Contents lists available at ScienceDirect

Computers & Graphics

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

Technical Section

Geometry-aware domain decomposition for T-spline-based manifold modeling

Hongyu Wang [a,*], Ying He [b], Xin Li [c], Xianfeng Gu [a], Hong Qin [a]

[a] Computer Science Department, Stony Brook University, Stony Brook, NY 11794-4400, USA
[b] Nanyang Technological University, Singapore
[c] Louisiana State University, USA

A R T I C L E I N F O

Article history:
Received 13 December 2008
Received in revised form
2 March 2009
Accepted 3 March 2009

Keywords:
Manifold splines
T-Splines
Tensor-product B-splines
Shape modeling
Solid modeling
Shape computing
Object segmentation

A B S T R A C T

This paper presents a new and effective method to construct manifold T-splines of complicated topology/geometry. The fundamental idea of our novel approach is the geometry-aware object segmentation, by which an arbitrarily complicated surface model can be decomposed into a group of disjoint components that comprise branches, handles, and base patches. Such a domain decomposition simplifies objects of arbitrary topological type into a family of genus-zero/one open surfaces, each of which can be conformally parameterized into a set of rectangles. In contrast to the conventional decomposition approaches, our method can guarantee that the cutting locus are consistent on the parametric domain. As a result, the resultant T-splines of decomposed components are automatically glued and have high-order continuity everywhere except at the extraordinary points. We show that the number of extraordinary points of the domain manifold is bounded by the number of segmented components. Furthermore, the entire mesh-to-spline data conversion pipeline can be implemented with full automation, and thus, has potential in shape modeling and reverse engineering applications of complicated real-world objects.

1. Introduction

With the ever-improved modern 3D scanning technologies comes the urgent demand for more efficient, robust, and powerful data modeling techniques for routinely acquired CAD-based digital prototypes which are in forms of raw points or triangular meshes. These data have to be converted into continuous, compact representations to enable geometric design and downstream product development processes (e.g., accurate shape analysis, finite element simulation, and e-manufacturing) in CAE environments. Subdivision surfaces and spline schemes have been extensively investigated during the recent past to fulfill the aforementioned goal.

Real-world physical prototypes are frequently 2-manifolds of complex geometry and arbitrary topology. Naturally, subdivision surfaces can start with a coarser piecewise linear polygonal mesh, and the smooth surface can be calculated as the limit of a sequence of successive refinements from the coarse mesh. Despite their modeling advantages for arbitrary complicated surfaces (especially in animation and digital entertainment), subdivision surfaces have certain drawbacks. Accurate surface evaluation is usually too computationally intensive for realtime applications since most subdivision schemes do not allow closed-form analytic formulation for their basis functions. In addition, extraordinary points solely depend on the connectivity of the control mesh and need special care. On the other hand, spline surfaces have demonstrated their significance in shape modeling, finite element analysis, scientific computation, visualization, manufacturing, etc. In order to model an arbitrary surface in 3D, conventional spline schemes will segment the surface to many smaller open patches, and cover each patch by a single coordinate system, so that each patch can be modeled by a spline surface. Finally, any generic approach must glue all the spline patches together by adjusting the control points and the knots along their common boundaries in order to ensure continuity of certain degree. The entire segmenting and patching process is performed manually, and it requires user knowledge and skills, and for non-trivial topology and complicated geometry this task is laborious and error-prone.

Manifold splines proposed by Gu et al. [1] provides a technical solution for directly defining continuous surfaces over arbitrary manifold domains. In their work, they extend the existing spline schemes defined over planar domains to any manifold domain of arbitrary topology using affine structures. To further promote their work in real-world applications, in [2] they present the manifold T-splines, a natural and necessary integration of

* Corresponding author. Tel.: +01 6316328470; fax: +01 6316328334.
E-mail addresses: wanghy@cs.sunysb.edu (H. Wang), yhe@ntu.edu.sg (Y. He), xinli@ece.lsu.edu (X. Li), gu@cs.sunysb.edu (X. Gu), qin@cs.sunysb.edu (H. Qin).

0097-8493/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2009.03.011

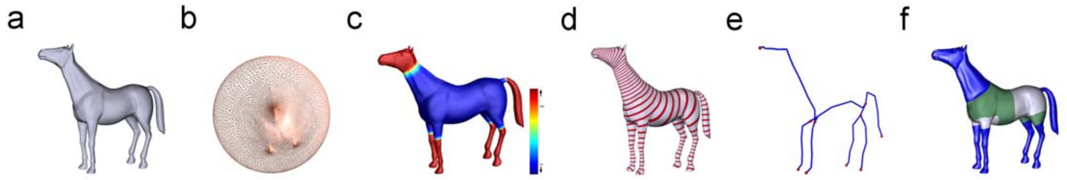

Fig. 1. Genus-zero horse model with long branches. (b) shows the spherical conformal map of the genus-zero closed surface shown in (a). (c) highlights the area distortion using color map. The Reeb graph of the given model shown in (e) is computed based on the harmonic function f defined on the given surface. (d) shows the isolines of f in red. The segmentation result based on the Reeb graph representation is shown in (f): four base patches (colored in green and gray), and six long branches (colored in blue).

T-splines and manifold splines, which naturally extends the concept and the currently available algorithms/techniques of the popular planar tensor-product NURBS and T-splines to arbitrary manifold domain of any topological type. Manifold T-splines can be directly defined over the manifold of arbitrary topology to accurately represent various shapes with complicated geometry/topology. It naturally inherits all the attractive properties from T-splines defined over a planar domain, including the powerful local refinement capabilities and the hierarchical organization for LOD control. Despite this earlier success, certain drawbacks of manifold T-splines still remain: (i) there must be singularities for any closed manifold except tori, and in practice small holes must be punched around the singularities in order to enable the easy construction of manifold splines in the finite dimension space. No efforts for hole-filling in the vicinity of extraordinary points were made in [2]; (ii) it is impossible to specify the locations of all the singularities on the domain manifold given the fact that the number of singularities is actually fixed, but their positions are somehow globally related; (iii) the proposed domain construction method is far from sufficient for surfaces with boundaries or surfaces with long branches. For surfaces with long branches (for example the horse model in Fig. 1), the existing global parameterization methods usually introduce extremely large area distortion and therefore make it even harder and numerically unstable for spline fitting process later on. The only feasible way is to introduce additional cuts in these areas to make it a surface with boundaries and then use double covering method to achieve a better parameterization result. However, this technique will at least double the time complexity and not practical for a large scale complex dataset.

Wang et al. proposed polycube T-splines [3], which unifies 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. Instead of further reducing the number of extraordinary points as Gu et al.'s work in [4], they aimed to reduce the total area distortion of the parameterization by introducing more extraordinary points (corners of the polycube) to facilitate a better spline surface fitting. In [5] they advanced their work by introducing the user's interaction into the process of polycube map construction. By allowing the user to directly select the corner points of the polycubes on the original 3D surfaces in an interactive manner, the location of singularities of the polycube map can be interactively controlled. Therefore, the subsequent hole-filling process and better data-fitting results can be easily accomplished by placing the singularities at regions where no rich geometric features exist. However, their interactive polycube map construction framework has the following limitations: (i) domain knowledge from users is required to select a feasible set of corner points which gives rise to a polycube map with high quality (that has small angle distortion and area distortion); (ii) the resulting polycube map is C^0 continuous across polycube edges that connect corners, which may introduce unpleasing results for later spline surface fitting; (iii) it is not possible to properly handle surfaces with long and thin branches (refer to Fig. 1 for an example) because users cannot easily specify corner points in long and thin branches and the computation of straight lines connecting corner points in these parts will be numerically unstable and error-prone.

