ACM Symposium on Solid Modeling and Applications (2004)
G. Elber, N. Patrikalakis, P. Brunet (Editors)

Planar Parameterization for Closed 2-Manifold Genus-1 Meshes

D. Steiner and A. Fischer

Laboratory for Computer Graphics and CAD, Faculty of Mechanical Engineering
Technion - Israel Institute of Technology, Haifa, Israel 32000

Abstract
Parameterization of 3D meshes is important for many graphics and CAD applications, in particular for texture mapping, re-meshing and morphing. Current parameterization methods for closed manifold genus-n meshes usually involve cutting the mesh according to the object generators, fixing the resulting boundary and then applying the 2D position for each of the mesh vertices on a plane, such that the flattened triangles are not too distorted and do not overlap. Unfortunately, fixing the boundary distorts the resulting parameterization, especially near the boundary. A special case is that of closed manifold genus-1 meshes that have two generators. They can therefore be flattened naturally to a plane without the use of a fixed boundary while still maintaining the continuity of the parameterization. Therefore, in treating genus-1 objects, this attribute must be exploited. This paper introduces a generalized method for planar parameterization of closed manifold genus-1 meshes. As in any planar parameterization with a fixed boundary, weights are assigned over the mesh edges. The type of weights defined depends on the type of mesh characteristics to be preserved. The paper proves that the method satisfies the non-overlapping requirement for any type of positive barycentric weights, including nonsymmetrical weights. Moreover, convergence is guaranteed according to the Gauss-Seidel method. The proposed method is simple to implement, fast and robust. The feasibility of the method will be demonstrated on several complex objects.

Categories: Planar parameterization, Genus-1 meshes, unfixed boundary.

1. Introduction

Parameterization of 3D meshes is important to many graphics and CAD applications, in particular for texture mapping, re-meshing and morphing. Until now, planar parameterization with fixed boundaries has primarily been considered [CSGL02] [DMA02] [DS95] [Flo97] [Flo03] [SDs01] [Tut63]. Planar parameterization of closed meshes that have been cut and flattened suffers from large distortions, especially on meshes whose genus is originally higher than zero. To cope with distortion from cutting the mesh and fixing its boundary, some works have considered spherical parameterization [Ale00] [GSG03] [GY02] [ST98] [SGD03] for genus-0 objects.

A recent work [GY00] has offered a solution for the open problem of conformal parameterization for manifolds with genus-n. In this solution, a gradient field is constructed over the mesh edges by solving a linear system that satisfies three conditions: (1) closedness; (2) harmonity and (3) conjugacy and duality. Closedness means that if the gradient field is integrated over an oriented loop such that it is homeomorpic to a disk, the integration result is zero. Harmonity means that the edges around a vertex must satisfy the Laplacian operator. Conjugacy and duality means that integration of the gradient field around the hole generator is equal to a predefined real number. The first and third conditions are approximated by summing the gradient field over the edges. The second condition, harmonity, is approximated by using the cotangent (harmonic) weights presented by [EH96]. Different weights can be used in place of the cotangent weights, but these weights must be symmetric. That is, in the case of an edge e_{ij} with a weight k_{ij} , k_{ij} must be equal to k_{ji}. Thus, the mean-value weights [Flo03] or the shape preserving weights [Flo97] cannot be used, nor can any other non-symmetric weight. Also, [GY00] uses the harmonic weights, indicating that the mesh triangles should not have obtuse angles. If the mesh includes triangles with obtuse angles, the mesh should be re-meshed without obtuse angles. An obtuse angle causes the harmonic weights to be negative. When negative weights are used, the process converges more slowly and the triangles might overlap.

A solution is given in [GSG03] for coping with the above problems of distortions near the boundary and continuity of the parameterization with respect to genus-0 objects. In this method, any positive weights for genus-0 meshes can be used. The result is a spherical parameterization, and the solution is achieved by solving a non-linear system.

A great deal of work has been devoted to finding a cut-graph for the class of genus-n meshes [DS95] [EH02] [Kar99] [LPVV01] [SF03] [SF02]. A cut graph is a connected graph containing 2*genus loops, also called generators [Hat00] [Mun84]. Cutting the mesh according to the cut graph yields a one-boundary mesh that is homeomorphic to a disk. The resulting one-boundary mesh can then be flattened using any planar parameterization method known today. The parameterization resulting from the above procedure has large distortions, especially near the boundaries. In this paper we offer a solution of the parameterization problem for genus-1 objects.

1.1. Contribution

A closed genus-1 manifold has exactly two generators. These two generators can be seen as the folding of a bounded plane to a torus. If we cut the mesh according to these generators and then flatten it, the resulting parameterization will suffer from large distortions, especially near the boundaries. Methods such as [DMA02] [SDs01] can be used to cope with the large distortion near the boundaries, but in such cases the continuity of the parameterization along the boundary cannot be controlled, nor can the characteristic of the mesh to be preserved in the parameterization.

This paper presents a parameterization method for genus-1 objects that is based on planar barycentric coordinates. The paper focuses on flattening of genus-1 objects onto the plane using any combination of barycentric weights [GSG03] while reducing the distortions over the boundaries and preserving the continuity of the parameterization. Moreover, a proof of the correctness of the method is provided. The proposed method can be solved using the Gauss-Seidel procedure. Hence, the solution of the linear system can be achieved by repeatedly updating each vertex value as a weighted average of its neighbors.

In our previous work [SF03], a method is given for finding the generators of genus-n objects. This method will be applied to find the generators of the genus-1 objects. The generators can also be found by using an algebraic method, such as the boundary operator and the Smith normal form [Mun84] [Kar99] [GY02]. In this paper the method in [SF03] was chosen because it is faster and does not involve mesh simplification. Moreover, it finds the types of the generators: the meridian (the generator around the body) and the longitude (the generator around the hole). We intend to use this property in the future to extend this work for parameterization of genus-n objects. In the next section we give a short preview of the method for finding generators described in [SF03] and of the barycentric coordinates method, which serve as a basis for the parameterization method for genus-1 meshes.

