The Visual Computer manuscript No.
(will be inserted by the editor)

Shuhua Lai · Fuhua (Frank) Cheng

Texture Mapping on Surfaces of Arbitrary Topology using Norm Preserving based Optimization

Abstract A simple and yet highly efficient, high quality texture mapping method for surfaces of arbitrary topology is presented. The new method projects the given surface from the 3D object space into the 2D texture space to identify the 2D texture structure that will be used to texture the surface. The object space to texture space projection is optimized to ensure minimum distortion of the texture mapping process. The optimization is achieved through a commonly used norm preserving minimization process on edges of the surface. The main difference here is, by using an initial value approach, the optimization problem can be set up as a quadratic programming problem and, consequently, solved by a linear least squares method. Three methods to choose a good initial value are presented. Test cases show that the new method works well on surfaces of arbitrary topology, with the exception of surfaces with exceptionally abnormal curvature distribution. Other advantages of the new method include uniformity and seamlessness of the texture mapping process. The new method is suitable for applications that do not require precise texture mapping results but demand highly efficient mapping process such as computer animation or video games.

Keywords Texture Mapping · Optimization · Realistic Rendering

1 Introduction

Texture mapping means the mapping of a function from a texture space onto a surface in 3D space [13]. Each point on the object surface is the image of an element in texture space. The domain of the mapping can be one, two, or three-dimensional, and it can be represented by either a discrete array or by a mathematical function.

Texture mapping was first introduced as a method of adding to the visual richness of a computer generated image without adding geometry in [6]. Its use by far is one of the most successful techniques in the quest for more realistic imagery. Texture mapping can enhance the visual effects of raster scan images immensely while entailing only a relatively small increase in computation [13]. The study of texture mapping is popular in both computer graphics and image processing because its methods are applicable to both areas. There are three main topics in the fundamentals of texture mapping: *acquiring a texture*, possibly including texture synthesis and texture scanning, *the geometric mapping* that warps a texture onto a surface, and *the filtering* that is necessary in order to avoid aliasing. The study of geometric mapping with a 2D domain is a major research concern.

Mapping a 2D texture onto a 3D surface usually requires a parametrization of the surface [13]. This comes naturally for surfaces that are defined parametrically, such as bi-cubic patches, but less naturally for other surfaces such as polyhedra or subdivision surfaces, which are usually defined implicitly. One of the first algorithms using the parametric representation of patches to find texture is [4]. However, in general, there is no natural mapping from a 2D texture space to a 3D object space. The texture is usually distorted [13]. This is especially true for surfaces with arbitrary topology.

Distortion of the texture can be avoided to certain degree through considering special functions such as *conformal mappings* and *isometric mappings*. A mapping is said to be *conformal* if it preserves angles between edges [5]. A mapping is called an *isometry* if it preserves the norm (length) of each edge [5]. For a triangular mesh, an isometric mapping is also conformal but the reverse is not true. Constructing a geometric mapping that is isometric or conformal, unfortunately, is not always possible. A second choice is to use a norm preserving based

Research work of the authors is supported by NSF under grants DMS-0310645 and DMI-0422126.

Shuhua Lai and Fuhua (Frank) Cheng
Graphics & Geometric Modeling Lab
Department of Computer Science
University of Kentucky
Lexington, Kentucky 40506–0046, USA
Tel.: +1 859 257 6760
Fax: +1 859 323 1971
E-mail: {slai2,cheng}@cs.uky.edu

or angle preserving based optimization to reduce overall distortion of the mapped image. The problem with this approach is its computation cost.

