3D MODEL DEFORMATION ALONG A PARAMETRIC SURFACE

Bing-Yu Chen * Yutaka Ono * Henry Johan * Masaaki Ishii * Tomoyuki Nishita * Jieqing Feng †
* The University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo, 133-0033
Japan
† Zhejiang University
Hangzhou, 310027,
China

Abstract

Free-Form Deformation (FFD) is an efficient technique for editing the shapes of 3D models, and widely used in Computer-Aided Design (CAD), computer animation, computer graphics entertainment etc. The basic idea of some famous previous FFD approaches deforms a target 3D model by adjusting some control points of a 3D lattice surrounding the model. It is a tedious work, especially when the lattice contains too many control points. Moreover, how to place the control points of the lattice to make them cover the region of the target model is also a problem, and it is also difficult to keep the geometric measurement for the deformed model. Therefore, in this paper, a novel FFD method without any lattice-like structure is proposed by borrowing the idea of non-distortion texture-mapping for free-form surfaces. By using this method, the shape of the deformed model due to a given parametric surface can be predicted easily, so that the user can get his or her desirable results more intuitively and has no necessary to place the control points of a proper lattice. Moreover, since this method is simple, efficient, and has a real-time response, it is more suitable for doing the animation with thin objects.

Key Words: 3D model deformation, non-distortion deformation, non-distortion mapping, flattened surface, animation

1. Introduction

Recently, computer graphics technologies have been applied to several fields, such as movie industry, video game, CAD, and scientific visualization. Obviously, the expression of 3D models is also an important part of the computer graphics technologies. The 3D models are usually represented by some primitives such as points, lines, and polygons. Although complicated and beautiful models can be rendered now with a lot of such primitives by using graphics hardware, how to deform such models to be people's desirable shapes generally, intuitively, and efficiently is still a problem.

FFD is an efficient technique in computer graphics for providing a general modification of a 3D model such as twisting, bending, and stretching. Sederberg and Parry first introduced this method from the practical viewpoint in 1986 [1]. By using this method, the users first generate some control points called a lattice to contain a 3D model which is a target object for deformation, and then adjust the control points to deform the parametric space inside. Finally, the deformed parametric space is mapped onto the target model automatically. A lot of related approaches are based on this method, and it has become a useful and well-known technique for 3D model deformation.

Several methods of FFD are mostly based on the above methods, although there are some differences about the definitions of the lattices. However there are some problems while doing the FFD processes. First, the relationship between the lattice and the target model is unclear, so it is hard to grasp intuitively that how desirable deformation can be obtained if the control points of the lattice are adjusted. Then, how to place the control points of the lattice to proper positions to make the users to control the shape well is very difficult. Furthermore, it is also difficult to keep the geometric shape of the model after deformation, and possible to have a distortion problem of the deformed shape.

Therefore, we propose a novel FFD method in this paper without any lattice-like structure to prevent the users to setup the control points and adjust them. The basic idea of our approach is to utilize the technology of texture-mapping [2], since texture-mapping can make a mapping relationship between a 2D texture image with a 3D free-form surface as to wrap the image to fit the shape of the surface. Hence, we can also find a mapping relationship between a 3D model and a parametric surface by just replacing the texture image in the texture-mapping to be a 3D model. Unfortunately, the main problem of the texture-mapping itself is the distortion problem if we just use the linear mapping. So, we borrow the technology of making non-distortion texture-mapping [3] to prevent this problem. Then, the user can deform a 3D surface as they wish by just making a desirable parametric surface which is easier than adjusting the control points of a lattice.

2. Related Work

Barr introduced an efficient method with some restrictions to deform a 3D model by using the combinations of four operations [4]. Then, a deformation technique is proposed by Sederbarg and Parry [1], which is widely available and almost all subsequent methods are based on it. Coquillart proposed an extension of it, called Extended Free-Form Deformation (EFFD), which uses several low resolution lattices, called "chunk", for deformation [5]. Thereby, a complicated lattice can be

defined without raising the resolution of the chunks, since it can be a combination of several chunks, but it is difficult to maintain the continuity of the form in the case of connection of the chunks. Coquillart and Jancéne used EFFD method to build animations called Animated Free-Form Deformation (AFFD) [6].

