http://www.zju.edu.cn/jzus; E-mail: jzus@zju.edu.cn
ISSN 1009-3095 *Journal of Zhejiang University SCIENCE* V.4, No.6, P.623-629, Nov.-Dec., 2003

GFFD: Generalized free-form deformation with scalar fields*

QIN Xu-jia(秦绪佳)†, HUA Wei(华炜), FANG Xiang(方向)
BAO Hu-jun(鲍虎军)†, PENG Qun-sheng(彭群生)†
(*State Key Lab. of CAD & CG, Zhejiang University, Hangzhou 310027, China*)
†E-mail: {qinxj; bao; peng}@cad.zju.edu.cn

Received Sept.15, 2002; revision accepted Dec.10, 2002

Abstract: The novel free-form deformation(FFD) technique presented in the paper uses scalar fields defined by skeletons with arbitrary topology. The technique embeds objects into the scalar field by assigning a field value to each point of the objects. When the space of the skeleton is changed, the distribution of the scalar field changes accordingly, which implicitly defines a deformation of the space. The generality of skeletons assures that the technique can freely define deformable regions to produce a broader range of shape deformations.

Key words: FFD, Computer aided geometric design(CAGD), Computer aided design(CAD), Scalar field
Document code: A **CLC number:** TP391

INTRODUCTION

NURBS and polygonal meshes are two major representation schemes for modeling complex 3D objects. Although some effective shape editing tools are now available for both of them, modifying the shape of an object intuitively is still tedious work. This is particularly true for some special local or global modifications, such as adding arbitrary shaped bumps, shape bending, twisting, tapering, etc. For polygonal mesh representation scheme the problem becomes even more serious because there are no intrinsic parameters involved in this representation.

The first representation independent deformation technique was proposed by Barr (1984), which defines the deformable space with Cartesian coordinate system. The technique bends, twists, sketches or tapers objects around an axis of the coordinate system by taking the transformation of each point as a function of its position. Barr's technique was later generalized to axis-based deformation techniques (Lazarus *et al.*, 1994) by allowing the coordinate system to move along a curve during the deformation. This technique defines the deformable space by a parametric curve, in which each point in the space is parameterized with respect to the closest coordinate system on the curve. When the curve is deformed, the space and hence the embedded objects are also deformed correspondingly. Several calculation approaches were proposed to implement the axis-based deformation (Lazarus *et al.*, 1994). Generally, the axis-based deformation technique is very powerful, intuitive and efficient, but it is limited to the curve-like shape deformations. The technique was further extended to edit the local shape of a surface(Singh and Fiume, 1998), in which the deformable space is defined by several curves with their own influence regions.

A more generalized deformation approach is the free-form deformation technique (FFD), proposed by Sederberg and Parry (1986), which uses a trivariate Bézier volume to define the deformable space. The trivariate Bézier volume defined by the lattice of control points is initialized as a parallelepiped structure, which allows points in the deformable space to be parameterized easily. When the parallelepiped lattice is deformed, the new positions of points of an embedded object can be calculated using the triva-

* Projects supported by the National Natural Science Foundation of China (Nos.60133020, 60021201) and by the National Grand Fundamental Research 973 (No.2002CB312104) and the key Project of Education Ministry of China(No.01094)

riate Bézier volume. Because the technique is very powerful for modifying local or global shapes of objects with little user interaction, the free-form deformation techniques have become very popular in 3D shape design and animation.

In the past years, there were many extensions to FFD (Coquillart, 1991; Griessmair and Purgathofer, 1989; Lamousin and Waggenspack, 1994). Griessmair and Purgathofer (1989) presented similar technique based on a trivariate B-spline representation. Coquillart (1991) extended the technique (called EFFD) by utilizing a more general lattice structure to define the deformable space. But his method suffers from expensive calculation and low stability due to the usage of numerical techniques for the parameterization of the space. A recent free-form deformation technique, developed by MacCracken and Joy (1996) generalized Sederberg and Parry's work by applying the extended Catmull-Clark subdivision scheme to lattices of arbitrary topology. Such a lattice is repeatedly refined to converge to the deformable space. Instead of the simple trivariate parameterization used in previous methods, the technique directly establishes a correspondence between the deformable space and the object space. In comparison to EFFD, MacCracken and Joy's method is much faster and stable. However, all these extended approaches suffers from the laborious task of constructing complex lattices from an inventory of lattices with user interaction.