In this paper we will introduce a geometric mapping method that can generate high quality texture on surfaces of arbitrary topology, but with a less expensive computation process. The method is simple, it maps the given surface from the 3D object space into the 2D texture space to identify the 2D texture structure that will be used to texture the surface. The 2D texture structure has the same topology as the given 3D surface. Therefore, the mapping guarantees a seamlessly textured surface. The object space to texture space projection is optimized to ensure minimum distortion of the texture mapping process. The optimization is achieved through a commonly used norm preserving minimization process on edges of the surface. However, the optimization problem can be set up as a quadratic programming problem and, hence, solved by a linear least squares method. The key here is to make a good guess on an initial value of the solution set, a seemingly trivial concept but with surprisingly important impact on the computation process. Three guessing methods that provide different level of visual effect will be proposed.

The remaining part of the paper is arranged as follows. A brief review of previous works related to this one is given in Section 2. A description of our texture mapping technique is given in Section 3. Three techniques to choose an initial guess are presented in Section 4. A summary of the algorithm is given in Section 5. Implementation issues and test cases are shown in Section 6. The concluding remarks are given in Section 7.

2 Previous and Related Work

Texture mapping with a 2D texture space is basically a surface parametrization process. Therefore, if a surface is already parametrized, then texture mapping of the surface is a straightforward process if parametrization of the surface is followed in the texture mapping process. The problem with this approach is, the result might not be uniform or seamless if the parametrization is patch based. See Figure 1 for an example of parametrization based texture mapping. Notice the non-uniform stretching of the texture and the existence of seams at several places of the surface. Actually, even a global parametrization cannot guarantee uniform and seamless texturing. One needs to impose extra constraints such as isometry or conformity on the mapping process to achieve uniformity and seamlessness. General methods for global parametrization are based on functional optimization, with special metrics defined to measure the deviation of the parametrization from an isometry [10]. Several methods have been proposed for texture mapping using global parametrization for surfaces with arbitrary topology. For example, in [11], a global conformal parametrization for surfaces with nontrivial topology is presented and quite good test results are generated. However, in general, a global parametric texture mapping technique that preserves distances and, consequently, can act as an isometry does not exist [5].

Fig. 1 An example of patch parametrization based texture mapping.

To improve the parametric mapping of (u, v) to $S(u, v)$ and to alleviate the distortion, several methods have been proposed for texture mapping with minimal distortion. For example, in [1], the image is projected onto a freeform surface so it is free from distance distortion when viewed from the projection direction. In [3], texture mapping through an intermediate simple surface, such as a cube, a cylinder, or a sphere is used to minimize the distortion in the texture of the final object. Relaxation techniques to minimize the distance distortion by employing a grid of sampled points on the surface and optimizing the deviation of the distances of neighboring grid points have also been considered [2,14]. There also exist *feature-based warping and blending* techniques for texture mapping [18,20]. The user controls the mapping process by defining a set of features consisting of 3D points picked on the surface and corresponding 2D points of the texture space. By minimizing some metric functions, the mapping interpolates the features with satisfactory distortion. In addition, energy based optimization methods have been studied as well. For example, in [16], several energies are defined for the texture mapping process. By minimizing the deformation energy, one gets a minimal distorted mapping.

Techniques aiming at improving the quality of realistic effects of texture mapping have also been proposed. For example, *view dependent texture mapping* (VDTM) [17] is a technique for generating novel views of a scene with approximately known geometry making maximal use of a sparse set of original views. Another useful texture mapping technique is *texture synthesis*. Texture synthesis is the process of replicating the statistical and perceptual properties of a user specified example over a larger surface [15]. Algorithms exist for synthesizing a wide variety of textures over surfaces with arbitrary topology [19,20]. However, for complicated meshes, there

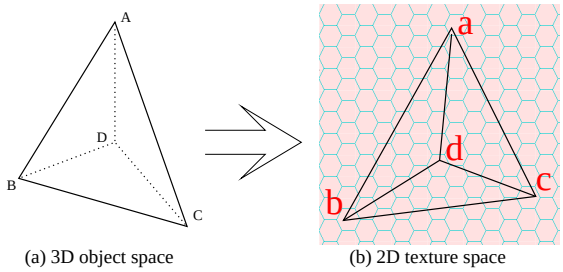
is no guarantee that the meshes are textured without noticeable seams or artifacts.

