# Geodesics-Based One-to-One Parameterization of 3D Triangle Meshes

**Haeyoung Lee**
*Hongik University, Korea*

**Yiying Tong and Mathieu Desbrun**
*California Institute of Technology*

**Digital geometry is a new data type for multimedia applications. To foster the use of 3D geometry, we introduce a piecewise linear parameterization of 3D surfaces that we can use for texture mapping, morphing, remeshing, and geometry imaging. Our method guarantees one-to-one mapping without foldovers in a geometrically intuitive way.**

Over the last five years, we've seen a growing interest in piecewise linear parameterizations of 3D surface meshes that guarantee one-to-one mapping without any foldovers. This particular property provides an elegant and robust solution to various problems in geometric modeling and computer graphics, such as texture mapping, morphing of triangulations,[1,2] remeshing,[3,4] or geometry images.[5] This parameterization type is crucial even for simple examples. For instance, Figure 1a shows artifacts in the tartan-fabric texture over the model because of foldovers in the parameter space.

As you can see, successful one-to-one mapping can make all the difference in how the final image looks. In the "Related Work" sidebar (on p. 29), we discuss the various solutions proposed so far, including the seminal work by Floater.

In this article, we improve upon Floater's original parameterization method[6] and propose a simpler one-to-one, shape-preserving parameterization technique for static meshes with improved numerical and visual quality. We demonstrate that, despite its simplicity, our technique further reduces the area distortion of the resulting parameterization without any computational overhead compared to Floater.[6,7] Furthermore, we provide a simple geometric interpretation of our new formula, involving local geodesics on the mesh and no local polar map. Our goal is to help ameliorate many applications that need guaranteed one-to-one mapping with little code modification.

## Shape-preserving parameterization

The most common one-to-one mapping used so far is undeniably the shape-preserving parameterization.[6] We'll first review this method, since our contribution proposes an extension to this previous work.

### Technique overview

As often with linear techniques, the general setup of the shape-preserving parameterization is the following: for each interior vertex $\mathrm{v}_i$ of a mesh, we will establish a linear relation between the $(u_i, v_i)$ coordinates of this point and the $(u_j, v_j)$ coordinates of its immediate neighbors $\{\mathrm{v}_j\}_{j\in N(i)}$, of the form

$$\sum_{j\in N(i)} a_{ij}\left(\mathrm{u}_j - \mathrm{u}_i\right) = 0 \qquad (1)$$

where $\mathrm{u}_i = (u_i, v_i)$ are the coordinates of vertex $\mathrm{v}_i$ in the parameter space, and $a_{ij}$ are the positive coefficients of a matrix $\boldsymbol{A}$. The boundary vertices are assigned to a fixed position (usually, the boundary is mapped to a square, a circle, or any convex shape while respecting the 3D-to-2D length ratio between adjacent boundary vertices). We then find parameterization by solving the resulting linear system $\boldsymbol{A}\mathrm{u} = \mathrm{b}$. Each line in the matrix $\boldsymbol{A}$ contains only a few nonzero elements (as many as the number of neighbors), therefore $\boldsymbol{A}$ is sparse. We use a preconditioned biconjugate gradient (PBCG) method[8] to iteratively solve this sparse linear system in an efficient manner. A simple diagonal preconditioner improves the convergence speed of the conjugate gradient solver, but more involved preconditioners may significantly reduce the overall computational cost. Notice that b is simply set to (0, 0) for inner vertices and to a fixed position $(u_i, v_i)$ for boundary vertices. As long as the boundary vertices are mapped to a convex shape, the resulting mapping is guaranteed to be one to one.

