



# Mesh parameterization with a virtual boundary

Yunjin Lee<sup>a</sup>, Hyoung Seok Kim<sup>b</sup>, Seungyong Lee<sup>a,\*</sup>

<sup>a</sup> Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang 790-784, South Korea

<sup>b</sup> Department of Multimedia Engineering, Donggeui University, Pusan 614-714, South Korea

## Abstract

Parameterization of a 3D triangular mesh is a fundamental problem in various applications of meshes. The convex combination approach is widely used for parameterization because of its good properties, such as fast computation and one-to-one embedding. However, the approach has a drawback: most boundary triangles have high distortion in the embedding compared with interior ones. In this paper, we present an extension of the convex combination approach that resolves the drawback by using a virtual boundary. Since the virtual boundary is fixed onto a given convex polygon instead of the real boundary, the real boundary triangles can better reflect the shape of the corresponding 3D triangles. The proposed approach obtains a parameterization of a 3D mesh with less distortion than with the original convex combination approach. © 2002 Elsevier Science Ltd. All rights reserved.

*Keywords:* Parameterization; Texture mapping; Convex combination; Virtual boundary

## 1. Introduction

Parameterization of a 3D triangular mesh is a fundamental process for many applications of meshes, such as texture mapping [1–6], multi-resolution modeling [7,8], and smooth surface fitting [9–11]. Mesh parameterization can be regarded as embedding a 3D mesh into a 2D parameter space, where a 2D coordinate  $(s, t)$  is assigned to each point on the surface of a mesh. In general, the mesh cannot be flattened over a plane without distortion [1,2]. In many applications using parameterization, it is important to minimize distortion, which is the primary objective of parameterization. One-to-one mapping is also a very important property of parameterization.

Two major paradigms exist in mesh parameterization: energy functional minimization and the convex combination approach. For the first paradigm, several approaches have been developed to define and minimize an energy functional that measures distortion in the

embedded mesh. Maillot et al. proposed a method to minimize a norm of the Green–Lagrange deformation tensor based on elasticity theory [2]. The harmonic embedding used by Eck et al. minimizes the metric dispersion instead of elasticity [7]. Lévy et al. proposed an energy functional minimization method based on orthogonality and homogeneous spacing [3]. A non-deformation criterion (i.e., Dirichlet energy per parameter area) is introduced in [10] with extrapolating capabilities.

The convex combination approach is an extension of the barycentric mapping approach proposed by Tutte [12]. This approach obtains parameterization by fixing the boundary vertices of a 3D mesh onto a 2D convex polygon and solving a linear system to determine the 2D embedded positions of the interior vertices. The linear system is constructed by representing each interior vertex as a convex combination of its neighborhood. In this approach, the major problem concerns how to determine the coefficients of the convex combination for each interior vertex. Floater [9] proposed shape-preserving parameterization, where the coefficients are determined by using conformal mapping and barycentric coordinates. The harmonic embedding [7] is also a special case of this approach, except that the coefficients may be negative.

\*Corresponding author.

*E-mail addresses:* jin@postech.ac.kr (Y. Lee), hskim@donggeui.ac.kr (H.S. Kim), leesy@postech.ac.kr (S. Lee).

*URL:* <http://home.postech.ac.kr/~jin>, <http://www.donggeui.ac.kr/~hskim>, <http://www.postech.ac.kr/~leesy>.

The convex combination approach to parameterization is simple and fast, while energy functional minimization may require extensive computation since the functional generally requires a non-linear solver. In addition, the convex combination approach always creates a one-to-one embedding. Because of these advantages, many mesh applications use this convex combination method [7,13–16]. However, this approach requires the boundary of a mesh to be fixed onto a convex polygon, which causes high distortion near the boundary. The embedding of a mesh cannot reflect the 3D boundary shape of the mesh because the boundary shape of the embedding is forced to be the same as the convex polygon.

In this paper, we extend the convex combination approach to reduce high distortion near the boundary. We enable the boundary shape of the embedding to reflect the 3D boundary shape by allowing vertices on the boundary to move in a 2D parameter space in the same manner as the inner vertices. To move boundary vertices freely, we attach a set of virtual vertices to the boundary of a mesh and map the virtual vertices to a 2D convex polygon instead of the real boundary vertices. Accordingly, our method can generate an embedding with less distortion for a mesh with a complicated boundary than the original approach. We can guarantee that a one-to-one mapping is generated by solving a linear system since the method is based on the convex combination approach.

The organization of this paper is as follows. Section 2 summarizes the convex combination approach as a preliminary. Section 3 introduces the basic concept and the process of the proposed method, and describes the construction of a virtual boundary and how to determine the coefficients of virtual vertices. Additional improvements to our method will be explained in Section 4. We show some examples in Section 5 and conclude this paper in Section 6.

