# Spherical DCB-Spline Surfaces with Hierarchical and Adaptive Knot Insertion

Juan Cao, Xin Li, *Member*, *IEEE*, Zhonggui Chen, and Hong Qin, *Senior Member*, *IEEE*

**Abstract**—This paper develops a novel surface fitting scheme for automatically reconstructing a genus-0 object into a continuous parametric spline surface. A key contribution for making such a fitting method both practical and accurate is our spherical generalization of the Delaunay configuration B-spline (DCB-spline), a new non-tensor-product spline. In this framework, we efficiently compute Delaunay configurations on sphere by the union of two planar Delaunay configurations. Also, we develop a hierarchical and adaptive method that progressively improves the fitting quality by new knot-insertion strategies guided by surface geometry and fitting error. Within our framework, a genus-0 model can be converted to a single spherical spline representation whose root mean square error is tightly bounded within a user-specified tolerance. The reconstructed continuous representation has many attractive properties such as global smoothness and no auxiliary knots. We conduct several experiments to demonstrate the efficacy of our new approach for reverse engineering and shape modeling.

**Index Terms**—Delaunay configurations, spherical splines, knot placement, knot insertion, non-tensor-product B-splines.

## 1 Introduction

The problem of converting dense point samples or piecewise meshes into compact and high-order continuous representations frequently arises in a large variety of applications in Computer-Aided Design (CAD), medical imaging, visualization, reverse engineering, etc. Continuous representations can facilitate tasks such as shape interrogation, segmentation, classification, and surface quality analysis/control. In principle, continuous surfaces can be represented and built using three general categories of methods: implicit surfaces, subdivision surfaces, and parametric spline surfaces.

Among these three categories, parametric spline surfaces have been favored in many applications, and such method enables many downstream procedures including free-form deformation, finite element analysis. For example, compared with subdivision surfaces, which keep global smoothness without cutting and stitching, but usually do not have analytic expressions, parametric surfaces admit efficient closed-form evaluation and compact representation. Compared with implicit surfaces whose derivative evaluations (e.g., tangency, curvature, or other higher order quantities) may need extensive discretization and numerical approximations, parametric surfaces are simpler and more efficient to design, control, evaluate, and render.

- *J. Cao is with the School of Mathematical Sciences, Xiamen University, Xiamen 361005, China. E-mail: Juancao@xmu.edu.cn.*
- *X. Li is with the Department of Electrical and Computer Engineering, and Center for Computation and Technology, Louisiana State University, Baton Rouge, LA 70803. E-mail: xinli@cct.lsu.edu.*
- *Z. Chen is with the Department of Computer Science, Xiamen University, Xiamen 361005, China. E-mail: chenzhonggui@xmu.edu.cn.*
- *H. Qin is with the Department of Computer Science, Stony Brook University, Stony Brook, NY 11794-4400. E-mail: qin@cs.sunysb.edu.*

*Manuscript received 4 Oct. 2010; revised 1 Sept. 2011; accepted 6 Sept. 2011; published online 13 Sept. 2011.*
*Recommended for acceptance by H. Pottmann.*
*For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2010-10-0245.*
*Digital Object Identifier no. 10.1109/TVCG.2011.156.*

### 1.1 Spherical Spline Surface Modeling

A large number of solid models in our daily life has genus-0 closed surfaces as their boundaries. It is, therefore, imperative to develop an effective way to model genus-0 surfaces using parametric splines with global smoothness.

Closed genus-0 spline surfaces can be constructed through many approaches. The commonly used parametric surface patches such as B-splines require domains to be simple planar regions such as rectangles. Hence, a straightforward constructive method is by partitioning the original domain into topological disks and composing individually constructed surface patches. The partitioning of the model into local charts and stitching of adjacent patches together usually require extensive user intervention and could be labor intensive. Enforcing high-order continuity along cutting boundary is also a challenging problem.

To avoid such tedious cutting and gluing, constructing a spline surface globally over one piece domain is a more desirable approach. General spline surfaces can be defined on manifolds that have nondisk topology. Many methods and function spaces (see surveys in [1, Ch. 9.7] and [2]) have been used successfully in coping with data-fitting and reconstruction problems on surfaces, such as radial basis functions [3], multiresolution methods [4], trivariate methods [5], spherical splines [6], polycube splines [7], and manifold splines [8]. Spline surfaces such as DMS-splines, T-splines, and polycube splines defined over discrete meshes usually need to punch a small number of holes on the domain before building the global affine mapping. It unavoidably leads to some discontinuities on the definition domain, and hence the constructed spline surfaces have singularities where additional geometric patching to fix such problems is necessary.

To avoid both partitioning discontinuity and singularities, the most natural parametric domain for closed genus-0 surfaces is sphere. Upon spherical domains, globally continuous parameterizations without cutting/stitching nor

singularity points can be computed. One can obtain an automatic spline reconstruction scheme and a globally $C^k$-continuous representation. This paper studies the construction of such a natural spline surface on the spherical domain.

## 1.2 Related Work

Spherical splines and their applications such as data interpolation and approximation have been studied by a number of researchers from viewpoints of CAGD and physical simulation applications [6] in recent decades. Comprehensive works on spherical splines and their interpolation and approximation have been surveyed or discussed in papers such as [6], [9], [10]. In fact, most existing splines can be generalized to spherical domain and will have similar properties with their planar counterparts. However, they may also inherit drawbacks from the planar case. In the followings, we discuss a few most-widely used splines.

### 1.2.1 Tensor-Product Splines and Radial Basis Functions

In [4], tensor-product polynomial splines and trigonometric splines were proposed for fitting functions/data on sphere, based on multiresolution methods. Buss and Fillmore defined barycentric combinations of spherical points as least-squares minimizations of weighted geodesic distance, which provide direct generalization of planar spline curves to spherical ones [9]. Fasshauer investigated adaptive approximation to scattered data given over the surface of the unit sphere by radial basis functions [3]. For spherical data interpolation/approximation, tensor products of univariate splines are not good choices, since data locations are usually not equally spaced over a regular grid. Radial basis functions are not good candidates either, since they are more suitable for rotationally symmetric data values [11].

### 1.2.2 Bernstein-Bézier Patches

Alfeld et al. [12] presented a natural way to define barycentric coordinates on spherical triangles by omitting the usual requirement of partition of unity. Based on the aforementioned work, the homogeneous *spherical Bernstein-Bézier* (SBB) [12], [13] and spherical simplex splines [14] were proposed, which are spherical analogues of Bernstein-Bézier polynomials and simplex splines, respectively. SBB polynomials are popular and have been widely studied [11], [15]. However, since the functional space spanned by Bernstein-Bézier elements highly depends on domain tessellations, the represented surface is uniquely defined subject to certain domain tessellation. The challenge to merge piecewise SBB polynomials with higher order continuity is another disadvantage of this approach.

### 1.2.3 Triangular B-Splines

Triangular B-spline (or DMS-spline) is another powerful and well-known scheme [16], [17] based on simplex splines [18]. It has been widely studied and applied to applications such as scattered functional data fitting, modeling, and visualization [19], [20], [21], [22], [23]. Because of the supreme ability of DMS-spline, its spherical analogue—scalar spherical triangular B-spline, continues to attract researchers' interest. Scalar spherical DMS-spline inherit many properties from their planar counterpart, such as the capability of representing any piecewise smooth surfaces of $C^{k-1}$ continuity by degree-$k$ splines and including SBB polynomials as a special case. It has been applied to data fitting applications [24], [25]. However, spherical triangular B-spline also inherits drawbacks from its planar counterpart: for any given set of knots, one has to explicitly add the "knot cloud" (i.e., auxiliary knots) in advance in order to form a knot sequence for all the basis function construction. The auxiliary knot placement is less intuitive and labor intensive. So far, it is still not clear how these auxiliary knots could affect the spline basis and the final surface in an intuitive and quantitative way. Additionally, surface constructed by DMS-splines may not be as visually smooth due to the "knot line" phenomenon [26]: the curvature along the images of the line between two knots in the parametric domain is larger than other regions, and a post fairing process is urgently needed [25], [27].

## 1.3 Motivation and Contribution

In order to reconstruct a useful genus-0 closed surface from data, a visually pleasant, everywhere $C^k$, and analytic surface representation is strongly desired. Recently, a new bivariate simplex spline scheme based on Delaunay configuration has been introduced into the geometric computing community by Neamtu [28], [29]. The simplex splines based on Delaunay configurations (we call them Delaunay configuration B-splines or DCB-splines for brevity) are judged to be the most convincing multivariate generalization of univariate B-splines [30], and planar DCB-splines have been used in the application of data reconstruction [31] [32]. DCB-splines have many attractive theoretic and computational properties, such as **optimal smoothness** and **polynomial reproducibility**, and free from auxiliary knots; therefore, it is ideal for the fitting purpose.

Since spherical splines have many important applications, it is important to develop the theory of DCB-splines on the spherical domain, and design useful algorithms for their computation and applications. In this paper, we further extend the concept of Delaunay configuration to sphere, formulate a spherical analogue of DCB-spline, and use it to automatically reconstruct genus-0 closed surfaces. The specific contributions of this work include

1. We construct the Delaunay configurations directly over the sphere and develop an effective computational method. A direct way to calculate spherical Delaunay configurations is to compute 3D higher order Voronoi diagram. As an effective alternative, we obtain spherical Delaunay configurations by merging together two sets of planar Delaunay configurations, while significantly reducing its computational complexity.
2. As our experiments demonstrate, if we construct the DCB-spline using degree-$k$ basis functions and there are no degenerate knots, the fitted surface is $C^{k-1}$ continuous everywhere. The visually pleasant surface is represented without any patching and stitching, and the continuity is naturally preserved without any additional constraints. The fitting process is adaptive and capable of satisfying any user-specified error tolerance.

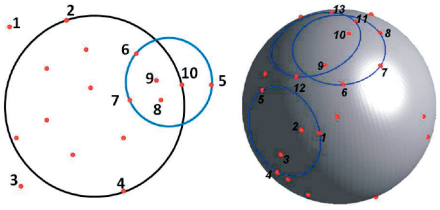


Fig. 1. Planar near-type and far-type Delaunay configurations (left), and spherical Delaunay configurations (right).

3. We propose an automatic and effective surface refining algorithm, in which the knot insertion is adaptively controlled by surface geometry and fitting error distribution in a hierarchical fashion. In the initial step, knots are distributed according to the curvedness of original data. In later iteration step, an appropriate number of new knots are added adaptively according to the distribution of fitting error.

The remainder of this paper is organized as follows: in Section 2, we extend the definition of Delaunay configuration to the spherical setting and propose an effective computation method. Section 3 describes the definition of spherical Delaunay configuration B-splines. In Section 4, we provide the overview and the technical details of our new algorithm for the automatic fitting scheme. We show our experimental results in Section 5, and conclude this paper in Section 6.

## 2 Spherical Delaunay Configurations

This section first discusses the concepts and notions of planar near/far type Delaunay configurations, spherical Delaunay configuration, and their intrinsic relationship, then proposes an efficient computation algorithm for spherical Delaunay configurations.