1.2. Finding the generators

In [SF03] a method is given for finding the two generators of each hole in a genus-n object. The method is based on an extension of the EdgeBreaker growing process presented in [LRSS02]. First, the mesh is traversed, while keeping a contour homotopic to a circle, also called the *active contour*. When the *active contour* touches itself, a split is generated. In case of a split, two contours are created. A merge indicates that a loop defined by the mesh has been closed. The two generators of each hole are found when a merge occurs. A merge occurs when an *active contour* touches another contour. That is, the *active contour* touches a contour created by the split that had created the current *active contour* or its primary father in the split lineage. While traversing the mesh, the algorithm adds faces, edges and points to the growing area, also called the *visited region* (Figure 1). The merge is handled in the following steps:

1. All generator edges found until this point (except the last one) are marked as *uncrossed*.
2. The *active contour* is selected as one of the generators.
3. The shortest path (using the Dijkstra algorithm) over the mesh from the *merge point* back to itself through the *visited region* is traced, yielding the second generator.
4. While the number of found generators < 2n where n is the genus, go to 1.

When an edge is set to an *uncrossed* edge, wave propagation from the merge point cannot pass this edge. Setting all generator edges found until now (except the last one) as *uncrossed* ensures that the new path will not define a generator that has already been found. The procedure for setting the *uncrossed* edges is similar to cutting the mesh using the generators (found until now) as cutting curves. Thus, the new path cannot pass through a cut edge and therefore cannot construct a generator that has been already found. If the genus is one, only steps 2,3 should be performed.

1.3. The method of weighted barycentric coordinates

This section gives a brief description of the weighted barycentric coordinates method. A full description can be found in [GSG03]. The basic idea of the weighted barycentric coordinate method is to fix the boundary of a manifold mesh with one boundary on a convex polygon and then repeatedly update each internal vertex as the weighted average of its neighbors. It was proven in [Flo97], which is a generalization of a proof given by Tutte [Tut63], that the resulting parameterization has no foldovers when using positive weights and convex boundary. The method of barycentric coordinates can be formulated as the solution to the 2D vector Laplace equation on the interior vertices, as given in Eq. 1, where L_w matrix is the Laplace matrix, X is the vector of variables and b is the vector of solutions. It has no zero entries due to the boundary conditions derived from the convex boundary. We will define L_w= (I-W), where I is the unity matrix and W are the weights matrix.

$$L_w X = b \qquad (1)$$

The following process describes the construction of W and b (Eq. 1) according to [GSG03]:

1. To each entry (i,j) of W that has a corresponding edge e_{ij}, assign a positive weight w_{ij} such that $\sum_{j\in N(i)} w_{ij} = 1$, where N(i) is the list of vertices neighboring the ith vertex.
2. To all other entries (i,j) of W, assign w_{ij}=0.
3. Embed the boundary vertices in the plane such that they form a closed convex polygon.
4. Solve the following linear systems for the x and y coordinates of the n interior vertices: $(I-W)x=b_x,(I-W)y=b_y$,
   where W is an n x n matrix containing w_{ij} and L_w=(I-W). b_x and b_y are vectors with non-zero entries corresponding to the vertices adjacent to the boundary.

The matrix L_w is a weakly dominated matrix and is therefore non-singular. The vector b has entries that are non-zero; therefore, this linear system has a unique solution. Due to the weakly dominated property of the L_w matrix, this linear system can be solved using the Gauss-Seidle procedure and is guaranteed to converge for any initial guess of X. The normalized Laplacian L_w has a unit diagonal, with negative entries at each mesh edge, and zero otherwise. Because the sum of all weights, w_{ij} around vertex v_i is one, all rows of L_w sum to zero, so that the matrix is singular. The rank of L_w is (n-1), where the matrix size is n x n. By fixing one vertex in the parameterization plane, the matrix becomes non-singular. The weakly dominated matrix is defined as follows:

$\sum_{\substack{j=1\\ j\neq i}}^{n} |C_{ij}| \le |C_{ii}|$ holds for all the matrix rows, and the inequality itself must hold at least once, where C_{ij} represents the i,j cell of a matrix. The weakly dominated matrix is known as a non-singular matrix. This non-singularity can be proved using the Gauss elimination procedure.

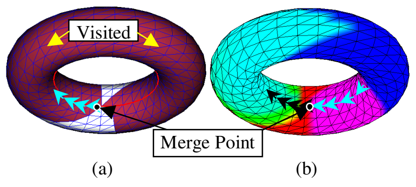

Figure 1: *Constructing the complementary generator when merge occurs: (a) visited region (in eggplant purple) grows until merge occurs, (b) color map of Dijkstra values over the visited region.*

2. The planar parameterization for genus-1 meshes.

The input to our algorithm is a closed genus-1 manifold mesh without boundaries and two generators, such that the generators intersect only once. Instead of cutting the mesh according to the generators and then fixing its boundary on a convex polygon, as is customary, we fix only one vertex and replace the fixed boundary conditions by cyclic boundary conditions. By fixing one vertex we ensure that L_w is non-singular. By applying the cyclic boundary conditions we ensure that the solution will not converge to the fixed vertex. The linear system can then be solved using the Gauss-Seidel procedure. Convergence of the Gauss-Seidel procedure is guaranteed for any initial guess of the solution X (Eq. 1). The proof for the convergence is given in Appendix A. In section 2.1, the cyclic boundary conditions are defined. Section 2.2 provides the proof that there is at most one foldover caused by the fixed vertex, which can be handled very simply. In section 3 the implementation of the method is given. Section 4 provides examples that demonstrate the feasibility of the method, and section 5 includes a summary and conclusions.

