Abstract

Polycube T-splines have been formulated elegantly that can unify T-splines and manifold splines to define a new class of shape representations for surfaces of arbitrary topology by using polycube map as its parametric domain. In essence, the data fitting quality using polycube T-splines hinges upon the construction of underlying polycube maps. Yet, 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 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 proposed polycube maps introduce lower area distortion and retain small angle distortion as well, and subsequently make the entire hole-filling process much easier to accomplish. Specific applications include reverse engineering from discrete points to spline-based continuous models and the integration of design and analysis.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: Manifold Splines, Polycube Map, T-Splines, Affine Atlas, Solid Modeling, Shape Computing

1 Introduction

Manifold splines, proposed by Gu, He, and Qin [Gu et al. 2005], have become a powerful computational framework to generalize splines defined on planar domains to manifolds of arbitrary topology. Conventional splines are constructed based on affine invariants. If a manifold is covered by an atlas, such that all chart transition functions are affine, then manifold splines can be globally defined. Due to the topological obstruction, such an affine atlas may not exist for general surfaces. Therefore, singularities are unavoidable. Given a closed domain manifold of genus $g$, [Gu et al. 2005] 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 Gu et al.’s work. One is to further reduce the number of extraordinary points. In [Gu et al. 2007], 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, in sharp contrast, is to increase the number of extraordinary points to reduce the total area distortion in the affine atlas. In [Wang et al. 2007], Wang et al. proposed polycube T-splines which are a variant of manifold splines such that the metric of the affine manifold (polycube without corners) is explicitly determined by the geodesic distance on the polycube. Compared with [Gu et al. 2007], the polycube domain offers a rectangular structure which necessarily facilitates subsequent 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 $S^2$, Euclidean plane $\mathbb{E}^2$, and hyperbolic disc $\mathbb{H}^2$, depending on the topology of the input model. Next, they try to find a map between the fundamental domain which induce the map between the input 3D shape and polycube. This method is completely different from the method introduced in [Tarini et al. 2004] such that the new approach is essentially intrinsic which completely avoids the direct projection of 3D points to the polycube domains.

Although the method presented in [Wang et al. 2007] can naturally compute the polycube map in an intrinsic way, it has some drawbacks: 1) there is very little user control which can be employed interactively in Wang et al.’s method. For example, the user can only specify three points on the 3D model and their images on polycube for genus-zero cases. Therefore, they can not directly control the desired location of extraordinary points (polycube corners). If the extraordinary points happen to locate on the highly detailed, feature-rich 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 [Gu and Yau 2003] which converts the open surfaces into closed ones. However, this technique will at least double the time complexity and not practical for large-scale complex datasets; 3) It is difficult to handle high genus models, since computing the fundamental domain of any high genus model is known to be error-prone since the numerically-truncated error may cause serious problems when the points are near the boundary of the Poincare disk; and 4) It is difficult to control the total area distortion if the user-designed polycube differs from the given 3D model significantly.

In this paper, we aim to further improve the work of [Wang et al. 2007] by developing a novel framework of user-controllable polycube maps, which overcomes the aforementioned disadvantages and tackles the technical challenges, 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 in order to facilitate the subsequent 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 construction, reverse engineering, and other applications. Figure 1 demonstrates one example of polycube spline construction upon 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 [Tarini et al. 2004] and the method proposed in [Wang et al. 2007]. The user is allowed to choose the extraordinary points directly and freely on the given 3D surfaces, thus, can avoid the high detailed feature-rich regions which then facilitates later hole-filling processes.

2. The proposed method for polycube map construction has lower area distortion compared with traditional methods and accommodates 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 the benefits of the user-controllable polycube map in manifold spline construction in Section 4. Finally, we conclude our paper in Section 5.

2 Previous Work

This section briefly reviews prior research on parameterization and spline computation.

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 [Floater and Hormann 2005; Sheffer et al. 2006]. Gu and Yau pioneered global conformal parameterization using holomorphic 1-forms [Gu and Yau 2003]. Jin et al. computed optimal holomorphic 1-form to reduce the area distortion of conformal mapping [Jin et al. 2004]. Kharevych et al. computed the conformal parameterization using circle patterns [Kharevych et al. 2006]. Dong et al. proposed a method for quadrilateral remeshing using harmonic functions [Dong et al. 2005]. Tong et al. generalized harmonic 1-forms to incorporate cone singularities and used them for quadrilateral remeshing [Tong et al. 2006]. Ray et al. parameterized surfaces using periodic potential functions guided by two orthogonal input vector fields [Ray et al. 2006]. Kälberer et al. computed global parameterization using branch covering and demonstrated their algorithm in high-quality quadrilateral remeshing [Kälberer et al. 2007].

