Texture mapping on 3D surfaces using clustering-based cutting paths

Tong-Yee Lee* and Shao-Wei Yen

Computer Graphics Group/Visual System Lab (CGVSL),
Department of Computer Science and Information Engineering,
National Cheng-Kung University, Tainan,
Taiwan, Republic of China
Email: tonylee@mail.ncku.edu.tw
*Corresponding author

Abstract: Texture mapping is a common technique in computer graphics used to render realistic images. Our goal is to achieve distortion-free texture mapping on arbitrary 3D surfaces. To texture 3D models, we propose a scheme to flatten 3D surfaces into a 2D parametric domain. Our method does not require the 2D boundary of flattened surfaces to be stationary. The proposed method consists of three steps: (1) we find high distortion areas in a 2D parametric domain and find a cutting path over these areas, (2) we add virtual points to adaptively find the better parametric domain boundary instead of a predefined boundary and (3) we perform a well-known smoothing technique for better texture mapping. The proposed scheme can be efficiently realised by a linear system and yields interactive performance. Several experimental results for both genus-0 and non-genus-0 models are presented to verify the proposed scheme.

Keywords: clustering; cutting path; parameterisation; texture stretch; virtual points.

Reference to this paper should be made as follows: Lee, T-Y. and Yen, S-W. (2007) ‘Texture mapping on 3D surfaces using clustering-based cutting paths’, *Int. J. Computational Science and Engineering*, Vol. 3, No. 1, pp.71–79.

Biographical notes: Prof. Tong-Yee Lee received his PhD in computer engineering from Washington State University, Pullman, in May 1995. Now, he is a Professor in the department of Computer Science and Information Engineering at National Cheng-Kung University in Tainan, Taiwan, Republic of China. He serves as an associate editor for IEEE Transactions on Information Technology in Biomedicine from 2000 to 2007. He is also on the Editorial Advisory Board of Journal Recent Patents on Engineering, editor for Journal of Information Science and Engineering and region editor for Journal of Software Engineering. His current research interests include computer graphics, non-photorealistic rendering, image-based rendering, visualization, virtual reality, surgical simulation, medical visualization and medical system, distributed & collaborative virtual environment. He leads a Computer Graphics Group/Visual System Lab at National Cheng-Kung University. He is a member of the IEEE and ACM.

Shao-Wei Yen received the BS degree in civil engineering from National Taiwan University, Taiwan, in 2000. He is currently working toward the PhD degree in the Department of Computer Science and Information Engineering, National Cheng-Kung University. His research interests include computer graphics and mesh parameterisation.

1 Introduction

Texture mapping is one of the most important techniques in computer graphics. It can enhance the reality of computer-generated 3D models. The most common way to texture 3D surfaces is to map the surface into an isomorphic planar representation. This operation is called surface parameterisation or surface flattening. This planar representation is called embedding. Except for simple surfaces, such as cylinders, the surface parameterisation always creates surface distortions in the 2D parametric domain. If we directly texture map this 2D parameterisation, the texture mapped onto a 3D surface is also distorted. It is a well-known differential geometry theorem that no isometric parameterisation is in the plane for a general surface patch (Ahlfors and Sario, 1960). Hence, there is no easy way to parameterise a general 3D surface over a 2D domain without introducing distortion. In this paper, the proposed method attempts to minimise the distortion in surface parameterisation and produces a distortion-free texture mapping on arbitrary 3D surfaces.

There have been many texture mapping methods proposed (Tutte, 1960; Maillot, Yahia and Verroust, 1993; Pinkall and Polthier, 1993; Floater, 1997; Hormann and Greiner, 2000; Piponi and Borshukov, 2000; Sander et al., 2001; Gu, Gortler and Hoppe, 2002; Lévy et al., 2002; Sheffer and Sturler, 2002; Sorkine et al., 2002; Zigelman, Kimmel and Kiryati, 2002; Lee and Huang, 2003). Most of them discuss the issue of surface parameterisation. In general, the topology of 3D surface meshes always are the same with a disk, because the connectivity of 3D surface meshes will have no changes and can be mapped to a plane during the surface parameterisation process. Maillot, Yahia and Verroust (1993) group the facets by their normals and attempt to reduce distortion by minimising a norm of the Green-Lagrange deformation tensor. Eck et al. (1995) use harmonic maps to minimise the distortion energy. Float (Floater, 1997) maps a disk-like open mesh to a plane. This approach can be solved efficiently by solving a linear system, however, the boundary vertices are required to be on a convex 2D polygon. Hormann and Greiner (2000) derive their deformation function from the ratio of singular values about the mapping function. This method does not require the boundary vertices to be fixed onto a convex 2D polygon. However, its computational cost is expensive. Sander et al. (2001) define a non-linear texture stretch metric for surface parameterisation. This approach uses a relaxation approach to iteratively flatten the 3D surfaces. Their texture stretch metric produces better results than those of several previous methods (Maillot, Yahia and Verroust, 1993; Eck et al., 1995; Hormann and Greiner, 2000). However, this technique may need lots of iterations for computing the optimal mapping. Sorkine et al. (2002) introduce a bounded-distortion parameterisation to guarantee that each triangle's distortion will strictly be below a user-defined threshold. This method does not need to map the surface into a convex region and automatically maps the boundary to the plane. For complicated 3D models, it would create many patches and become more discontinuous in texture mapping. Zigelman, Kimmel and Kiryati (2002) map the open mesh onto a plane using a multi-dimensional scaling method that tries to preserve the geodesic distance criterion. This method does not ensure no self-intersections between the triangles and the computation cost is expensive because of the geodesic distance calculations. Lévy et al. (2002) approximate the conformal maps with least square sense. This mapping approach does not need to fix boundary and the computation can be performed efficiently. However, this approach may generate fold-over maps after parameterisation.