Griessmair and Purgathofer proposed a new FFD method based on B-Spline with three valuables, and optimized the mesh division after deformation [7]. By using the general FFD method, the relationship between a lattice and a target model is obscure, so that to adjust the control points of the lattice is difficult to understand intuitively. Hsu et al. aimed to solve this problem and proposed a method to control the target model directly [8]. By using this method, a desirable shape could be generated by adjusting arbitrary vertices directly of the model, and then the corresponding control points of the lattice are calculated backward to acquire the shape. Therefore, desirable results can be acquired even for the local deformation.

Kalra et al. thought up a Rational Free-Form Deformation (RFFD) method to use a rational parametric volume to simulate the movement of facial muscles [9]. The method proposed by Lamousin and Waggenspack, Jr. allows the users to do the local deformations by using a B-Spline or controlling the shape by weighting the control points of the lattice, called NURBS-based Free-Form Deformation (NFFD), and succeeded in raising the flexibility of FFD [10]. Like EFFD method, it is also possible to combine two or more lattices, and the shape could still be smoothly connected only by piling up a segment in the connection part of the lattices. Moreover, Feng and Peng also proposed a fast accurate B-Spline FFD method [11]. In this method, the target model is surrounded by a B-Spline volume, but the problems of placing and controlling the control points are the same as the previous methods.

Obviously, to adjust the control points of a lattice is not intuitive; to make the 3D model deformation to be easier, a more intuitive method is needed. Lazarus et al. used an axis instead of using a lattice to try to provide an efficient and intuitive deformation method called Axial Deformations (AxDf) [12]. Chang and Rockwood used a Bézier curve to define the desirable skeleton of the deformed object [13]. Singh and Fiume used wires for deformation [14]. Although these methods provide simpler control methods than previous ones, the user also needs to place a proper axis or some suitable wires for deformation. Feng et al. provided another FFD method by using two parametric surfaces called shape surface and height surface [15]. This method gives users more flexibility for deformation and has an intuitive controlling, but the deformed results have a distortion problem.

Therefore, we propose a novel method to deform the surface of the target 3D model by a given parametric surface, so that the user can image what will be generated after the deformation. By utilizing the technology of non-distortion texture-mapping, the distortion effect of the deformed object is decreased. Moreover, for doing the animation of the 3D model deformation, a real-time system is desired by the users. Since the proposal method is efficient and has a real-time response, it is suitable for doing the animation task.

3. Deformation by a Given Parametric Surface

To use a given parametric surface to do the 3D model deformation, the user first selects a well-defined surface, such as the surface of a cone or a cylinder, or generates a parametric one by himself or herself. The selected well-defined surface or generated parametric one is the desired shape after the target model is deformed, i.e. the deformed result will be like the shape along the given surface.

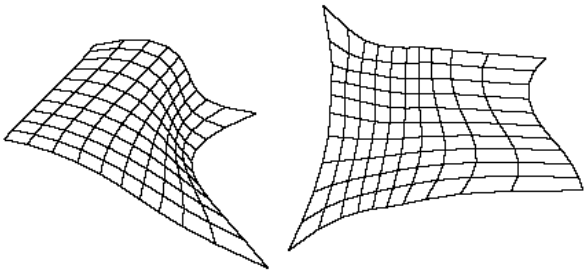
The mapping relationship between the given parametric surface and the target 3D model is like the mapping between a free-form surface and a 2D texture image, because in texture-mapping, the texture image is mapped onto the curved surface, and in our approach, the target model is mapped onto the given parametric surface, if we think of the texture image as a plane in a three dimensional space.

Unfortunately, although texture-mapping is a well-known technology in computer graphics, and all graphics libraries support it, such as OpenGL, there is a well-known problem that the texture image is distorted because the texture data on the flat image plane is mapped onto the curved surface. Hence, if we just use the traditional texture-mapping method to map the target model to the given parametric surface, the distortion is also a problem for us. Therefore, how to perform the mapping with less distortion is what we have to solve first.