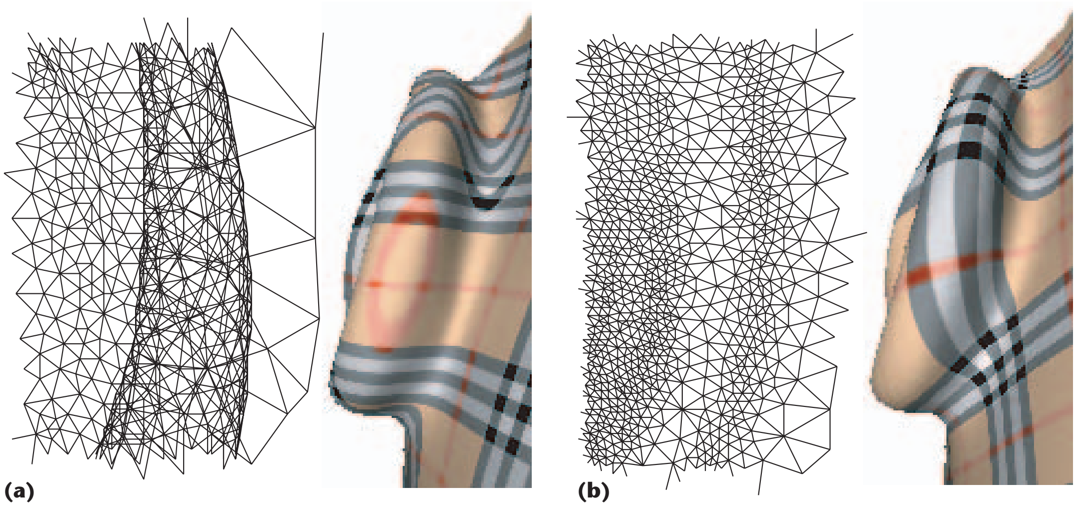


*Figure 1. (a) Foldovers in parameter space can easily happen if no special care is taken during parameterization of complex geometry (see the tartan-textured ear). (b) Instead, our technique guarantees a smooth mapping with no foldovers.*

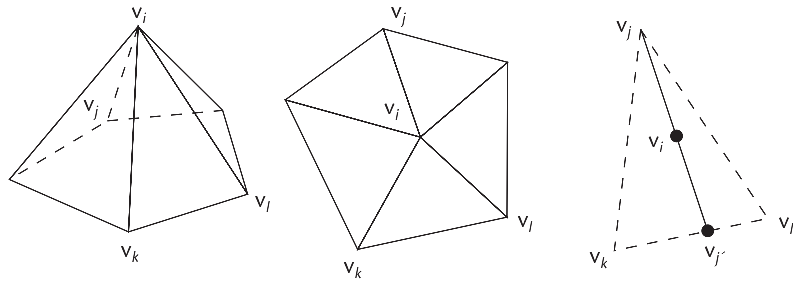


*Figure 2. Floater's shape-preserving method: Flatten first with the geodesic polar map, then calculate coefficients for each triangle formed by $v_j$, and the opposite edge $(v_k, v_l)$.*

The success of this shape-preserving mapping technique lies mainly in the procedure used to find nonnegative coefficients $a_{ij}$ that will result in a shape-preserving parameterization: constant coefficients like the ones proposed by Tutte[9] locally introduce large angle and length distortion instead. Next we'll review in detail how to find these coefficients based on the local mesh geometry.

## Finding nonnegative coefficients

Floater proposed to find these crucial nonnegative coefficients in a two-step procedure, applied for each interior vertex $v_i$ of the mesh:

1. Flatten the 1-ring around $v_i$.

2. On the flattened 1-ring, express the coordinates of $v_i$ as a convex combination of its neighbors' positions.

**1-ring flattening.** The first step uses a local geodesic polar map first introduced by Welch and Witkin.[10] The vertex $v_i$ and its neighbors are mapped to an isomorphic 1-ring in the 2D parameter plane so that the edge length is exactly preserved, and the angles between two consecutive edges are preserved up to a common factor (to account for the fact that the sum of the angles around $v_i$ on the original mesh may not be $2\pi$). This can easily be done incrementally, by selecting a first edge and constructing the flattened 1-ring by pivoting around the middle vertex while ensuring the desired properties. This local mapping may be regarded as approximately conformal, since it preserves the angle ratios and some local lengths as well. Now that the 1-ring has been flattened, convex combinations can be deduced to serve as nonnegative coefficients.

**Convex combination in flatland.** The second step proceeds as follows. On the flattened 1-ring, extend each of the $n$ edges emanating from the middle vertex until it crosses one of the boundary edges (see Figure 2). The edge $(v_j, v_i)$ will, once extended, cross an edge $(v_k, v_l)$. The middle vertex therefore belongs to the $n$ triangles defined by the triplets $(v_j, v_k, v_l)$. Floater proposed to simply compute the barycentric coordinates of $v_i$ in these triangles and average them all to get the final convex combination in Equation 1.