## 2. Parameterization based on convex combinations

Floater proposed the convex combination approach to parameterization [9]. Let  $u_1, \dots, u_N$  be the 2D embedded positions of 3D vertices,  $v_1, \dots, v_N$ , where  $v_1, \dots, v_n$  and  $v_{n+1}, \dots, v_N$  are the interior and boundary

vertices of a mesh, respectively. The convex combination approach determines the values of  $u_{n+1}, \dots, u_N$  by mapping the 3D boundary onto a given convex polygon in 2D parameter space. To obtain the values of  $u_1, \dots, u_n$ , the approach represents  $u_i$  as a convex combination of  $u_j$ , where  $v_j$  are the one-ring neighborhood vertices of  $v_i$ . In other words,

$$u_i = \sum_{j=1}^N \lambda_{i,j} u_j, \quad i = 1, \dots, n,$$

$$\lambda_{i,j} > 0, \quad (i,j) \in E, \quad \lambda_{i,j} = 0, \quad (i,j) \notin E,$$

$$\sum_{j=1}^N \lambda_{i,j} = 1, \quad (1)$$

where  $E$  is the edge set of the mesh. We can compute the values of  $u_1, \dots, u_n$  by solving the linear system in Eq. (2), which is derived from Eq. (1).

$$u_i - \sum_{j=1}^n \lambda_{i,j} u_j = \sum_{j=n+1}^N \lambda_{i,j} u_j, \quad i = 1, \dots, n. \quad (2)$$

Floater proved that a unique solution of the linear system in Eq. (2) always exists. He also proved that if mapping between the 3D boundary and the 2D convex polygon is one-to-one, mapping for the interior vertices becomes an embedding without overlaps. The shape of the embedding depends on the coefficients  $\lambda_{i,j}$  of the convex combination. Floater proposed three methods to obtain the coefficients: uniform parameterization, chord length parameterization, and shape-preserving parameterization. Among them, the last method best reflects the shape of a mesh since the method has the affine invariant property.

Figs. 1(b) and (c) are 2D embeddings of the 3D mesh shown in Fig. 1(a) using shape-preserving parameterization [9], which use a square and a circle as the bounding polygon, respectively. We ascertain a drawback with the convex combination approach: high distortion occurs near the boundary. We can observe that triangles around the corners of a square differ considerably from triangles in the 3D mesh.

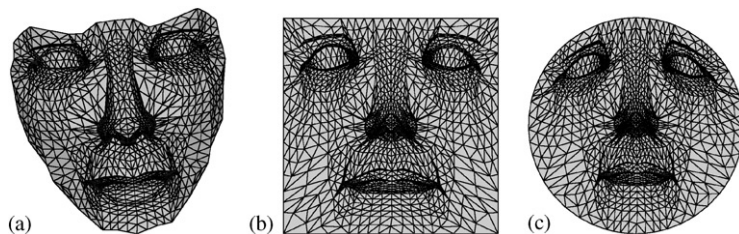


Fig. 1. Parameterization of a triangular mesh with the convex combination approach: (a) 3D triangular mesh; (b) square; (c) circle.

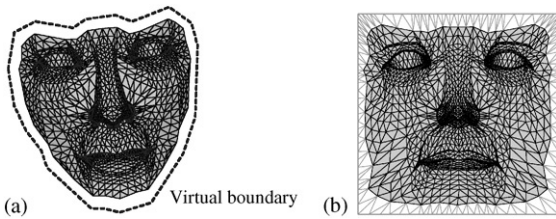


Fig. 2. Virtual boundary in 3D and 2D space: (a) Mesh and virtual boundary; (b) virtual boundary on the parameter space.

### 3. Parameterization with a virtual boundary

#### 3.1. Basic idea

The major problem of parameterization of a 3D triangular mesh concerns how to embed a mesh onto a 2D parameter space so that the shapes of triangles can be well preserved. In the convex combination approach, fixing the boundary of a mesh onto a convex polygon highly deforms the triangles near the boundary in 2D parameter space. In contrast, if boundary vertices can move to reflect the 3D boundary shape of a mesh, we can decrease distortion near the boundary of an embedding. Hence, our basic idea starts from how to make vertices on the boundary move to reflect the 3D boundary shape of a mesh in 2D parameter space. To freely move boundary vertices, we attach a set of virtual vertices to the boundary, as illustrated in Fig. 2(a), and map the virtual vertices onto a bounding convex polygon instead of real boundary vertices. Then we can reduce the distortion near the boundary in the embedding, as shown in Fig. 2(b).

#### 3.2. Parameterization process

In the proposed method, the parameterization process can be summarized as follows (see Fig. 3).

- (1) Virtual vertices are attached to the real boundary without 3D geometric information.
- (2) The coefficients of convex combinations are computed for interior vertices and real boundary vertices.
- (3) The shape of the bounding convex polygon is determined in the parameter space, and the virtual boundary is mapped onto the polygon.
- (4) The linear system from the convex combinations is solved to determine the embedded positions of the interior and real boundary vertices.

All processes are the same as the convex combination approach, except the creation and mapping of a virtual boundary and the computation of coefficients for real boundary vertices. We will account for these additional and changed parts in detail in the following subsections.