3 Texture Mapping using Norm Preserving based Optimization

3.1 Basic Idea

The idea is to project the object into a big piece of elastic texture paper so that for each projected face (visible or invisible) of the object, a piece of the texture paper can be identified. This piece of the texture paper is then duplicated and used to cover the corresponding face of the 3D object. The elasticity of the texture paper enables us to cover the entire face of the object even if the texture paper piece is not identical to the object face. For instance, if the 3D triangular pyramid shown in Figure 1(a) is projected into a 2D texture space like the one shown in Figure 1(b), then the image regions Δabd, Δabc, Δacd and Δcbd are used to texture the faces ΔABD, ΔABC, ΔACD and ΔCBD of the 3D pyramid, respectively. The texturing process will stretch the image regions if necessary to ensure each face is covered properly.

For a projected object, call the collection of the corresponding image regions in the texture space (such as Δabd, Δabc and Δacd in the above example) a *texture structure*. It is easy to see that the quality of the resulting texture mapping is completely determined by properties of the texture structure. For instance, one can tell immediately that the texture mapping generates seamless result because the texture structure has the same topology as the 3D object. If the length of each image region edge is the same as the length of the corresponding object face's edge, then for a triangular polyhedron, the result of the texture mapping process will be the best one can get because in this case, no stretching or distortion of the image region will be necessary at all during the texture mapping process. However, projecting a 3D object into a 2D plane with all the edge lengths preserved is not possible in general, especially when the object is of arbitrary topology [5,10]. A second choice is to define a projection that preserves the length of each object edge as much as possible. This approach gives good results even though it cannot completely avoid distortion of the texture [16]. However, since the length preserving based optimization process is a non-linear problem [16], it can only be solved by a non-linear least squares method which is both costly and unstable. Hence, the key in providing a better solution to this problem is to find a way to preserve the advantage of the length preserving based optimization, but eliminate its disadvantage - the costly computation process.



(a) 3D object space (b) 2D texture space

Fig. 2 Basic idea of texture mapping using norm preserving based optimization.

3.2 Mathematical Setup

The norm preserving requirement is satisfied if

$$\| P_i - P_j \|_q = \| T_i - T_j \|_q, \quad \text{for all } (i,j) \in \text{Edges} \quad (1)$$

where $\| \cdot \|_q$ denotes the l_q-norm of a vector, P_k is a vertex of the surface in the 3D object space and T_k is the corresponding point in the uv texture (parameter) space. A mapping satisfying the above requirement is a non-distortion mapping. However, in general, this requirement cannot be satisfied by surfaces of arbitrary topology. A compromise is to minimize the sum of the differences of the edges:

$$\begin{aligned} &\textstyle\sum_{(i,j)\in \text{Edges}} (\| P_i - P_j \|_q - \| T_i - T_j \|_q)^2 = \\ &\textstyle\sum_{(i,j)} (\| P_i - P_j \|_q - \sqrt[q]{(u_i - u_j)^q + (v_i - v_j)^q})^2 \end{aligned} \quad (2)$$

where P_k are knowns and u_k, v_k are unkowns. This optimization problem is a non-linear problem, no matter what value is used for q. To find u_k, v_k that minimize Eq. 2, a non-linear least squares method [7] has to be used. For a 3D surface with only a few vertices, a non-linear least squares method works well and gives good results [16]. When the number of vertices in a 3D polyhedron is large, say more that 1,000 vertices as in many cases, a non-linear least squares method becomes very slow and unstable, sometime cannot even solve the problem. Hence, a different approach has to be used.