### 2.1 Planar and Spherical Delaunay Configurations

Let $W$ be a set of nondegenerate knots on plane $T$, i.e., no more than three knots are cocircular or colinear. If $A$ is a finite set, we denote its size by $\#A$. We define the *near-type* Delaunay configuration and the *far-type* Delaunay configuration as follows:

**Definition 1.** *A degree-$k$* near-type *Delaunay configuration of a given set of knots $W$ is a pair of sets $D_n = (D_B, D_I)$ such that 1) $D_B, D_I \subset W$ satisfy that $\#D_B = 3$, $\#D_I = k$ and 2) the circumcircle of $D_B$ contains only $D_I$ (i.e., no other knots from $W$) in its* ***interior****.*

In Definition 1, the subscript "B" and "I" represent "boundary" and "interior" knots, respectively, since knots $D_B$ and $D_I$ lie on the boundary and in the interior of a circumcircle, respectively. Similarly, the subscript "E" stands for "exterior" knots in the following definition.

**Definition 2.** *A degree-$k$* far-type *Delaunay configuration of a given set of knots $W$ is a pair $D_f = (D_B, D_E)$ such that $D_B, D_E \subset W, \#D_B = 3, \#D_E = k$ and the circumcircle of $D_B$ contains only $D_E$ (i.e., no other knots from $W$) in its* ***exterior****.*

Two examples are illustrated in Fig. 1(left): a degree-3 near-type Delaunay configuration $D_n = \{\{5, 6, 7\}, \{8, 9, 10\}\}$ is shown in the blue circle, and a degree-3 far-type Delaunay configuration $D_f = \{\{2, 4, 10\}, \{1, 3, 5\}\}$ is shown in the black circumcircle.

We denote the families of all near-type and far-type Delaunay configurations of degree $k$ associated with the set $W$ as $D_n(W)$ and $D_f(W)$, respectively. The near-type Delaunay configuration is the ordinary planar Delaunay configuration.

A sphere can be partitioned into two disjoint parts by specifying and removing a circle on the sphere. To avoid ambiguity, we denote the part with smaller area as the interior of the circle on sphere. A great circle $c$ partitions a sphere into two equal-area regions $M_1$ and $M_2$, either one can be considered as the interior in such case.

Then, in complete analogy to Definition 1, we can define set $U$ of $n$ knots on the sphere $S^2 = \{\mathbf{x} | \ \|\mathbf{x}\| = 1, \mathbf{x} \in R^3\}$ and Delaunay configuration by spherical geodesics and call this spherical Delaunay configuration.

**Definition 3.** *A degree-$k$* spherical Delaunay configuration *of a given set of knots $U$ is a pair of sets $X = (X_B, X_I)$ such that 1) $X_B, X_I \subset U$ satisfy that $\#X_B = 3, \#X_I = k$ and 2) the spherical geodesic circumcircle of $X_B$ contains only $X_I$ (i.e., no other knots from $U$) in its* ***interior****.*

We denote the family of all spherical Delaunay configurations of degree $k$ associated with the knot set $U$ on sphere as $SD(U)$. Examples of spherical Delaunay configurations are shown in Fig. 1(right): two degree-2 spherical Delaunay configurations $\{\{1, 4, 5\}, \{2, 3\}\}, \{\{11, 12, 13\}, \{9, 10\}\}$ and a degree-3 spherical Delaunay configuration $\{\{6, 7, 8\}, \{9, 10, 11\}\}$. We denote *unordered spherical Delaunay configurations* as $\widetilde{X} = X_B \bigcup X_I$. The set of all spherical Delaunay configurations $X$ corresponding to the same unordered Delaunay configuration $\widetilde{X}$ is denoted as $<X>$. The family of unordered spherical Delaunay configurations of the set $U$ on $S^2$ is denoted as $[\widetilde{X}]$.

### 2.2 Spherical Delaunay Configuration Computation

Since each Delaunay configuration corresponds to a vertex of high-order Voronoi cell, Delaunay configurations can be obtained during the construction of the high-order Voronoi diagrams [33]. Similarly, the spherical Delaunay configurations can be obtained by computing high-order Voronoi diagrams on sphere. In other words, we can obtain spherical Delaunay configurations by computing 3D Voronoi diagrams on sphere. However, this computation method is very expensive, i.e., computing 3D Voronoi diagrams needs quadratic complexity even for the first-order case [33], [34], [35]. Here, we propose a simpler and more efficient method for spherical Delaunay configuration computation, as an extension to the spherical Voronoi diagram computation [36]. In [36], using two inversions, the spherical Voronoi diagram of a given set of points on a sphere is obtained by gluing two planar Voronoi diagrams together. Similar to [36], we choose two special inversions, and prove that any spherical Delauany configuration is invariate under at least one of the two chosen inversion. Then, the Delaunay configuration set is obtained by merging two sets of planar Delaunay configurations. Note that the spherical Delaunay triangulation in [36] is only a degree-0 spherical Delaunay configuration, which is a special case discussed in our paper. With spherical Delaunay triangulation, one can only define

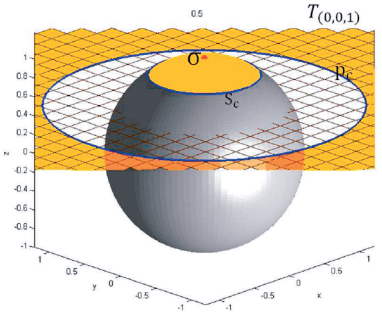



Fig. 2. Inversion function $\phi_\sigma(v)$. When $\sigma = (0,0,1)$, $\phi_\sigma$ transfers unit sphere onto plane $T_{(0,0,1)} : z = 0.5$.

degree-0 basis functions, which is not enough for the purpose of general spline construction and surface fitting. In this section, we compute general Delaunay configurations on sphere, which enables us to define basis functions of higher degrees on the spherical domain.

We first introduce an inverse transformation in 3D euclidean vector space: The *Inversion* with the inversion center at point $\sigma$ is defined by

$$\phi_\sigma(v) = \frac{v - \sigma}{\|v - \sigma\|^2} + \sigma, \tag{1}$$

where $\|\cdot\|$ is the euclidean $L_2$ norm. For both circles on sphere and plane, we now use prefix $ext$ and $int$ to represent the interior and exterior regions of circles, respectively. For example, $extS_c$ stands for the exterior of spherical circle $S_c$ on sphere; $intP_c$ stands for the interior of circle $P_c$ on plane. Let inversion center $\sigma$ be on the unit sphere $S^2$ and $S_c$ be a spherical circle on the unit sphere, then $\phi_\sigma$ has the following properties:

1. $\phi_\sigma$ maps $S^2$ to a plane, which we call the *inverse plane* $T_\sigma$ associated with the inversion center $\sigma$.
2. $\forall S_c \subset S^2, \phi_\sigma(S_c) = P_c, P_c \subset T_\sigma$.
3. If $\sigma \in intS_c$, then $\phi_\sigma(intS_c \backslash \{\sigma\}) = extP_c$.
4. If $\sigma \in extS_c$, then $\phi_\sigma(intS_c) = intP_c$.

Fig. 2 shows an example of inversion with its inversion center at point (colored in red) $\sigma = (0,0,1)$, which maps unit sphere $S^2$ onto plane $T_{(0,0,1)} : S^2 \to \{(x,y,z) \in R^3 | z = 0.5\}$. Meanwhile, it maps the spherical circle $S_c$ (decorated by the blue curve on the sphere) onto the planar circle $P_c$ (decorated by the big blue curve) on plane $T_{(0,0,1)}$. Since $S_c$ has $\sigma$ in its interior, $\phi_\sigma$ maps the interior of $S_c$ (colored as the yellow region on the sphere) to the exterior of $P_c$ (colored as the yellow region on the plane).

Given a point set $X = \{\mathbf{t}_i \, | i = 0, \ldots, m-1\} \subset S^2$, we denote their images under inversion $\phi_\sigma$ by $X^\sigma = \phi_\sigma(X)$, where $\phi_\sigma(\mathbf{t}_i) \in T_\sigma$. When there is no ambiguity, we simply use the indexing integers to represent points when their coordinates are not involved. For example, a set of points $\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_{m-1}$ are denoted as $\{0, 1, \ldots, m-1\}$. The above observations immediately lead to following properties.

**Property 1.** *A degree-k spherical Delaunay configuration $SD = \{X_B, X_I\}$ has the same combinational structure as a near-type (far-type) Delaunay configuration $D_n = \{X_B^\sigma, X_I^\sigma\}$ ($D_f = \{X_B^\sigma, X_E^\sigma\}$) if the spherical circumcircle of $X_B$ has inversion center $\sigma$ in its exterior (interior).*

We say that two pairs have the "same combinational structure" if their elements and orders are the same: $X_B = X_B^\sigma$ and $X_I = X_I^\sigma$ ($X_I = X_E^\sigma$).

**Property 2.** *Given a spherical Delaunay configuration $SD = \{X_B, X_I\}$ on $S^2$, suppose there are two inversion centers $\sigma_1$ and $\sigma_2$, locating in the interior and exterior of spherical circumcircle of $X_B$, respectively, then under the inverse transformations $\phi_{\sigma_1}$ and $\phi_{\sigma_2}$, SD will be mapped to a planar far-type Delaunay configuration $D_f = \{X_B^{\sigma_1}, X_E^{\sigma_1}\}$ on inverse plane $T_{\sigma_1}$ and a near-type Delaunay configuration $D_n = \{X_B^{\sigma_2}, X_I^{\sigma_2}\}$ on inverse plane $T_{\sigma_2}$, respectively.*

Property 1 implies that, for a given knot set $U$ on sphere $S^2$, one part of $SD(U)$ has the same combinatorial structure as the near-type Delaunay configurations $D_n(U^\sigma)$, while the rest part has the same combinatorial structure as the far-type Delaunay configurations $D_f(U^\sigma)$. Computing near-type Delaunay configurations has been widely studied, so if we can compute far-type Delaunay configurations, then the spherical Delaunay configurations can be obtained.

According to Property 2, a far-type Delaunay configuration on one inverse plane has the same combinational structure with a near-type Delaunay configuration on another inverse plane associated with an appropriate inversion center. Furthermore, for the given knot set $U$ and an inversion center $\sigma_1$ on sphere, suppose $S_{c_i}$, $i = 0, 1, .., q-1$ are all the spherical Delaunay configurations whose circumcircle includes $\sigma_1$ in their interior. Then, Property 2 also implies that if there is another inversion center $\sigma_2$ satisfying $\sigma_2 \in \bigcap_{i=0}^{q-1} extS_{c_i}$, then for each far-type Delaunay configuration on inverse plane $T_{\sigma_1}$, there is a near-type Delaunay configuration on $T_{\sigma_2}$ correspondingly. In other words, all far-type Delaunay configurations on inverse plane $T_{\sigma_1}$ can be computed from near-type Delaunay configurations on the second inverse plane $T_{\sigma_2}$.