#### 3.3. Virtual boundary vertices

To use a virtual boundary for parameterization, we must assign topological information to virtual vertices and embed them onto a given bounding convex polygon. As illustrated in Fig. 4, the vertices on the real boundary and the virtual boundary are denoted by  $v_i$  and  $v'_i$ , respectively. In the convex combination approach, the vertices on the boundary are parameterized on the bounding convex polygon by considering the edge length between two adjacent vertices. In the case of the virtual boundary, we do not have the 3D positions of virtual vertices. Hence, we map the virtual vertices onto

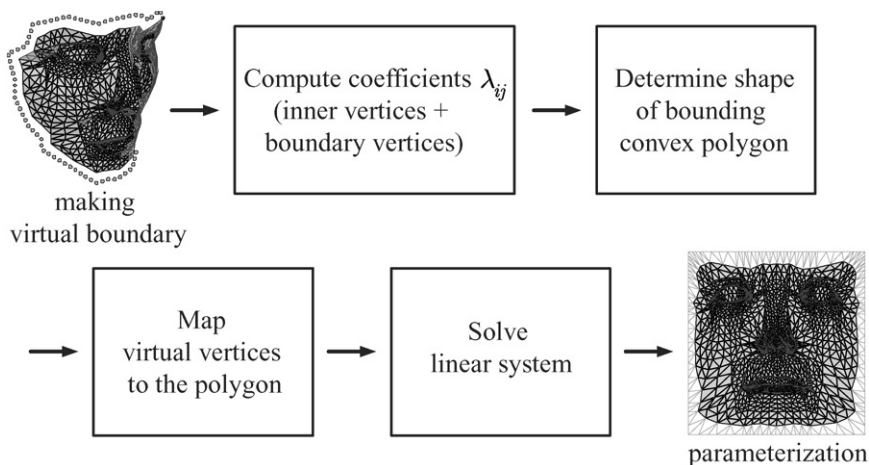


Fig. 3. Parameterization process.

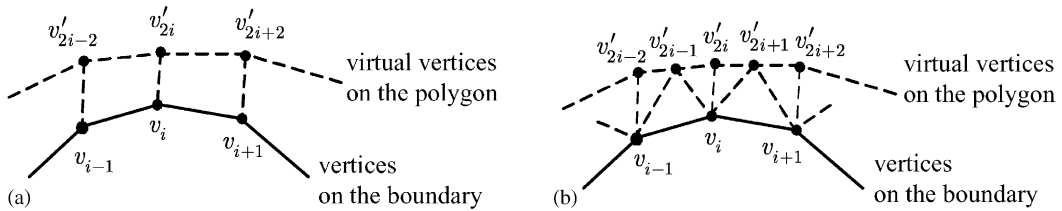


Fig. 4. Virtual vertices and their connectivity to real boundary vertices: (a) Initial placement of virtual vertices; (b) symmetric triangulation.

the convex polygon by using edge lengths between the real boundary vertices.

To use the edge lengths of the 3D boundary, we first make the same number of virtual vertices and connect a virtual vertex  $v'_{2i}$  to a real boundary vertex  $v_i$ , as shown in Fig. 4(a). Next, we place the vertices  $v'_{2i}$  on the bounding convex polygon so that the lengths of the edges between  $v'_{2i}$  and  $v'_{2i+2}$  are proportional to the distances between  $v_i$  and  $v_{i+1}$ . These attachment and placement of virtual vertices prevent drastic changes of the relative distances between adjacent real boundary vertices in the parameterization. That is, if two adjacent vertices on the real boundary are close, they will also be close in the 2D embedding.

To triangulate the faces between real and virtual boundaries, we insert another virtual vertex  $v'_{2i+1}$  at the middle of virtual vertices  $v'_{2i}$  and  $v'_{2i+2}$  (see Fig. 4(b)). With the insertion of vertices  $v'_{2i+1}$ , we can obtain symmetric connectivity between real and virtual boundaries, and make each real boundary vertex adjacent to the same number of virtual vertices. Consequently, the number of virtual vertices is twice that of the real boundary vertices and each real boundary vertex  $v_i$  is adjacent to three virtual vertices,  $v'_{2i-1}$ ,  $v'_{2i}$ , and  $v'_{2i+1}$ . We also expect the real boundary vertices to move more freely, with virtual vertices twice number of the real boundary vertices.

### 3.4. Local parameterization of real boundary vertices

We use shape-preserving parameterization to compute the coefficients of vertices in the convex combinations.

The coefficients are computed from the local parameterization of a vertex and its one-ring neighborhood. Conformal mapping is used for local parameterization, and the coefficients are acquired by averaging the barycentric coordinates. Refer to [9] for the details.

In the case of our augmented mesh, we should compute the coefficients for the real boundary vertices as well as the interior ones. Note that the virtual vertices, not the real boundary vertices, are fixed onto the given bounding convex polygon. In the local parameterization of a real boundary vertex, we cannot directly use conformal mapping because the virtual vertices have no geometry.

To preserve the local shape of the boundary, we first map a real boundary vertex and its real neighbor vertices onto the 2D parameter space. They can be mapped onto the 2D space without distortion, while the angles and lengths of the real triangles are preserved. Figs. 5(a) and (b), respectively, show 3D configuration and 2D mapping of a real boundary vertex and its neighbors. We then place the virtual vertices  $v'_{2i-1}$ ,  $v'_{2i}$ , and  $v'_{2i+1}$  as shown in Fig. 5(c). Here, the distances between  $v_i$  and  $v'_j$ ,  $j = 2i - 1, 2i, 2i + 1$ , are the same as the average of lengths of the real boundary edges adjacent to  $v_i$ . The angles around  $v_i$  of the virtual triangles are the same as  $(2\pi - \theta)/4$ , where  $\theta$  is the sum of angles around  $v_i$  of the real triangles. Note that four virtual triangles always exist around a real boundary vertex. Finally, we compute the coefficients for the boundary vertices by averaging the barycentric coordinates, as in shape-preserving parameterization.