When $q = 1$ in Eq. 2, we have

$$\sum_{(i,j)\in Edges} (\| P_i - P_j \|_1 - (|u_i - u_j| + |v_i - v_j|))^2 \quad (3)$$

Eq. 3 is still a non-linear problem due to the existence of the absolute value operator. But we can remove the absolute value operator if we know the signs of $(u_i - u_j)$ and $(v_i - v_j)$ for each (i,j). If this is possible, then minimizing Eq. 3 becomes a quadratic programming problem, which can be readily solved by a linear least squares method [8]. We will address the sign determination problem in the next section. Here we assume the signs are known to us and show how to solve the resulting problem.

Without loss of generality, let $u_i \geq u_j$ and $v_i \geq v_j$. Then Eq. 3 becomes

$$\sum_{(i,j)} (|x_i - x_j| + |y_i - y_j| + |z_i - z_j| - (u_i - u_j + v_i - v_j))^2 \quad (4)$$

where (x_k, y_k, z_k) are coordinates of 3D object space vertex P_k. This is a quadratic programming problem. A linear least squares method can be used to find a minimum of this problem [8]. However, to ensure uniqueness of the solution, we need some extra constraints. Due to the fact that a general solution is subject to a translation and a rotation, we can resolve the uniqueness problem by fixing two points in the uv parameter space. Although any two points would work, a better choice is to select points corresponding to adjacent vertices of the 3D surface in the object space. We assume that $T_a = (\bar{u}_a, \bar{v}_a)$ and $T_b = (\bar{u}_b, \bar{v}_b)$ are the fixed points in the uv texture space corresponding to vertices P_a and P_b in the object space.

If we put Eq. 4 in matrix form, then our task is to find a solution $w = (u_1, v_1, u_2, v_2, \cdots, u_n, v_n)$ for the following constrained minimization problem:

$$\begin{cases} \min \ (C \cdot w - d)^T (C \cdot w - d) \\ A \cdot w \leq 0 \\ u_a = \bar{u}_a, \quad v_a = \bar{v}_a \\ u_b = \bar{u}_b, \quad v_a = \bar{v}_b \end{cases} \quad (5)$$

where matrix C and vector d are derived from Eq. 4, and matrix A is determined by the signs of $(u_i - u_j)$ and $(v_i - v_j)$ for all pairs (i, j).

4 Determine an Initial Guess

In our algorithm, the signs of $(u_i - u_j)$ and $(v_i - v_j)$ in Eq. 4 are defined using signs of corresponding values in the initial guess $\tilde{w} = (\tilde{u}_1, \tilde{v}_1, \tilde{u}_2, \tilde{v}_2, \cdots, \tilde{u}_n, \tilde{v}_n)$, i.e., $(u_i - u_j)$ is set to positive if $(\tilde{u}_i - \tilde{u}_j)$ is positive, and $(v_i - v_j)$ is set to positive if $(\tilde{v}_i - \tilde{v}_j)$ is positive. Hence, the initial guess is important not only in reducing the number of iterations of the minimization process, but also in setting up a good approximation of the final mapping. The initial guess does not have to be numerically precise, but it must accurately reflect the order of points in the solution set. Three possible ways to make an initial guess are presented below.

4.1 XY-MinMax Method

Suppose the maximum and minimum of all the 3D vertices in X and Y directions are $Xmax$, $Xmin$, $Ymax$ and $Ymin$, respectively. Then the initial guess $(\tilde{u}, \tilde{v})$ for a vertex (x, y, z) is set to

$$\begin{cases} \tilde{u} = (x - Xmin)/(Xmax - Xmin) \\ \tilde{v} = (y - Ymin)/(Ymax - Ymin) \end{cases}$$

This setting of initial guess is intuitive and straightforward. It does not take into consideration the Z component of a 3D vertex. But this method catches the overall structure of the 3D object in the view plane, hence in some cases it can result in good result, especially when topology of the object is simple and it has only a few control points. The glass in Fig. 3(g) is textured using this method.

