Spherical Mapping for Processing of 3-D Closed Surfaces

Li Shen\textsuperscript{a,b,*} Fillia Makedon\textsuperscript{b}

\textsuperscript{a}Computer and Information Science, University of Massachusetts Dartmouth, 285 Old Westport Rd., North Dartmouth, MA 02747, USA

\textsuperscript{b}Dartmouth Experimental Visualization Lab (DEVLAB), Computer Science, Dartmouth College, 6211 Sudikoff Lab, Hanover, NH 03062, USA

Abstract

The spherical harmonic (SPHARM) description is a powerful surface modeling technique used by many applications in biomedical imaging. However, earlier SPHARM studies employ a spherical parameterization procedure that can be applied only to voxel surfaces. This paper presents CALD, a new spherical parameterization algorithm that makes the SPHARM model applicable to general triangle meshes. The CALD algorithm starts from an initial mapping and performs novel local and global smoothing methods alternately until a solution is approached. This new algorithm can also be used for processing of 3D closed surfaces in many different areas, including medical image analysis, computer vision, and computer graphics.

Key words: Spherical parameterization, shape analysis, surface modeling

\* Corresponding author. Phone: 1 508 910 6691, Fax: 1 508 999 9144

\textit{Email addresses: lshen@umassd.edu} (Li Shen), \textit{makedon@cs.dartmouth.edu} (Fillia Makedon).

Preprint submitted to Elsevier Science 3 February 2006
1 Introduction

Recent non-invasive scanning techniques in biomedical imaging have resulted in an explosive growth of research into the analysis of high quality 3D volumetric images. Characterizing surfaces of structures embedded in these images in a way that allows accurate comparison of object shapes has become a fundamental problem. Once descriptors that represent shape in some standard fashion are extracted, shape classification \[1,2\] can be done to help diagnosis, and statistical shape models \[3,4\] can be created to better understand the structure or help segmentation of the same structure in new images.

Numerous 3D shape representation techniques have been proposed in several fields, including computer vision, medical image analysis, and computer graphics. For example, several shape descriptors (\textit{e.g.}, \[5–11\]) have been studied for retrieving similar objects from databases of 3D models. For the purpose of similarity search, these descriptors are designed to be quick to compute, concise to store, and easy to index \[6\], and so they are often relatively compact. In the areas of computer vision and medical image analysis, more powerful shape descriptors have been developed for statistical shape analysis that aims to detect or locate shape changes between two groups of 3D objects. These descriptors include landmark-based descriptors \[12,13\], deformation fields generated by mapping a segmented template image to individuals \[14,15\], distance transforms \[16\], medial axes \[4,17,18\], and parametric surfaces \[16,19,20\]. In this work, we focus on parametric surfaces using spherical harmonics.

Spherical harmonics were originally used as a type of parametric surface representation for radial or stellar surfaces \(r(\theta, \phi)\) \[16\], and later extended to more
general shapes by representing a surface using three functions of \( \theta \) and \( \phi \) [19]. Brechbühler, Gerig, Quicken et al. [19,21] parameterize a voxel surface on a sphere by minimizing area distortion and then employ the spherical harmonic (SPHARM) expansion to describe the original shape. A voxel surface is a square surface mesh converted from an isotropic voxel representation for a 3D object. Fig. 1 shows a sample voxel surface and its SPHARM reconstructions.

Among the major shape descriptors mentioned before, SPHARM is one of the best shape description methods for arbitrarily shaped but simply connected 3D objects. It is suitable for surface comparison and can deal with protrusions and intrusions. It is a fine-scale global shape descriptor with several advantages, including inherent interpolation, implicit correspondence, and accurate scaling. Because of this, it can be used to derive many other widely used shape descriptors such as landmark-based descriptors [1], deformation fields [22], and medical axes [4]. In addition, SPHARM has been successfully applied to many applications in biomedical imaging [1,3,4,23,24].

The key step for deriving a SPHARM description for a 3D object is to parameterize its surface onto a sphere by minimizing area distortion. However, earlier spherical mapping techniques [19,21] can be applied only to voxel surfaces and not to general triangle meshes, since it exploits the uniform quadrilateral structure of a voxel surface. In addition, according to [19], the voxel surface representation has an undesired property: given a planar surface, the area of its voxel quantization depends on its orientation (e.g., \( \sqrt{3} \) times the actual area when orthogonal to \((1,1,1)^T\)), which breaks the area preserving property.

To overcome these limitations, we develop a new spherical parameterization
algorithm for general genus-zero surfaces\footnote{According to [25], \textit{genus}, a topologically invariant property of a surface, is defined as the largest number of nonintersecting simple closed curves that can be drawn on the surface without separating it. Generally speaking, it is the number of holes in a surface. Therefore, a \textit{genus-zero surface} is a surface with spherical topology.} that aims to control area and length distortions at the same time. This new technique makes the SPHARM model applicable to general triangle meshes. It can also be used for surface representation and manipulation in different areas such as medical image analysis, computer vision, multimedia databases (for shape indexing, compression, and similarity search), and computer graphics (for texture mapping, morphing, and remeshing).

This paper is organized as follows. The rest of Section 1 provides a brief discussion on surface parameterization and introduces some necessary concepts. Section 2 describes our algorithm and its components in detail. Section 3
demonstrates the usefulness of our technique with a few surface processing applications. Section 4 concludes this paper.

1.1 Surface Parameterization

Surface parameterization [26] creates a one-to-one mapping from some parameter domain to a surface. The parameter domain itself is often a surface, and so creating a parameterization means mapping one surface $S_1$ to another surface $S_2$. A good mapping requires a minimization of some types of distortions. Three typical distortions are length distortion, angle distortion, and area distortion. To minimize these distortions, there are three respective types of mappings as follows.

- **Isometric mappings**: A mapping from $S_1$ to $S_2$ is isometric or length-preserving if each arc on $S_1$ is mapped to an arc on $S_2$ with the same length.

- **Conformal mappings**: A mapping from $S_1$ to $S_2$ is conformal or angle-preserving if the angle of each pair of intersecting arcs on $S_2$ is the same as that of the corresponding arc pair on $S_1$.

- **Equiareal mappings**: A mapping from $S_1$ to $S_2$ is equiareal or area-preserving if each part on $S_1$ is mapped onto a part on $S_2$ with the same area.

A mapping is isometric if and only if it is conformal and equiareal [26]. Thus, an isometric mapping is an ideal mapping without any length, angle, or area distortion. However, isometric mappings only exist in very special cases, for example, the mapping of a cylinder onto a plane that transforms cylindrical
coordinates into cartesian coordinates. In practice, surface parameterization often aims to seek a mapping that is conformal \( (e.g., [27,28]) \) or equiareal \( (e.g., [19,21,29]) \), or minimizes either some type of distortion or a combination of different distortions \( (e.g., [29–31]) \).

In the area of the SPHARM surface modeling, an equiareal mapping is more attractive than a conformal mapping because we want to treat each area unit on object surface equally by assigning the same amount of parameter space to it. Note that we always scale an object surface so that its total area is equal to the surface area of a unit sphere. The equiareal mapping strategy seems a reasonable one for the purpose of surface comparison: the common parameter space facilitates the comparison between two surfaces, and a region on one surface is always compared with a region on the other surface with the same area. Certainly, in some cases, an additional step of surface alignment needs to be done for parameterized surfaces before an actual surface comparison can be performed. This step may involve surface registration in both parameter space and object space \([19,32]\).

Many studies have been done in surface parameterization, see \([26]\) for a comprehensive survey. \textit{Spherical parameterization} is a special type of surface parameterization in which the parameter domain is the surface of a unit sphere. In the field of spherical parameterization, there are very few studies that attempt to create equiareal mappings. One such approach is by Brechbühler, Gerig, Quicken, \textit{et al.} \([19,21]\); however, as mentioned before, it does not apply to general triangle surfaces. Another approach by Sheffer, Gotsman, and Dyn \([29]\) seeks to create an area preserving mapping by solving a constrained minimization problem. However, this approach is quite slow and not practical for meshes with more than a few hundred vertices.
In this paper, we present a new spherical parameterization algorithm that attempts to minimize the area distortion of a spherical mapping. Praun and Hoppe [30] have recently developed an effective spherical mapping method that minimizes distortion in vector length, but without considering the area distortion. Inspired by their work, we aim to control area and length distortions at the same time, since uncontrolled area preserving mapping often introduces relatively large length distortions (e.g., see [26,31,33]). We call our algorithm \textbf{CALD}, which stands for \textit{Control of Area and Length Distortions}. In the next subsection, we introduce some necessary concepts and describe our goals.