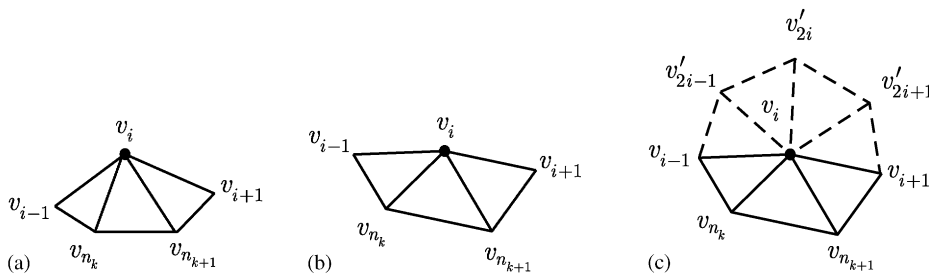


Fig. 5. Local parameterization of a real boundary vertex and its 1-ring neighborhood: (a) 3D configuration; (b) 2D mapping; (c) placement of virtual vertices.

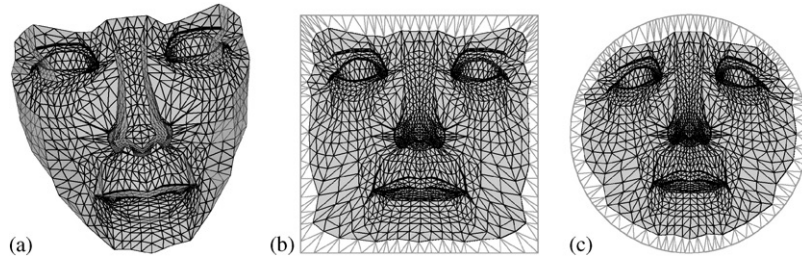


Fig. 6. Embedding results with virtual boundaries: (a) 3D triangular mesh; (b) square; (c) circle.

By using the average length of real edges for the virtual edge length, we can prevent the generation of skinny triangles. No specific virtual vertex of the three has a dominant effect on the real boundary vertex with the assignment of the same length and angle.

Figs. 6(b) and (c) show the embedding results obtained when the bounding polygon is a square and circle, respectively. The examples show that we can obtain better results than those of the original convex combination approach [9] in Fig. 1. This is because the virtual boundary is fixed instead of the real boundary; that is, the real boundary vertices can freely move.

#### 4. Additional improvement

##### 4.1. Shape of a bounding polygon

We can use various 2D polygons, such as a square and circle, for the bounding convex polygon onto which the virtual boundary is fixed. Although we can acquire a better result with a virtual boundary, the shape of an embedding is still strongly influenced by the bounding convex polygon. For example, the overall shapes of the real boundary in Figs. 6(b) and (c) are close to the square and circle, respectively. In Fig. 6(b), the embedded triangles around the corner of the square have higher distortions than other boundary triangles. We can reduce the distortions of boundary triangles by using the bounding convex polygon whose shape closely resembles that of the 3D boundary.

A simple approach for deriving such a bounding convex polygon is to project the real boundary onto the plane obtained by the least-square fitting of the 3D boundary vertices. When the projected boundary is concave, we can use the convex hull of the projected boundary as the bounding convex polygon. However, a limitation with this approach is that the projected boundary may not well reflect the 3D boundary, depending on the configuration of the 3D boundary vertices.

To obtain a bounding convex polygon that resembles the 3D boundary, we first derive a 2D polygon that

preserves the adjacent angles and lengths of 3D boundary edges. In general, such a 2D polygon may not exist when the 3D boundary is non-planar. Sederberg et al. considered a similar problem for 2D shape blending and proposed the edge tweaking method [17]. In edge tweaking, the changes of edge lengths are minimized while the angles between edges are preserved. In this paper, we adopt the edge tweaking method to obtain a 2D polygon from the 3D boundary. The bounding convex polygon is determined as the convex hull of the 2D polygon. Now we explain how the edge tweaking method is used in this paper.

For the edge tweaking method, we represent the original edge lengths, the amounts of length adjustment, and the adjacent angles between edges as  $l_i$ ,  $s_i$ , and  $\theta_i$ , respectively, for  $i = n + 1, n + 2, \dots, N$ . Fig. 7(a) shows an illustration. The problem is to find the values of  $s_i$  that minimize the objective function

$$f(s_{n+1}, \dots, s_N) = \sum_{i=n+1}^N \frac{s_i^2}{l_i^2}$$

subject to the equality constraints

$$\varphi_1(s_{n+1}, \dots, s_N) = \sum_{i=n+1}^N (l_i + s_i) \cos(\alpha_i) = 0,$$

$$\varphi_2(s_{n+1}, \dots, s_N) = \sum_{i=n+1}^N (l_i + s_i) \sin(\alpha_i) = 0.$$

Here,  $N - n$  is the number of the edges, and  $\alpha_i$  is the angle between the  $x$ -axis and the directed edge from  $u_i$  to  $u_{i+1}$ . The two equality constraints mean that the 2D polygonal curve with the edge lengths of  $l_i + s_i$  is a closed polygon. The solution for the values of  $s_i$  can be obtained by using Lagrange multipliers. For the details of the solution process, refer to [17].