Given a knot set $U \subset S^2$, let $\sigma_1 = (0,0,1)$ and $\sigma_2$ be its antipole, say, $(0,0,-1)$. Suppose $\sigma_1, \sigma_2 \notin U$ and $U \bigcup \{\sigma_1, \sigma_2\}$ is not degenerated, i.e., no more than three knots are spherically colinear and no more than four knots are spherically co-circular. We reduce the computation of Delaunay configurations of points on a sphere to, respectively, computing Delaunay configurations of two sets of points in $R^2$ and merging identical ones in different Delaunay configuration sets. The algorithm is as follows:

**Algorithm 1.** Computation of degree-$k$ spherical Delaunay configurations.
**Input:** knot set $U \subset S^2$.
**Output:** degree-$k$ spherical Delaunay configuration set $\Delta$.

1: $\Delta \leftarrow \emptyset$
2: map knot set $U$ by function $\phi_{\sigma_i}(v)$ to planes $T_{\sigma_i}$ and get images $U_{\sigma_i}$, $i = 1, 2$
3: compute the degree-$k$ near-type Delaunay configurations $\{D_n^{\sigma_i}\}$ of $U_{\sigma_i}$, $i = 1, 2$
4: $\Delta \leftarrow \Delta \bigcup \{D_n^{\sigma_1}\}$
5: **for** each $D_n = (X_B^{\sigma_2}, X_I^{\sigma_2}) \in \{D_n^{\sigma_2}\}$ **do**
6: **if** $\phi_{\sigma_2}(\sigma_1)$ is in the interior of circumcircle of $X_B^{\sigma_2}$ **then**
7: $\Delta \leftarrow \Delta \bigcup D_n$
8: **end if**

9: **end for**
10: **return** $\Delta$

The computational complexity of Step 2 is $O(n)$. In Step 3, since Delaunay configuration is implied in computing higher order Vonoroi diagram, we can finish its calculation in $O(n \log n)$ using the method in [33]. In Step 5, no more than $(2k+1)n$ configurations will be searched, which takes $O(n)$ to finish. In Step 6, each operation takes $O(1)$ to finish. The entire algorithm has only $O(n \log n)$ complexity, much lower than that of direct computation of high-order Vonoroi diagram in 3D space, which is usually higher than $O(n^2)$ [33].

## 3 Spherical Delaunay Configuration B-Splines (Spherical DCB-Splines)

Spherical simplex splines are spherical analogues of planar simplex splines. These spherical simplex splines are locally supported smooth functions on spherical domain. We introduce concepts and necessary notations of spherical simplex splines here and refer readers to [14], [24] for more details.

Given a knot set on the unit sphere $V = \{\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_{k+2}\} \subset S^2$ and a split set $W = \{\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}\} \subset V$, $W$ forms a spherical triangle $\Delta \mathbf{t}_{i_0}\mathbf{t}_{i_1}\mathbf{t}_{i_2}$ on $S^2$. Then, for point $\mathbf{p} \in S^2$, its spherical barycentric coordinates $(p^0, p^1, p^2)$ with respect to $\Delta \mathbf{t}_{i_0}\mathbf{t}_{i_1}\mathbf{t}_{i_2}$ is defined as

$$(p^0, p^1, p^2) = \left(\frac{\det(\mathbf{p}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2})}{\det(\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2})}, \frac{\det(\mathbf{t}_{i_0}, \mathbf{p}, \mathbf{t}_{i_2})}{\det(\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2})}, \frac{\det(\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{p})}{\det(\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2})}\right), \tag{2}$$

where we treat points as vectors, and $\det(\mathbf{a}, \mathbf{b}, \mathbf{c})$ indicates the signed volume of the tetrahedron formed by the origin and $\mathbf{a}, \mathbf{b}, \mathbf{c}$. Much work [9], [12], [37] has focused on the definition and discussion of spherical barycentric coordinates, and we choose the one (2) developed in [12] because of its simplicity as well as its many properties shared by its planar counterpart. Unlike the planar braycentric coordinates, when $\mathbf{p}$ lies on or within spherical $\triangle \mathbf{t}_{i_0}\mathbf{t}_{i_1}\mathbf{t}_{i_2}$, we have $p^0 + p^1 + p^2 \geq 1$.

A degree-$k$ spherical simplex spline associated with knot set $V$ is recursively defined as

$$M(\mathbf{p}|V) = \sum_{j=0}^{2} p^j(\mathbf{p}|W) M(\mathbf{p}|V \backslash \{\mathbf{t}_{i_j}\}), \quad \mathbf{p} \in S^2, \tag{3}$$

when $k = 0$, $V = \{\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2\}$ and degree-zero simplex spline is defined as

$$M(\mathbf{p}|\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2) = \frac{\chi[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)(\mathbf{p})}{|\det(\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)|},$$

which is the normalized characteristic function on the spherical half open convex hull of $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$.

Given a knot set $U \subset S^2$, the spherical DCB-spline associated with an unordered Delaunay configuration $\widetilde{X}$ is denoted as $B_{\widetilde{X}}$, and defined as

$$B_{\widetilde{X}}(\mathbf{p}) = \sum_{X \in <X>} \frac{1}{\det(X_B)} M(\mathbf{p}|B_{\widetilde{X}}), \quad \mathbf{p} \in S^2. \tag{4}$$

For spherical barycentric coordinates that do not yield partition of unity, i.e.,

$$I(\mathbf{p}) = \sum_{\widetilde{X} \in [\widetilde{X}]} B_{\widetilde{X}}(\mathbf{p}) \geq 1, \quad \mathbf{p} \in S^2,$$

in order to guarantee the partition of unity for spherical DCB-splines, we normalize each basis in (4) as

$$B_{\widetilde{X}}(\mathbf{p}) = \frac{B_{\widetilde{X}}(\mathbf{p})}{I(\mathbf{p})}, \quad \mathbf{p} \in S^2. \tag{5}$$

The normalized bases satisfy the convex hull property.

Suppose $[\widetilde{X}]$ has $n$ elements, we index the unordered spherical Delaunay configurations in $[\widetilde{X}]$ as $\widetilde{X}_i, i = 0, 1, \ldots, n-1$. Then, the spherical DCB-spline surface constructed by bases in (5) is defined as

$$\mathbf{F}(\mathbf{p}) = \sum_{i=0}^{n-1} B_i(\mathbf{p}) \mathbf{c}_i, \quad \mathbf{p} \in S^2, \tag{6}$$

where $B_i(\mathbf{p})$ is the basis function defined by $\widetilde{X}_i$, and $\mathbf{c}_i \in R^3$ is its corresponding control point.

## 4 Surface Fitting Using Spherical DCB-Splines

### 4.1 Problem Statement

Spline surface fitting is a fundamental problem in computer graphics, visualization, computer-aided design, and many other application fields. Our goal is to find a parametric spherical DCB-spline surface defined on the unit sphere $S^2$, approximating an unknown surface $M$ sampled by a set of points $\mathcal{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}\}$. In our initial input, these sample points are vertices of a genus-0 polygonal (triangular) mesh M, and we seek a rational parametric spherical DCB-surface (defined in (6)) to fit the input data $\mathcal{X}$, satisfying certain criteria that measure the approximation quality. Let $\mathbf{F}(\mathbf{p})$ denote the reconstructed spherical DCB-surface, and $\mathbf{u}_i$ be the parameter value on spherical domain associated with vertex $\mathbf{x}_i$, then we use the euclidean distance between $\mathbf{F}(\mathbf{u}_i)$ and $\mathbf{x}_i$, $e_i = \|\mathbf{x}_i - \mathbf{F}(\mathbf{u}_i)\|$, to measure the distance between the reconstructed surface $\mathbf{F}(S^2)$ and the original surface $M$, we call $e_i$ the fitting error of vertex $\mathbf{x}_i$.

The tolerance of root mean square fitting error (RMSE) is specified by the user. The surface fitting problem, therefore, becomes the minimization of fitting error $\{e_i\}$ so that the RMSE is less than the specified tolerance. Like most fitting processes, we minimize $e_i$ in the least-squares sense

$$\min \sum_{i=0}^{n-1} e_i = \min \sum_{i=0}^{n-1} \|\mathbf{x}_i - \mathbf{F}(\mathbf{u}_i)\|^2. \tag{7}$$

### 4.2 Algorithmic Overview

Our goal is to create a spherical DCB-spline surface $\mathbf{F}(S^2)$ that best approximates $M$. Like classical univariate B-spline curves, the following three factors typically influence the shape of spherical DCB-spline surface in the fitting procedure: 1) surface parameterization, 2) knot selection and placement, and 3) control point locations. It is possible to construct an optimal spherical DCB-spline surface from

given scattered points by solving a nonlinear optimization problem when all parameter values, knot positions, and control point positions become unknown. However, this leads to a high-order nonlinear optimization problem (with many variables) that can hardly be solved efficiently. Therefore, like most existing standard approaches, we take a more efficient divide-and-conquer fitting strategy, following the three-stage procedure: parameterization, knot placement, and least squares minimization.

Given a surface $M$, we first compute its spherical parameterization: $\varphi : S^2 \to M$. Ideally, a parameterization should have neither angular distortion nor area distortion. Such an ideal parameterization is called an *isometry*. Unfortunately, given an arbitrary surface, due to the intrinsic geometry obstacle, its isometry to a sphere $S^2$ rarely exists. We, therefore, seek a parameterization that minimizes angle and area distortion for subsequent spherical spline fitting. Conformal maps preserve the shape angle of the surface over the parametric domain; therefore, it is potentially important for high-quality fitting in the subsequent steps. However, conformal maps could introduce relatively big area distortion (for example, a long branch region on $M$ may shrink to a small parametric region on $S^2$). This is undesirable for spline fitting, because in these regions with large area distortion, we could have far fewer knots for geometric details on the long branch. Thus, a parameterization that effectively minimizes both angle and area distortions is most desirable.

Upon parameterization, we place knots on the spherical domain and construct spherical DCB-spline basis functions. For a given knot sequence on unit sphere, we can construct basis functions and associate each of them with a control point, then we optimize control point positions by solving a linear least-squares problem.

Finally, since it is usually unknown in advance how many knots and control points are required for a specified accuracy, we use a progressive procedure to adaptively adjust the number of knots and control points. In general, this knot adjustment process can proceed in either of the following two ways: 1) start with a minimal or a smaller number of knots and iteratively increase the number of knots until satisfying the error bound; or 2) start with a maximal or a larger number of knots and iteratively reduce the number of knots in order to satisfy the error bound. The second way is usually more time consuming especially when the initial knots are not well determined [38]. Therefore, we adjust knots using the first strategy. If the current fitting error $e_i$ does not satisfy the preassigned tolerance, then we add more knots and create more bases to refine the surface fitting. We also exploit a heuristic method to optimize the positions of new knots, so that in each refinement a number of new knots can be well distributed and adapted to surface geometry and fitting error.

The surface fitting process is progressive. The input of each refinement is a closed genus-0 surface $M$ sampled by $\{\mathbf{x}_i\}$; the output is the spherical DCB-spline with control points and $e_i$. We have the following fitting pipeline:

1. Specify a tolerance of the root mean square of fitting error $\varepsilon > 0$;
2. Generate the spherical parameterization of $M$;
3. Initialize knot positions;
4. Compute Delaunay configurations by Algorithm 1 and construct basis functions (5); then associate each basis function with a control point.
5. Optimize control point positions by solving the linear least-squares problem: $\min \sum_{i=0}^{n-1} e_i^2$.
6. If the current fitting RMSE is less than $\varepsilon$, then STOP, otherwise, insert new knots on parametric regions where fitting errors are big, and go back to Step 3.

