Space Deformations and their Application To Shape Modeling

Alexis Angelidis
Dynamic Graphics Project, University of Toronto*

Karan Singh
Dynamic Graphics Project, University of Toronto

Abstract

We present an overview of a set of techniques called space deformations, also known as free-form deformations, warps, skinning or deformers. This family of techniques has various applications in modeling, animation, rendering or simulation, and we focus especially on their application to modeling. Space deformation techniques are mappings of space onto another space, and can therefore be applied conveniently to any embedded geometry. This independence from the underlying geometric representation of deformed shape makes even the simplest and earliest deformation techniques still applicable and popular in current industrial practice.

1 Introduction

Computer Graphics representations of shape are typically defined using a discrete set of parameters that have a visual manifestation, such as the vertices of a mesh, control points of parameteric curves and surfaces or skeletal shapes of implicit surfaces. These visual parameters, traditionally, also serve as handles for the interactive manipulation of the underlying shape. Unfortunately, simply using underlying mathematical handles as a manipulation interface has two major disadvantages. First, there is no connection between the resolution and visual layout of the shape handles and the user desired manipulation. Creating a diagonal surface ridge, for example, by moving control vertices of a rectangular patch is a extremely difficult. Second,deformations defined using the handles of a specific representation cannot be trivially be applied to other shape representations or even different instances of the same shape representation. Space deformations are a family of techniques that address these defficiencies by defining manipulations of space that are directly applicable to any embedded shape representation. We present an overwiew of space deformation in Section 2, followed by a more detailed presentation of various space deformation techniques that are particularly applicable to shape modeling in Section 3. In Section 4, we summarize the techniques presented with a taxonomy of space deformation.

1.1 Principle of Modeling by Space Deformation

With space deformations, a deformed shape is obtained by repeated deformation of the space in which the initial shape is embedded. A convenient formalism can be used for specifying any modeling operation by deformation: the final shape $S(t_n)$ is defined by composition of functions applied to the initial shape $S(t_0)$:

$$S(t_n) = \left\{ \bigop_{i=0}^{n-1} f_{t_i \mapsto t_{i+1}}(\mathbf{p}) \mid \mathbf{p} \in S(t_0) \right\} \quad (1)$$

$$\text{where } \bigop_{i=0}^{n-1} f_{t_i \mapsto t_{i+1}}(\mathbf{p}) = f_{t_{n-1} \mapsto t_n} \circ \cdots \circ f_{0 \mapsto 1}(\mathbf{p})$$

The operator Ω expresses the finite repeated composition of functions. Each function $f_{t_i \mapsto t_{i+1}} \mathbb{R}^3 \mapsto \mathbb{R}^3$ is a deformation that transforms every point $\mathbf{p}$ of space at time t_i into a point of space at time t_{i+1}. Sections 2 will focus on defining functions $f_{t_i \mapsto t_{i+1}}$. Section 3 will address issues related to representing $S(t_i)$.

Normal Deformation: Computing accurate normals to the surface is very important, since normals are used for shading and their level of quality will dramatically affect the visual quality of the shape. Let us recall that in order to compute the new normal after deformation, the previous normal needs to be multiplied by the co-matrix[1] of the Jacobian of the deformation [Barr 1984]. The Jacobian of f at $\mathbf{p}$ is the matrix $J(f, \mathbf{p}) = (\frac{\partial f}{\partial x}(\mathbf{p}), \frac{\partial f}{\partial y}(\mathbf{p}), \frac{\partial f}{\partial z}(\mathbf{p}))$, and the following expression is a convenient way to compute the co-matrix of $J = (\mathbf{j}_x, \mathbf{j}_y, \mathbf{j}_z)$, where the vectors $\mathbf{j}_x$, $\mathbf{j}_y$ and $\mathbf{j}_z$ are column vectors and $\times$ denotes the cross product:

$$J^{\mathrm{C}} = (\mathbf{j}_y \times \mathbf{j}_z, \mathbf{j}_z \times \mathbf{j}_x, \mathbf{j}_x \times \mathbf{j}_y) \quad (2)$$

Note that the multiplication of a vector by J^{C} does not preserve its length. It is therefore necessary to divide a deformed normal by its magnitude.

Generic blending: An practical advantage of space deformation techniques is that they may be treated as blackboxes and blended in a generic manner. Let us consider n deformations f_i and define a partition of unity w_i, possibly scalar fields. The deformations can be applied simultaneously to a point:

$$\mathbf{p} + \sum_{i=1}^{n} w_i (f_i(\mathbf{p}) - \mathbf{p}) \quad (3)$$

The space deformation family of techniques can therefore be seen as a toolbox in which the tools can be combined by handcrafting the weights w_i. In the following, we will however present them independently from each other to underline their strength and weaknesses.

2 Space Deformations Techniques

This section reviews several space deformation techniques, organized in four groups based on the *geometric connectivity* between the control handles: point/parameter controls,

*On leave from the Graphics & Vision Research Lab., U. of Otago, New Zealand.

[1] Matrix of the co-factors.

curve controls, surface controls, lattice-based controls and blendable controls. Although all space deformation may be blended using the above generic method, some of the techniques include convenient blending techniques in their formalism, and the interacting handles are not restricted by any connectivity.

For the sake of clarity, we present existing space deformations aligned with the orthonormal axes $\mathbf{e}_x$, $\mathbf{e}_y$ and $\mathbf{e}_z$ and within the unit cube $[0,1]^3$ whenever possible, because a mere change of coordinates allows the artist to place the deformation anywhere in space. To compare existing deformation techniques from the same point of view, we also use $\mathbf{e}_z$ as the common axis of deformation when applicable, thus we reformulate some of the original formulas. To begin with, note that affine transformations (translation, rotations, and scale) are the simplest examples of space deformations.

2.1 Point/parameters Control

This section contains the subset of space deformations whose control parameters are disconnected elements often without any explicit visual handle.

2.1.1 Global and Local Deformations of Solid Primitives

A. Barr defines space tapering, twisting and bending via matrices whose components are functions of one space coordinate [Barr 1984]. We denote $(x,y,z)^{\mathrm{T}}$ the coordinates of a point. We show in Figures 1, 2, and 3 the effects of these operations, and we give their formula in the form of 4×4 homogeneous matrices to be applied to the coordinates of every point to be deformed.

Tapering operation: The function r is monotonic in an interval, and is constant outside that interval.

$$\begin{pmatrix} r(z) & 0 & 0 & 0 \\ 0 & r(z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 1: Taper deformation of a super-ellipsoid shape.

Twisting operation: The function θ is monotonic in an interval, and is constant outside that interval.

$$\begin{pmatrix} \cos(\theta(z)) & -\sin(\theta(z)) & 0 & 0 \\ \sin(\theta(z)) & \cos(\theta(z)) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 2: Twist deformation of a super-ellipsoid.

Bending operation: This operation bends space along the axis y, in the $0 < z$ half-space. The desired radius of curvature is specified with ρ. The angle corresponding to ρ is $\theta = \hat{z}/\rho$. The value of $\hat{z}$ is the value of z, clamped in the interval $[0, z_{\max}]$.
A. Barr observes that rendering the deformed shape with rays of light is equivalent to rendering the undeformed shape with curves of light. The curves of light are obtained by applying the inverse of the deformation to the rays, assuming the deformation is reversible.

$$\begin{pmatrix} \cos\theta & 0 & \sin\theta & \rho - \rho\cos\theta - \hat{z}\sin\theta \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & \rho\sin\theta - \hat{z}\cos\theta \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

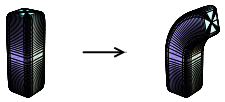


Figure 3: Bend deformation of a super-ellipsoid.

2.1.2 A Generic Implementation of Axial Procedural Deformation Techniques,

C. Blanc extends A. Barr's work to mold, shear and pinch deformations [Blanc 1994]. Her transformations use a function of one or two components. She names this function the *shape function*. Examples and formulas are shown in Figures 4, 5, and 6.

$$\begin{pmatrix} r(z) & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

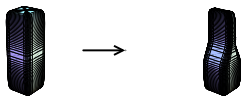


Figure 4: Pinch deformation of a super-ellipsoid.

$$\begin{pmatrix} r(\tan^{-1}(x,y)) & 0 & 0 & 0 \\ 0 & r(\tan^{-1}(x,y)) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

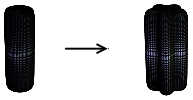


Figure 5: Mold deformation of a super-ellipsoid.

2.1.3 Geometric Deformation by Merging a 3D Object with a Simple Shape

P. Decaudin proposes a technique that allows the artist to model a shape by iteratively adding the volume of simple 3D shapes [Decaudin 1996]. His method is a metaphor of clay sculpture by addition of lumps of definite size and shape. His deformation function is a closed-form, as opposed to a numerical method that would explicitly control the volume [Hirota et al. 1999].

Loosely speaking, this technique inflates space by blowing up a tool in space through a hole. This will compress space around the point in a way that preserves the outside volume. Hence if the tool is inserted inside the shape, the tool's volume will be added to the shape's volume. On the other hand, if the tool is inserted outside the shape, the shape will be deformed but its volume will remain constant. This is illustrated for the 2D case in Figure 9. A restriction on the tool is to be star-convex with respect to its center $\mathbf{c}$. The deformation function is[2] (see Figure 8):

$$f_{3D}(\mathbf{p}) = \mathbf{c} + \sqrt[3]{\rho(\mathbf{p})^3 + r(\mathbf{p})^3}\ \mathbf{n} \qquad (4)$$

- $\rho(\mathbf{p})$ is the magnitude of the vector $\mathbf{u} = \mathbf{p} - \mathbf{c}$.

[2]The 2D case is obtained by replacing 3 with 2.

$$\begin{pmatrix} 1 & 0 & 0 & s(z) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

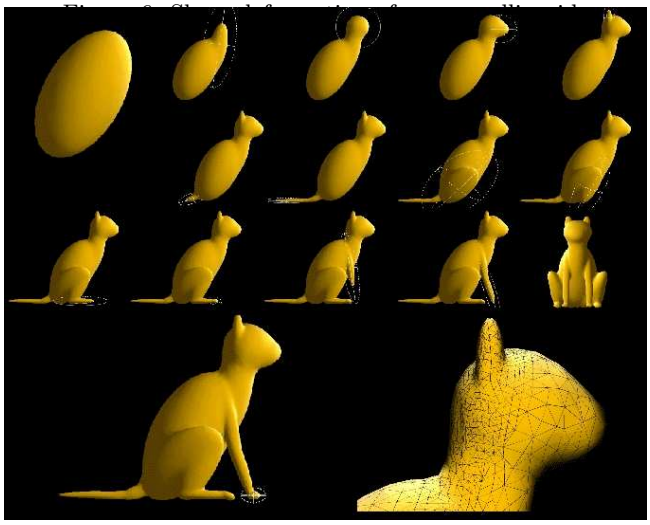


Figure 7: Steps of the modeling of a cat, image by P. Decaudin.

- $r(\mathbf{p})$ is the distance between $\mathbf{c}$ and the intersection of the tool with the half-line $(\mathbf{c}, \mathbf{u})$.
- $\mathbf{n} = \mathbf{u}/\|\mathbf{u}\|$ is the unit vector pointing from $\mathbf{c}$ to $\mathbf{p}$.

If the tool was not a star-convex in $\mathbf{c}$, then $r(\mathbf{p})$ would be ambiguous. The deformation is foldover-free. It is continuous everywhere except at the center $\mathbf{c}$. The effect of the deformation converges quickly to the identity with the increasing distance from $\mathbf{c}$. The deformation can be considered local, and is smooth everywhere except at $\mathbf{c}$. An example in $3D$ is shown in Figure 7. A feature of this space deformation which is rare, is that it has an exact yet simple inverse in the space outside the tool:

$$f_{3D}^{-1}(\mathbf{p}) = \mathbf{c} + \sqrt[3]{\rho(\mathbf{p})^3 - r(\mathbf{p})^3}\ \mathbf{n} \tag{5}$$

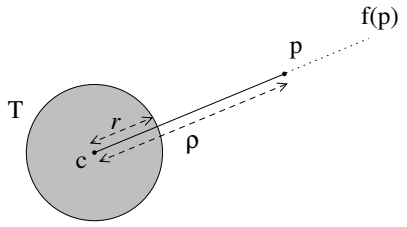


Figure 8: The insertion of a tool at center $\mathbf{c}$ affects the position of point $\mathbf{p}$. See the deformation in Equations (4).

2.1.4 Interactive Space Deformation with Hardware Assisted Rendering

Y. Kurzion and R. Yagel present *ray deflectors* [Kurzion and Yagel 1997]. The authors are interested in rendering the shape by deforming the rays, as opposed to directly deforming the shape. To deform the rays, one needs the inverse of the deformation that the artist intends to apply to the shape. Rather than defining a deformation and then trying to find its inverse, the authors directly define deformations by their inverse. Their tool can translate, rotate and scale space contained in a sphere, locally and smoothly. Moreover they define a discontinuous deformation that allows the artist to cut space, and change a shape's topology. A tool is defined within a ball of radius r around a center $\mathbf{c}$. Let ρ be the distance from the center of the deflector $\mathbf{c}$ and a point $\mathbf{p}$.

