HAL
open science

Extended free-form deformation : a sculpturing tool for 3D geometric modeling

Sabine Coquillart

To cite this version:

Sabine Coquillart. Extended free-form deformation : a sculpturing tool for 3D geometric modeling. [Research Report] RR-1250, INRIA. 1990. inria-00075308

HAL Id: inria-00075308

https://inria.hal.science/inria-00075308

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.: (1) 39 63 55 11

Rapports de Recherche

1992

25ème anniversaire

N° 1250

Programme 6
Robotique, Image et Vision

EXTENDED FREE-FORM DEFORMATION : A SCULPTURING TOOL FOR 3D GEOMETRIC MODELING

Sabine COQUILLART

Juin 1990

RR_1250

Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling

Sabine Coquillart

INRIA-Rocquencourt
Domaine de Voluceau
78153 Le Chesnay, France

Abstract

Current research efforts focus on providing more efficient and effective design methods for 3D modeling systems. In this paper a new deformation technique is presented. Among other things, arbitrarily shaped bumps can be designed and surfaces can be bent along arbitrarily shaped curves.

The purpose of this research is to define a highly interactive and intuitive modeling technique for designers and stylists. A natural way of thinking is to mimic traditional trades, such as sculpturing and moulding.

Furthermore, with this deformation technique, the modeling tool paradigm is introduced. The object is deformed with a user-defined deformation tool.

This method is an extension of the Free-Form Deformation (FFD) technique proposed by Sederberg and Parry [17].

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]:Computational Geometry and Object Modeling - Curve, surface, solid, and object representation; Geometric algorithms, languages, and systems; Hierarchy and geometric transformations; I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction techniques.

Additional Keywords and Phrases: Solid geometric modeling, deformations.

---

*Presently on sabbatical at Thomson Digital Images, Paris

Déformation de Surfaces et Modélisation Géométrique 3D

Résumé

Le processus de modélisation géométrique peut être décomposé en deux phases : la construction des objets ou surfaces puis leur déformation. Ces deux étapes sont necessaires dès que l'objet est trop complexe pour être créé directement sous sa forme définitive.

Les systèmes de modélisation offrent des méthodes de création de plus en plus performantes et conviviales les techniques offèrtent pour déformer sont souvent beaucoup plus frustes. Ce papier présente une nouvelle technique interactive de déformation de surfaces ou d'objets.

L'objectif des recherches exposées est de définir une méthode de déformation interactive et intuitive, utilisable par des stylistes et des animateurs. Une façon naturelle de procéder consiste à s'inspirer de techniques issues des métiers traditionnels tels que la sculpture ou le moulage.

La technique de déformation présentée ici exploite la notion d'outil. L'outil est façonné par l'utilisateur puis stocké. Il peut ensuite etre appliqué à volonté sur différentes surfaces ou objets pour les déformer.

Cette méthode est une extension de la méthode connue sous le nom de FFD ou "Free-Form Deformation" proposée par Sederberg et Parry [17].

1 Introduction

Geometric modeling has always been a major research area in computer graphics. Geometric modeling includes both the definition of the geometric model and the development of design methods. Often, systems offer design methods imposed by the underlying geometric model or use geometric models imposed by the design methods. This solution is efficient for specific applications. However, general modeling systems require less specific geometric models and several design methods that are as easy as possible to use and that can be combined with each other to increase the power of the system. A growing trend is thus to dissociate the underlying geometric model and the design methods so that the geometric model becomes transparent to the user.

This paper describes an interactive deformation technique independent of the geometric model. As we wanted to define a highly interactive and intuitive modeling technique usable by designers and stylists, it was natural to try to mimic traditional tools, such as sculpturing or moulding. The use of the sculpturing metaphor for geometric modeling is not recent. Several authors have suggested tools that allow a designer to see the design operations as *sculpturing tools* [12, 19, 7, 2, 1, 5, 17, 8, 15].

Our goal is to change the shape of an existing surface either by adding arbitrarily shaped bumps to it or by bending it along an arbitrarily shaped curve. Four problems must be considered:

- The position of the deformed region on the surface.
- The size of the deformed region.
- The shape of the boundary of the deformed region.
- The shape of the deformed region (inside the boundary).

A common practice consists of interactively moving the control points of a spline surface. This solution is not satisfactory for the following reasons:

- The number of control points the user will have to move depends on the size of the deformed region. For example, the design of a large bump may require moving many control points whereas designing small bumps may be impossible.
- The shape of the deformed region (both along its boundary and within its interior) is imposed by the shape of the surface isoparametric lines, that is, by the position of the neighbouring control points. Designing a bump with a circular boundary is almost impossible.

- The position of the deformed region on the surface is imposed by the position of the control points since only the control points are moved.

Some of these problems, namely small bumps, can be partially solved by using refinement techniques. Note, however, that refinement has the unpleasant property of being non-local — it causes regions far from the region of interest to be refined as well.

In [14] and [15] Piegl proposes a combination of control point-based and weight-based modifications. The weight-based technique, valid for rational B-spline surfaces, is a nice solution for the size problem. The position problem is partially solved by an automatic refinement technique.

In [8], Forsey and Bartels describe a new geometric model where a surface is represented as a hierarchy of refined surfaces. This representation solves the size problem as the user can choose the resolution of each region of the surface. However, the shape problem is not considered since the shape of the deformation is still influenced by the position of the neighbouring control points. The position problem is not solved either because the control point positions are fixed.

In [1], Barr suggests a set of powerful transformations for deforming a solid object. The transformations he presents include stretching, bending, twisting, and tapering operators. In spite of the fact that arbitrarily shaped deformations are not possible, it is a very efficient method.