Among the previous works, the proposed scheme is most related to (Piponi and Borshukov, 2000; Gu, Gortler and Hoppe, 2002). Piponi and Borshukov (2000) cut the closed surface and then use the cut seams as the boundaries for parameterisation. They also blend the seams to reduce the discontinuity effect. This method may need lots of user-interactions to define the boundaries to the 2D plane. In contrast to Piponi and Borshukov (2000), our proposed method would avoid iterations using a standard numerical method to solve the linear system quickly. Gu, Gortler and Hoppe (2002) propose a method for cutting through high distortion area in the parametric domain. They use an iteration procedure that finds only one path to cut at a time until the distortion is below some prescribed threshold value. In contrast to this method, our proposed method can cut over all high distortion regions using a clustering strategy. In addition, there have been many clustering approaches proposed in computer graphics research. Kalvin and Taylor (1996) merge faces into face clusters to achieve the mesh simplification. Willmott, Heckbert and Garland (1999) propose a hierarchical face-clustering algorithm that provides an efficient radiosity simulation. Sander et al. (2001) apply the merge operation to partition meshes into charts to their progressive mesh parameterisation.

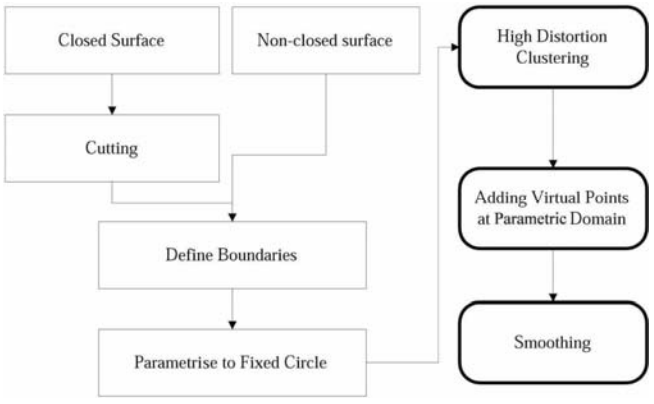
2 Methodology

Generally, parameterisation techniques can be classified into two types: (1) fixed boundaries in the parametric domain and (2) non-fixed boundaries. The methods with fixed boundaries often have large distortions because the boundary shapes of the original 3D models are very different from those of their flattened surfaces in the 2D domain. The proposed method can fast flatten arbitrary 3D surfaces without the fixed boundary constraints. The main contribution of this paper is to present a faster and a lower texture stretch scheme for texture mapping on 3D surface. To reduce texture stretch, we propose a clustering strategy to find a cutting path for reparameterisation and apply virtual points and smoothing for further reduction in texture stretch. In addition, the proposed method can handle both genus-0 and non-genus-0 models.

2.1 The initial set-up

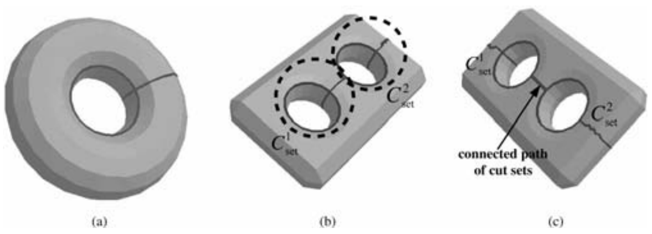
Figure 1 is the flow chart of the proposed scheme. We will explain in detail for the last three steps in later sections.

Figure 1 Flow chart of the proposed scheme


Closed Surface
Non-closed surface
High Distortion Clustering
Cutting
Adding Virtual Points at Parametric Domain
Define Boundaries
Parametrise to Fixed Circle
Smoothing

To flatten a model into the 2D domain, we require the 3D model topology to be consistent with an open disk. If the 3D model has a closed surface, we must first cut the model to make the topology equivalent to an open disk. The user can interactively perform the cut procedure in our system. For models with genus greater than 1, we use some guide rules to cut all of them to a disk-like surface described as follows. For a torus with genus 1, which has one hole, a valid cut exists and consists of two closed paths like Figure 2(a). The two cut paths form the primitive cut set C_{set} for one hole. For a model with genus 2 like Figure 2(b), we need two cut sets C^1_{set} and C^2_{set} to cut the surface. Once the two cut sets do not connect to each other, we add a path with no self-intersection to link the two sets. The surface with genus 2 can then be cut to a topological disk (see Figure 2(c)). By applying the same rule, a model with genus *g* must first find *g* primitive cut sets $C^1_{\text{set}}, C^2_{\text{set}}, \ldots, C^g_{\text{set}}$ and then paths are added to link all of them.