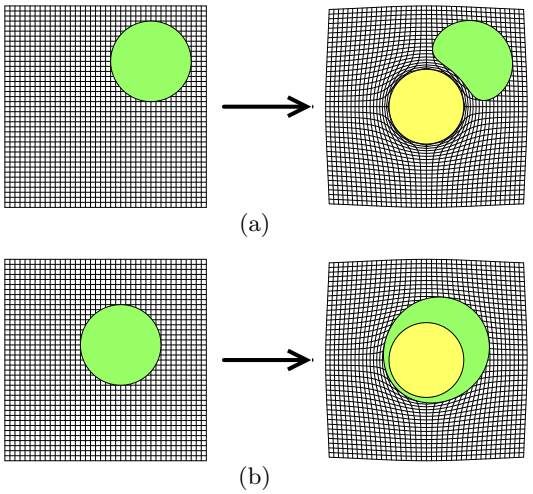


Figure 9: (a) Deformation of a shape (green) by blowing up a tool (yellow) outside the shape. The shape's area is preserved. (b) Deformation of a shape by blowing up a tool inside the shape. The shape's area is increased by that of the tool.

$$\rho = \|\mathbf{p} - \mathbf{c}\| \tag{6}$$

Translate deflector: To define a translate deflector, the artist has to provide a translation vector, $\mathbf{t}$. The effect of the translate deflector will be to transform the center point, $\mathbf{c}$, into $\mathbf{c} + \mathbf{t}$.

$$f_{\mathrm{T}}(\mathbf{p}) = \begin{cases} \mathbf{p} - \mathbf{t}(1 - \frac{\rho^2}{r^2})^2 & \text{if } \rho < r \\ \mathbf{p} & \text{otherwise} \end{cases} \quad \text{where } \theta \in \mathbb{R} \tag{7}$$

Rotate deflector: To define a rotate deflector, the artist has to provide an angle of rotation, θ, and a vector, $\mathbf{n}$, about which the rotation will be done. The reader can find the expression of a rotation matrix, $R_{\theta',\mathbf{n},\mathbf{c}}$, in Appendix **??**. Let us call θ' an angle of rotation that varies in space:

$$\theta' = -\theta(1 - \frac{\rho^2}{r^2})^4$$

$$f_{\mathrm{R}}(\mathbf{p}) = \begin{cases} R_{\theta',\mathbf{n},\mathbf{c}} \cdot \mathbf{p} & \text{if } \rho < r \\ \mathbf{p} & \text{otherwise} \end{cases} \quad \text{where } \|\mathbf{t}\| \in [0, \frac{3\sqrt{3}r}{8}] \tag{8}$$

Scale deflector: To define a scale deflector, the artist has to provide a scale factor s. The scale deflector acts like a

magnifying glass.

$$f_S(\mathbf{p}) = \begin{cases} \mathbf{p} - (\mathbf{p}-\mathbf{c})(1-\frac{\rho^2}{r^2})^4 s & \text{if } \rho < r \\ \mathbf{p} & \text{otherwise} \end{cases} \quad (9)$$
$$\text{where } s \in [-1,1]$$

Discontinuous deflector: To define a discontinuous deflector, the artist has to provide a translation vector, $\mathbf{t}$. The deflector is split into two halves, on each side of a plane going through $\mathbf{c}$ and perpendicular to $\mathbf{t}$. In the half pointed at by $\mathbf{t}$, the discontinuous deflector will transform $\mathbf{c}$, into $\mathbf{c}+\mathbf{t}$, while in the other half, the discontinuous deflector will transform $\mathbf{c}$, into $\mathbf{c}-\mathbf{t}$. The effect will be to cut space. The deformation applied to the rays is:

$$f_D(\mathbf{p}) = \begin{cases} \mathbf{p} - \mathbf{t}(1-\frac{\rho^2}{r^2})^2 & \text{if } \rho<r \text{ and } 0<(\mathbf{p}-\mathbf{c})\cdot\mathbf{t} \\ \mathbf{p} + \mathbf{t}(1-\frac{\rho^2}{r^2})^2 & \text{if } \rho<r \text{ and } (\mathbf{p}-\mathbf{c})\cdot\mathbf{t}<0 \\ \mathbf{p} & \text{otherwise} \end{cases} \quad (10)$$
$$\text{where } \theta \in \mathbb{R}$$

Since this deformation is discontinuous on the disk separating the two halves of the deformation, a ray crossing that disk will be cut in two, as we show in Figure 10(c). Thus a shape intersection algorithm will have to march along the ray from the two sides of the ray, until each curve crosses the separating disk. This deformation assumes that the shape's representation has an inside and outside test. Note that other authors have extended FFD for dealing with discontinuities [Schein and Elber 2004].

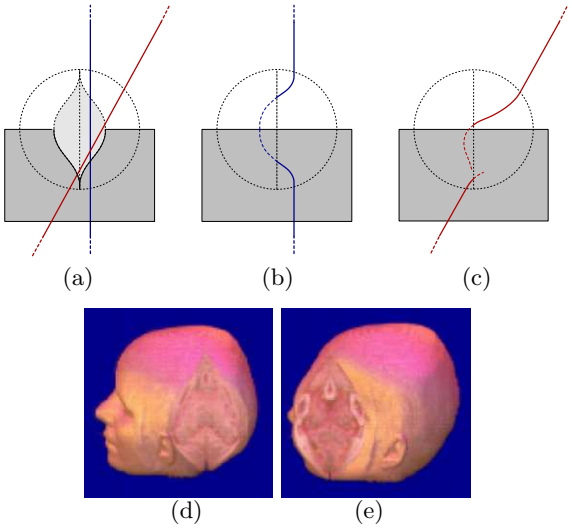


Figure 10: (a) Discontinuous deflector as observed by the artist. Two arbitrary rays are shown. (b) Simple case, where the ray of light crosses only one hemisphere. (c) When the ray of light changes hemisphere, the curve of light is subject to a discontinuity. (d, e) Images by Y. Kurzion and R. Yagel.

2.1.5 Twister

I. Llamas et al propose a method called twister in which a twist transformation of points is weighted with a scalar function [Llamas et al. 2003], i.e. in a similar way to IFFD but with a transformation restrained to a twist. With this restriction, they propose to weight single twists along the trajectory of transformation rather than weighting the displacement. They define a twist by transforming an orthonormal coordinate system $(\mathbf{o},\mathbf{u},\mathbf{v},\mathbf{w})$ into $(\mathbf{o}',\mathbf{u}',\mathbf{v}',\mathbf{w}')$. The axis of the twist is defined by a direction $\mathbf{d}$ and point $\mathbf{a}$ on the axis, while the angle of rotation around the axis is α and the translation factor along the axis is d:

$$\begin{aligned} \mathbf{d} &= \tfrac{\mathbf{g}}{\|\mathbf{g}\|} \\ &\text{where } \mathbf{g} = (\mathbf{u}'-\mathbf{u})\times(\mathbf{v}'-\mathbf{v}) + (\mathbf{v}'-\mathbf{v})\times(\mathbf{w}'-\mathbf{w}) + (\mathbf{w}'-\mathbf{w})\times(\mathbf{u}'-\mathbf{u}) \\ \alpha &= 2\arcsin(\tfrac{\|\mathbf{u}'-\mathbf{u}\|}{2\|\mathbf{d}\times\mathbf{u}\|}) \\ d &= \mathbf{d}\cdot(\mathbf{o}'-\mathbf{o}) \\ \mathbf{a} &= \tfrac{\mathbf{o}+\mathbf{o}'-d\mathbf{d}}{2} + \tfrac{\mathbf{d}\times(\mathbf{o}-\mathbf{o}')}{2\tan(\alpha/2)} \end{aligned} \quad (11)$$

Their procedure for deforming a point $\mathbf{p}$ with a twist parameterized in t is:

1. Bring $\mathbf{p}$ into local coordinates: translate by $-\mathbf{a}$ and then rotate by a rotation that maps $\mathbf{d}$ onto $\mathbf{z}$.
2. Apply the twist in local coordinates: translate by $t\,d$ along $\mathbf{z}$ and rotate by $t\,\alpha$ around $\mathbf{z}$
3. Finally bring $\mathbf{p}$ back into world coordinates: rotate by a rotation that maps $\mathbf{z}$ onto $\mathbf{d}$ and translate by $\mathbf{a}$

To weight the twist, they propose to use a piecewise scalar function:

$$t(\mathbf{p}) = \cos^2(\pi\|\mathbf{p}-\mathbf{o}\|/2r) \quad (12)$$

For operations that require simultaneous twists, they propose simply to add the displacement of the weighted twist. Details for defining a two-point constraint can be found in the paper.

2.2 Curve Control

Curve Control deformations are a subset of space deformations whose control-points are geometrically connected along a curve. The curve may be initially straight or bent. To compare existing deformation techniques from the same point of view, we use $\mathbf{e}_z$ as the common axis of deformation, thus we reformulate some of the original formulas.

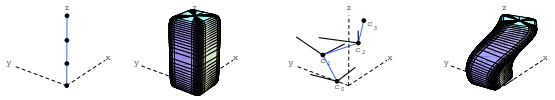
2.2.1 A Generalized de Casteljau Approach to 3d Free-Form Deformation

Y.K. Chang and A.P. Rockwood propose a polynomial deformation that efficiently achieves "Barr"-like deformations and more [Chang and Rockwood 1994], using a Bézier curve with coordinate systems defined along $\mathbf{e}_z$ at the curve's control knots $(z_0, z_1 \ldots, z_n) \in [0,1]^{n+1}$. The initially straight segment $z \in [0,1]$ is deformed by defining coordinate systems $(\mathbf{c}_i, \mathbf{u}_i, \mathbf{v}_i, \mathbf{w}_i)$ along that segment. The shape follows the deformation of the segment, as shown in Figure 11.
To compute the image $\mathbf{q}$ of a point $\mathbf{p}$ of the original shape, the matrix transforming a point to a local coordinate systems is needed:

$$M_i = \begin{pmatrix} u_{i,x} & v_{i,x} & w_{i,x} & c_{i,x} \\ u_{i,y} & v_{i,y} & w_{i,y} & c_{i,y} \\ u_{i,z} & v_{i,z} & w_{i,z} & c_{i,z} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (13)$$

where $\mathbf{w}_i = \mathbf{c}_{i+1} - \mathbf{c}_i$, and $\mathbf{u}_i$, $\mathbf{v}_i$ are the handles.

Using this matrix, the deformation of a point is obtained recursively with the de Casteljau algorithm for evaluating a



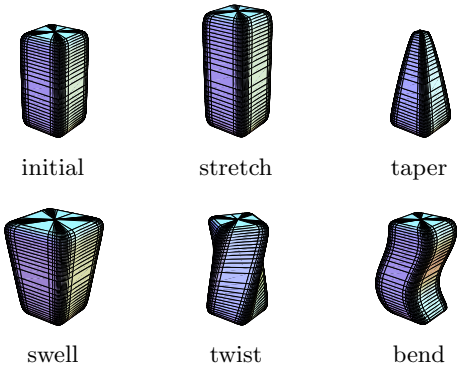
straight axis | initial shape | control points and handles | deformed shape

Figure 11: Example of the deformation of Y.K. Chang and A.P.Rockwood applied to a super-ellipsoid. There is no need to define a pair of handles for the end control point.

Bézier curve:

$$f_i^j(\mathbf{p}) = (1-\mathbf{p}_z) f_i^{j-1}(\mathbf{p}) + \mathbf{p}_z f_{i+1}^{j-1}(\mathbf{p}) \quad (14)$$
$$\text{where } f_i^0(\mathbf{p}) = M_i \cdot \mathbf{p}$$

The original generalized de Casteljau algorithm presented by Y.K. Chang and A.P. Rockwood is a recursion on affine transformations rather than on points. They remark that their recursion simplifies to the classic de Casteljau algorithm when the affine transformations are degenerate, and use the degenerate case in all their examples. As we show in Figure 12, this method is capable of performing "Barr"-like deformations and more.



initial | stretch | taper

swell | twist | bend

Figure 12: Deformation of a super-ellipsoid.

2.2.2 Axial Deformation

A limitation of the above method is the initial rectilinear axis. If the shape is initially bent, the manipulation of an initially straight control axis will not induce a predictable behavior of the shape. F. Lazarus et al. develop an extension of axial-based deformations using an initially curved axis [Lazarus et al. 1994]. Let us define a parametric curve $\mathbf{c}(u)$. A point $\mathbf{p}$ in space is attached to local coordinates along the curve. The origin of this local coordinate system is $\mathbf{c}(u_\mathbf{p})$, the closest point to $\mathbf{p}$ on the curve, and the axes are those of an extended Frenet frame that discards vanishing points [Bloomenthal 1990]. To find the closest point to $\mathbf{p}$ on curves, B. Crespin proposes an efficient algorithm based on subdivision [Crespin 1999]. The axes are computed by propagating along the curve a frame defined at one extremity of the curve. The axes consist of three vectors: a tangent $\mathbf{t}(u)$, a normal $\mathbf{n}(u)$ and a binormal $\mathbf{b}(u)$. The propagated frame is computed as follows:

- the unit tangent at the origin is given by the equation of the curve: $\mathbf{t}(0) = \frac{\mathbf{dc}(0)}{\mathrm{d}u} / \| \frac{\mathbf{dc}(0)}{\mathrm{d}u} \|$.
- the normal and binormal are given by the Frenet frame, or can be any pair of unit vectors such that the initial frame is orthonormal.

To compute the next frame, a rotation matrix is needed. The purpose of this matrix is to minimize torsion along the curve. Numerous constructions of the rotation matrix require a fast formulation:

$$R = \begin{pmatrix} a_{xx}+\theta & a_{xy}+b_z & a_{zx}-b_y \\ a_{xy}-b_z & a_{yy}+\theta & a_{yz}+b_x \\ a_{zx}+b_y & a_{yz}-b_x & a_{zz}+\theta \end{pmatrix} \quad (15)$$

$$\text{where } \begin{array}{ll} (a_x,a_y,a_z)^\top = \frac{\mathbf{t}(u_i)\times\mathbf{t}(u_{i+1})}{\|\mathbf{t}(u_i)\times\mathbf{t}(u_{i+1})\|} & \alpha = 1-\theta \\ \theta = \mathbf{t}(u_i)\cdot\mathbf{t}(u_{i+1}) & \beta = \sqrt{1-\theta^2} \end{array} \quad (16)$$

$$\begin{array}{lll} a_{xx} = \alpha a_x^2 & a_{xy} = \alpha a_x a_y & b_x = \beta a_x \\ a_{yy} = \alpha a_y^2 & a_{yz} = \alpha a_y a_z & b_y = \beta a_y \\ a_{zz} = \alpha a_z^2 & a_{zx} = \alpha a_z a_x & b_z = \beta a_z \end{array} \quad (17)$$

Given a frame at parameter u_i, the next axes of a frame at u_{i+1} are computed as follows:

- the tangent is defined by the equation of the curve: $\mathbf{t}(u_{i+1}) = \frac{\mathbf{dc}(u_{i+1})}{\mathrm{d}u} / \| \frac{\mathbf{dc}(u_{i+1})}{\mathrm{d}u} \|$.
- the normal is given by the rotation of the previous normal: $\mathbf{n}(u_{i+1}) = R \cdot \mathbf{t}(u_i)$.
- the binormal is given by a cross product: $\mathbf{b}(u_{i+1}) = \mathbf{t}(u_i) \times \mathbf{n}(u_i)$.

The choice of the size of the step, $u_{i+1} - u_i$, depends on the trade-off between accuracy and speed. B. Crespin extends the axial deformation to surface deformation [Crespin 1999].

2.2.3 Blendeforming: Ray Traceable Localized Foldover-Free Space Deformation

As explained in the introduction, the motivation for which a space deformation should be foldover-free is its reversibilty, with applications such as undoing operations or raytracing. D. Mason and G. Wyvill introduce *blendeforming* [Mason and Wyvill 2001]. A deformation is specified by moving a point or the control points of a curve along a constrained direction. Space follows the deformation of these control features in a predictable manner.