Cobb [5] presents the first modeling tool allowing the user to define bumps with different shapes. She extends the basic warp technique previously discussed in [12, 19, 7, 2] and introduces the region warp and the skeletal warp. With region warp, the user specifies a polygonal region that defines the shape of the warp boundary. Skeletal warp is a variation of the region warp where the region is defined by its skeleton. The size and the position of the deformed region, as well as the shape of its boundary, are user defined without any limitations but the shape of the interior is not free. Notice that Cobb solves most of the previously listed problems by the addition of a structure which consists of a region or of a skeleton. This structure is independent of the surface geometry. The user does not need to know the underlying geometric model to deform the surface.

Sederberg and Parry [17] present a powerful deformation tool in which the representation of the surface is also hidden by a FFD lattice embedding the object. The deformations of the FFD lattice are automatically passed to the object. FFD has proved to be a very intuitive and efficient modeling technique highly appreciated by designers [3]. It solves the size and the position problems but not the shape one. The intrinsic parallelepipedical shape of the FFD lattice prohibits arbitrarily shaped deformations.

This paper introduces an extension of the FFD technique called EFFD, for Extended Free-Form Deformation. The new method uses non-parallelepipedical 3D lattices. The shape of the user defined lattice will induce the shape of the deformation. This paper mainly describes surface deformation although the technique is suitable for object deformation as well. Deformations produced by this technique are more general than Cobb's warps, they are not restricted to bumps, and all the advantages of FFD are not only retained but extended. In addition, both the boundary and the interior of the deformation are arbitrarily shaped.

After a presentation of our implementation of Sederberg and Parry's FFD technique, the EFFD method is described. The steps of the deformation process are detailed and different classes of EFFD lattices are presented. Finally, some examples illustrate our approach.

2 Free-Form Deformations

Free-Form Deformation (FFD) [17, 16, 13] consists of embedding the geometric model or the region of the model that has to be deformed into a parallelepipedical 3D lattice regularly subdivided, as shown in Figure 1. The deformations of the FFD lattice are then automatically passed to the model. Let l, m and n be the number of subdivisions along each of the three directions, U, V and W. These numbers can be chosen by the user depending on the deformation he wants to produce (in Figure 1, $l = 2$, $m = 1$ and $n = 2$).

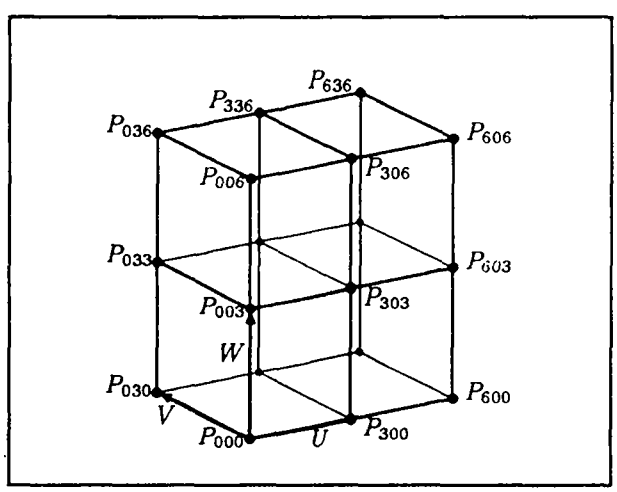

Figure 1: A parallelepipedical lattice

In our implementation the 3D lattice is represented by a tensor product piecewise tricubic Bézier volume. This volume is defined by an array of $(3l+1)\times(3m+1)\times(3n+1)$

control points P_{ijk}. Each subdivision element, also named "chunk" by Clark in [4], is thus defined by :

$$L(u,v,w) = \sum_{i,j,k=0}^{3} B_i(u)B_j(v)B_k(w)P_{ijk} \tag{1}$$

with $0 \leq u, v, w \leq 1$, where the $B_i(t)$ are the degree 3 Berstein polynomials, the P_{ijk} are the chunk control points.

The Free-Form Deformation technique is decomposed into two steps :

- Before deforming the 3D lattice, the coordinates u_s, v_s and w_s, in the lattice parameter space, of each object point are computed. With parallelepipedical lattices, this step requires only the solution of three linear equations. For any point X interior to the lattice, $0 < u_s < l$, $0 < v_s < m$ and $0 < w_s < n$.
- After deforming the 3D lattice, the deformed positions of the object points are computed. The deformed position X_{ffd} of an arbitrarily point X with coordinates (u_s, v_s, w_s) in the lattice parameter space is computed in two steps. First, determine the chunk where the point lies by computing the floor values (u_0, v_0, w_0) of u_s, v_s and w_s. Let $u = u_s - u_0, v = v_s - v_0$ and $w = w_s - w_0$ be the X coordinates in the chunk parameter space. The second step consists of computing the Cartesian coordinates of X_{ffd} from u, v, w and the matrix of the $4 \times 4 \times 4$ control points P_{ijk} of the chunk, according to equation (1).

Tensor product Bézier volumes are used throughout the paper. Naturally, as claimed by Sederberg and Parry, other bases such as B-splines or volumes of higher degree could be considered as well. The piecewise structure of the volume allows the user to design local deformations on the 3D lattice. This will be very important for the proposed extensions.

In our implementation the deformation is specified by moving the $(l+1) \times (m+1) \times (n+1)$ control points (the P_{3i3j3k}) corresponding to the corner control points of the volume elements (or chunks). Only these points are represented on Figures 1 and 2. The tangents at the corner control points can also be modified by the user. The other control points are automatically updated. Two modes exist for the manipulation of corner control points. Constant tangent mode, where the tangents of the point remain constant when the point is moved, and non-constant tangent mode where the tangents of the point are updated according to the position of the neighbour points simulating a C-Spline interaction [4]. These two modes can be chosen independently for each of the three directions.