Figure 2 (a) a torus model with one cut set, (b) connected cut sets and (c) unconnected cut set



After cutting a 3D model to a disk-like surface, we utilise a relaxation-based scheme from our previous work (Lee and Huang, 2003) to parameterise the 3D model into the 2D circular embedding. This surface parameterisation can be solved using a sparse linear system. The relaxation equation is shown in Equation (1). To simulate a spring-mass system, we select *w* as the inverse of the edge length in 3D. Later in Section 2.3, we will explain the choice of the spring-mass system.

$$p_i' = (1-\lambda)p_i + \lambda \frac{\sum_{j=1}^{k_i}(w_j p_j)}{\sum_{j=1}^{k_i} w_j} \tag{1}$$

where *w*, weight; λ, the speed of relaxation; p_j, 1-ring neighbouring vertices of p_i.

2.2 *Reducing distortion using clustering and cutting*

After the surface is parameterised into a circular embedding using the relaxation-based scheme, we find many vertices clustered in some areas of the embedding.

These clustered areas always have higher parametric distortion. Gu, Gortler and Hoppe (2002) suggest using cutting to reduce the distortion. Their parameterisation approach iteratively finds regions of maximal distortion and connects the regions to the boundary using a path. However, this is very computationally expensive to repetitively execute parameterisation and find each region with maximal distortion. In contrast, the proposed method does not require repetitive surface parameterisation. We propose a clustering algorithm to find a cutting path going through the distorted areas. We then connect the cutting path to the boundary, cut the model and then execute the surface parameterisation. Our clustering scheme is described in the following:

2.2.1 *Clustering algorithm*

1. Determine the vertices that would be joined to the computation of clustering algorithm. In our implementation, we calculate texture stretch *L* (Sander et al., 2001) of all vertices. If its texture stretch value is larger than the mean of all vertices, it would be added into a queue for clustering calculation. The texture stretch *L* can be defined in the following equation:

   $$L(T) = \sqrt{(\Gamma^2 + \gamma^2)/2}$$

   where Γ and γ represent the largest and smallest scale value while mapping a unit length from parametric domain to 3D surface domain, and *T* is a triangle of the mesh.

2. Define a Guassian function G_R that indicates the effective region of a cluster.

   $$G_R(C - P_i) = e^{-\|C-P_i\|^2/R^2}$$

   C: vertex uv position on the parametric domain

   P_i: the position of *i*-th cluster's centre

   R: the cluster radius.

3. Initialise the first cluster. We choose the vertex with the maximum texture stretch value to be the centre of the first cluster.

4. Calculate the centres of the other clusters

   ```
   repeat
       arg_j max ($N(C_j; P_1,\ldots, P_M$, j =1…$n$))
   until ($N(C_j; P_1,\ldots, P_M) \le \varepsilon$
   ```

   P_i: clusters which have already been determined

   C_j: uv position of the vertex *j*

   ε: defined value for the termination of repeat loop

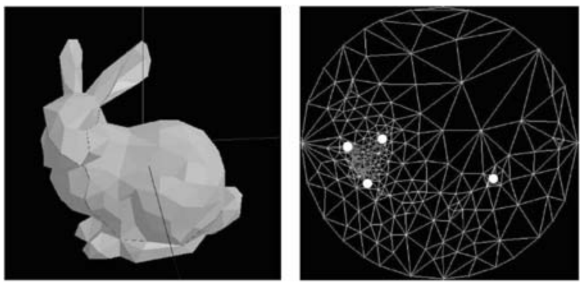
   The definition of function *N* is below:

   $$N(C_j; P_1, \ldots P_M) = \prod_{k=1}^{M}[1 - G_R(C - P_k)]$$

   where the *N* function can be thought as the probability that vertex C_j is not be- longing to other clusters $P_1,\ldots,P_M$.

From the above procedure, we determine all cluster centres. For each loop, the proposed clustering algorithm will determine the vertex that is the most irrelevant to the other clusters. If the function *N* value is less than the user-defined threshold ε, the clustering procedure would be terminated. Figure 3 shows an example of clustering and three centres are found.

Figure 3 A bunny model and its clustering result (solid white points are the centre of the clusters)



2.2.2 Minimal spanning tree cutting

After executing clustering algorithm, we need to get a path to connect all clusters and connect this path to the boundary in the embedding. For this purpose, we find a tree to connect these cluster centres. In our implementation, we adopt the standard Kruskal's minimal spanning tree algorithm. To assign the weight of a graph edge, we combine both the edge length and the texture stretch together. We attempt to find a minimal spanning tree that is over high distortion areas and has the minimal path length. The cutting path is founded using the following steps:

1 For each pair of clusters, we compute the shortest path. The edge weight is set using the following equation:

  Edge Weight = 3D length + 1/(texture stretch)

2 We then obtain a complete graph for connecting all clusters

3 Calculate the minimal spanning tree of this complete graph.

2.3 Virtual points

Once the cutting path is found, we cut the model and reparameterise the model to a circular embedding again. At this stage, the boundary of the embedding is still fixed. In this section, we will add virtual points to pull the boundary of this circular embedding and to adaptively modify the boundary for reducing distortion.