1.2 Concepts and Goals

Let $M = \{t_i\}$ be a triangle mesh and let $\Psi$ be a continuous invertible map which maps $M$ to another mesh $\Psi(M) = \{\Psi(t_i)\}$. $A(\cdot)$ is used to denote the area of a triangle or a mesh. The area distortion cost (ADC) $C_a$ with respect to $\Psi$ is defined as follows:

- For each triangle $t_i \in M$,
  \[ C_a(t_i, \Psi) = \frac{A(\Psi(t_i))}{A(t_i)}. \tag{1} \]
  This measures the local ADC of a single triangle.

- For each mesh vertex $v$ in $M$,
  \[ C_a(v, \Psi) = \frac{\sum_{t_i \in M_v} A(\Psi(t_i))}{\sum_{t_i \in M_v} A(t_i)}, \tag{2} \]
  where $M_v$ is the set of triangles incident upon $v$, i.e., a local submesh. This measures the local ADC around a single vertex.
For the whole mesh $M$,

$$C_a(M, \Psi) = \sum_{t_i \in M} \frac{\max(C_a(t_i, \Psi), \frac{1}{C_a(t_i, \Psi)}) \times A(\Psi(t_i))}{A(\Psi(M))}.$$  \hspace{1cm} (3)

This measures the overall ADC for the whole mesh. By taking

$$\max(C_a(t_i, \Psi), \frac{1}{C_a(t_i, \Psi)})$$

as the ADC contribution from each triangle, we treat contraction and expansion equally, and so always have $C_a(M, \Psi) \geq 1$.

The stretch concept introduced by Sander et al. [31] is employed in this study. They consider the case of a mapping $\Psi : (s, t) \in R^2 \rightarrow (x, y, z) \in R^3$ from a planar domain to a 3D surface. At any point $(s, t)$, the singular values $\Gamma$ and $\gamma$ of the $3 \times 2$ Jacobian matrix $J_\Psi[d\Psi/ds \ d\Psi/dt]$ represent the largest and smallest length distortions (called stretches) when mapping a vector from the 2D domain to the 3D surface. In our case, we can always rotate each triangle on an object surface to a planar domain, and then calculate the stretch by mapping it to its spherical parameterization. For our triangle mesh, the mapping $\Psi$ is piecewise linear and its Jacobian $J_\Psi$ is constant over each triangle.

Given a mesh mapping $\Psi$ from $M$ to $\Psi(M)$, we define the average length distortion cost (LDC) $C_s$ and the worst LDC $C^w_s$ as follows:

$$C_s(M, \Psi) = \sqrt{\frac{\sum_{t_i \in M} (\Gamma^2(t_i) + \frac{1}{\gamma^2(t_i)}) \times A(\Psi(t_i))}{2 A(\Psi(M))}}, \hspace{1cm} \text{and} \hspace{1cm} (4)$$

$$C^w_s(M, \Psi) = \max\{\max(\Gamma(t_i), \gamma(t_i)) \mid t_i \in M\}, \hspace{1cm} (5)$$

where $\Gamma(t_i)$ and $\gamma(t_i)$ are the largest and smallest length distortions for a triangle $t_i$. Again, we treat contraction and expansion equally in both definitions.
Now the problem can be described as follows. Let $M$ be a triangle mesh of a closed genus-zero surface and $S$ be the surface of a unit sphere. A spherical parameterization of $M$ is a continuous invertible map $\Psi : M \rightarrow S$ from the object surface to the unit sphere, where each mesh vertex $v$ is assigned a parameterization $\Psi(v) \in S$. Thus, each mesh edge $e$ is mapped to a great circle arc $\Psi(e)$, and each mesh triangle $t$ is mapped to a spherical triangle $\Psi(t)$ bounded by these arcs. *In this work, we assume that any given object surface has already been scaled to have the same total area as that of the unit sphere, which is $4\pi$. Thus, our goal is to create a mapping $\Psi$ which has minimized $C_a(M, \Psi)$ and controlled $C_s(M, \Psi)$.*

## 2 CALD Framework

The CALD algorithm is developed to achieve the above goal of minimizing the area distortion $C_a(M, \Psi)$ while controlling the length distortion $C_s(M, \Psi)$. Note that, in our case, each parameterization is actually a spherical mesh. We say that this mesh is of good quality if both the area and length distortions of the corresponding parameterization are small. The CALD algorithm starts from an initial parameterization and then employs an iterative procedure to improve the quality of the spherical mesh gradually for a better parameterization. The operation used in the iterative procedure is called mesh smoothing. Mesh smoothing [34,35] can be described as a procedure that relocates mesh vertices to improve the mesh quality without changing mesh topology.

There are two common types of smoothing techniques; see [34] for a brief survey. The most commonly used smoothing method is Laplacian smoothing, which moves each mesh vertex to the geometric center of its neighbour.
vertices. This method is simple and fast but it does not guarantee a mesh quality improvement. Another type of smoothing is optimization-based: the vertices are relocated to minimize a given distortion metric. Optimization-based smoothing has a much higher computational cost than Laplacian smoothing but provides better results.

Most mesh smoothing problems aim to improve a mesh so that the shape of mesh elements does not vary significantly [35]. This work has a different goal: we want to improve a spherical mesh so that each mesh element preserves the area of its corresponding mesh element on the object surface, while we want to control length distortions at the same time. To achieve this goal, two new smoothing techniques, a local smoothing approach and a global smoothing approach, are developed and integrated in the CALD framework.

The CALD framework contains three key components. The initial parameterization step is an extension of Brechbühler’s method [19] for triangle meshes. A novel local smoothing method and a novel global smoothing method are developed to improve the quality of the parameterization iteratively. The local smoothing step aims to minimize the ADC at a local submesh by solving a linear system and also to control its worst LDC at the same time. The global smoothing step calculates the distribution of ADCs for all the mesh vertices and tries to equalize them over the whole sphere. The overall algorithm combines the local and global methods together, and performs each method alternately until a solution is achieved. The following three subsections describe three key components in the CALD framework.
2.1 Initial Parameterization

Praun and Hoppe [30] and Brechbühler et al. [19] employ different overall strategies in their spherical mapping algorithms.

- Praun and Hoppe use a coarse-to-fine strategy, by first mapping a base tetrahedron to the sphere, traversing a progressive mesh sequence, and inserting vertices on the sphere while minimizing the length distortion.
- Brechbühler performs an initial parameterization and then improves the mapping by solving a constrained optimization problem.

Our approach follows Brechbühler’s work, adapts his initial mapping method to triangle meshes, and applies local and global mesh smoothing steps instead of solving an optimization problem.

Under this framework, when necessary, an input mesh is always subdivided so that the length of its longest edge does not exceed some threshold. Thus, for each input mesh, a fine sampling resolution is achieved over the entire surface. This allows us to manipulate any local submesh approximately in a piecewise linear way.

For convenience, we employ the following convention for spherical coordinates which is the same as that used in the spherical harmonics definition [36]: $\theta$ is taken as the polar (colatitudinal) coordinate with $\theta \in [0, \pi]$, and $\phi$ as the azimuthal (longitudinal) coordinate with $\phi \in [0, 2\pi]$.

Algorithm 1 shows the outline of our initial parameterization approach. In the first step, the north pole ($\theta = 0$) and the south pole ($\theta = \pi$) are chosen to be two vertices whose projections onto the principal axis of the object are the
Algorithm 1 Initial parameterization.

1: select two poles from the input mesh $M$

2: parameterize $\theta$ and $\phi$ to form the mapping $\Psi_1(M)$

3: rotate $\Psi_1(M)$ about x-axis by $\frac{\pi}{2}$, reparameterize $\phi$, rotate back, and get the mapping $\Psi_2(M)$

4: rotate $\Psi_2(M)$ about y-axis by $\frac{\pi}{2}$, reparameterize $\phi$, rotate back, and get the mapping $\Psi_3(M)$

5: pick the best from $\Psi_1(M)$, $\Psi_2(M)$, and $\Psi_3(M)$

furthest apart, since poles that are further apart often yield a better initial mapping. The second step solves two sets of linear systems to get colatitudes ($\theta$) and longitudes ($\phi$) for all the mesh vertices. This is done by requiring each vertex value (i.e., its $\theta$ or $\phi$) to be the weighted average of its neighbours, where weights are inversely proportional to its distances from the neighbours. The above two steps form an extension of Brechbühl’s initial parameterization approach for general triangle mesh cases.

We observe that, in terms of area preservation, the above process often results in a better distributed longitude than colatitude. Thus, we design the third and fourth steps to better distribute the colatitude by recalculating the longitude for a rotated mapping. The sixth step simply picks the best mapping (i.e., with the minimal overall ADC defined in Equation (3)) as the initial parameterization.