3 Extended Free-Form Deformations

FFD is a very intuitive modeling technique but it is too restrictive to allow real sculpturing of surfaces. The restriction is mainly due to the shape of the lattice. As seen previously, FFD solves only the size and the position problems but not the shape one. For example, defining a circular bump on a surface is not possible with FFD (see Figure 5a). One would like to use a cylindrical lattice instead of the parallelepipedical one (see Figure 5b). The EFFD technique presented in this paper allows arbitrarily shaped deformations by using non-parallelepipedical lattices. EFFD lattices are equivalent to FFD lattices; only the initial lattice shape is different. The EFFD technique can be described in four steps:

1. Editing an EFFD lattice.
2. Associating an EFFD lattice with the surface.
3. "Freezing" an EFFD lattice.
4. Deforming the surface.

Notice that the EFFD lattice is defined independently of the surface to which it will be applied. The EFFD lattice is a deformation tool that is designed by the user and stored into a toolbox or a library until it is used. The modeling tool paradigm faithfully reproduces traditional tools and greatly increases the power of the modeling system. The user can adapt the modeling system to his needs by defining his own tools. Each of the four steps of EFFD will now be explained in detail.

3.1 EFFD lattices

3.1.1 Prismatic lattices

The prismatic lattice is a very significant special case. Prismatic lattices are especially useful for applying a deformation to a surface. We have seen previously that the deformation technique consisting of moving interactively the control points of a spline is not satisfactory because the shape, the size, and the position of the deformation are constrained by the geometry of the surface. The purpose of prismatic lattices is to redefine the geometry of the surface. From a user point of view, the geometry as well as the type (polygonal, B-spline, Bézier...) of the surface are hidden by a new user defined structure, the EFFD lattice. The prismatic lattice is positioned on the

surface such that the surface passes through the lattice (see Figures 6a to 11a). Only the corner control points, the P_{3i3j3k} are shown on the shaded pictures presented in this paper. Then, the user works directly on the EFFD lattice by moving some of its points and the deformations are automatically passed to the surface. All the surface points inside the EFFD will be deformed. The EFFD can be applied to non-planar surfaces or, for example, to surfaces that have already been deformed with another EFFD lattice. The height of the prismatic lattice must thus be adjusted such that the desired region of the surface fits into it. The shape of the prismatic lattice is of paramount importance. Control points and consequently isoparametric lines must be carefully positioned in order to allow the desired deformation.

Two classes of prismatic lattices are defined, the elementary prismatic lattices and the composite prismatic lattices. There is no restriction on the shape of elementary prismatic lattices. All prismatic lattices obtained by moving or merging any points of a parallelepipedical lattice are valid. It is therefore advised not to define lattices that intersect themselves. The cylindrical lattice (see Figure 2) is a useful lattice obtained by welding two opposite faces of a parallelepipedical lattice and by merging all the points of the cylinder axis. Control points along one of the directions (V on Figure 2) are defined in order to approximate circles. An exact representation of a cylindrical lattice is only possible with rational splines. Other elementary prismatic lattices can be designed by moving and merging some of the points of a parallelepipedical lattice.

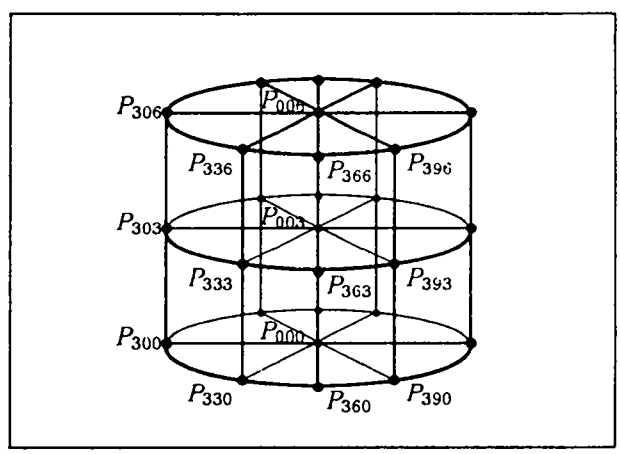

Figure 2: A cylindrical lattice

3.1.2 Composite prismatic lattices

Elementary prismatic lattices are not general enough. Composite lattices must be introduced in order to allow the design of some unconnected shapes (see the "8" example in Figure 9). Composite prismatic lattices are defined as several elementary lattices welded together; see 3.1.4 below.

3.1.3 Non-prismatic lattices

Non-prismatic lattices can also be used to create deformations of objects and some non-prismatic lattices such as spherical lattices can be very attractive. Composite non-prismatic lattices are also valid. However, the use of lattices which are too complex can lead to unpredictable results.

3.1.4 EFFD lattice design process

From a user point of view, an EFFD lattice is defined either from a predefined three-dimensionnal lattice or from two-dimensionnal lattices.

- Predefined EFFD lattices include parallelepipedical and cylindrical lattices. The number of subdivisions (or chunks) along each axis is user definable. In Figure 6a, a predefined cylindrical lattice with respectively 2, 12 and 1 subdivisions along each of the three U, V and W axis has been selected. This lattice has then been transformed by selecting one plane of points out of two and by moving them toward the axis. Valid editing methods include moving (both points and tangents can be moved either alone or as a group), merging, inserting (by subdividing the lattice) and removing points.
- EFFD lattices can also be created from two-dimensional lattices in the same way as surfaces are defined from curves (loft, sweep, extrusion,...). 2D lattices are similar to surfaces. Traditional surface modeling methods are employed to define them. Valid editing methods for non-predefined 3D lattices are the same as for predefined 3D lattices. In the "S" example (see Figure 8a), the 3D EFFD lattice is defined from a two-dimensional lattice. The 2D lattice is a loft on three curves, two of them being an offset from the middle one.