Let  $v_i$  and  $u_i$ , for  $i = n + 1, n + 2, \dots, N$ , denote vertices on the 3D mesh boundary and the corresponding vertices on the 2D polygon, respectively. To obtain  $u_i$  from  $v_i$  by using the edge tweaking method, we must determine the original lengths  $l_i$  and the angles  $\theta_i$ . For the lengths  $l_i$ , we can simply set  $l_i$  as the length of the 3D boundary edge between  $v_i$  and  $v_{i+1}$ . In the case of the angles  $\theta_i$ , we compute  $\theta_i$  from the angles  $\beta_{i,j}$  around  $v_i$  of

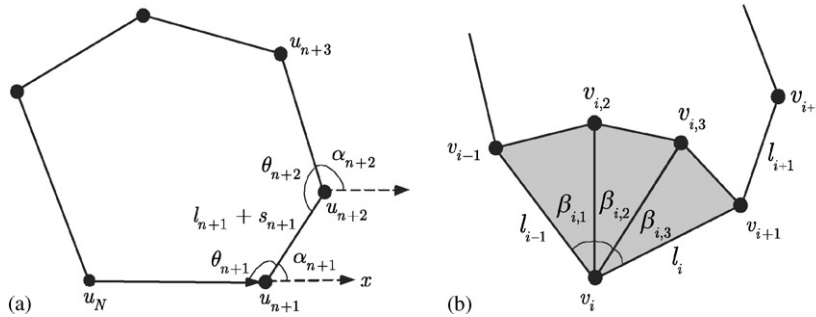


Fig. 7. Edge tweaking for the bounding convex polygon: (a) 2D polygon; (b) 3D mesh boundary.

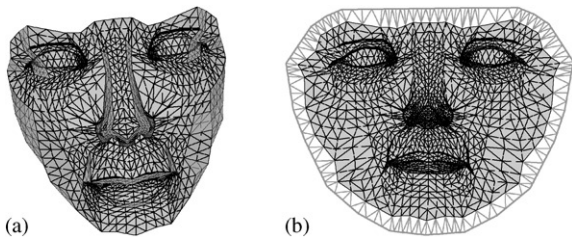


Fig. 8. Embedding result with the bounding polygon that resembles the 3D boundary: (a) 3D mesh; (b) bounding polygon from edge tweaking.

the boundary triangles adjacent to  $v_i$  as follows (see Fig. 7(b)).

A simple way to compute  $\theta_i$  is to sum the angles  $\beta_{i,j}$ . However, the angles  $\beta_{i,j}$  may be changed when the boundary triangles are embedded onto a 2D polygon, and the sum of  $\beta_{i,j}$  in 3D may be improper for  $\theta_i$ . To estimate the values of angles  $\beta_{i,j}$  in the final embedding of the 3D mesh, we compute the averaged conformal angles  $\gamma_{i,j}$ . Let  $V_i$  be the set of vertices of the 3D mesh which are adjacent to the boundary vertex  $v_i$ . The set  $V_i$  is divided to two subsets; a boundary vertex set  $\{v_{i,1}, v_{i,c}\}$  and an interior vertex set  $\{v_{i,2}, \dots, v_{i,c-1}\}$ , where  $v_{i,1} =$

$v_{i-1}$  and  $v_{i,c} = v_{i+1}$ , and  $c$  is the number of vertices in  $V_i$ . See Fig. 7(b) for an illustration. When we apply conformal mapping to the one-ring neighborhood of an interior vertex  $v_{i,j}$ , we obtain values  $\beta_{i,j-1}^2$  and  $\beta_{i,j}^1$  for the angles  $\beta_{i,j-1}$  and  $\beta_{i,j}$ , respectively. We define the averaged conformal angles  $\gamma_{i,j}$  by

$$\gamma_{i,1} = \beta_{i,1}^2, \quad \gamma_{i,c-1} = \beta_{i,c-1}^1, \quad \text{and} \quad \gamma_{i,j} = \frac{\beta_{i,j}^1 + \beta_{i,j}^2}{2}$$

$$\text{for } j = 2, \dots, c - 2.$$

Finally, we determine the angle  $\theta_i$  as the sum of the averaged conformal angles  $\gamma_{i,j}$ ;

$$\theta_i = \sum_{j=1}^{c-1} \gamma_{i,j}.$$

In Fig. 8, the bounding convex polygon is the convex hull of the 2D polygon generated by the edge tweaking method. The embedding result is better than those in Fig. 6 because the bounding polygon reflects the 3D boundary of the mesh. Fig. 9 shows an example of parameterizing a mesh with a more complicated boundary. Figs. 9(b)–(d) are the embedding results obtained by fixing the virtual boundary onto a square, a circle, and the bounding polygon from edge tweaking, respectively. We can also observe that the embedding in