Besides the Euclidean plane, other domains can also serve as the parametric domain for surface parameterization. Spherical parameterization for genus zero surfaces is introduced in [Gotsman et al. 2003; Gu et al. 2004]. Jin et al. computed hyperbolic surface parameterization of surfaces with negative Euler characteristic using discrete Ricci flow [Jin et al. 2006]. Khodakovsky et al. parameterized the surfaces using simplicial complexes [Khodakovsky et al. 2003]. Tarini et al. pioneered the concept of polycube maps which aims to reduce both the angular distortion and area distortion [Tarini et al. 2004]. 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 [Wang et al. 2007].
T-spline was presented by Sederberg et al. in [Sederberg et al. 2003] as a generalization of the traditional non-uniform B-spline surfaces. By permitting T-junctions, T-splines enable a true local refining mechanism without introducing additional, unnecessary control points in nearby regions. They also developed an algorithm to convert industry standard NURBS surfaces into T-spline surfaces, in which a large percentage of superfluous control points are eliminated [Sederberg et al. 2004]. Zheng et al. developed a technique for adaptively fitting T-splines to functional data [Zheng et al. 2005]. Recently, Li et al. introduced an automatic technique to convert polygonal meshes to T-splines using periodic global parameterization [Li et al. 2006]. Wang and Zheng addressed the issue of control point removal for T-spline surfaces [Wang and Zheng 2006].

Manifold splines have been introduced by Gu et al. [Gu et al. 2005] which is a general theoretical framework to generalize spline surfaces defined over planar domains to any manifold domain of arbitrary topologies. He et al. further developed modeling techniques for applications of manifold splines using triangular B-splines [He et al. 2006a], Powell-Sabin splines [He et al. 2005], and T-splines [He et al. 2006b]. Most recent research results along this direction also include [Gu et al. 2007; Wang et al. 2007].

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 [Tarini et al. 2004; Wang et al. 2007] 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 surface to the polycube. As a result, their method is usually quite difficult to handle cases where the surface and the polycube domain differ significantly, because the point projection does not guarantee a one-to-one correspondence; the methods used in [Wang et al. 2007] 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 in practice is that the user has very limited control on the global mapping. For example, users cannot control the positions of extraordinary (corner) points. If the vicinity regions of those points have rich geometric features, the subsequent hole-filling processes will unavoidably become very challenging and error prone. In contrast, our new method offers users the full control of the corner point placement, therefore, users can choose the corner points at regions with fewer geometric features to simplify the hole filling procedure. Furthermore, the method in [Wang et al. 2007] first builds the polycube manually, then construct the mapping between the surface and the polycube. If the polycube changes, the mapping need to be re-calculated completely; whereas, in our current method, we establish the mapping first, then determine the polycube based on the mapping. If we modify the shape of the polycube, the correspondence between the surface and the polycube does not change. Therefore, we can adjust the shape of the polycube easily to obtain a better fitting for the original surface defined over the polycube. Our experimental results show that the introduced polycube maps also introduce lower area distortion. Lower area distortion in the vicinity of corner points will ensure better hole filling results.

The polycube map is constructed in the following way:

1. Users 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 quadrilateral to a planar rectangle by setting the corner angles to be $\pi/2$’s and running Ricci flow.
5. Assembly all the planar rectangles to the desired polycube.

Next we will give a brief introduction to discrete Ricci flow in 3.1, then the detailed algorithm for polycube map construction will be presented in 3.2.
induced by \( g \) and \( \tilde{g} \), respectively. Then \( K, \tilde{K} \), and \( u \) are governed by the following Yamabe equation:

\[
\tilde{K} = e^{2u} K (\Delta g u + K),
\]

where \( \Delta g \) is the Laplace-Beltrami operator determined by \( g \). The desired metric can be computed using Ricci flow method:

\[
\frac{du(t)}{dt} = \tilde{K} - K(t),
\]

where the initial condition is \( u(0) = 0 \), and \( K(t) \) is the Gaussian curvature induced by the metric \( e^{2u(t)} g \). Ricci flow is proven to be convergent to the unique solution under the constraint that the surface area is preserved during the flow [Hamilton 1988].

Discrete Ricci flow method is introduced in [Chow and Luo 2003] and discrete Euclidean Ricci flow is applied for solid modeling in [Gu et al. 2007]. 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 with each other. We can change the circle radii and preserve the intersection angles. Let the circle radii at vertex \( v_i \) be \( \gamma_i \), \( u_i \) be \( \ln \gamma_i \), then discrete Ricci flow has exact the same form as the smooth Ricci flow (Eqn.2) 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 \( u \) be the vector of logarithms of radii \((u_1, u_2, \cdots, u_n)\), \( K \) be the vector of vertex Gaussian curvature \((K_1, K_2, \cdots, K_n)\). Let \( u_i \) be \((0, 0, \cdots, 0)\), then the discrete Ricci energy is given by

\[
E(u) = \int_{u_i} u_i \sum_{i=1}^n (K_i - K_i) du_i.
\]

It is proven that the discrete Ricci energy is convex, therefore has a unique global minimum, which induces the curvature \( K \). Hence, we can use Newton’s method to compute the desired metric from the user-specified curvature.