Two elementary two-dimensional lattices can be welded together in order to form a composite two-dimensional lattice and further a composite three-dimensional lattice.

The welding operation is realised by merging the points of each lattice. Two or more points of the same lattice can also be merged. When merging two points ($P0$ and $P1$), two of their tangents ($t0$ with $t1$ and $t'0$ with $t'1$) are merged either automatically or on user request such that merged points are equivalent to other points (see Figure 3). In order to be able to assure tangent continuity at a merged point, the two tangents $t''0$ and $t''1$ must be marked as aligned, which is also done either automatically or by user request.

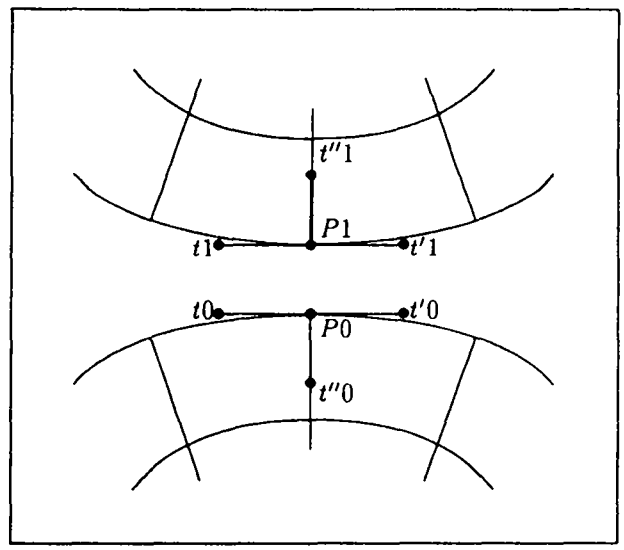

Figure 3: Merging two points

Some tricky cases cannot be solved automatically, such as the one representing the center point of the "8" lattice, in Figure 4. In this case, the four points, $P0$, $P1$, $P2$ and $P3$ are merged together, as well as $t0$, $t1$, $t2$, $t'2$ and $t'0$, $t'1$, $t3$, $t'3$.

When several points are merged together, such as the center of a disc, some continuity problems may occur. These problems are discussed in paragraph 3.1.5. While the implemented welding method is very simple, some more sophisticated ones such as the one presented in [9], could be implemented as well. In the "8" example (see Figure 9a), the 3D EFFD lattice is defined from a composite two-dimensional lattice made from 3 elementary two-dimensional lattices, two discs and an exterior lattice.

In the future, more specific two-dimensional lattice design methods will be developed. An example of these methods is to automatically compute the 2D lattice from either the skeleton of the shape or from its boundaries.

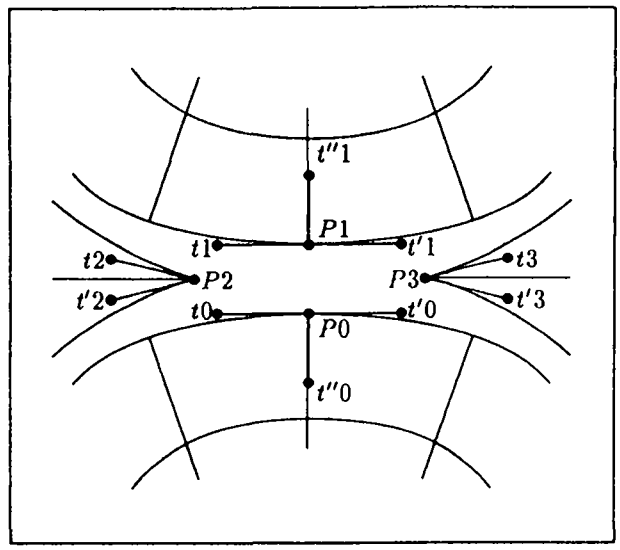

Figure 4: Merging four points

3.1.5 Continuity versus complexity

Continuity is one of the most important problems to consider when working with piecewise surfaces or volumes. Before examining continuity constraints for volumes, let us recall some results on piecewise surfaces continuity; see [6] for a complete survey. Assuming non-degenerate 4-sided cubic patches, known results are as follows:

- C^1 and G^1 smooth connection between patches defined over a topologically rectangular network can be guaranteed.
- C^1 continuity cannot be guaranteed if more than 4 patches meet at a point.
- For G^1 continuity around an n-patch corner (n>4), constraints intertwining often requires either to subdivide patches or to increase their degree (cf. [6]).

With degenerate patches constraints propagation is even more important. With volumes the problem is more tricky. Surface continuity property can easily be extended to prismatic volumes but general non-prismatic volumes can lead to unsolved continuity problems. Even when a solution to the continuity problem exists, maintaining this continuity may be penalizing for the EFFD technique. For example, as continuity constraints require the increase of the degree or of the subdivision level of chunks, editing points have to be added automatically. This is not convenient for the user and allowing only lattices for which continuity problems are easily solved (without adding points)

is too restrictive. Our choice is thus not to restrain volume complexity but rather to insure lattice continuity only for the simplest cases. What is important for the user is the surface continuity but not the lattice continuity. Depending on the surface type, it is often possible to guarantee the surface continuity even if lattice continuity is not assured (for example with spline surfaces). Thus, from our point of view, lattice continuity is not a primary concern.