To solve the distortion problem of the texture-mapping, some methods are proposed. Peachey first faced to minimize this distortion problem [16]. Bier and Sloan, Jr. separated the texture-mapping process into two passes [17]: the 2D texture image is first mapped onto a simple intermediate surface, and then this surface is projected onto the target 3D model. Lévy and Mallet comprised numerical computations of physical properties stored in fine grids within texture space, and then generated the grids which are suitable for finite element analysis [18]. Ma and Lin also proposed an approximate non-distortion texture-mapping method to solve this problem [3].

3.1 Flattened Surface Generation

In Ma and Lin's method, the curved surface is flattened to a 2D plane. Except for some kinds of 3D curved surfaces such as the surface of a cylinder or a corn which is called "developable surface", other general surfaces like the Bézier surface cannot be completely flattened to 2D planes without any distortion, so they could only have approximate flattened surfaces as shown in Figure 1.



(a) (b)

Figure 1: (a) a Bézier surface and (b) its flattened surface.

The algorithm for flattening the parametric surface is described in Appendix. When generating a flattened surface, the range of P_j in Appendix is a critical point to be concern about. Assume the range is set to be r, where r is a positive integer. If r is enlarged, the precision of the flattened surface is better, and since there were few steps to convergence in practice, it could also get a good performance.

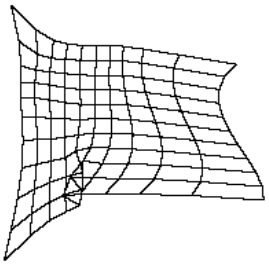


Figure 2: another flattened result of Figure 1 (a) when setting $r=1$.

As the Bézier surface shown in Figure 1 (a), if we set $r=1$, the flattened result will be unusable as shown in Figure 2. To get a proper flattened surface, a larger r is needed. The flattened surface shown in Figure 1 (b) is the result of setting $r=3$. How to set a proper r for a good flattened surface is an experimental problem and depends on the number of grids of the Bézier surface.

In Figure 3, the relationship of the number of grids of the Bézier surface is shown in Figure 1 (a) and the flattened surface generation performance with different r is measured. When the number of grids is increased, the performance becomes worse, and the result with a small r becomes unusable like Figure 2. Moreover, to enhance the calculating performance, the sampling points for the range r is set to be the distance of the grid of the parametric surface as shown in Figure 4.

Therefore, the corresponding points for mapping are computed by comparing coordinate system of the flattened surface with that of the texture, so that the texture-mapping with less distortion which geometrical distances are maintained is achieved. In our approach, before doing the 3D model deformation, the given parametric surface is flattened to a 2D plane by using this method. That means the geometrical distance between two vertices of the target model could be kept on the deformed model.

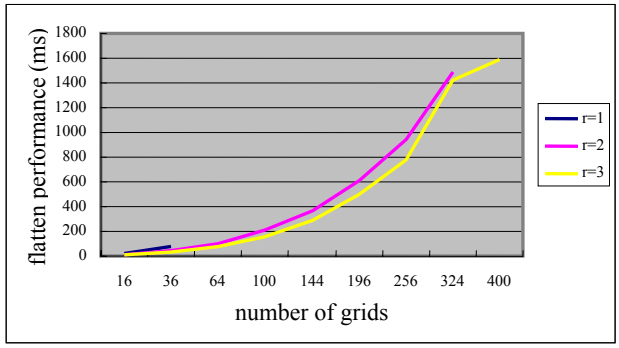
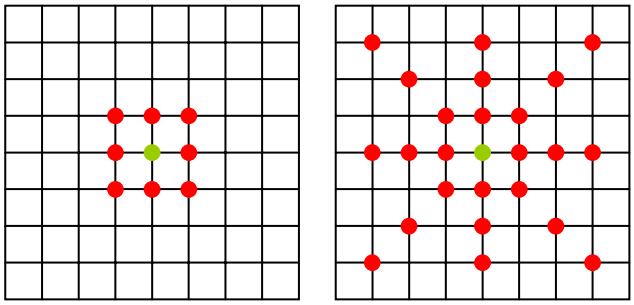


Figure 3: performance testing for flattened surface generation due to the number of grids and r.