3.2 Construction of Polycube Maps

The fully user-controllable polycube map can then be interactively built using the following procedure.

Corner Selection. Given a mesh \( M \) with arbitrary topology, users 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 \((2 - \frac{1}{k})\pi\), where \( k \) is the valence of \( c \) on the polycube \( p \). Namely, protruding corners are with \( \frac{2}{k} \), recessed corners are with \( -\frac{2}{k} \). The total curvatures of all corners equals \( 2\pi \chi(M) \), where \( \chi(M) \) is the Euler-characteristic number of \( M \). Figure 2 shows the selected corner points on Buddha model. For non-corner vertices, we set the curvature to be zero.

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 (Figure 3). All such shortest paths segment the mesh to patches. Figure 4(a) shows one patch from the partition of the Buddha mesh using this step, which corresponds to one face of the polycube.

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 zero. 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 4(b) illustrates the rectification result.

Polycube Assembly. Assemble all the rectangles to a polycube, scale each rectangle whenever 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 \( \mathbb{R}^3 \). Figure 5 shows the polycube map for genus-zero buddha model. Please pay attention to the corner points correspondence between the model (Figure 3) and the resulting polycube (Figure 5).

If two rectangles on the polycube share one edge, enforce 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.

Using the above construction procedure, 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 become critical. The followings are the important criteria for choosing the corner positions: (1) the corners should be at regions with fewer geometric features for the purpose of better hole filling; and (2) the configuration of the corners should reflect the symmetry of the original surface.

Our experimental results in Section 4 will show that the current method gives users much more freedom to design the polycube; it induces lower area distortion between the surface and the polycube; and it is capable of handling surfaces with more complicated topologies, such as high genus surfaces or open surfaces, which are difficult to handle by using conventional methods.
Figure 6: Genus-one polycube map induced by Euclidean structure. The genus one Kitten model $M$ in (a) is conformally mapped to the Euclidean plane in (b). The fundamental domain is a rectangle region enclosed by the green boundary in (b). Then, a polycube $P$ in (c) is also parameterized over the rectangular region in the same way in (d). By matching the two fundamental regions in (b) and (d) via an affine map, the polycube map for the Kitten model is established in (e). (f) demonstrates the initial fitting result based on (e).

4 Defining Manifold Splines Over Polycube Maps

As shown in [Wang et al. 2007], the polycube map of given 3D surfaces naturally induces the affine structure with a finite number of extraordinary points (corners). Any planar spline schemes which satisfy the parametric affine invariant property can be generalized to a manifold domain via affine structure [Gu et al. 2003]. Therefore, spline surfaces can be defined over the polycube map directly. The data fitting quality using polycube T-splines depends heavily on the construction of underlying polycube maps. In this section, we explain with examples that the introduced user-controllable polycube maps are better for manifold spline construction compared with traditional methods [Wang et al. 2007].

In [Wang et al. 2007], 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. Figure 6 demonstrates the above mapping method for constructing a polycube map of the genus-one kitten model. Both the kitten model and its polycube approximation are parameterized using the holomorphic 1-form method. Their fundamental polygons are extracted and mapped by an affine map. The affine map further induces a bijective map between the kitten model and the polycube. The drawback of this intrinsic method is that the user has very limited control over the entire mapping. For example, the user cannot control the positions of extraordinary points. If the vicinity regions of those points have important geometric features, the following hole-filling process will be very challenging and error prone. Figure 6(f) shows the initial fitting result from the polycube map in Figure 6(e). One of the corner points is at the tip of the ear, which is difficult to fill, compared with other smooth regions.

Since our method allows users to select the locations of corner points on the 3D surfaces directly, we can put corner points at regions where no important geometric features exist. As shown in Figure 7(a), we place corner points at smooth regions. The resulting polycube map is shown in Figure 7(b). Figure 7(c) demonstrates the initial fitting result from the user-controlled polycube map. We can also tell from the shape of T-curves (Figure 6(f), and Figure 7(c)) that the introduced polycube map is more conformal compared with [Wang et al. 2007]. The affine map used in [Wang et al. 2007] to align the two fundamental polygons in $E^2$ sacrifices quite a lot the conformality of the mapping. We measure the angles of each triangle in $P$ and compare with the original ones in $M$. The distribution of the angle ratio is illustrated in Figure 8. We can see that the ratio is mainly concentrated around 1 (which should be ideal) using our new method, and it shifts away from 1 for the method in [Wang et al. 2007] due to the affine map being used to align the fundamental polygons.