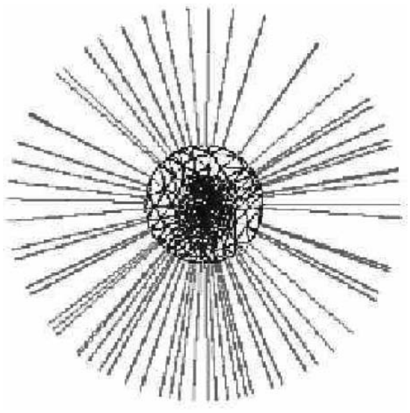
The circular embedding in our system is a simulation of the spring-mass system. Each edge of a 3D model is placed with a spring. The potential energy of a spring is equal to $E = (1/2)Kx^2$. K is the spring constant and x is the length of the spring. In our set-up, K is equal to the inverse of its corresponding 3D edge length. Then the total energy of the whole mesh is

$$E_{\text{total}} = \sum_{(v_i,v_j)\in\text{Edge}} \frac{1}{2} K_{ij} (f(v_i)) - (f(v_i))^2, \qquad (2)$$

where $f(v)$ is the uv position in the 2D parametric domain. If the system reaches stable equilibrium, the total energy E_{total} would be minimised. Then the minimisation of the total energy can be realised using a linear system (Greiner and Hormann, 1997).

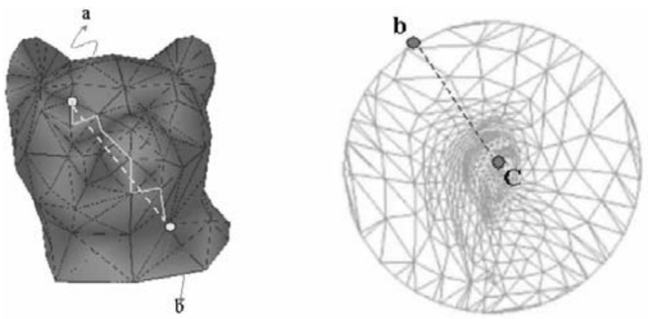
In the outside of the circular embedding, for each boundary vertex, we put a corresponding vertex called virtual point along the direction from the centre of the circle to the boundary vertex (Figure 4). We add a virtual spring between a boundary vertex and a virtual point. The spring constant of each virtual point is in the proportion to the inverse of the 3D length the distance of the shortest path from a boundary vertex to the centre. The virtual springs then pull each boundary vertex outward. With adding these extra virtual points, we have the flexibility to redefine the boundary positions and can potentially reduce the distortion of parameterisation.

Figure 4 The vertices of outside circle is virtual points



For calculating the spring constants of boundaries, we may directly apply the Dijkstra's single source shortest path algorithm to compute the 3D surface length. Because the path computation is limited on triangle edges, this method may not always produce good and smooth results (Figure 5, left). For this problem, we can subdivide the 3D mesh to increase the accuracy of the 3D path length, but it is computationally expensive. Alternatively, we find each path directly from the embedding by a line connects the centre to each boundary vertex (see an example in Figure 5, right). We then find line intersections and compute their corresponding 3D positions. In this manner, we can obtain an approximately shortest path in 3D. Finally, similar to (Lee and Huang, 2003), we can organise this spring-based system as a sparse linear system and we can solve it numerically.

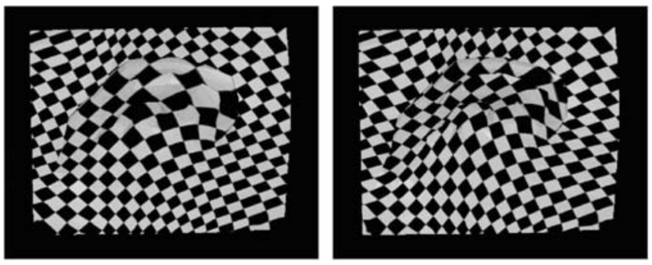
Figure 5 The line is between a boundary vertex and the centre of the embedding



2.4 Smoothing

Smoothing is the last step of our procedure. The main smoothing concept involves warping the texture used for texture mapping. The mapping distortion can then be further improved. Figure 6 illustrates the effects of smoothing for texture mapping. In the current implementation, we do not actually warp the texture. Alternatively, we further smooth the embedding by the method in (Sheffer and Sturler, 2002). However, we adopt texture stretch (Sander et al., 2001) to be our distortion ratio instead of the edge ratio of 2D edge length and 3D edge length in (Sheffer and Sturler, 2002).

Figure 6 Left: texture mapping without smoothing. Right: texture mapping with smoothing



The smoothing procedure is described briefly as follows:

1 Create a bounded square *B* for the flattened map, and make a Cartesian grid *G* as shown in Figure 7. Consequently, we can easily warp the texture map by the grid map *G*.

2 For each grid point N_i, compute the texture stretch L_i. The L_i is the texture stretch for the triangle encloses the grid point.

3 For each grid edge $e = (N_a, N_b)$, calculate its distortion $l(e) = \frac{L_a + L_b}{2}$. If the grid point N_a (or N_b) is outside the flatten map, we assign the L_a (or L_b) to the average texture stretch on whole model.