4.2 Combination Method

One can take the Z component of a 3D vertex into account by making simple modification of the first method. For example, one can rotate the object for certain degrees in several different directions, and then combine the corresponding XY-MinMax initial guesses linearly to form a new initial guess. One thing one has to be careful with when rotating the object is that one needs to avoid rotating the object symmetrically, because symmetric rotations cancel out the effect of rotation.

Another possible modification to the first method is to use an object-to-sphere mapping to compute the value of $(u,\ v)$. First, enclose the 3D object in a sphere large enough to contain all the vertices of the object. The sphere is centered at the centroid of the object and parameterized as

$$S(u, v) = (\cos(v)\cos(u), \cos(v)\sin(u), \sin(v))$$

where $0 \leq u < 2\pi$ and $-\pi/2 \leq v < \pi/2$. A ray is then emitted from the center of the sphere to each vertex of the polygonal surface. The (u, v) values of the intersection point of the ray with the sphere are then combined with the rotated XY-MinMax initial guess value to get a new combined initial guess value.

These are two small improvements to the first method but, surprisingly, the results are much better, even for complex objects. Fig. 3(f), Fig. 3(h), and Fig. 3(j) are textured using this method.

4.3 Graph Based Method

The first two methods do not work well for some objects, such as objects with many handles. A method that works for all kinds of objects, especially objects with complicated topology, is needed. In the following, we propose a graph based method for such a purpose.

First, treat the 3D mesh as a graph and traverse the mesh using depth first search. For edges at the same depth level, the one with the longest length is traversed first. After the traversal, a linear list of all or some of the 3D vertices is obtained. Assume the list is $(P_1, P_2, \cdots, P_n)$. Let L_{ij} denote the norm of the vector from P_i to P_j and L_i denote the norm of the path from P_1 to P_i, i.e.

$$L_i = L_{12} + L_{23} + \cdots + L_{i-1,i}$$

For each vertex in the above list, the corresponding pair $(\tilde{u}_i, \tilde{v}_i)$ is assigned a set of values using the XY-MinMax method. Then, we interpolate these $(\tilde{u}_i, \tilde{v}_i)$ values in the 2D uv parameter space using a uniform cubic B-spline curve. If the resulting uniform B-spline curve is $f(t)$, $0 \leq t \leq 1$, then we reset $(\tilde{u}_i, \tilde{v}_i)$ to $f(L_i/L_n)$.

If there are vertices that have not been traversed yet, use a technique similar to the construction of an Euler circuit to traverse the remaining vertices. For each remaining component, start the traversal of that component with a vertex that has already been visited (i.e., an vertex in the above list). The new list of vertices constructed will also end with a vertex in the above list. Hence the first and the last vertices in the new list have assigned initial guess. Using the XY-MinMax method we can assign initial guess values to other vertices in the list and then repeat the interpolation process to reset these values.

This method is better than the previous two methods in that it fully takes the Z component into account by considering the norm (length) of each edge. Since the norm of each edge is determined by X, Y and Z components of a 3D vector, using a norm based parameter to reset the initial guess values obviously results in a less distorted texture mapping. In our implementation, Fig. 3(a), 3(b), 3(c), 3(d), 3(e), 3(i) are generated with this method.

5 The Algorithm

Once we have initial values for all the vertices, we need to transform them so that the constrains in Eq. 5 hold, i.e., we need to determine the matrix that transforms the initial guesses for T_a and T_b from $(\tilde{u}_a, \tilde{v}_a)$ and $(\tilde{u}_b, \tilde{v}_b)$ to $(\bar{u}_a, \bar{v}_a)$ and $(\bar{u}_b, \bar{v}_b)$, respectively. This transformation matrix M is of size 3×3 and has the following form:

$$M = \begin{bmatrix} c & -s & r_1 \\ s & c & r_2 \\ 0 & 0 & 1 \end{bmatrix}$$

M can be determined from the following linear system:

$$\begin{cases} M \cdot (\tilde{u}_a, \tilde{v}_a, 1)^T = (\bar{u}_a, \bar{v}_a, 1)^T \\ M \cdot (\tilde{u}_b, \tilde{v}_b, 1)^T = (\bar{u}_b, \bar{v}_b, 1)^T \end{cases}$$

Once M is available, we transform all the initial guess values by left multiplying M, and set up the quadratic programming problem in Eq. 5 and solve it using the method proposed in [8]. *Matlab* also provides a command (*lsqlin*) to solve this kind of problem.