### 4.3 Spherical Parameterization

The spherical parameterization has been studied as a key enabling technology in many geometric modeling and processing tasks such as meshing, morphing, and shape analysis. Spherical parameterization algorithms shall minimize angle distortions [39], [40], area distortions [41], or a trade-off between these two indicators [42], [43]. For our spline fitting purpose, minimizing angle distortion and area distortion together is most desirable. We discuss our spherical mapping distortion metric in Section 4.3.1 and briefly address how to solve it in Section 4.3.2.

#### 4.3.1 Distortion of Spherical Mapping

A spherical mapping $\varphi$ is a function from the unit sphere domain $S^2$ to a genus-0 closed surface $M \subset \mathbb{R}^3$, $\varphi : S^2 \to M$. We approximate $M$ and $S^2$ using triangular meshes with the same connectivity of $M$, denoted as $M = \{\mathcal{T}, \mathcal{X}\}$, $S^2 = \{\mathcal{T}, \mathcal{U}\}$, where $\mathcal{T} = \{T_1, \ldots, T_{N_T}\}$ is the triangular facet set, and $\mathcal{X} = \{\mathbf{x}_i\}, \mathcal{U} = \{\mathbf{u}_i\}$ are vertex sets. We consider the inverse parameterization $\psi := \varphi^{-1}$ to be linear within each triangle. Then, the mapping $\psi$ is uniquely determined by its values on mesh vertices, and for each vertex $\mathbf{v}_i$ we want to solve its image $\mathbf{u}_i$ on the spherical domain. We denote the position of vertex $\mathbf{x}_i$ on $M$ as $(x_i^1, x_i^2, x_i^3)$, and its corresponding parameter on $S^2$ as $\mathbf{u}_i = (u_i^1, u_i^2, u_i^3)$, where $\mathbf{u}_i$ has the unit $L_2$ norm: $\|\mathbf{u}_i\|^2 = 1$.

The angle distortion and area distortion of the mapping can be measured as follows:

$$E_\alpha = E_{angle} = \frac{\tau_1}{\tau_2} + \frac{\tau_2}{\tau_1}, \tag{8}$$

$$E_A = E_{area} = \tau_1 \tau_2 + \frac{1}{\tau_1 \tau_2}, \tag{9}$$

where $\tau_1$ and $\tau_2$ are the maximal and minimal singular value of the Jacobian defined on each triangle under $\varphi$. Following [44], [45], we discretize the energy on each triangle $T_{S^2}$ (and $T_M = \psi(T_{S^2})$) by

$$E_\alpha = \frac{\cot \xi |a|^2 + \cot \eta |b|^2 + \cot \zeta |c|^2}{2\mathrm{area}(T_{S^2})}, \tag{10}$$

where $\xi, \eta, \zeta$ are angles on the original triangle $T_M$ and $a, b, c$ are corresponding opposite edges of triangle $T_{S^2}$ on sphere, respectively, and

$$E_A = \frac{\mathrm{area}(T_{S^2})}{\mathrm{area}(T_M)} + \frac{\mathrm{area}(T_M)}{\mathrm{area}(T_{S^2})}. \tag{11}$$

We always normalize $M$ so that its area equals to $area(S^2) = 4\pi$. Then, in the optimal case, an isometry has both $E_\alpha = 2$ and $E_A = 2$.

We want to minimize the following energy:

$$E = \sum_{T_M \in \mathcal{T}} area(T_M) E_\alpha(T_M) \cdot (E_A(T_M))^\lambda, \tag{12}$$

where $\lambda$ is the weight balancing the angle and area distortion. Since $E_A \geq 2$, bigger $\lambda$ indicates larger emphasis on area preserving, and we will get a more uniform mapping by sacrificing conformality.

#### 4.3.2 Optimizing Spherical Mapping Distortion

To minimize the piecewise nonconvex energy in (12), we use a coarse-to-fine optimization to alleviate the local optima problem. Like [45] and [42], we also use progressive mesh [46] to get a uniformly sampled coarsest mesh, and then minimize the distortion upon progressively refined resolutions.

1. On the coarsest level, compute an initial mapping $\varphi : S^2 \rightarrow M$ using [43] (because of its efficiency).
2. Minimize the spherical parameterization distortion energy in (12).
3. Progress to a finer level, the newly split vertices are optimized within its one-ring region on the sphere by minimizing distortion energy in (12).

The optimization performed over a spherical triangle mesh can be formulated as

$$\begin{array}{ll} \min & E(\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}), \\ \text{s.t.} & \|\mathbf{u}_i\|^2 = 1, i = 0, \ldots, N-1, \end{array} \tag{13}$$

where $N$ is the number of vertices, $\mathbf{u}_i$ is a point on the unit sphere parametric domain. This is a nonlinear optimization subject to quadratic constraints. We develop an efficient optimizer for this problem on triangular meshes. It makes good use of the derivative on every spherical point, and perform curve line search for each vertex by examining the function value and gradient on that point.

Iteratively, we pick a vertex and optimize it on the sphere when fixing all other points. In other words, for a point $\mathbf{u} = \mathbf{u}_i$, we shall minimize $f(\mathbf{u}) = E(\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{i-1}, \mathbf{u}, \mathbf{u}_{i+1}, \ldots, \mathbf{u}_{n-1})$, enforcing $\|\mathbf{u}\|^2 = 1$. We evaluate the gradient of $f(\mathbf{u})$ at this point $\mathbf{u}$, denoted as $\nabla f(\mathbf{u})$, and check its magnitude on the tangent plane: let $g_\mathbf{u} = \nabla f(\mathbf{u})^T \mathbf{u}$, then $\rho(\mathbf{u}) = \|\nabla f(\mathbf{u})\|(1 - (\frac{g_\mathbf{u}}{\|\nabla f(\mathbf{u})\|})^2)$. Note that $\rho(\mathbf{u})$ is the magnitude of $\nabla f(\mathbf{u})$ projected onto $\mathbf{u}$'s tangent plane. Therefore, during iterations, each time we pick a point having largest such magnitude to optimize

$$\mathbf{u} = \text{Argmax}_{\mathbf{w} \in \{\mathbf{u}_i, i=0,\ldots,n-1\}} \rho(\mathbf{w}).$$

Now on $\mathbf{u}$, along its negative gradient direction $-\nabla f(\mathbf{u})$, we perform a curve line search on the great circle, denoted as $c_g$, obtained by the intersection of the unit sphere and the hyperplane passing through the origin, $\mathbf{u}$, and $-\nabla f(\mathbf{u})$. Furthermore, we can keep the curve line search within the union of $\mathbf{u}$'s one ring spherical triangles to avoid unnecessarily searching region that will cause flip over. Denote the union of all one-ring spherical arcs as $c_r$, we start the search from the intersection point of $c_g$ and $c_r$, along $c_g$ toward $\mathbf{u}$, and iteratively divide the step length by 2 before normalization.

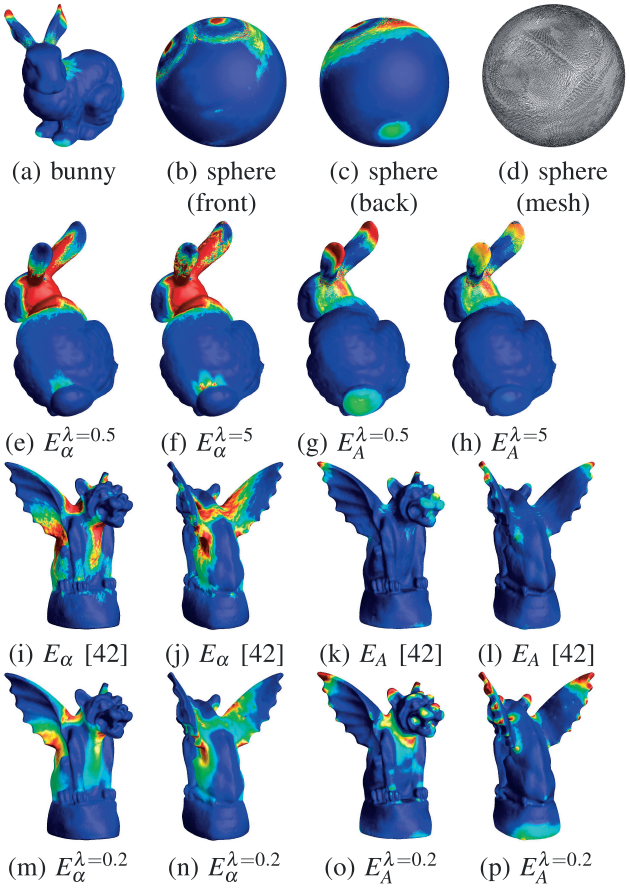



Fig. 3. Distortion of spherical parameterization under different weights. (a-h) Bunny parameterization: (b,c) The front and back of the bunny mapping ($\lambda = 0.5$) on the sphere, (d) The mesh on sphere; (e,f) Angle distortion (red: $E_\alpha \geq 4$, blue: $E_\alpha = 2$, distortion values in between are evenly distributed from blue to red) of the mappings under $\lambda = 0.5$ and $\lambda = 5$; (g,h) Area distortion (red: $E_A \geq 3$, blue: $E_A = 2$) under different $\lambda$s. (i-p) Gargoyle parameterization: (i,j) Front and back views of the color-encoded angle distortion of mapping computed using [42], (k,l) area distortion color encoded; (m-p) the corresponding models computed using our method $\lambda = 0.2$, from same views directions. All the color coding in (i-p) are consistent, (for angle distortion, red: $E_\alpha \geq 6$, blue: $E_\alpha = 2$, for area distortion, red: $E_A \geq 3$, blue: $E_A = 2$). More statistical results can be found in Table 1.