(a) (b)

Figure 4: sampling points for (a) $r=1$ and (b) $r=3$.

3.2 3D Model Deformation

By using a given parametric surface to do the 3D model deformation, we first flatten the surface, and then put the model onto the flattened surface, so that the model can get a proper corresponding mapping with the parametric surface. Moreover, unlike the other conventional FFD methods, the proposed method is done without generating any lattice-like structure, and the deformation task is also easier than before.

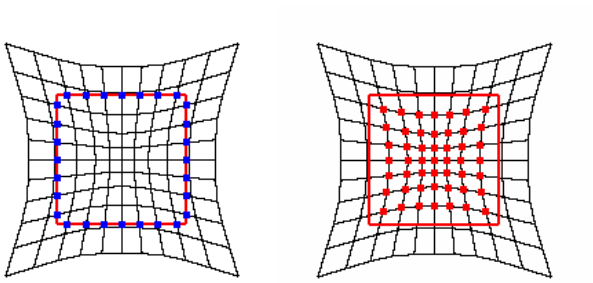
To get a smooth deformed object, we have to subdivide the target 3D model, if the size of the polygon of the model is larger than the size of the grid of the flattened surface. We first project all the vertices, edges, and faces of the target model to the flattened surface, and then compute the following three types of points with the grid of the flattened surface of the given parametric surface:

(a) The *vertex points* which are defined as the original projected vertices of the target model.

(b) The *edge points* which are generated by intersected the original projected edges of the target model with the grid of the flattened surface.

(c) The *face points* which are generated by inserted the cross points of the grid of the flattened surface inside the original projected faces of the target model.

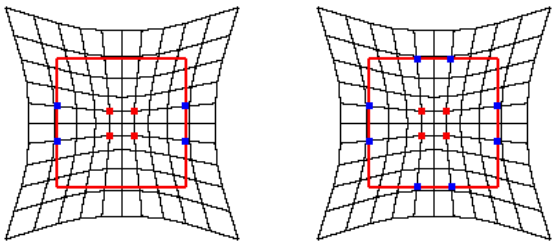
For example, if we project the polygon at the bottom of a cube onto a flattened Bézier surface, there will be a square on the flattened surface, and the generated edge points and face points are shown in Figure 5. The vertex points are the four corners of the square, the edge points are the blue points on the four edges, and the face points are the red points inside the square. The flattened surface in Figure 5 is the flattened result of the Bézier surface shown in Figure 9 (a).



(a) (b)

Figure 5: (a) edge points and (b) face points.

Then, the edge points and face points are projected backward to the original parametric surface, and the corresponding points on the target model are also generated by using bi-linear interpolation. Finally, the generated corresponding points are displaced along the normal vectors of the parametric surface. Hence, there are two corresponding points for each vertex point, edge point, and face point: one is on the parametric surface, and the other is on the surface of the target model. Although there are several generated edge points and face points, to make the deformed model not to contain too much newly generated vertices, the curvatures of the corresponding points on the parametric surface of the edge points and face points are computed, so that only the bumpy region due to the parametric surface needs to be subdivided.



(a) (b)

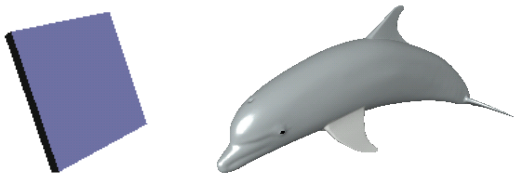
Figure 6: (a) mark the un-removable edge points and face points on the edges consisted by them; (b) mark other edge points due to the face points.

To reduce the numbers of the edge points and face points, we first calculate the curvature of one projected edge with the parametric surface, and mark the edge points, where the curvatures are higher than a given threshold ε, to be un-removable. The edge point at the opposite side of the marked edge point is also marked. Then, the face points on the edge linked by the marked edge points are detected as shown in Figure 6 (a). If there is a face point on the edge also has a curvature higher than the threshold ε, the endpoints of the other edge which consists of the face point are also marked as Figure 6 (b). And then, check the other projected edges again to see if there still have un-removable edge points. Finally, the target model can be subdivided due to the newly generated points.

