Fast Isometric Parametrization of 3D Triangular Mesh

Xianfang Sun[1,2] and Edwin R. Hancock[2]
[1] School of Automation Science and Electrical Engineering
Beihang University, Beijing 100083, P.R. China
[2] Department of Computer Science, University of York
York YO10 5DD, UK
{xfsun, erh}@cs.york.ac.uk

Abstract

In this paper we describe a new mesh parametrization method that is both computationally efficient and yields minimized distance errors. The method has four steps. First, the multidimensional scaling is used to locally flatten each vertex. Second, an optimal method is used to compute the linear reconstructing weights of each vertex with respect to its neighbours. Thirdly, a spectral decomposition method is used to obtain initial 2D parametrization coordinates. Fourthly, we rotate and scale the initial coordinates to minimize the distance errors. Examples are provided to show the effectiveness of this parametrization method compared with alternatives.

1 Introduction

Triangular mesh parametrization aims to determine a 2D triangular mesh with its vertices, edges, and triangles corresponding to that of the original 3D triangular mesh, satisfying an optimality criterion. The technique has been applied in a wide range of problems in computer graphics and image processing, including texture mapping [12], morphing [8], and remeshing [5]. Extensive research has been undertaken into the theoretical issues underpinning the method and its practical application. For a tutorial and survey, the reader is referred to [4].

A well-known parametrization method is that proposed by Floater [2]. It is a generalization of the basic procedure originally proposed by Tutte [10] which was used to draw planar graphs. The basic idea underpinning this method is to use the vertex coordinates of the original 3D triangular mesh to compute reconstructing weights of each interior vertex with respect to its neighbour vertices.These weights are subsequently used together with the boundary vertex coordinates on a plane to compute the interior vertex coordinates of a 2D triangular mesh. A drawback of Floater's parametrization method is that the boundary vertex coordinates must be determined manually beforehand.

Another parametrization method is that proposed by Zigelman et al. [12]. It first uses Dijkstra algorithm to compute the geodesic distances between each pair of the vertices, and then uses multidimensional scaling (MDS) to determine the vertex coordinates on a 2D plane. This method does not need the boundary vertex coordinates to be determined

BMVC 2005 doi:10.5244/C.19.11

manually beforehand. However, it is highly time-consuming because it needs to compute the geodesic distances between every pair of vertices.

In fact, the parametrization method proposed by Zigelman et al. [12] is a special case of a newly emerging nonlinear dimensionality reduction method, Isomap, proposed by Tenenbaum et al. [9]. Isomap maps high-dimensional data points into a low dimensional space. Naturally, it can be used to map the 3D vertices into a 2D plane as is required in the parametrization of 3D triangular meshes. The only difference between Zigelman's method and Isomap is that the former attempts to improve the precision of the geodesic distance computation by incorporating the geometry of the problem into the solution.

Another important nonlinear dimensionality reduction method is the Locally Linear Embedding (LLE) method proposed by Roweis and Saul [6]. This method uses an analytical method to compute the low dimensional embedding, and hence is more computationally efficient. Like Isomap, it could also potentially be used for mesh parametrization. Unfortunately, because it rotates and scales the vertex coordinates, it can not preserve inter-vertex distances, and thus its use for mesh parametrization is limited.

Motivated by the basic idea of the LLE method, we propose a new analytical parametrization method in this paper, which has the properties of both fast computation and minimization of the distance errors. This method consists of four steps. We first flatten each sub-mesh which is constructed by a vertex and its neighbours, then we compute the linear reconstructing weights of each vertex with respect to its neighbours. The linear reconstructing weights are further used to compute initial rotated and scaled coordinates, and finally, the initial coordinates are rotated and re-scaled to minimize the inter-vertex distance errors between the 2D triangular mesh and the original 3D mesh. In each step, we use a criterion dictated by the need to minimize the inter-vertex distance errors.

The paper is organised as follows. Section 2 introduces the basic problem and provides an overview of the proposed algorithm. Section 3 describes the method of sub-mesh flattening. Section 4 deals with the computations of linearly reconstructing weights and the initial coordinates. Section 5 describes the computation of the optimal 2D vertex coordinates. In Section 6 we provide two experimental examples to show the properties of the algorithm. Finally, in Section 7 we conclude the paper.