In this work our objective is to further improve the existing work in data modeling, which overcomes the aforementioned drawbacks, and is much more efficient, robust, and applicable in real-world applications and industrial CAD environments. We propose a geometry-aware framework for manifold T-spline construction, which first decomposes any given surface into three categories: long branch (genus-zero patch with single boundary), handle (genus-one patch with single boundary) and base patch (genus-zero patch with at least three boundaries) by using the pants decomposition method and exploiting the skeleton representation of the surface, then locally parameterize all of these patches into regular domains using Ricci flow, build the domain manifold for each patch independently, and finally glue them together to form a complete domain manifold for later spline fitting. Note that, the T-junctions are allowed along the patch boundaries to ensure certain continuity. The proposed construction pipeline is extremely flexible: (i) it can be made fully automatic, which is therefore very useful and applicable in industrial settings; (ii) users' interaction is also allowed (refer to Section 3 for details) during the process to arrive at a result they prefer. Fig. 1 shows the horse model with long thin branches, which is very difficult to handle by using existing data modeling techniques, but can be handled elegantly by our new method. Fig. 8 shows the manifold T-spline surface for this model constructed by using our proposed algorithm.

1.1. Contribution

The specific contributions of this paper are as follows:

1. We provide a systematic way to segment any given surface into three categories (branches, handles, and base patches), and handle each category using different strategies to ensure high-quality parameterization and fitting results. Object segmentation and local parameterization enhance the system's flexibility while improving time/space performance by avoiding time-consuming and error-prone global parameterization.
2. We show that the number of extraordinary points equals $2 * n_{branch} + n_{handle} + 2 * n_{base}$ and the resultant spline surface is C^2 everywhere except C^1 at the extraordinary points.
3. The entire object segmentation always leads to a set of four-sided patches for any input surface with diverse topological types. Tensor-product B-splines or T-splines are naturally serving as basic building blocks, bridging the large gap between NURBS-centric existing CAD software in industry and manifold surface modeling algorithms.
4. The entire construction pipeline is flexible: it can be made fully automatic, which makes the proposed framework very

valuable in industrial settings. Users' interaction can also be enabled in certain parts of the pipeline to lead to a user-controllable object segmentation and local parameterization that respects both feature alignment and geometric constraints simultaneously.

5. The entire data processing pipeline enables the flexible and accurate modeling of manifold surfaces within the currently-available industrial CAD environment. The rectangular structure of each modeled piece completely avoids the trimming operation, while ensuring the "one piece" representation for manifold surfaces satisfying high-order continuity requirements.

The remainder of this paper is organized as follows. We review the related work on surface modeling, parameterization, skeleton extraction, and handle/tunnel loops computation in Section 2. In Section 3 we present the detailed algorithms for our geometry-aware domain decomposition pipeline. Experimental results are shown in Section 4. Finally, we conclude our paper in Section 5.

2. Related work

This section briefly reviews prior research on surface modeling, surface parameterization, handle/tunnel loop computation and skeleton extraction.

2.1. Surface modeling

The Catmull–Clark subdivision scheme was designed as a generalization of bi-cubic uniform B-spline surfaces to arbitrary topology [6]. It defines a smooth surface as the limit of a sequence of successive refinements from a given coarse polygonal mesh. Ref. [7] presents an exact evaluation scheme for Catmull–Clark subdivision surfaces and show that the limit surface of Catmull–Clark subdivision surfaces can also be evaluated directly, without any recursive refinement.

T-spline was presented by Sederberg et al. in [8] 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 [9]. Zheng et al. developed a technique for adaptively fitting T-splines to functional data [10]. Recently, Li et al. introduced an automatic technique to convert polygonal meshes to T-splines using periodic global parameterization [11,12]. Wang and Zheng addressed the issue of control point removal for T-spline surfaces [13].

Manifold splines have been introduced by Gu et al. [1] 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 [14], Powell–Sabin splines [15], and T-splines [2]. Most recent research results along this direction also include [3–5].

2.2. Surface parameterization

Surface parameterization has been a very active research area in the past decade [16]. Parameterization can be viewed as a mapping from a surface in 3D to a 2D canonical domain. Since isometric mappings only exist in very special cases, many approaches to surface Euclidean parameterization therefore attempt to find a mapping which is either conformal (i.e., no angular distortion) [12,17–21], or equiareal (i.e., no area distortion) [22–24]. Hyperbolic parametrization for high-genus surfaces is presented in [25]. Spherical parametrization for genus-zero surfaces are introduced in [26,27].

Patanè et al. in [28] proposed within the framework of triangulation remeshing a novel approach to the parameterization of triangle meshes representing 2-manifolds with an arbitrary genus. Their method is based on a topology-based decomposition of the shape. Each chart is then parameterized using an extension of the barycentric coordinates method. However, a non-trivial alignment of surface patches with shape features is required. In addition, the consistent vertex distribution on the parametric domain along the shared boundaries of different segments cannot be guaranteed, which is the requirement of manifold splines construction to ensure that the resulting domain manifold admits an affine atlas, therefore the continuity along the shared boundaries can be achieved naturally.

2.3. Handle/tunnel loop computation

The handle and tunnel loops can be defined as follows (see also [29] for the definition): a loop b_i on a surface M is a *handle* if it spans a disk in the bounded space $\mathbb{I}$; if one cuts M along b_i and fills the boundary with that disk, one eliminates a handle. A loop a_i on a surface M is a *tunnel* if it spans a disk in the unbounded space $\mathbb{O}$, whose its removal eliminates a tunnel. These loops characterize important topological information of the surface, and automatic detection of these loops are necessary in many applications such as topology repair of 3D models, surface parameterization , and feature recognition.

Various algorithms for computing different types of non-trivial loops on surfaces have been proposed in recent years. They either do not guarantee detecting handle and tunnel loops [30–32]; or need some graph structures built from the input model to compute the handles and tunnels such as Reeb graph [33], medial axis [34], or curve skeletons [29]. More recently, Dey et al. proposed a persistence based algorithm to compute well defined handle and tunnel loops for a 3D model in [35]. The algorithm provides a mathematical guarantee on detecting handle and tunnel loops and does not require computing any extra structures.

2.4. Skeleton extraction

Curve-skeletons are 1D structures that represent a simplified version of the geometry and topology of a 3D object. The extraction of curve-skeletons from 3D models is a fundamental problem in computer graphics and visualization, which has received a lot of attention in recent decades. We refer the readers to [38] for a detailed overview of curve-skeleton properties, applications and algorithms.

Reeb-graph-based methods have gained much attention in recent years. The Reeb graph [37] is a fundamental data structure that captures the topology of a compact manifold by following the evolution of the level-sets of a real-valued function defined on the respective manifold. It is obtained by contracting to a point the connected components of the level-sets of a function defined on a mesh. A lot of algorithms have been proposed to compute Reeb graph of an object using various real-value functions. Attene et al. [41] presented an automatic method to extract the Reeb graph of the manifold with respect to the height function. Aujay et al. [39] proposed a harmonic Reeb graph that uses the harmonic function, found by solving the Laplace equation. In [40] they use an enhanced Reeb graph of the input surface to guide the hierarchical segmentation procedure. A robust on-line algorithm for computing Reeb graphs was presented in [42].

3. Algorithm

3.1. Algorithm overview

As discussed in Section 1, the key idea of our proposed approach is the geometry-aware object segmentation, by which the given surface is first decomposed into a group of disjoint components: branches, handles and base patches. We then apply conformal parameterization and construct the domain manifold for each individual component. Finally the domain manifold for each component can be glued together to form a complete domain manifold followed by a global relaxation for later spline fitting. The proposed construction pipeline is flexible and robust: it can be made fully automatic, which makes the proposed framework very valuable in industrial settings. Users' interaction can also be enabled during certain parts of the pipeline to lead to a user-controllable object segmentation and local parameterization that respects both feature alignment and geometric constraints.