We demonstrate some mapping results computed using our algorithm. Fig. 3 shows the spherical parameterization computed on the Bunny and Gargoyle model. Figs. 3a, 3b, 3c, 3d, 3e, 3f, 3g, 3h show the mapping of bunny. Figs. 3a, 3b, 3c, 3d are the mapping computed under $\lambda = 0.5$, and the color coding visualizes the area distortion, where red indicates $E_\alpha \geq 4$, blue indicates $E_\alpha = 2$ (note that 2 is the minimal value), and distortion values in between are evenly distributed from blue to red. Figs. 3e, 3f, 3g, 3h show that under two different $\lambda$s, two parameterizations lead to different angle distortion and area distortion (in (Figs. 3g, 3h) red: $E_A \geq 3$, blue: $E_A = 2$). Figs. 3i, 3j, 3k, 3l, 3m, 3n, 3o, 3p illustrate the spherical parameterization of gargoyle model, with a side by side comparison between [42] and our method. Note that the algorithm introduced in [42] minimizes the $L^2$ stretch [47] which nicely preserves both angle and area distortions. We show that using different $\lambda$ we can flexibly control the mapping behavior. For example, in Figs. 3e, 3f, 3g, 3h, we increase $\lambda$ to improve the area-preserving1.0 property of the mapping; on the other hand,

TABLE 1
Spherical Parameterization under Different Weights

| Models | Methods | $max_{E_\alpha}$ | $avg_{E_\alpha}$ | $max_{E_A}$ | $avg_{E_A}$ |
|---|---|---|---|---|---|
| Igea | [42] | 11.95 | 2.06 | 3.93 | 2.019 |
| | $\lambda = 0.2$ | 11.29 | 2.004 | 55.7 | 2.14 |
| | $\lambda = 1$ | 11.40 | 2.06 | 3.85 | 2.019 |
| | $\lambda = 5$ | 13.58 | 2.06 | 3.16 | 2.012 |
| Bunny | [42] | 19.12 | 2.76 | 5.42 | 2.11 |
| | $\lambda = 0.2$ | 15.60 | 2.62 | 6.16 | 2.34 |
| | $\lambda = 10$ | 38.66 | 2.83 | 4.28 | 2.08 |
| Gargoyle | [42] | 16.95 | 2.89 | 8.96 | 2.14 |
| | $\lambda = 0.2$ | 16.55 | 2.72 | 37.67 | 2.36 |

*Here, $max$ and $avg$ indicate the maximal energy on triangle face and the average energy (weighted by the face area on $M$); $E_\alpha$ and $E_A$ are angle and area distortions, respectively.*

in Figs. 3m, 3n, 3o, 3p, we set $\lambda = 0.2$ to improve the mapping conformality.

Table 1 shows more statistical results of the spherical mapping. Under different weights, parameterizations have different angle and area distortions. Large $\lambda$ emphasizes the area preserving (e.g., see the rows where $\lambda = 5, 10$) while small $\lambda$ better keeps conformality (e.g., see the rows where $\lambda = 0.2$). All models in this table are data provided by [42], where the Bunny model has $69.6k$ triangles, Igea has $100k$ triangles, and Gargoyle has $200k$ triangles.

Our optimization is very efficient: when $\lambda = 1$, $10k$ iterations usually take only 0.5 second (bigger $\lambda$ makes the derivative computation slightly slower; e.g., when $\lambda = 10$, $10k$ iterations takes about 0.7 second). Note that in each iteration we move one vertex along its negative gradient to a locally optimal position within its one-ring neighborhood.

### 4.4 Adaptive Knot Insertion

Knot sequence is important in recovering the underlying smooth curve and surface from discretely sampled data points for any spline-centric representation. In particular, discrete data points acquired from a physical object have considerable geometric complexity, and the reconstructed shape may have significant difference from the actual physical surface if the knots are not reasonably distributed according to the geometric shape variation. Despite the fact that the quality of the reconstructed surface depends heavily on different strategies for knot placement, there is no effective and intuitive way to place knots at their best possible positions according to the minimization of RMSE. In this section, we introduce a heuristic method to optimize knot locations and adjust the number of knots adaptively in a hierarchical fashion: we start with a small number of knots, and then add more knots adaptively in each subsequent fitting iteration so that the reconstructed surface approximates scattered data samples progressively.

In Section 4.4.1, we first introduce a geometric measurement and then define an energy function. Through minimizing such energy function, we obtain the knots in the initial fitting step according to the surface geometry. After the initial fitting step, fitting errors will better guide the placement of new knots. In Section 4.4.2, a similar energy function is defined, and by minimizing this new energy function, we can insert knots in a greedy way: in each fitting step, more knots will be added to the regions with larger fitting errors. In Section 4.4.3, the adaptive knot placement algorithm is described and parameters used in the energy functions are explained.

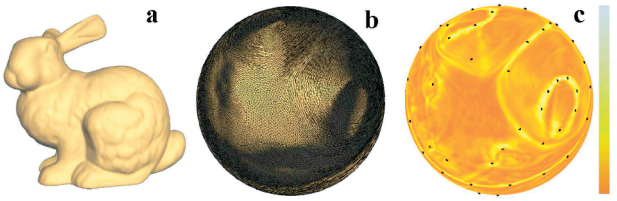



Fig. 4. The Bunny model (a) with 35k vertices, its parameterization (b), and initial knot placement. (c) Color-coded curvedness and 100 initial knots placed following curvedness distribution.

#### 4.4.1 Placement for Initial Knots

The number of knots in some regions of the parametric domain is closely related to the number of control points over the corresponding surface regions. In principle, more control points are necessary in order to model sharp features, ridges, valleys, or prongs [48]. That means, more knots shall be placed in the parametric domain corresponding to feature regions. Hence, we first introduce a measurement of surface geometry to identify regions with features.

**Surface geometry measurement.** Curvature is the most important intrinsic quantity to differentially characterize the local shape of curves and surfaces. We use the sum of absolute principal curvatures $|k_1| + |k_2|$ as the measurement of how much a surface bends (a.k.a. the **"curvedness"**) at any point, and we denote it as $\kappa$. The calculation of the principal curvature on discrete representation such as triangular meshes is not trivial. One popular approach is to first estimate curvature tensor, from which principal curvatures can be extracted [49], [50], [51]. In this paper, we compute the curvature tensor on each mesh vertex $\mathbf{x}_i$ using the method proposed in [49]. Then, the principal curvatures and the local "curvedness" can be obtained. We use $\kappa_i$ to denote the curvedness of vertex $\mathbf{x}_i$. In Fig. 4c, the normalized curvedness of each vertex is color-coded and visualized on the parametric domain. As shown in the color bar, the more curved regions are in white and less curved regions are in orange.

Suppose we have $K$ knots to be placed over the spherical domain in this initial step and denote them as $T^0 = \{\mathbf{t}_k | k = 0, 1, \ldots K-1\}$. Since curvedness is a measurement of the geometry of input mesh, we hope that more knots are placed in regions with larger curvedness. We define a curvedness function $\kappa(\mathbf{t}), \mathbf{t} \in S^2$ on the parametric domain as follows: $\kappa(\mathbf{t}) = \kappa_i$ if $\mathbf{t}$ is the parametric point corresponding to vertex $\mathbf{x}_i$; otherwise, $\kappa(\mathbf{t})$ in a triangle formed by three parametric points is defined by linear interpolation of its values at the vertices of the triangle. Assume each knot $\mathbf{t}_k$ covers a subregion $\Omega_k \subset S^2$, we compute the accumulated curvedness in $\Omega_k$'s corresponding surface patch $X_k$ by

$$\varepsilon_k = \int_{\mathbf{t} \in \Omega_k} \kappa^\alpha(\mathbf{t}) \mathrm{d}\sigma, \tag{14}$$

where $\sigma$ is the area element on sphere and $\alpha > 0$ is a control parameter. We shall decompose the parametric domain into subregions $\{\Omega_k | k = 0, 1, \ldots K-1\}$ with similar accumulated

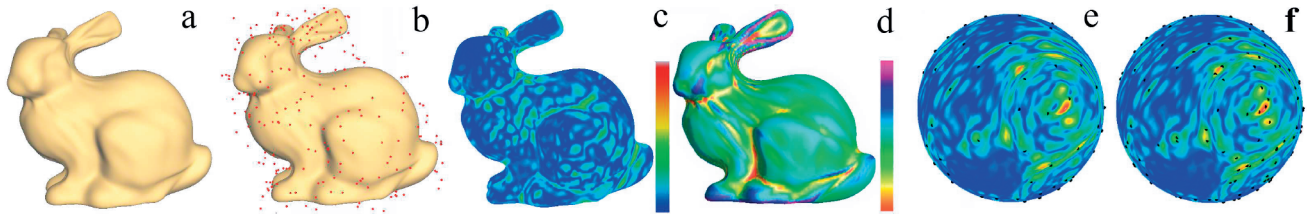



Fig. 5. Reconstructed Bunny after the first fitting step. (a) Reconstructed surface; (b) Reconstructed surface with 680 control points; (c) Fitting error (maximum error 2.63 percent and root mean square error 0.376 percent); (d) Mean curvature distribution; (e) Initial knots and color-coded fitting error map in the parametric domain; (f) Adaptive knot insertion in the second fitting step based on the fitting error.

curvedness $\varepsilon_k$ and naturally choose its mass center as a knot. Notice that, a large $\alpha$ will lead to a finer decomposition of the region with large curvedness.

However, there are a number of ways to partition the parametric domain, such that each subpatch has evenly accumulated curvedness $\varepsilon_k$, evaluated by the discrete version of (14). It is desirable if each subregion can be as round as possible, such that a knot $\mathbf{t}_k$ can be a good representative of this subregion $\Omega_k$. Therefore, we measure the distances between $\mathbf{t}_k$ and other points in $\Omega_k$, and refine (14) to

$$\varepsilon_k^* = \int_{\mathbf{t}\in\Omega_k} \kappa^\alpha(\mathbf{t})\|\mathbf{t}-\mathbf{t}_k\|^2 \mathrm{d}\sigma, \tag{15}$$

from which an energy function is defined as

$$\Phi_1(\mathbf{t}_k, \Omega_k, k=0,1,..,K-1) = \sum_{k=1}^{K} \varepsilon_k^*. \tag{16}$$

The knot locations $\{\mathbf{t}_k|k=0,1,\ldots K-1\}$, as well as the decomposition $\{\Omega_k|k=0,1,\ldots K-1\}$, can be obtained by minimizing the energy function formulated in (16). Later, we will show the method on how to minimize the energy function of (16).

Fig. 4c shows that 100 initial knots are placed by minimizing the energy function of (16), where $\alpha$ is set to 4. We can see that knots tend to concentrate in the regions with large curvedness. Given such knots $\{\mathbf{t}_k|k=0,1,\ldots K-1\}$, the spherical Delaunay configurations can be obtained by Algorithm 1. Then, the basis functions can be constructed immediately, followed by the computation of associated control points to be optimized via solving the linear least squares problem defined in (7).

Figs. 5a, 5b illustrate the spline surface and control points constructed from the initial knots, respectively. Meanwhile, the fitting error at each vertex of the mesh surface can be evaluated. We denote the maximum and minimum values of $e_i, i=0,\ldots,n-1$, as $e_{max}$ and $e_{min}$, respectively, and normalize the fitting error at vertex $\mathbf{x}_i$ by $e_i = \frac{e_i - e_{min}}{e_{max}-e_{min}}$. Fig. 5c visualizes $e_i$ of the reconstructed surface after the initial fitting. The blue and white colors indicate the minimum and maximum of $e_i$, respectively. The color-coded fitting errors, as well as the initial knots, are also shown in the parametric domain in Fig. 5e.

