Fast Spherical Mapping for Genus-0 Meshes

Paper ID: 10
Category: Technical Paper
The 5th International Symposium on Visual Computing (ISCV09)
Las Vegas, Nevada, November 30 - December 2, 2009

Abstract Parameterizing a genus-0 mesh onto a unit sphere means assigning a 3D position on the unit sphere to each vertex of the mesh, such that the spherical mapping induced by the mesh connectivity is not too distorted and does not have overlapping areas. The non-overlapping requirement is technically the most difficult component also the most critical component of many spherical parametrization methods. In this paper we propose a fast spherical mapping approach that can map any closed genus-0 mesh onto a unit sphere without overlapping any part of the given mesh. The mapping process does not require setting up any linear systems, nor any expensive matrix computation, but is done simply by iteratively moving vertices of the given mesh locally until a desired spherical mapping is reached. Therefore the new spherical mapping approach is very fast and, consequently, can be used for meshes with large number of vertices. Moreover, the iterative process is guaranteed to converge. Another interesting phenomenon about this new approach is, it can generate meaningful results without considering the angle-preserving or edge-length-preserving constraint in the mapping process. Our apporach can be used for texture mapping, remeshing, 3D morphing and, more importantly, can be used as input for other more rigourous and expensive spherical parametrization methods to achieve more accurate parametrization results. Several test results are included to demonstrate the new apporach's capability in performing spherical mapping without any overlapping.

1 Introduction

Surface parameterization refers to the process of bijectively mapping the entire surface or a region of the surface onto a 2D plane, a 2D disk or a 3D sphere. Parameterization is a central issue and plays a fundamental role in computer graphics [19]. Parameterizing a 3D mesh means to compute a correspondence between a discrete surface patch and an isomorphic planar mesh through a piecewise linear function or mapping. Parameterization of 3D mesh data is important for many graphics applications, in particular for texture mapping, surface fitting, remeshing, morphing and many other applications. In practice, parameterization is obtained simply by assigning each mesh vertex a pair of coordinates (u, v) referring to its position in the planar region. Such a one-to-one mesh mapping provides a 2D flat parametric space, allowing one to perform any complex graphics application directly on the 2D flat domain rather than on the 3D curved surface [17].

Perfect parameterization of meshes of arbitrary topology currently is still unavailable [9]. Many parameterization techniques are only suitable for genus-0 meshes because of its simplicity and ubiquitous [9]. For example, meshes for almost all the animals are genus-0. Closed manifold genus-0 meshes are topologically equivalent to a sphere, hence a sphere is the natural parameter domain for them. Parameterization of a genus-0 mesh onto the sphere means assigning a 3D position on the unit sphere to each of the mesh vertices, such that the spherical mesh (having the same topology and connectivity of the orignal given mesh) is not too distorted and does not overlap. There are some techniques that can be used to achieve less mapping distortion, such as harmonic mappings and conformal mappings, which have been intensively studied recently and many nice methods have been proposed. However, for most spherical mapping methods, satisfying the non-overlapping requirement is still no gurantee, although this requirement is a critical component of any spherical mapping process.

Considering the fact that most parametrization methods are still very expensive and taking long time to achieve a desirable mapping [12, 18, 19], in this paper we describe a new iteraitve approach for fast spherical mapping of 3D models from a genus-0 mesh to a 3D unit sphere. The new method is easy to understand, easy to implement,

and can achieve relatively good mapping results. More importantly, our approach guarantees that the resulting mapping has no overlapping area on the sphere. The basic idea is to first project a closed genus-0 3D model into a unit sphere, then using subdivison techniques to smooth out the overlapping areas in the sphere. The projection and smoothing process do not require setting up any linear systems, nor any matrix computation, but is done simply by iteratively moving vertices of the genus-0 mesh locally until a meaningful mapping without any overlapping is reached. Therefore the new iterative method is very fast and consequently can be used for meshes with large number of vertices. Moreover, the iterative process is guaranteed to converge. The subdivision scheme considered in this paper is Catmull-Clark subdivision scheme [1]. But our approach works for other subdivision schemes as well, such as Loop subdivision [3] or Doo-Sabin subdivision scheme [2].