3.2 Associating a lattice with the surface

The next step consists in taking an EFFD lattice out of the library and associating it with the desired surface. A list of EFFD lattices may be associated with the surface. Associating an EFFD lattice with a surface consists of adding the lattice to the list. While an EFFD lattice is associated with a surface, one can still edit it without deforming the surface. At this time, an attractive capability is the positioning command which allows moving the EFFD lattice to a user specified point on the surface.

3.3 Freezing a lattice

Everything is now ready to deform the surface. Assuming that several lattices are associated with the surface, the user must first select one of the EFFD lattices and "freeze" it. Freezing a lattice consists in computing the u_s, v_s and w_s coordinates of each point of the surface in the EFFD lattice parameter space. For each surface only one EFFD lattice can be frozen at a time. With arbitrarily shaped lattices, finding the (u_s, v_s, w_s) coordinates of the surface points is decomposed into two steps. First, the chunk where the point is supposed to lie is determined by using the convex hull property of Bézier volumes. The (u, v, w) coordinates inside the chunk are then computed using Newton approximation. Two problems have to be considered: the technique convergence and the degenerated chunks treatment.

- The convergence and consequently the determination of the starting point of Newton iteration is usually considered as a delicate problem. However, for our problem, experience has proved that choosing $u = 0.5$, $v = 0.5$ and $w = 0.5$ as a starting point leads to very good convergence. No divergent cases have been so far noted. A simple solution has thus been chosen. It consists of subdividing the chunk in order to get a better starting point when no convergence is detected.

- With degenerated chunks, matrix inversion required by the Newton technique may not be possible because of differential vanishing. In this case, as proposed by Lukács in [11], the pseudo-inverse matrix method is used.

3.4 Deforming the surface

When an EFFD lattice is frozen, all the transformations applied by the user to the lattice are passed to the surface when the user selects the update command. Only moving transformations are valid for frozen lattices. The C^1 continuity along the intersection of an exterior face of the lattice with the object can be assured either by keeping the two planes of control points adjacent to the lattice border fixed or by guaranteeing the surface continuity as suggested in 3.1.5. The computation of the X_{ffd} coordinate points of the deformed surface is equivalent to the FFD one.

The presented method has been implemented with polygonal surfaces but, as FFD, it also works with other surfaces, such as spline surfaces, and it should work with hierarchical surfaces [8] as well. Whatever surface is used, a subdivision technique such as that of Griessmair et al. [10] is recommended in order to maintain an acceptable resolution of the surface. The technique of Griessmair et al. is valid for polygonal surfaces. Each polygon is subdivided into triangles that are again subdivided according to a given accuracy threshold.

Considering a surface with several lattices positioned on it, the deformation process can be described as follows:

```
Loop 1:
    Deform the unfrozen lattices (move, insert, remove and merge control points)
    Freeze one of the surface EFFD lattices
    Loop 2:
        Deform the frozen EFFD lattice (move points)
        Update the surface
    End loop 2
    Unfreeze the EFFD lattice
End loop 1
```

The ability to work with several EFFD lattices associated with the same surface is very important; it allows the user to apply successively different shaped deformations.

In order to allow for an exact repetition of the same deformation on several surfaces, a recording operator has to be implemented.

4 Examples and concluding remarks

Some simple examples of surfaces deformed using the EFFD technique are illustrated in Figures 6 to 11. Figures 6a to 11a present the initial surfaces with the EFFD lattices positioned on them. In Figures 6b to 11b, the EFFD lattices have been frozen, some of their points have been moved, and the surfaces updated. Both the deformed lattices and the deformed surfaces are shown. In Figures 6c to 11c, shaded pictures of the resulting surfaces or objects are shown.

In Figures 6 and 7, the same EFFD lattice (see Figure 7a) is used to define two different deformations. In Figure 6, the axis and the intermediate cylinders of points are translated whereas in Figure 7, the axis and every second column of points of the intermediate cylinder have been moved back. As shown in Figures 6c and 7c, sandpies are easily modeled with EFFD. In Figures 8 and 9 two characters are impressed onto a surface. The "8" is sculptured into a piece of marble by "pulling" some of the lattice points whereas the granite "S" is sculptured by "pushing" the points.

Sculpturing and moulding are accurately simulated by EFFD. Other types of deformations can also be reproduced with this technique. The shape of cloth-like surfaces can also be simulated. Figures 10 and 11 are two examples where folds are modeled with EFFD. In Figure 10c, a leather-like cushion is shown. Starting with a surface of revolution embedded into a cylindrical EFFD lattice, the points of the lattice axis are first moved in order to create a hull at the center of the cushion, then the folds are designed by moving some of the intermediate points of the lattice (see Figure 10b). In Figure 11, an oilcloth on a round table has been modeled. Starting with a planar surface embedded into a cylindrical lattice, the outermost points of the EFFD lattice are moved as shown in Figure 11b to create the folding effect. The resulting textured picture is shown in Figure 11c.

EFFD is an easy to use and efficient method for modeling cloth-like surfaces. Shapes cannot, of course, be as natural as with physical methods [20] [18] but it can be an interesting alternative when other methods are computationally prohibitive or when naturalness is not the main objective.

Deforming a surface with EFFD technique is very efficient. Only a few minutes were needed to design most of the previous examples. It is very easy to implement EFFD on a system including the FFD capability. This deformation technique is part of ACTION3D, a general interactive modeling system developed jointly by SOGITEC and INRIA.

5 Acknowledgements

I would like to thank Laurent Alt, Wen-Hui Du, Michel Gangnet, Tony Kasvand, and Marie-Luce Viaud for helpful discussions and for reviewing early drafts of this paper. I am grateful to INRIA's audiovisual department for their assistance with color images and video demonstrations. I would also like to thank the reviewers for their helpful comments.