The manifold T-spline construction pipeline for a given surface M is as follows:

1. Segment the surface to branches, handles and base patches.
2. Parameterize branches, handles and base patches using discrete Ricci flow.
3. Construct the domain manifold and set the knots.
4. Fit manifold T-spline and handle extraordinary points.

Fig. 2 shows the decomposition procedure for the genus-two David model. We define different segmented components for a given surface M as follows:

- A *branch* of M is a genus-zero patch with single boundary, which is a region of M corresponding to the arc in its Reeb graph representation with two end nodes of degree one and degree three, respectively (refer to Section 3.2 for details). A *long branch* of M with respect to a given threshold ε is the branch of M with arc length (in its Reeb graph representation) longer than ε (blue components in Fig. 2(f)).
- A *handle* of M is a region of M with genus-one and single boundary (red components in Fig. 2(f)).
- A *base patch* of M is a genus-zero patch with at least three boundaries. By removing all the handles and long branches, the remaining region of M is a base patch (gray component in Fig. 2(f)); if the remaining region of M is further decomposed into a set of *pants patches* (refer to Section 3.3 for details), each pants patch is a base patch of M (green and gray components in Fig. 1(f)).
- All the branches, handles and base patches of M are disjoint, and their union equals M.

3.2. Branch segmentation

For surfaces with long branches, the existing global parameterization methods usually introduce very large area distortion and therefore make it numerically unstable for spline fitting process later on (refer to Fig. 1 for an example). To reduce the parameterization distortion, we first remove long branches from the given surface, and then parameterize them separately.

Reeb graph [37] is an ideal tool to detect the long branches from a given surface. It is a 1D structure whose nodes are critical points (maxima, minima, and saddles) of a real-value function f defined on the model surface. It encodes the topology of the model and can be constructed by contracting the connected components of the isolines (level sets or contours) of f to a point. Given its intrinsic properties Reeb graph becomes an ideal tool for us to find and remove the long branches from a given surface M: each branch of M corresponds to an arc in its Reeb graph representation with two end nodes of degree one and degree three, respectively. Given the property that each arc A in the Reeb graph represents a family of contours C_A that do not change topology, we can easily remove the long branch B (suppose its corresponding arc is A) from the given surface by cutting the surface along one contour $c \in C_A$ (the set of contours of arc A). In practice, the long branches to be removed and the corresponding cut contours used to remove the branches from the given surface can be either specified interactively by the user from its Reeb graph representation; or decided by a preset length threshold ε_{length}, and a removal ratio r, such that all the branches with corresponding arc length larger than ε_{length} will be removed by the given ratio r. In the latter way the branch removal process will be automatic.

Many algorithms for Reeb graph computation have been proposed during the recent decades. We adopt the on-line algorithm presented in [42] to compute the Reeb graph presentation of a given surface because of its robustness and scalability.

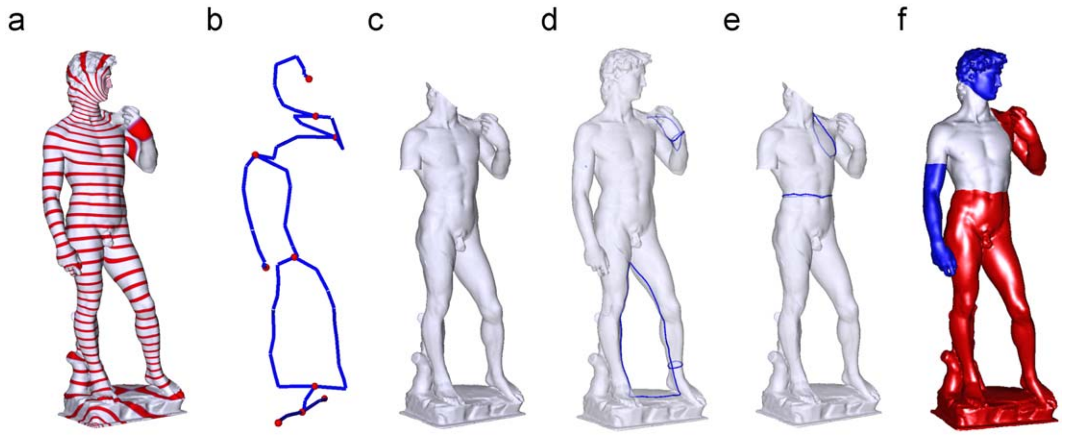

Fig. 2. Decomposition of genus-two David model. The Reeb graph of the given model shown in (b) is computed based on the harmonic function f defined on the given surface shown in (a). (c) shows the result after long branches removal. (d) highlights in blue the handle and tunnel loops computed by using the method in [35], and the loops used to remove the handles are shown in blue in (e). (f) shows the decomposition result: two branches colored in blue, one base patch colored in gray, and two handles colored in red.

Fig. 2(b) shows the Reeb graph of the genus-two David model based on the harmonic function f shown in Fig. 2(a). Fig. 2(c) shows the remaining region after removing the long branches.

3.3. Handle and base patch segmentation

In [43] a consistent pants decomposition algorithm was presented which takes as the input the handle and tunnel loops of the surface, and then segments the given surface into a set of *pants patches* (genus-zero patch with three boundaries) in a consistent manner. In our construction pipeline, we use the similar algorithm to remove the handles from the given surface with long branches removed.

The handle and tunnel loop information is required for automatic pants decomposition of the 3D surfaces [43]. There are various existing algorithms for computing critical loops on surfaces, but many of them do not guarantee detecting handle and tunnel loops. We use the persistence based algorithm proposed by Dey et al. in [35] which computes well defined handle and tunnel loops for a 3D model, and guarantees that the resulting handle and tunnel loops are topologically correct and geometrically small. The handle and tunnel loop computation is conducted on the original surface instead of the remaining patch with branches removed since the algorithm in [35] requires that the input surface is a closed one. Fig. 2(d) highlights in blue curves the handle and tunnel loops computed by using the algorithm in [35].

Once the indexed g handle and tunnel loops $(a_i, b_i, 0 \leqslant i < g)$ of the given surface M with genus g are computed, we first map them to the remaining patch M' (M with long branches removed), and then conduct a subsequent decomposition on M' to obtain a set of *handles* (genus-one patch with one boundary) and one *base patch* (genus-zero patch with at least three boundaries). The algorithm is detailed in [43], here we briefly outline the idea:

Step 1: Slice/remove all handles from M'. Repeat the following steps until all handles are removed:

(1.1) Compute a loop bounding the handle-i (topologically, such a loop $c_i = a_i^1 \circ b_i^1 \circ a_i^{-1} \circ b_i^{-1}$).
(1.2) Shrink c_i homotopically to the shortest loop w_i (Fig. 3(a), blue loops).
(1.3) Remove the handle-i from M' by slicing the loop w_i.

Step 2: (The remaining patch M'' is a topological sphere with at least three holes.) Decompose M'' into pants patches (Fig. 3(b) and (c)):

(2.1) Put all boundaries w_i of M'' into a queue Q.
(2.2) If Q has $\leqslant 3$ boundaries, end; else goto (2.3).
(2.3) Compute shortest loop w' homotopic to $w_i \circ w_j$.
(2.4) (w', w_i and w_j bound a pants patch $p_{w'}$) Remove $p_{w'}$ from M''. Remove w_i and w_j from Q. Put w' into Q. Goto (2.2).