Moreover, the user can choose where to put the target model onto a part of the given parametric surface for animation or just fitting the region of the surface. To keep the continuity of the surface of the deformed object, the user could only put the target model onto the given surface, and the projected vertices could only be located inside the flattened surface.

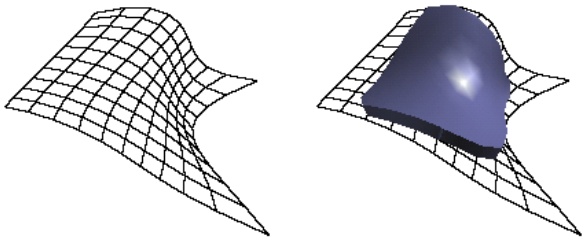
4. Results

There are two 3D models in Figure 7: (a) is a simple thin plate which has only 6 polygons, and (b) is a dolphin model. Figures 8 - 9 show the deformation results of the simple model in Figure 7 (a) along the shape of different Bézier surfaces and a cylinder, respectively.



(a) (b)

Figure 7: (a) a simple thin plate and (b) a dolphin model.

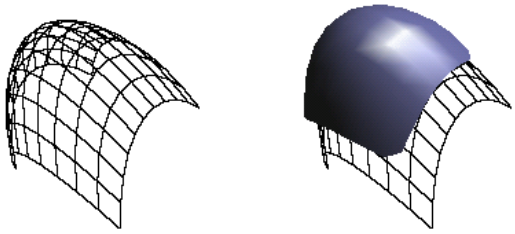


(a) (b)

Figure 8: (a) a Bézier surface and (b) a deformation result along it.

The performance of our approach is listed in Table 1. The process of both flattening the given parametric surface and deforming the target 3D model due to the surface are performed. The testing platform is a PC with an Intel Xeon 2GHz CPU, 1GB memory, and a 3Dlabs Wildcat II 5110 graphics accelerator with OpenGL support. The range r used for flatten the surface is set to be 3. Although the time for flattening the surface is much larger than the time for deforming the target model, the flattening process is just a pre-process and could be

done at off-line and independent with the complexity of the target model.

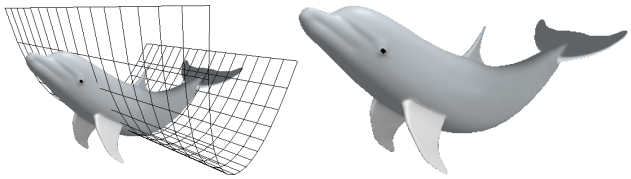


(a) (b)

Figure 9: (a) another Bézier surface and (b) a deformation result along it.

given surfaces	Figure 8	Figure 9	Figure 10
number of grids	10×10		16×16
flatten (ms)	166	128.4	777
deform (ms)	12	16	186.2

Table 1: performance testing for flattening the given surfaces shown in Figures 8 - 10 and the deformation due to them.



(a) (b)

Figure 10: (a) a Bézier surface and (b) a dolphin model deformed along the surface.

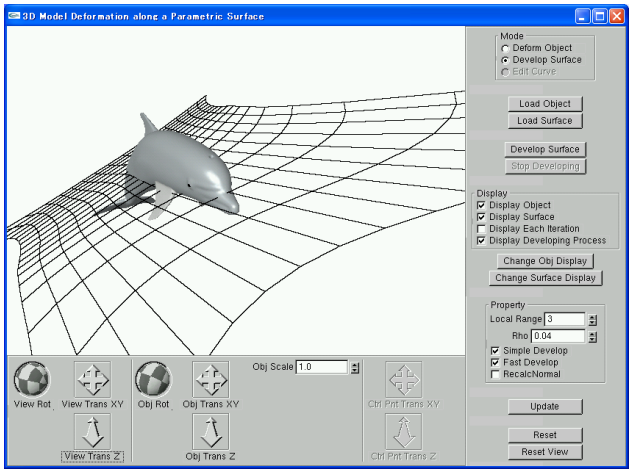


Figure 11: the user interface of our system and there is a dolphin model put on a flattened surface.