## Discussion

Finding the barycentric coordinates to position a vertex with respect to its immediate neighbors has been identified by many authors[1,6,11] as a way to obtain shape-preserving parameterizations, sometimes even minimizing a form of distortion.[12] With the previously described procedure, we always obtain nonnegative weights that produce very good results on fairly regular meshes as little shape distortion is introduced

## Related Work

Although there's much interest within the multimedia community in parameterizing 3D surface meshes, few have provided adequate one-to-one mapping solutions. Classical parameterization techniques based on varied approaches often lead to fast results that solve a simple linear system. While these approaches can be visually convincing, the absence of foldovers can't be guaranteed for an arbitrary genus-0 mesh (examples of these are available elsewhere[1-5]). We can design algorithms, however, that guarantee this property by construction, as Sheffer and de Struler[6] and Sander et al.[7] propose. Although such nonlinear techniques have been proven reliable and desirable in specific applications, they often require much more computational time than a mere linear solver. This overhead is rarely acceptable, and most researchers would like a faster solution.

We've also seen proposals of guaranteed one-to-one parameterizations that only require a linear solver. Tutte, in the context of graph drawing, proposed a method for guaranteed embedding of arbitrary triangulation.[8] The idea is fairly simple: consider each edge of the triangulation as a linear spring, and once the boundary vertices are fixed to a convex shape, find the resting position of the internal vertices within the network of springs. This technique, however, is sullied by the large distortion it creates when the result is used for texture mapping (see Figure A). Each mesh triangle gets mapped to a triangle with a totally different shape. In 1997, Floater proposed a similar technique, called *shape-preserving parameterization*,[9] with a stiffness for each spring that depends on the local geometry (we will review this method in detail). The distortion appears much reduced on regularly sampled geometry. Finally, Floater recently introduced a new mean value parameterization[10] with the important property of smooth dependency on the vertices.

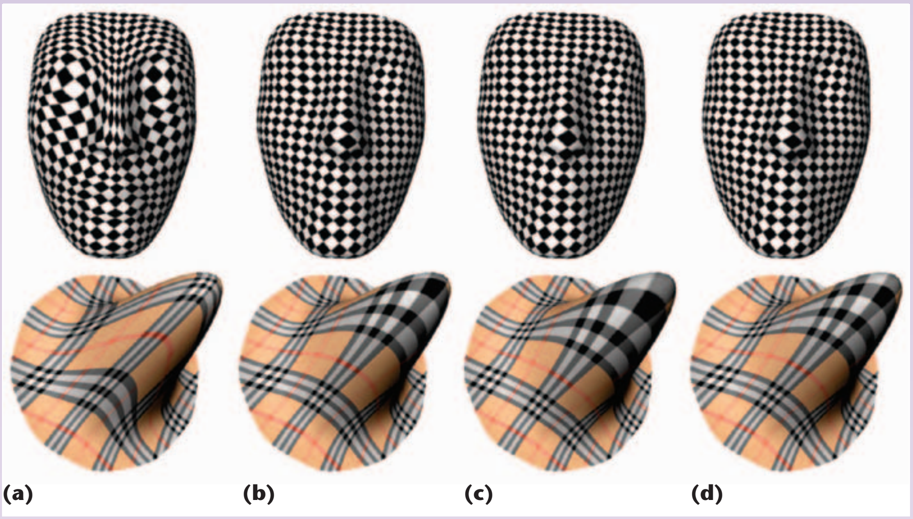


***Figure A. Comparisons with two models, Nefertiti and Hat1. From left to right, texture-mapped 3D surfaces by (1) Tutte,[8] (2) Floater,[9] (3) new Floater's,[10] and (4) our own new parameterization. For Nefertiti, Tutte's generates the biggest distortion but for Hat1, Floater's new method generates the biggest area distortion at the top of Hat1. Our method generates the lowest distortions for both models.***