2.1. The cyclic boundary condition

Applying cyclic boundary conditions in the parameterization process preserves the continuity of the parameterization over boundaries. The generators are directed and marked as *a* and *b*. To set the cyclic boundary conditions, the algorithm first traverses the generators and marks all vertices that are neighbors and are on the right side of the generators as Nr(a) or Nr(b). Neighbor vertices of the generators will be marked as $N_g(a)$ or $N_g(b)$. During the traverse each edge of the generator is visited only once. Calculation of each vertex v_i position during each iterative step is given as follows:

$$v_{i_x} = \sum_{j=1..|N(i)|} \frac{W_{ij}\left(v_{j_x} + f_x(i,j)\right)}{\sum_{j=1..|N(i)|} W_{ij}} \qquad (2)$$

$$v_{i_y} = \sum_{j=1..|N(i)|} \frac{W_{ij}\left(v_{j_y} + f_y(i,j)\right)}{\sum_{j=1..|N(i)|} W_{ij}}$$

where

$$f_x(i,j) = \begin{cases} 0 & v_i \notin a \wedge v_i \notin N_g(a) \\ 2\pi & v_i \in a \wedge v_j \notin Nr(a) \\ -2\pi & v_i \in Nr(a) \wedge v_j \in a \end{cases} \qquad (3)$$

$$f_y(i,j) = \begin{cases} 0 & v_i \notin b \wedge v_i \notin N_g(b) \\ 2\pi & v_i \in b \wedge v_j \notin Nr(b) \\ -2\pi & v_i \in Nr(b) \wedge v_j \in b \end{cases} .$$

and W_{ij} is a positive weight calculated for edge e_{ij}. The the w_{ij} is given as:

$$w_{ij} = \frac{W_{ij}}{\sum_{j=1..|N(i)|} W_{ij}} \qquad (4)$$

v_{i_x} is given as:

$$v_{i_x} = \sum_{j=1..|N(i)|} \left(w_{ij} v_{j_x} + w_{ij} f_x(i,j)\right) \qquad (5)$$

By separating the sigma on the right side we get:

$$v_{i_x} = \sum_{j=1..|N(i)|} w_{ij} v_{j_x} + \sum_{j=1..|N(i)|} w_{ij} f_x(i,j) \qquad (6)$$

and by moving the neighbors' weighted average to the left side of the equation we get:

$$v_{i_x} - \sum_{j=1..|N(i)|} w_{ij} v_{j_x} = \sum_{j=1..|N(i)|} w_{ij} f_x(i,j) \qquad (7)$$

The same process can be applied on v_{i_y} .Since the connectivity of the mesh does not change during the process, the neighbors of *a* and *b* do not change their connectivity relations. Therefore, the right side of Eq. 7 is a constant.

We can see that on the left side of Eq. 7 we have L_w such that:

$$L_w(i,j) = \begin{cases} -w_{ij} & \forall i \neq j : \text{edge}(i,j) \in M \\ 0 & \forall i \neq j : \text{edge}(i,j) \notin M \\ 1 & i = j \end{cases}$$

where M is the mesh. The right side of Eq. 7 defines the entries on the right side vector of Eq. 1. To make L_w non-singular, we fixed the crossing vertex between the generators to be at $(2\pi, 2\pi)$. In Eqs. 2-7 we described how to replace the fixed boundary condition with cyclic boundary conditions. As a result, continuity between the two sides of the imaginary boundary created by the generators is preserved.

2.2. Parameterization without foldovers

Due to the definition of the L_w matrix and the proof for the convergence of the linear system (Appendix A), all vertices except the fixed vertex v_{fix} must be equal to the weighted average of their neighboring vertices. We will refer to the first neighboring vertices of v_{fix} as 1-neighbors. At first we prove that all vertices except v_{fix} and its 1-neighbors vertices cannot cause foldovers. Then, we relate to v_{fix} and its 1-neighbors.

2.2.1. No foldovers in vertices that are not 1-neighbors. First we consider all vertices except v_{fix} and its 1-neighbors vertices. We divide all edges on the boundary of the fan around vertex v_i into three classes: Edge-A, Edge-B, Edge-C. Class Edge-A contains all the edges that are on the convex hull of the neighboring vertices of vertex v_i (Figure 2.a, edge e3); class Edge-B contains all the edges that have concave vertex v_c at their ends (Figure 2.a, edge e4); class Edge-C contains all the edges that have two convex vertices at their ends but are not on the convex hull (Figure 2.d, edge e5). Those three classes contain all the edges on the boundary of a fan around vertex v_i. Therefore, there cannot be foldover because each of the following cases contradicts the assumption that all weights are positive and that each vertex is equal to the weighted average of its neighbors.

1. If a vertex v_i creates a foldover through an edge belonging to class Edge-A (Figure 2.b), the penetrated edge is a hyperplane H (Figure 2.b) dividing the plane into two sides. One side of the plane contains all the neighbors of v_i, while v_i is on the other side.
2. If v_i penetrates an edge of class Edge-B (Figure 2.c), a hyperplane H can be found such that all neighbors of v_c are placed on one side and v_c is on the other side (Figure 2.c).
3. If v_i penetrates an edge of class Edge-C, a concave vertex v_c (Figure 2.d) must be found as well as a hyperplane H such that all neighbors of v_c are placed on one side and v_c is on its other side (Figure 2.e).

If a vertex v_c cannot be found, the penetrated edge must then be on the convex hull. That is, it belongs to class Edge-A, which we have already proven cannot be penetrated.