Figure 10 (b) shows the deformation of a complex dolphin model as shown in Figure 7 (b). The deformation is along a Bézier surface shown in Figure 10 (a). The graphical user interface (GUI) of our system is shown in Figure 11. The desirable Bézier surface and the relative position of the target model on the flattened surface of the Bézier surface could be controlled through the system. In the screen of Figure 11, the dolphin model of Figure 7 (b) is put onto the flattened surface of the Bézier surface of Figure 10 (a). Moreover, since each polygon of the original dolphin model is smaller than the grid of the flattened surface, there is no subdivision in the deformation process.

Since our method can get a real-time response, it is suitable for doing animation about deformation. For example, the dolphin model in Figure 7(b) is deformed to simulate the swimming as shown in Figure 12. To do the animation, the dolphin is put onto a parametric surface like Figure 13, then the dolphin can be deformed along the surface in real-time, although it contains 73,665 polygons.

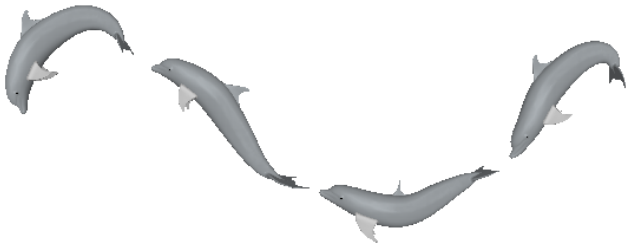


Figure 12: a swimming dolphin.

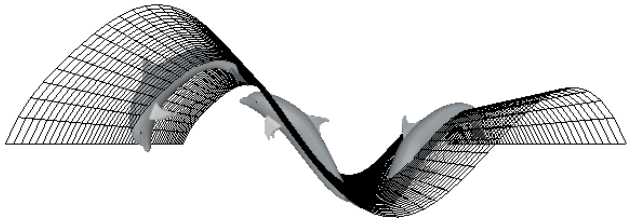


Figure 13: a dolphin deformed along a parametric surface.

The number of grids of the given parametric surface affects the performance significantly, which could be controlled by users through our system, because the complex parametric surface has more grids to be flattened and the new points used for subdividing the target model are also increased. However, to use a parametric surface with more grids for model deformation also means to get a better quality of the deformed result. Moreover, although the numbers of the edge points and face points are increased if the number of grids is increased, the number of vertices of the deformed model is not increased too much, since the newly generated vertex is added to the target model when it is needed.

5. Conclusion

We proposed a novel FFD method to realize non-distortion deformation which maintains the geometric shape even after deformation by utilizing the technique of texture mapping with less distortion. Because the deformation is performed along a given 3D parametric surface which the user could generate easily, it is easy to predict the result shape after the deformation. Moreover, using a given parametric surface to do the deformation, the adjustment of the control points of the parametric surface more is intuitive and intelligible than the one of the lattice used in previous FFD methods.

To deform the thin model by using our system, the deformed model could get the result with less distortion, since the faces of the model far from the parametric surface have worse deformed results than others. To minimize the distortion, the user could move the flattened surface in the middle of the model.

Acknowledgement

One of the authors, Jieqing Feng, is partially supported by National Natural Science Foundation of China (No. 69903008), and Henry Johan is supported in part by the Japan Society for the Promotion of Science Research Fellowship.

References

[1] T. W. Sederberg and S. R. Parry, Free-form deformation of solid geometric models, *ACM Computer Graphics (Proc. SIGGRAPH 86' Conf.), 20*(4), 1986, 151-160.

[2] E. Catmull, *A subdivision algorithm for computer display of curved surface* (Ph.D. Thesis, Report UTEC-CSc-74-133, Computer Science Dept., Univ. of Utah, 1974).

[3] S. D. Ma and H. Lin, Optimal texture mapping, *Proc. Eurographics 88' Conf.*, 1988, 421-428.

[4] A. H. Barr, Global and local deformations of solid primitives, *ACM Computer Graphics (Proc. SIGGRAPH 84' Conf.), 18*(3), 1984, 21-30.