After we get the solution of the quadratic problem, we may still need to clamp them such that all the texture coordinates fall in between 0 and 1. At rendering time, if $m \times n$ texture images are tiled together to texture the 3D object surface, we just need to scale all the texture coordinates in u-direction by m, and in v-direction by n.

The overall algorithm can be summarized as follows.

NormPreservingTM(Mesh, Texture, m, n)
1. triangulate the input polygonal surface,
2. determine $(\bar{u}_a, \bar{v}_a)$ and $(\bar{u}_b, \bar{v}_b)$,
3. get an initial guess,
4. compute the transformation matrix M,
5. transform the initial guess by M,
6. set up the quadratic problem in Eq. 5,
7. solve the quadratic problem,
8. clamp all the texture coordinates to (0, 1),
9. scale texture coordinates by m and n,
10. do the rendering with the texture coordinates.

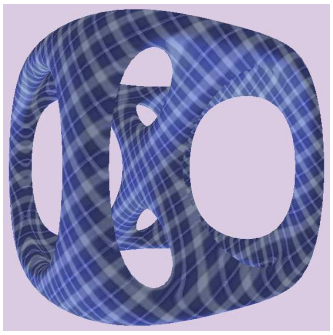
6 Implementation and Test Results

The proposed approach has been implemented in *C++* using *OpenGL* as the supporting graphics system on a Windows system. Quite a few examples have been tested with the method described here. Since the initial guess values play an important role in our algorithm, test cases with different initial guess values are presented. For example, Fig. 3(g) is textured using the XY-MinMax method, while Fig. 3(f) Figs. 3(j), and 3(h) are textured using the combination method. The remaining test cases in Fig. 3 are textured using the graph based method. We can see that all the results are seamless and continuous. A comparison of the textured teapots shown in Fig. 3(d) and Fig. (1) makes this point obviously clear. The one shown in Fig. 3(d) is uniform and seamless while the one shown in Fig. (1) has noticeable seams and the texture is non-uniformly stretched.

Although all the tested results have some distortion, we can tell the results generated by the graph based method have the best visual effect. We have also tested our algorithm on surfaces with multiple holes. For example, Fig. 3(a), Fig. 3(b), and Fig. 3(i) have one or more holes, none of them can be represented by a single *NURBS* surface. Hence it is very difficult to do texture mapping on this kind of surfaces using a parametrization method. However, with our method, it is not more difficult than doing a texture mapping on a cube.

7 Conclusion

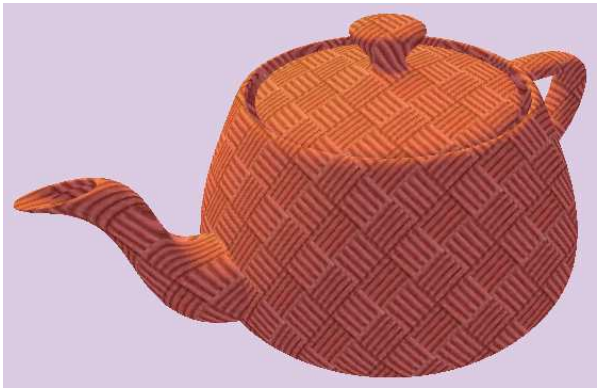
A texture mapping method for surfaces of arbitrary topology is proposed. This method maps the surface from the 3D object space into the 2D texture space to identify the 2D texture structure that will be used to texture the surface. The 2D texture structure has the same topology as the 3D object surface. Therefore, the new method guarantees a uniformly and seamlessly textured surface. The 3D to 2D mapping performs norm preserving based optimization on each edge to lower overall distortion of the