2 Problem and Algorithm Overview

Consider a triangular mesh $T = T(V,E,F,X)$ with vertex set $V = \{i : i = 1,2,...,N\}$ and corresponding coordinate set $X = \{x_i : x_i \in R^d,\ i \in V\}$ ($d = 2$ or 3), edge set $E = \{(i,j) : (i,j) \in V \times V\}$, and triangular face set $F = \{(i,j,k) : (i,j),(i,k),(j,k) \in E\}$. Here an edge (i,j) is represented by a straight line segment between vertices i and j, and a triangular face (i,j,k) is a triangular facet bounded by three edges (i,j), (i,k) and (j,k). When $d = 2$, T is drawn on a plane and represents a planar triangular mesh, while $d = 3$, T is drawn in a 3-dimensional space and represents a 3D triangular mesh. A triangular mesh is called valid if the only intersections between edges are at common end points (vertices) and the only intersections between triangular faces are on the common edges. Hereafter, when a triangular mesh is referred without qualification, it implies that the triangular mesh is valid.

Here the parametrization is made on a valid 3D triangular mesh. A parametrization of a valid 3D triangular mesh $T = T(V,E,F,X)$ is any valid planar triangular mesh $T_p =$

$T_p(V,E,F,Y)$ with $Y = \{y_i : y_i \in R^2,\ i \in V\}$ being the corresponding coordinates of V.

For each vertex i, let $T_i = T_i(V_i,E_i,F_i,X_i)$ be the sub-mesh of T, whose vertex set V_i consists of vertex i and its neighbours in V, and whose edge set E_i consists of edges in E which connect pairs of vertices in T_i. The parametrization algorithm proposed here consists of the following four steps.

- Each submesh T_i is mapped into the 2D plane, such that the distortion to form a local parametrization $T_{Li} = T_{Li}(V_i,E_i,F_i,Y_{Li})$ is minimized;
- For each vertex i, compute the weights $W_{i,j}$ that best linearly reconstruct Y_{Li} from its neighbours;
- Using all the weights $W_{i,j}$ obtained in the above step, compute initial 2D coordinates Y_I such that Y_I give the best reconstruction of their neighbours;
- Rotate and scale the initial 2D coordinates Y_I to obtain new final coordinates Y such that the total distortion is minimized.

Note that the algorithm proposed here is very general. One can have different definitions of the distortion or the quality of the best reconstruction. In the following sections, we propose an algorithm based on some specific definitions of these concepts.

3 Local Parametrization

There are different ways of mapping T_i into the plane [2]. Some of them are unstable when there are large angles between the triangular facets in T_i. Others have poor performance according to some distortion criteria. A potentially good method is that proposed by Welch and Witkin [11], which was also adopted by Floater [2]. Although this method preserves the arc length in each radial direction from the vertex i, there are large errors in the boundary arc lengths. In this section, we propose a mapping method that couples the errors in the radial directions and the boundary arcs. We will apply the classical MDS method to map the 3D mesh T_i into the 2D plane such that the total distortion of geometric distances of all the edges to be minimized.

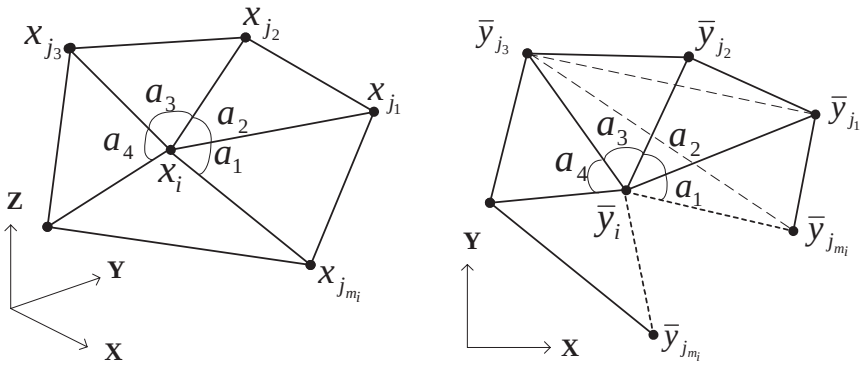
To use the MDS method, we need to compute the geodesic distances between each pair of vertices. Let us first consider an interior vertex i of the 3D triangular mesh (refer to Figure 1(a)). We cut the sub-mesh T_i along any one radial arc (e.g., the edge (i, j_{m_i}) in Figure 1(a)), and then develop it naturally on the plane (Figure 1(b)). The geodesic distances $d_{0,k}$ between vertex i and j_k $(k = 1,\ldots,m_i)$, and $d_{k,k+1}$ $(k = 1,\ldots,m_i - 1)$ and d_{1,m_i} between adjacent vertices are the same as their corresponding Euclidean distances $l_{0,k}$, $l_{k,k+1}$ and l_{1,m_i}. The other required distances between the two boundary vertices j_a and j_b can be computed using the following formula