Note that deformations produced by the mentioned techniques are completely free, without imposing any constraints on objects. In some applications, however, the displacements of some points, i.e. constraints are specified in advance, the deformable space must be deformed to satisfy these constraints. Hsu *et al*. (1992) proposed a constraint-based deformation technique for calculating the deformation of the lattice in FFD to satisfy the deformation requirements on some points. Borrel and Bechmann (1991) decomposed the deformation function into the combination of a user-specified function, such as a tensor product B-spline in a box or sphere, and a linear transformation, in which the constraint-based deformation can be achieved by solving a linear system equation. These techniques can effectively decrease user's interaction.

Although the above free-form deformation techniques can be directly used to modify the shapes of polygonal meshes, the necessity for multiresolution editing or deformation techniques (Kobbelt *et al*., 1998; Lee *et al*., 1998) for editing multiresolution polygonal meshes are also widely addressed. In these techniques, the multiresolution representation for a mesh is first constructed using mesh decimation techniques. A low-resolution model is selected to define the deformable space. Each level of detail is recorded or parameterized in local frames on its next lower resolution model. As the model is deformed, the deformation is achieved by transferring the vertex displacements through the multiresolution representation. To keep objects smooth, the technique usually involves a fairing process during deformation transfer.

A disadvantage of FFD methods is in the indirect control of the deformation through adjusting control points or weights of the embedding volume. It is difficult to force the object to pass through desired points precisely. Moreover, the large number of control points in complex models makes it impractical for determining the exact number of control points to be changed and how they must be changed to produce a desired deformation. Additionally, the axis-based deformation is a supplementary technique of FFD, and is limited to the curve-like shape deformation.

In this paper, we extend the free-form deformation techniques further based on scalar fields defined by skeletons of arbitrary topology, which defines the deformable space. In this case, rather than establishing a parameterization for the deformable space, our approach just assigns a field value to each point of an embedded object. As the skeleton or the scalar field is deformed, the embedded object is correspondingly deformed by moving each point of the object in a specified direction to a new position to maintain its original field value. Since skeletons may be flexibly constructed in an intuitive manner using curves, parametric patches or polygonal meshes, this approach can generate a wide range of shape modifications.

The paper is organized as follows. In Section 2, we introduce a scalar field construction method based on skeletons. First, we independently define a distance field from a primitive of simple geometry. The blending operator is then used to

merge the distance fields into the final scalar field. In Section 3, we present several methods for tracking new positions for points of the embedded object as the field is deformed. This produces the required deformations. In Section 4, an adaptive sampling method is introduced to ensure that the embedded object undergoes the desired deformation. The implementation and results are discussed in Section 5, and the conclusions are given in Section 6.

THE SCALAR FIELD CONSTRUCTION

As we know, distance field is the simplest kind of scalar field. Technically, all the oriented manifolds can be used to construct distance fields. However, it may be time-consuming for complex manifolds. To facilitate the user interaction requirement, our scalar field construction should be implemented as fast as possible. In this section, we present an efficient way for constructing a scalar field from a skeleton consisting of some simple primitives.

Instead of directly constructing the distance field from a complex skeleton, we first calculate a distance field for each component of the skeleton, then merge all these distance fields into the final scalar field using the blending operator described in Fang *et al*. (2001). The primitives adopted by our approach include points, curves (polylines), parametric patches (open polygonal meshes) and polyhedra (closed polygonal meshes). The constructed distance fields can be represented in discrete or continuous form. Actually, the discrete field can be regarded as a sampling of the continuous field. In our current implementation, we adopt the variational interpolation scheme described in (Turk and O'Brien, 1999) to construct an approximate distance field for each primitive. As illustrated in Fig. 1, let $\{V_i\}$ be the sample points for primitive P, and d_i be the signed distance from V_i to P; the interpolation scheme calculates the distance field $f(P)$ of primitive P satisfying $f(V_i) = d_i$. For polyhedron, we usually choose the vertexes of the skeleton and the points along their normals as the sampling points. For points, curves and parametric patches, the sampling points can be chosen by some rules, for example, uniformly sample the distance field. Some isosurfaces extracted from the distance fields defined by different kind of skeletons are shown in Fig. 2.