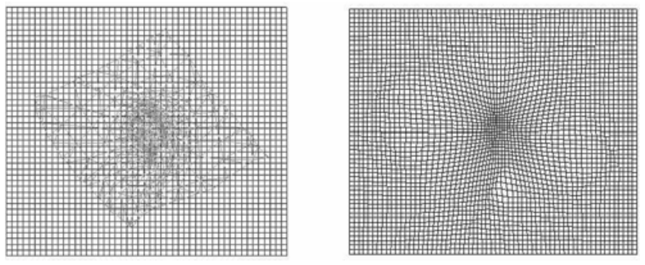
4 For each grid point N_i, compute its position P_i iteratively.

$$P_i = \frac{\sum_{e=(N_i,N_j)} \frac{1}{l(e)} P_j}{\sum_{e=(N_i,N_j)} \frac{1}{l(e)}}$$

where N_j is the neighbour grid points of N_i. The system is similar to parameterisation relaxation scheme, so we can also solve the sparse linear system efficiently.

5 Warp the flattened map by the smoothing texture map (grid map *G*).

Figure 7 Left: a grid map of bounded square; Right: smoothing results of grid map



3 Experimental results

In this section, we first compare our cutting strategy and the cutting method in Gu, Gortler and Hoppe (2002). We compute eight different examples shown in Figures 8–15. And the least three models have topology with genus 1 or genus 2. The statistical results are shown in Table 1. In experimental comparison with (Gu, Gortler and Hoppe, 2002), our improved stretch ratio in Table 1 varies from +7 to +42% except for Figure 9, i.e. –1%. Furthermore, the computation overhead of (Gu, Gortler and Hoppe, 2002) requires much more time than our method. We demonstrate the same eight examples to verify the proposed scheme. In these examples, we show

1 original model

2 circular embedding and cutting path from the proposed clustering scheme

3 embedding after pulling by virtual points and finally

4 model with texture after smoothing.

To better visualise the cutting, we colour the embedding. The region with a warm colour has larger distortion than that of the region with a cool colour. From these figures, the clustering centres are always located in the regions with a warm colour. Furthermore, the paths also go through the warm colour regions. We show the quantity information in Table 2 for these eight examples. In this table, we measure the quality of texture mapping using mean texture stretch ratio (Sander et al., 2001). The ideal mean stretch ratio is equal to one. The experimental measurement shows this stretch metric can be further improved using the consecutive steps of the proposed method. Each step of the proposed method is executed in few seconds on a PC with Pentium III 1 GHz and 256 MB RAM. Therefore, we can interactively perform texture mapping on arbitrary 3D surfaces. The user can also optionally bypass some step of the proposed method if the current results are good enough. For example, we can bypass the smoothing or even the cutting if satisfactory results are obtained without these steps.

Figure 8 Left: rabbit model; Middle left: clusters and the connected minimal spanning tree; Middle right: embedding without a fixed boundary; and Right: rabbit with texture

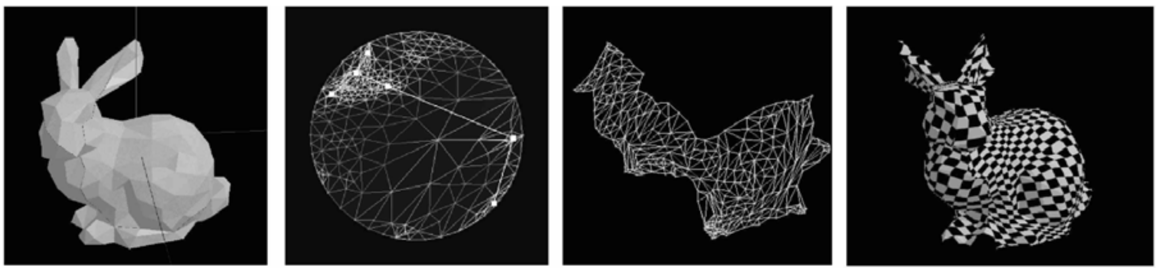


Figure 9 Left: pig model; Middle left: clusters and the connected minimal spanning tree; Middle right: embedding without a fixed boundary; and Right: pig with texture

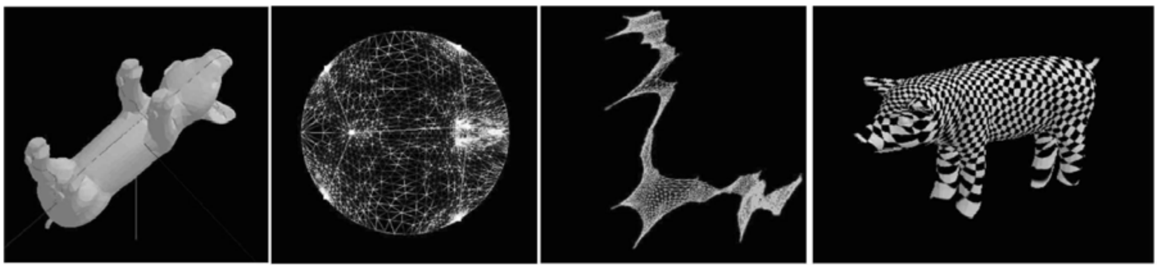


Figure 10 Left: dinosaur model; Middle left: clusters and the connected minimal spanning tree; Middle right: embedding without a fixed boundary; and Right: dinosaur with texture