$$d_{a,b} = \sqrt{l_{0,a}^2 + l_{0,b}^2 - 2l_{0,a}l_{0,b}\cos\alpha_{a,b}}\ , \tag{1}$$

where $\alpha_{a,b}$ is the short path angle between edges (i, j_a) and (i, j_b), which is computed by (without loss of generality, assuming $b > a$)

$$\alpha'_{a,b} = \textstyle\sum_{k=a+1}^{b} \alpha_k,\ \alpha_{\text{sum}} = \sum_{k=1}^{m_i} \alpha_k\ , \tag{2}$$

$$\alpha_{a,b} = \begin{cases} \alpha'_{a,b}\ , & \text{if } \alpha'_{a,b} \le \alpha_{\text{sum}}/2 \\ \alpha_{\text{sum}} - \alpha'_{a,b}\ , & \text{if } \alpha'_{a,b} > \alpha_{\text{sum}}/2 \end{cases}\ , \tag{3}$$



(a) (b)

Figure 1: A sub-mesh T_i (a) and its cut and development on a plane (b)

where α_k $(k = 2, \cdots, m_i)$ is the angle between edges (i, j_{k-1}) and (i, j_k), and α_1 is the angle between edges (i, j_{m_i}) and (i, j_1). The justification of the above computations is clear from Figure 1.

If the vertex i is a boundary vertex, then we can directly develop the sub-mesh T_i on the 2D plane. The computation formula is the same as that for an interior vertex other than that α_1 is equal to the angle between developed edges (i, j_{m_i}) and (i, j_1), and thus $\alpha_{\text{sum}} = 2\pi$.

After all the required geodesic distances have been obtained the classical MDS method proceeds the following steps [1].

- Form the matrix of squared geodesic distances D_I. For our case, D_I is defined as

$$D_I = \begin{bmatrix} d_{0,0}^2 & d_{0,1}^2 & \cdots & d_{0,m_i}^2 \\ d_{1,0}^2 & d_{1,1}^2 & \cdots & d_{1,m_i}^2 \\ \vdots & \vdots & \ddots & \vdots \\ d_{m_i,0}^2 & d_{m_i,1}^2 & \cdots & d_{m_i,m_i}^2 \end{bmatrix} . \tag{4}$$

  Note that because the distance between two vertices is nondirectional, $d_{a,b} = d_{b,a}$, hence the matrix D_I is symmetric;

- Apply double centering to D_I, we obtain

$$B = -\frac{1}{2} J D_I J \ , \tag{5}$$

  where $J = I - \frac{1}{m_i+1}\mathbf{1}\mathbf{1}^T$, I is identity matrix, and $\mathbf{1}$ is a vector of ones of length $m_i + 1$;

- Compute the eigendecomposition of $B = Q\Lambda Q^T$;

- Denote the matrix of the first two largest positive eigenvalues as Λ_+, and the corresponding first two columns of Q as Q_+. The local parametrization coordinates

are now given by $Y_{Li} = Q_+ \Lambda_+^{\frac{1}{2}}$, where the transpose of the first row of Y_{Li} is the coordinate y_{Li} of vertex i, and the transpose of the second to the (m_i+1)'th rows are the coordinates y_{Lj_1} to $y_{Lj_{m_i}}$ of vertices j_1 to j_{m_i}.

Because the classical MDS minimizes the loss function $L(Y_{Li}) = \|\frac{1}{2}J[\hat{D}_I(Y_{Li}) - D_I]J\|$, where $\hat{D}_I(Y_{Li})$ is the squared Euclidean distances computed by Y_{Li}, this local parametrization has the property of minimizing the distance distortion.