References

[1] A. H. Barr. Global and Local Deformations of Solid Primitives. In *SIGGRAPH'84*, volume 18, pages 21–30. ACM, July 1984.

[2] W.E. Carlson. *Techniques for the Generation of Three Dimensional Data for Use in Complex Image Synthesis.* PhD thesis, Ohio State University, 1982.

[3] J.E. Chadwick, D.R. Haumann, and R.E. Parent. Layered Construction for Deformable Animated Characters. In *SIGGRAPH'89*, volume 23, pages 243–252. ACM, 1989.

[4] J.H. Clark. Parametric curves, surfaces and volumes in computer graphics and computer-aided geometric design. Technical Report 221, Stanford University, 1981.

[5] B.S. Cobb. *Design of Sculptured Surfaces Using the B-Spline Representation.* PhD thesis, University of Utah, June 1984.

[6] W.H. Du and F.J.M. Schmitt. Free-Form Surface Modelling using Tensor Product Bézier Patches: A Review with New Solutions. Technical Report Télécom Paris 89 D 014, Ecole Nationale Supérieure des Télécommunications, 1989.

[7] J.P. Duncan and G.W. Vickers. Simplified Method for Interactive Adjustment of Surfaces. *Computer Aided Design*, 12(6):305–308, November 1980.

[8] D.R. Forsey and R.H. Bartels. Hierarchical B-Spline Refinement. In *SIGGRAPH'88*, volume 22, pages 205–212. ACM, August 1988.

[9] M.P. Gascuel. Welding and Pinching Spline Surfaces: New Methods for Interactive Creation of Complex Objects and Automatic Fleshing of Skeletons. In *Graphics Interface'89*, pages 20–27, 1989.

[10] J. Griessmair and W. Purgathofer. Deformation of Solids with Trivariate B-Splines. In *EUROGRAPHICS'89*, pages 137–148. North-Holland, 1989.

[11] G. Lukács. The Generalized Inverse Matrix and the Surface-Surface Intersection Problem. In *Theory and Practice of Geometric Modeling*, pages 167–185. Springer-Verlag.

[12] R. E. Parent. A System for Sculpting 3-D Data. In *SIGGRAPH'77*, volume 11, pages 138–147. ACM, July 1977.

[13] S.R. Parry. *Free-Form Deformations in a Constructive Solid Geometry Modeling System.* PhD thesis, Brigham Young University, 1986.

[14] L. Piegl. Modifying the Shape of Rational B-Splines. Part 1 : Curves. *Computer Aided Design*, 21(8):509–518, October 1989.

[15] L. Piegl. Modifying the Shape of Rational B-Splines. Part 2 : Surfaces. *Computer Aided Design*, 21(9):538–546, November 1989.

[16] T.W. Sederberg and S.R. Parry. Free-Form Deformation of Polygonal Data. In *Second Image Symposium*, pages 633–639. CESTA, April 1986.

[17] T.W. Sederberg and S.R. Parry. Free-Form Deformation of Solid Geometric Models. In *SIGGRAPH'86*, volume 20, pages 151–160. ACM, August 1986.

[18] D. Terzopoulos and K. Fleischer. Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture. In *SIGGRAPH'88*, volume 22, pages 269–278. ACM, August 1988.

[19] G.W. Vickers, J.P. Duncan, and V. Lee. Interactive Surface Adjustment of Marine Propellers. *Computer Aided Design*, 10(6):375–379, November 1978.

[20] J. Weil. The Synthesis of Cloth Objects. In *SIGGRAPH'86*, volume 20, pages 49–54. ACM, August 1986.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique

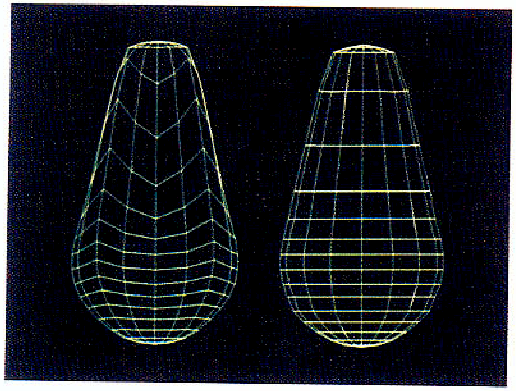

Figure 5 a: A sphere deformed with a parallelepipedical lattice
b: A sphere deformed with a cylindrical lattice