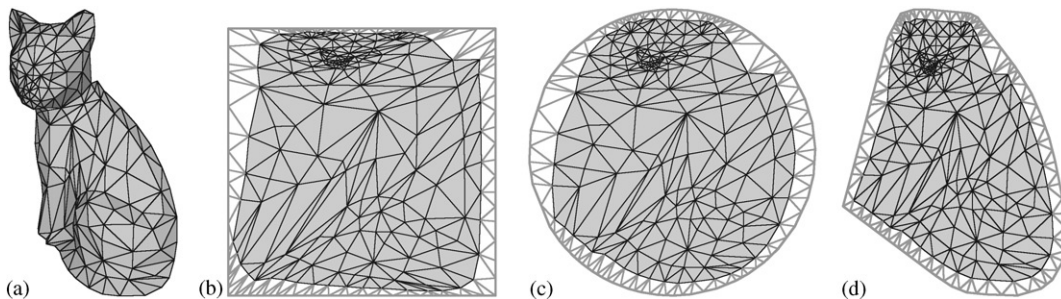


Fig. 9. Embedding results with different bounding polygons: (a) 3D mesh; (b) square; (c) circle; (d) edge tweaking.

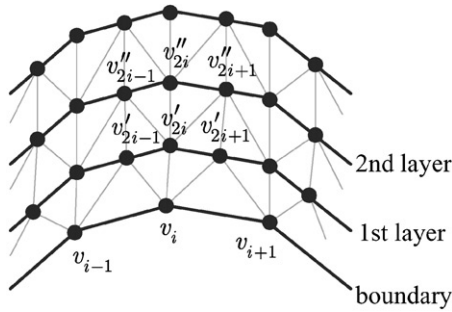


Fig. 10. Construction of a multi-layered virtual boundary.

Fig. 9(d) has less distortion near the boundary than the others.

4.2. Multi-layered virtual boundary

Although we can reduce the distortion near the boundary by using virtual vertices, the shape of the 2D embedding still remains under the strong influence of the bounding convex polygon. Specifically, high distortion can appear in the concave region of the 2D polygon generated by the edge tweaking method. In Fig. 6, we observe that the triangle shapes in a region far from the boundary are well-preserved. Hence, to reduce the influence of the bounding polygon, we can add more virtual vertices in a multi-layered structure.

To prevent an exponential increase in the number of virtual vertices, each layer is crafted to contain the same number of virtual vertices. Fig. 10 shows the connectivity of virtual vertices between layers. With this connectivity, all virtual vertices except the first layer have the same number of adjacent vertices in a symmetric and consistent structure.

In contrast to the single-layered structure, we must compute the coefficients of the convex combinations for virtual vertices, except in the last layer, since their positions are not determined by the mapping of the virtual boundary onto the bounding convex polygon. For simplicity, the coefficients are set to be uniformly  $1/d_i$ , where  $d_i$  is the degree of a virtual vertex  $v_i$ . For example, in the case of  $v'_{2i}$  in Fig. 10, the coefficients are  $1/6$ .

Fig. 11 shows the results of the parameterization using multi-layered virtual boundaries. When the 3D boundary severely runs in and out, as shown in Fig. 11(a), the convex hull of the 2D polygon obtained by the edge tweaking method considerably differs from the shape of the 3D boundary. Fig. 11(b) shows the result of 2D embedding when the real boundary is fixed to the convex hull. The triangles around the nose are stretched because of the concave part of the 3D boundary. The distortion is reduced when we add virtual vertices in a multi-layered structure to the real boundary, as shown in Figs. 11(c)–(f).

5. Experimental results

Table 1 shows the comparison of distortion measurements for the face model in Fig. 11(a) with three cases: the bounding convex polygon is a square, a circle, and a 2D polygon from edge tweaking. We use the texture stretch metric  $L^2$  and  $L^\infty$  defined in [18] as the measures for comparison. The  $L^2$  norm corresponds to the mean stretch over all directions, and the worst-case norm  $L^\infty$  relates to the greatest stretch.

From Table 1, we observe that the distortion measurement changes according to the shape of the bounding polygon, and the value decreases as the

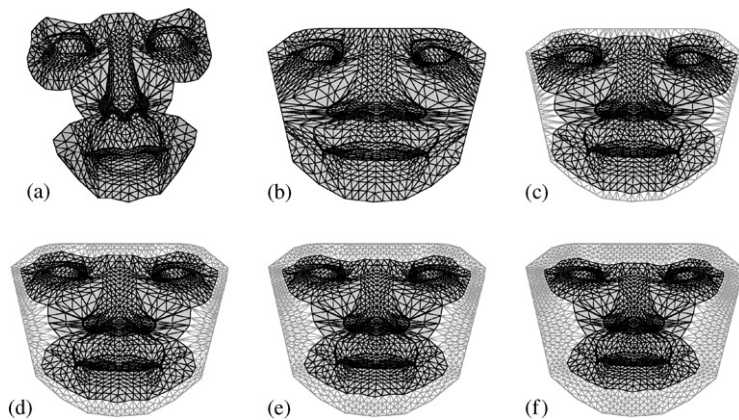


Fig. 11. Effects of multi-layered virtual boundaries: (a) 3D mesh; (b) no virtual vertices; (c) one layer; (d) two layers; (e) three layers; (f) four layers.