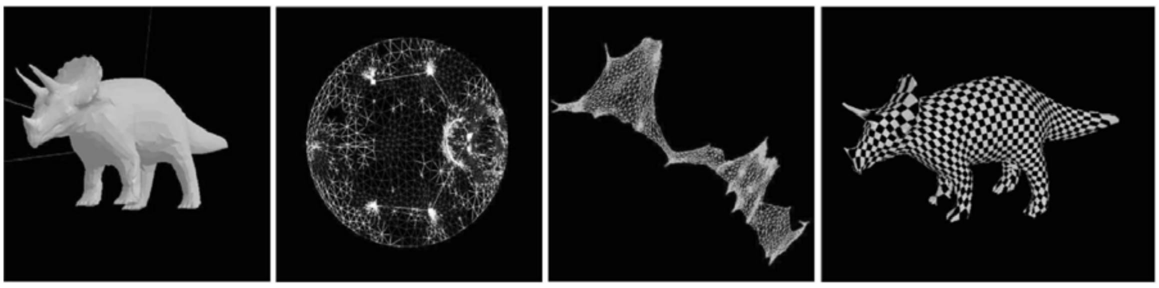


Figure 11 Left: head model; Middle left: clusters and the connected minimal spanning tree; Middle right: embedding without a fixed boundary; and Right: head with texture

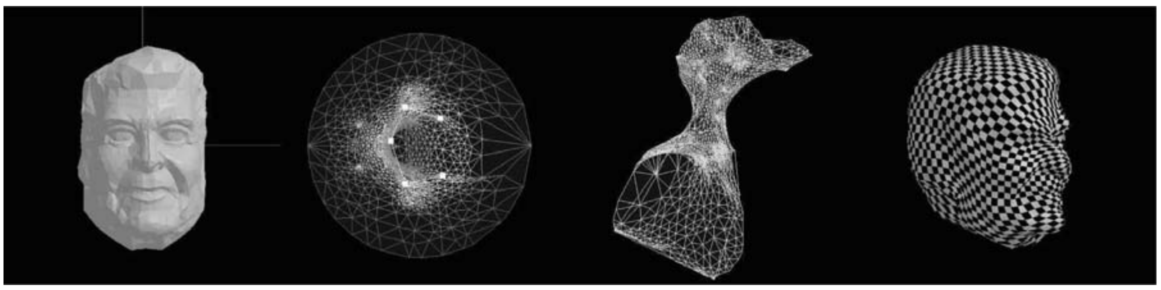


Figure 12 Left: ball model; Middle left: clusters and the connected minimal spanning tree; Middle right: embedding without a fixed boundary; and Right: ball with texture

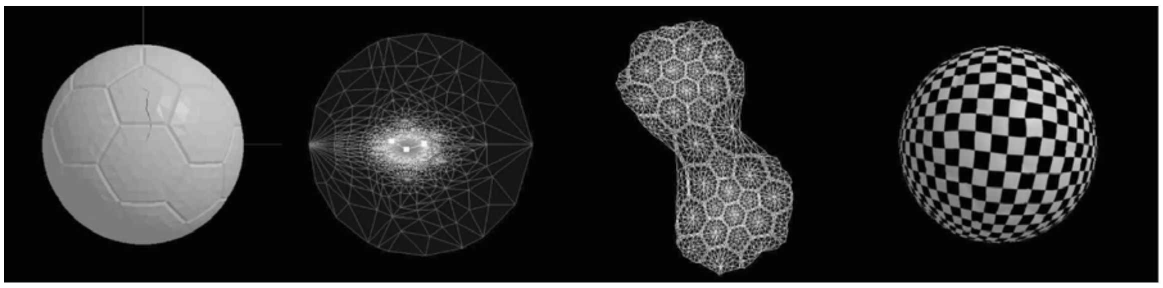


Figure 13 Left: knot model; Middle left: clusters and the connected minimal spanning tree; Middle right: embedding without a fixed boundary; and Right: knot with texture

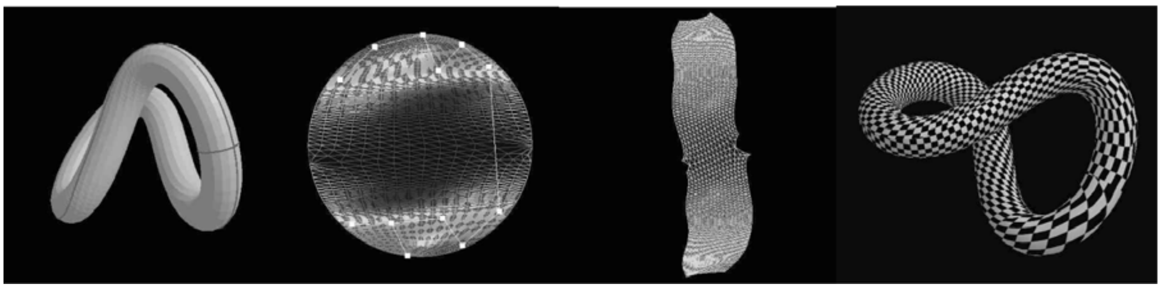


Figure 14 Left: two-holes model; Middle left: clusters and the connected minimal spanning tree; Middle right: embedding without a fixed boundary; and Right: two-holes with texture