The remaining part of the paper is arranged as follows. A brief review of previous techniques on surface parametrization, spherical mapping, and subdivision surfaces is given in Section 2. The basic idea of our spherical mapping approach is presented in Section 3. The process of our fast spherical mapping technique based on Catmul-Clark subdivsion technique is discussed in Section 4. In Section 5, some test cases are shown and discussed. Concluding remarks and future work are presented in the last section.

2 Previous Work

2.1 Related work on 3D model parametrization/spherical mamping

Surface mapping or parametrization is a popular research topic recently and has been intensively studied in the last few years [9]. Many nice methods have been developed for genus-0 or arbitrary topology meshes with the former case being the focus, because for arbitrary genus meshes, a well-known parametrization approach can be used to somehow segment the mesh into disk-like patches such that each of the patches is genus-0. The challenge of this approach is to obtain mappings that are smooth across the patch boundaries and these methods, like the one presented in [17], suffered from this problem although recently Gu and Yau proposed a different method to compute parameterizations that are globally smooth with singularities occurring at only a few extraordinary vertices [18].

Among the many methods publiched on parametrizing genus-0 meshes, the majority is based on conformal mapping [9]. A conformal mapping is a mapping that preserves angles between edges on the mesh. Perfrect conformal mapping is difficult to achieve, hence most of the spherical mapping methods attempt to mimic conformal maps in order to preserve angles as much as possible by optimizing some constraints. In the following we briefly summarize the many related recent works on spherical parametrization using conformal mapping. Floater introduced a mesh parameterization technique based on convex combinations [10]. For all vertices, their 1-ring stencils are parameterized using a local parameterization space. The overall parameterization is obtained by solving a sparse linear system with constrains of preserving angles. Quicken et al. parametrized the surface of a voxel volume onto a sphere. Their nonlinear objective functional exploits the uniform quadrilateral structure of the voxel surface [11]. It tries to preserve areas and angles of surface elements. Desbrun et al. presented a method for computing conformal parameterization by minimizing the Dirichlet energy defined on triangle meshes [12]. Gortler et al. proposed a mesh parameterization approach using discrete 1-forms[14]. Their approach provided an interesting result in mesh parameterization, however it fails to control the curvatures on the surface boundaries. Sheffer et al. presented an angle-based flattening (ABF) approach, in which the changes of angles after flattening is minimized [15]. Grimm partitioned a surface into 6 pieces, and maps these to faces of a cube, and then to a sphere [13]. A priori chart of the surface partitions are used as constrain in the spherical parametrization process. Kharevych et al. [16] obtained lobally conformal parameterization by preserving intersection angles among circum-circles, each of which is derived from a triangle on the given mesh.

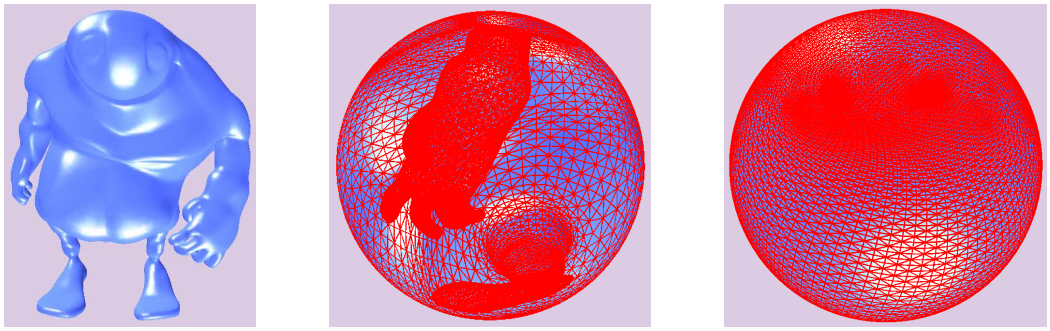
2.2 Subdivision Surfaces

Given a control mesh, a subdivision surface is generated by iteratively refining (subdividing) the control mesh to form new and finer control meshes. The refined control meshes converge to a limit surface called a *subdivision surface*. So a subdivision surface is determined by the given control mesh and the mesh refining (subdivision) process. The control mesh of a subdivision surface can contain vertices whose *valences* (numbers of adjacent edges) are different from four. Those vertices are called *extra-ordinary vertices*. The limit point of a vertex is the

point of the subdivision surface that is corresponding to the vertex. It is well known that all the limit points can be calucated directly from a given mesh. Popular subdivision surfaces include Catmull-Clark subdivision surfaces [1], Doo-Sabin subdivision surfaces [2] and Loop subdivision surfaces [3].