#### 4.4.2 Knot Placement in the Adaptive Fitting Process

After the initial fitting process, the fitting errors will better guide the placement of new knots. Obviously, on the parametric domain, regions having larger fitting errors will need more knots during the refinement process. Without loss of generality, let us assume that $K$ knots $T^j = \{\mathbf{t}_k|k=0,\ldots K-1\}$ are going to be placed over the domain in the $j$th adaptive fitting iteration (here, $j \geq 1$, and if $j=0$, it is the initial iteration/process). A fitting error function $e(\mathbf{t})$ in the $j$th iteration step can be defined in the same fashion as $\kappa(\mathbf{t})$: if $\mathbf{t}$ is the parameter point corresponding to vertex $\mathbf{x}_i$, $e(\mathbf{t}) = e_i$, where fitting error $e_i$ is obtained from the $(j-1)$th fitting iteration; otherwise, $e(\mathbf{t})$ in a triangle formed by three parametric points is defined by linear interpolation of its values at the vertices of the triangle. Similar to the placement for initial knots, we replace the curvedness $\kappa(\mathbf{t})$ in the energy function of (16) with the fitting error $e(\mathbf{t})$ and minimize function

$$\Phi_2(\mathbf{t}_k, \Omega_k, k=0,1,..,K-1) = \sum_{k=0}^{K-1} \int_{\mathbf{t}\in\Omega_k} e^\alpha(\mathbf{t})\|\mathbf{t}-\mathbf{t}_k\|^2 d\sigma. \tag{17}$$

By minimizing energy in (17), we can insert new knots adaptively subject to the fitting error distribution.

In Fig. 5f, 100 new knots are inserted after the initial fitting step by minimizing energy function of (17). It can be seen that the newly added knots are concentrated at the regions with large fitting errors. More knots are inserted progressively until the fitting RMSE satisfies the given threshold.

#### 4.4.3 Energy Minimization and Parameter Selection

It may be noted that, the energy functions in (16) and (17) also occur during calculation of centroidal Voronoi tessellations (CVT) constrained on sphere, which was introduced in [52]. Both energy functions in (16) and (17) are the so-called CVT energy functions

$$\Phi(\mathbf{t}_k, \Omega_k, k=0,1,..,K-1) = \sum_{k=0}^{K-1} \int_{\mathbf{t}\in\Omega_k} \rho(\mathbf{t})\|\mathbf{t}-\mathbf{t}_k\|^2 d\sigma, \tag{18}$$

with density function $\rho(\mathbf{t})$ being $\kappa^\alpha(\mathbf{t})$ and $e^\alpha(\mathbf{t})$, respectively. Hence, (16) and (17) can be minimized in the same way as (18). A widely used method for minimizing (18) is an iterative method proposed by Lloyd [53]. By incorporating a quasi-Newton method to compute centroidal Voronoi tessellations proposed in [54], Yan et al. [55] introduced a more efficient algorithm to minimize the energy function of (18). In this paper, we adopt the optimization scheme in [55] to compute the tessellation and the projections of centroids

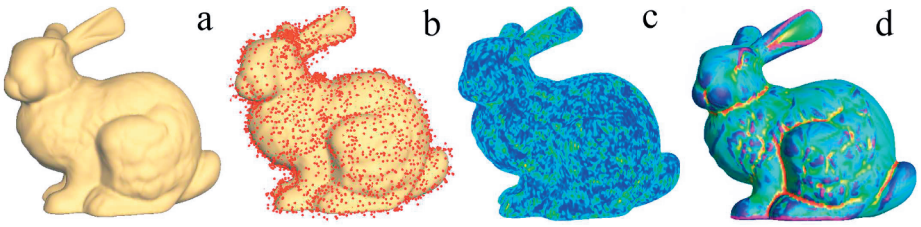



Fig. 6. Reconstructed Bunny after seven fitting steps. (a) Reconstructed quartic surface; (b) Reconstructed surface with 5,315 control points; (c) Fitting error (maximum error 0.47 percent and root mean square error 0.067 percent); (d) Mean curvature distribution.

to the sphere. For more details about the theory of CVT and its computational method, please refer to [52], [54], [55].

It has been shown that a minimizer of (18) is a special tessellation, where $\Omega_k$ is the Voronoi region, and each knot $\mathbf{t}_k$ is the projection of the centroid of $\Omega_k$ to the sphere. According to Gersho's conjecture [56], upon convergence, the energy defined in (18) evenly distributes in each patch. The knots are equally distant from each other as much as possible, and the knot distribution faithfully respects certain density functions, i.e., curvedness function or fitting error functions in this paper. Hence, knots obtained by minimizing (16) and (17) are very suitable for the application of surface fitting.

As discussed above, it is almost impossible to determine the right number of knots in either initial step or later adaptive fitting steps such that the reconstructed surface can reach the preassigned error tolerance. Fewer knots inserted in each step means more iteration steps and slower fitting processes, while more newly inserted knots in each step means fewer steps but can lead to a larger number of total knots at the end of surface fitting. Therefore, we empirically place 100 knots during the initialization stage and add 100 more knots during each refinement. As mentioned, larger $\alpha$ will force knots to be more concentrated on the region with large density. In all of our experiments, we use parameter $\alpha = 4$ in (16) and (17), which usually affords the fitting process to reach the tolerance with a good balance between the number of refinement steps and the number of knots. The algorithm for adaptive knot insertion is illustrated in Algorithm 2.

**Algorithm 2.** Adaptive knot placement algorithm for degree-$k$ spherical DCB-spline surface fitting.
**Input:** mesh $M = \{\mathbf{x}_i\}$ with parameterization mesh $P = \{\mathbf{u}_i\}$, threshold of root mean square fitting error $\varepsilon$.
**Output:** degree-$k$ spherical DCB-spline surface.

$j \leftarrow 0$ {fitting iteration number}
density function $\rho(\mathbf{t}) \leftarrow 0$
$K \leftarrow 100$ {the number of knots added in each step}
$T \leftarrow \emptyset$ {knot set for surface reconstruction}
**while** $RMSE > \varepsilon$ **do**
  **if** $j = 0$ **then**
    calculate the curvedness function $\kappa(\mathbf{t})$
    $\rho(\mathbf{t}) \leftarrow \kappa(\mathbf{t})$
  **else**
    calculate the fitting error function $e(\mathbf{t})$
    $\rho(\mathbf{t}) \leftarrow e(\mathbf{t})$
  **end if**
  obtain knot set $T^j$ of $K$ new knots by minimizing Equation (18) with density $\rho(\mathbf{t})$
  $T \leftarrow T \bigcup T^j$
  fit mesh $M$ based on knot set $T$ and update fitting error $e_i$ of each vertex $\mathbf{x}_i$
  $j \leftarrow j + 1$
**end while**
**return** spline surface

Note that, in the later adaptive fitting steps, already existed knots are all fixed, and the new knots are placed in regions with large fitting errors. Namely, in the $j$th fitting iteration, we construct a spline surface on the knot set $\bigcup_{l=0}^{j} T^l$ (see Step 14 of Algorithm 2). This may not be the best strategy for the placement of the entire set of knots. Nonetheless, 1) it usually leads to the monotonic decrease of the fitting error, which is crucial to the convergence of our refinement strategy; and 2) for each new insertion, we only need to minimize (17) subject to the newly added knots locally, and it is much more efficient than running function minimization on (17) again with respect to all the currently available (existing plus new) knots for each iteration.

## 5 Experimental Results

This section presents the experimental results of our surface fitting framework. We perform all our fitting experiments on a laptop PC with a 2.2 GHz Intel Duo-Core processor and 2 GB memory. We apply our surface fitting algorithm to several discrete models. All these data sets are uniformly scaled to fit within a unit cube in order to normalize fitting errors across different models. A perturbation method is used to avoid knot degeneracy.

In Fig. 6, a quartic spline surface, its control points, the normalized fitting error map, and the mean curvature of Bunny obtained after seven rounds of refinement are shown in Figs. 6a, 6b, 6c, and 6d, respectively. Figs. 7a, 7c, 7e illustrate more quartic spline surfaces reconstructed from models of Brain, Gargoyle, and Pierrot. Their mean curvature distributions are shown in Figs. 7b, 7d, and 7f, respectively. The convergence speed is plotted in Fig. 11a.

Fig. 8 illustrates an example of Fandisk model reconstructed by cubic splines. We can see in Fig. 8a, computed knots automatically locate near the region that corresponds to the model's sharp feature in the initial step. The reconstructed surface, however, does not show sharp edges/corners at the initial stage Fig. 8b. During subsequent fitting iterations, new knots are inserted into the region with large fitting error progressively, and the surface rounding effect in Figs. 8e, 8f, 8g, has almost been eliminated.


Fig. 7. Examples of degree-4 spherical DCB-splines. From left to right: the fitted spline surfaces followed with mean curvature distribution: Brain, Gargoyle, and Pierrot. Their maximum fitting error/root mean square error are: Brain ($0.493\%/0.090\%$), Gargoyle ($0.636\%/0.065\%$), and Pierrot ($0.574\%/0.058\%$).

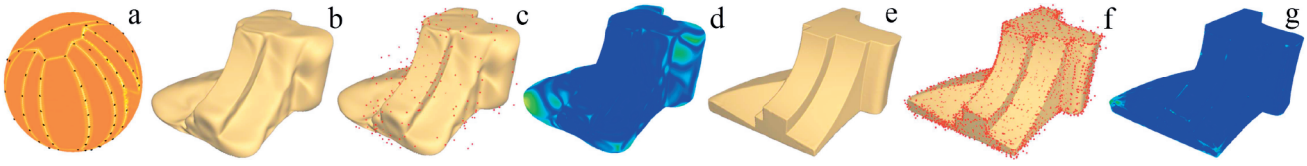



Fig. 8. Sharp feature modeling of Fandisk model using cubic splines. (a) Knot distribution according to the curvedness in the initial fitting step; (b) Reconstructed surface in the initial fitting iteration; (c) Control points (the number is 582) of surface in (b); (d) Color-coded fitting error map over surface in (b), with maximum error 9.56 percent and root mean square error 1.163 percent; (e) Reconstructed surface obtained in the eighth iteration; (f) Control points (the number is 4,994) of surface in (e); (g) Color-coded fitting error map on surface in (e), with maximum error 0.866 percent and root mean square error 0.028 percent.

If knots are placed in general positions, i.e., locally no more than three knots lie on the same great circle, then the reconstructed spline surface is globally $C^{k-1}$ continuous. On the other hand, if the knots are cocircular, then the reconstructed surface can have lower degrees of continuity in the corresponding region. Certainly, modeling sharp features is possible if we intentionally place multiple knots or cocircular knots along feature lines on the parametric domain. Using a similar strategy to [22], we can detect sharp features and apply additional constraints to enforce relevant knots to be cocircular. In our framework, we can also integrate a feature extraction preprocessing step and apply knot positional constraints in the adaptive knot placement stage to enforce some knots to be cocircular.

On the other hand, in this Fandisk experiment, benefited from our CVT method which uses curvedness and fitting errors as density functions, knots tend to be placed automatically on the parametric region with large density. The Fandisk model has clear feature lines on the parametric domain, so even though we do not intentionally place relevant knots cocircularly, we see that knots are automatically placed close to sharp curves on the parametric domain. As a result, the reconstructed surface has recovered sharp features elegantly.