4 Linear Reconstruction

This section concerns Steps 2 and 3 of the algorithm. We first discuss Step 2, which is about the computation of the locally linear reconstructing weights. That is, finding $W_{i,j}$ for each vertex i, such that

$$y_{Li} = \sum_{k=1}^{m_i} W_{i,j_k} y_{Lj_k} \ , \ \sum_{k=1}^{m_i} W_{i,j_k} = 1 \ . \tag{6}$$

There are some different methods for computing the weights. Floater proposed a shape-preserving scheme [2] and a mean-value coordinates scheme [3]. The weights computed using these two schemes can result in a good parametrization in the case that the original 3D triangular mesh is an embedding on an intrinsic 2D manifold, and that the boundary vertices can be determined properly. However, these weights do not have a physically meaningful optimal property when the original 3D triangular mesh is an embedding on an intrinsic 3D manifold. In addition, since these two schemes require the boundary vertices to be determined beforehand, they do not provide methods of computing the weights for the boundary vertices.

Roweis and Saul computed the weights using a least-squares method [6]. However, because the solution to the least-squares problem is singular for most of the vertices, the modification introduced in [6] results in weights that are not optimal.

In this section, we introduce a new method of computing the weights, which is optimal in the sense that the coordinates errors are minimized.

From (6), it can be seen that when y_{Lj_k} has error Δy_{Lj_k} $(k = 1, \cdots, m_i)$, the error of y_{Li} will be

$$\Delta y_{Li} = \sum_{k=1}^{m_i} W_{i,j_k} \Delta y_{Lj_k} = \Delta Y_{Lj}^T W_i \ , \tag{7}$$

where $W_i = [W_{i,j_1}, \cdots, W_{i,j_{m_i}}]^T$, and $\Delta Y_{Lj} = [\Delta y_{Lj_1}, \cdots, \Delta y_{Lj_{m_i}}]^T$. The squared magnitude of Δy_{Li} is

$$\|\Delta y_{Li}\|^2 = W_i^T \Delta Y_{Lj} \Delta Y_{Lj}^T W_i \leq \lambda_{\max}(\Delta Y_{Lj} \Delta Y_{Lj}^T) \|W_i\|^2 \ , \tag{8}$$

where $\lambda_{\max}(\cdot)$ is the maximum eigenvalue.

Since the distribution of ΔY_{Lj} is unknown, we can only deal with the worst case error of Δy_{Li} caused by ΔY_{Lj}. From (8) it can be seen that when the magnitude of ΔY_{Lj} is given, the worst case squared error is proportional to $\|W_i\|^2$. Hence, $\|W_i\|^2$ should be minimized to minimize the worst case error. Thus, we need to solve the following minimisation problem

$$\begin{array}{l} \min \|W_i\|^2 \\ \text{s.t.} \quad y_{Li} = \sum_{k=1}^{m_i} W_{i,j_k} y_{Lj_k} \ , \ \sum_{k=1}^{m_i} W_{i,j_k} = 1 \end{array} \ . \tag{9}$$

Let $z_i = \begin{bmatrix} y_{Li} \\ 1 \end{bmatrix}$, $Z_j = \begin{bmatrix} Y_{Lj} & \mathbf{1} \end{bmatrix}$, then (9) becomes

$$\begin{array}{l} \min W_i^T W_i \\ \text{s.t.} \quad z_i = Z_j^T W_i \end{array} \quad . \tag{10}$$

The solution of this minimisation problem can be easily obtained using the Lagrange operator method, which yields

$$W_i = Z_j (Z_j^T Z_j)^{-1} z_i \quad . \tag{11}$$

Note that for any interior vertex i of the triangular mesh, the number of its neighbours $m_i \geq 3$, and all of its neighbours are not on the same straight line. Hence, $Z_j^T Z_j$ are not singular, and (11) has a unique solution. For a boundary vertex i, however, the number of its neighbours m_i may be equal to 2. And even if $m_i \geq 3$, its neighbours may be distributed on the same straight line occasionally. On these cases, $Z_j^T Z_j$ is singular, and we can not obtain a unique solution of W_i from (11). To deal with this problem, we introduce a new vertex, which is itself the neighbour of one of the neighbours of vertex i. This new vertex combined with the neighbours of vertex i forms the new set of neighbours, and the new $Z_j^T Z_j$ is nonsingular. A unique solution of W_i can be computed.

After the W_i's have been obtained, we proceed to Step 3.

