Verlag, 2005.

[5] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3d meshes. *ACM Trans. Graph.*, 22(3):358–363, 2003.

[6] X. Gu, Y. He, M. Jin, F. L. 0002, H. Qin, and S.-T. Yau. Manifold splines with single extraordinary point. In *Symposium on Solid and Physical Modeling*, pages 61–72, 2007.

[7] X. Gu, Y. He, and H. Qin. Manifold splines. In *ACM Symposium on Solid and Physical Modeling*, pages 27–38, 2005.

[8] X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, and S.-T. Yau. Genus zero surface conformal mapping and its application to brain surface mapping. *IEEE Transactions on Medical Imaging*, 23(8):945–958, Aug. 2004.

[9] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Processing*, pages 127–137, 2003.

[10] R. S. Hamilton. The Ricci flow on surfaces. *Mathematics and general relativity*, 71:237–262, 1988.

[11] M. Jin, F. L. 0002, and X. Gu. Computing surface hyperbolic

structure and real projective structure. In *Symposium on Solid and Physical Modeling*, pages 105–116, 2006.

[12] M. Jin, Y. Wang, S.-T. Yau, and X. Gu. Optimal global conformal surface parameterization. In *IEEE Visualization*, pages 267–274, 2004.

[13] F. Kälberer, M. Nieser, and K. Polthier. Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum*, 26:375–384, 2007.

[14] L. Kharevych, B. Springborn, and P. Schröder. Discrete conformal mappings via circle patterns. *ACM Trans. Graph.*, 25(2):412–438, 2006.

[15] A. Khodakovsky, N. Litke, and P. Schröder. Globally smooth parameterizations with low distortion. *ACM Trans. Graph.*, 22(3):350–357, 2003.

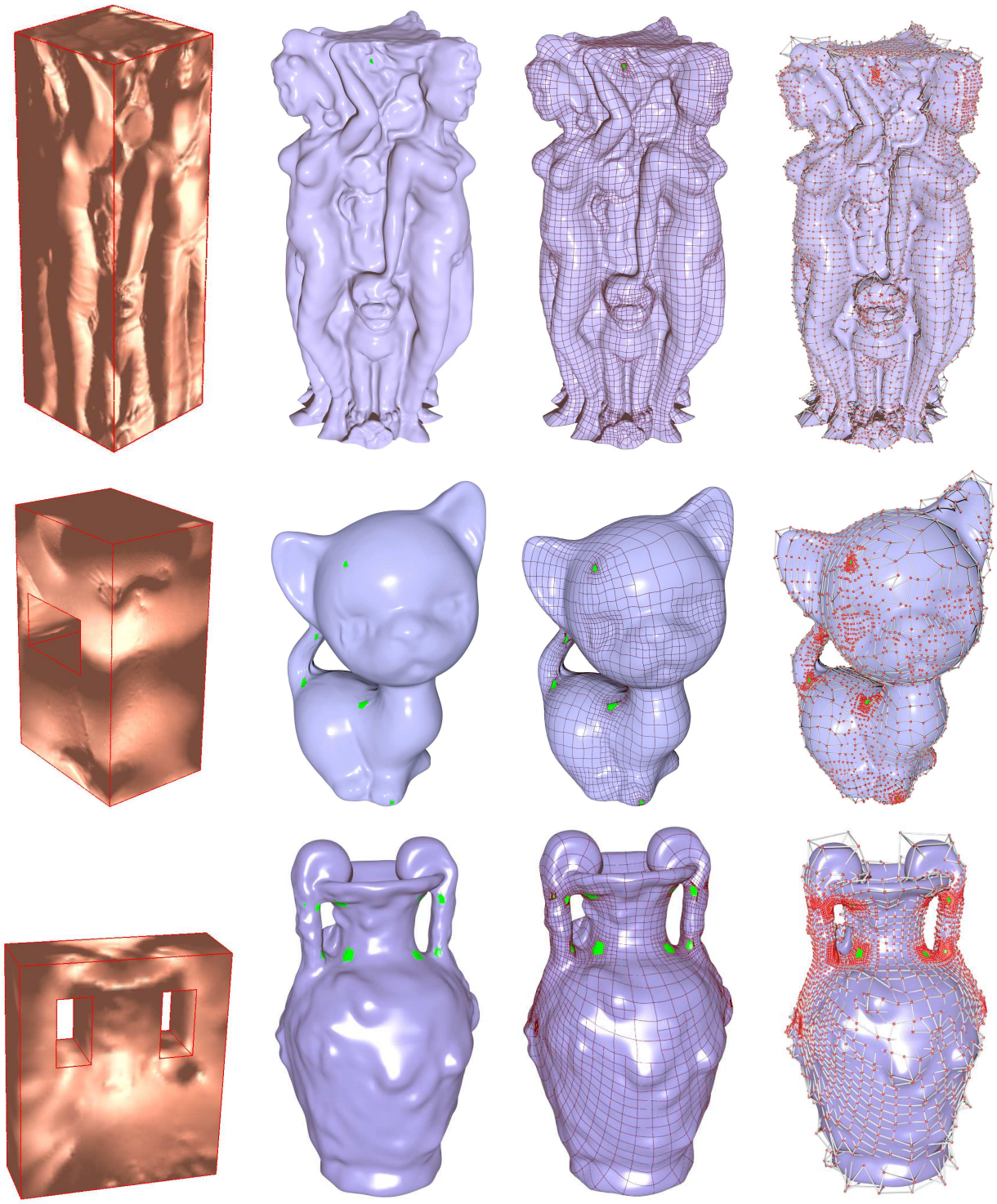
[16] N. Ray, W. C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Transactions on Graphics*, 2006.

[17] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision.*, 2006.

[18] M. Tarini, K. Hormann, P. Cignoni, and C. Montani. Polycube-maps. *ACM Trans. Graph.*, 23(3):853–860, 2004.

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

[20] H. Wang, Y. He, X. Li, X. Gu, and H. Qin. Polycube splines. In *Symposium on Solid and Physical Modeling*, pages 241–251, 2007.



(a) Polycube map (b) Polycube T-spline (c) Polycube T-spline (d) Control points

Figure 6: Examples of Polycube T-splines.