Subdivision surfaces can model/represent complex shape of arbitrary topology because there is no limit on the shape and topology of the control mesh of a subdivision surface. Subdivision surfaces are intrinsically discrete. Recently it was proved that subdivision surfaces can also be parametrized [7]. Therefore, subdivision surfaces cover both *parametric forms* and *discrete forms*. Parametric forms are good for design and representation, discrete forms are good for machining and tessellation (including FE mesh generation). Hence, we have a representation scheme that is good for all graphics and CAD/CAM applications. Subdivision surfaces by far are the most general surface representation scheme. They include non-uniform B-spline and NURBS surfaces as special cases [6]. In this paper we only consider objects represented by Catmull-Clark subdivision surfaces. But our approach works for other subdivision schemes as well.

3 Basic Idea



(a) original model (b) after projection (c) after smoothing

Figure 1: Spherical mapping using projection and smoothing.

Given a genus-0 mesh M with arbitrary topology, our task is to find another mesh Q such that the new mesh Q has the same topology and connectivity as the given mesh M, meanwhile all the vertices of the new mesh Q lie on a unit sphere. The basic idea to achieve this is through projection and smoothing. First all the vertices of the given mesh M are projected (with respect to the center of the unit sphere) to a unit sphere. After this step all the vertices are on a unit shpere. However, most likely there would be some overlapping areas on the unit shpere. Hence we need a second step to adjust the new mesh such that no overlapping areas exists on Q. The second step can be done with mesh smoothing techniques. Figure (1) shows the process of our spherical mapping approach. Figure (1(a)) is the given mesh M, Figure (1(b)) is the mesh after M is projected to a unit sphere. Figure (1(c)) is the mesh obtained after smoothing, which can be regared as a spherical mapping of the original mesh M because there is no overlapping in the sphere now.

The above process seems very simple and straightforward, but the implementation is very tricky. For example, in the first step, how to position the unit sphere so that the results of the projection process will make the next smoothing process easier and consequently achieve more meaningful spherical mapping? If the unit sphere is not positioned well, the given mesh could only be proejcted to part of the sphere and some wide areas of the unit sphere could be left empty without any vertices on it. For the second step, how to design an efficient smoothing approach in order to remove all the overlapping areas on the sphere is very important. The chosen smoothing method will determine the final quality of the spherical mapping results. In the following sections, we will present our approaches to the projection and smoothing process.

4 Spherical Mapping

4.1 Direct Calculation of the Limit Position of a Vertex

We need to calculate the limit positions of all the vertices in the spherical mapping process using any existing subdivision techniques. In our method, we choose to use general Catmull-Clark subdivsion scheme. But similar formulars can be derived for other subdivsion schemes as well. Bascially we need to find a matrix A, such that for a given mesh M, $\mathrm{A} * M$ is the mesh that all the vertices of $\mathrm{A} * M$ are the limit points of M. The matrix A is not needed in the implementation. The whole process can be done locally according to the matrix A. A can be constructed as follows.

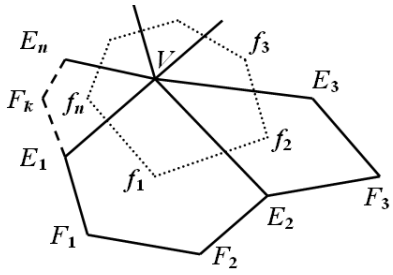


Figure 2: Vertex $\mathbf{V}$ and its neighboring vertices.

For generalized Catmull-Clark subdivision scheme, new face points and edge points are calculated the same way as the standard Catmull-Clark subdivision scheme, but the new vertex points are calculated differently, using the following formula

$$\mathbf{V}' = \frac{n-2}{n}\mathbf{V} + \frac{1}{n^2}\sum_{j=1}^{n}(\alpha\mathbf{V} + (1-\alpha)\mathbf{E}_j) + \frac{1}{n^2}\sum_{j=1}^{n}\mathbf{f}_j$$

where $0 \leq \alpha \leq 1$ and $\mathbf{f}_j$ are new face points after one subdivision [1]. When $\alpha = 0$, we get the standard Catmull-Clark subdivision scheme. The limit point [5] of a vertex $\mathbf{V}_i$ of degree n_i can be calculated as:

$$\mathbf{V}_i^{\infty} = \frac{1}{n_i(n_i+5)}(b_{ii}\mathbf{V}_i + \sum_j b_{ij}\mathbf{E}_j + \sum_j b_{ij}\mathbf{F}_j)$$

where

$$\begin{aligned}
&b_{ii} = (n_i - 1)n_i + n_i\alpha + \textstyle\sum \frac{4}{d_{ij}} \\
&b_{ij} = (2 - \alpha + \tfrac{4}{d_{ij}} + \tfrac{4}{d_{ji}}), \text{ if } (\mathbf{V}_\mathrm{i}, \mathbf{V}_\mathrm{j}) \text{ is an edge} \\
&b_{ij} = 4/d_{ij}, \text{ if } (\mathbf{V}_\mathrm{i}, \mathbf{V}_\mathrm{j}) \text{ is a diagonal line of a face} \\
&b_{ij} = 0, \text{ if } \mathbf{V}_\mathrm{i} \text{ and } \mathbf{V}_\mathrm{j} \text{ do not belong to the same face}
\end{aligned}$$

Note that in the above formula the surrounding faces could be not-four-sided (see figure 2). d_{ij} is the number of sides of the face of which $(\mathbf{V}_i, \mathbf{V}_j)$ is an edge or a diagonal line. Note that d_{ij} and d_{ji} could have different values because faces adjacent to the edge (V_i, V_j) could have different number of sides. But if $(\mathbf{V}_i, \mathbf{V}_j)$ is a diagonal line of a face, then $d_{ij} = d_{ji}$. According to the above definition, we have

$$\mathrm{A}_{ij} = \frac{1}{n_i(n_i+5)}b_{ij}.$$

It can be proven [20] that all the eigenvalues of A belong to $(0, 1]$. Therefore $\mathrm{A}^i * M$ is guaranteed to converg to a single point.

4.2 Mesh projection

The first step of our spherical mapping process is to project the given mesh to a unit sphere. The key of this step is to determine the center of the sphere because the projection is done with respect to the center of the

sphere. The chosen center of the sphere has to be located inside of the given mesh and has to be statistically same distance from all the vertices of the given mesh as much as possible so that after projection, there would be no much distortion. There are many choices of positioning the sphere. For example, average of all vertices, centroid of the mesh, or others. However, none of them would guarantee that the chosen point is inside the given mesh considering that the mesh could be of arbitrary shape. If not chosen properly, this step may fail or make the following steps of the spherical mapping process more difficult.

To make sure the chosen center is inside the given mesh and approximately close to all vertices, we need to smooth the given mesh $M_0 = M$. As we know subdivsion process is a smoothing process, hence we can use subdivision technique to smooth the mesh M_0. However, we do not want to introduce any new vertices in the process of subdivsion. To do this, we simply calculate the limit positions of all the vertices of the given mesh M_0 repeatedly using the matrix A given in the previous section as follows.

$$M_i = \mathrm{A} * M_{i-1}$$

In other words $M_i = \mathrm{A}^i * M_0$. Because all the eigenvalues of A belong to $(0, 1]$, eventualy, when $i \to \infty$, M_i becomes a single point [20]. Therefore M_i tends to be more like a sphere when i approaches ∞, which helps us better position the projection sphere. The process for calculating the limit points of a given mesh is very fast because it is a local operation. So it would not take much time to get a smoother version of the given mesh. To find a good candidate for the sphere center, we have to take the contribution of each vertex into consideration so that less distoration would be introduced in the projection process. In our implementation, instead of using the centroid of the mesh, we use weighted centroid of the mesh, which is calculated as follows.

$$C = \sum_{k}^{N} \frac{|V_k - G|}{\sum_j^N |V_j - G|} * V_k,$$

where G is the centoid of the mesh (which is the average of all the vertices), N is the total number of vertices in the given mesh and V_k and V_j are the vertices of the mesh M_i (Note that it is not the given mesh M). Once we have the sphere center C and M_i (in our implementation, we use M_{10}), we can project M_i to a unit sphere with respect to the center of the sphere.