Fig. 2 shows a sample run of Algorithm 1, where we can see that step 3 and step 4 in the algorithm do help improve the quality of the mapping in the given case. Fig. 3 shows another sample initial mapping result.
Fig. 2. Sample initial parameterization for a triangle mesh: (a) object surface, (b–d) spherical mappings after steps 2–4 in Algorithm 1, where overall ADC $C_a$ is provided for each mapping. We observe that step 3 and step 4 in Algorithm 1 do help improve the quality of the mapping in this case. The line on the object surface in (a) corresponds to the line on each spherical mapping.

2.2 Local Smoothing Algorithm

Our local smoothing step uses a loop to traverse the whole spherical mesh, and at each iteration it tries to adjust a vertex in a local submesh consisting of
Fig. 3. Sample local submesh is indicated in green color on the object surface (left) and its initial parameterization (right). The submesh to be improved is the one on the sphere.

the elements incident to the vertex to gradually improve the mapping quality. Fig. 3 shows a sample local submesh on both object surface and its initial parameterization. To avoid confusion, we call a submesh on an object surface an object submesh, and call a submesh on a spherical parameterization a parameter submesh. This local smoothing step tries to improve the quality of each parameter submesh. Since a fine resolution mesh representation is used in this study, each parameter submesh, which is a small part of the sphere, can be treated as approximately piecewise linear. Thus, it can be projected onto an appropriate 2D plane. Under the assumption of piecewise linearity, the projection preserves the relative area of each triangle on the parameter submesh.

Fig. 4 shows such a sample local parameter mesh on a 2D plane, where $v_0$ is the free vertex to adjust. Given $v_i = (x_i, y_i)$, the signed area $A$ of triangle $t_1 \equiv \{v_0, v_1, v_2\}$ can be calculated as follows [37]:

$$A = \frac{1}{2} ((x_2 - x_1) \times (y_0 - y_1) - (x_0 - x_1) \times (y_2 - y_1)).$$

(6)
Based on the corresponding object submesh, we know the ideal area $A_{\text{ideal}}$ that each triangle on the parameter submesh should achieve. Thus, replacing $A$ with $A_{\text{ideal}}$ and treating $x_0$ and $y_0$ as the only unknowns in Equation (6), we can formulate a linear system using all the triangles in the parameter submesh and solve it in a least squares fashion to locate a new center vertex position. This new center position minimizes the square sum of the relative area differences between the object submesh elements and the parameter submesh elements. Note that only the center position is concerned here, which indicates that the border of a parameter submesh is fixed. Thus, the total area of the parameter submesh cannot be changed. Therefore, in this step, each triangle in the local parameter submesh aims to achieve not its correct absolute area but the correct area relative to the other triangles in the selected object submesh.

Take the case shown in Fig. 4 as an example. For each triangle $t_i$ ($i \in \{1, 2, 3, 4, 5\}$) on this 2D projection of a parameter submesh, let $t'_i$ be its corresponding triangle on the object surface. Thus, the corresponding object submesh is formed by $\{t'_1, t'_2, t'_3, t'_4, t'_5\}$. If we use $A(\cdot)$ to denote the area of a triangle, the relative area of $t'_i$ in this object submesh can be given by $\frac{A(t'_i)}{\sum_{k=1}^{5} A(t'_k)}$. In order to preserve this relative area, $t_i$ on the 2D project should
have an ideal area of

\[ A_i = \frac{A(t_i^\prime)}{\sum_{k=1}^{5} A(t_k^\prime)} \times A_{\text{total}}, \]

where \( A_{\text{total}} \) is the total area of the 2D projection of the parameter submesh.

To calculate the new location \((x, y)\) for the parameter submesh center \(v_0\) on its 2D projection, the following set of linear equations can then be formulated:

\[
\begin{align*}
(x_2 - x_1) \times (y - y_1) - (x - x_1) \times (y_2 - y_1) &= 2A_1 \\
(x_3 - x_2) \times (y - y_2) - (x - x_2) \times (y_3 - y_2) &= 2A_2 \\
(x_4 - x_3) \times (y - y_3) - (x - x_3) \times (y_4 - y_3) &= 2A_3 \\
(x_5 - x_4) \times (y - y_4) - (x - x_4) \times (y_5 - y_4) &= 2A_4 \\
(x_1 - x_5) \times (y - y_5) - (x - x_5) \times (y_1 - y_5) &= 2A_5
\end{align*}
\]

The new center location \((x, y)\) is obtained by solving this linear system in a least squares sense.

Algorithm 2 shows the pseudo-code of this local smoothing algorithm, and Fig. 5 shows some notation used in Algorithm 2. The algorithm uses a loop to traverse the whole spherical mesh. In each iteration, it deals with one local submesh at a time. In Lines 2–3, a local submesh \(M_v\) on the object surface and its local parameterization \(P_v = \{\hat{v}, \hat{v}_1, ..., \hat{v}_n\}\) are extracted. The goal is to calculate a new position for \(\hat{v} = \{v_x, v_y, v_z\}\). In Lines 4–10, \(P_v\) is projected onto the tangent plane \(T\) of \(\hat{v}\) to form \(P_v^t\) and then to a 2D coordinate plane to form \(P_v^p\), where the coordinate plane is chosen according to the direction of \(\hat{v}\) to avoid degeneracy. As mentioned earlier, our input mesh has a fine sampling resolution, and so \(P_v^t\) is assumed to be a piecewise linear approximation of \(P_v\).

In addition, the projection from \(P_v^t\) to \(P_v^p\) does not change the relative areas of the elements in the submesh. Thus, in Lines 11–12, we can formulate the linear system using \(P_v^p\) and \(M_v\), solve for the new center vertex position \(c_p\),
Algorithm 2 Local smoothing.

1: for each vertex $v$ in object mesh $M$ do
2: get local object submesh $M_v = \{v, v_1, ..., v_n\}$
3: get local parameterization $P_v = \{\hat{v}, \hat{v}_1, ..., \hat{v}_n\}$
4: project $P_v$ onto $\hat{v}$'s tangent plane $T$ to form $P_v^t$
5: if $\max\{|\hat{v}_x|, |\hat{v}_y|, |\hat{v}_z|\} = |\hat{v}_x|$ then
6: project $P_v^t$ onto $yz$ plane to form a planar mesh $P_v^p$
7: if $\max\{|\hat{v}_x|, |\hat{v}_y|, |\hat{v}_z|\} = |\hat{v}_y|$ then
8: project $P_v^t$ onto $xz$ plane to form a planar mesh $P_v^p$
9: if $\max\{|\hat{v}_x|, |\hat{v}_y|, |\hat{v}_z|\} = |\hat{v}_z|$ then
10: project $P_v^t$ onto $xy$ plane to form a planar mesh $P_v^p$
11: get new center $c_p$ on $P_v^p$ by solving a linear system
12: project $c_p$ back to the tangent plane $T$ to get $c$
13: $\text{cost} = \infty$
14: for $u_t \in \{w \times \hat{v} + (1 - w) \times c \mid w \in \{0, \frac{1}{3}, \frac{2}{3}, 1\}\}$ do
15: project $u_t$ to the unit sphere to get $u$
16: $P_v' = \{u, \hat{v}_1, ..., \hat{v}_n\}$, $\Psi$ maps $M_v$ to $P_v'$
17: if $C_w^s(M_v, \Psi) < \text{cost}$ then
18: $\hat{v} = u$, $\text{cost} = C_w^s(M_v, \Psi)$

and then project it back to the tangent plane $T$ to get $c$.

As mentioned before, uncontrolled area preserving mapping often creates large length distortions. To avoid this, we consider controlling area and length distortions at the same time. We pick four points on the segment $\overline{\hat{v}c}$, evenly spaced from old position $\hat{v}$ towards the new position $c$, and then project these points back to the unit sphere. Now we have four cases of local reparameterization. We calculate the worst length distortion cost (LDC) according to
Fig. 5. Notation used in Algorithm 2. $P_v = \{\hat{v}, \hat{v}_1, ..., \hat{v}_n\}$ (red mesh) is the local parameterization on the sphere, and its center vertex is $\hat{v} = \{-0.0380, 0.5064, 0.8615\}$. $T$ (yellow plane) is $\hat{v}$’s tangent plane, and $P^t_v$ (blue mesh) is the projection of $P_v$ onto $T$. According to Algorithm 2 and the value of $\hat{v}$, $P^t_v$ is then projected onto the $xy$ plane to form a planar mesh $P^p_v$ (green mesh). On $P^p_v$, a new center vertex $c_p$ (green diamond) is obtained by solving a linear system. After projecting $c_p$ back to the tangent plane $T$, Algorithm 2 creates $u_i$’s (blue stars) on $T$ and then projects $u_i$’s onto the sphere to get $u$’s (red circles). These $u$’s become the candidate center vertices for the new local parameterization.