Table 2 summarizes the statistics of our surface fitting procedure on the aforementioned models, where $N_v$ denotes the vertex number of the discretized models, $Deg$ denotes the degree of spherical DCB-splines used for surface reconstruction, and $N_c$ is the number of control points. The maximum fitting error is denoted as $m.e.$ while the root-mean-square error is denoted as $rms$. For each of these models, 100 new knots are inserted during each refinement step to improve the surface quality, and the entire process of knot placement and optimization takes 12-37 seconds throughout the surface fitting process. Note that, the subsequent process of basis function updating and control point optimization is the most time-consuming step, which can usually be finished in less than 1 minute for all the test models. For example, the smallest Bunny model (35K vertices) takes seven iterations to reach root-mean-square error of 0.067 percent, and in each iteration process, the basis function updating takes from 11.556 to 14.308 seconds and the control point optimization process takes from 4.354 to 7.296 seconds. The largest Dog model (180K vertices) takes nine iterations to reach root-mean-square error of 0.021 percent, in each iteration process, the basis function updating takes from 22.352 to 26.093 seconds and the control point optimization process takes from 14.266 to 17.689 seconds. There exist many effective methods for solving linear least squares problems, and in this paper, we use the Singular Value Decomposition (SVD) method because of its stability.

Our proposed framework is also suitable for reconstructing surfaces with $C^{k-1}$ continuity. Compared with lower degree spline surfaces, higher degree spline surfaces 1) inherently have higher order continuity, 2) usually take less iteration steps for better fitting, but 3) require more computation time to satisfy the same threshold requirement.

TABLE 2
Statistics of Surface Fitting Experiments

| Model | $N_v$ | $Deg$ | $N_{it}$ | $N_c$ | $m.e.(\%)$ | $rms(\%)$ |
|---|---|---|---|---|---|---|
| Bunny | 35K | 4 | 7 | 5315 | 0.470 | 0.067 |
| Brain | 100K | 4 | 11 | 8267 | 0.493 | 0.090 |
| Gargoyle | 100K | 4 | 8 | 6040 | 0.636 | 0.065 |
| Pierrot | 93K | 4 | 4 | 3001 | 0.574 | 0.058 |
| Igea | 50K | 3 | 8 | 5556 | 0.378 | 0.042 |
| Igea | 50K | 4 | 7 | 5304 | 0.461 | 0.042 |
| Igea | 50K | 5 | 6 | 5286 | 0.559 | 0.040 |
| Fandisk | 56K | 3 | 8 | 4994 | 0.866 | 0.028 |
| Dog | 180K | 5 | 9 | 7790 | 0.344 | 0.021 |

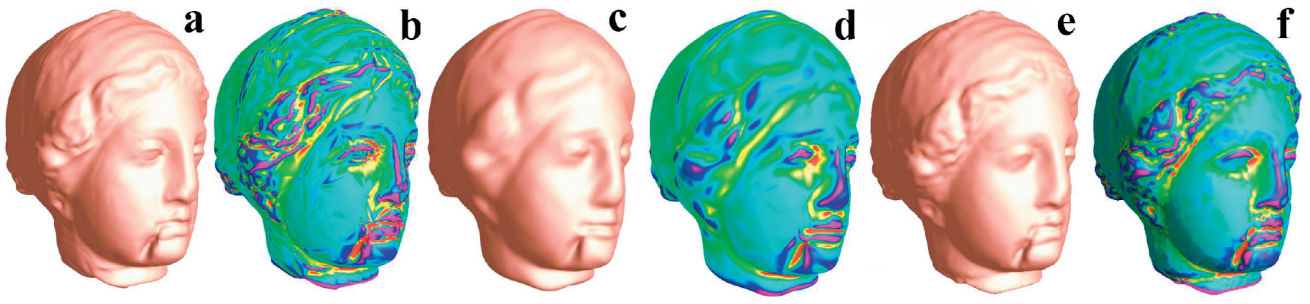



Fig. 9. Reconstructed Igea Surfaces by Quintic Splines using different methods. (a) Reconstructed quintic DMS-spline surface in [27] and its mean curvature distribution (b); (c) DMS-spline surface of (a) after fairing, whose mean curvature distribution is shown in (d); (e) Our quintic spherical DCB-spline surface and its (f) mean curvature distribution.

The convergence speed of cubic, quartic, and quintic spline surfaces reconstructed from Igea model are shown in Fig. 11b. It can be seen that, to reach the same root mean square error, splines with higher degree will take fewer iteration steps. For example, cubic, quartic, and quintic spline surfaces take eight, seven, and six iterations to satisfy the requirement of the same root mean square error 0.042 percent, respectively (see Table 2).

## 6 Comparison and Conclusion

In this paper, we have articulated a surface reconstruction scheme for fitting genus-0 closed surfaces based on the spherical generalization of Delaunay configuration B-spline (DCB-spline). The reconstructed genus-0 closed spline surface is smooth, $C^k$ continuous everywhere, and has analytic representation. Furthermore, the continuity is naturally preserved without enforcing any additional constraints.

**Comparison with previous DCB-splines.** In comparison with our previous work [31], the current framework has three significant improvements: 1) To define the spherical counterpart of planar DCB-splines in [31], we generalize the definition of Delaunay configurations from planar to spherical domain. We also articulate an efficient method for spherical Delaunay configuration computation to make this spline formulation more useful in practical applications; 2) In [31], to reconstruct a closed genus-0 surface, spline patches defined over planar disks are stitched together by $C^0$ blending functions; hence, the final representation has only $C^0$ continuity across the patches' boundaries. In this paper, the reconstructed degree-$k$ spherical DCB-spline surfaces are $C^{k-1}$ continuous across the entire domain without any segmentation and stitching process, high-order continuity is obtained elegantly; 3) In [31], we use the $k$-mean cluster method to insert only one new knot in each fitting iteration. We now generate knots by minimizing a CVT energy function with adaptive density functions. A larger set of knots are simultaneously inserted to regions with sharp geometric features or large fitting errors. The fitting efficiency has, therefore, greatly improved.

**Comparison with DMS-splines.** Degree-$k$ spline surfaces typically have continuity of order $k-1$; hence, their $(k-1)$th order derivatives will have discontinuities along the so called "knot lines," i.e., line segments between two knots used for spline reconstruction. In case of DMS-splines, a "cloud" of auxiliary knots with size of $k$ are associated with each original knot in order to reconstruct a degree-$k$ spline surface. Since the knot lines are intensively distributed near edges of the domain triangular mesh (from the triangulation of original knots), the mean curvature distributions of the surface along the curved triangular boundaries (corresponding to the edges of the domain triangulation) become much worse than other regions (see [27, Fig. 10a] and [25, Fig. 9b]). To generate a visually smooth high-quality surface, postprocessing procedures [25], [27] are necessary. Unfortunately, the fairing process unavoidably eliminate some fine details of the fitted surface as well (see Figs. 9c, 9d). In contrast, spherical DCB-splines

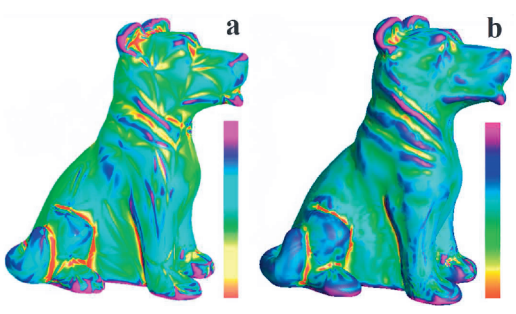



Fig. 10. Mean curvature distribution comparison of DMS-spline and spherical DCB-spline for Dog model. (a) Mean curvature of degree-5 DMS-spline surface in [25]; (b) Mean curvature of degree-5 spherical DCB-spline surface.

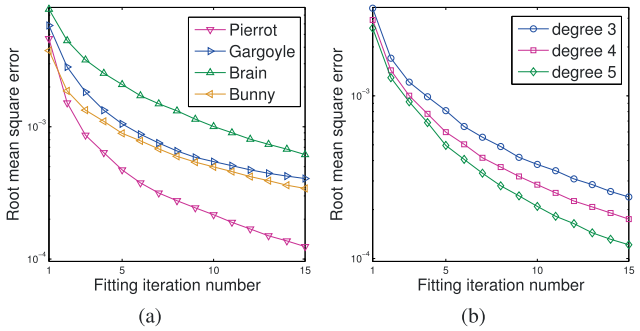



Fig. 11. Root mean square errors of surface fittings. (a) The log of RMSEs ($y$-axis) versus the number of fitting iterations ($x$-axis) for quartic surfaces reconstructed from models of Bunny, Brain, Gargoyle, and Pierrot, with 100 knots being added in each fitting step. (b) The log of RMSEs versus the number of fitting iterations for cubic, quartic, and quintic surfaces reconstructed from Igea model with 100 knots being added in each fitting step.

are free of auxiliary knots, so the "knot lines" evenly distribute over the entire parametric domain, and the curvature changes smoothly. As shown in Figs. 9e, 9f, the degree-5 spherical DCB-spline surface is visually smooth with many fine geometric details preserved.

**Limitation and future work.** One limitation of our current scheme is that, we only focus on closed genus-0 surfaces in this paper. It is much more desirable if our scheme could handle general surfaces with higher genus. One necessary step of improving our scheme is to generalize spherical mapping for high genus models, and such spherical mapping for models of complicated topology is theoretically possible. According to Riemann surface theory, a conformal map between a surface and the sphere is equivalent to a meromorphic function defined on the surface. Intuitively speaking, this map wraps the surface onto the sphere with several layers while having branch points. The number of layers and branch points are solely determined by the surface topology and Riemann-Hurwitz theorem. Different layers can be considered as different spherical domains, sharing at most two common points between each other. Upon such parameterization, $C^{k-1}$ continuous surfaces could therefore be reconstructed everywhere except at these $2g+2$ branch points (here $g$ is the genus number). We plan to explore its effective computation in surface fitting procedures.

Although we only focus our research endeavors on surface fitting in this paper, potential applications of spherical splines are much broader and not limited to shape modeling and graphics. In geophysical applications, high-order splines with data interpolation are often desirable. We plan to further refine our algorithm by integrating an effective interpolation constraint and apply it in geology, geography, and geophysics tasks.

## ACKNOWLEDGMENTS

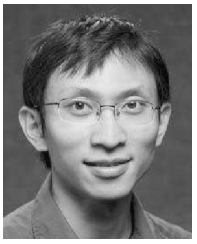
This work has been supported in part by US National Science Foundation (NSF) grants: IIS-0949467, IIS-0710819, and IIS-0830183, IIS-1047715, and IIS-1049448; the National Natural Science Foundation of China (No. 61100105, No. 61100107, No. 61170323); the Natural Science Foundation of Fujian Province of China (No. 2011J05007); and LA Board of Regents RCS LEQSF(2009-12)-RD-A-06, PFund: NSF(2011)-PFund-236, and LSU Faculty Research Grant 2010.