They define the blendeforming deformation as a bundle of non-intersecting streamlines. The streamlines are parallel, and described by a pair of functions: $b_{\mathbf{x},\mathbf{y}} : \mathbb{R}^2 \mapsto [-d_{\max}, d_{\max}]$ and $b_\mathbf{z} : [0,1] \mapsto [0,1]$. Function $b_{\mathbf{x},\mathbf{y}}$ controls the amount of deformation for each individual z-streamlines, and the choice of function $b_\mathbf{z}$ affects the maximum compression of space along the streamlines. The deformation of point $\mathbf{p} = (x,y,z)^\top$ is

$$\mathbf{p}_{def} = (x, y, z_{def})^\top \quad (18)$$
$$\text{where } z_{def} = z + b_{\mathbf{x},\mathbf{y}}(x,y)\ b_\mathbf{z}(z)$$

It is the definition of $b_\mathbf{z}$ together with a corresponding threshold $d_{\max}$ that prevents foldovers, as shown in Figure 13. The following function is a possible choice for $b_\mathbf{z}(z)$, used in the example:

$$b_\mathbf{z}(z) = \begin{cases} 16z^2(1-z)^2 & \text{if } z \in [0,1] \\ 0 & \text{otherwise} \end{cases} \quad (19)$$
$$\text{with } \ d_{\max} = \frac{3\sqrt{3}}{16} \simeq 0.324$$

Functions permitting larger values for $d_{\max}$ can be found in the original paper. Since $b_{\mathbf{x},\mathbf{y}}$ is independent of z, any function with values in $[-d_{\max}, d_{\max}]$ can be used for it, regardless of the slope. Because the amplitude of the effect of a blendeforming function is bounded by the $d_{\max}$ threshold, it is obvious that modeling an entire shape uniquely with blendeforming functions can be rather tedious. In the original paper, the authors also propose blendeforming bending functions defined in cylindrical coordinates, or using control curves

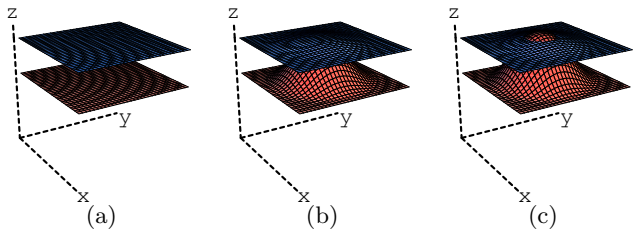


Figure 13: (a) Initial scene: two parallel planes. (b) Blendeforming, with $b_{\mathbf{x},\mathbf{y}}(x, y) = (x^2 - x + y^2 - y - 1/2)^2$. The value of $d_{\max}$ guarantees that the two planes will never intersect. (c) With $d_{\max} < d$, foldover occurs: the lower plane intersects the higher plane.

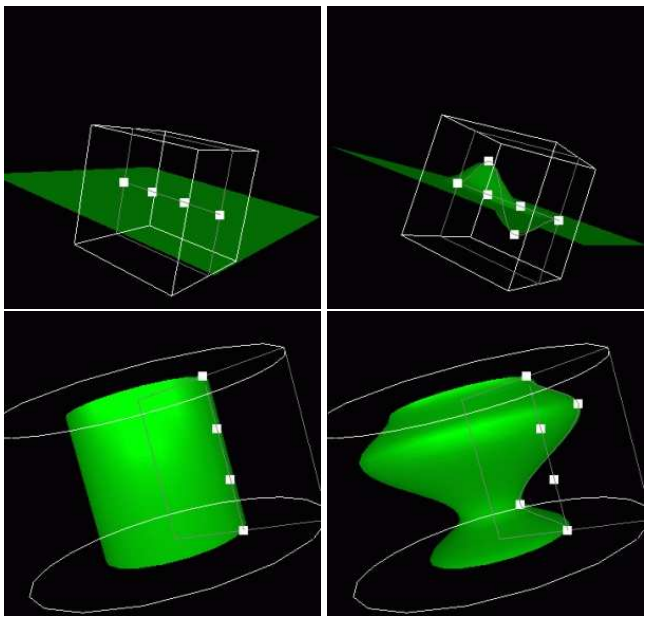


Figure 14: *Top*: a control curve with height control points. *Bottom*: a control curve with radial control points. Image by D. Mason and G. Wyvill.

2.2.4 Interactive Skeleton-Driven Dynamic Deformations

S. Capell et al. propose a framework for skeleton-driven animation of elastically deformable characters [Capell et al. 2002]. This techique defines over a character's skeleton-structure a layer of FFD lattice [Sederberg and Parry 1986] with the control points driven by the dynamic equation of elasticity. Simulation is beyond the scope of this overview, thus we refer to the original paper for detail.

2.3 Surface Control

2.3.1 Surface-Oriented Free-Form Deformation (SOFFD)

K. Singh and E. Kokkevis introduce *Surface-Oriented Free-Form Deformation* (SOFFD) in the context of character animation [Singh and Kokkevis 2000].

To deform a shape S, a SOFFD is defined as a triple $(\mathrm{D}, \mathrm{R}, l)$: the reference surface R, the driver surface D and a scalar value l that controls the influence. The SOFFD process is made of three phases: bind, registration and deformation.

In the binding phase, the surfaces R and D are constructed as low resolution representation of S. The surfaces R and D are initially identical, and their patches define local coordinate systems $M_{\mathrm{R},i}$ $M_{\mathrm{D},i}$ that correspond to each other.

In the registration phase, the local position of $\mathbf{p}_{\mathrm{S}}$ in each patch of R is computed using $\mathbf{q}_{\mathrm{S},i} = M_{\mathrm{R},i}^{-1} \cdot \mathbf{p}_{\mathrm{S},i}$, and the influence u_i of each patch of R at a point $\mathbf{p}_{\mathrm{S}}$ of S are computed using the distance d_i to the i^{th} patch of R :

$$\begin{aligned} u_i &= \frac{w_i}{\sum_j w_j} \\ \text{where } w_i &= \frac{1}{1 + d_i^l} \end{aligned} \tag{20}$$

In the deformation phase, the weighted effect are added to compute the final deformation $\mathbf{p}'_{\mathrm{S}}$ a point $\mathbf{p}_{\mathrm{S}}$ of S:

$$\mathbf{p}'_{\mathrm{S}} = \sum u_i \; M_{\mathrm{D},i} \cdot \mathbf{q}_{\mathrm{S}} \tag{21}$$

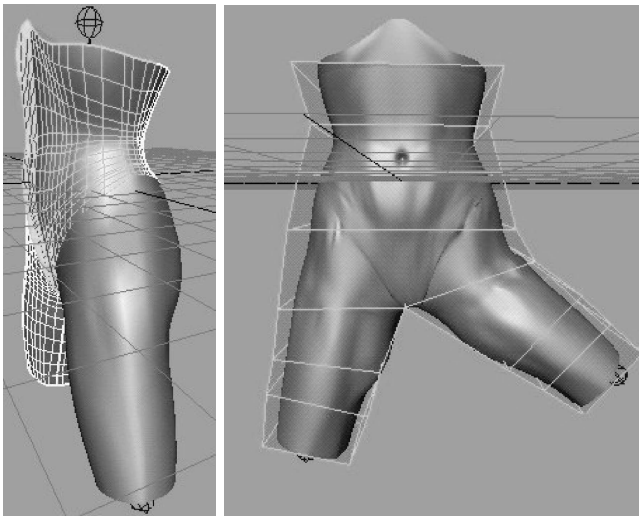


Figure 15: Deformation with SOFFD. Image by K. Singh and E. Kokkevis

2.4 Lattice Control

The limitation of curve or surface controlled space deformation is the arrangement of the controls along a curve or on a surface. Note that this statement is untrue only for wires, which permits the blending of the controls [Singh and Fiume 1998]. Lattice-based space deformations are techniques that allow control points to be connected along the three dimensions of space. There are two ways of understanding

lattice-based deformation, related to the manner in which the artist expresses the deformation. Let us denote the space deformation function by f.

In the first interpretation of lattice-based deformations, the artist provides pairs of points: a source point and a destination point, $(\mathbf{p}_i, \mathbf{q}_i)$. The deformation f will interpolate or approximate the pairs in this way $f(\mathbf{p}_i) = f_{\mathbf{p}}(\mathbf{p}_i) \approx \mathbf{q}_i$. The function $f_{\mathbf{p}}$ is a position field. A position field does not have any physical equivalent to which the artist or scientist can relate, and requires a certain amount of imagination to be visualized.

In the second interpretation of lattice-based deformations, the artist provides a source point and a displacement of that point, $(\mathbf{p}_i, \mathbf{v}_i)$. The deformation f will interpolate or approximate the pairs in this way $f(\mathbf{p}_i) = \mathbf{p}_i + f_{\mathbf{v}}(\mathbf{p}_i) \approx \mathbf{p}_i + \mathbf{v}_i$. The function $f_{\mathbf{v}}$ is a vector field. There is a convenient physical analogy to a vector field. Vector fields are used in fluid mechanics to describe the motion of fluids or to describe fields in electromagnetics [Rutherford 1990; Griffiths 1999]. This analogy is of great help for explaining and creating new space deformations.

While the effect of using either a position field or a vector field is equivalent, the vector field also gives more insight in the process of deforming space: in lattice-based space deformations, the path that brings the source point onto the desired target point is a straight translation using a vector. In this section on lattice-based space deformation, we will therefore consider the construction of a vector field rather than a position field whenever possible.

2.4.1 Free-Form Deformation of Solid Geometric Models

The effect of Free-Form Deformation (FFD) on a shape is to embed this shape in a piece of flexible plastic. The shape deforms along with the plastic that surrounds it [Sederberg and Parry 1986].

The idea behind FFD is to interpolate or approximate vectors defined in a 3d regular lattice. The vectors are then used to translate space. In their original paper, T. Sederberg and S. Parry propose to use the trivariate Bernstein polynomial as a smoothing filter. Let us denote by $\mathbf{v}_{ijk}$ the $(l+1)\times(m+1)\times(n+1)$ *control vectors* defined by the artist. The smoothed vector field is a mapping $\mathbf{p} \in [0,1]^3 \mapsto \mathbb{R}^3$.

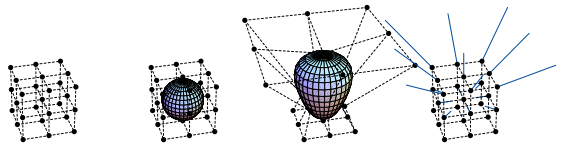
$$\mathbf{v}(\mathbf{p}) = \sum_{i=0}^{l} \binom{i}{l} (1-x)^{l-i} x^i \left(\sum_{j=0}^{m} \binom{j}{m} (1-y)^{m-j} y^j \left(\sum_{k=0}^{n} \binom{k}{n} (1-z)^{n-k} z^k \right) \right) \mathbf{v}_{ijk} \tag{22}$$

Then the deformation of a point is a translation of that point

$$\mathbf{p}_{def} = \mathbf{p} + \mathbf{v}(\mathbf{p}) \tag{23}$$

In order for the deformation to be continuous across the faces of the FFD cube, the boundary vectors should be set to zero. A drawback of using the Bernstein polynomial is that a control vector $\mathbf{v}_{ijk}$ has a non-local effect on the deformation. Hence updating the modification of a control vector requires updating the entire portions of shape within the lattice. For this reason, J. Griessmair and W. Purgathofer propose to use B-Splines [Griessmair and Purgathofer 1989].

In commercial software, the popular way to let the artist specify the control vectors is to let him move the control points of the lattice, as shown in Figure 16(c). A drawback often cited about this interface is the visual self occlusion of the control points. This problem increases with the increase in resolution of the lattice. Another drawback is that the manipulation of of lattice of control points requires a strong sense of spatial perception from the artist. Clearly, practical FFD manipulation through control-points can only be done with reasonably small lattices.



(a) (b) (c) (d)

Figure 16: FFD deformation. (a) Lattice of size 3^3. (b) Initial shape. (c) The popular interaction with an FFD lattice consists of displacing the control points. (d) The discrete vectors.

2.4.2 Extended Free-Form Deformation (EFFD)

Due to the practical limit of the size of the FFD-lattice, the major restriction of an FFD is strongly related to the arrangement of control-points in parallelepipeds. The parallelepipeds are also called *cells*. To provide the artist with more control, S. Coquillart proposes a technique with non-parallelepipedic and arbitrarily connected *cells*. The technique is called Extended Free-Form Deformation (EFFD) [Coquillart 1990].

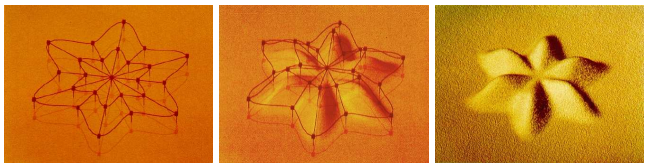
To model with EFFD, the artist first builds a lattice by placing the extended cells anywhere in space, and then manipulates the cells to deform the shape. An extended cell is a small FFD of size 4^4. The transformation from the cell's *local coordinates* $\mathbf{s} = (u, v, w)^\top$ to world coordinates is:

$$\mathbf{p}(\mathbf{s}) = \sum_{i=0}^{3} \binom{i}{3} (1-u)^{3-i} u^i \left(\sum_{j=0}^{3} \binom{j}{3} (1-v)^{3-j} v^j \left(\sum_{k=0}^{3} \binom{k}{3} (1-w)^{3-k} w^k \right) \right) \mathbf{p}_{ijk} \tag{24}$$

The eight corners $\mathbf{p}_{ijk\in\{0,3\}^3}$ of a cell are freely defined by the artist. The position of the remaining $4^4 - 8$ are constrained by the connection between cells, so that continuity is maintained across boundaries. This is done when the artist connects the cells. Because the lattice is initially deformed, finding a point's coordinates $\mathbf{s}$ in a cell is not straightforward. The local coordinates of a point $\mathbf{p}$ in a cell are found by solving Equation (24) in $\mathbf{s}$ using a numerical iteration. This can be unstable in some cases, although the authors report they did not encounter such cases in practice. Once $\mathbf{s}$ is found, the translation to apply to $\mathbf{p}$ is found by substituting in Equation (24) the control points $\mathbf{p}_{ijk}$ with the control vectors $\mathbf{v}_{ijk}$. Note that specifying the control points, the cells and the control vectors is rather tedious, and results shown in the paper consist essentially of imprints. An example is shown in Figure 17.

2.4.3 Preventing Self-Intersection under Free-Form Deformation

In FFD, EFFD and DMFFD, if the magnitude of a control-vector is too high, the deformation may produce a self-intersection of the shape's surface (see a self-intersection in Figure 13). Once the shape's surface self-intersects, there is no space deformation that can remove the self-intersection. The appearance of this surface incoherency is the result of



(a) (b) (b)

Figure 17: EFFD deformation, images by S. Coquillart. (a) Control lattice. (b) Deformed lattice. (c) Result: a sand-pie.

a space foldover: the deformation function is a surjection of $\mathbb{R}^3$ onto $\mathbb{R}^3$, not a bijection. J. Gain and N. Dodgson present foldover detection tests for DMFFD deformations that are based on uniform B-Splines [Gain and Dodgson 2001]. They argue that a necessary and sufficient test is too time consuming, and present an alternative sufficient test. Let us define q_{ijk}, the deformed control points of the lattice. If the determinants of all the following 3×3 matrices are all positive, there is no foldover.

$$\phi_{ijk} = s \det\left(\mathrm{q}_{i\pm 1jk} - \mathrm{q}_{ijk} \ , \ \mathrm{q}_{ij\pm 1k} - \mathrm{q}_{ijk} \ , \ \mathrm{q}_{ijk\pm 1} - \mathrm{q}_{ijk}\right)$$

where the sign s is 1 if $(i\pm1, j\pm1, k\pm 1)$ are clockwise, else –1. The idea underlying the test is that the determinant of three column vectors is the volume of the parallelepiped defined by these vectors. A negative volume detects a possible singularity in the deformation. A technique for efficiently testing several determinants at once can be found in the original paper.

This test can then be used to repair the DMFFD. Let us define $(\mathbf{p}_i, \mathbf{v}_i)$, the pairs of points and vectors defining the DMFFD. If a foldover is detected, the DMFFD is recursively split into two parts: $(\mathbf{p}_i, \mathbf{v}_i/2)$ and $(\mathbf{p}_i + \mathbf{v}_i/2, \mathbf{v}_i/2)$. The procedure eventually converges, and the series of DMFFDs obtained are foldover-free and can be applied safely to the shape.