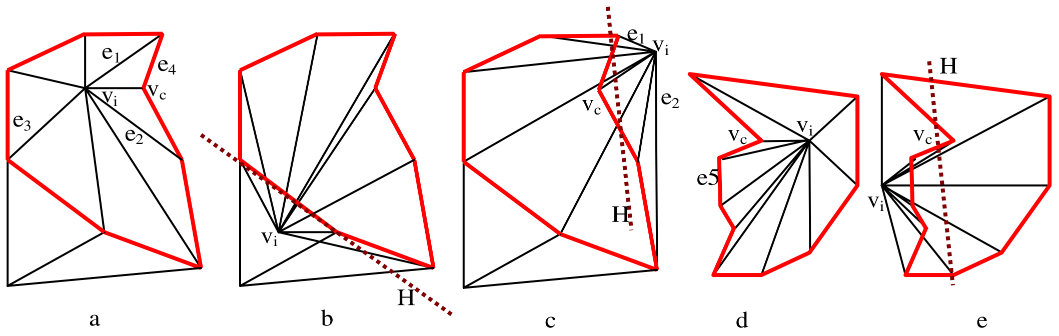

Figure 2: *Edge states: (a) The boundary of the fan around v_i, (b) penetrating a convex-hull edge, (c) penetrating an edge with concave vertex, (d) a boundary of a fan with convex region which is not on the convex-hull, (e) penetrating an edge of a convex region which is not on the convex-hull.*

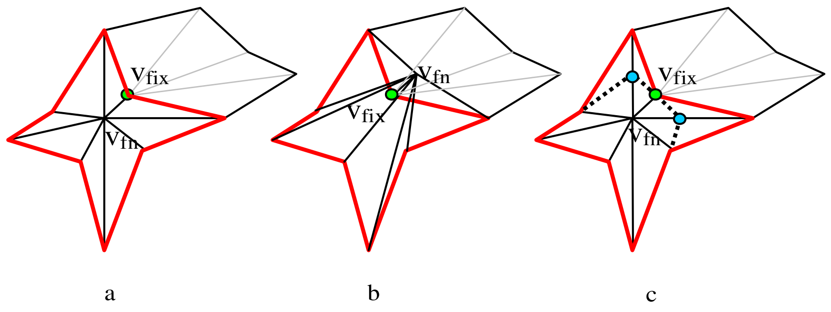

Figure 3: *Fixed vertex, overload state: (a) v_{fix} on a concave region of the fan boundary, (b) penetrating an edge belongs to v_{fix}, (c) subdivision of v_{fix} fan.*

2.2.2. Foldovers in the fixed vertex and its 1-neighbors vertices. This section considers the 1-neighbors vertices of the fixed vertex v_{fix}. A vertex that is a 1-neighbor of v_{fix} will be referred to as v_{fn}. v_{fn} cannot penetrate an edge e_{ij} such that v_i is a vertex neighboring to v_{fix} and v_j is not v_{fix}. This contradicts the first part of the proof given in section 2.2.1. Thus, the only possible way for foldover to happen is when v_{fix} holds a concave angle on one of the fan boundaries of its neighbors, as seen in Figure 3.a. If v_{fix} holds a convex angle relative to the fan boundary around v_{fn}, the edges connected to v_{fix} construct a convex region, and as proven in section 2.2.1 v_{fix} cannot cause foldovers. When v_{fix} holds a concave angle on the v_{fn} fan boundary, the resulting weighted average calculation for v_{fn} can be outside the region of its fan, as can be seen in Figure 3.b. Due to the fact that v_{fix} is fixed, this (Figure 3.b) can be a possible solution of the linear system. Then, if v_{fix} holds an obtuse angle, there is a possibility that foldover will happen. In practice on all the modules with flattened using cyclic boundary conditions, foldover did not occur. A simple solution that solves this problem is to subdivide the two overlapped triangles connected to v_{fix}, as shown in Figure 3.c. Further details are given in the implementation section, Section 3.

3. Implementation

The implementation complexity of the proposed method is similar to the complexity of the process of flattening a mesh using a fixed boundary. The steps are:

1. Positive weight is calculated for each edge.
2. The fixed vertex values are determined.
3. All vertices on the right of the generators are marked.
4. The position of each vertex except the fixed one is calculated to be the weighted average of its neighbors, while considering the neighboring vertices type and the current vertex type.

In the following we will explain Steps 2-4 and how to handle the only foldover that might occur.

3.1. Fixing the crossing vertex values

The crossing vertex is the vertex belonging to both generators. If the solution of the linear system demands $L_{w,}$ we set all entries of the crossing vertex row in L_w to zero except the diagonal, which is set to one. If the system is iteratively solved by setting each vertex value to the weighted average values of its neighbors, then we simply set its values ones and then ignore it throughout the iteration process.

3.2. Marking right-side vertices of the generators

The right sides of the generators are marked using a simple procedure based on the following steps:

1. Start from the crossing vertex and traverse all generator-oriented edges only once. While traversing the edges, mark each face that lies on an oriented edge and push those faces into a list.
2. Pop a face from the list and mark its neighboring faces that are not marked and that have a vertex on the generators. Then, push the face to the list.
3. While the list is not empty, return to 2.

While marking the right side of the generators, each marked vertex should be signed with the generator that marked it. Also, a vertex can be marked by both generators.