To obtain the best linear reconstruction of the original 3D triangular mesh, we need to minimize the following cost function

$$\Phi(Y_I) = \sum_{i=1}^{N} \| y_{Ii} - \sum_{k=1}^{m_i} W_{i,j_k} y_{Ij_k} \|^2 \quad , \tag{12}$$

where N is the number of vertices.

This problem is the same as that in [6], and can be solved in the same way. It is clear that all the coordinates, i.e. the Y_{Ii}'s, can be translated by a constant displacement and rotated by a constant angle without affecting the cost. We remove the translational and rotational degrees of freedom by forcing the following constraints.

$$\sum_{i=1}^{N} y_{Ii} = 0 \quad , \tag{13}$$

$$\frac{1}{N} \sum_{i=1}^{N} y_{Ii} y_{Ii}^T = I \quad , \tag{14}$$

Note that in the second constraint, we have fixed the scale of coordinates to avoid degenerate solutions.

Let $M = (I - W)^T (I - W)$, where matrix $W = [W_{i,j}]$ with $W_{i,j} = 0$ for j not being a neighbour of i. Spectral decomposition of M gives the lowest 3 eigenvectors (corresponding to smallest 3 eigenvlues). The lowest eigenvector of M has eigenvalue 0 and corresponds to the all-ones vector, is discarded here. The remaining two eigenvectors then form the initial 2D coordinates Y_I.

5 Global parametrization

The initial coordinates Y_I obtained in the last section have the property that they are the optimal linear reconstructions of the original 3D triangular mesh subjecting to scaling the

coordinates. When we use Y_I to construct a 2D triangular mesh, the shape of the mesh can best approach that of the original 3D triangular mesh. However, because of the scaling, the errors of edge lengths between the original mesh and the new one may be significant. To obtain a global optimal parametrization, these errors should be minimized. In this section, we rotate and re-scale Y_I to generate new coordinates, such that the errors of edge lengths are minimized.

Let the rotation matrix be R and the diagonal re-scaling matrix be S, then for each y_{Ii}, we can compute a new coordinate y_i by

$$y_i = SRy_{Ii} \quad . \tag{15}$$

To minimize the errors of edge lengths, we use the following cost function

$$\Psi = \sum_{(i,j)\in E} (d_{i,j}^2 - l_{i,j}^2)^2 \quad , \tag{16}$$

where $d_{i,j}$ and $l_{i,j}$ are the lengths of the edge (i, j) in the 2D and 3D triangular mesh, respectively. From (15), we have

$$\begin{aligned} d_{i,j}^2 &= (y_i - y_j)^T (y_i - y_j) = (y_{Ii} - y_{Ij})^T R^T S^2 R (y_{Ii} - y_{Ij}) \\ &= (y_{Ii} - y_{Ij})^T A (y_{Ii} - y_{Ij}) = (y_{Ii1} - y_{Ij1})^2 A_{11} \\ &+ 2(y_{Ii1} - y_{Ij1})(y_{Ii2} - y_{Ij2}) A_{12} + (y_{Ii2} - y_{Ij2})^2 A_{22} \end{aligned} \quad , \tag{17}$$

where we have defined

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{12} & A_{22} \end{bmatrix} = R^T S^2 R \quad . \tag{18}$$

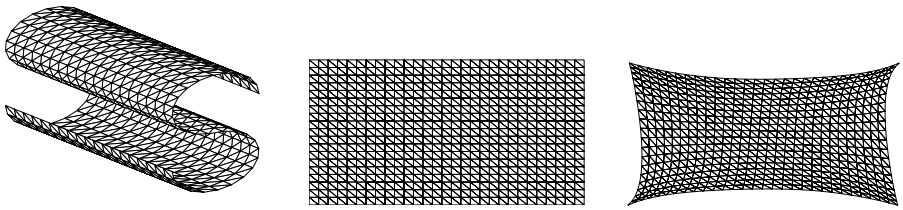
Substitute (17) into (16), we can see that minimizing (16) becomes a least-squares problem, and we can easily obtain (A_{11}, A_{12}, A_{22}) through solving the least-squares problem. Then according to (18), S and R can be obtained through spectral decompotion of matrix A. After S and R are obtained, y_i can be computed through (15). Finally, a parametrization $T_p = T_p(V, E, F, Y)$ is obtained, which has the property that the global edge lengths errors are minimized.