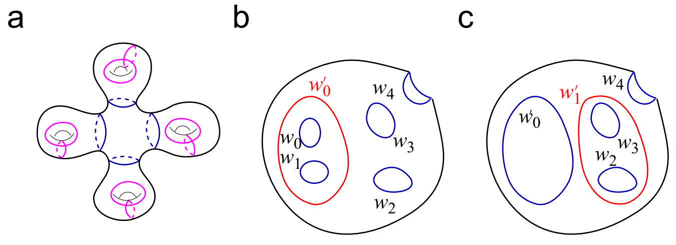

Fig. 3. Pants Decomposition. (a) Remove handle patches. (b, c) Decompose base patch: (b) Slice w'_0, get a new pants patch. Boundary number decreases by 1. (c) Set w'_0 as a new boundary, go on to compute w'_1.

After step 1, we get a set of handles and one base patch M'' which is a genus-zero patch with at least three boundaries. Step 2 is optional in our construction pipeline: we can either parameterize M'' directly, or further decompose it into a set of *pants patches* using the algorithm in step 2, then parameterize each pants patch using the method presented in Section 3.6. In Fig. 1(f), the remaining region of horse model after removing six long branches is further decomposed into four base patches (colored in green and gray).

3.4. Branch parameterization

Each branch B of the given surface is a genus-zero patch with one boundary. Fig. 4 shows the parameterization procedure for a branch from the David model shown in Fig. 2. The algorithm is as follows:

Algorithm 1. Branch parameterization.

```
In: Branch $B$ with boundary length $l_B$.
Out: A rectangular domain $D$ of $B$.
1. Find a point $p$ which is the farthest point to the boundary of $B$ (Fig. 4(a)).
2. Conformally map $B$ to a unit disk $\Omega$ [18]. If $p$ is not the center of $\Omega$, use a
   Möbius transformation to move $p$ to the center (Fig. 4(b)).
3. Find a diameter $d$ of $\Omega$, which separates the disk to two halves with minimal
   area difference. Choose two points $a_1$ and $a_2$ on $d$ with equal distance to $p$ in
   $\mathbb{R}^3$, such that the line segment $c$ connecting $a_1$ and $a_2$ passes through $p$, and
   its length $l_c$ in $\mathbb{R}^3$ satisfies $|2 * l_c - l_B| < \varepsilon$. Find another line segment $c'$
   perpendicular to $c$ which starts from $p$ and intersects with the boundary of $\Omega$.
   $c$ and $c'$ correspond to two smooth curves on $B$.
4. Slice $B$ using $c$ and $c'$,and run Ricci flow to get the rectangle domain $D$ (Fig.
   4(c)).
```

3.5. Handle parameterization

Each handle H is a genus-one surface with single boundary. The handle and tunnel loop information is computed by using Dey's algorithm [35] on the original surface and mapped to H.

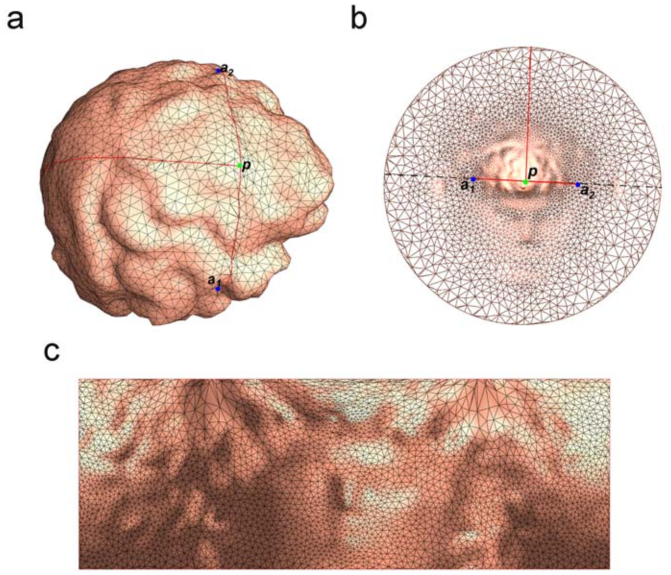

Fig. 4. Parameterizing the branch. (a) shows the pivot point p (colored in green) on the given branch B which is the farthest vertex from the boundary. In (b) the branch B is conformally mapped to a unit disk Ω with p as the disk center. Two line segments c and c' are selected on Ω which correspond to two smooth lines on B (colored in red in (a) and (b)). (c) shows the rectangle domain D obtained by running Ricci flow after slicing B by using c and c'. a_1 and a_2 marked in blue in (a) and (b) are the extraordinary points for the branch.

Fig. 5 shows an example for the procedure of handle parameterization. The algorithm is as follows:

Algorithm 2. Handle parameterization.

In: Handle H with computed handle and tunnel loops (Fig. 5(a)).
Out: A set of four rectangle domains D_i of H ($0 \leqslant i \leqslant 3$).

1. Slice H along the handle and tunnel loops , and map it to a rectangle domain D with one inner circle using Ricci flow (Fig. 5(b)): the inner circle corresponds to the original boundary of H, and the four corners of D are all images of c (common points of the handle and tunnel loop).
2. Find a point p on the tunnel loop so that p, its image p' and the center o of the inner circle are as colinear as possible. Draw straight lines from o to p and p' with two intersection points a and b with the inner circle (Fig. 5(b)) which partition the domain D into two parts D' and D'' (Fig. 5(d) shows one part).
3. For D', find an arc A passing through p and p' such that A has no other intersection points with D' except p and p'. D' can be further divided into two parts by slicing along A. Find another arc A' for D'' with the same property, and D is finally decomposed into four parts by the lines $\overline{op}$ and $\overline{op'}$, and the arcs A and A'.
4. Parameterize each of the four parts from the step 3 into a rectangle with corner points from a, b, p, c using Ricci flow.

3.6. Base patch parameterization

Each base patch is a genus-zero patch with at least three boundaries. Fig. 6(a) shows an example of the base patch from David model in Fig. 2. Given a base patch B with k boundaries, it can be parameterized into a set of $2 * k$ rectangles using the following algorithm:

Algorithm 3. Base patch parameterization.

In: Base patch B with k boundaries.
Out: $2 * k$ rectangle domains D_i of B ($0 \leqslant i \leqslant 2 * k - 1$).

1. Find two center points c_1 and c_2, and then draw k curves from each which are perpendicular to the k boundaries:
   (1.1) For each vertex v on base patch, compute its shortest distance to the k boundaries: $d_1, d_2, \ldots, d_k$. Compute c_1 as the one with the minimum range of distances to the boundaries (Fig. 6(a)).
   (1.2) Remove m-ring neighbors of c_1 from B, and map the remaining patch B' to a circle Ω with k holes (circles) inside, which correspond to the original k boundaries, and the outmost boundary of Ω corresponds to the hole introduced by removing the m-ring neighbors of c_1 (Fig. 6(b) and (c)).
   (1.3) Compute c_2 as the one with the minimum distance range to the k boundaries on Ω, and draw lines from c_2 to each center of the k circles(holes) inside Ω (Fig. 6(c)).
   (1.4) Remove n-ring neighbors of c_2 from B, and map the remaining patch to a circle Ω' with k inner circles, draw k curves on Ω' from c_1 to the center of each inner circle (Fig. 6(e)).
2. Slice B into k patches using the $2 * k$ curves computed from step 1, each of which contains c_1 and c_2, and four intersection points with the original k boundaries. Parameterize each patch into a regular hexagon, and partition it into two parts by the line connecting c_1 and c_2 (Fig. 6(f) and (g)).
3. Finally B is decomposed into $2 * k$ patches, each of which is parameterized into a rectangle (with four corners: c_1, c_2, and two of the $2 * k$ intersection points with the k boundaries) using Ricci flow.

3.7. Domain manifold construction

Since the branches, handles and base patches are parameterized individually (refer to Sections 3.4–3.6), the parameterization may not be consistent along the shared cutting boundaries, i.e., the same cutting boundary of the 3D mesh is mapped to lines of different length by the parameterization of different patches. This inconsistency in the parametric domain causes significant troubles in constructing manifold splines, since the knot vectors of adjacent patches do not meet along the boundaries. We apply a post-processing to eliminate these inconsistency.