### References

1. M. Eck et al., "Multiresolution Analysis of Arbitrary Meshes," *Proc. Siggraph*, ACM Press, 1995, pp. 173-182.
2. U. Pinkall and K. Polthier, "Computing Discrete Minimal Surfaces," *Experimental Mathematics*, vol. 2, no. 1, 1993, pp. 15-36.
3. S. Haker et al., "Conformal Surface Parameterization for Texture Mapping," *IEEE Trans. Visualization and Computer Graphics*, vol. 6, no. 2, 2000, pp. 181-189.
4. M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parameterizations of Surface Meshes," *Computer Graphics Forum (Eurographics)*, vol. 21, no. 3, 2002, pp. 209-218.
5. B. Lévy, S. Petitjean, and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *Proc. Siggraph*, ACM Press, 2002, pp. 362-371.
6. A. Sheffer and E. de Struler, "Surface Parameterization for Meshing by Triangulation Flattening," *Proc. 9th Int'l Meshing Roundtable*, Sandia Nat'l Laboratories, 2000, pp. 161-172.
7. P.V. Sander et al., "Texture-Mapping Progressive Meshes," *Proc. Siggraph*, ACM Press, 2001, pp. 409-416.
8. W.T. Tutte, "How to Draw a Graph," *Proc. London Math. Soc.*, vol. 13, no. 3, 1963, pp. 743-768.
9. M.S. Floater, "Parameterization and Smooth Approximation of Surface Triangulations," *Computer-Aided Geometric Design*, vol. 14, no. 3, 1997, pp. 231-250.
10. M.S. Floater, "Mean Value Coordinates," *Computer-Aided Geometric Design*, vol. 20, no. 1, 2003, pp. 19-27.

between the 3D and 2D triangles. However, using the geodesic polar map introduces unnecessary local distortion. If both edge lengths and angles are preserved and the Gaussian curvature is not $2\pi$, the boundary edges should not be straight, as Figure 3 (see p. 30) indicates. Therefore, the convex combinations rely on a linearized version of the exponential map, which makes the barycentric coordinates only valid in this distorted space. In the next section, we show that the flattening step is unnecessary and simplifying the procedure actually improves the mapping quality.

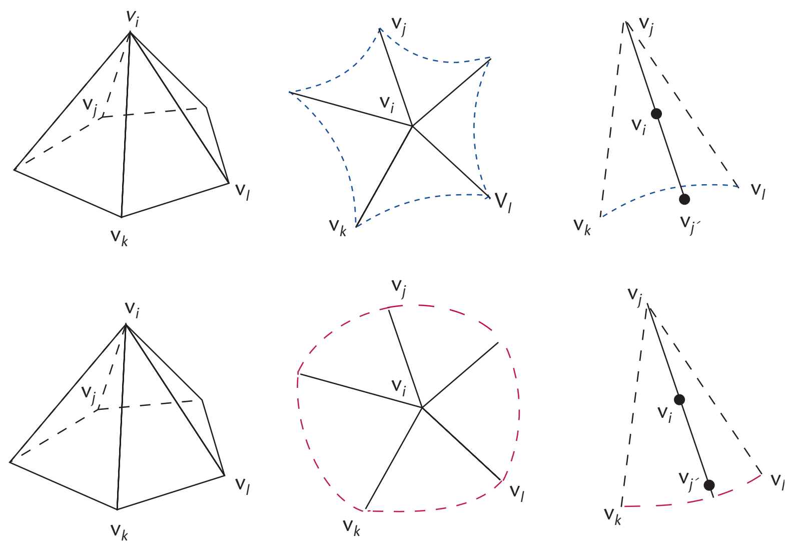
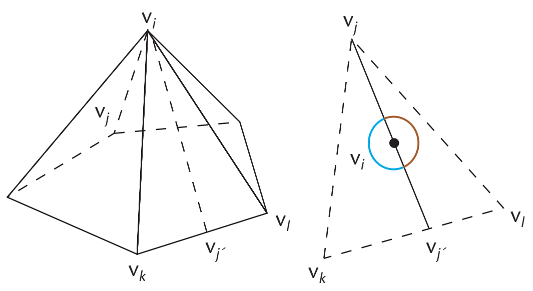


*Figure 3. Distortions in 1-ring flattening: If the Gaussian curvature at $v_i$ is less or greater than $2\pi$, the actual position of $v_{j'}$ is inside or outside of the triangle.*

*Figure 4. In our method, local straightest geodesics are used to determine the barycentric coordinates. The actual position of $v_{j'}$ on the edge ($v_k$, $v_l$) of the mesh is used for the barycentric coordinates.*



## One-to-one intrinsic parameterization

The original shape-preserving method uses a two-step procedure to compute convex combinations, potentially introducing distortion when the mesh has high curvature or isn't uniform. If we could use the same average of local barycentric coordinates, but this time compute directly on the mesh by an intrinsic property (such as local geodesic distances) the procedure would not only be much simpler, but would also result in less distortion. We propose a simple way to do this.