4.3 Mesh smoothing

Once we have a desirable projection of the given mesh M to a unit sphere, next step is to remove the overlapping in the unit sphere. Again here we use subdivsion techniques to smooth out the overlapping areas of the new mesh Q_0 obtained from the projection process. For all vertices of mesh Q_0, we find their limit positions using the matrix A and then map them back to the unit sphere and then repeat the same procedure again and again as follows.

$$Q_i = P(\mathrm{A} * Q_{i-1}),$$

where P is the projection operator with respect to the chosen unit sphere center. Because Q_{i-1} lies on a unit sphere and matrix A is a smoothing operator, $(\mathrm{A} * Q_{i-1})$ has less overlapping areas than Q_{i-1}. So is Q_i. Therefore by repeating the process, Q_i will have less and less overlapping, and eventually it will have no overlapping areas at all. Because matrix A's eigenvalues are not bigger than one, the above process is guaranteed to be convergent. However, in order to remove all the overlapping areas in Q_i, i does not have to be infinity. Q_i will already be free of overlapping after some number of iterations of the above smoothing process.

The above process works fine theoretically. However, it convergs very slow. In order to remove all the overlapping areas, it takes a lot of iterations. To overcome this, we use the following two methods to speed up the spherical mesh smoothing process. The first one we can use is to offset the vertex positions of Q_i after each iteration as follows.

$$Q_i = u * (Q_i - Q_{i-1}) + Q_i,$$

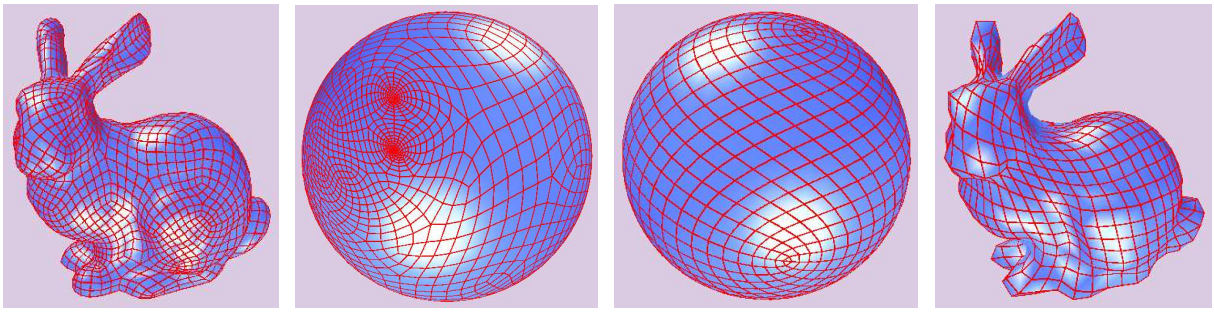
where u is an offset factor, which has to be chosen carefully in order to keep the iteration process stable. In our implementation, we choose $u = 0.25$. The second one we can use to speed up the smoothing process is to

adjust the vertex postions of Q_i after each iteration such that each vertex is locate at the centroid of its first ring surroudning triangles. The new location of a vertex V can be calculated as follows.

$$\bar{V} = \sum_{i}^{k} \frac{F_i * A_i}{\sum_i^k A_i},$$

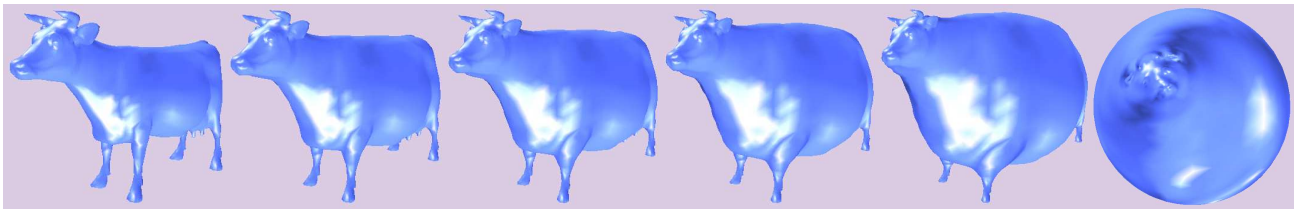
where k is the number of surrounding triangles, A_i is the area of the ith spherical triangle and F_i is the location of the centroid of the ith spherical triangle, which is the projection of the average of the vertex locations of the triangle. The area of a spherical triangle ABC on a unit sphere can be calculated explicitly as $(\angle A+\angle B+\angle C-\pi)$.