6 Examples

In this section, two examples are provided to illustrate some of the properties of the algorithm proposed in this paper. We will compare the results of our method with that of the Isomap method [9] using two measures – the isometric precision and the CPU time cost. Note that here we have not compared our method with Zigleman's method [12] because in our experimental cases, the properties of Zigleman's method are worse than that of the Isomap method. We have also not compared our method with Floater's [2, 3] because an apparent advantage of our method over Floater's is that Floater's method requires that the boundary vertices of the parametrization mesh be determined manually beforehand, while ours does not.

In the first example, we consider an S-shaped manifold [7]. It is an intrinsically two dimensional manifold. We have performed experiments on this manifold in two different sampling cases. In the first case, we have sampled points on the manifold regularly, and constructed the Delaunay triangulation of the sample points. Experiments with different

numbers of points show that the proposed algorithm results in a perfect parametrization. As a matter of fact, in all the experiments using our algorithm, the residual variances of the edge lengths are of an order of magnitude less than 10^{-21}, while the residual variances using the Isomap method are of the order of magnitude $10^{-3} \sim 10^{-4}$. Figure 2(a) shows a regular sample of $N = 600$ data points and its triangulation in the 3D space. Figure 2(b) and (c) show its parametrization using our method and the Isomap method, respectively. It can be seen that our result is almost the same as the development of the original 3D triangular mesh, and obvious errors appear in the result by the Isomap method.

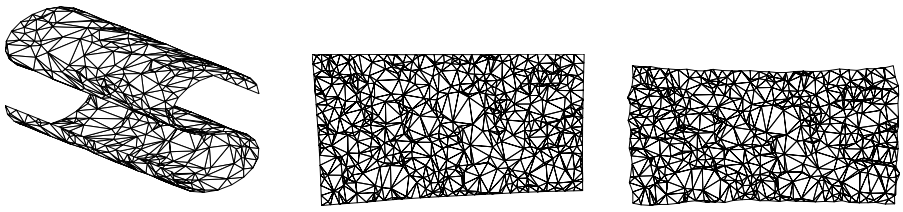


(a) (b) (c)

Figure 2: Parametrization of a regular mesh (a) using our method (b) and the Isomap method(c)

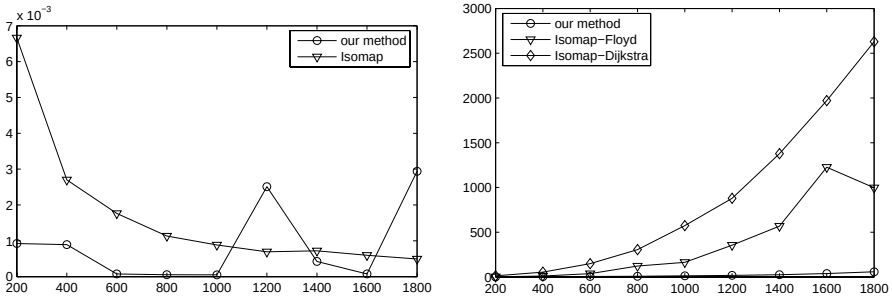
In the second case of the first example, we have sample points on the S-shaped manifold randomly, and again constructed the Delaunay triangulation of the sample points. Figure 3(a) shows a random sample of $N = 600$ data points on the S-shaped manifold. Figure 3(b) and (c) show its parametrization using our algorithm and the Isomap method, respectively. It can be seen that the appearance of the parametrization using our method is closer to the development of the original mesh than that using the Isomap method. Figure 4 (a) gives a comparison of the residual variances of our method and the Isomap method for different numbers of points in the random sampling case. It can be seen that the precision of our method is mostly better than that of the Isomap method. Figure 4 (b) gives a comparison of the CPU time cost by both methods. To be fair, we have used Matlab 7.0 to code both methods with no special coding optimization for either of the methods. For the isomap method, we have used both Floyd's algorithm and Dijkstra's algorithm to compute the geodesic distances. The times shown in the figure were obtained when the programs were run on a PC with Pentium 4 CPU 2.40GHz and 2.41GHz, 512MB of RAM. It can be seen that the CPU time cost by the Isomap method is significantly more than that used by our method, and with an increasing number of vertices, the ratio of CPU times grows rapidly. It can also be seen that in the Isomap method, the Dijkstra algorithm is more time-consuming than Floyd's algorithm.