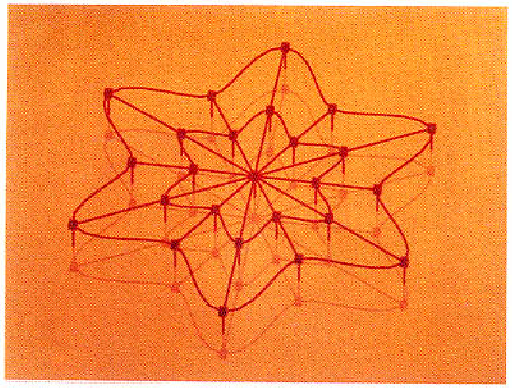

Figure 6a: A lattice positioned on a planar surface

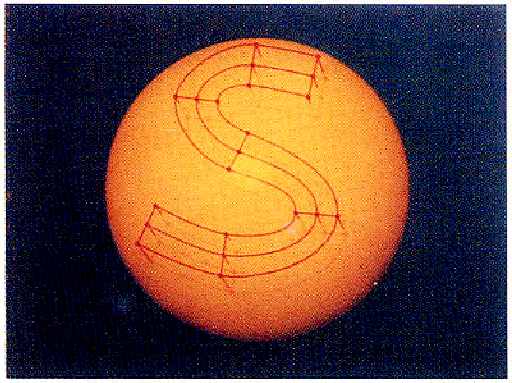

Figure 8a: An "S" lattice positioned on a sphere

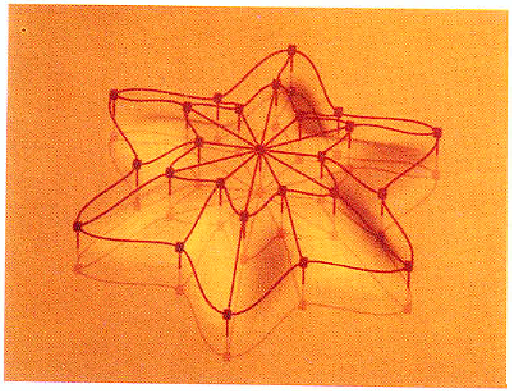

Figure 6b: The deformed lattice and the deformed surface

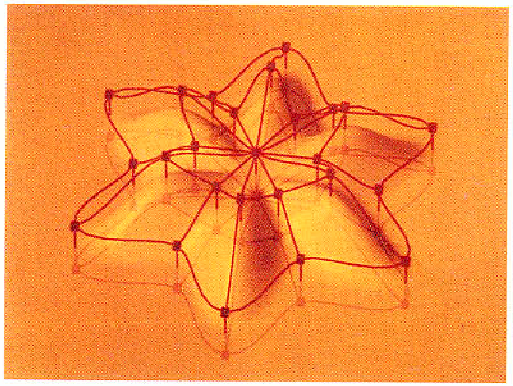

Figure 7b: Another lattice transformation and the deformed surface

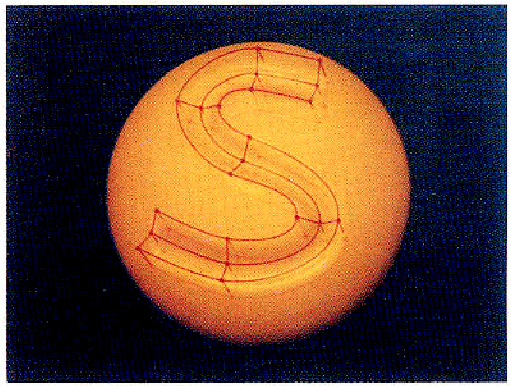

Figure 8b: The deformed lattice and the deformed surface

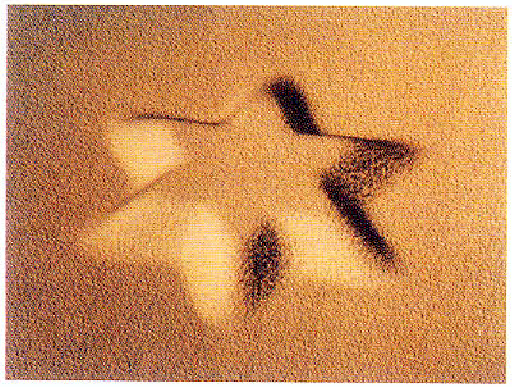

Figure 6c: A sandpie

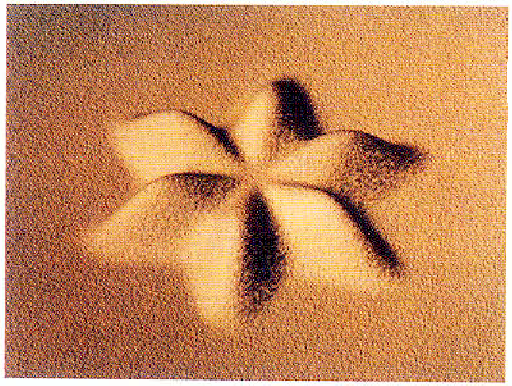

Figure 7c: Another sandpie

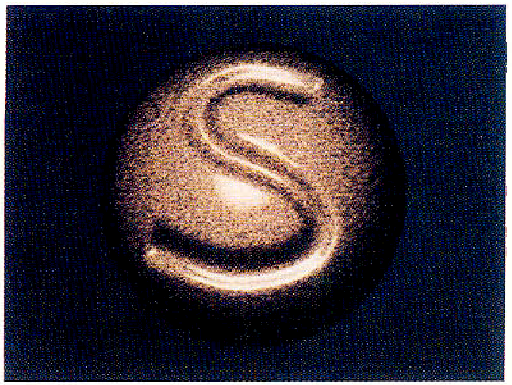

Figure 8c: An "S" sculpted into a granit sphere

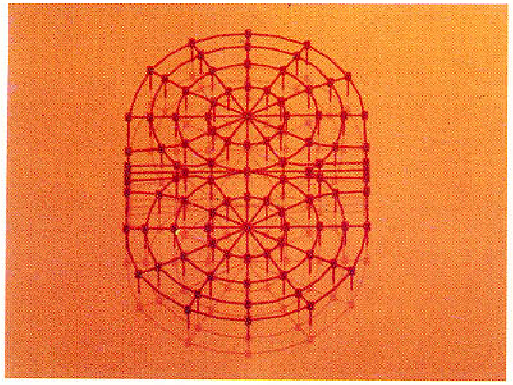

Figure 9a: An "8" lattice positioned on a planar surface