Note that we map all patches (branches, handles and base patches) to rectangles. Let $\phi : P \to D$ denote the parameterization, where P is the 3D patch and D is the rectangle on the parametric domain. The four corners of D are v_0, v_1, v_2, and v_3. Then we solve a harmonic map $\psi : D \to D$ such that $\triangle \psi = 0$ with following boundary conditions:

(1) $\psi(v_i) = v_i$, $i = 0, 1, 2, 3$;
(2) $\psi(v) = (1 - \alpha)v_i + \alpha v_{i+1}$ for any boundary vertex $v \in (v_i, v_{i+1})$ and $\alpha = \mathit{length}(\phi^{-1}(v_i), \phi^{-1}(v))/\mathit{length}(\phi^{-1}(v_i), \phi^{-1}(v_{i+1}))$. The function $\mathit{length}(\mathbf{p}, \mathbf{q})$ measures the arc length of the boundary curves with end points $\mathbf{p}$ and $\mathbf{q}$.

Solving the above harmonic map for each individual parameterization can guarantee that the shared boundary for two adjacent patches is mapped to two straight lines (side of the rectangle) that only differ by a translation, a rotation and a scaling. In other words, given two parameterized rectangles $ABCD$ and $A'B'EF$ where AB and $A'B'$ are the sides corresponding to the same cutting boundary of the original mesh, we can find an affine transformation (a composite map of one translation, one rotation and one scaling) such that AB and $A'B'$ coincide.

To facilitate the implementation, we scale all parameterized rectangles to make sure that two adjacent patches have the same side lengths on the parametric domain. We should also point out that T-junctions are allowed along the boundaries of the patches. Thus, the resultant T-mesh is ready to serve as the domain manifold for a manifold T-spline, where the knot interval of each edge is just its length on the parametric domain.

Given the fact that each patch is parameterized individually, thus, the above T-mesh may result in angle/area distortion along the boundaries. To reduce the distortion, we use the following technique.

For each cutting boundary, we extract the k-ring neighbors (k is the user-specified parameter, $k = 2$ in our implementation) and then map it to a rectangle. Then we solve a harmonic map for the rectangle where the vertices along the rectangle sides are fixed. This harmonic map is helpful to reduce the angle distortion of the parameterization. Fig. 7(a) shows the domain manifold for David model in Fig. 2 after the global relaxation.

3.8. Surface fitting

Once the domain manifold M with conformal structure $f : M \to R^2$ is given, we proceed to solve the problem of finding a good approximation of a given polygonal mesh P with vertices $\{p_i\}_{i=1}^m$ by a manifold T-spline. We adopt the same strategy presented in [2] to minimize a linear combination of interpolation and fairness functionals, i.e.,

$$\min E = E_{dist} + \lambda E_{fair}. \tag{1}$$

The first part is

$$E_{dist} = \sum_{i=1}^{m} \|\mathbf{F}(\mathbf{u}_i) - \mathbf{p}_i\|^2,$$

where $\mathbf{u}_i \in M$ is the parameter for $\mathbf{p}_i$, $i = 1, \ldots, m$. The second part E_{fair} in (1) is a smoothing term with a fairness weight $\lambda \geqslant 0$. In our proposed framework, the parameterizations for all decomposition components are quasi-conformal which leads to a set of good initial values for the control points, so we can obtain satisfactory results using simply a small, constant λ as suggested in [44]. We choose $\lambda = 0.2$ in our experimentation. We refer readers to [2] for the detailed definitions of $\mathbf{F}(\mathbf{u}_i)$ and E_{fair}. Both parts are quadratic functions of the unknown control points, leading to a linear system. We solve Eq. (1) for unknown control points using conjugate gradient method. The value and gradient of the interpolation functional and fairness functional can be computed straightforwardly.

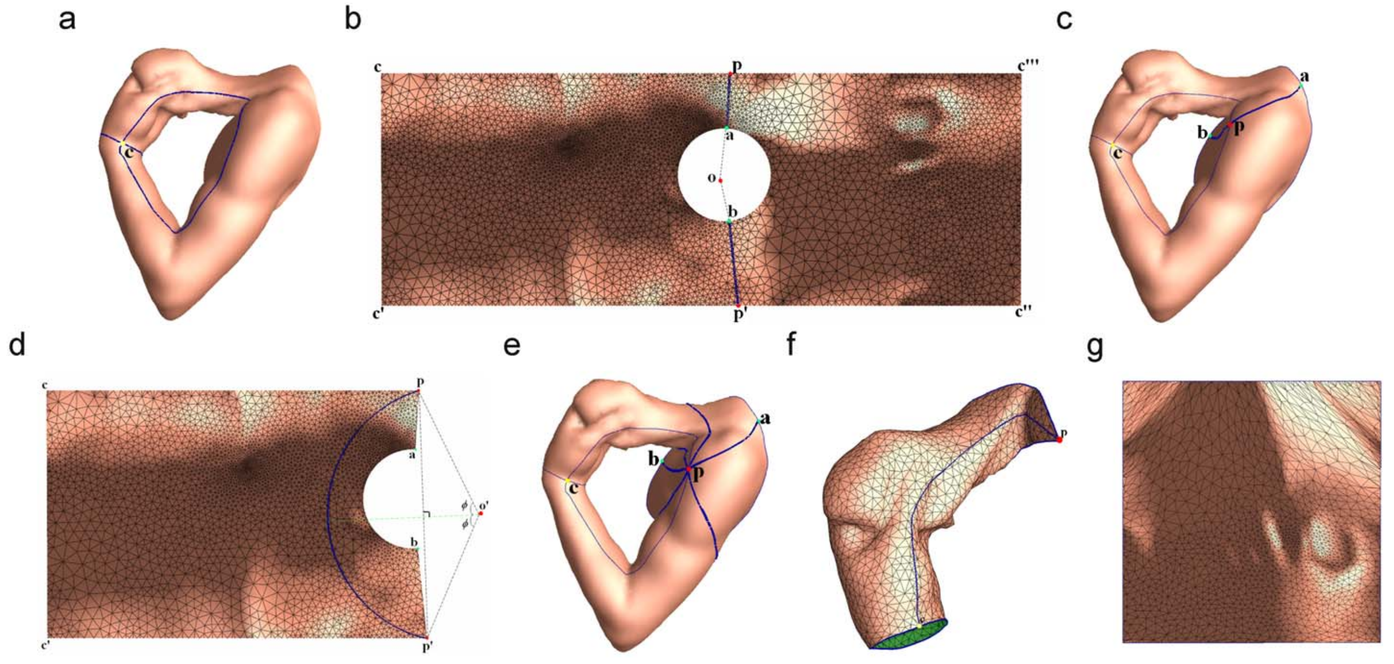

Fig. 5. Parameterizing the handle. (a) shows one handle from David model with handle and tunnel loops highlighted in blue, and the common point c of the loops shown in yellow. Point p in step 2 can be found by traversing all vertices on the tunnel loop to minimize the angle difference $|\angle pop' - \pi|$. (c) shows on the original handle patch the preimages of the cut lines (in thick blue) connecting the center of the circle and point p in the rectangular domain. The arc with the property in step 3 is not unique. One feasible way to find such an arc is to simplify the problem into finding an angle ϕ so that the resulting arc satisfies the required property (refer to (d)). In practice, ϕ can be either decided automatically by iterating all possible values to find the one with the required property and minimizing the area difference of the two parts obtained by slicing along the corresponding arc, or specified by the user in an interactive fashion to achieve a user-preferred segmentation result. (e) highlights in thick blue the lines by which the original handle is decomposed into four pieces. (f) shows one piece of the decomposition, and (g) shows its corresponding rectangular domain. p is the extraordinary point for the handle.