This above two methods will dramatically improve the convergence speed of the smoothing process, because for a vertex with overlapping surrounding triangles, the two methods will adjust the location of this vertex to a new location such that its surrounding trangles have much less overlapping areas. This is true because all the vertices are on a unit sphere.



(a) original model (b) spherical mapping (c) resampling (d) remeshing

Figure 3: Remeshing using spherical mapping.

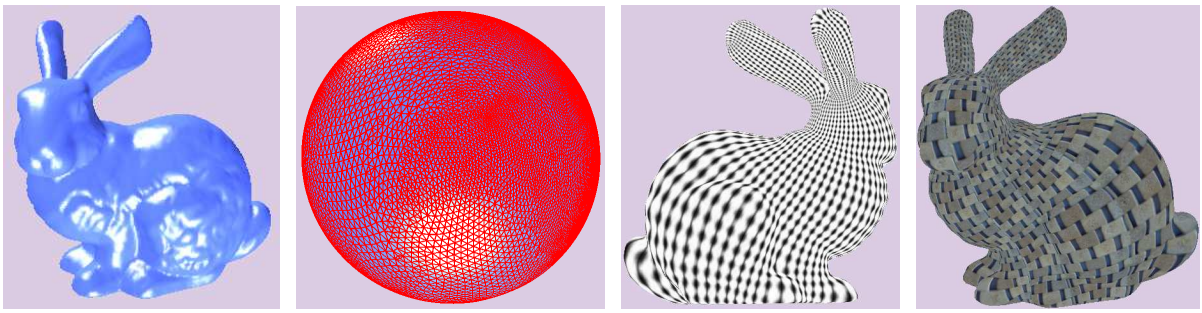


(a) Original Mesh (b) Morphing 1 (c) Morphing 2 (d) Morphing 3 (e) Morphing 4 (f) Morphing Final

Figure 4: Morphing of an object to a sphere

5 Test cases

The proposed approach has been implemented in *C++* using *OpenGL* as the supporting graphics system on the Windows platform. Quite a few examples have been tested with the method described here. All the examples have extra-ordinary vertices. Some of the tested cases are shown in Figures 3, 4 and 5 for remeshing, morphing and texture mapping applications, repectively. For the remeshing application given in Figure 3, first we get a spherical mapping using the approach presented in this paper. Then we use the method given by [19] to resample the sphere using an octahedron and to map it back to the original model to obtain a new mesh for the same model. For the mophing example given in Figure 4, it is simply a linear animation of the given model and the sphere obtained from the spherical mapping process. In the mesh sieres of the morphing process, some of the meshes could be extracted to be used as new meshes for other applications. So this provides a way to generate



(a) original model (b) spherical mapping (c) texture mapping (d) texture mapping

Figure 5: Texture mapping using spherical mapping.

new 3D models. For the texture mapping case given in Figure 5, once we have the spherical mapping, there are many ways to obtain the texture coordinates for each vertex. In this paper the method presented in [19] is used to get the uv texture coordinates. From all the test cases, we can see that even though our mapping process does not try to preserve angles or edge length, the spherical mapping results are still meaningful and desirable. We beleive some of the nice properties of subdision surfaces, such as convex hull property, contribute to this.

Figures 3, 4 and 5 demonstrate the capability of the new iterative method in mesh spherical mapping. It not only can smooth out all the overlapping areas, but maintains most of the geometric properties of the original mesh as well. The new spherical mapping method can handle meshes with large number of vertices in a matter of seconds on an ordinary laptop (2.16GHz CPU, 2GB of RAM). For example, the model shown in figures 5, has 35948 vertices and 69451 faces and, the model shown in figures 1, has 23202 vertices and 46400 faces. It takes 55 seconds and 31 seconds to obtain the spherical mapping shown in figure 5(b) and 1(c) respectively. For smaller meshes, like models shown in figures 3, and 4, the spherical mapping process is done even quicker. Hence our spheical mapping method is suitable for interactive applications, where simple shapes with small or medium-sized control vertex sets are usually used.