Equation (5) for each case and pick the best one to relocate $\hat{v}$. The above process is described in Lines 13–18. This completes the description of the algorithm.

The intuition behind the above strategy is to control both area and length distortions by moving the free vertex towards the area preserving direction.
until the length distortion increases. Although this strategy is very simple, we observe that it is an effective one. Note that we employ an iterative procedure in our CALD framework. Thus, the old position of a free vertex is actually preferable to some extent, since it was the result of the previous iteration and was aimed to reduce the length distortion. In the current iteration, our strategy aims to reduce this length distortion further by moving the center vertex towards the area-preserving direction.

Note that we always calculate signed areas for mesh elements in this step. For any inverted mesh element, it has a negative area. Our object surface mesh is assumed to be free of inverted elements, and so each of its elements has a positive area. Thus, by design, it is evident that our local smoothing step tends to result in a spherical parameterization in which all the mesh elements also have positive areas. Therefore, this local smoothing step has the “mesh untangling” functionality. That is, this step has the ability to move the position of a vertex from an infeasible region to a feasible region where all the incident elements have positive areas, and consequently an inverted element can become a valid one. This is a very desirable feature in a surface parameterization or a mesh smoothing study.

2.3 Global Smoothing Algorithm

The global smoothing step calculates the distribution of area distortion costs (ADCs) for all the mesh vertices according to Equation (2) and tries to equalize them over the whole sphere. First, we introduce the concept of area scaling ratio function (ASRF) to capture the distribution of ADCs over the sphere. Second, we discuss an approach to convert the current mapping to an equal
area mapping using the ASRF concept in an ideal case. Finally, we present our global smoothing algorithm by adapting the above approach to practical situations.

2.3.1 Area Scaling Ratio Function

$A(\cdot)$ is used to denote the area of a triangle, a mesh, or a surface region. Let $M$ be the object surface mesh and $\Psi(M)$ be the current parameterization. For each vertex $v$ in $M$, we use $(\theta_v, \phi_v)$ to denote its parameterization on the unit sphere. The distribution of ADCs for parameterization $\Psi(M)$ can then be formulated as a spherical function $F_\Psi(\theta, \phi)$ such that for each $v$ in $M$,

$$F_\Psi(\theta_v, \phi_v) = C_a(v, \Psi) = \frac{\sum_{t_i \in M_v} A(\Psi(t_i))}{\sum_{t_i \in M_v} A(t_i)}, \quad (7)$$

where $M_v$ is the set of triangles incident upon $v$, and $C_a(v, \Psi)$ is the local ADC of $v$ (see Equation (2)). In this work, we call $F_\Psi(\theta, \phi)$ the area scaling ratio function (ASRF in short) of $\Psi$. Note that the concept of ASRF is similar to the concept captured by the Jacobian determinant [38] for a mapping between two (continuous) surfaces: the absolute value of the Jacobian determinant at a certain surface point $p$ gives us the factor by which the mapping expands or shrinks the area near $p$.

As mentioned earlier, before the analysis, the object surface is always scaled to have the same area as that of the unit sphere which is $4\pi$. Therefore, an area preserving mapping $\Psi$ should have $F_\Psi(\theta, \phi) = 1$ at any location. Thus, our goal is now to equalize the ASRF values over the whole sphere by redistributing those $(\theta_v, \phi_v)$'s.
2.3.2 An Ideal Case

For simplicity, let us first consider the continuous case. In this case, an object surface is assumed to be a continuous domain instead of a discrete mesh, and the spherical surface is likewise assumed to be continuous. We can think of this as the extreme discrete case where a surface mesh consists of an infinite number of infinitely small elements. Some useful notation is introduced as follows.

Given $0 \leq \theta_1 \leq \theta_2 \leq \pi$ and $0 \leq \phi_1 \leq \phi_2 \leq 2\pi$, $R(\theta_1, \phi_1, \theta_2, \phi_2)$ is used to denote the region bounded by $\theta_1 \leq \theta \leq \theta_2$ and $\phi_1 \leq \phi \leq \phi_2$ on the sphere:

$$R(\theta_1, \phi_1, \theta_2, \phi_2) \equiv \{(\theta, \phi) \mid \theta_1 \leq \theta \leq \theta_2, \phi_1 \leq \phi \leq \phi_2\}.$$  

Let $\Psi$ be a spherical mapping from an object surface $M$ to the surface of a unit sphere. Given a region $X$ on the unit sphere, $\Psi^{-1}(X)$ is used to denote the region on the object surface that is mapped to $X$ according to $\Psi$. Thus $\Psi^{-1}$ is the inverse mapping of $\Psi$. In this section, for convenience, $\Psi$ is used to denote not only the mapping but also the spherical surface itself. Note that, in previous sections, the spherical surface is often referred to as $\Psi(M)$.

For $0 < \theta < \pi$, $0 < \phi < 2\pi$ and a small enough $\delta > 0$,

$$R_\delta(\theta, \phi) \equiv R(\theta - \delta, \phi - \delta, \theta + \delta, \phi + \delta)$$

defines a small neighbourhood region of point $(\theta, \phi)$ on the unit sphere.

Thus, according to Equation (7), in the continuous case it is reasonable to assume that $F_\Psi(\theta, \phi)$ captures the real area scaling ratio at the parameter
location \((\theta, \phi)\):

\[
F_\Psi(\theta, \phi) = \lim_{\delta \to 0} \frac{A(R_\delta(\theta, \phi))}{A(\Psi^{-1}(R_\delta(\theta, \phi)))}.
\] (8)

Note that for simplicity, here we assume that \(F_\Psi(\theta, \phi)\) is a continuous function defined on the sphere.

Given any vertex \((\theta_v, \phi_v)\) on the current mapping \(\Psi\), we consider the region

\[ R_v \equiv R(0, 0, \theta_v, \phi_v), \]

which is the parameter region covered by \(0 \leq \theta \leq \theta_v\) and \(0 \leq \phi \leq \phi_v\); see Fig. A.1(a) for a visualization. By definition, its area is given by

\[
A(R_v) = \int_0^{\theta_v} \int_0^{\phi_v} \sin \theta \, d\phi \, d\theta = \phi_v (1 - \cos \theta_v).
\] (9)

Let \(R_{obj} \equiv \Psi^{-1}(R_v)\) be the object surface part that maps to \(R_v\) according to \(\Psi\); and, based on Equation (8), its area is given by

\[
A(R_{obj}) = A(\Psi^{-1}(R_v)) = \int_0^{\theta_v} \int_0^{\phi_v} \frac{1}{F_\Psi(\theta, \phi)} \sin \theta \, d\phi \, d\theta.
\] (10)

Therefore, the total area \(A(M)\) of the object surface \(M\) can be given by

\[
A(M) = A(\Psi^{-1}(R(0, 0, \pi, 2\pi))) = \int_0^{\pi} \int_0^{2\pi} \frac{1}{F_\Psi(\theta, \phi)} \sin \theta \, d\phi \, d\theta = 4\pi,
\] (11)

where \(R(0, 0, \pi, 2\pi)\) is the surface of the unit sphere.

Our strategy for global smoothing is to rearrange the parameter space so that every \(R_v\) is converted to a new spherical region that has the same area as the corresponding \(R_{obj}\). In other words, we convert the current spherical mapping
Ψ to a new spherical mapping Ψ_2 by redistributing every \((\theta_v, \phi_v)\) such that \(R_v\) in Ψ is converted to a new region \(R_{v2}\) in Ψ_2 and \(R_{v2}\) satisfies \(A(R_{v2}) = A(R_{\text{obj}})\). This can be done in two steps:

1. Create a new spherical mapping Ψ_1 by moving each \((\theta_v, \phi_v)\) in Ψ to a new position \((\theta'_v, \phi_v)\) in Ψ_1 such that
   \[
   \theta'_v = f(\theta_v) = \arccos \left( 1 - \frac{1}{2\pi} \int_0^{2\pi} \frac{1}{F_{\Psi}(\theta, \phi)} \sin \theta \, d\phi \right). \tag{12}
   \]
   Note that only the colatitude is changed in this step. Let \(R_{v1}\) be the new parameter region in Ψ_1 that maps to \(R_{\text{obj}}\). From Equation (12), we have
   \[
   R_{v1} = R(0, 0, \theta'_v, \phi_v). \tag{13}
   \]

2. Create a new spherical mapping Ψ_2 by moving each \((\theta'_v, \phi_v)\) in Ψ_1 to a new position \((\theta'_v, \phi'_v)\) in Ψ_2 such that
   \[
   \phi'_v = g(\theta'_v, \phi_v) = \int_0^{\phi_v} \frac{1}{F_{\Psi_1}(\theta'_v, \phi)} \, d\phi, \tag{14}
   \]
   where \(F_{\Psi_1}\) is the ASRF of Ψ_1. Note that only the longitude is changed in this step. Let \(R_{v2}\) be the new parameter region in Ψ_2 that maps to \(R_{\text{obj}}\). From Equation (12) and Equation (14), we have
   \[
   R_{v2} = \{(\theta, \phi) \mid 0 \leq \theta \leq \theta'_v, 0 \leq \phi \leq g(\theta, \phi_v)\}. \tag{15}
   \]

The above procedure can be formulated as the following Theorem and its correctness is shown in Appendix A.