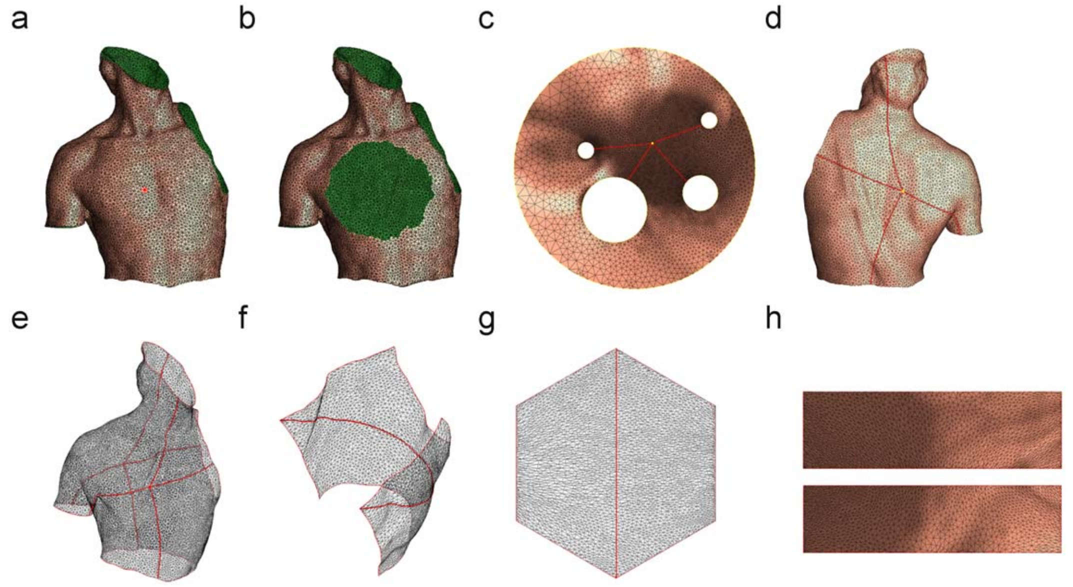

Fig. 6. Parameterizing the base patch. (a) shows the base patch (with four boundaries) of the David model with c_1 marked with sharp edges, and in (b), a new boundary b' is introduced by removing m-ring neighbors of c_1. The remaining patch is then parameterized into a circle Ω with four inner circles (c) which correspond to the original four boundaries. The outmost boundary of Ω corresponds to b'. (c) highlights in red the lines connecting c_2 and the centers of four inner circles, and (d) shows their preimages on the original base patch. (e) shows the eight curves from c_1 and c_2 and (f) shows one of the eight patches by slicing the original base patch using these curves. The patch in (f) is then parameterized into a regular hexagon (g), and further decomposed into two parts by slicing along the line connecting c_1 and c_2 on the hexagon, each of which is parameterized into a rectangle as shown in (h). c_1 and c_2 are the two extraordinary points for the base patch.

As discussed in Section 3.7, our parameterization ensures the consistency along the shared boundaries of different segmented components on the parametric domain. Furthermore, our method guarantees the transitions among local charts on the domain manifold to be affine. According to manifold spline theory [1], the construction leads to an affine atlas, and the continuity along the shared boundaries is ensured automatically. The resulting spline surface is C^2 everywhere except at the extraordinary point.

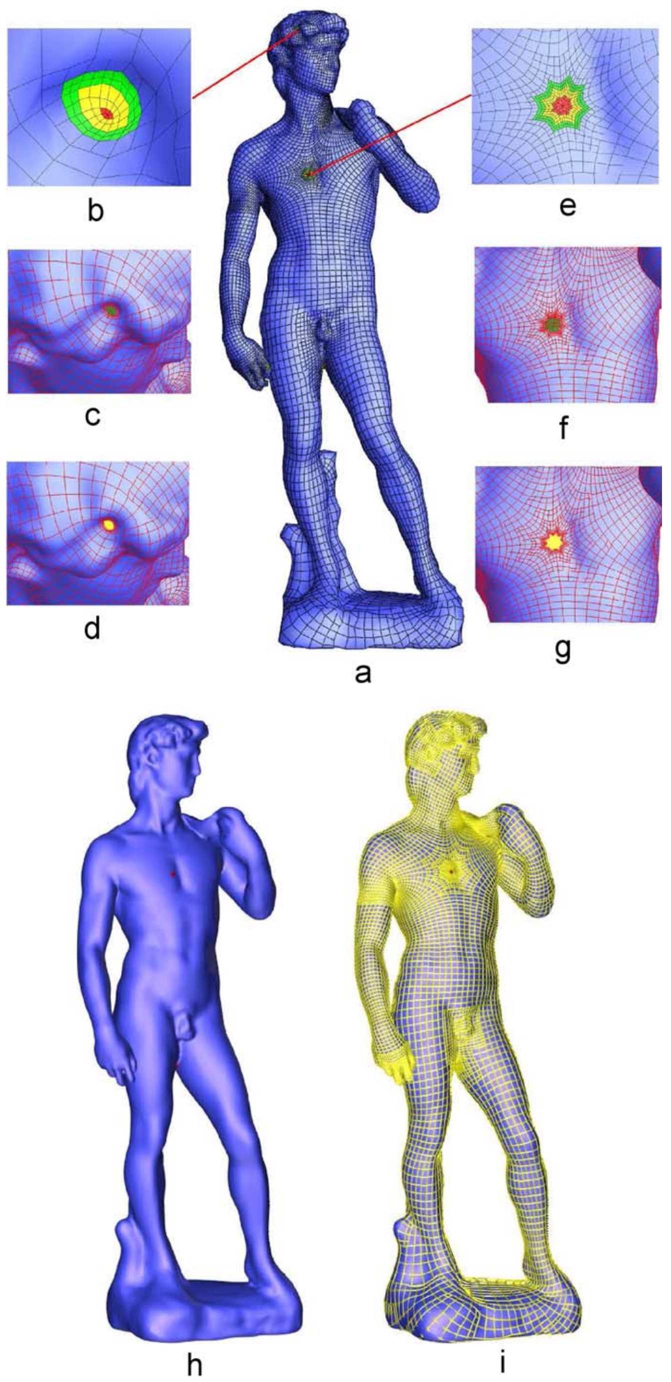

Fig. 7. Surface fitting for David model. Extraordinary points ((b) and (e)) on the domain manifold correspond to the holes on the spline surface (shown in (c) and (f)). For each hole, we construct a Catmul–Clark subdivision surface with high continuity along their shared boundaries. (d) and (g) show the results after hole-filling (hole areas are colored in yellow). (h) shows the manifold T-spline surface. The yellow curves on the spline surface in (i) highlights the T-junctions (singularities are colored in red).

Handling the extraordinary point: In [1], Gu et al. proved that manifold splines *MUST* have singularities if the domain manifold is closed and not a torus. The number of extraordinary points in our geometry-aware manifold T-spline construction pipeline equals $2 * n_{branch} + n_{handle} + 2 * n_{base}$, and they can be classified into three categories: (1) extraordinary points for handles: each handle has a single extraordinary point with valence 8 (p in Fig. 5(e)); (2) extraordinary points for branches: each branch has two extraordinary points with valence 2 (a_1 and a_2 in Fig. 4(a) and (b)); (3) extraordinary points for base patches: each base patch with k boundaries has two extraordinary points with valence $2 * k$ (c_1 and c_2 in Fig. 6). Fig. 7 shows one extraordinary point with valence 2 in (b) for the branch shown in Fig. 4, and one extraordinary point with valence 8 in (e) for the base patch shown in Fig. 6(a) with four boundaries.