6 Summary and Future Work

A fast spherical mapping approach is presented which can map any closed genus-0 mesh onto a unit sphere without overlapping any part of the given mesh. The mapping process is straightforward and easy to implement. It is achieved through mesh projection and smoothing and relies heavily on subdivision techniques to locally adjust locations of vertices such that the mapping has less distortion and has no overlapping. Angle-preserving and edge-length-preserving are not considered in the new mapping approach. But test cases show that the new method can still obtain desirable results. The mapping process does not require setting up any linear systems, nor any expensive matrix computation, but is simply done by iteratively moving vertices of the given mesh locally until a desired spherical mapping is reached. Therefore the new spherical mapping approach is very fast and consequently can be used for meshes with large number of vertices. Moreover, the iterative process is guaranteed to converge. Our apporach can be used for texture mapping, remeshing, 3D morphing, and more importantly, can be used as input for other more rigourous and expensive spherical parametrization methods to achieve more accurate parametrization results. Some test results obtained using this method are included and they demonstrate that the new pporach can achieve spherical mapping results without any overlapping.

A disadvantage of our method is that it could fail for some cases. In some cases, one needs to tune some parameters in the implementation to achieve better results. One of our future research works is to investigate the reasons to improve our method. In addition, in order to get a desirable spherical mapping, the stop criterion of iteration used in our implementation is simply controlled by user iteraction. This is because we do not have a metric for measuring the quality of spherical mapping yet. We will work on it to obtain better quality of spherical mappings in the near future.

References

[1] Catmull E, Clark J, Recursively generated B-spline surfaces on arbitrary topological meshes, *Computer-Aided Design*, 1978, 10(6):350-355.

[2] Doo D, Sabin M, Behavior of recursive division surfaces near extraordinary points, Computer-Aided Design, 1978, 10(6):356-360.

[3] Loop CT, Smooth Subdivision Surfaces Based on Triangles, MS thesis, Department of Mathematics, University of Utah, August, 1987.

[4] DeRose T, Kass M, Truong T, Subdivision Surfaces in Character Animation, *Proceedings of SIGGRAPH*, 1998: 85-94.

[5] Halstead M, Kass M, DeRose T, Efficient, fair interpolation using Catmull-Clark surfaces, *ACM SIGGRAPH*, 1993:35-44.

[6] Sederberg T W, Zheng J, Sewell D, Sabin M, Non-uniform recursive subdivision surfaces, *Proceedings of SIGGRAPH*, 1998:19-24.

[7] Stam J, Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values, *Proceedings of SIGGRAPH* 1998:395-404.

[8] Adi Levin, Modified Subdivision Surfaces with Continuous Curvature, *Proceedings of SIGGRAPH*, 1035-1040, 2006.

[9] M. S. Floater and K. Hormann, Surface parameterization: a tutorial and survey. *Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization*, pages 157C186. Springer, 2005.

[10] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231C250, 1997.

[11] M. Quicken, C. Brechbühler, J. Hug, H. Blattmann, G. Szḱely, Parametrization of closed surfaces for parametric surface de-scription, *CVPR 2000*, pp. 354-360.

[12] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum (Proc. Eurographics 2002)*, 21(3):209C218, 2002.

[13] C. M. Grimm, Simple Manifolds for Surface Modeling and Parameterization, *Proceedings of the Shape Modeling International 2002*, p.277, May 17-22, 2002

[14] S. Gortler, C. Gotsman, and D. Thurston, Discrete oneforms on meshes and applications to 3D mesh parameterization. *Computer Aided Geometric Design*, v23 i2. 83-112, 2005.

[15] A. Sheffer, B. Levy, M. Mogilnitsky, and A. Bogomyakov. ABF++: Fast and robust angle based flattening. *ACM Transactions on Graphics*, 24(2):311C330, 2005.

[16] L. Kharevych, B. Springborn, and P. Schŕoder. Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(2), 2006.

[17] E. Praun, W. Sweldens, and P. Schr̈der. Consistent mesh parameterizations. In *Proceedings of SIGGRAPH 2001*, pages 179-184, 2001.

[18] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proceedings of the 1st Symposium on Geometry Processing*, pages 127-137, 2003.

[19] Spherical parametrization and remeshing. E. Praun, H. Hoppe. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3), 340-349.

[20] F. Cheng, F. Fan, S. Lai and etc., Loop Subdivision Surface based Progressive Interpolation, *Journal of Computer Science and Technology*, Vol.24, No.1, pp.39-46, Springer 2009.