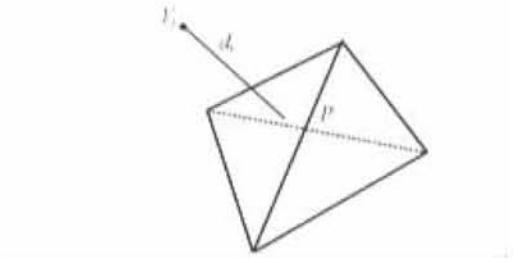

Fig. 1 Scalar field sampling

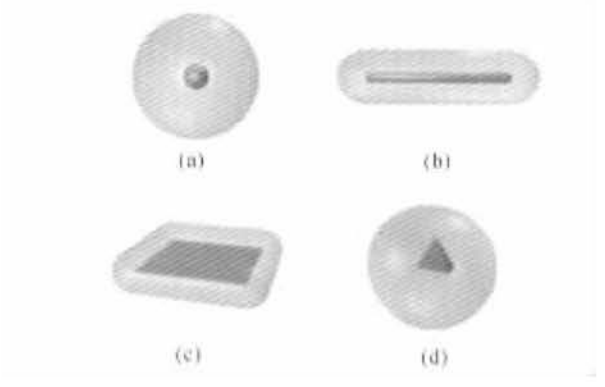

Fig. 2 Interpolated distance field of different kind of skeletons

(a) point; (b) line; (c) plane; (d) polyhedron

Note that the distance field of single primitive mentioned above is usually relatively simple, which may be unsuitable for complex shape deformations. Therefore, we further merge the distance fields defined by the components of the skeletons into the final scalar field. In order to control the influence of each primitive, an influence radius r_s is introduced for every primitive. Then the merged fields can be described as

$$\sum_{i=0}^{n} E(f_i(P)) \qquad (1)$$

Here, $f_i(P)$ is a scalar field defined by one skeleton. Function $E(d)$ is a potential energy function which defines the potential energy value for each point in that scalar field. In order to keep each field with a limited influence region, we choose the potential energy function conforming to the distribution shown in Fig. 3. The following potential distribution function is adopted. Other functions can be found in (Blinn, 1982; Nishimura *et al*., 1985; Wyvill *et al*., 1986).

$$E(d) = \exp(-f(P)) - \exp(-r_s) \quad (2)$$

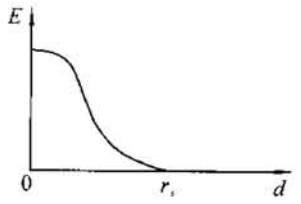

Fig.3 Potential energy distribution

Fig.4 shows a field formed by merging two fields which are defined by single skeletons.

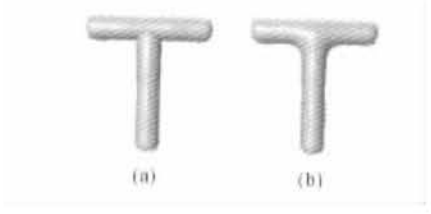

Fig.4 Merged field
(a) fields defined by two single skeletons; (b) the merged field

Constructing a skeleton from several primitives is straightforward and can be done with little user interaction. All the traditional modeling tools can be used to model primitives mentioned before. As for dense polygonal meshes, we can use their simplified models as skeletons to control their deformations. The simplified models may be generated using LOD algorithms(Eck *et al*., 1995; Hoppe, 1996).

REPOSITIONING POINTS IN THE DEFORMABLE SPACE

As discussed in Section 1, each of the previous free-form deformation techniques involves parameterization of the deformable space. The new position of a point after deformation is then directly calculated. However, in our case, since no parameterization is performed for the deformable space, we must track the new positions of points on the embedded object according to the field deformation.

Our goal is to find a new position for each point such that the point will keep its original potential value in the deformed space. Since there are many positions with the same field value, which form an iso-surface, the algorithm needs to determine adaptively the tracking direction for each point to reflect the field deformation.

One strategy is to track along the gradient direction of the field, Eq.(3). For each point in the field, its gradient direction can be calculated easily by a numerical method.

$$\boldsymbol{V} = \frac{[f_x(P), f_y(P), f_z(P)]}{\| [f_x(P), f_y(P), f_z(P)] \|} \quad (3)$$

When the skeleton is deformed, the field values of the points which locate in the influence region of the skeleton also change. These points should then move along their gradient directions until they recover their original field value. Sometimes, we cannot get satisfactory result by using such simple tracking method. In these cases, the tracking direction is determined by interpolating the corresponding points. Let P_{j0} and $P_{j1}(j = 1, 2, \cdots, n)$ be the corresponding point pairs on the original skeletons and the changed skeletons, respectively. Using the variational interpolation again, we can construct a warp function(Cohen-Or *et al*., 1998; Fang *et al*., 2001) $W(X, t)$ satisfying:

$$\begin{cases} W(P_{j0}, 0) = P_{j0} \\ W(P_{j0}, 1) = P_{j1} \end{cases} \quad j = 1, \cdots, n \quad (4)$$

For each point P in the influence region, its new position P' can be calculated by this warp function.

$$P' = W(P, 1) \quad (5)$$

And the vector defined by the point and its new position can be chosen as the tracking direction $\boldsymbol{V}$.

$$\boldsymbol{V} = (P' - P)/\| P' - P \| \quad (6)$$

The corresponding warp points can be defined by the corresponding points of changed skeletons or specified directly by users.