Table 1  
Comparison of distortion measurements

| Boundary shape, boundary type (# virtual layers) | $L^2$   | $L^\infty$ |
|--|---------|------------|
| Square, real (0)                                 | 1.24656 | 5.23226    |
| Square, virtual (1)                              | 1.16722 | 5.08168    |
| Circle, real (0)                                 | 1.44516 | 12.68943   |
| Circle, virtual (1)                              | 1.25441 | 7.73539    |
| Tweaking edges, real (0)                         | 1.20768 | 4.96846    |
| Tweaking edges, virtual (1)                      | 1.14169 | 4.89727    |
| Tweaking edges, virtual (2)                      | 1.13204 | 4.76854    |
| Tweaking edges, virtual (7)                      | 1.16177 | 6.32086    |

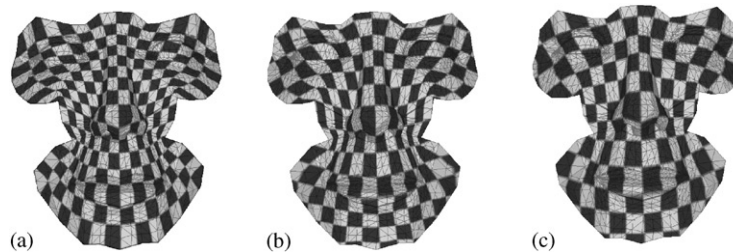


Fig. 12. Texture mapping results for a face model: (a) Rectangle, real boundary; (b) edge tweaking, real boundary; (c) edge tweaking, two virtual layers.

bounding polygon is closer to the shape of the 3D boundary. Hence, the embedding with the edge tweaking method has a better result than the others. Moreover, the result with a virtual boundary is superior to that of the original convex combination approach for every bounding convex polygon.

In Table 1, the embedding with two virtual layers shows a better result than with a single layer. However, the results with seven virtual layers is worse than the others. This implies that adding more virtual layers does not always generate a better embedding result. The threshold for the number of virtual layers which obtains the best embedding result in terms of  $L^2$  and  $L^\infty$  stretch metrics is not clear.

Parameterization of a 3D mesh has various applications in computer graphics, such as texture mapping, multi-resolution modeling, and smooth surface fitting. Fig. 12 shows the results of texture mapping, where a check-board pattern is mapped onto the mesh in Fig. 11(a). In the case of a square boundary without virtual vertices, the size and shape of rectangles in the check-board pattern are much distorted in the texture mapping, as shown in Fig. 12(a). Figs. 12(b) and (c) are the results from edge tweaking with no virtual layer and with two virtual layers, respectively. Fig. 12(b) shows that a 2D bounding polygon similar to a 3D boundary decreases the distortion near the boundary. However, the texture around the nose is still more distorted than around the jaw, due to severe concavity. In Fig. 12(c),

the texture is more uniformly mapped around the nose by using two virtual layers.

Fig. 13 shows another texture mapping example. Fig. 13(a) is a 3D mesh model. Two figures in Fig. 13(b) are the embedding and texture mapping results, where we embed the 3D boundary onto a square. Two figures in Fig. 13(c) are the results when we use a virtual boundary and map it to the polygon obtained by edge tweaking. We can observe that the bounding polygon in the left figure of Fig. 13(c) resembles the 3D boundary in Fig. 13(a) more closely than Fig. 13(b). Hence, the texture is more uniformly mapped on the surface in Fig. 13(c) than in Fig. 13(b).

Table 2 shows the computation time required to parameterize the models shown in this paper with and without virtual boundaries. In the current implementation, we use the Gauss–Seidel method to solve the linear system from the convex combinations. Since the number of boundary vertices is much smaller than that of all vertices in a mesh, the computation time with virtual boundaries increases only in a little amount, as shown in Table 2.

## 6. Conclusion

In this paper, we proposed an extension of the convex combination approach to reduce high distortion around the boundary in the parameterization of a 3D mesh. We

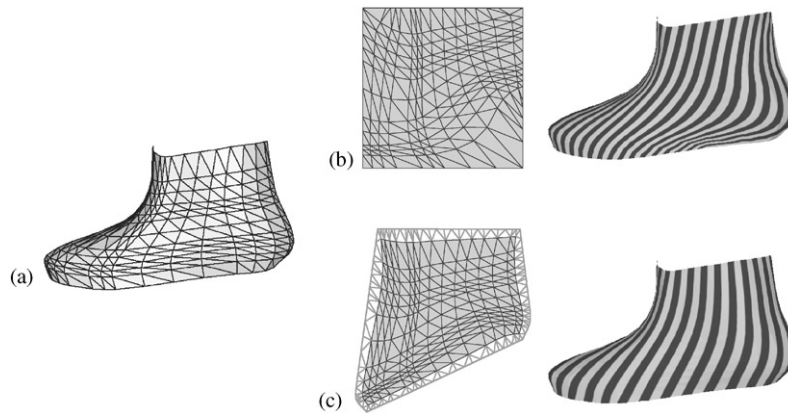


Fig. 13. Texture mapping results for a Shoe model: (a) 3D mesh; (b) square, no virtual boundary; (c) edge tweaking, one virtual layer.