**Theorem 1** Let \(R_v\) be a region on a spherical mapping Ψ. Change Ψ to Ψ_1 by redistributing every colatitude using Equation (12), and \(R_v\) on Ψ becomes \(R_{v1}\) on Ψ_1. Change Ψ_1 to Ψ_2 by redistributing every longitude using Equation (14), and \(R_{v1}\) on Ψ_1 becomes \(R_{v2}\) on Ψ_2. After the above procedure, \(R_{v2}\) on Ψ_2
preserves the area of its corresponding part on the object surface: $A(R_{v2}) = A(\Psi_2^{-1}(R_{v2}))$.

**PROOF.** See Appendix A.

### 2.3.3 A Practical Algorithm

According to Theorem 1, the reparameterization procedure described by Equation (12) and Equation (14) can convert every $R_v$ on $\Psi$ to a new region $R_{v2}$ on $\Psi_2$ that preserves the area of the corresponding object surface part. Theoretically, this procedure works in the continuous case for creating a mapping with minimized area distortion. However, it considers only the minimization of area distortion and not that of length distortion. Thus, for an original mapping with relatively large area distortion, it is often the case that the above procedure introduces additional length distortions in order to achieve the goal of area preservation.

One reason behind the above phenomenon could be as follows. Our reparameterization procedure has two design constraints: (1) the north and south poles are not changed; see Equation (12) and Lemma 2; and (2) the longitude of any point on the date line (i.e., the line with $\phi = 0$) also remains the same, see Equation (14) and Lemma 4. Note that Lemmas 2 and 4 are introduced in Appendix A.

It is evident that these constraints are rotation-dependent. If we rotate a spherical mapping to a different orientation, the poles, the date line, or both will change to different mapping locations. Thus, a different set of constraints will be implied, and usually these different constraints will result in a differ-
ent reparameterization. Therefore, given an original mapping with a *random* orientation, it is highly unlikely that this reparameterization procedure can result in an equal area mapping that happens to also have well-controlled length distortions.

From our experiments in which we directly apply both Equation (12) and Equation (14) to a spherical mapping, we also observe that this reparameterization procedure can introduce additional length distortions. In these experiments, each spherical mapping is represented by a discrete mesh instead of a continuous domain. As a result, additional length distortions introduced by this procedure are sometimes reflected as tangled mesh elements in a reparameterized mapping. From these experiments, we observe that these additional length distortions are often caused by relatively large movements of either longitudes or colatitudes. By our design, longitudes tend to move more “aggressively” than colatitudes. Based on Equation (14), a substantial local area distortion can directly cause longitudes to make large movements. However, it may not have such a significant impact on colatitudes, since the smoothing strategy for colatitudes (see Equation (12)) considers the average area distortion over all possible longitudes and this average distortion may reduce or even cancel the effect introduced by a single local distortion.

With the above observations, in order to get a better smoothing result, we develop a modified reparameterization procedure, which we call the *global smoothing* procedure. Our overall strategy is to make a modest improvement towards the area preservation goal in this global smoothing step, and then integrate it with the local smoothing approach described before in an iterative framework. In this framework, we perform the local and global smoothing steps alternately. We hope to use the local smoothing step to control length
Algorithm 3 Global smoothing.

1: for each vertex $v(\theta, \phi)$ in mesh $M$ do

2: calculate ASRF $F_\Psi(\theta_v, \phi_v) = C_v(v, \Psi)$

3: rotate the most distorted vertex to the north pole

4: expand $F_\Psi(\theta, \phi)$ into a continuous form using spherical harmonics

5: $F_\Psi(\theta, \phi) = \min(\max(F_\Psi(\theta, \phi), t^{-0.5}), t^{0.5}) \times s$

6: for each vertex $v(\theta, \phi)$ in mesh $M$ do

7: redistribute colatitude $\theta_v = \theta'_v$ according to Equation (12)

To avoid introducing large additional length distortions, in each global smoothing step, we perform smoothing only for colatitudes $\theta$ using Equation (12). This is because, as mentioned before, the smoothing strategy for longitudes tend to cause substantial movements and thus introduce additional length distortions. Before the smoothing for colatitudes, we rotate the parameterization so that the vertex with the largest area distortion cost (see Equation (2)) becomes the north pole. Thus, this most distorted region gets the priority to be smoothed first. In addition, its distortion can be distributed towards all different directions on the surface, since this region is at the north pole and the smoothing procedure is designed entirely for colatitudes.

Based on the above discussion, we present our global smoothing approach in Algorithm 3. In Lines 1–2, we calculate the ASRF value for each mesh vertex in the current mapping $\Psi$. In Line 3, we rotate the most distorted vertex to the north pole. In Line 4, spherical harmonics basis functions [36] are used to expand and approximate the ASRF of $\Psi$ into a continuous form $F_\Psi$; and the smoothing operation based on Equation (12) can then be implemented by
performing numerical integration on $F\Psi$ at a later stage.

As mentioned above, large movements of colatitudes also tend to cause additional length distortions or even tangled mesh elements. Therefore, in Line 5, we modify $F\Psi$ in such a way that the ratio between its maximal value and minimal value does not exceed a certain threshold to avoid big movements. This is achieved by applying the following formula:

$$F\Psi(\theta, \phi) = \min(\max(F\Psi(\theta, \phi), t^{-0.5}), t^{0.5}) \times s,$$

where we set $t = 2$ in our experiments. In the above equation $s$ is a scaling factor, which can be calculated easily, to make sure that for a modified $F\Psi$ we still have

$$\int_0^{2\pi} \int_0^{2\pi} \frac{1}{F\Psi(\theta, \phi)} \sin \theta \, d\phi \, d\theta = 4\pi,$$

which guarantees that the south pole $\theta_{\text{south}} = \pi$ in an original mapping stays as the south pole in its reparameterization because of the relation

$$\theta'_{\text{south}} = f(\theta_{\text{south}}) = f(\pi) = \arccos(1 - \frac{1}{2\pi} \int_0^{2\pi} \int_0^{2\pi} \frac{1}{F\Psi(\theta, \phi)} \sin \theta \, d\phi \, d\theta) = \pi$$

implied by Equation (12). Note that Line 5 modifies $F\Psi(\theta, \phi)$ only if this is necessary, i.e., the value of $F\Psi(\theta, \phi)$ is either relatively large ($> t^{0.5}$) or relatively small ($< t^{-0.5}$). In fact, since our whole framework employs an iterative procedure, this modification does not happen at all in the last few iterations in our experiments, at which point any large distortion has already been removed by previous iterations. In Lines 6–7, we basically perform smoothing for colatitudes using Equation (12). Note that Algorithm 3 describes only one step of global smoothing. See next section for the whole framework that combines
both local and global smoothing steps.

2.4 The Framework

Our CALD framework starts from an initial parameterization and performs one step of global smoothing and \( n \) steps of local smoothing alternately until a stop criterion is achieved. We have tested different values of \( n \) in our experiments, and have found that performing \( n = 10 \) local smoothing steps at each iteration derives satisfactory results for the 3D models used in this paper.

Now let us look at the stop condition of this algorithm. Let \( M_R = \{e_i\} \) be a regular mesh of \( n \times n \) equal area elements on the unit sphere, where \( e_i \) is the element center. Fig. 6 shows a sample \( M_R \). Let \( F_{sh} \) be the spherical harmonic expansion of the ASRF. We define \( C_F \) as follows:

\[
C_F = \frac{\sum_{e_i \in M_R} \max(F_{sh}(\theta_{e_i}, \phi_{e_i}), \frac{1}{F_{sh}(\theta_{e_i}, \phi_{e_i})})}{n \times n}.
\]  

(16)

\( C_F \) measures the average distortion characterized by \( F_{sh} \), and it can have a minimum value of 1 when \( F_{sh} \) is 1 at every location on the sphere. The stop condition of our CALD algorithm is that \( C_F \) does not decrease for three consecutive iterations. This condition has been determined from our experimental study. In our experiments, if we keep running the algorithm after the above stop condition is met, the value of \( C_F \) will bounce slightly but won’t have a significant change.