Furthermore, the introduced polycube map usually induces lower area distortion. For the examples shown in Figure 6 and Figure 7, area distortion for the method employed in [Wang et al. 2007] is 8.7548, while it is 0.8477 for the user-controllable polycube map. The area distortion is calculated as $\sum_{\Delta \subset M} \left( \frac{\text{area}(\Delta')}{|\text{area}(\Delta)} - 1 \right)^2 \times \text{area}(\Delta)$. The areas of the input model $M$ and the resulting polycube $P$ are normalized to 1 before the computation. $\Delta'$ is the image of the triangle $\Delta$ of $M$ on $P$. Lower area distortion ensures that the holes on the resulting spline surface will be small by making the holes around corner points on the domain polycube small.

Another advantage of the proposed new method for spline construction is that it is capable of handling surfaces with more complicated topologies, such as high genus surfaces or open surfaces, without introducing extra overhead. In reality, these surfaces are very difficult to handle by using conventional methods. Figure 1 shows the polycube map for David body model which is a genus-one surface with two boundaries. It is difficult to construct the polycube
Figure 8: Comparison of angle ratio distribution: (a) The result for the polycube map in [Wang et al. 2007] (Figure 6(e)); (b) The result for the user-controllable polycube map (Figure 7(b)).

map for it by using the traditional method. The only feasible way is to use double covering technique which converts the open surfaces into closed ones. However, this technique is far less efficient because it at least doubles the topology complexity and is not practical for complicated, large-scale datasets. By allowing the mechanism of selecting the corner points on the surfaces interactively, the construction of the polycube map for these complicated surfaces is now as easy as that of simple surfaces. Figure 1 shows the spline surface built upon the resulting polycube map. Figure 9 show more examples of polycube maps and spline surfaces. The statistics of the examples are shown in Table 1.

Table 1: Statistics of various test examples: \( g \), genus of polycube \( P \); \( N_b \), # of boundaries; \( N_s \), # of singularities; \( N_c \), # of control points; \( \text{rms} \), root-mean-square error; \( \Delta \text{area} \), area distortion.

<table>
<thead>
<tr>
<th>Object</th>
<th>( g )</th>
<th>( N_b )</th>
<th>( N_s )</th>
<th>( \Delta \text{area} )</th>
<th>( N_c )</th>
<th>( \text{rms} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buddha (Fig. 5)</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0.6785</td>
<td>8542</td>
<td>0.06%</td>
</tr>
<tr>
<td>Three-people (Fig. 9)</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0.3529</td>
<td>9402</td>
<td>0.2%</td>
</tr>
<tr>
<td>Kitten (Fig. 9)</td>
<td>1</td>
<td>0</td>
<td>16</td>
<td>0.8477</td>
<td>4678</td>
<td>0.1%</td>
</tr>
<tr>
<td>David body (Fig. 1)</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>0.5378</td>
<td>9781</td>
<td>0.3%</td>
</tr>
<tr>
<td>Amphora (Fig. 9)</td>
<td>2</td>
<td>0</td>
<td>24</td>
<td>1.02</td>
<td>6791</td>
<td>0.08%</td>
</tr>
</tbody>
</table>

5 Conclusion

In this paper, we have advocated and developed a novel framework of user-controllable polycube maps, which can overcome the disadvantages of the conventional methods and can be easily generalized to complicated surfaces of arbitrary topology. The newly-proposed method allows users to directly select the corner points of the polycubes on the original 3D surfaces in an interactive manner, then constructs the polycube maps by using discrete Euclidean Ricci flow. The resulting polycube map usually has lower area distortion and retains small angle distortion as well, both of which are strongly desirable for spline construction in reverse engineering and finite element analysis. We design and develop algorithms for computing such polycube maps in this paper, and show that the resulting user-controllable polycube map can serve as an ideal parametric domain for constructing spline surfaces. The location of singularities can now be interactively controlled. Therefore, the subsequent hole-filling process and better data-fitting results can easily be accomplished by placing the singularities at regions where no rich geometric features exist. Through extensive experiments on various models, we demonstrate that the new, user-controllable polycube maps are well suited for spline construction of complicated geometric models of arbitrarily complicated topology towards better data-fitting results in reverse engineering, solid modeling, and design and analysis applications.

Acknowledgements

The Buddha models is provided courtesy of INRIA by the AIM@SHAPE Shape Repository. Ying He is partially supported by NTU SUG 19/06 and AcRF 69/07. This project is partially supported by NSF grants CCF-448339, DMS-0528363, DMS-0626223, IIS-0713145, and IIS-0710819.

References


(a) 3-people sculpture: a genus-zero open surface.

(b) Kitten: a genus-one surface.

(c) Amphora: a genus-two surface.

**Figure 9:** Various examples of polycube $T$-splines.