Table 2

Computation time for parameterization (measured in milliseconds on a 500 MHz Pentium III PC)

| Model          | # vertex | # Boundary vertex | # Virtual layers | Computation time (ms) |
|----------------|----------|-------------------|------------------|-----------------------|
| Face (Fig. 6)  | 1607     | 68                | 0                | 430                   |
|                |          |                   | 1                | 471                   |
|                |          |                   | 2                | 530                   |
| Face (Fig. 11) | 1547     | 92                | 0                | 350                   |
|                |          |                   | 1                | 460                   |
|                |          |                   | 2                | 560                   |
| Shoe (Fig. 13) | 195      | 52                | 0                | 1                     |
|                |          |                   | 1                | 16                    |
|                |          |                   | 2                | 17                    |

introduced the virtual boundary of a mesh, which is fixed to the bounding convex polygon instead of the real boundary. To improve the embedding results, we presented an approach to determine the 2D bounding polygon that reflects the 3D boundary. We also proposed a multi-layered structure of virtual vertices, which is useful for a mesh with a severely concave boundary.

In future work, we will investigate the adaptive addition of virtual layers that reflects concavity of the 2D boundary obtained by edge tweaking. In addition, we will apply a speed up technique with a multi-level approach, such as multi-grid relaxation, to solve the linear system.

#### Acknowledgements

The cat model in Fig. 9 and the shoe model in Fig. 13 are acquired from the 3DCAFE's web page (<http://www.3dcafe.com>). This work was supported in part by

the Korea Ministry of Education through the Brain Korea 21 program, the KIPA Game Animation Research Center, the KOSEF Virtual Reality Research Center, and Korea Research Foundation Grant (KRF-99-042-E00053).

#### References

- [1] Ma SD, Lin H. Optimal texture mapping. *Computer Graphics Forum (Proc Eurographics '90)* 1988;7(3):421–8.
- [2] Maillot J, Yahia H, Verroust A. Interactive texture mapping. *ACM Computer Graphics (Proc. SIGGRAPH '93)* 1993;27–34.
- [3] Lévy B, Mallet J-L. Non-distorted texture mapping for sheared triangulated meshes. *ACM Computer Graphics (Proc. SIGGRAPH '98)* 1998;343–52.
- [4] Praun E, Finkelstein A, Hoppe H. Lapped textures. *ACM Computer Graphics (Proc. SIGGRAPH 2000)* 2000;465–70.
- [5] Piponi D, Borshukov G. Seamless texture mapping of subdivision surfaces by model pelting and texture blending. *ACM Computer Graphics (Proc. SIGGRAPH 2000)* 2000;471–8.

- [6] Lévy B. Constrained texture mapping for polygonal meshes. *ACM Computer Graphics (Proc. SIGGRAPH 2001)* 2001;417–24.
- [7] Eck M, DeRose T, Duchamp T, Hoppe H, Lounsbery M, Stuetzle W. Multiresolution analysis of arbitrary meshes. *ACM Computer Graphics (Proc. SIGGRAPH '95)* 1995;173–82.
- [8] Lee AWF, Sweldens W, Schröder P, Cowsar L, Dobkin D. MAPS: multiresolution adaptive parameterization of surfaces. *ACM Computer Graphics (Proc. SIGGRAPH '98)* 1998;95–104.
- [9] Floater MS. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design* 1997;14(3):231–50.
- [10] Hormann K, Greiner G. MIPS: an efficient global parameterization method. In: Laurent P-J, Sablonnière P, Schumaker LL, editors. *Curve and surface design: saint-malo 1999*. Nashville: Vanderbilt University Press, 2000. p. 153–62.
- [11] Sheffer A, de Sturler E. Surface parameterization for meshing by triangulation flattening. In: *Proc 9th International Meshing Roundtable*, Sandia National Laboratories, 2000. p. 161–72.
- [12] Tutte W. How to draw a graph. In: Birch B, Erdmann K, editors. *Proceedings of the London Mathematical Society*, vol. 13, 1963. p. 743–68.
- [13] Duchamp T, Certain A, Derose A, Stuetzle W. Hierarchical computation of pl harmonic embeddings. Technical Report, University of Washington, July 1997.
- [14] Floater MS. Parametric tilings and scattered data approximation. *International Journal of Shape Modeling* 1998;4:165–82.
- [15] Floater MS, Reimers M. Meshless parameterization and surface reconstruction. *Computer Aided Geometric Design* 2001;18:77–92.
- [16] Praun E, Sweldens W, Schröder P. Consistent mesh parameterizations. *ACM Computer Graphics (Proc. SIGGRAPH 2001)* 2001;179–84.
- [17] Sederberg TW, Gao P, Wang G, Mu H. 2-D shape blending: an intrinsic solution to the vertex path problem. *ACM Computer Graphics (Proc. SIGGRAPH '93)* 1993;15–8.
- [18] Sander PV, Snyder J, Gortler SJ, Hoppe H. Texture mapping progressive meshes. *ACM Computer Graphics (Proc. SIGGRAPH 2001)* 2001;409–16.