Although the singularities are just points on the domain manifold, in practice, we have to remove these points and their 1-ring or 2-ring neighbors. As a result, the holes are unavoidable in the spline surface. Thus, we need to find a blending surface patch to fill the holes smoothly. In our implementation, we use a Catmul–Clark subdivision surface to fill each hole such that the surface is C^2 everywhere except C^1 at the extraordinary point. For each extraordinary point, we remove its 2-ring neighbors from the domain manifold (red quads in Fig. 7(b) and (e)), and use its 6-ring neighbors as the domain for the Catmul–Clark subdivision surface to fill the introduced hole. The T-spline surface is evaluated without using the yellow quads but taking into account their contributions (to the green quads). The continuity along the shared boundary between the T-spline surface and the Catmul–Clark subdivision surface (i.e. the shared boundary between yellow quads and green quads in Fig. 7(b) and (e)) is ensured naturally due to the same set of control points for that shared boundary on both the T-spline domain manifold and the domain of the Catmul–Clark subdivision surface. Fig. 7(h) shows the manifold T-spline surface built upon the domain manifold shown in Fig. 7(a). In Fig. 7(i) the yellow curves highlight the T-junctions on the spline surface. The singularities are colored in red in Fig. 7(h) and (i).

4. Implementation and results

Our prototype system is implemented in C + + on windows platform. We built a complete system for the Reeb graph computation, handle and tunnel loops detection, surface decomposition and parameterization, and T-spline surface fitting. We tested our algorithms on various models with complicated topologies. More examples are shown in Fig. 8. The results demonstrate both the theoretic rigor and feasibility in practice. The statistics of the examples are shown in Table 1.

5. Conclusion

In this paper, we have developed a new and effective method to construct manifold T-splines for surfaces of complicated topology/geometry. The most significant new idea of our approach is the geometry-aware object segmentation that simultaneously respects local geometric features and global topological structures. Our divide-and-conquer strategy can decompose an arbitrarily complicated surface into a group of non-overlapping components that comprise branches, handles, and base patches. This object segmentation greatly simplifies objects of arbitrary topological type into a family of genus-zero regular surfaces with four curved boundaries. Popular spline schemes such as tensor-product B-splines and T-splines can be easily employed to model segmented patches with high accuracy. Furthermore, the entire segmentation process is extremely flexible and intuitive, accommodating either full automation or interactive user control. This local-to-global surface reconstruction is made possible through a global gluing process followed by a global relaxation algorithm. We show that the number of extraordinary points of the domain manifold is bounded by the number of segmented components. Since tensor-product B-splines and NURBS are currently standards in CAD software industry, our entire mesh-to-spline data

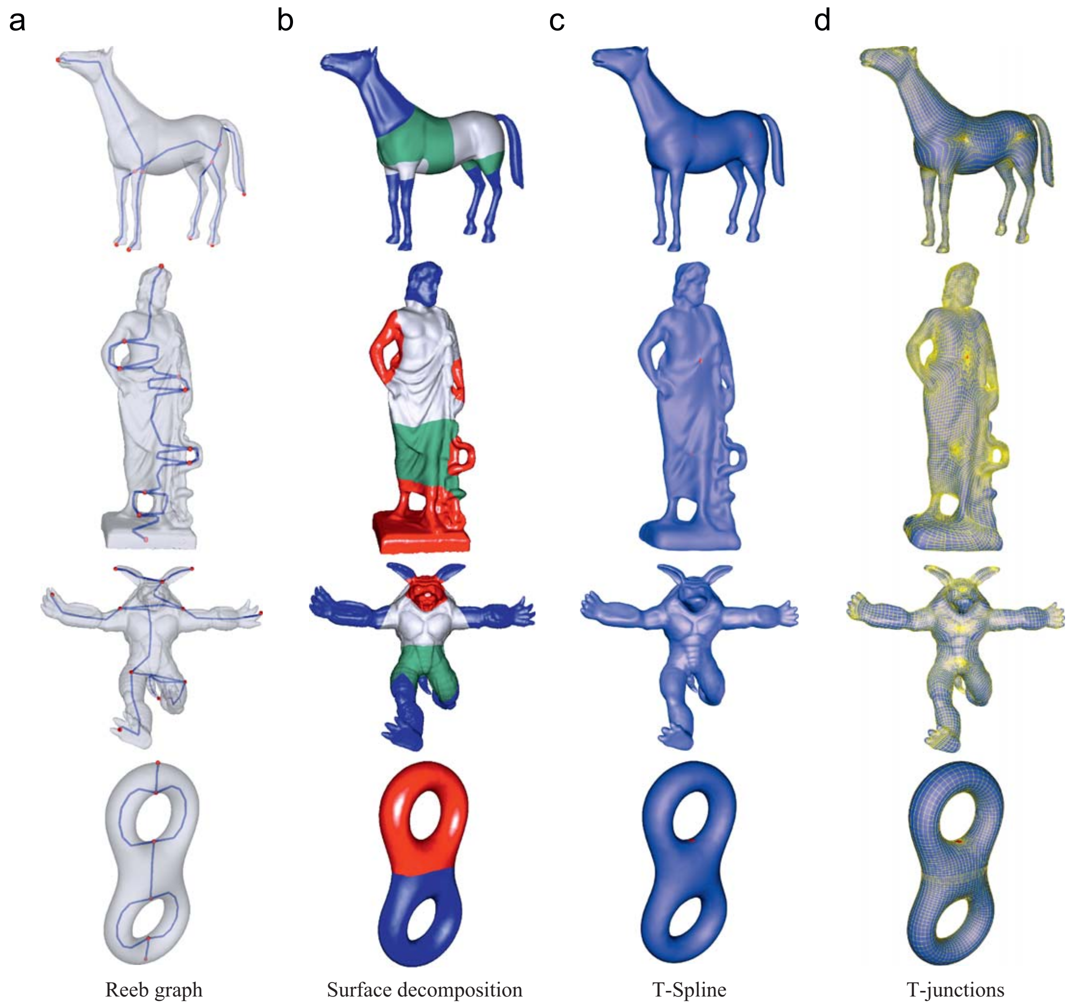

Fig. 8. Experimental results.

Table 1
Statistics of various test examples: g, genus of polycube P; N_{branch}, # of branches; N_{handle}, # of handles; N_{base}, # of base patches; N_e, # of extraordinary points; rms, root-mean-square error.

Object	g	N_{branch}	N_{handle}	N_{base}	N_e	rms (%)
David (Fig. 7)	2	2	2	1	8	0.12
Horse (Fig. 8)	0	6	0	4	20	0.06
Greek (Fig. 8)	4	1	4	2	10	0.13
Armadillo (Fig. 8)	0	7	0	3	20	0.09
Eight (Fig. 8)	2	0	2	0	2	0.02

transformation pipeline enables and expedites the manifold surface design over existing CAD software platform industry (without any trimming), and thus, has great potential in shape modeling and reverse engineering applications of complicated real-world objects.

Acknowledgments

We would like to thank Prof. Tamal Dey for providing his HanTun software to compute the geometry-aware loops and tunnels. The models are courtesy of the AIM@SHAPE Shape Repository. Hong Qin and Hongyu Wang are supported by NSF IIS-0710819 and NSF IIS-0326388. Ying He is supported by the Singapore National Research Foundation Interactive Digital Media R&D Program, under Research Grant NRF2008IDM-IDM004-006. Xianfeng Gu is supported by NSF DMS-0626223, NSF CCF-0448399 and NSFC 60628202.