2.4.4 Free-form Deformations with Lattices of Arbitrary Topology (SFFD)

R.A. MacCracken and K.I. Joy have established a method that allows the user to define lattices of arbitrary shape and topology [MacCracken and Joy 1996]. The method is more stable than EFFD since it does not rely on a numerical iteration technique.

Their method is based on subdivision lattices. We will refer to it as SFFD, for subdivision FFD. The user defines a control lattice, L: a set of vertices, edges, faces and cells. A set of refinement rules are repeatedly applied to L, creating a sequence of increasingly finer lattices $\{L_1, L_2, \ldots L_l\}$. The union of cells define the deformable space. After the first subdivision, all cells can be classified into cells of different type: type-n cells, $n \geq 3$. See [MacCracken and Joy 1996] for the rules.

Although there is no available trivariate parameterization of the subdivision lattice, the correspondence between world coordinates and lattice coordinates is possible thanks to the subdivision procedure. The location of a vertex embedded in the deformable space is found by identifying which cell contains it. Then, for a type-3 cell, trilinear parameterization is used. For a type-n cell, the cell is partitioned in $4n$ tetrahedra, in which the vertex takes a trilinear parameterization. Each point is tagged with its position in its cell.

Once a point's location is found in the lattice, finding the point's new location is straightforward. When the artist displaces the control points, the point's new coordinates are traced through the subdivision of the deformed lattice.

2.4.5 Scalar-Field Guided Adaptive Shape Deformation and Animation (SFD)

J. Hua and H. Qin create a technique called SFD [Hua and Qin 2004]. They define a deformation by attaching space to the level-sets of an animated scalar field. The artist is offered three different techniques for animating a scalar field. Since there are many ways of attaching a point to a level-set of a scalar field, the authors choose the way that keeps the shape as rigid as possible.

They define $\phi(t, \mathbf{p}(t))$, the scalar field which is animated in time, t. Since a moving point, $\mathbf{p}(t)$, is attached to a level-set of the scalar field, the value of ϕ at $\mathbf{p}$ is constant in time:

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = 0 \tag{25}$$

The square of Equation (25) gives a constraint:

$$(\frac{\mathrm{d}\phi}{\mathrm{d}t})^2 = 0 \tag{26}$$

There are several ways of attaching a point to a level set while the scalar field is moving. The simplest way would be to make a point follow the shortest path, found when the magnitude of the point's speed, $\|\mathbf{v}(t)\|$, is minimized. Another possibility, chosen by the authors, is to minimize the variation of velocity, so that the deformation is as rigid as possible. Instead of using the divergence of the speed to measure rigidity, they use an estimate by averaging the variation of speed between that point's speed, $\mathbf{v}$, and its neighbors' speed, $\mathbf{v}_k$:

$$(\nabla \cdot \mathbf{v})^2 \approx \frac{1}{k} \sum_k (\mathbf{v} - \mathbf{v}_k)^2 \tag{27}$$

Since this is a constrained optimization problem [Weisstein], there exists a Lagrange multiplier λ such that:

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{v}}(\frac{\mathrm{d}}{\mathrm{d}t}\phi(t, \mathbf{p}(t)))^2 + \lambda \frac{\mathrm{d}}{\mathrm{d}\mathbf{v}}(\nabla \cdot \mathbf{v})^2 = \vec{0} \tag{28}$$

According to the authors, λ is an experimental constant, used to balance the flow constraint and speed variation constraint. Its value ranges between 0.05 and 0.25. We rearrange this equation and expand the derivative of ϕ with the chain rule:

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{v}}\left((\nabla\phi \cdot \mathbf{v} + \frac{\partial\phi}{\partial t})^2 + \lambda(\nabla \cdot \mathbf{v})^2\right) = \vec{0} \tag{29}$$

Let us define $\hat{\mathbf{v}}$, the average of the velocity of all the adjacent neighbors connected with edges to point $\mathbf{p}$. If we substitute $(\nabla\cdot\mathbf{v})^2$ for its approximate given by Equation (27), and then apply the derivative with respect to $\mathbf{v}$, we obtain:

$$(\nabla\phi \cdot \mathbf{v} + \frac{\partial\phi}{\partial t})\nabla\phi + \lambda(\mathbf{v} - \hat{\mathbf{v}}) = \vec{0} \tag{30}$$

By solving the system of Equation (30), the updated position is:

$$\mathbf{v} = \hat{\mathbf{v}} - \frac{\hat{\mathbf{v}} \cdot \nabla\phi + \frac{\partial\phi}{\partial t}}{\lambda + (\nabla\phi)^2}\nabla\phi \tag{31}$$

The algorithm deforms a set of vertices in n substeps. If n is set to one, the deformation takes one step:

for i = 1 to n **do**
  for all $\mathbf{p}_k$ in the list of vertices to update **do**
    Update the scalar field $\phi(t + \Delta t, \mathbf{p}_k)$.
    Deduce $\frac{\partial \phi}{\partial t} = \frac{\phi(t+\Delta t, \mathbf{p}_k) - \phi(t, \mathbf{p}_k)}{\Delta t}$
    Calculate $\nabla\phi$, possibly with finite differences.
    Compute $\hat{\mathbf{v}}$ according to neighbors' velocities.
    Deduce $\mathbf{v}$ according to Equation (31).
    Update vertex positions with $\mathbf{p}_k(t + \Delta t) = \mathbf{p}_k(t) + \mathbf{v}\frac{\Delta t}{n}$
    Improve surface representation using a mesh refinement and simplification strategy.
    if $\phi(t + \Delta t, \mathbf{p}_k(t + \Delta t)) \approx \phi(t, \mathbf{p}_k(t))$ **then**
      remove $\mathbf{p}_k$ from the list of vertices to update.
    end if
  end for
end for

In the first step, since all the speeds are zero, we suggest that they could be initialized with:

$$\mathbf{v} = -\frac{\frac{\partial \phi}{\partial t}}{\lambda + (\nabla\phi)^2}\nabla\phi \tag{32}$$

(a) (b) (c)

Figure 18: SFD applied to a digitized model of a dinosaur, images by J. Hua and H. Qin.

Note that this technique requires an explicit surface in order to compute the divergence of the speed. The advantage of a large set of possible SFD shape operations (as large as the set of possible animated scalar fields) is at the cost of making the artist's task rather tedious: specifying the animated field does not permit quick and repeated operations on the shape, necessary for shape modeling.

2.5 Blendable

All deformations can be combined together by combining the deformations with a partition of unity defined in space. Some deformation technique however include geometric blending more strongly in their formalism, and define blending methods that provides a variety of user control and total freedom in placing the control handles.

2.5.1 Direct Manipulation of Free-Form Deformations (DMFFD)

The manipulation of individual control points makes FFD and EFFD tedious methods to use. Two groups of researchers, P. Borrel and D. Bechmann, and W.M. Hsu et al. propose a similar way of doing direct manipulation of FFD control points (DMFFD) [Borrel and Bechmann 1991; Hsu et al. 1992]. The artist specifies translations $\mathbf{v}_i$ at points $\mathbf{p}_i$ in the form $(\mathbf{p}_i, \mathbf{v}_i)$. The DMFFD algorithm finds control vectors that satisfy, if possible, the artist's desire. Let us define a single input vector $\mathbf{v}$ at point $\mathbf{p}$. The FFD Equation (22) must satisfy

$$\mathbf{v} = \mathrm{B}(\mathbf{p})(\mathbf{v}_{ijk}) \tag{33}$$

Let $\nu = (3(l+1)(m+1)(n+1))$. The matrix B is the $3 \times \nu$ matrix of the Bernstein coefficients, which are functions of point $\mathbf{p}$. Note that their method is independent of the chosen filter: instead of the Bernstein polynomials, W.M. Hsu et al. use B-Splines and remark that Bernstein polynomials can be used. P. Borrel and D. Bechmann on the other hand found that using simple polynomials works just as well as B-Splines. The size of the vector of control vectors $(\mathbf{v}_{ijk})$ is $3(l+1)(m+1)(n+1)$. When the artist specifies μ pairs $(\mathbf{p}_i, \mathbf{v}_i)$, the FFD Equation (22) must satisfy a larger set of equations:

$$\begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_\mu \end{pmatrix} = \mathbf{B} \cdot \begin{pmatrix} \mathbf{v}_{ijk} \\ \vdots \\ \mathbf{v}_{ijk} \end{pmatrix} \quad \text{where } \mathbf{B} = \begin{pmatrix} \mathrm{B}(\mathbf{p}_1) \\ \vdots \\ \mathrm{B}(\mathbf{p}_\mu) \end{pmatrix} \tag{34}$$

This set of equations can either be overdetermined or under determined. In either case, the matrix $\mathbf{B}$ cannot be inverted in order to find the $\mathbf{v}_{ijk}$. The authors use the Moore-Penrose pseudo-inverse, $\mathbf{B}^+$. If the inverse of $\mathbf{B}^\top \cdot \mathbf{B}$ exists, then

$$\mathbf{B}^+ = (\mathbf{B}^\top \cdot \mathbf{B})^{-1} \cdot \mathbf{B}^\top \tag{35}$$

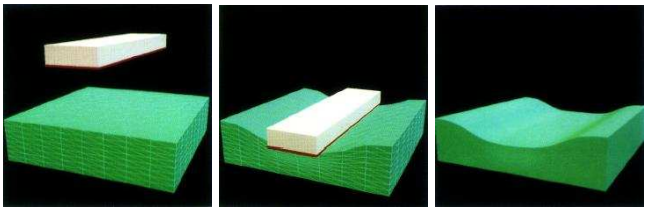
It is however preferable to compute the Moore-Penrose pseudo-inverse using singular value decomposition (SVD). The $\mu \times \nu$ matrix $\mathbf{B}$ can be written

$$\mathbf{B} = U \cdot D \cdot V^\top \tag{36}$$

where U is an $\mu \times \mu$ orthogonal matrix, V is an $\nu \times \nu$ orthogonal matrix and D is an $\mu \times \nu$ diagonal matrix with real, non-negative elements in descending order.

$$\mathbf{B}^+ = V \cdot D^{-1} \cdot U^\top \tag{37}$$

Here, the diagonal terms of D^{-1} are simply the inverse of the diagonal terms of D.



(a) (b) (b)

Figure 19: DMFFD deformation, images by W.M. Hsu et al. (a) Initial scene. (b) The deformation is created according to the displacement of several vertices of the green object. (c) Result. The authors do not describe how the vertices on the green object are selected.

The size of the basis, or, equivalently the number of control points, has a direct effect on the locality of the deformation around the selected point. In their approach, P. Borrel and D. Bechmann pursue the reasoning even further, and define a technique suitable for n-dimensional objects [Borrel and Bechmann 1991]. In the context of shape animation, i.e. in $\mathbb{R}^4$ with time as the fourth dimension, the Bernstein, B-Splines or simple polynomials are inappropriate. They propose to use a basis that does not change the initial time, t_0, and final time, t_f, of an object:

$$\mathrm{B}_t(\mathbf{p}, t) = \begin{pmatrix} (t - t_0)(t - t_f) \\ (t - t_0)(t - t_f)t \\ (t - t_0)(t - t_f)t^2 \\ \vdots \end{pmatrix} \tag{38}$$

2.5.2 Simple Constrained Deformations for Geometric Modeling and Interactive Design (scodef)

In simple constrained deformations (scodef), P. Borrel and A. Rappoport propose to use DMFFD with radial basis functions (RBF) [Borrel and Rappoport 1994]. The artist defines constraint triplets $(\mathbf{p}_i, \mathbf{v}_i, r_i)$: a point, a vector that defines the desired image of the point, and a radius of influence. Let $\phi_i(\mathbf{p})$ denote the scalar function $\phi(\frac{\|\mathbf{p}-\mathbf{p}_i\|}{r_i})$ for short. The motivation of using RBF is to keep the deformation local, in the union of spheres of radius r_i around the points $\mathbf{p}_i$. A naive vector field would be:

$$\mathbf{v}(\mathbf{p}) = \sum_{i=1}^{n} \mathbf{v}_i \phi_i(\mathbf{p}) \tag{39}$$

Unless the points $\mathbf{p}_i$ are far apart enough, Equation (39) will not necessarily satisfy the artist's input $\mathbf{v}(\mathbf{p}_i) = \mathbf{v}_i$ if the functions ϕ_i overlap. However, this can be made possible by substituting the vectors $\mathbf{v}_i$ with suitable vectors $\mathbf{w}_i$.

$$\mathbf{v}(\mathbf{p}) = \sum_{i=1}^{n} \mathbf{w}_i \phi_i(p) \tag{40}$$

These vectors $\mathbf{w}_i$ can be found by solving a set of $3n$ equations:

$$\mathbf{v}_i = (\mathbf{w}_1 \dots \mathbf{w}_n) \cdot \begin{pmatrix} \phi_1(\mathbf{p}_i) \\ \vdots \\ \phi_n(\mathbf{p}_i) \end{pmatrix} \text{ where } i \in [1, n] \tag{41}$$

Let us take the transpose, and arrange the n equations in rows. The following equation is the equivalent of Equation (34), but with radial basis functions:

$$\begin{pmatrix} \mathbf{v}_1^\top \\ \vdots \\ \mathbf{v}_n^\top \end{pmatrix} = \begin{pmatrix} \phi_1(\mathbf{p}_1) & \dots & \phi_n(\mathbf{p}_1) \\ \vdots & & \\ \phi_1(\mathbf{p}_n) & \dots & \phi_n(\mathbf{p}_n) \end{pmatrix} \cdot \begin{pmatrix} \mathbf{w}_1^\top \\ \vdots \\ \mathbf{w}_n^\top \end{pmatrix} \quad \text{where } i \in [1, n] \tag{42}$$

Let $\boldsymbol{\Phi}$ be the $n \times n$ square matrix of Equation (42). This matrix takes the role of $\mathbf{B}$ in Equation (34). Since $\boldsymbol{\Phi}$ can be singular, the authors also use its pseudo-inverse $\boldsymbol{\Phi}^+$ to find the vectors $\mathbf{w}_i$.

2.5.3 Dirichlet Free-Form Deformation (DFFD)

With DFFD, L. Moccozet and N. Magnenat-Thalmann propose a technique that builds the cells of a lattice automatically [Moccozet and Magnenat-Thalmann 1997], relieving the artist from a tedious task. The lattice cells are the cells of a Voronoï diagram of the control points, shown in Figure 20. The location of a point within a cell is neatly captured by the Sibson coordinates. The naive deformation of a point $\mathbf{p}$ is given by interpolating vectors defined at the control points with the Sibson coordinate.

$$\mathbf{p} \mathrel{+}= \sum_{i=1}^{n} \frac{a_i}{a} \mathbf{v}_i \tag{43}$$

Where a_i is the volume of cell i stolen by $\mathbf{p}$, and a is the volume of the cell of $\mathbf{p}$. This interpolation is only C^0. They use a method developed by G. Farin [Farin 1990] to define a continuous parameterization on top of the Sibson coordinates. The interpolation is made of four steps:

- build the local control net
- build *Bézier abscissa*
- define *Bézier ordinates* such that the interpolant is C^1
- evaluate the multivariate Bernstein polynomial using Sibson coordinates.