### Local straightest geodesics

The notion of straightest geodesics on triangulated surfaces was introduced by Polthier and Schmies,[13] and used to compute and visualize the evolution of wavefronts on discrete surfaces.[14,15] It turns out that Floater's method of extending an edge through the middle vertex (as we previously explained) exactly corresponds to tracing the straightest geodesic on the mesh. Because the polar map was only scaling the angles by a constant, the angles on each side of the line ($v_j$ to $v_{j'}$ in Figure 2) drawn by extending the edge are equal: the corresponding curve ($v_j$ to $v_{j'}$ in Figure 4) traced on the mesh is the straightest geodesic piecewise-linear curve. The difference is that this extended edge will cross an opposite edge of the surface mesh at a point that doesn't lie on the corresponding edge in the polar map, and therefore differs from the point found in Floater's method.

Now, we could hope to simply compute the same areas as in Floater's method, directly on the mesh. However, this isn't possible, as it would require computing two other piecewise linear straightest geodesics, namely between the $v_j$ and $v_k$ and between $v_j$ and $v_l$ on Figure 4. Such an operation would require too many computations, as these geodesics may even exceed the limits of the local 1-ring. Instead, we propose to use length-ratio-based barycentric coordinates that exactly match the well-known barycentric coordinates for a flat triangle, but are straightforward to compute on nonflat manifolds.

### Barycentric coordinates as length ratios

To avoid computing areas on a complex surface, we can rewrite the usual barycentric coordinates in a triangle as length ratios only. Given the construction of the straightest geodesics as we previously explained, we prefer to use the lengths of readily available geodesics, which will limit the number of computations. If we suppose that the triangle ($v_j$, $v_k$, $v_l$) in Figure 4 lies on the surface, we already have the segments [$v_j$ $v_i$], [$v_i$ $v_{j'}$], and [$v_k$ $v_l$] computed (the last one is an existing edge of the 1-ring). We can then propose a straightforward rewriting of the usual barycentric coordinates in this form:

### Multimedia and Digital Geometry

The multimedia revolution has mostly encompassed four types of data so far: text, sound, image, and video. The recent advent of 3D scanning has launched the next wave of multimedia data: geometry. Unlike previous data types, we can't rely on existing theoretical and algorithmic tools since geometric data has intrinsic properties, such as topology, curvature, and nonuniform sampling that render most traditional tools useless. Today, as geometry is used in many multimedia applications (such as virtual reality and video games), efficient, reliable, and affordable processing tools for 3D digital geometry (such as parameterization) are highly needed in online and offline applications.

$$
\begin{aligned}
w_j &= |v_i v_{j'}| / \left( |v_j v_i| + |v_i v_{j'}| \right) \\
w_k &= (1 - w_j) |v_l v_{j'}| / |v_k v_l| \\
w_l &= (1 - w_j) |v_k v_{j'}| / |v_k v_l|
\end{aligned}
\qquad (2)
$$

Because the same procedure is applied to calculate $w_k$ and $w_l$, we will only explain how to get $w_k$. As shown in Figure 5, $w_k$ is the ratio of the areas of triangle ($v_j$, $v_i$, $v_1$) and areas of triangle ($v_j$, $v_k$, $v_l$). This area ratio can be indirectly calculated with a length ratio as follows:

$$
\begin{aligned}
w_k &= \frac{\Delta(v_j v_i v_l)}{\Delta(v_j v_k v_l)} \\
&= \frac{\Delta(v_j v_{j'} v_l)}{\Delta(v_j v_k v_l)} \times \frac{\Delta(v_j v_i v_l)}{\Delta(v_j v_{j'} v_l)} \\
&= \frac{|v_l v_{j'}|}{|v_k v_l|} \times \frac{|v_j v_i|}{|v_j v_{j'}|} \\
&= \frac{|v_l v_{j'}|}{|v_k v_l|} \times (1 - w_j)
\end{aligned}
$$

Now, since all these lengths are already computed directly on the surface, these three weights (summing to one) are all we need to complete our technique, as recapped next.