For the embedded object, the field values of its vertexes were recorded when it was embedded into a scalar field. After the field was deformed, the new position of a point can be determined by the following functions.

For current point P, while($|\delta| \geqslant \varepsilon$)
// δ is the current point field value alteration,
// ε is the approaching accurate

```
{
 calculate the tracking direction V
ΔV = δ • V;
P' = P + ΔV
δ' = exp⁻¹(E(P')) − exp⁻¹(e₀)
// e₀ is the original field value
while(δ • δ' < 0) {
      ΔV = 0.5ΔV
      P' = P + ΔV
      δ' = exp⁻¹(E(P')) − exp⁻¹(e₀)
}
if |δ| > |δ'|
    P = P', δ = δ'
else ΔV = −ΔV
}
```

Fig.5 Calculate the new position of a point

ADAPTIVELY SAMPLING THE EMBEDDED OBJECTS

The deformation may take place in local region or nonuniformly, then the embedded object will be deformed unsmoothly if the mesh is not dense enough. In order to keep the smoothness of the embedded object after deformation, new sampling points are added adaptively where needed.

During the deformation process, the points within the influence regions are moved first. Then for each cell on the embedded object mesh, we check its edge length after the deformation. If one edge length becomes too long, a new sampling point is added at the middle point of the original edge, whose field value takes the average of that on the two end points. The added sampling points are then also moved conforming to the deformation constraints. This sampling procedure is repeated until no additionary sampling is necessary and the deformation region within the portion of the embedded object is adaptively resampled. The adaptive sampling procedure is described below:

```
Move the mesh vertexes in the influence regions with the
procedure shown in Fig.5.
for every cell of the mesh which contains moving points {
for every edge {
   calculate the new edge length |E'ᵢ|
   if |E'ᵢ| ≥ 1.8|Eᵢ| {
   P = 0.5(V_i0 + V_i1)
// Here V_i0 and V_i1 are the start and end
// point of current edge separately.
   d = 0.5(E(V_i0) + E(V_i1))
   }
   }
repolygonize the current cell
Move the added sampling points conforming to the de-
formation constraints.
}
Repeat the above procedure until no additional sam-
pling is necessary.
```

Fig.6 Adaptive sampling procedures

IMPLEMENTATIONS AND EXPERIMENTAL RESULTS

Before deforming a model, users must specify skeletons to define a scalar field so as to control the deformation. Then users can deform this model by two ways. One is to edit the skeleton shape interactively. After the skeleton is edited, its corresponding field is also changed. Fig. 7 and Fig. 10 show the deformation results by editing the shape of the skeleton directly. Users can also deform an embedded model by replacing the original skeleton with another one. In Fig. 8 and Fig. 9, the embedded models are deformed using this method. Because the gradients of the two fields are different, the deformation results produced are also different. By calculating the point moving direction from a corresponding points warping function, twisting a local region can be implemented. This is shown in Fig. 11. The skeleton is obtained from the LOD model of the original model.

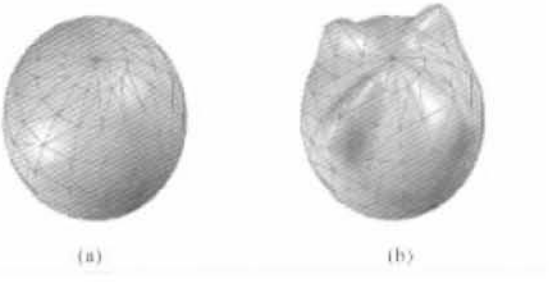

Fig.7 Deform the embedded model by stretching the field skeleton directly

(a) original model; (b) deformed model

Additionally, the deformation of the object is only dependent on the changes of the scalar fields. So the deformation value of the object is

relative to the size of the warping skeleton. When the object surfaces are close to the skeleton, the skeleton changes will excessively impact on the object (shown in Fig.8). Otherwise, if the embedded object is far from the skeleton, the skeleton changes have little effect on the object (shown in Fig.9).