[5] S. Coquillart, Extended free-form deformation: a sculpturing tool for 3d geometric modeling, *ACM Computer Graphics (Proc. SIGGRAPH 90' Conf.), 24*(4), 1990, 187-196.

[6] S. Coquillart and P. Jancéne, Animated free-form deformation: An interactive animation technique, *ACM Computer Graphics (Proc. SIGGRAPH 91' Conf.) 25*(4), 1991, 23-26.

[7] J. Greissmair and W. Purgathofer, Deformation of solids with trivariate B-splines, *Proc. Eurographics 89' Conf.*, 1989, 137-148.

[8] W. M. Hsu, J. F. Hughes, and H. Kaufman, Direct manipulation of free-form deformations, *ACM Computer Graphics (Proc. SIGGRAPH 92' Conf.), 26*(2), 1992, 177-184.

[9] P. Kalra, A. Mangili, N. M. Thalmann, and D. Thalmann, Simulation of facial muscle actions based on rational free form deformation, *Computer Graphics Forum (Proc. EUROGRAPHICS 92' Conf.), 11*(3), 1992, 59-69.

[10] H. J. Lamousin and W. N. Waggenspack, Jr., NURBS-based free-form deformations, *IEEE Computer Graphics and Applications, 14*(6), 1994, 59-65.

[11] J. Feng and Q. Peng, Accelerating accurate B-spline free-form deformation of polygonal objects, *ACM Journal of Graphics Tools, 5*(1), 2000, 1-8.

[12] F. Lazarus, S. Coquillart, and P. Jancéne, Axial deformations: an intuitive deformation technique, *Computer-Aided Design, 26*(8), 1994, 607-613.

[13] Y.-K. Chang and A. P. Rockwood, A generalized de Casteljau approach to 3d free-form deformation, *Proc. ACM SIGGRAPH 94' Conf.*, 1994, 257-260.

[14] K. Singh and E. L. Fiume, Wires: a geometric deformation technique, *Proc. ACM SIGGRAPH 98' Conf.*, 1998, 405-414.

[15] J. Feng, L. Ma, and Q. Peng, A new free-form deformation through the control of parametric surfaces, *Computers and Graphics, 20*(4), 1996, 531-539.

[16] D. R. Peachey. Solid texturing of complex surfaces, *ACM Computer Graphics (Proc. SIGGRAPH 85' Conf.), 19*(3), 1985, 279-286.

[17] E. A. Bier and K. R. Sloan, Jr., Two part texture mapping, *IEEE Computer Graphics and Applications, 6*(9), 1986, 40-53.

[18] B. Lévy and J.-L. Mallet, Non-distorted texture mapping for sheared triangulated meshes, *Proc. ACM SIGGRAPH 98' Conf.*, 1998, 343-352.

Appendix: Optimal Texture Mapping [3]

The next equation defines the errors between the curved surface and its flattened surface:

$$C_S = \sum_{P_i \in S} \sum_{P_j \in \Omega_i} \left(d_{ij} - d_{i'j'} \right)^2 ,$$

where S is a set of the points on the curved surface, Ω_i is a set of the points which exist continuous with the point P_i among S, d_{ij} is the distance between P_i and P_j on the curved surface, and $d_{i'j'}$ is the distance between corresponding two points on the flattened surface. The flattened surface is obtained by iterating the above equation to minimize C_S. Then, the above equation is rewritten in the coordinate expression:

$$C_S = \sum_{P_i \in S} \sum_{P_j \in \Omega_i} \left(d_{ij} - \sqrt{\left(X_{i'} - X_{j'} \right)^t \left(X_{i'} - X_{j'} \right)} \right)^2 ,$$

where $X_{i'} = \left(x_{i'}, y_{i'} \right)^t$ and $X_{j'} = \left(x_{j'}, y_{j'} \right)^t$, and the coordinate value which used in the next step of the iterative algorithm is obtained according to the following equation:

$$X_{i'}^{k+1} = X_{i'}^{k} - \rho \frac{\partial C_S}{\partial X_{i'}} ,$$

where ρ is a small positive number. The flattened surface of the target surface is obtained by reducing the value of C_S.