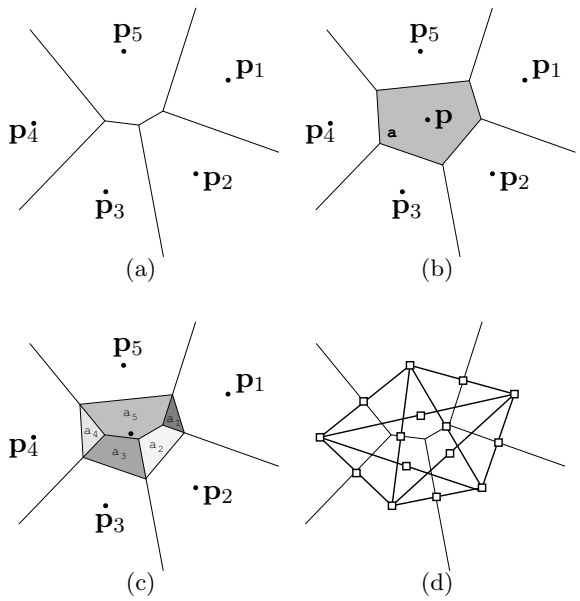


Figure 20: 2D illustration of the Sibson coordinates (a) Voronoï cells of the control points. (b) Voronoï diagram is updated after the insertion of point $\mathbf{p}$. (c) The areas stolen by the point $\mathbf{p}$ from its natural neighbors give the Sibson coordinates a_i/a. (d) Local control net, with *Bézier abscissa*.

2.5.4 Implicit Free-Form Deformations (IFFD)

B. Crespin introduces Implicit Free-Form Deformations (IFFD) [Crespin 1999]. Note that though it is called implicit, the deformation is explicit. IFFD is rather a technique inspired by implicit surfaces, a vast branch of computer graphics whose presentation is beyond the scope of this document [Bajaj et al. 1997]. The field $\phi \in [0, 1]$ generated by a skeleton modulates a transformation, M, of points. The deformation of point $\mathbf{p}$ with a single function is:

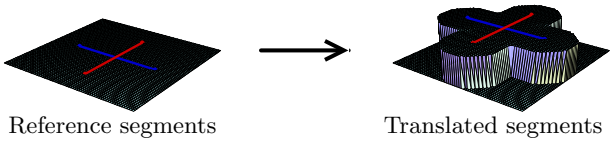
$$f(\mathbf{p}) = \mathbf{p} + \phi(\mathbf{p})(M \cdot \mathbf{p} - \mathbf{p}) \tag{44}$$

He proposes two ways to combine many deformations simultaneously. Let use denote $\mathbf{p}_i$ the transformation of $\mathbf{p}$ with deformation f_i. The first blending is shown in Figure 21. For M, we have used a translation matrix. The second blending attempts to solve the continuity issue, but requires the definition of supplementary profile functions, γ_i. The purpose of the index i is to assign individual profiles to skeletons.

In order to produce Figure 22, the following γ_i function was used:

$$\gamma_i(\mathbf{p}) = \begin{cases} 1-(1-\sigma^2)^2 & \text{if } \sigma \in [0,1], \text{ where } \sigma = \sum_{i=1}^{n} \phi_i(\mathbf{p}) \\ 1 & \text{otherwise} \end{cases} \tag{45}$$

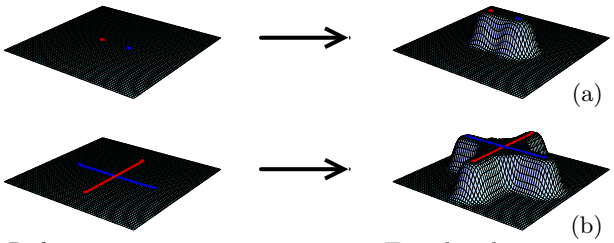
$$\mathbf{p}_{def} = \mathbf{p} + \frac{\sum_{i=1}^{n}(\mathbf{p}_i - \mathbf{p})\phi_i(\mathbf{p})}{\sum_{i=1}^{n}\phi_i(\mathbf{p})}$$



Reference segments Translated segments

Figure 21: Blending weights based on summed displacement magnitudes. The deformation is only defined where the amounts ϕ are not zero, and is discontinuous at the interface $\sum_i \phi_i = 0$. This blending is useful when the deformed shape is entirely contained within the field.

$$\mathbf{p}_{def} = \mathbf{p} + \frac{\sum_{i=1}^{n}(\mathbf{p}_i - \mathbf{p})\phi_i(\mathbf{p})\gamma_i(\mathbf{p})}{\sum_{i=1}^{n}\phi_i(\mathbf{p})}$$



Reference segments Translated segments

Figure 22: Blending weights based on displacement magnitudes and profile functions. For control points, the technique works well. For segments, there is a discontinuity near their intersection.

2.5.5 Wires: a Geometric Deformation Technique

K. Singh and E. Fiume introduce *wires*, a technique which can easily achieve a rich set of deformations with curves as control features [Singh and Fiume 1998]. Their technique is inspired by armatures used by sculptors.

A wire is defined by a quadruple $(\mathrm{R}, \mathrm{W}, s, r)$: the reference curve R, the wire curve W, a scaling factor s that controls bulging around the curve, and a radius of influence r. The set of reference curves describes the armature embedded in the initial shape, while the set of wire curves defines the new pose of the armature.

On a curve C, let $\mathbf{p}_C$ denote the parameter value for which $\mathrm{C}(\mathbf{p}_C)$ is the closest point to $\mathbf{p}$. Let us also denote $\mathrm{C}'(\mathbf{p}_C)$ the tangent vector at that parameter value.

The reference curve, R, generates a scalar field $\mathrm{F} : \mathbb{R}^3 \mapsto [0, 1]$. The function F which decreases with the distance to R, is equal to 1 along the curve and equals 0 outside a neighborhood of radius r. The algorithm to compute the image $\mathbf{q}$ of a point $\mathbf{p}$ influenced by a single deformation consists of three steps, illustrated in Figure 23:

- Scaling step. The scaling factor is modulated with F. The image of a point $\mathbf{p}$ after scaling is: $\mathbf{p}_s = \mathrm{R}(\mathbf{p}_R) + (\mathbf{p} - \mathrm{R}(\mathbf{p}_R))(1 + s\mathrm{F}(\mathbf{p}))$, where $\mathbf{p}_R$ denotes the parameter value for which $\mathrm{R}(\mathbf{p}_R)$ is the closest to $\mathbf{p}$.
- Rotation step. Let θ be the angle between the tangents $\mathrm{R}'(\mathbf{p}_R)$ and $\mathrm{W}'(\mathbf{p}_R)$. The point p_s is rotated around axis $\mathrm{R}'(\mathbf{p}_R) \times \mathrm{W}'(\mathbf{p}_R)$ about center $\mathrm{R}(\mathbf{p}_R)$ by the modulated angle $\theta\ \mathrm{F}(\mathbf{p})$. This results in point $\mathbf{p}_r$
- Translation step. Finally, a translation is modulated to produce the image $\mathbf{p}_{def} = \mathbf{p}_r + (\mathrm{W}(\mathbf{p}_R) - \mathrm{R}(\mathbf{p}_R))$.

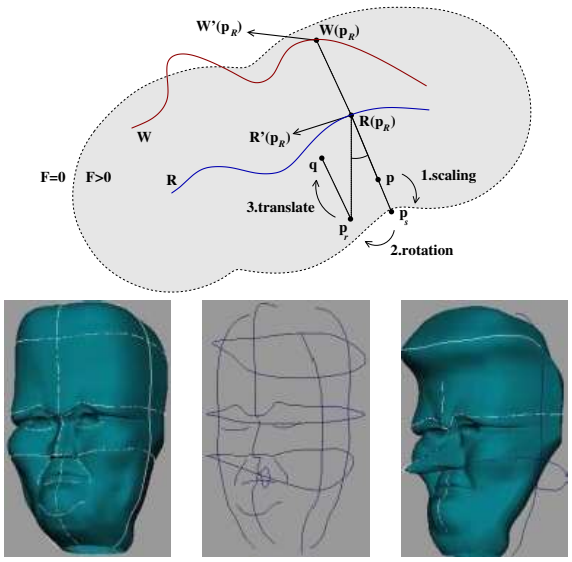
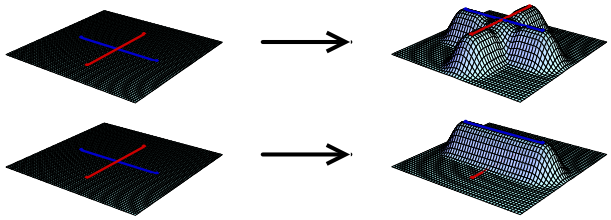


Figure 23: Left: deformation of a point by a single wire: the reference curve is in blue and the wire curve is in red. Right: deformation of a shape with multiple wires (the three images on the right by K. Singh and E. Fiume). The first image shows the initial shape, the second shows the reference curves and the third shows the wire curves and the deformed shape.

They propose different blending methods in the case when a point is subject to multiple wires. These methods work by taking weighted combinations of the individually deformed point. Let us denote $\mathbf{p}_i$ the deformation of $\mathbf{p}$ by wire i. Let $\Delta\mathbf{p}_i = \mathbf{p}_i - \mathbf{p}$. The simplest deformation is:

$$\mathbf{p}_{def} = \mathbf{p} + \frac{\sum_{i=1}^{n}\Delta\mathbf{p}_i \|\Delta\mathbf{p}_i\|^m}{\sum_{i=1}^{n}\|\Delta\mathbf{p}_i\|^m}$$



Reference curves Wire curves

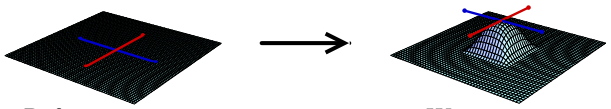
Figure 24: Blending weights based on summed displacement magnitudes. This blending is not free from artifacts: notice the creases around the intersection in the upper-right figure.

The scalar m is defined by the artist. This expression is not defined when m is negative and $\|\Delta\mathbf{p}_i\|$ is zero. To fix this, they suggest to omit the wires for which this is the case. Their second solution is to use another blending defined for both positive and negative values of m:

In order to use unmoved wires as anchors that hold the surface, they use $\mathrm{F}_i(\mathbf{p})$ instead of $\Delta\mathbf{p}_i$ as a measure of proximity:

Other capabilities of wires can be found in the original paper [Singh and Fiume 1998]. Note that important to the algorithm is computing the distance from each curve to each deformed surface point.

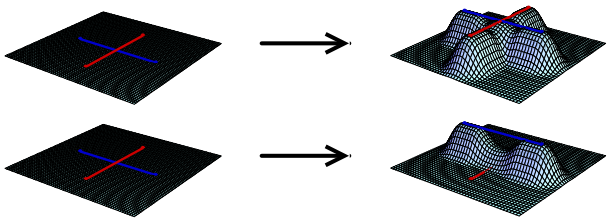
$$\mathbf{p}_{def} = \mathbf{p} + \frac{\sum_{i=1}^{n} \Delta\mathbf{p}_i \prod_{j\neq i} \|\Delta\mathbf{p}_j\|^{|m|}}{\sum_{i=1}^{n} \prod_{j\neq i} \|\Delta\mathbf{p}_j\|^{|m|}}$$



Reference curves Wire curves

Figure 25: Blending weights based on multiplied displacement magnitudes. The deformation is defined at the intersection of the reference curves.

$$\mathbf{p}_{def} = \mathbf{p} + \frac{\sum_{i=1}^{n} \Delta\mathbf{p}_i \mathrm{F}_i(\mathbf{p})^m}{\sum_{i=1}^{n} \mathrm{F}_i(\mathbf{p})^m}$$



Reference curves Wire curves

Figure 26: Blending weights based on influence function. The unmoved wire holds space still. This blending is not free from artifacts: notice the creases around the intersection in the upper-right figure.

2.5.6 Sweepers: Modeling with Gesture

Sweepers is a frameworks for shape modeling by gesture[Angelidis et al. 2004b]. Note that there exist similar work [Gain and Marais 2005; Kil et al. 2006]. The input that defines transformations is a gesture, obtained with a mouse or hand tracking device. A simple space deformation can be defined with a 4×4 transformation matrix M (translation, rotation, scale, etc.) whose effect is spatially weighted with a scalar field $\varphi(\mathbf{p}) \in [0, 1]$. The function φ encodes the amount of transformation at $\mathbf{p} \in \mathbb{R}^3$. To define a tool, a funtion φ is defined by composing a smooth function μ to the distance to a shape.

$$\mu_\lambda(d) = \begin{cases} 0 & \text{if} \quad \lambda \leq d \\ 1 + (\frac{d}{\lambda})^3(\frac{d}{\lambda}(15 - 6\frac{d}{\lambda}) - 10) & \text{if} \quad d < \lambda \end{cases} \tag{46}$$

This function is C^2-continuous and antisymmetric about 0.5. The tools proposed in the original papers are a ball tool, a filled ellipsoid tool, or more generic mesh tools.

There are several ways to weight a transformation with a weight, and sweepers uses fractions of transformation, by using the exponential and logarithm of matrices (see a complete overview in [Alexa 2002]). Note that as opposed to the numerical method proposed by Alexa operator, sweepers *do not evaluate* exp *and* log *numerically*, since some cases reduce to more efficient and elegant closed-form formulas. Thus the transformation M can be weighted with φ as follows:

$$\ddot{f}(\mathbf{p}) = \exp(\varphi(\mathbf{p}) \log M) \cdot \mathbf{p} \tag{47}$$

The deformation $\ddot{f}$ is naive since it can create a foldover. For example, if M is a translation of large magnitude, it can map points within the support of φ onto points outside from the support of φ, thus folding space onto itself.

Single Sweeper: By decomposing the transformation into a series of s small enough transformations, and applying each of them to the result of the previous one, foldovers are avoided, for the same reason that the solution to a first order differential equation is foldover-free (there is no acceleration). The decomposition in s steps for a general transformation is expressed as follows:

$$\begin{aligned} f(\mathbf{p}) &= \bigcirc_{k=0}^{s-1} f_k(\mathbf{p}) \\ \text{where } f_k(\mathbf{p}) &= \exp(\tfrac{\varphi^k(\mathbf{p})}{s} \log M) \cdot \mathbf{p} \\ \text{and } \varphi^k(\mathbf{p}) &= \varphi(\exp(-\tfrac{k}{s} \log M) \cdot \mathbf{p}) \end{aligned} \tag{48}$$

The value returned by φ^k is that of the scalar field φ transformed by $\exp \frac{k}{s} \log M$, a fraction of M. It can be shown that there exists a finite number of steps such that the deformation is foldover-free (see [Angelidis 2005]). We propose the following as a lower bound to the required number of steps s:

$$\max_{\mathbf{p}} \|\nabla\varphi(\mathbf{p})\| \max_{l\in[1,8]} \|\log(M) \cdot \mathbf{p}_l\| < s \tag{49}$$

where $\mathbf{p}_{l\in[1,8]}$ are the corners of a box oustide which the function φ equals zero.