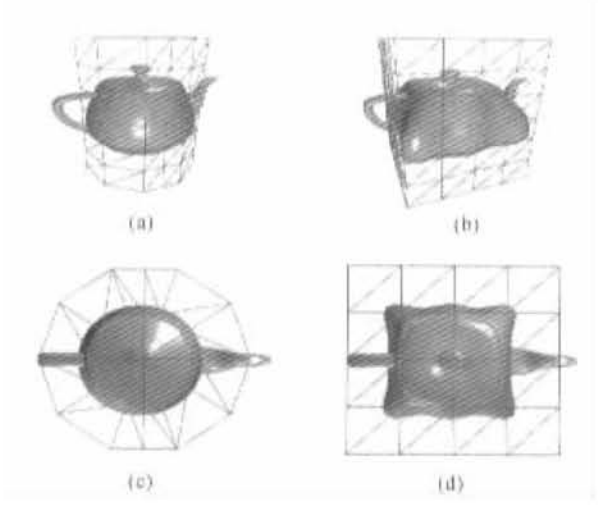

Fig.8 Deform the embedded model by replacing the field skeleton

(a),(c) the model embedded in a field defined by a cylinder skeleton; (a),(b) the side view; (c),(d) the top view; (b),(d) the model deformed after the field skeleton was replaced

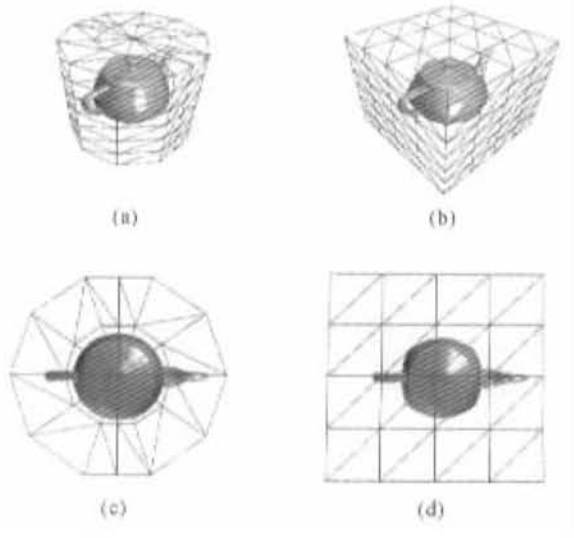

Fig.9 The embedded model was deformed by replacing field skeleton

(a),(c) original model and its skeleton; (b), (d) replaced skeleton and deformed model

Modifying the influence radius can control the deformation region. Points on the embedded object out of the influence region will not be moved. This can avoid degeneration or topology change of the object after deformation.

This deformation algorithm was implemented in C++ on a PC. Since there is little limitation on the shape of the field skeletons, users can control the deformation intuitively and accurately.

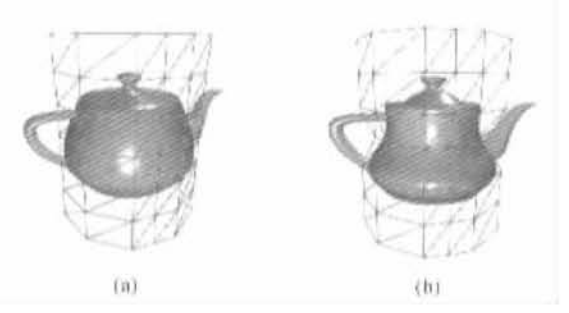

Fig.10 The teapot model was deformed by editing the field skeleton

(a) original model; (b) deformed model

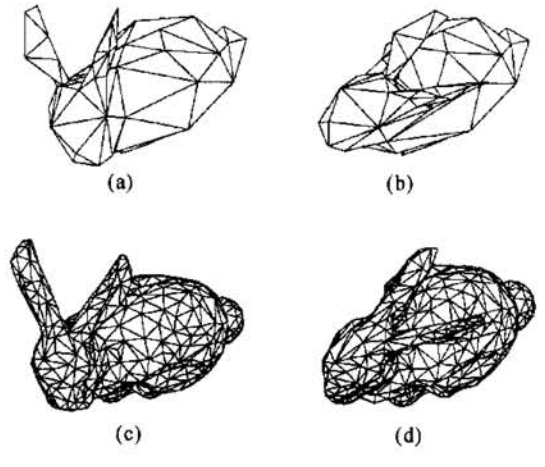

Fig. 11 Turn a bunny's head by calculating the points moving direction using corresponding points warping function

(a), (b) the field skeleton; (c), (d) the embedded model

CONCLUSIONS

We have described a generalized free-form deformation technique which controls the deformation by a scalar field. Any object can be em-

bedded into a scalar field and be deformed by changing the field. Because the scalar field is defined by 3-dimensional skeletons and there is little limitation on the skeleton types, this deformation method can implement most deformation functions, such as Barr's method, Axis Deformation and FFD. On the other hand, users can easily construct a skeleton and control the deformation intuitively.