References

[1] Gu X, He Y, Qin H. Manifold splines. Graphical Models 2006;68(3):237–54.
[2] He Y, Wang K, Wang H, Gu X, Qin H. Manifold T-spline. In: Proceedings of GMP '06. Lecture notes in computer science, vol. 4077; 2006. p. 409–22.
[3] Wang H, He Y, Li X, Gu X, Qin H. Polycube splines. Computer Aided Design 2008;40(6):721–33.
[4] Gu X, He Y, Jin M, Luo F, Qin H, Yau S-T. Manifold splines with single extraordinary point. Computer Aided Design 2008;40(6):676–90.
[5] Wang H, Jin M, He Y, Gu X, Qin H. User-controllable polycube map for manifold spline construction. In: Proceedings of the 2008 ACM symposium on solid and physical modeling; 2008. p. 397–404.
[6] Catmull E, Clark J. Recursively generated B-spline surfaces on arbitrary topological meshes. Computer Aided Design 1978;10(6):350–5.
[7] Stam J. Exact Evaluation of Catmull–Clark subdivision surfaces at arbitrary parameter values. In: Proceedings of SIGGRAPH'98; 1998. p. 395–404.
[8] Sederberg TW, Zheng J, Bakenov A, Nasri AH. T-splines and T-NURCCs. ACM Transactions on Graphics 2003;22(3):477–84.
[9] Sederberg TW, Cardon DL, Finnigan GT, North NS, Zheng J, Lyche T. T-spline simplification and local refinement. ACM Transactions on Graphics 2004;23(3):276–83.
[10] Zheng J, Wang Y, Seah HS. Adaptive T-spline surface fitting to z-map models. In: GRAPHITE; 2005. p. 405–11.
[11] Li W-C, Ray N, Lévy B. Automatic and interactive mesh to T-spline conversion. In: EG/ACM symposium on geometry processing; 2006.
[12] Ray N, Li WC, Lévy B, Sheffer A, Alliez P. Periodic global parameterization. ACM Transactions on Graphics 2006;25(4):1460–85.
[13] Wang Y, Zheng J. Control point removal algorithm for T-spline surfaces. In: GMP; 2006. p. 385–96.

[14] He Y, Gu X, Qin H. Automatic shape control of triangular B-splines of arbitrary topology. Journal of Computer Science and Technology 2006;21(2): 232–7.
[15] He Y, Jin M, Gu X, Qin H. A C^1 globally interpolatory spline of arbitrary topology. In: Lecture notes in computer science, vol. 3752; 2005. p. 295–306.
[16] Floater MS, Hormann K. Surface parameterization: a tutorial and survey. In: Dodgson NA, Floater MS, Sabin MA, editors. Advances in multiresolution for geometric modelling. Berlin: Springer; 2005. p. 157–86.
[17] Sheffer A, Lévy B, Mogilnitsky M, Bogomyakov A. ABF++: fast and robust angle based flattening. ACM Transactions on Graphics 2005;24(2):311–30.
[18] Gu X, Yau S-T. Global conformal surface parameterization. In: Proceedings of the eurographics/ACM SIGGRAPH symposium on geometry processing; 2003. p. 127–37.
[19] Lévy B, Petitjean S, Ray N, Maillot J. Least squares conformal maps for automatic texture atlas generation. In: SIGGRAPH '02: proceedings of the 29th annual conference on computer graphics and interactive techniques. New York, USA: ACM Press; 2002. p. 362–71.
[20] Jin M, Wang Y, Yau S-T, Gu X. Optimal global conformal surface parameterization. IEEE Visualization 2004;25:267–74.
[21] Kharevych L, Springborn B, Schröder P. Discrete conformal mappings via circle patterns. ACM Transactions on Graphics 2006;25(2):412–38.
[22] Maillot J, Yahia H, Verroust A. Interactive texture mapping. In: SIGGRAPH '93: proceedings of the 20th annual conference on computer graphics and interactive techniques. New York, USA: ACM Press; 1993. p. 27–34.
[23] Surazhsky V, Gotsman C. Explicit surface remeshing. In: SGP '03: proceedings of the 2003 eurographics/ACM SIGGRAPH symposium on geometry processing. Aire-la-Ville, Switzerland: Eurographics Association; 2003. p. 20–30.
[24] Khodakovsky A, Litke N, Schröder P. Globally smooth parameterizations with low distortion. ACM Transactions on Graphics 2003;22(3):350–7.
[25] Jin M, Luo F, Gu X. Computing surface hyperbolic structure and real projective structure. In: Symposium on solid and physical modeling; 2006. p. 105–16.
[26] Gotsman C, Gu X, Sheffer A. Fundamentals of spherical parameterization for 3d meshes. ACM Transactions on Graphics 2003;22(3):358–63.
[27] Gu X, Wang Y, Chan TF, Thompson PM, Yau S-T. Genus zero surface conformal mapping and its application to brain surface mapping. IEEE Transactions on Medical Imaging 2004;23(8):945–58.
[28] Patanè G, Spagnuolo M, Falcidieno B. Para-graph: graph-based parameterization of triangle meshes with arbitrary genus. Computer Graphics Forum 2004;23(4):783–97.
[29] Dey TK, Li K, Sun J. On computing handle and tunnel loops. In: Proceedings of the 2007 international conference on cyberworlds; 2007. p. 357–66.
[30] Erickson J, Whittlesey K. Greedy optimal homotopy and homology generators. In: Proceedings of the 16th annual ACM–SIAM symposium on discrete algorithms; 2005. p. 1038–10460.
[31] Chen C, Freedman D. Quantifying homology classes. In: Proceedings of the 25th annual symposium on the theoretical aspects of computer science; 2008. p. 169–80.
[32] Colin de Verdière É, Lazarus F. On optimal system of loops on an orientable surface. In: Proceedings of the 43rd annual IEEE symposium on foundations of computer science; 2005. p. 627–36.
[33] Shattuck DW, Leahy RM. Automated graph-based analysis and correction of cortical volume topology. IEEE Transactions on Medical Imaging 2001;20: 1167–77.
[34] Zhou Q-Y, Ju T, Hu S-M. Topology repair of solid models using skeletons. IEEE Transactions on Visualization and Computer Graphics 2007;13:675–85.
[35] Dey TK, Li K, Sun J, Cohen-Steiner D. Computing geometry-aware handle and tunnel loops in 3D models. ACM Transactions on Graphics 2008;27:1–9.
[37] Reeb G. Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique [on the singular points of a complete integral pfaff form or of a numerical function]. Comptes Rendus de l Academie des Sciences Paris 1946;222:847–9.
[38] Cornea ND, Min P, Silver D. Curve-skeleton properties, applications, and algorithms. IEEE Transactions on Visualization and Computer Graphics 2007:530–48.
[39] Aujay G, Hétroy F, Lazarus F, Depraz C. Harmonic skeleton for realistic character animation. In: Symposium on computer animation; 2006. p. 151–60.
[40] Tierny J, Vandeborre J-P, Daoudi M. Topology driven 3D mesh hierarchical segmentation. In: IEEE international conference on shape modeling and applications; 2007. p. 215–20.
[41] Attene M, Biasotti S, Spagnuolo M. Shape understanding by contour-driven retiling. The Visual Computer 2003;19(2–3):127–38.
[42] Pascucci V, Scorzelli G, Bremer P-T, Mascarenhas A. Robust on-line computation of Reeb graphs: simplicity and speed. ACM Transactions on Graphics 2007;58:1–9.
[43] Li X, Gu X, Qin H. Surface matching using consistent pants decomposition. In: Proceedings of the 2008 ACM symposium on solid and physical modeling; 2008. p. 125–36.
[44] Eck M, Hoppe H. Automatic reconstruction of B-spline surfaces of arbitrary topological type. In: SIGGRAPH '96: proceedings of the 23rd annual conference on computer graphics and interactive techniques; 1996. p. 325–34.