3.3. Weighted average calculation

To calculate the weighted average of each vertex v_i we will consider three types of vertices: a type VA vertex is a vertex that is not on the generators and not on the right side of a generator; a type VB vertex is a vertex that is on the generator; a type VC vertex is a vertex that is on the right side of the generator. If v_i is of type VA, Eq. 3 should be used while setting f(v) to zero. If v_i is of type VB, when using Eq. 3 each vertex that is a neighbor to v_i and is also right-sided with regard to the generator will have an additional 2π to its value. If v_i is of type VC, when using Eq. 3 each vertex that is a neighbor to v_i and is also on the generator will have an additional (-2π) in its value. The bounding of u and v can differ, i.e., f(u) and f(v) can be of the form given in Eq. 9, where u and v form the axis of the parameterization space.

$$f_u(i,j)=\begin{cases}0 & v_i\notin a\wedge v_i\notin N(a)\\ u\_\text{bounding value} & v_i\in a\wedge v_j\notin Nr(a)\\ -u\_\text{bounding value} & v_i\in Nr(a)\wedge v_j\in a\end{cases} \quad (9)$$

$$f_v(i,j)=\begin{cases}0 & v_i\notin b\wedge v_i\notin N(b)\\ v\_\text{bounding value} & v_i\in b\wedge v_j\notin Nr(b)\\ -v\_\text{bounding value} & v_i\in Nr(b)\wedge v_j\in b\end{cases}$$

3.4. Handling a foldover

As was proven in section 2.2.2, if v_{fix} holds a concave angle around the fan boundary of one of its neighbors, there is a possibility of a foldover. In practice, this did not happen on any of the modules we flattened. A simple solution to this problem is to subdivide the two overlapped triangles connected to v_{fix}, as shown in Figure 3.c. The algorithm for handling the foldover problem includes the following steps:

1. When the iterated process converges, look for flipped faces belonging to $v_{fix.}$
2. If there is no flipped face, then finish.
3. Subdivide the two flipped faces to eliminate the concave angles at the surrounding of v_{fix}. For the subdivision of the two triangles, two additional vertices are added to the mesh.
4. Continue the iteration process from the last result calculated in step 1, before the local subdivision is performed.
5. Go to 1.

4. Examples

The proposed parameterization method with cyclic boundary conditions was applied to close manifold genus-1 meshes. The embedding characteristic is controlled by the weights defined by the user. The method has been applied over meshes with obtuse angles and also over sparse and irregular meshes. Figure 4 shows the difference between using fixed boundary (Figure 4.b) and cyclic boundary (Figure 4.c) on a torus with harmonic weights. The corresponding parameterization is given in Figures 4.d and 4.e. One can see clearly the distortion near the generators when using the fixed boundary (Figure 4.c). In addition, Figure 4 emphasizes the lack of connection between the generators (Figure 4.a) selected when using cyclic boundary and the resulting parameterization. Figure 5 shows a comparison between the harmonic and mean-value weights on a loop model with a large number of obtuse angles. Figure 6 gives a comparison between harmonic and mean-value weights on a motorcycle helmet model. The mesh of the model is irregular and includes many obtuse angles. Figure 7 gives the results of the texture mapping and the corresponding parameterization space using Tutte, edge-length, harmonic and mean-value weights. The model chosen for Figure 7 is very sparse and the mesh is considered very low quality (Figure 7.a). Figure 7.g shows that the results for angle preserving when using mean-value weights are quite good regarding the given mesh. In Figure 7.i and Figure 7.j the cyclic boundaries are demonstrated by cutting the parameterization space and gluing it through the cyclic boundaries.

5. Summary

This paper has introduced a parameterization method for genus-1 objects using cyclic boundary conditions. The method is a generalization of the barycentric coordinates flattening process that uses fixed boundary. Moreover, the paper provides a proof for unfolding the flattened mesh and for process convergence using the Gauss-Seidel procedure. The basic idea of the method is based on replacing the fixed boundary conditions with cyclic boundary conditions. The only constraint on the weights is that they must be positive, so that any type of barycentric weights can be used, including the non-symmetric ones such as mean-value weights and shape preserving. The proposed parameterization method with cyclic boundary condition is as robust and fast as the conventional method with fixed boundary. The advantage of the proposed method is in the low distortion, resulting in the parameterization of genus-1 meshes, especially near the generators.

Appendix A: Convergence of the Gauss-Seidel procedure for weakly dominated matrices

The non-singularity property of strictly/weakly dominated matrices can be proven using the Gauss elimination procedure. In this section, the non-singularity property is used to prove that the Gauss-Seidel procedure converges for weakly-dominated matrices.

We will regard the linear system AX=b such that A is an n x n, weakly dominated matrix. We will define $A=D-L-U$ as follows:

1. L is the minus of the lower triangle of A.
2. U is the minus of the upper triangle of A.
3. D is the diagonal of A.
4. L and U do not contain the diagonal of A.

Based upon the above definition, the Gauss-Seidel iteration matrix G is defined as $G=(D-L)^{-1}U$. To show that Gauss-Seidel converges, it is sufficient to show that all eigenvalues of G in their absolute values are less than one. The proof is as follows:

$\det(\lambda I-G)=\det\left(\lambda(D-L)^{-1}\left[(D-L)\right]+(D-L)^{-1}\left[-U\right]\right)$ If λ is an eigenvalue of G, $\det(\lambda I-G)=0$ can be written as:

$$\det\left(\lambda(D-L)^{-1}\left[(D-L)\right]+(D-L)^{-1}\left[-U\right]\right)=0$$

Therefore $\det\left((D-L)^{-1}\left[\lambda(D-L)-U\right]\right)=0$ and finally $\det\left(\lambda(D-L)-U\right)=0$

Assume that $|\lambda|\geq 1$; then $\lambda(D-L)-U$ is strictly/weakly diagonally dominated, meaning that $\det\left(\lambda(D-L)-U\right)\neq 0$, which is a contradiction of our first assumption that strictly/weakly diagonally dominated matrices are non-singular and does not ensure convergence to the solution.