By choosing the influence radius, the deformation region can be controlled. Since no point-to-point correspondence between the object space and the deformation space is available, the vertexes on the embedded model may miss moving directions during deformation. This problem can be solved by constructing a local parameterization.

References

Barr, A., 1984. Global and local deformations of solid primitives. *Proceedings of SIGGRAPH'84, Computer Graphics*, **18**(3): 21 – 30.

Blinn, J., 1982. A generalization of algebraic surface drawing. *ACM Trans. Graphics*, **1**(3): 135 – 256.

Borrel, P. and Bechmann, D., 1991. Deformation of N-dimensional objects. *International Journal of Computational Geometry & Applications*, **1**: 427 – 453.

Cohen-Or, D., Levin, D. and Solomovici, A., 1998. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics*, **17**(2): 116 – 141.

Coquillart, S., 1991. Extend free-form deformation: A sculpturing tool for 3D geometric models. *Proceedings of SIGGRAPH'91, Computer Graphics*, **25**(4): 21 – 26.

Eck, M., Derose, T., Duchamp, T., Hoppe, H., Lounsbery, M. and Stuetzle, W., 1995. Multiresolution Analysis of Arbitrary Meshes. Proceedings SIGGRAPH '95, ACM Press, New York, p.173 – 182.

Fang, X., Bao, H., Heng, P. A., Wang, T. T. and Peng, Q.S., 2001. Continuous field-based free-form surface modeling and morphing. *Computers & Graphics*, **25**(2): 235 – 243.

Griessmair, J. and Purgathofer, W., 1989. Deformation of Solids with Trivariate B-Splines. Eurographics'89, North-Holland, p.137 – 148.

Hoppe, H., 1996. Progressive Meshes. Proceedings of SIGGRAPH'96, Annual Conference Series, ACM Press, New York, p.99 – 108.

Hsu, W., Hughes, J. and Kaufmann, H., 1992. Direct manipulation of free-form deformations. *Computer Graphics*, **26**: 177 – 184.

Kobbelt, L., Campagna, S., Vorsatz, J. and Seidel, H.P., 1998. Interactive Multi-Resolution Modeling on Arbitrary Meshes. Proceedings of SIGGRAPH'98, ACM Press, New York, p.105 – 114.

Lamousin, H. J. and Waggenspack, W. N., 1994. NURBS-based free-form deformation. *IEEE Computer Graphics and Applications*, **14**(6): 59 – 65.

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

Lee, A., Sweldens, W., Schröder, P., Cowsar, L. and Dodkin, D., 1998. MAPS: Multiresolution Adaptive Parameterization of Surfaces. Proceedings of SIGGRAPH'98, ACM Press, New York, p.95 – 104.

MacCracken, R. and Joy, K.I., 1996. Free-Form Deformation with Lattices of Arbitrary Topology. Proceedings of SIGGRAPH'96, ACM Press, New York, p.181 – 188.

Nishimura, H., Hirai, M., Kawai, T., Kawata, T., Shirakawa, I. and Omura, K., 1985. Object modeling by distribution function and a method of image generation. *Transactions of the Institute of Electronics and Communication Engineers of Japan*, **J68-D**(4): 718 – 725.

Sederberg, T. W. and Parry, S., 1986. Free-form deformation of solid geometric models. *Proceedings of SIGGRAPH'86, Computer Graphics*, **20**(4): 151 – 160.

Singh, K. and Fiume, E., 1998. Wires: A Geometric Deformation Technique. Proceedings of SIGGRAPH'98, ACM Press, New York, p.405 – 414.

Turk, G. and O'Brien, J., 1999. Shape Transformation Using Variational Implicit Functions. Proceedings of SIGGRAPH'99, ACM Press, New York, p.335 – 342.

Wyvill, G., McPheeters, C. and Wyvill, B., 1986. Data structure for soft objects. *The Visual Computer*, **2**(4): 227 – 234.