In the experiments, we use the first 49 spherical harmonics (i.e., degrees up to 6) to get the spherical harmonic expansion \( F_{sh} \) of the ASRF. Note that \( F_{sh} \) is an approximation of the actual ASRF and captures mainly the low frequency
Fig. 6. (a) A regular mesh of $40 \times 40$ equal area elements on the unit sphere. (b) Its equal area projection on a 2D cartesian plane created by plotting $\cos \theta$ against $\phi$. This type of mesh is used to estimate $C_F$, the average distortion characterized by $F_{sh}$.

distortion signals over the sphere. Thus, $C_F$ derived from $F_{sh}$ is usually less than the overall ADC defined in Equation (3). It is observed that the global smoothing step seems to work effectively at removing these low frequency distortions, while the local smoothing step has the potential to further remove those high frequency distortions not captured by $F_{sh}$. Fig. 7 shows a sample smoothing procedure, which results in a moderately good parameterization with overall ADC $C_a = 1.106$ and LDC $C_s = 1.186$.

3 Results and Applications

The test objects used in our experiments include real and synthetic hippocampal structures created in our previous shape classification study [1,2], and 3D models downloaded from the OSU SAMP Lab [39] and the GNU Triangulated Surface Library [40]. For some models, surface holes are patched to get genus-zero meshes. Most of the models are also subdivided to get enough surface
Fig. 7. Sample smoothing procedure: (a) the top plot shows the object surface, and the bottom plot shows the initial parameterization; (b–f) the top plot shows the spherical harmonic expansion $F_{sh}$ of the ASRF after iteration 0, 5, 10, 25, or 39, and the bottom plot shows the corresponding parameterization. $C_F$ (average distortion characterized by $F_{sh}$), $C_a$ (overall ADC), and $C_s$ (overall LDC) are also provided for each parameterization.

We have tested about a hundred objects, with vertex numbers ranging from
600 to several thousand. The running times range from 30 seconds to 3.5 minutes on a Dell Optiplex GX260 Pentium 4 PC with a 2.4 GHZ CPU and 512 MB of RAM, which is running WinXP Professional OS and Matlab version 6.5. The initial parameterization step and the global smoothing step are implemented in Matlab. The local smoothing part is written by C, where a linear algebra package, LAPACK [41], is used for solving linear systems.

Fig. 8 shows some typical results, where the proposed CALD algorithm is used to create spherical parameterization for three models: (a) a crystal model, (b) a bowl model, and (c) a head model. In each case, we observe that the initial mapping is not desirable, but it can be greatly improved by the CALD algorithm to achieve a final result that has better controlled area and length distortions. Note that there are more mapping results available in Fig. 11.

Spherical parameterization is a very useful technique that can be applied in many different areas for processing 3D closed surfaces. In the rest of this section, we discuss several of them.

3.1 Statistical Shape Analysis

One application of spherical parameterization is to use the SPHARM expansion [19] to create a shape descriptor for a surface, where SPHARM coefficients can be manipulated to remove the effects of scaling, translation, and rotation. The result of spherical parameterization is a mapping of two spherical coordinates $\theta$ and $\phi$ to each point $v(\theta, \phi)$ on an object surface: $v(\theta, \phi) = (x(\theta, \phi), y(\theta, \phi), z(\theta, \phi))^T$. SPHARM expansion can then be used to expand the object surface into a complete set of SPHARM basis functions.
Fig. 8. Sample results of the CALD algorithm. In each of (a–c), the left plot shows the object surface, the middle plot shows the initial parameterization, and the right plot shows the final spherical parameterization. The area distortion cost $C_a$ and the length distortion cost $C_s$ are given for each parameterization. In each case, compared with the initial mapping, the improvement of the final mapping is obvious.
\( Y^m_l \), where \( Y^m_l \) denotes the spherical harmonic of degree \( l \) and order \( m \) \cite{19} for details. The expansion takes the following form:

\[
v(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} c^m_l Y^m_l(\theta, \phi),
\]

(17)

where \( c^m_l = (c^m_{x_l}, c^m_{y_l}, c^m_{z_l})^T \). The coefficients \( c^m_l \) up to a user-desired degree can be estimated by solving a set of linear equations in a least square fashion. The object surface can be reconstructed using these coefficients, and using more coefficients leads to a more detailed reconstruction; see Fig. 1 for a sample SPHARM expansion and reconstruction.

This SPHARM modeling technique has been applied in many applications including model-based segmentation \cite{3}, medial shape modeling \cite{4}, statistical shape analysis \cite{23}, and shape classification \cite{1,24}. However, previous studies require that an input object surface is defined by a square surface parameter mesh converted from an isotropic voxel representation. Now, using the new CALD algorithm, SPHARM shape analysis can be performed for general triangle surfaces. Actually, in volumetric image analysis, techniques like 3D reconstruction from 2D contours \cite{42}, isosurface extraction \cite{43}, and model-based segmentation \cite{3,20} have already been used to create general triangle surfaces. These surfaces may or may not have an underlying parameterization.

For surfaces without an underlying parameterization, the CALD algorithm can be applied to create one for them. For surfaces with an underlying parameterization, a further smoothing step is sometimes required to adjust the existing parameterization. For example, Kelemen, Szekely, and Gerig \cite{3} use Brechbühler’s approach \cite{19} to create a statistical shape model for a structure of interest and then deform the model surface to fit the structure in volumetric
images in order to perform automatic segmentation. The result is a deformed
triangle surface model, and its natural parameterization cannot guarantee the
area preserving property. The CALD algorithm can help improve the para-
meterization in this case, while Brechbühler’s approach does not apply due to
the lack of a voxel surface representation.

3.2 Similarity Search

SPHARM surface modeling can be used in similarity search. The mean squared
distance (MSD) between corresponding surface points can be used as the dis-
tance metric between two surfaces and calculated directly from SPHARM co-
efficients. Let \( S_1 \) and \( S_2 \) be two SPHARM surfaces, where their SPHARM coef-
cients are formed by \( c_{1,l}^m \) and \( c_{2,l}^m \), respectively, for \( 0 \leq l \leq \infty \) and \(-l \leq m \leq l\).
See Equation (17) for the definition of SPHARM coefficients. Thus, according
to [23], the MSD between \( S_1 \) and \( S_2 \) can be calculated as follows:

\[
MSD = \frac{1}{4\pi} \sum_{l=0}^{\infty} \sum_{m=-l}^{l} ||c_{1,l}^m - c_{2,l}^m||^2.
\]

This MSD can be used to measure the difference between two surfaces for the
purpose of similarity search. For general triangle surfaces, spherical parameter-
ization and the SPHARM expansion are performed to extract their SPHARM coefficients. To compare different surface shapes, SPHARM coefficients needs
to be normalized by removing the effects of scaling, rotation, and translation.
After that, MSD can be calculated to measure the difference between surfaces.

Fig. 9 provides such an example by showing a set of triangle surfaces, their
SPHARM reconstructions (degrees up to 15), and their MSDs by comparing
with the first surface. Note that these surfaces have already been normalized by
aligning their first degree ellipsoids to a canonical position in both parameter space and object space; see [1] for details about how to do the alignment. In this example, we can see that the MSD value captures some intuitions in terms of similarity measurement between two objects.

In SPHARM modeling, the underlying parameterization plays an important role, since it implicitly defines the surface correspondence between different models. An arbitrary parameterization cannot correctly establish the surface correspondence across objects and so won’t be able to derive correct MSDs among them. Our CALD framework derives a spherical parameterization with controlled area and length distortions. After this parameterization is rotated to a canonical position [19], the corresponding parts on two surfaces can be aligned together and their surface correspondence can be established successfully. Therefore, the CALD framework can help generate SPHARM models suitable for similarity search using MSD. This observation is also verified by our experimental results shown in Fig. 9, where the surface correspondence is indicated by the mesh superimposed on each SPHARM reconstruction.

3.3 Remeshing and Morphing

Remeshing is an important application of parameterization of 3D mesh data. In our case, once a spherical mapping is available, the original object can be remeshed to have a desired uniform structure.

Given an genus-zero surface mesh, its spherical mapping \( \Psi \) parameterizes each mesh vertex onto the unit sphere. The idea of remeshing is to employ the inverse mapping \( \Psi^{-1} \) and map a regular mesh on the sphere back to the
Fig. 9. The SPHARM expansion and similarity comparison. In each of (a–h), the top plot shows the original surface and the bottom plot shows its SPHARM reconstruction (using degrees up to 15); MSDs are calculated by comparing with the object in (a). The mesh on each SPHARM reconstruction indicates the surface correspondence across different objects.
object surface so that a remeshed geometry is constructed. This requires $\Psi$ to be a continuous form, and we must define the mapping within each triangle interior, since a vertex on the regular mesh is very likely to be located inside some spherical triangle.