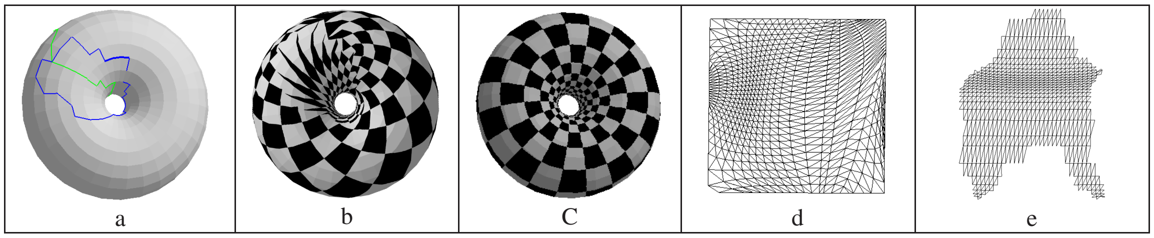

Figure 4: *Torus parameterization: (a) Torus and its two generators, (b) Texture mapping using harmonic weights on fixed boundary, (c) Texture mapping using harmonic weights on cyclic boundary, (d) Parameterization space when using fixed boundary, (e) Parameterization space when using cyclic boundary.*

References

[Ale00] ALEXA M.: Merging polyhedral shapes with scattered features*. The Visual Computer 16,* 1 (2000), 26-37.

[CSGL02] COHEN-OR D., SORKINE O., GOLDENTHAL R., LISCHINSKI D.: Bounded–distortion piecewise mesh parameterization, *IEEE Visualization* (2002), 355-362.

[DMA02] DESBRUN M., MEYER M., ALLIEZ, P.: Intrinsic parameterizations of surface mesh*es. Computer Graphics Forum 21,* 3 (2002), 210-218.

[DS95] DEY T. K., SCHIPPER H.: A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection. *Discrete Comput. Geom. 14* (1995), 93-110.

[EH96] ECK M., HOPPE, H.: Automatic Reconstruction of B-spline Surfaces of Arbitrary Topological Type. *Siggraph* (1996), 325-334.

[Flo03] FLOATER M.: Mean-value coordinates. *Computer Aided Geometric Design 20* (2003), 19-27.

[Flo97] FLOATER M. S.: Parameterization and smooth approximation of surface triangulation*. Computer Aided Geometric Design 14* (1997), 231-25.

[GSG03] GOTSMAN C., SHEFFER A., GU X.: Fundamentals of spherical parameterization for 3D meshes. *Siggraph* (2003), 358-363.

[GY02] GU X., YAU S.: Computing conformal structures of surface*s. Communications in Information and Systems 2,* 2 (2002), 121-146.

[GY00] GU X., YAU S.: Global conformal surface parameterization. *Siggraph* (2003), 127-137.

[Hat00] HATCHER A.: *Algebraic Topology* 2000, http://www.math.cornell.edu/~hatcher/.

[LRSS02] LOPES H., ROSSIGNAC J., SAFANOVA A., SZYMCZAK A., TAVARES G.: Edgebreaker: A simple compression for surfaces with handles. *ACM Seventh Solid Modeling and Application Conference*, Saarbrücken, Germany (June, 2002), 289-296.

[EH02] JEFF E., SARIEL H. P.: Optimally cutting a surface into a disk. *SoCG.* Barcelona, Spain (June, 2002), 244-253.

[Kar99] KARTASHEVA E.: Reduction of h-genus polyhedrons topology. *International Journal of Shape Modeling 5,* 2 (1999), 179-194.

[LPVV01] LAZARUS F., POCCHIOLA M., VEGTER, G., VERROUST A.: Computing a canonical polygonal schema of an orientable triangulated surface. *Proc. 17th Annu. ACM Sympos. Comput. Geom* (2001), 80-89.

[Mun84] MUNKERS J. R.: *Elements of algebraic topology*. Addison-Wesley, Redwood City, CA, 1984.

[ST98] SHAPIRO A., TAL A.: Polygon realization for shape transformation. *The Visual Computer 14* (1998), 8-9, 429-444.

[SGD03] SHEFFER A., GOTSMAN C., DYN N.: Robust spherical parameterization of triangular meshes. *Proceedings of 4th Israel-Korea Binational Workshop on Computer Graphics and Geometric Modeling*, Tel Aviv (2003), 94-99.

[SDs01] SHEFFER A., DE STURLER E.: Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers 1*7, 3 (2001), 326-337.

[SF02] STEINER D., FISCHER A.: Cutting 3D freeform objects with genus-n into one-boundary surface using topological graphs. *ACM Seventh Solid Modeling and Application Conference*, Saarbrücken, Germany (June 2002), 336-343.

[SF03] STEINER D., FISCHER A.: Explicit representation of object holes in topological and cut graphs. *Proceedings of the 4th Israel-Korea Bi-National Conference*, Tel-Aviv, Israel (February 2003), 44-44.

[Tut63] TUTTE W. T.: How to draw a graph. *Proc. London Math. Soc. 13* (1963), 743-768.