Efficiency: In a single tool scenario, the transformations convenient to input are translations, non-uniform and uniform scaling and rotations. In these cases, there is a closed-form to the logarithm and exponential of matrices. If M is a translation of vector $\mathbf{d}$, the minimum number of steps is:

$$\max_{\mathbf{p}} \|\gamma(\mathbf{p})\| \ \|\mathbf{d}\| < s \tag{50}$$

The s vertex and normal deformations are:

$$f_k(\mathbf{p}) = \quad \mathbf{p} + \tfrac{\varphi_k(\mathbf{p})}{s}\mathbf{d} \tag{51}$$

$$g_k(\mathbf{n}) = \quad \mathbf{n} + \tfrac{1}{s}(\gamma_k \times \mathbf{n}) \times \mathbf{d} \tag{52}$$

If M is a uniform scaling operation of center c and scaling factor σ, the minimum number of steps is:

$$\max_{\mathbf{p}} \|\gamma(\mathbf{p})\| \ \sigma \log(\sigma) d_{\max} < s \tag{53}$$

where $d_{\max}$ is the largest distance between a point in the deformed area and the center $\mathbf{c}$, approximated using a bounding box. Let $\vec{\chi} = \frac{\log(\sigma)}{s}(\mathbf{p} - \mathbf{c})$. The s vertex and normal deformations are:

$$f_k(\mathbf{p}) = \quad \mathbf{p} + (\sigma^{\frac{\varphi_k(\mathbf{p})}{s}} - 1)(\mathbf{p} - \mathbf{c}) \tag{54}$$

$$g_k(\mathbf{n}) = \quad \mathbf{n} + (\gamma_k \times \mathbf{n}) \times \vec{\chi} \tag{55}$$

If M is a rotation of angle θ, center $\mathbf{r}$ and axis $\mathbf{v} = (v_x, v_y, v_z)^\top$, the minimum number of steps is:

$$\max_{\mathbf{p}} \|\gamma(\mathbf{p})\| \ \theta r_{\max} < s \tag{56}$$

where $r_{\max}$ is the distance between the axis of rotation and the farthest point from it, approximated using a bounding box. The s vertex deformations are:

$$f_k(\mathbf{p}) = \mathbf{p} + (\cos\frac{\varphi_k\theta}{s} - 1)\xi \times \mathbf{n} + \sin\frac{\varphi_k\theta}{s}\xi \quad \text{where } \xi = \mathbf{v} \times (\mathbf{p} - \mathbf{r}) \tag{57}$$

The s normal deformations are:

$$\begin{aligned} g_k(\mathbf{n}) &= (\mathbf{n} \cdot \mathbf{v})\mathbf{v} + \mathbf{v} \times (\cos(h)\mathbf{n} \times \mathbf{v} - \sin(h)\mathbf{n}) \\ &\quad + \theta\gamma \times (\mathbf{n} \times \xi \\ &\quad + ((\cos(h) - 1)(\mathbf{n} \times \xi) \cdot \mathbf{v} \\ &\quad + \sin(h)\mathbf{n} \cdot \xi)\mathbf{v}) \\ \text{where } h &= \tfrac{\varphi_k\theta}{s} \end{aligned} \tag{58}$$

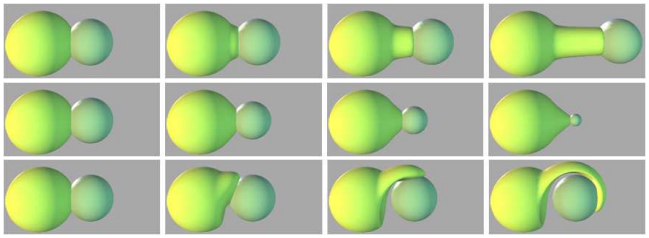


Figure 27: Translatiom, scale and rotation.

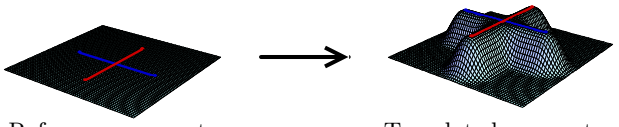
Simultaneous Sweepers: Let us consider n operations, defined with $M_{i\in[1,n]}$ and $\varphi_{i\in[1,n]}$. A naive way to achieve simultaneous deformations is

$$f(\mathbf{p}) = \exp(\sum_{i=1}^{n} \varphi_i(\mathbf{p}) \log M_i) \cdot \mathbf{p} \tag{59}$$

This function is naive because it adds the effect of each operation. The following expression provides a *normalized* and *smooth*[3] combination of all the transformations at any point $\mathbf{p}$ in space[4]:

$$\begin{cases} \mathbf{p} & \text{if } \sum_k \varphi_k = 0 \\ \exp(\sum_{i=1}^{n} \left(\frac{1-\prod_k (1-\varphi_k)}{\sum_k \varphi_k} \varphi_i \log M_i \right) \cdot \mathbf{p}) & \text{otherwise} \end{cases} \tag{60}$$

where $\frac{1}{\sum_k \varphi_k}$ is required to produce a *normalized* combination of the transformations and $1-\prod_{k=1}^{n}(1-\varphi_k(\mathbf{p}))$ *smooths* the deformation in the entire space. Figure 29 shows a comparison between additive blending of Equation (59) and the correct one of Equation (60). In Figure 28, we show the blending of sweepers in a secenario similar to other blending presented in this section.



Reference segments Translated segments

Figure 28: Blending with sweepers. The resulting surface is nice and smooth, as opposed to surfaces in Figures 24 25, 26, 21 and 22.

Equation (60) may produces foldovers for similar reasons to the case of a single tool, with Equation (47). If we decompose it into small steps, foldovers can be avoided:

$$\begin{aligned} f(\mathbf{p}) &= \bigcirc_{k=0}^{s-1} f_k(\mathbf{p}) \\ \text{where } f_k(\mathbf{p}) &= \begin{cases} \mathbf{p} & \text{if } \sum_j \varphi_j^k = 0 \\ & \text{otherwise} \\ \exp(\sum_{i=1}^{n} \left(\frac{1-\prod_j (1-\varphi_j^k)}{\sum_j \varphi_j^k} \varphi_i^k \log M_i \right)) \cdot \mathbf{p} \end{cases} \\ \text{and } \varphi_j^k(\mathbf{p}) &= \exp(\varphi_j((\tfrac{k}{s} \log M_j^{-1})) \cdot \mathbf{p} \end{aligned} \tag{61}$$

The following expression is a lower bound to the required number of steps, generalizing the single tool condition (see justification in [Angelidis 2005]):

$$\sum_j \max_{\mathbf{p}}(\|\nabla\varphi_j(\mathbf{p})\|) \max_{l\in[1,8]} \left|\left|\log M_j \cdot \mathbf{p}_{l_j}\right|\right| < s \tag{62}$$

[3] as smooth as the φ_i.

[4] The operator $\bigoplus$ expresses a repetive sum: $\bigoplus_{i=1}^{n} M_i = M_1 \oplus M_2 \oplus \cdots \oplus M_n$.

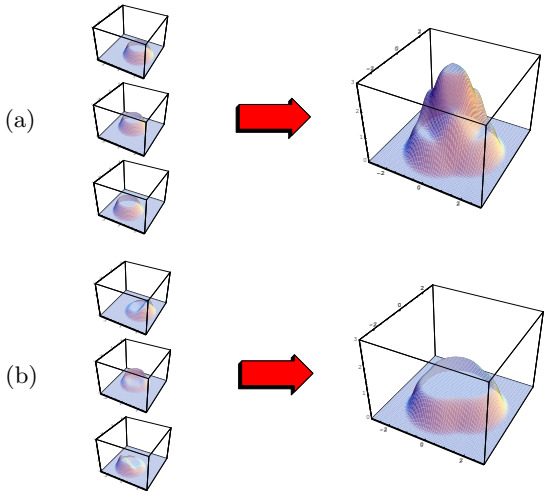


Figure 29: Blending of three scalar fields. To illustrate the behaviour of our blending in this figure, we directly combine the scalar fields instead of using them to modulate a transformation. (a) Adding the scalar fields. (b) By multiplying each field with $(1-\prod(1-\varphi_k))/\sum\varphi_k$, the sum of the fields is normalized.

where $\mathbf{p}_{l_j\in[1,8]}$ are the corners of a bounding box oustide which the function φ_j equals zero. For an operation symmetric about a plane, the transformation matrices are of the same type, thus blending them leads to simple expressions (see [Angelidis 2005]).

The set of possible deformations with sweepers is quite large because of the arbitrary shape of the tools and also because many tools' deformations can be blended. The shapes shown in Figure 30 were modeled in real-time in *one hour* at most, and were all made starting with a sphere.

Figures 30(a) and 30(b) show the use of the multi-tool to achieve smooth and symmetric objects. Figure 30(d) shows that sharp features can be easily modeled. Figures 30(c) and 30(i) show the advantage of foldover-free deformations, as the artist did not have to concentrate on avoiding self-intersections: our deformations do not change the topology of space and thus preserve the topology of the initial object.

2.5.7 Swirling-sweepers: Constant Volume Modeling

In a non-virtual modeling context, one of the most important factors which affects the artist's technique is the amount of available material. The notion of an amount of material is not only familiar to professional artists, but also to children who experience it with Play-Doh® at kindergarten, and to adults through everyday life experience. A shape modeling technique that preserves volume will take advantage of this, and increase the intuitiveness of use. In Swirling-Sweepers, the artist inputs a position $\mathbf{h}$ and translation $\mathbf{t}$, and the technique will create a deformation that transforms $\mathbf{h}$ into $\mathbf{h}+\mathbf{t}$ while the volume of the shape is preserved implicitly, simply because the deformation satisfies a differential property. Thus the volume of the shape does not require to be computed, and the deformation can be applied to an open surface.

Swirling-sweepers use *swirls* as building blocks. A swirl twists space locally without compression or dilation (see proof in [Angelidis et al. 2004a]): it preserves volume. A swirl is defined by a point $\mathbf{c}$, a rotation of angle θ around an axis $\mathbf{v}$ (see Figure 31), and a scalar function φ describing

the amount of swirl

$$\varphi(\mathbf{p},\lambda) = \mu(\frac{\|\mathbf{p}-\mathbf{c}\|}{\lambda}) \quad (63)$$

$$\text{where } \mu(d) = \begin{cases} 0 & \text{if } 1 \le d \\ 1 + d^3(d(15-6d)-10) & \text{if } 1 > d \end{cases}$$

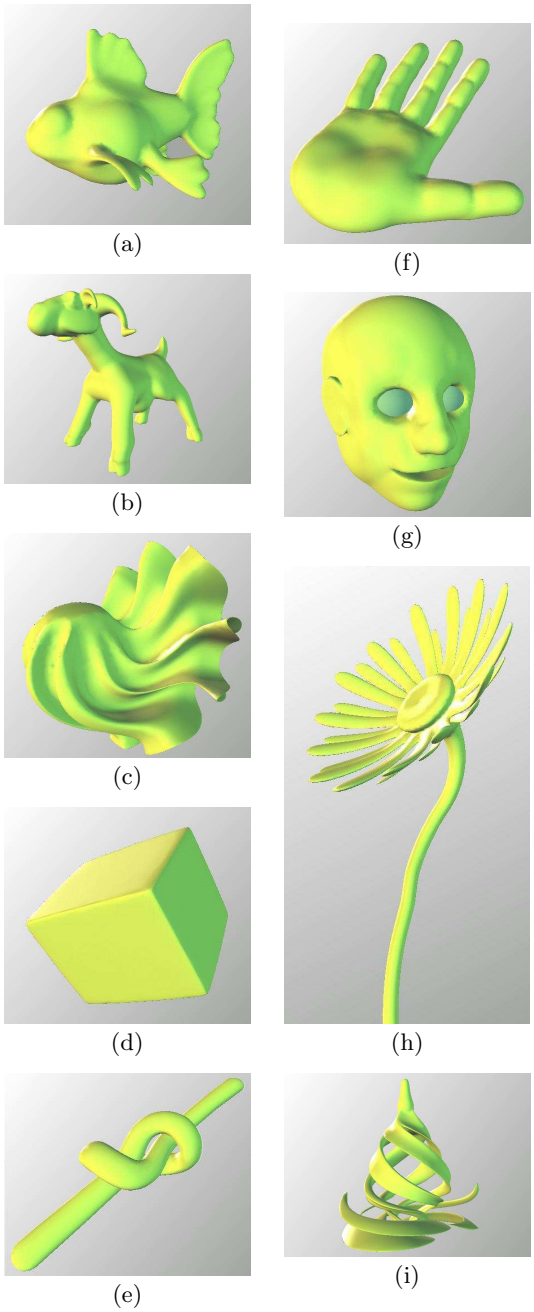


Figure 30: All these shapes were modeled starting with a sphere, in at most one hour. In (c), the first modeling step was to squash the sphere into a very thin disk. In (g), eyeballs were added.

The effect of a swirl is defined using the exponential and logarithm of matrices (see a complete overview in **??**), for which there are *closed-forms* given bellow:

$$f(\mathbf{p}) = \exp(\varphi(\mathbf{p},\lambda)\log R)\cdot\mathbf{p} \quad (64)$$

where R denote the 4×4 matrix of a rotation of center $\mathbf{c}$, axis $\mathbf{v}$ and angle θ. This equation could have been defined in several equivalent ways, for example using quaternions or an algebraic formula. The above notation is however convenient to combine multiple swirls:

$$f(\mathbf{p}) = \left(\exp(\sum_{i=0}^{n-1}(\varphi_i(\mathbf{p},\lambda)\ \log R_i))\right)\cdot\mathbf{p} \quad (65)$$

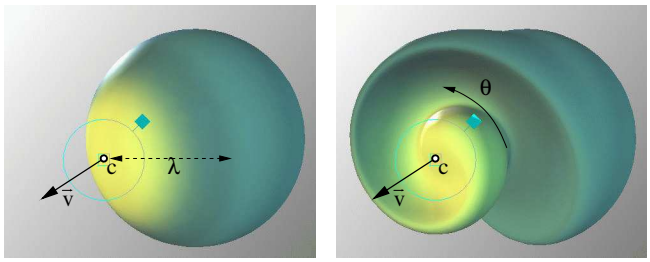


Figure 31: The effect on a sphere of a swirl centered at c, with a rotation angle θ around $\vec{v}$. The two shapes have the same volume.