Let $\{A_s, B_s, C_s\}$ on the sphere be the parameterization of a mesh triangle $\{A_o = \Psi^{-1}(A_s), B_o = \Psi^{-1}(B_s), C_o = \Psi^{-1}(C_s)\}$ on the object surface. Let $P_o$ be a point within the mesh triangle and $(\alpha, \beta, \gamma)$ be its Barycentric coordinates [44]:

$$P_o = \alpha A_o + \beta B_o + \gamma C_o \quad \text{with} \quad \alpha + \beta + \gamma = 1.$$

We use the Gnomonic projection approach described in [30] to define its parameterization

$$P_s = \frac{\alpha A_s + \beta B_s + \gamma C_s}{\|\alpha A_s + \beta B_s + \gamma C_s\|}.$$

Now we are ready to create the remeshed geometry using our spherical parameterization result. In our experiments, we always map a nearly uniform icosahedron subdivision at level 4 on the sphere back to an object surface to form a remeshed geometry. Please refer to Fig. 10 for similar icosahedron subdivisions at levels 0–3.
Fig. 11. Parameterization and remeshing results: In each row, the left plot shows the original object, the middle plot shows the parameterization, and the right plot shows the remeshed geometry. The area distortion cost $C_a$ and the length distortion cost $C_s$ are given for each parameterization. We observe that the final parameterization yields a much better remeshed geometry than the initial parameterization.
Both the initial parameterization and the final parameterization are tested in the remeshing experiments. Two sample results are given in Fig. 11. In either case, it is clear that the final parameterization produces a much better remeshed result, where the result not only preserves the shape of the original geometry but also remeshes it in a nearly uniform way. As to the initial parameterization, since it has relatively large area and length distortions, it either fails to capture some surface details (e.g., details around mouth, nose, and eyes in the first model) or produces relatively irregular mesh elements (e.g., the mesh elements with significantly varying sizes in the second model). Therefore, the remeshing quality depends on the quality of the underlying parameterization. The CALD algorithm can help create a good quality parameterization for remeshing.

Note that remeshed objects have the same mesh topology and can not only be compared with one another but also morphed to one another. Fig. 12 shows sample results of morphing one shape to another using remeshing, which is a useful application in computer graphics. Certainly, to create a good morphing result, the underlying parameterization used for creating the mutual tessellation should also have a good quality. Our CALD algorithm aims to construct such a good quality parameterization.

3.4 Other Applications

Spherical parameterization has many applications in different areas. After showing some of those in previous subsections, here we briefly discuss a few more. For example, based on spherical parameterization, *shape compression* can be performed using either spherical harmonics or spherical wavelets [45].
Texture mapping [31] is another application, where the key issue is to create a parameterization from a texture image to a 3D surface. If the texture image resides on a domain with spherical topology, spherical parameterization can be applied directly for texture mapping. In most cases, the texture image is located on a 2D rectangular region. Note that there are standard ways to unfold a spherical surface to a regular 2D grid [30]. Therefore, spherical parameterization can also help map a 2D texture image onto a 3D closed surface.

Creation of geometry images [46] is a new application of spherical parameterization, which is closely related to remeshing and texture mapping. In geometry images, geometry is resampled into a completely regular 2D grid. Each grid element stores the geometric information of the corresponding surface location according to a remeshing scheme, and the information could be $x$, $y$, and $z$ coordinate values, or other surface attributes such as color or normal. Clearly, given a genus-zero surface, its geometry images can be created by using a spherical parameterization and an unfolding mapping from a spherical surface to the regular 2D grid.

The advantage of having geometry images is that 3D geometric objects can be manipulated using standard image processing techniques applied to these
images. For example, 2D image wavelets on geometry images can be used to achieve 3D shape compression [30]; image registration techniques (e.g., [47–49]) become a potentially useful tool for registering 3D models via geometry images.

4 Conclusions

Area preserving spherical parameterization has been studied by Brechbühler et al. [19,21] for the surface of a voxel volume, but their approach is not applicable to general triangle meshes. Praun and Hoppe [30] have recently proposed a general spherical mapping method that minimizes length distortion, but without considering area distortion. Inspired by these studies, we have proposed a new algorithm CALD for mapping general 3D closed genus-zero surfaces onto a sphere that aims to control both area and length distortions.

A novel local mesh smoothing method based on solving a linear system is developed to reduce both area and length distortions at each local submesh. In addition, we capture the distribution of area distortion on the sphere using the concept of area scaling ratio function (ASRF) and develop a novel global smoothing method which aims to equalize the ASRF values over the whole sphere. The overall algorithm combines the local and global methods together and performs the two methods alternately until a solution is achieved.

The experimental study shows not only the effectiveness of our algorithm but also some useful applications. This technique can be used for processing of 3D closed surfaces in many areas, including computer vision, image processing, computer graphics, and multimedia databases.
Acknowledgements

This work is supported by NSF IDM 0083423. We thank Dr. Andrew Saykin at Dartmouth Brain Imaging Laboratory for providing us with the hippocampus data. We thank the anonymous reviewers for their insightful comments and suggestions that help improve the paper.

A Appendix A

In this Appendix, we provide the proof for Theorem 1, which is introduced in Section 2.3.2. First, let us use Fig. A.1 to give a conceptual example for the procedure described in Theorem 1:

(a) Initially, $R_v$ is a region on a spherical mapping $\Psi$.

(b) Step 1 changes the mapping from $\Psi$ to $\Psi_1$ by redistributing the colatitude using Equation (12), and $R_v$ on $\Psi$ becomes $R_{v1}$ on $\Psi_1$.

(c) Step 2 changes the mapping from $\Psi_1$ to $\Psi_2$ by redistributing the longitude using Equation (14), and $R_{v1}$ on $\Psi_1$ becomes $R_{v2}$ on $\Psi_2$.

(d) The corresponding part $R_{obj}$ on the object surface satisfies

$$R_{obj} = \Psi^{-1}(R_v) = \Psi_1^{-1}(R_{v1}) = \Psi_2^{-1}(R_{v2}).$$

The goal of this redistribution of spherical coordinates is to achieve $A(R_{v2}) = A(R_{obj})$.

The correctness of the above procedure is shown as follows. First, the current mapping $\Psi$ is assumed to be an effective parameterization: that is, $\Psi$ is a bijective mapping between the object surface and the surface of a unit sphere, and
Fig. A.1. Rearranging spherical coordinates for global smoothing: (a) region $R_v \equiv R(0, 0, \theta_v, \phi_v)$ on a spherical mapping $\Psi$; (b) Step 1 changes the mapping from $\Psi$ to $\Psi_1$ by redistributing the colatitude using Equation (12), and $R_v$ on $\Psi$ becomes $R_{v1}$ on $\Psi_1$; (c) Step 2 changes the mapping from $\Psi_1$ to $\Psi_2$ by redistributing the longitude using Equation (14), and $R_{v1}$ on $\Psi_1$ becomes $R_{v2}$ on $\Psi_2$; and (d) the corresponding part $R_{obj} = \Psi^{-1}(R_v) = \Psi_1^{-1}(R_{v1}) = \Psi_2^{-1}(R_{v2})$ on the object surface. The goal of the rearrangement of spherical coordinates is to achieve $A(R_{v2}) = A(R_{obj})$. 

43
its ASRF $F_\Psi$ is a continuous spherical function that satisfies $0 < F_\Psi(\theta, \phi) < \infty$ for every point $(\theta, \phi)$ on the sphere. Next, a few lemmas are introduced to show that both $\Psi_1$ and $\Psi_2$ are effective spherical parameterizations. Finally, the proof of Theorem 1 is given to show $A(R_{v2}) = A(R_{obj})$.