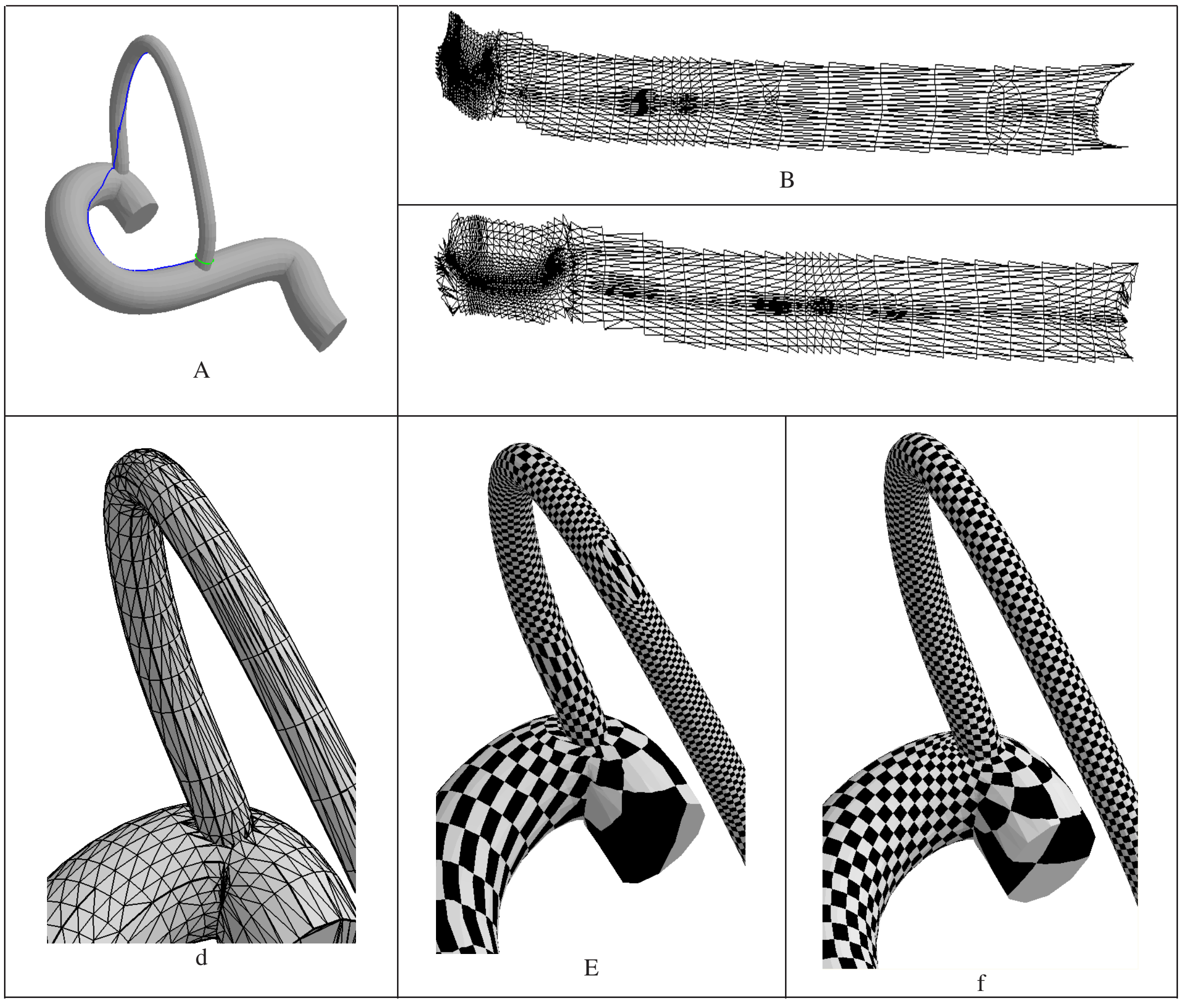

Figure 5: *Loop, (a) loop with its two generators, (b) parameterization space using harmonic weights, (c) parameterization space using mean-value weights, (d) zooming on a problematic area with obtuse angles, (e) texture mapping using harmonic weights, (f) texture mapping using mean-value weights.*