We now define how to place the above swirls to transform $\mathbf{h}$ into $\mathbf{h}+\mathbf{t}$. Let us consider n points, $\mathbf{c}_i$, on the circle of center $\mathbf{h}$, and radius r lying in a plane perpendicular to $\mathbf{t}$. To these points correspond n consistently-oriented unit tangent vectors $\mathbf{v}_i$ (see Figure 32). Each pair, $(\mathbf{c}_i,\mathbf{v}_i)$, together with an angle, θ_i, define a rotation. Along with radii of influence $\lambda_i = 2r$, we can define n swirls. The radius of the circle r, is left to the user to choose. The following value for θ_i will transform $\mathbf{h}$ exactly into $\mathbf{h}+\mathbf{t}$ (see justification in [Angelidis et al. 2004a]):

$$\theta_i = \frac{2\|\mathbf{t}\|}{nr} \quad (66)$$

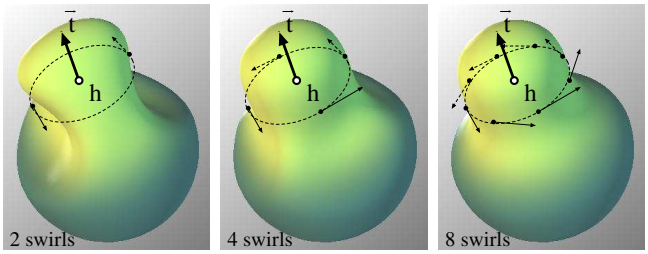


Figure 32: By arranging n basic swirls in a circle, a more complex deformation is achieved. In the rightmost image: with 8 swirls, there are no visible artifacts due to the discrete number of swirls.

We show in Figure 32 the effect of the tool for different values of n; in practice, we use 8 swirls. If the magnitude of the input vector $\mathbf{t}$ is too large, the deformation of Equation (65) will produce a self-intersecting surface, and will not preserve volume accurately. To correct this, it is necessary to subdivide $\mathbf{t}$ into smaller vectors for the same reasons that applies to solving discretely a first order differential equations. The number of steps must be proportional to the speed and inversely proportional to the size of the tool. We use:

$$s = \max(1, \lceil 4\|\mathbf{t}\|/r\rceil) \quad (67)$$

As the circle sweeps space, it defines a cylinder. Thus the swirling-sweeper is made of ns basic deformations. Figure 33 illustrates this decomposition applied to a shape. We summarize here the swirling-sweepers algorithm:

Input point $\mathbf{h}$, translation $\mathbf{t}$, and radius r
Compute the number of required steps s
Compute the angle of each step, $\theta_i = \frac{2\|\mathbf{t}\|}{nrs}$
for each step k from 0 to $s-1$ **do**
  for each point $\mathbf{p}$ in the tool's bounding box **do**
    $M = 0$
    for each swirl i from 0 to $n-1$ **do**
      $M \mathrel{+}= \varphi_k^i(\mathbf{p}) \log R_{i,k}$
    end for
    $\mathbf{p} = (\exp M) \cdot \mathbf{p}$
  end for
end for

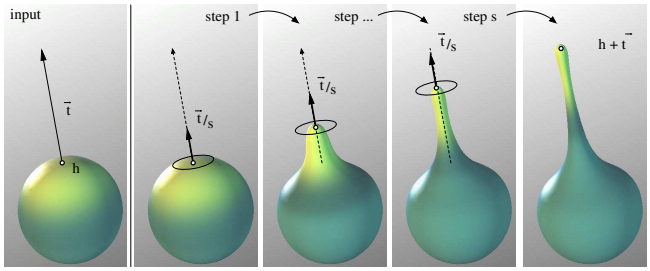


Figure 33: A volume preserving deformation is obtained by decomposing a translation into circles of swirls. 3 steps have been used for this illustration. As the artist pulls the surface, the shape gets thinner. The selected point's transformation is precisely controlled.

The point $\mathbf{c}_{ik}$ denotes the center of the i^{th} swirl of the k^{th} ring of swirls. For efficiency, a table of the basic-swirl centers, $\mathbf{c}_{ik}$, and a table of the rotation matrices, $\log R_{i,k}$, are precomputed. We have a closed-form for the logarithm of the involved matrix, given in Equations (68) and (69), saving an otherwise expensive numerical approximation:

$$\begin{aligned} \mathbf{n} &= \theta_i \mathbf{v}_i \\ \mathbf{m} &= \mathbf{c}_{i,k} \times \mathbf{n} \end{aligned} \tag{68}$$

$$\log M_{i,k} = \begin{pmatrix} 0 & -n_z & n_y & m_x \\ n_z & 0 & -n_x & m_y \\ -n_y & n_x & 0 & m_z \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{69}$$

Since these matrices almost antisymmetric, they are handled as pairs of vectors, $(\mathbf{n}, \mathbf{m})$. Once M is computed, we use a closed-form for computing $\exp M$. Since the matrix M is a weighted sum of matrices $\log R_{i,k}$, the matrix M is of the form of Equation (69), and can be represented with a pair $(\mathbf{n}_M, \mathbf{m}_M)$. If $\mathbf{n}_M = 0$, then $\exp M$ is a translation of vector $\vec{m}_M$. Else, if the dot product $\mathbf{m}_M \cdot \mathbf{n}_M = 0$, then $\exp M$ is a rotation of center $\mathbf{c}$, angle θ axis $\mathbf{v}$, as given by Equation (70):

$$\begin{aligned} \mathbf{c} &= \frac{\omega \times \mathbf{m}}{\|\omega\|^2} \\ \theta &= \|\mathbf{n}_M\| \\ \mathbf{v} &= \mathbf{n}_M / \theta \end{aligned} \tag{70}$$

Finally, in the remaining cases, we denote $l = \|\vec{n}_M\|$, and we use Equation (71) (see Appendix 2.5.7 for efficiency):

$$\exp M = \mathrm{I} + M + \frac{1-\cos l}{l^2} M^2 + \frac{l - \sin l}{l^3} M^3 \tag{71}$$

Symmetrical objects can be easily modeled by introducing a plane of symmetry about which the tool is reflected. The shapes shown in Figure 34 were modeled in real-time in *half an hour* at most, and were all made starting with a sphere. For instance 80 swirling-sweepers have been used to model the alien.

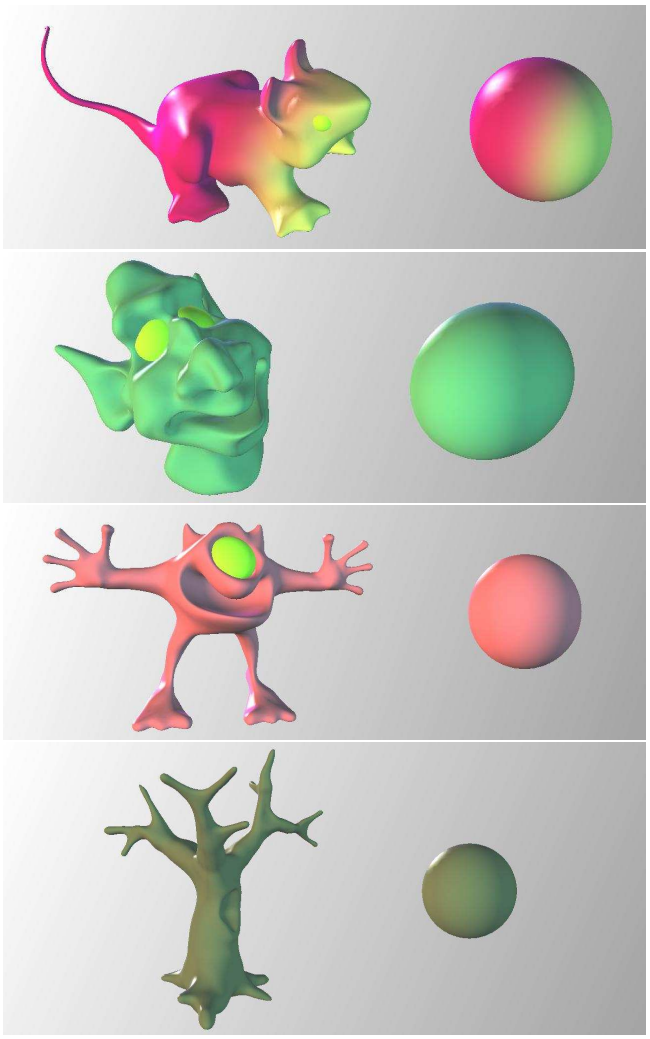


Figure 34: Examples of models modeled with swirling-sweepers. The mouse, the goblin, the alien and the tree have respectively 27607, 25509, 40495 and 38420 vertices. These objects were modeled in less than 30 min by one of the authors. Eyeballs have been added. The shapes volumes are respectively 101.422%, 99.993%, 101.158% and 103.633% of the initial sphere, due to the finite number of steps, and to our choice of shape representation.

3 Spape Deformation and Modeling

3.1 Desirable Properties for Modeling

The large number of space deformation techniques can lead quickly to the naive conclusion that in any shape modeling by deformation scenario, the limitation of a technique may be simply circumvented by using another technique. This reasoning presents several flaws. Firstly, from the point of view of a programmer, the amount of effort required to

implement a space deformation Swiss-army knife for shape modeling would be considerable. Secondly, from the point of view of an artist, choosing quickly the most appropriate space deformation would require a vast amount of knowledge of the underlying mathematics of many techniques, which is a skill that should not be required. Thirdly, from a researcher's point of view, all space deformation techniques are not necessarily designed for the specific purpose of shape modeling, and there are surely efficient ways of dealing with specific problems. Here a few guidelines for designing space deformation techniques for the purpose of interactive shape modeling.

Firstly, the subset of space deformations whose effect on a shape is not local makes these techniques unsuitable for the task of modeling shapes, since an artist's operation on a visible portion of the shape will have effects on portions that are further away [Barr 1984; Blanc 1994; Chang and Rockwood 1994; Lazarus et al. 1994]. Controlling the effect of global deformations using weights would requires a certain amount of craftsmanship from the artist.

Secondly, a large number of space deformation techniques requires the artist to specify a rather large number of control parameters [Sederberg and Parry 1986; Coquillart 1990; MacCracken and Joy 1996; Moccozet and Magnenat-Thalmann 1997; Hirota et al. 1999; Hua and Qin 2004]. We believe that for modeling, increasing the number of parameters does not increase the amount of control by an artist, but rather it makes the task longer and more tedious. Many techniques illustrate their capabilities on imported models, that were either digitized or pre-modeled with conventional modeling techniques with a few exceptions [Decaudin 1996; Hsu et al. 1992; Llamas et al. 2003]. The absence of a model entirely developed in one piece with a single technique may be evidence that the technique is tedious to use for the dedicated purpose of modeling shapes.

Finally, many space deformation techniques do not prevent a surface from self-intersecting after deformation, aside from a couple of exceptions [Mason and Wyvill 2001; Gain and Dodgson 2001]. A self-intersecting surface is a rather annoying situation in modeling with deformation, since it is impossible for a space deformation to remove a previously introduced self-intersection. Thus we believe that the following are reasonable guidelines for deformation operations for shape modeling:

- Its effective span should be controllable.
- Its input parameters should be reduced to their strict minimum: a gesture.
- It should be predictable, in accordance with a metaphor.
- It should be foldover-free.
- It should be sufficiently fast for existing computing devices.

3.2 A Shape Description for Modeling

Because space deformations operations are independent from the shape description, several choices are available to represent a shape being deformed: mesh [Gain and Dodgson 1999], particles [Pauly et al. 2003], deformed raytracing [Barr 1984], hybrid [Enright et al. 2002], and all the popular shape descriptions: subdivision surfaces, NURBS and more. In the context of shape modeling, the number of deformations is possibly excessively large, and issues related to such excess have to be taken into consideration when defining a shape description. This section presents a shape description for interactive modeling which supports high deformation and does not break when highly stretched [Angelidis et al. 2004b].

A simple way of representing a deformable shape is to place a set of samples on the surface of the shape: this makes the task of deforming the shape as straightforward as deforming the points on its surface. Points are discrete surface samples, and need to be somehow connected using splatting, interpolation or approximation scheme in order to display a continuous surface.

The presented method uses vertices connected with triangles. Connectivity provides convenient 2D boundary information for rendering the surface as well as surface neighborhood information, which enables the artist to define very thin membranes without having them vanish, as shown in Figure 30(c). The use of triangular C0 patches circumvents issues related to non-regular vertices and smoothness maintenance across the boundaries that join patches. Also, current hardware handles polygons very efficiently, which is relevant to us since interactivity is among our objectives. The reader however should be aware that point-sampled geometry is an active area of research [Pauly et al. 2003].

The possibly large number of deformations applied by an artist requires some minimum surface sampling density. In order to maintains this density, the presented method requires the deformation to be capable of being split into substeps.

Let us assume the scene is initialized with a polygonal model, e.g. a sphere with a homogeneous density of nearly equilateral triangles. To fetch the vertices that are deformed, a query is done with the tool's bounding box. Conveniently, this bounding box is also used in Equation 49. Since the principle of our swept deformations is to subdivide the input gesture into a series of smaller ones, all the transformations applied to the vertices are bounded. To take advantage of this decomposition in steps, we apply a modified version of a more generic algorithm [Gain and Dodgson 1999]. Our method requires keeping two vertices and two normals per vertex, corresponding to the previous and following state of some small step operation f_k. Loosely speaking, our surface-updating algorithm assumes that smooth curves run on the surface, and that the available vertices and normals should be able to represent them well enough. If this is not the case after deformation, then it means the surface is undersampled. On the other hand, if an edge is well enough represented by a single sample, then it is collapsed.

Let us consider an edge e defined by two vertices $(\mathbf{v}_0, \mathbf{v}_1)$ with normals $(\mathbf{n}_0, \mathbf{n}1)$, and the deformed edge e' defined by vertices $(\mathbf{v}'_0, \mathbf{v}'_1)$ with normals $(\mathbf{n}'_0, \mathbf{n}'_1)$. In addition to the conditions in [Gain and Dodgson 1999] based on edge length and angle between normals, we also base the choice of splitting edge e_0 on the error between the edge and a fictitious vertex, which belongs to a smooth curve on the surface. The fictitious vertex is used only for measuring the error, and is not a means of interpolating the vertices. If the error between the fictitious vertex and the edge is too large, the edge e is split, and the new vertex and normal are deformed. On the other hand if the fictitious vertex represents the edge e_0 well enough, then edge e is collapsed, and the new vertex is deformed. We define the fictitious vertex as the mid-vertex of a C^1 curve, since vertices and normals only provide 1^{rst} order information about the surface. The following cubic polynomial curve interpolates the vertices $\mathbf{v}'_0$ and $\mathbf{v}'_1$ with

corresponding shape tangents $\mathbf{t}_0$ and $\mathbf{t}_1$, defined below:

$$\begin{aligned}\mathbf{c}(u) &= (\mathbf{v}'_0(1+2u)+\mathbf{t}_0 u)(1-u^2)+ \\ &\quad (\mathbf{v}'_1(1+2(1-u))-\mathbf{t}_1(1-u))(1-(1-u)^2)\end{aligned} \quad (72)$$

The only constraint on tangent $\mathbf{t}_i$ is to be perpendicular to the corresponding normal $\mathbf{n}_i$. The following choice defines tangents of magnitude proportional to the distance between the vertices:

$$\begin{aligned}\mathbf{t}_0 &= \mathbf{g}-\mathbf{g}\cdot\mathbf{n}'_0\ \mathbf{n}'_0 \\ \mathbf{t}_1 &= \mathbf{g}-\mathbf{g}\cdot\mathbf{n}'_1\ \mathbf{n}'_1 \quad \text{where } \mathbf{g}=\mathbf{v}'_1-\mathbf{v}'_0\end{aligned} \quad (73)$$

With the above tangents, the expression of the middle vertex simplifies:

$$\mathbf{c}(0.5) = (\mathbf{v}'_0+\mathbf{v}'_1+(\mathbf{g}\cdot\mathbf{n}'_0-\mathbf{g}\cdot\mathbf{n}'_1)/4)/2 \quad (74)$$

With the fictitious vertex $\mathbf{c}(0.5)$, the tests to decide whether an edge should be split or collapsed can now be defined:

Too-long edge: An edge e0 is too long if *at least one* of the following conditions is met:

- The edge is longer than $L_{\max}$, the size of a grid-cell. This condition keeps a minimum surface density, so that the deformation can be caught by the net of vertices if the coating thickness λ_j is greater than $L_{\max}$.
- The angle between the normals $\mathbf{n}'_0$ and $\mathbf{n}'_1$ is larger than a constant $\theta_{\max}$. This condition keeps a minimum curvature sampling.
- The distance between the fictitious vertex and the mid-vertex of e' is too large (we used $L_{\max}/20$). This condition prevents the sampling from folding on itself, which would produce multiple sampling layers of the same surface.