### Finding nonnegative coefficients

In summary, the positive coefficients needed to set up the $i$th row of the linear system can be computed as follows.

For the $n$ 1-ring neighbors $v_j$ of an internal vertex $v_i$:

- Find the straightest geodesic from $v_i$ that extends the edge $v_j v_i$ and compute its first intersection $v_{j'}$ with the 1-ring neighborhood. This vertex $v_{j'}$ lies on an edge $v_k v_l$.
- Compute the weights $w_j$, $w_k$, $w_l$ for $v_j$, $v_k$, and $v_l$ as described in Equation 2.
- Sum all of the contributions (for example, weights from $n$ triangles) and divide by $n$ (to average them).

These coefficients, once computed, are used to fill up the matrix, exactly as in the original method since they're the coefficients $a_{ij}$ described in the "Technique overview" section. Once we've set up the sparse linear system by applying our algorithm for every internal vertex on the surface, we proceed as explained previously to obtain results (such as those in Figures 6 and A).

Notice that computing the straightest geodesics that extend an edge and find the first intersection with an opposite edge requires a few additional computations compared to finding the opposite edge in Floater's method, as described in more detail elsewhere.[14,15] However, because the barycentric coordinates are found only with length ratios instead of area computation in the original technique, the overall computation complexity is,

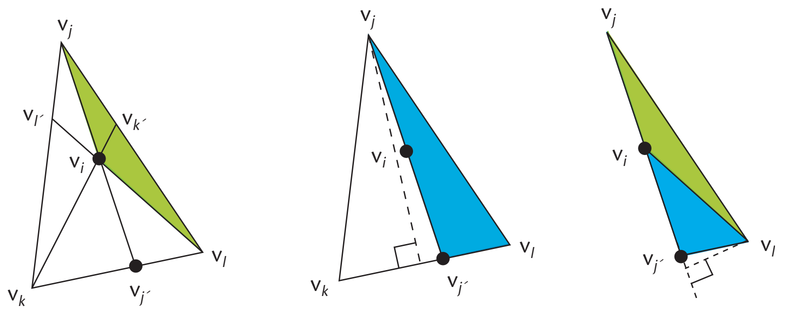


*Figure 5. How to get $w_k$: Instead of calculating the area $\Delta v_j v_i v_l$ or the geodesics $|v_k v_{k'}|$, we can use lengths—because if two triangles have the same height, the areas are determined by the lengths of the base lines.*

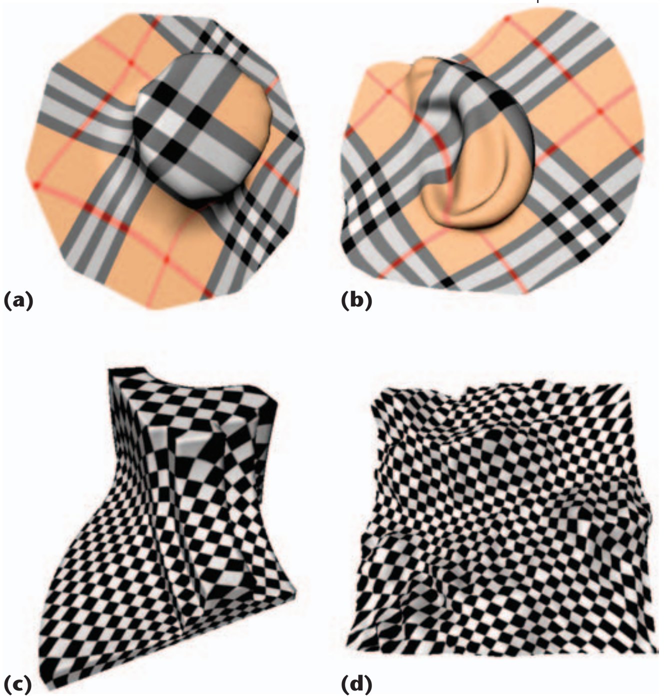


*Figure 6. Various meshes and their resulting parameterization using our technique: (a) Hat2, (b) ear, (c) top fan, and (d) mountain.*