The second example uses the *peaks* function of Matlab. We have scaled down the z-coordinates by $1/3$ for an efficient parametrization. Figure 5(a) shows a regular triangulation of this function. It consists of 1681 vertices and 4880 edges. Figure 5(b) and (c) show its parametrization using our algorithm and the Isomap method, respectively. It can be seen that the Isomap method yields a parametrization with some edges folded over, while ours does not. The CPU time cost by our method is 20.94 seconds, while that of the Isomap method using Floyd's and Dijkstra's algorithms for geodesic distance computation are 633.97 and 2263.42 seconds, respectively. Even using the faster Floyd algorithm, the Isomap method costs more than 30 times the CPU time as our method does.



(a) (b) (c)

Figure 3: Parametrization of an irregular mesh (a) using our method (b) and the Isomap method(c)



(a) (b)

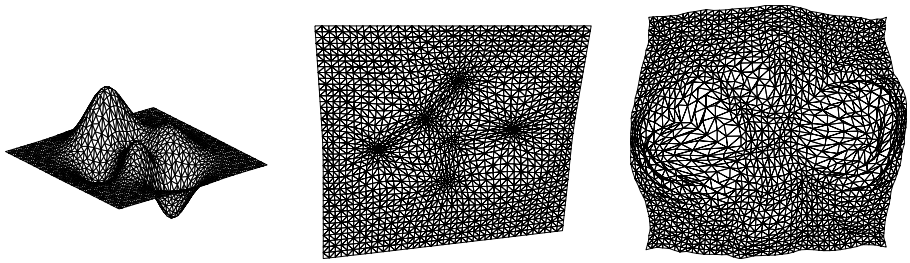
Figure 4: Comparison of the residual variance (a) and the CPU time (seconds) (b)

Interestingly, however, the residual variance of the Isomap method is less than that of our method. They are 3.985×10^{-3} and 5.081×10^{-3}, respectively.

We have also made experiments on irregular triangulation of the *peaks* function. Similar conclusions are obtained to those from the experiments on the regular triangulation. In addition, we have used the results of parametrization in texture mapping. It was shown that our result of texture mapping has less deformation, although it inflates at locations close to the peak points. The result of using the Isomap method gives apparent deformations, especially at positions close to the peak points. Because of space limitations, we do not show these experimental results here.

7 Conclusion

The parametrization method proposed by Floater [2, 3] has the drawback that the boundary vertex coordinates must be determined manually beforehand. Moreover, the parametrization method proposed by Zigelman et al. [12], which is similar to the Isomap method, has the drawback that it is highly time-consuming. In this paper, we have proposed a new parametrization method to overcome these two drawbacks. The method is divided into four steps, and in each step, a specific criterion is chosen to ensure that the final errors of edge lengths between the original triangular mesh and the parametrized one are minimized. Examples show that the proposed parametrization method can effectively overcome the two drawbacks, and that it is practically useful.



(a) (b) (c)

Figure 5: Parametrization of an irregular mesh (a) using our method (b) and the Isomap method(c)

References

[1] I. Borg and P. Groenen. *Modern Multidimensional Scaling Theory and Application*. Springer, 1997.

[2] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Comp. Aided. Geom. Design*, 14:231–250, 1997.

[3] M. S. Floater. Mean value coordinates. *Comp. Aided. Geom. Design*, 20:19–27, 2003.

[4] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 157–186. Springer-Verlag, Heidelberg, 2004.

[5] E. Praun and H. Hoppe. Spherical parametrization and remeshing. *ACM Trans. Graphics*, 22:340–349, 2003.

[6] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[7] L. K. Saul and S. T. Roweis. An introduction to locally linear embedding. Technical report, AT & T Labs-Research, 2000.

[8] V. Surazhsky and C. Gotsman. Intrinsic morphing of compatible triangulations. *Int. J. Shape Modelling*, 9:191–201, 2003.

[9] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[10] W. T. Tutte. How to draw a graph. *Proc. London Math. Soc.*, 13:743–768, 1963.

[11] W. Welch and Witkin A. Free-form shape design using triangulated surfaces. In *Computer Graphics, SIGGRAPH 94*, pages 247–256. 1994.

[12] G. Zigelman, R. Kimmel, and Kiryati N. Texture mapping using surface flattening via multi-dimensional scaling. *IEEE Trans. Visualization Comp. Graphics*, 8:198–207, 2002.