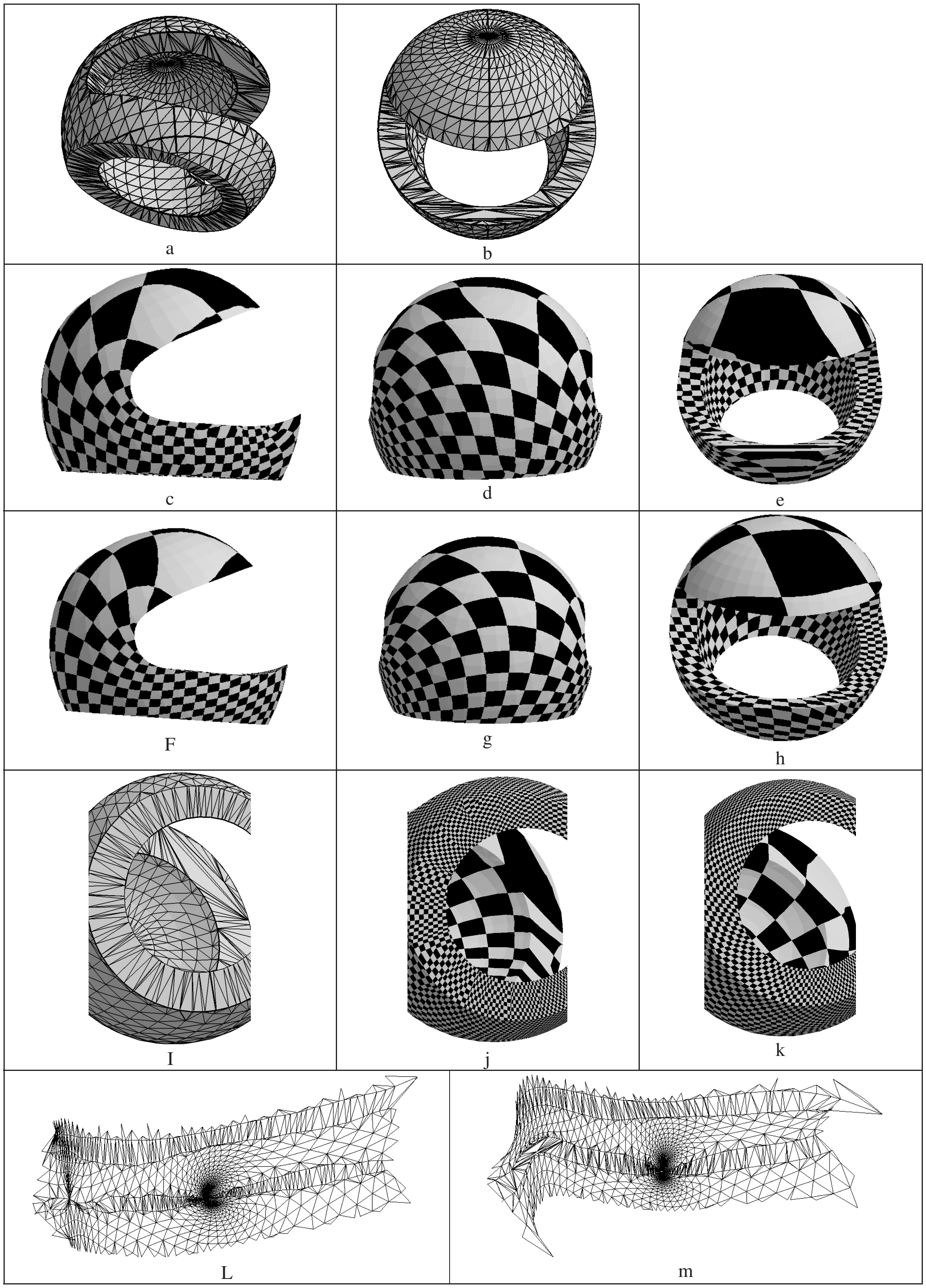

Figure 6: *Motorcycle helmet: (a,b) Irregular mesh with obtuse angles, (c,d,e) texture mapping results using harmonic weights, (f,g,h) texture mapping results using mean value weights, (i,j,k) zooming on a problematic area with obtuse angles, (l,m) parameterization space using harmonic and mean-value weights, respectively.*

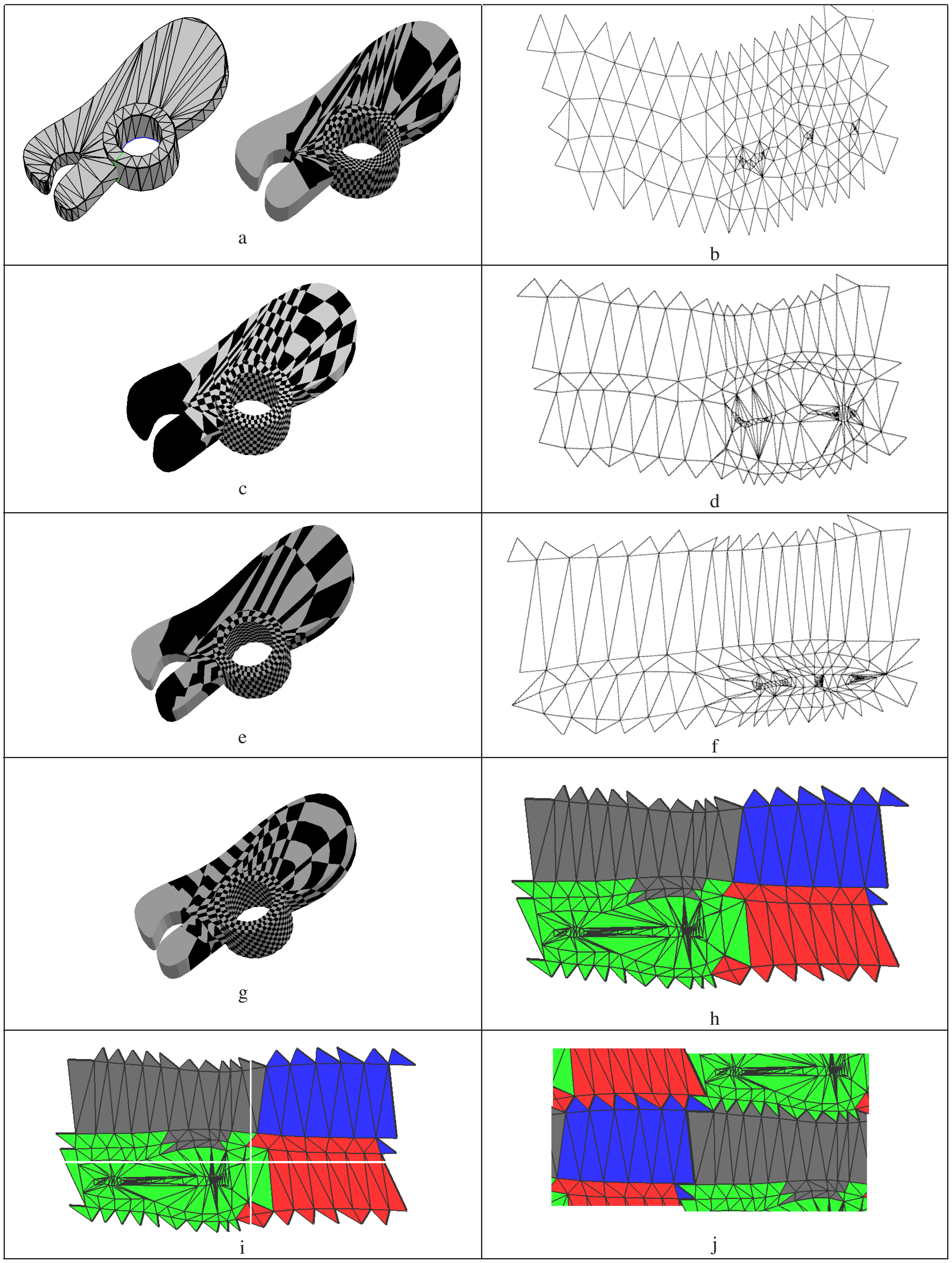

Figure 7: *Different weights texture mapping results on a highly irregular mesh and the resulting parameterization space: (a) Mesh and Tutte barycentric coordinates, (b) Tutte, parameterization space, (c) edge length, (d) Edge length, parameterization space, (e) harmonic, (f) Harmonic, parameterization space, (g) mean value, (h) Mean value, parameterization space, (i) cutting the parameterization space ,(j) gluing through the cyclic boundaries.*