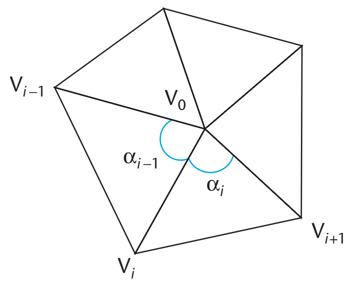


*Figure 7. In Floater's new method, coefficients are derived from the mean value theorem for harmonic functions.*

in our experience, reduced. We therefore have improved the original parameterization technique (both conceptually—with no projection needed—and practically) without noticeable overhead.

## Discussion

Our new shape-preserving parameterization combines the two steps of Floater's shape-preserving technique into one single step: no intermediate geodesic polar map is needed, since we compute all the coefficients directly in the identity map, using ratios of local geodesics. The positivity is guaranteed, and it still reduces to the same barycentric coordinates for any flat (or developable) 1-ring. Floater also presented a new method[7] for barycentric coordinates as a convex sum with neighboring vertices from the mean value theorem for harmonic functions. For a vertex $v_0$ with neighboring vertices $v_i$ in Figure 7, the following weights $\lambda_i$ are coefficients for $v_0$ with respect to $v_1 \ldots v_k$:

$$\lambda_i = \frac{w_i}{\sum_j w_j}, w_i = \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)}{\|v_i - v_0\|} \tag{3}$$

In short, Floater's shape-preserving and new mean-value methods as well as our own proceed in similar ways, and only differ on how they generate the coefficients $a_{ij}$ of the sparse matrix $\boldsymbol{A}$, marked as (∗) in the pseudocode:

```
mapBoundaryVertices;
    for each inner vertex $v_i$
        for each neighboring vertices $v_j$
            calculate $a_{ij}$ in the matrix $A$; (∗)
    perform a PBCG as described in the "Technique overview" section;
```

Note that the time complexity for the computations of the coefficients is negligible in practice when compared to the time required to solve the final linear system.

We tried Floater's newest method for various models and found that it often generates worse area distortion than either the previous one or ours (see the Hat1 model in Figure A). Although all three methods (shape-preserving, mean value coordinates, and ours) result in visually pleasing parameterization, area distortion as measured by distortion = (area3D – area2D)$^2$/area2D is consistently less with our technique than the original shape-preserving method, as Table 1 shows. In case of fine meshes with smooth gradation, our area distortions are also smaller than Floater's newer method of mean value coordinates.

In terms of computational efficiency, note that these results are obtained with no computational overhead: in fact, in our implementation, computing the new coefficients takes slightly less time (.07 second on average) than computing the others (.11 second on average for the original technique and .15 second for new Floater's). More importantly, we experienced improved convergence speed for our linear system (using the same conjugate gradient code for all three techniques): for a same-stopping criterion, our method converged in 1.40 seconds on average, while the original technique required 1.44 seconds and 1.78 seconds for Floater's newer method. Even if a study of the actual condition number of the linear system would be required to claim definite numerical superiority, these numbers indicate a good scalability for our method. CPU times were measured on a PC with a Pentium IV 2-GHz with 1Gbyte of RAM.

## Conclusion

We presented an improved method to compute one-to-one, shape-preserving parameterizations. It simplifies the original approach upon which it's

*Table 1. Distortions measured for models in Figures 7 and A.*

| Model | Number of Faces in Model | Our Method* | Shape-Preserving Method[6] | Mean Value Coordinates Method[7] |
|---|---|---|---|---|
| Ear | 8,471 | 0.2779 | 0.2782 | 0.2200 |
| Hat1 | 10,240 | 0.3078 | 0.3089 | 1.1294 |
| Nefertiti | 8,992 | 0.5313 | 0.5373 | 0.6448 |
| Top fan | 9,926 | 1.4050 | 1.4058 | 1.3148 |
| Mountain | 4,802 | 0.5495 | 0.5512 | 0.4827 |
| Hat2 | 2,560 | 0.7422 | 0.7485 | 1.3177 |

* Our method always produces fewer distortions than the original.

based and provides a simple, intuitive geometric meaning to the coefficients used in the linear system. Our technique can be qualified as intrinsic as we calculate the weights directly on the mesh, just like an (imaginary and flat) inhabitant of the mesh could. To our knowledge, the only drawback of this method is that the positive coefficients don't depend smoothly on the position of a vertex (as pointed out in Meyer et al.[11]); Floater's coefficients are only $C^0$, as are the ones for our method. This can potentially lead to visual artifacts if the mesh is animated, as the parameterization across the motion might not be smooth. However, for static surfaces (which were our original motivation), this doesn't impair the quality of parameterization at all, as we proved in all our examples.