**Lemma 2** $\Psi_1$ is a valid spherical parametrization.

**PROOF.** We only need to consider if the mapping of colatitudes is correct, since there is no change for any longitude in Step 1. It is easy to see that $f(\theta_v)$ in Equation (12) is a monotonic function that defines a bijective mapping from $[0, \pi]$ to $[0, \pi]$ (i.e., the range of colatitudes), since we have

$$f(0) = \arccos(1 - \frac{1}{2\pi} \int_0^{2\pi} \int_0^\pi \frac{1}{F_\Psi(\theta, \phi)} \sin \theta \, d\phi \, d\theta)$$
$$= \arccos(1 - \frac{1}{2\pi} \times 0)$$
$$= 0$$

and, by Equation (11),

$$f(\pi) = \arccos(1 - \frac{1}{2\pi} \int_0^{2\pi} \int_0^\pi \frac{1}{F_\Psi(\theta, \phi)} \sin \theta \, d\phi \, d\theta)$$
$$= \arccos(1 - \frac{1}{2\pi} \times 4\pi)$$
$$= \pi.$$ 

This validates that Step 1 creates an effective spherical mapping $\Psi_1$. □

**Lemma 3** Given any region $R(0,0,\theta'_v,2\pi)$ on $\Psi_1$, it preserves the area of its corresponding part on the object surface: $A(R(0,0,\theta'_v,2\pi)) = A(\Psi_1^{-1}(R(0,0,\theta'_v,2\pi)))$.  

44
PROOF. By definition, we have

\[ A(R(0, 0, \theta'_v, 2\pi)) = \int_0^{\theta'_v} \int_0^{2\pi} \sin \theta \, d\phi \, d\theta = 2\pi (1 - \cos \theta'_v). \]

Replacing \( \theta'_v \) with \( f(\theta_v) \) described in Equation (12), we get

\[ A(R(0, 0, \theta'_v, 2\pi)) = 2\pi (1 - \cos f(\theta_v)) = 2\pi (1 - (1 - \frac{1}{2\pi} \int_0^{\theta'_v} \int_0^{2\pi} \frac{1}{F_\Psi(\theta, \phi)} \sin \theta \, d\phi \, d\theta)) = \int_0^{\theta_v} \int_0^{2\pi} \frac{1}{F_\Psi(\theta, \phi)} \sin \theta \, d\phi \, d\theta. \]

By Equation (10),

\[ A(\Psi^{-1}(R(0, 0, \theta_v, 2\pi))) = \int_0^{\theta_v} \int_0^{2\pi} \frac{1}{F_\Psi(\theta, \phi)} \sin \theta \, d\phi \, d\theta, \]

and thus

\[ A(R(0, 0, \theta'_v, 2\pi)) = A(\Psi^{-1}(R(0, 0, \theta_v, 2\pi))). \]

Since Step 1 and Equation (12) imply that \( \Psi^{-1}(R(0, 0, \theta_v, 2\pi)) \) and \( \Psi^{-1}_1(R(0, 0, \theta'_v, 2\pi)) \) refer to the same region on the object surface, we finally have

\[ A(R(0, 0, \theta'_v, 2\pi)) = A(\Psi^{-1}_1(R(0, 0, \theta'_v, 2\pi))). \]

Lemma 4 \( \Psi_2 \) is a valid spherical parametrization.

PROOF.

We only need to consider if the mapping of longitudes is correct, since there is no change for any colatitude in Step 2. It will be enough to show that, for any given fixed colatitude \( \theta'_v \in (0, \pi) \), \( g(\theta'_v, \phi_v) \) in Equation (14) is a monotonic
function of $\phi_v$ that defines a bijective mapping from $[0, 2\pi]$ to $[0, 2\pi]$ (i.e., the range of longitudes). Note that here we ignore the north pole ($\theta'_v = 0$) and the south pole ($\theta'_v = \pi$), since, as two degenerate cases, no matter what longitude values they take, they are always correctly mapped to themselves respectively.

By Lemma 2, $\Psi_1$ is an effective spherical parameterization. Thus, for any fixed $\theta'_v \in (0, \pi)$, it is easy to see that $g(\theta'_v, \phi_v)$ is a monotonic function of $\phi_v$ that defines a bijective mapping from $[0, 2\pi]$ to $[g(\theta'_v, 0), g(\theta'_v, 2\pi)]$. Now we just need to show that $g(\theta'_v, 0) = 0$ and $g(\theta'_v, 2\pi) = 2\pi$.

By definition from Equation (14), we have

$$g(\theta'_v, 0) = \int_0^\theta \frac{1}{F_{\Psi_1}(\theta'_v, \phi)} \ d\phi = 0.$$ 

Now let us examine $g(\theta'_v, 2\pi) \sin \theta'_v$. By definition, we have

$$g(\theta'_v, 2\pi) \sin \theta'_v = \frac{d}{dx} \int_0^x g(\theta, 2\pi) \sin \theta \ d\theta \bigg|_{x=\theta'_v}.$$ 

Expanding $g(\theta, 2\pi)$ using Equation (14), we get

$$g(\theta'_v, 2\pi) \sin \theta'_v = \frac{d}{dx} \int_0^{2\pi} \frac{1}{F_{\Psi_1}(\theta, \phi)} \ d\phi \sin \theta \ d\theta \bigg|_{x=\theta'_v}.$$ 

We observe from Equation (10) that

$$A(\Psi_1^{-1}(R(0, 0, x, 2\pi))) = \int_0^{2\pi} \frac{1}{F_{\Psi_1}(\theta, \phi)} \ d\phi \sin \theta \ d\theta,$$

and thus

$$g(\theta'_v, 2\pi) \sin \theta'_v = \frac{d}{dx} A(\Psi_1^{-1}(R(0, 0, x, 2\pi))) \bigg|_{x=\theta'_v}.$$
By Lemma 3,

\[ A(R(0, 0, x, 2\pi)) = A(\Psi_1^{-1}(R(0, 0, x, 2\pi))), \]

and so

\[
g(\theta'_v, 2\pi) \sin \theta'_v = \frac{d}{dx} A(R(0, 0, x, 2\pi)) \bigg|_{x=\theta'_v}
= \frac{d}{dx} \int_0^{2\pi} \int_0^\pi \sin \theta d\phi d\theta \bigg|_{x=\theta'_v}
= \frac{d}{dx} (2\pi(1 - \cos x)) \bigg|_{x=\theta'_v}
= 2\pi \sin \theta'_v.
\]

Thus, for any \( \theta'_v \in (0, \pi) \), we have \( \sin \theta'_v \neq 0 \) and so \( g(\theta'_v, 2\pi) = 2\pi \). This completes the proof. \( \square \)

Now we can give the proof of Theorem 1.

**Theorem 1** Let \( R_v \) be a region on a spherical mapping \( \Psi \). Change \( \Psi \) to \( \Psi_1 \) by redistributing every colatitude using Equation (12), and \( R_v \) on \( \Psi \) becomes \( R_{v1} \) on \( \Psi_1 \). Change \( \Psi_1 \) to \( \Psi_2 \) by redistributing every longitude using Equation (14), and \( R_{v1} \) on \( \Psi_1 \) becomes \( R_{v2} \) on \( \Psi_2 \). After the above procedure, \( R_{v2} \) on \( \Psi_2 \) preserves the area of its corresponding part on the object surface: \( A(R_{v2}) = A(\Psi_2^{-1}(R_{v2})). \)

**PROOF.** First, Lemma 2 and Lemma 4 guarantees that \( \Psi_1 \) and \( \Psi_2 \) are valid spherical parameterization respectively.
According to the described procedure (see Equation (12) and Equation (14)),
the region $R_{v2}$ is given by

$$ R_{v2} = \{(\theta, \phi) \mid 0 \leq \theta \leq \theta'_v, 0 \leq \phi \leq g(\theta, \phi_v)\} $$

where

$$ \theta'_v = \arccos \left(1 - \frac{1}{2\pi} \int_0^{2\pi} \int_0^1 F_{\Psi}(\theta, \phi) \sin \theta \, d\phi \, d\theta \right) $$

and

$$ g(\theta, \phi_v) = \int_0^{\phi_v} \frac{1}{F_{\Psi_1}(\theta, \phi)} \, d\phi. $$

Therefore, we have

$$ A(R_{v2}) = \int_0^{\theta'_v} \int_0^{\phi_v} \sin \theta \, d\phi \, d\theta $$

$$ = \int_0^{\theta'_v} g(\theta, \phi_v) \sin \theta \, d\theta $$

$$ = \int_0^{\theta'_v} \int_0^{\phi_v} \frac{1}{F_{\Psi_1}(\theta, \phi)} \, d\phi \, \sin \theta \, d\theta $$

We observe from Equation (10) and Equation (13) that

$$ A(\Psi_1^{-1}(R_{v1})) = \int_0^{\theta'_v} \int_0^{\phi_v} \frac{1}{F_{\Psi_1}(\theta, \phi)} \, d\phi \sin \theta \, d\theta. $$

Thus we have $A(R_{v2}) = A(\Psi_1^{-1}(R_{v1}))$. Based on the procedure, $\Psi_1^{-1}(R_{v1})$ and $\Psi_2^{-1}(R_{v2})$ refer to the same region on the object surface, and therefore we finally have

$$ A(R_{v2}) = A(\Psi_2^{-1}(R_{v2})). $$

□
References