## REFERENCES

[1] J. Hoschek and D. Lasser, *Fundamefntals of Computer Aided Geometric Design.* A.K. Peters, 1993.
[2] M. Neamtu, "Splines on Surfaces," *Handbook of Computer Aided Geometric Design,* Chapter 9, pp. 229-253, Elsevier, 2002.
[3] G.E. Fasshauer, "Adaptive Least Squares Fitting with Radial Basis Functions on the Sphere," *Mathematical Methods for Curves and Surfaces,* pp. 141-150, Vanderbilt Univ. Press, 1995.
[4] T. Lyche and L.L. Schumaker, "A Multiresolution Tensor Spline Method for Fitting Functions on the Sphere," *SIAM J. Scientific Computing,* vol. 22, no. 2, pp. 724-746, 2000.
[5] R.E. Barnhill, K. Opitz, and H. Pottmann, "Fat Surfaces: A Trivariate Approach to Triangle-Based Interpolation on Surfaces," *Computer Aided Geometric Design,* vol. 9, no. 5, pp. 365-378, 1992.
[6] W. Freeden, M. Schreiner, and R. Franke, "A Survey on Spherical Spline Approximation," *Surveys Math. Industry,* vol. 7, pp. 29-85, 1997.
[7] H. Wang, Y. He, X. Li, X. Gu, and H. Qin, "Polycube Splines," *Proc. ACM Symp. Solid and Physical Modeling,* pp. 241-251, 2007.
[8] X. Gu, Y. He, and H. Qin, "Manifold Splines," *Proc. ACM Symp. Solid and Physical Modeling,* pp. 27-38, 2005.
[9] S.R. Buss and J.P. Fillmore, "Spherical Averages and Applications to Spherical Splines and Interpolation," *ACM Trans. Graphics,* vol. 20, no. 2, pp. 95-126, 2001.
[10] G.E. Fasshauer and L.L. Schumaker, "Scattered Data Fitting on the Sphere," *Proc. Int'l Conf. Math. Methods for Curves and Surfaces II,* pp. 117-166, 1998.
[11] V. Baramidze, M.J. Lai, and C.K. Shum, "Spherical Splines for Data Interpolation and Fitting," *SIAM J. Scientific Computing,* vol. 28, pp. 241-259, 2005.
[12] P. Alfeld, M. Neamtu, and L.L. Schumaker, "Bernstein-Bézier Polynomials on Spheres and Sphere-Like Surfaces," *Computer Aided Geometric Design,* vol. 13, pp. 333-349, 1996.
[13] P. Alfeld, M. Neamtu, and L.L. Schumaker, "Fitting Scattered Data on Sphere-Like Surfaces Using Spherical Splines," *J. Computational and Applied Math.,* vol. 73, pp. 5-43, 1996.
[14] M. Neamtu, "Homogeneous Simplex Splines," *J. Computational and Applied Math.,* vol. 73, pp. 1-2, 1996.
[15] M.-J. Lai and L.L. Schumaker, *Spline Functions on Triangulations.* Cambridge Univ. Press, 2007.
[16] G. Greiner and H.-P. Seidel, "Modeling with Triangular B-Splines," *IEEE Computer Graphics and Applications,* vol. 14, pp. 211-220, 1993.
[17] M.G.J. Franssen, "Evaluation of DMS-Splines," master's thesis, Eindhoven Univ. of Technology, 1995.
[18] C. de Boor, "On Calculating with B-Splines," *J. Approximation Theory,* vol. 6, pp. 50-62, 1972.
[19] R. Pfeifle and H.-P. Seidel, "Fitting Triangular B-Splines to Functional Scattered Data," *Proc. Graphics Interface,* pp. 26-33, 1995.
[20] H. Qin and D. Terzopoulos, "Triangular NURBS and Their Dynamic Generalizations," *Computer Aided Geometric Design,* vol. 14, no. 4, pp. 325-347, 1997.
[21] S. Han and G. Medioni, "Triangular NURBS Surface Modeling of Scattered Data," *Proc. Seventh Conf. Visualization,* pp. 295-302, 1996.
[22] Y. He and H. Qin, "Surface Reconstruction with Triangular B-Splines," *Proc. Geometric Modeling and Processing,* pp. 279-290, 2004.
[23] Y. He, X. Gu, and H. Qin, "Automatic Shape Control of Triangular B-Splines of Arbitrary Topology," *J. Computer Science and Technology,* vol. 21, pp. 232-237, 2006.
[24] R. Pfeifle and H.-P. Seidel, "Spherical Triangular B-Splines with Application to Data Fitting," *Proc. Eurographics,* pp. 89-96, 1995.
[25] Y. He, X. Gu, and H. Qin, "Rational Spherical Splines for Genus Zero Shape Modeling," *Proc. Int'l Conf. Shape Modeling and Applications,* pp. 82-91, 2005.
[26] R. Gormaz, "B-Spline Knot-Line Elimination and Bézier Continuity Conditions," *Proc. Int'l Conf. Curves and Surfaces in Geometric Design,* pp. 209-216, 1994.
[27] Y. He, X. Gu, and H. Qin, "Fairing Triangular B-Splines of Arbitrary Topology," *Proc. Pacific Graphics,* pp. 153-156, 2005.
[28] M. Neamtu, "What Is the Natural Generalization of Univariate Splines to Higher Dimensions?," *Mathematical Methods for Curves and Surfaces,* Vanderbilt Univ., 2001.
[29] M. Neamtu, "Bivariate Simplex B-Splines: A New Paradigm," *SCCG '01: Proc. 17th Spring Conf. Computer Graphics,* pp. 71-78, 2001.
[30] C. de Boor, "The Way Things Were in Multivariate Splines: A Personal View," *Mulitscale, Nonlinear and Adaptive Approximation,* pp. 10-37, Springer, 2009.
[31] J. Cao, X. Li, G. Wang, and H. Qin, "Surface Reconstruction Using Bivariate Simplex Splines on Delaunay Configurations," *Computer and Graphics,* vol. 33, no. 3, pp. 341-350, 2009.
[32] B. Dembart, D. Gonsor, and M. Neamtu, "Bivariate Quadratic B-Splines Used as Basis Functions for Data Fitting," *Mathematics for Industry: Challenges and Frontiers 2003. A Process View: Practice and Theory,* SIAM, pp. 178-198, 2005.
[33] J.-D. Boissonnat, O. Devillers, and M. Teillaud, "A Semidynamic Construction of Higher-Order Voronoi Diagrams and Its Randomized Analysis," *Algorithmica,* vol. 9, pp. 329-356, 1993.
[34] A. Okabe, B. Boots, K. Sugihara, and S.N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams,* second ed. Wiley, 2000.
[35] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[36] H.-S. Na, C.-N. Lee, and O. Cheong, "Voronoi Diagrams on the Sphere," *Computational Geometry,* vol. 23, no. 2, pp. 183-194, 2002.
[37] J.L. Brown and A.J. Worsey, "Problems with Defining Barycentric Coordinates for the Sphere," *Math. Modelling and Numerical Analysis,* vol. 26, pp. 37-49, 1992.
[38] W. Li, S. Xu, G. Zhao, and L.P. Goh, "Adaptive Knot Placement in B-Spline Curve Approximation," *Computer-Aided Design,* vol. 37, no. 8, pp. 791-797, 2005.
[39] C. Gotsman, X. Gu, and A. Sheffer, "Fundamentals of Spherical Parameterization for 3D Meshes," *Proc. ACM SIGGRAPH,* pp. 358-363, 2003.
[40] X. Gu and S.-T. Yau, "Global Conformal Surface Parameterization," *Proc. Symp. Geometry Processing,* pp. 127-137, 2003.
[41] M. Quicken, C. Brechbuhler, J. Hug, H. Blattmann, and G. Szekely, "Parameterization of Closed Surfaces for Parametric Surface Description," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 354-360, 2000.
[42] E. Praun and H. Hoppe, "Spherical Parametrization and Remeshing," *Proc. ACM SIGGRAPH '03,* pp. 340-349, 2003.
[43] R. Zayer, C. Rössl, and H.-P. Seidel, "Curvilinear Spherical Parameterization," *Proc. Int'l Conf. Shape Modeling and Applications,* pp. 57-64, 2006.
[44] P. Degener, J. Meseth, and R. Klein, "An Adaptable Surface Parameterization Method," *Proc. Int'l Meshing Roundtable,* pp. 201-213, 2003.
[45] K. Hormann and G. Greiner, "Mips: An Efficient Global Parametrization Method." *Curve and Surface Design,* pp. 153-162, Vanderbilt Univ. Press, 2000.
[46] H. Hoppe, "Progressive Meshes," *Proc. ACM SIGGRAPH,* pp. 99-108, 1996.
[47] P.V. Sander, J. Snyder, S.J. Gortler, and H. Hoppe, "Texture Mapping Progressive Meshes," *Proc. ACM SIGGRAPH,* pp. 409-416, 2001.
[48] Y. Lai, S.-H. Hu, and H. Pottmann, "Surface Fitting Based on a Feature Sensitive Spatial Tessellations: Concepts and Applications Parametrization," *Computer-Aided Design,* vol. 38, no. 7, pp. 800-807, 2006.
[49] D. Cohen-Steiner and J.-M. Morvan, "Restricted Delaunay Triangulations and Normal Cycle," *Proc. Symp. Computational Geometry,* pp. 312-321, 2003.
[50] G. Taubin, "Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation," *Proc. Int'l Conf. Computer Vision (ICCV),* pp. 902-907, 1995.
[51] M. Meyer, M. Desbrun, P. Schröder, and A.H. Barr, "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds," *Visualization and Mathematics III,* H.-C. Hege and K. Polthier, eds., pp. 35-57, Springer-Verlag, 2003.
[52] Q. Du, M.D. Gunzburger, and L. Ju, "Constrained Centroidal Voronoi Tessellations for Surfaces," *SIAM J. Scientific Computing,* vol. 24, no. 5, pp. 1488-1506, 2002.
[53] S. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Information Theory,* vol. 28, no. 2, pp. 129-137, Jan. 1982.
[54] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang, "On Centroidal Voronoi Tessellation—Energy Smoothness and Fast Computation," *ACM Trans. Graphics,* vol. 28, no. 4, pp. 1-17, 2009.
[55] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, "Isotropic Remeshing with Fast and Exact Computation of Restricted Voronoi Diagram," *Computer Graphic Forum,* vol. 28, no. 5, pp. 1445-1454, 2009.
[56] A. Gersho, "Asymptotically Optimal Block Quantization," *IEEE Trans. Information Theory,* vol. IT-25, no. 4, pp. 373-380, July 1979.

**Juan Cao** received the PhD degree in applied mathematics from Zhejiang University. She is currently an assistant professor at the School of Mathematical Sciences, Xiamen University. Her research interests include computer aided geometric design, computer graphics. For more information, visit http://59.77.0.69/~juancao/.



**Xin Li** received the BS degree in computer science from University of Science and Technology of China, the MS and PhD degrees in computer science from Stony Brook University (SUNY). He is an assistant professor of the Department of Electrical and Computer Engineering and Center for Computational and Technology at Louisiana State University. His research interests include geometric modeling, computer graphics, and vision. He is a member of the IEEE and the IEEE Computer Society. For more information, please visit http://www.ece.lsu.edu/xinli.

**Zhonggui Chen** received the BS and PhD degrees in applied mathematics from Zhejiang University, China. He is currently an assistant professor at the Department of Computer Sciences, School of Information Science and Technology, Xiamen University. His research interests include computer graphics and computational geometry.

**Hong Qin** received the BS and MS degrees in computer science from Peking University. He received the PhD degree in computer science from the University of Toronto. He is a professor in the Department of Computer Science at Stony Brook University (SUNY). His research interests include geometric modeling, graphics, physics-based modeling and simulation, visualization, and scientific computing. He is a senior member of the IEEE and the IEEE Computer Society. For more details, please visit http://www.cs.sunysb.edu/~qin.