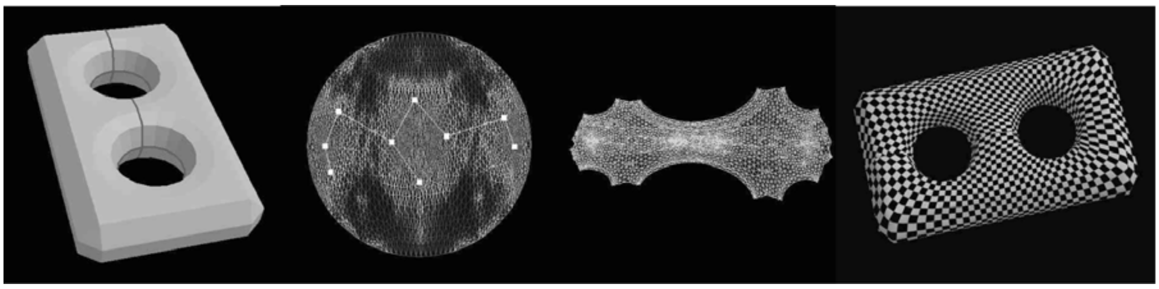


Figure 15 Left: holed-pig model; Middle left: clusters and the connected minimal spanning tree; Middle right: embedding without a fixed boundary; and Right: holed-pig with texture

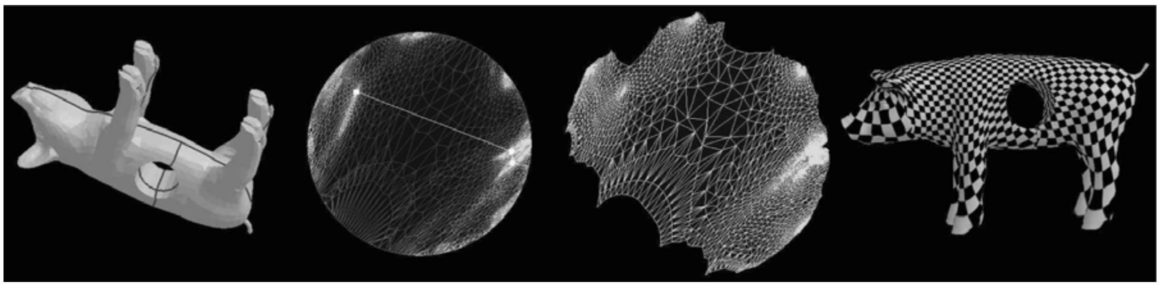


Table 1 Comparison between proposed method and Gu, Gortler and Hoppe (2002)

Model	*Rabbit*	*Pig*	*Dinosaur*	*Head*	*Ball*	*Knot*	*Two-holes*	*Holed-pig*
No. of vertices	263	3584	2917	1954	1760	1920	2522	3865
No. of triangles	522	7164	5660	3904	3516	3840	5048	7372
Texture stretch (S1) after cut (Gu, Gortler and Hoppe, 2002)	1.352	1.712	1.936	1.632	1.237	2.346	2.526	2.264
Texture stretch (S2) after cut (our method)	1.256	1.731	1.12	1.285	1.096	1.837	1.603	1.516
improved stretch ratio ((S1-S2)/S1)%	7.1	–1.1	42.1	21.3	11.4	21.7	36.5	33.0
Run time for cut and flattening (Gu, Gortler and Hoppe, 2002)	19 s	372 s	327 s	243 s	139 s	251 s	302 s	403 s
run time for cut and flattening (our methods)	2 s	13 s	10 s	7 s	6 s	7 s	9 s	14 s

Table 2 Mean texture stretch ratio statistics for five examples

Model	*Rabbit*	*Pig*	*Dinosaur*	*Head*	*Ball*	*Knot*	*Two-holes*	*Holed-pig*
No. of vertices	296	3684	2917	1954	1760	1920	2522	3865
No. of triangles	522	7164	5660	3904	3516	3840	5048	7372
Stretch ratio before clustering	25.101	112.353	18.557	3.504	8.711	6.231	4.519	67.765
Stretch ratio after clustering (without pulling by virtual points)	1.707	2.539	2.833	2.451	1.353	4.871	3.107	3.124
Stretch ratio after clustering (with pulling by virtual points)	1.314	2.031	1.851	1.514	1.231	1.951	1.823	1.833
Stretch ratio after smoothing	1.256	1.731	1.12	1.285	1.096	1.837	1.603	1.516

4 Conclusion and future work

In this paper, we propose a texture-mapping scheme on arbitrary 3D surfaces. This new scheme can be solved very efficiently. All experiment results are executed on a PC with Pentium III 1 GHz and 256 MB RAM at an interactive performance. In contrast to (Gu, Gortler and Hoppe, 2002), the computational performance is very promising and texture stretch is improved. The proposed method consists of several steps. These steps consecutively improve the quality of embedding. From the experimental results, the stretch ratio for texture mapping on the embedding is significantly improved by these steps. In the near future, we plan to extend this work to the texture mapping with hard constraint problem. We will apply the proposed method to model remeshing applications. Texture synthesis over arbitrary 3D surfaces is another interesting topic for our future work. Given a small sample texture, this technique attempts to synthesise a seamless any sized texture pattern on 3D surfaces (Wei and Levoy, 2000; Tong et al., 2002).