We plan to concentrate our future research efforts on relaxing the boundary conditions to further reduce distortions. We also hope to work on parameterizations applicable to dynamic meshes, as they're increasingly used in multimedia applications. **MM**

## Acknowledgments

This work was supported in part by the Korean Science and Engineering Foundation (KOSEF-042010702) and the National Science Foundation (CCR-0133983, DMS-0221666, DMS-0221669, and EEC-9529152).

## References

1. M. Floater and C. Gotsman, "How to Morph Tilings Injectively," *J. Comp. Appl. Math.*, 1999, vol. 101, pp. 117-129.
2. V. Surazhsky and C. Gotsman, "High Quality Compatible Triangulations," *Engineering with Computers*, vol. 20, no. 2, 2004, pp. 147-156.
3. M. Eck et al., "Multiresolution Analysis of Arbitrary Meshes," *Proc. Siggraph*, ACM Press, 1995, pp. 173-182.
4. P. Alliez, M. Meyer, and M. Desbrun, "Interactive Geometry Remeshing," *Proc. Siggraph*, ACM Press, 2002, pp. 347-354.
5. X. Gu, S.J. Gortler, and H. Hoppe, "Geometry Images," *Proc. Siggraph*, 2002, ACM Press, pp. 355-361.
6. M.S. Floater, "Parameterization and Smooth Approximation of Surface Triangulations," *Computer-Aided Geometric Design*, vol. 14, no. 3, 1997, pp. 231-250.
7. M.S. Floater, "Mean Value Coordinates," *Computer-Aided Geometric Design*, vol. 20, no. 1, 2003, pp. 19-27.
8. W. Press et al., *Numerical Recipes in C*, 2nd ed., Cambridge Univ. Press, 1992.
9. W.T. Tutte, "How To Draw A Graph," *Proc. London Math. Soc.*, vol. 13, no. 3, 1963, pp. 743-768.
10. W. Welch and A. Witkin, "Free-Form Shape Design Using Triangulated Surfaces," *Proc. Siggraph*, ACM Press, 1994, pp. 247-256.
11. M. Meyer et al., "Generalized Barycentric Coordinates to Irregular *n*-gons," *J. Graphics Tools*, vol. 7, no. 1, 2002, pp. 13-22.
12. M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parameterizations of Surface Meshes," *Computer Graphics Forum (Eurographics)*, vol. 21, no. 3, 2002, pp. 209-218.
13. K. Polthier and M. Schmies, "Straightest Geodesics on Polyhedral Surfaces," *Math. Visualization*, Springer Verlag, 1998, pp. 135-150.
14. K. Polthier and M. Schmies, "Geodesic Flow on Polyhedral Surfaces," *Proc. Eurographics—IEEE Symp. Scientific Visualization*, Springer Verlag, 1999.
15. H. Lee et al., "Meshes on Fire," *Proc. Computer Animation and Simulation*, Eurographics, 2001, pp. 75-84.

**Haeyoung Lee** is a faculty member at Hongik University in Seoul, Korea. Her research interests include 3D mesh processing such as compression, parameterization, remeshing, and 3D animation. Lee received a PhD in computer science from the University of Southern California.

**Yiying Tong** is currently a PhD student in computer science at the University of Southern California. His research interests include computer graphics and discrete differential geometry. Tong received an MS from Zhejiang University, China.

**Mathieu Desbrun** is an associate professor at the California Institute of Technology. His research focuses on discrete geometry, particularly animation and simulation of 3D objects, processing of polygonal meshes, as well as theoretical work on the foundation of computations on discrete manifolds.

Readers may contact Haeyoung Lee at leeh@cs.hongik.ac.kr.