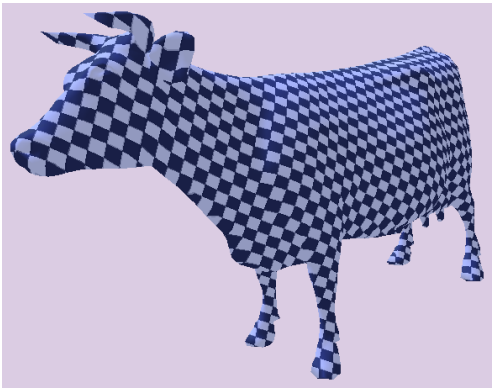
(a) Toy station

(b) Ventilation control component

(c) Stanford Bunny

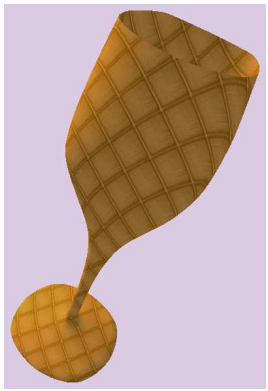


(d) Utah Teapot

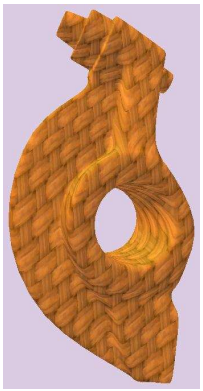


(e) Cow

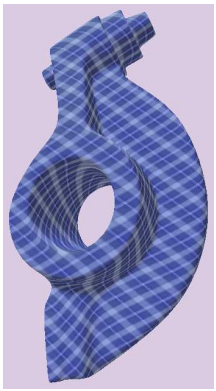
(f) Horse



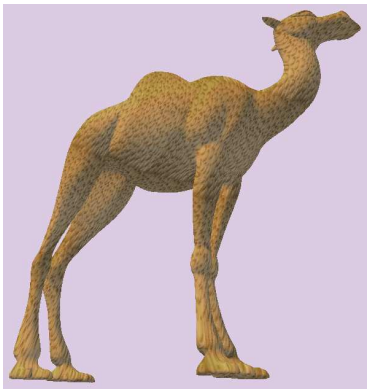
(g) Glass



(h) Rocker arm



(i) Rocker arm



(j) Camel

Fig. 3 Texture mapping on surface with arbitrary topology.

subsequent texture mapping process. The main contribution of the paper is to make the optimization process a quadratic programming problem so it can be solved by a linear least squares method. Several initial guess methods have been put forward for such a purpose, and our experiments show that the graph based method has the best result.

Our algorithm does not work well for complex surfaces with exceptionally abnormal curvature distribution. This problem can be alleviated by splitting the object into several disjoint regions [16]. Nevertheless, our method works well in most cases. The resulting surfaces generated by our method look more realistic than those generated by patch parametrization based approaches, which do not texture the surface uniformly, because different patches may have different sizes. Our method is especially suitable for applications that do not require precise texture mapping results but demand highly efficient mapping process such as computer animation or video games.

Acknowledgements Data set for Figure 3(e) and data sets for Figures 3(f) and 3(j) are downloaded from the following web sites

- http://graphics.cs.uiuc.edu/~garland/research/quadrics.html.
- http://graphics.csail.mit.edu/~sumner/research/deftransfer/data.html

respectively.