Acknowledgements

This project is supported by National Science Council, Taiwan, ROC under Contract No. NSC-92-2213-E-006-066 and NSC-92-2213-E-006-067.

References

Ahlfors, L.V. and Sario, L. (1960) *Riemann Surfaces*. Princeton, NJ: Princeton University Press.

Dey, T.K. and Schipper, H. (1995) ‘A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection’, *Discrete and Computational Geometry*, Vol. 14, pp.93–110.

Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M. and Stuetzle, W. (1995) ‘Multiresolution analysis of arbitrary meshes’, Paper presented in the Proceedings of *SIGGRAPH*, pp.173–182.

Floater, M.S. (1997) ‘Parametrization and smooth approximation of surface triangulations’, *Computer Aided Geometric Design*, Vol. 14, pp.231–250.

Gotsman, C., Gu, X. and Sheffer, A. (2003) ‘Fundamentals of Spherical Parameteriza-tion for 3D Meshes’, Paper presented in the Proceedings of *SIGGRAPH*, pp.358–363.

Greiner, G. and Hormann, K. (1997) ‘Interpolating and Approximation Scattered 3D Data with Hierarchical Tensor Product B-splines’, in A. Le Méhauté, C. Rabut and L.L. Schumaker (Eds), *Surface Fitting and Multiresolution Methods* (pp.163–172). Nashville, TN: Vanderbilt University Press.

Gu, X., Gortler, S.J. and Hoppe, H. (2002) ‘Geometry Images’, Paper presented in the Proceedings of *SIGGRAPH*, pp.355–361.

Hormann, K. and Greiner, G. (2000) ‘MIPS: An efficient global parameterization method’, in P-J. Laurent, P. Sablonnière and L.L. Schumaker (Eds), *Curve and Surface Design: St. Malo 1999* (pp.153–162). Nashville, TN: Vanderbilt University Press.

Kalvin, A.D. and Taylor, R.H. (1996) ‘Superfaces: polygonal mesh simplification with bounded error’, *IEEE Computer Graphics and Appl.*, Vol. 16. pp.64–77.

Lee, T-Y. and Huang, P.H. (2003) ‘Fast and intuitive metamorphosis of 3D polyhedral models using SMCC mesh merging scheme’, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, pp.85–98.

Lévy, B., Petitjean, S., Ray, N. and Maillot, J. (2002) ‘Least Squares Conformal Maps for Automatic Texture Atlas Generation’, Paper presented in the Proceedings of *SIGGRAPH*, pp.362–371.

Maillot, J., Yahia, H. and Verroust, A. (1993) ‘Interactive texture mapping’, Paper presented in the Proceedings of *SIGGRAPH*, pp.27–34.

Pinkall, U. and Polthier, K. (1993) ‘Computing discrete minimal surfaces and their conjugates’, *Experimental Mathematics*, Vol. 2, pp.15–36.

Piponi, G. and Borshukov, D. (2000) ‘Seamless Texture Mapping of Subdivision Surfaces by Model Pelting and Texture Blending’, Paper presented in the Proceedings of *SIGGRAPH*, pp.471–478.

Sander, P., Snyder, J., Gortler, S. and Hoppe, H. (2001) ‘Texture mapping progressive meshes’, Paper presented in the Proceedings of *SIGGRAPH*, pp.409–416.

Sheffer, A. and Sturler, E.D. (2002) ‘Smoothing an overlay grid to minimize linear distortion in texture mapping’, *ACM Transactions on Graphics*, Vol. 21, pp.874–890.

Sorkine, O., Cohen-Or, S.D., Goldenthal, R. and Lischinski, D. (2002) ‘Bounded-distortion Piecewise Mesh Parameterization’, Paper presented in the Proceedings of *IEEE Visualization*, pp.355–362.

Tong, X., Zhang, J., Liu, L., Wang, X., Guo, B. and Shum, H-Y. (2002) ‘Synthesis of Bi-directional Texture Functions on Arbitrary Surfaces’, Paper presented in the Proceedings of *SIGGRAPH*, pp.665–672.

Tutte, W. (1960) 'Convex Representation of Graphs', Paper presented in the proceedings of the *London Math. Soc.*, Vol. 10, pp.304–320.

Wei, L-Y. and Levoy, M. (2000) 'Fast texture synthesis using tree-structured vector quan-tization', Paper presented in the Proceedings of *SIGGRAPH*, pp.479–488.

Wei, L-Y. and Levoy, M. (2001) 'Texture synthesis over arbitrary manifold surfaces', Paper presented in the Proceedings of *SIGGRAPH*, pp.355–360.

Willmott, A.J, Heckbert, P.S. and Garland, M. (1999) *Face cluster radi-osity*, Eurographics Workshop on Rendering.

Wood, Z., Hoppe, H., Desbrun, M. and Schröder, P. (2004) 'Removing excess topology from isosurfaces', *ACM Transactions on Graphics*, Vol. 23, pp.190–208.

Zigelman, G., Kimmel, R. and Kiryati, N. (2002) 'Texture mapping using surface flattening via multidimensional scaling', *IEEE Transactions on Visualization and Computer Graphics*, Vol. 8, pp.198–207.