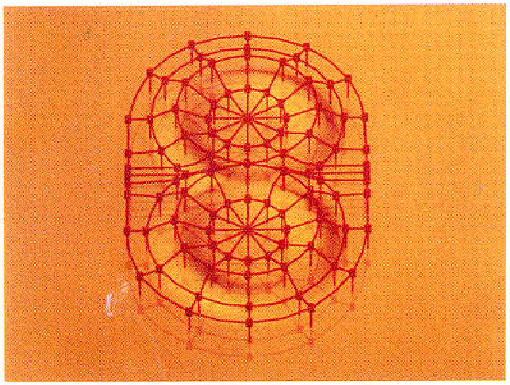

Figure 9b: The deformed lattice and the deformed surface

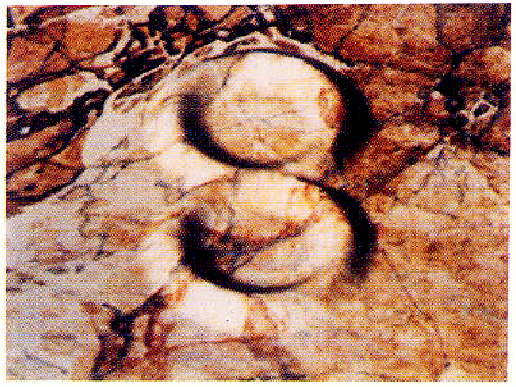

Figure 9c: An "8" sculpted into a piece of marble

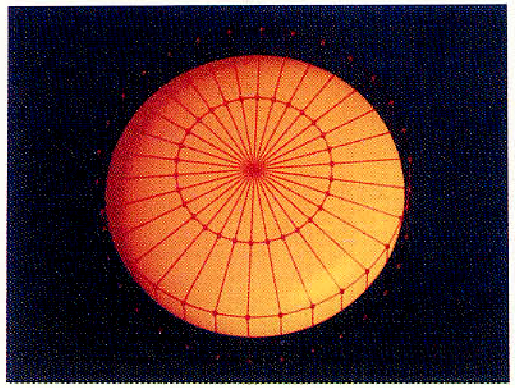

Figure 10a: A cylindrical lattice positioned on a surface of revolution

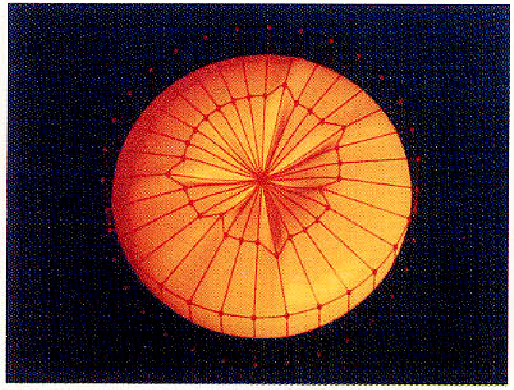

Figure 10b: The deformed lattice and the deformed surface

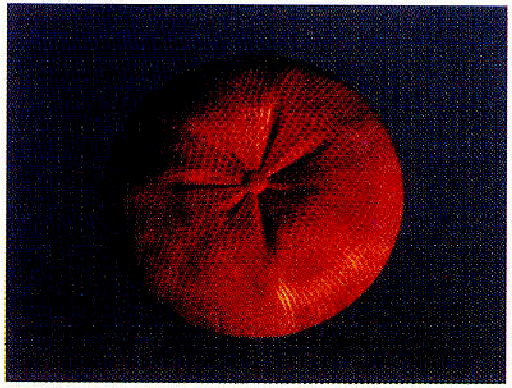

Figure 10c: A leather like cushion

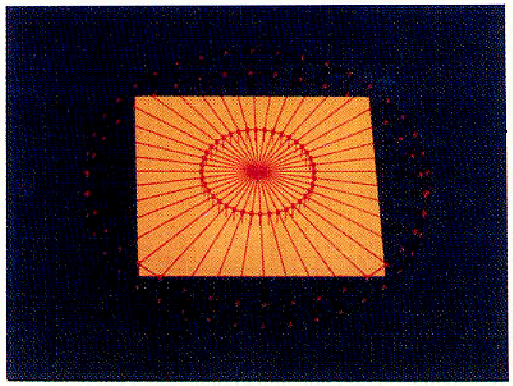

Figure 11a: A cylindrical lattice positioned on a planar surface

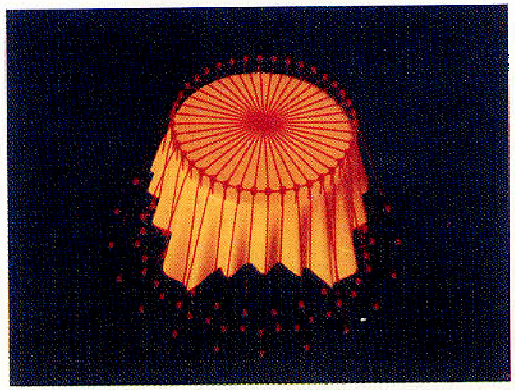

Figure 11b: The deformed lattice and the deformed surface

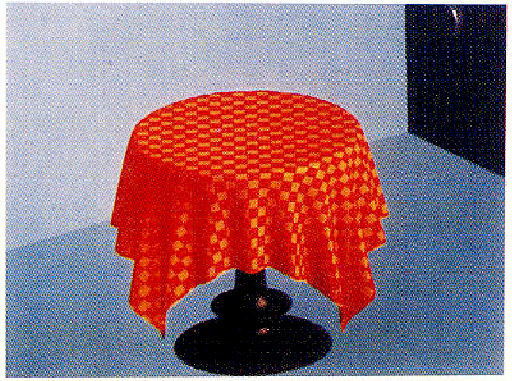

Figure 11c: An oilcloth

ISSN 0249-6399

ISSN 0249 - 6399