References

1. Nur Arad, Gershon Elber, Isometric Texture Mapping for Free-form Surfaces, *Computer Graphics Forum,* 16(5):247-256. December 1997.
2. C. Bennis, J.M. Vezien, G. Iglesias, Piecewise Surface Flattening for Non-Distorted Texture Mapping. *SIGGRAPH 1991,* 25(4):237-246. July 1991.
3. E. Bier, K.Sloan, Two-part texture mapping. *IEEE Computer Graphics and applications,* 40-53. September 1986.
4. Blinn, J. F. and Newell, M. E., Texture and Reflection in Computer Generated Image, *Communication of the ACM,* 19(10):542-547. October 1976.
5. do Carmo. M. P., Differential Geometry of Curves and Surfaces, Prentice-Hall, 1976.
6. Catmull, E., A subdivision Algorithm for Computer Display of Curved Surfaces, *Ph.D. Thesis,* Department of Computer Science, University of Utah, December, 1974.
7. Coleman, T.F. and Y. Li, An Interior, Trust Region Approach for Nonlinear Minimization Subject to Bounds, *SIAM Journal on Optimization,* Vol. 6, pp. 418-445, 1996.
8. Coleman, T.F. and Y. Li, A Reflective Newton Method for Minimizing a Quadratic Function Subject to Bounds on Some of the Variables, *SIAM Journal on Optimization,* Vol. 6, Number 4, pp. 1040-1058, 1996.
9. F. Crow, Summed-area tables for texture mapping. *SIGGRAPH 1984,* 18(3):207-212. July 1984.
10. Floater, M.S. and K. Hormann, Surface Parametrization: a Tutorial and Survey, in *Advances on Multiresolution in Geometric Modelling*, Dodgson N., Floater M.S., Sabin M., (Eds.), Springer-Verlag, Heidelberg, Heidelberg, Denmark, 2004.
11. Xianfeng Gu and Shing-Tung Yau, Global Conformal Surface Parametrization. *ACM Symposium on Geometry Processing,* 2003.
12. P. Hanrahan, P. Haeberly, Direct WYSIWYG painting and texturing on 3D shapes. *SIGGRAPH 1990,* 24(4):215-223. August 1990.
13. Heckbert, P. S., Survey of Texture Mapping, *IEEE Computer Graphics and Applications,* 6(11):56-67. November 1986.
14. S.D. Ma, H. Lin, Optimal texture mapping. *EUROGRAPHICS 1988,* 421-428. September 1988.
15. Magda S. and D Kriegman D. Fast Texture Synthesis on Arbitrary Meshes. *Eurographics Symposium on Rendering*, pp. 82-89, 2003.
16. Maillot, J., Yahia H. and Verroust A. Interactive texture mapping. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (SIGGRAPH 93), pp. 27-34, 1993.
17. Sormann, M., Zach, C. and Karner, K. Texture Mapping for View-Dependent Rendering. *Proceedings of the 19th spring conference on Computer graphics,* pp. 131-148, 2003.
18. Tang, Y., Wang, J., Bao, H., etal. RBF-based constrained texture mapping. *Computers & Graphics*, Vol. 27 (3), pp. 415-422, June, 2003.
19. Li-Yi Wei and Marc Levoy. Texture Synthesis over Arbitrary Manifold Surfaces *ACM SIGGRAPH 2001*, pp. 355-360, 2001.
20. Zhang, J., Zhou, K., Velho, L., etal. Synthesis of progressively-variant textures on arbitrary surfaces, *ACM SIGGRAPH 2003*, pp. 295 - 302, July, 2003.

SHUHUA LAI is currently a Ph.D student in the Department of Computer Science at the University of Kentucky. He received a BS in Applied Mathematics and Computer Applications from the East China Normal University, and an MS in Computer Science and Engineering from the Shanghai Jiaotong University. His research interests include computer graphics and computer aided geometric modeling.

FUHUA (FRANK) CHENG is Professor of Computer Science and Director of the Graphics & Geometric Modeling Lab at the University of Kentucky. He holds a PhD from the Ohio State University, 1982. His research interests include computer aided geometric modeling, computer graphics, parallel computing in geometric modeling and computer graphics, approximation theory, and collaborative CAD.