Too-short edge: An edge e' is too short if all of the following conditions are met:

- The edge's length is shorter than $L_{\min}$ (we used $L_{\max}/2$).
- The angle between the normals $\mathbf{n}'_0$ and $\mathbf{n}'_1$ is smaller than a constant $\theta_{\min}$.
- The distance between the fictitious vertex and the mid-vertex of e' is too small (we used $L_{\min}/20$).

Also, to avoid excessively small edges, an edge is merged regardless of previous conditions if it is too small (we used $L_{\min}/20$).

We stress that the procedure for updating the mesh is applied at each small step, rather than after the user's deformation function has been applied. Because vertex displacements are bounded by the foldover-free conditions, the update of our shape description does not suffer from problems related to updating a greatly distorted triangulation. Figure 35 shows a twist on a simple U-shape. Figure 36 shows the algorithm preserving a fine triangulation only where required. Figure 37 shows the algorithm at work in a more practical situation. The procedure outline is:

```
Compute the number of steps required s
for each step k do
  Deform the points, and hold their previous values
  for each too-long edge do
    split the edge and deform the new point.
  end for
  for each too-short edge do
    collapse the edge and deform the new point.
  end for
end for
```

Limitation: With the updated mesh method, we choose to ignore the history of functions applied to the shape by the artist. Thus we "collapse" the history by freezing it in the current shape. To explain the major consequence of this, let us suppose the scene at a time t_k, such that the shape $S(t_k)$ is shown to the user. The next deformation produced by the artist with the mouse is function $f_{t_k \mapsto t_{k+1}}$, and all the mesh renements and simplications are performed in $S(t_k)$. This is however an approximation: ideally the last operation should be concatenated to the history of deformations, and the whole series should be applied to the initial shape $S(t_0)$, i.e. $\bigcirc_{i=0}^{n} f_{t_i \mapsto t_{i+1}}$ should be applied to each new vertex. This would however become more and more time consuming as the sequence of deformations gets longer (n gets larger), and the modeling software would eventually become unusable.

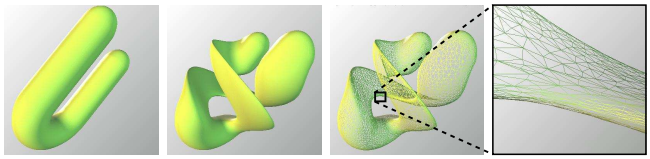


Figure 35: Example of our mesh-updating algorithm on a highly twisted U-Shape. The close-up shows a sharp feature, with finer elongated triangles.

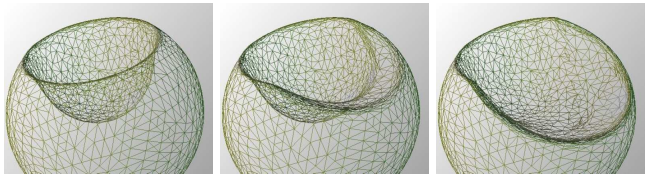


Figure 36: Behaviour of our mesh-updating algorithm on an already punched sphere. The decimation accompanying the second puch simplifies the small triangles of the first punch. The tool has been removed for a better visualization.

4 Comparing Techniques

Space deformations can be compared according to several criteria, thus the sequential presentation of Section 2 does not give the entire picture of the landscape of space deformations. We will identify objective criteria to attempt comparing techniques on a fair ground.

- *Modeling philosophy*: for the task of deforming a shape, the intended usage of a technique can be either to use many simple deformations, or a few but complex deformations. For example, a deformation that require the user to define a complex control structure will most likely be of the "few but complex deformations" kind. These approaches tend to be most efficient in the context of animation.

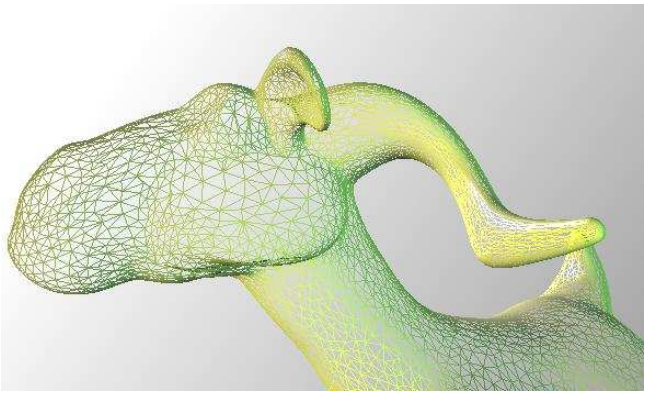


Figure 37: Close-up of the goat. Notice the large triangles on the cheek and the fine ones on the ear. The initial shape is a sphere.

- *Connectivity of control space*: deformations define several control parameters of type position, direction, affine transform, thickness, and more. Some of these controls have a direct relation with the Euclidean space, and we accord more importance to these since they are manipulated by the user in a geometric sense. The connectivity of the Euclidean controls can be $0D$, $1D$, $2D$ or $3D$, corresponding to the notion of parameter/point, a curve, a surface, or a block of jello. For example, a mouse can handle a $0D$ control, and will need to be used repeatedly to control higher dimensions, as opposed to a curve control interface [Grossman et al. 2003] that can contol a $1D$ control space all at once. Note that this is different from the dimension of the input related to hardware limitations: e.g. a mouse inputs $2D$ coordinates, while a curve control interface inputs $3D$ coordinates.

- *Free Control Blending*: all deformations can be combined together by combining the deformations with a partition of unity defined in space (see Section 1.1). Some deformation techniques include geometric blending more strongly in their formalism, and define blending methods that provides a variety of user control and a level of freedom in placing the control handles. These methods are have advantages as techniques with $3D$ connectivity control space, without any cumbersome structural constrain.

- *Differential properties*: by taking into account the time parameter, a deformation can be understood as a continuum deformation. By satisfying some differential properties, a deformation can implicitly preserve some properties of the shape being deformed such as surface self-interection avoidance, preserving volume or dynamics.

5 Conclusion

Space deformation is a set of very generic techniques that may be used in the context of modeling, rendering [Coleman and Singh 2004; Mei et al. 2005], animation and simulation. This overview focuses on the context of shape modeling, and we also present a method for representing the shape being deformed. Recent work has shown that it is possible to define properties of the shape with differential properties of the deformation. This may be a promissing direction for future work. Also, the similarity between space deformation and vector fields makes space deformation a pedagogical tool for understanding continuum mechanics, and they can be somehow used for handcrafting physical phenomena. For example, swirling-sweepers [Angelidis et al. 2004a] shares similarities with vortex-based smoke simulation [Angelidis and Neyret 2005].

Acknowledgments Many thanks to Marie-Paule Cani, Geoff Wyvill and Scott King for their contribution to the work presented in this chapter.

References

Alexa, M. 2002. Linear Combination of Transformations. *ACM Trans. Graph. 21*, 3 (Jul), 380–387.

Angelidis, A., and Neyret, F. 2005. Simulation of Smoke Based on Vortex Filament Primitives. In *SCA'05: Proc. of the 2005 Symposium on Computer Animation*, 87–96.

Angelidis, A., Cani, M.-P., Wyvill, G., and King, S. 2004. Swirling-sweepers: Constant-volume modeling. In *Pacific Graphics 2004*, IEEE, 10–15. Best paper award at PG04.

Angelidis, A., Wyvill, G., and Cani, M.-P. 2004. Sweepers: Swept user-defined tools for modeling by deformation. In *Proceedings of Shape Modeling and Applications*, IEEE, 63–73. Best paper award at SMI04.

Angelidis, A. 2005. *Shape Modeling by Swept Space Deformation*. PhD thesis, University of Otago.

Bajaj, C., Blinn, J., Bloomenthal, J., Cani-Gascuel, M.-P., Rockwood, A., Wyvill, B., and Wyvill, G. 1997. *Introduction to Implicit Surfaces*. Morgan-Kaufmann.

Barr, A. 1984. Global and Local Deformations of Solid Primitives. In *ACM Trans. Graph. (Proc of SIGGRAPH'84)*, 21–30.

Blanc, C. 1994. A generic implementation of axial procedural deformation techniques. In *Graphics Gems*, vol. 5, 249–256. Academic Press.

Bloomenthal, J. 1990. Calculation of reference frames along a space curve. *Graphics gems*, 567–571.

Borrel, P., and Bechmann, D. 1991. Deformation of n-dimensional objects. In *Proceedings of the first symposium on Solid modeling foundations and CAD/CAM applications*, 351–369.

Borrel, P., and Rappoport, A. 1994. Simple constrained deformations for geometric modeling and interactive design. In *ACM Transactions on Graphics*, vol. 13(2), 137–155.

Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z. 2002. Interactive Skeleton-Driven Dynamic Deformations. *ACM Trans. Graph. 21*, 3 (Jul), 586–593.

Chang, Y.-K., and Rockwood, A. P. 1994. A generalized de Casteljau approach to 3d free-form deformation. In *Proceedings of SIGGRAPH'94*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 257–260.

Reference	aka	Section	Phylosophy	Connectivity	Blendable	Differential properties
[Barr 1984]		2.1.1		0D	✗	
[Blanc 1994]		2.1.2		0D	✗	
[Decaudin 1996]		2.1.3	m/s	0D	✗	
[Kurzion and Yagel 1997]	Ray-deflectors	2.1.4	m/s	0D	✗	
[Llamas et al. 2003]	Twister	2.1.5	m/s	0D	✗	
[Chang and Rockwood 1994]		2.2.1	f/c	1D	✗	
[Lazarus et al. 1994]		2.2.2	f/c	1D	✗	
[Crespin 1999]		2.2.2	f/c	1D,2D	✗	
[Mason and Wyvill 2001]	Blendeformer	2.2.3	m/s	0D,1D	✗	foldover-free
[Capell et al. 2002]		2.2.4	f/c	1D	✗	dynamics
[Singh and Kokkevis 2000]	SOFFD	2.3.1	f/c	2D	✗	foldover-free
[Sederberg and Parry 1986]	FFD	2.4.1	f/c	3D	✗	
[Coquillart 1990]	EFFD	2.4.2	f/c	3D	✗	
[Gain and Dodgson 2001]		2.4.3	m/s	3D	✗	foldover-free
[MacCracken and Joy 1996]	SFFD	2.4.4	f/c	3D	✗	
[Hua and Qin 2004]	SFD	2.4.5	f/c	3D	✗	
[Borrel and Bechmann 1991; Hsu et al. 1992]	DMFFD	2.5.1	m/s	0D	✓	
[Borrel and Rappoport 1994]	scodef	2.5.2	f/c	3D	✓	
[Moccozet and Magnenat-Thalmann 1997]	DFFD	2.5.3	f/c	0D	✓	
[Crespin 1999]	IFFD	2.5.4	m/s	0D	✓	
[Singh and Fiume 1998]	Wires	2.5.5	f/c	1D	✓	
[Angelidis et al. 2004b]	Sweepers	2.5.6	m/s	0D	✓	foldover-free
[Gain and Marais 2005]		2.5.6	m/s	0D	✓	foldover-free
[Angelidis et al. 2004a]	Swirling-sweepers	2.5.7	m/s	0D	✓	volume-preserving

Table 1: f/c means "few complex" and m/s means "many simple". All deformations are blendable in a sense, thus blendable above means that the deformation provide additionnal blending features.

Coleman, P., and Singh, K. 2004. Ryan: rendering your animation nonlinearly projected. In *NPAR '04*, ACM, 129–156.

Coquillart, S. 1990. Extended free-form deformation: A sculpturing tool for 3d geometric modeling. In *Proceedings of SIGGRAPH'90*, ACM Press / ACM SIGGRAPH, vol. 24(4) of *Computer Graphics Proceedings, Annual Conference Series*, ACM, 187–195.

Crespin, B. 1999. Implicit free-form deformations. In *Proceedings of the Fourth International Workshop on Implicit Surfaces*, 17–24.

Decaudin, P. 1996. Geometric deformation by merging a 3d object with a simple shape. In *Graphics Interface*, 55–60.

Enright, D., Fedkiw, R., Ferziger, J., and Mitchell, I. 2002. A Hybrid Particle Level Set Method for Improved Interface Capturing. *J. Comput. Phys. 183*, 1, 83–116.

Farin, G. 1990. Surfaces over Dirichlet tessellations. *Computer Aided Geometric Design 7(1-4)* (June), 281–292.

Gain, J., and Dodgson, N. 1999. Adaptive refinement and decimation under free-form deformation. *Eurographics'99 7*, 4 (April), 13–15.

Gain, J., and Dodgson, N. 2001. Preventing self-intersection under free-form deformation. *IEEE Transactions on Visualization and Computer Graphics 7*, 4 (October-December), 289–298.

Gain, J., and Marais, P. 2005. Warp sculpting. *IEEE Transactions on Visualization and Computer Graphics 11(2)* (Apr), 217–227.

Griessmair, J., and Purgathofer, W. 1989. Deformation of solids with trivariate b-splines. In *Eurographics Conference Proceedings*, Elsevier Science, 137–148.

Griffiths, D. J. 1999. *Introduction to Electrodynamics*, third ed. Prentice Hall.

Grossman, T., Balakrishnan, R., and Singh, K. 2003. An interface for creating and manipulating curves using a high degree-of-freedom input device. In *CHI 2003 Conference Proceedings*, 185–192.

Hirota, G., Maheshwari, R., and Lin, M. 1999. Fast volume-preserving free form deformation using multi-level optimization. In *Proceedings of the fifth ACM symposium on Solid modeling and applications*, ACM, 234–245.

Hsu, W. M., Hughes, J. F., and Kaufman, H. 1992. Direct manipulation of free-form deformations. In *Proceedings of SIGGRAPH'92*, ACM Press / ACM SIGGRAPH, vol. 26(2) of *Computer Graphics Proceedings, Annual Conference Series*, ACM, 177–184.

Hua, J., and Qin, H. 2004. Scalar-field-guided adaptive shape deformation and animation. *The Visual Computer 1*, 1 (April), 47–66.

Kil, Y., Renzulli, P., Kreylos, O., Hamann, B., Monno, G., and Staadt, O. 2006. 3d warp brush modeling. *Journal of Computer and Graphics, ELSEVIER 30(4)*.

KURZION, Y., AND YAGEL, R. 1997. Interactive space deformation with hardware assisted rendering. *IEEE Computer Graphics and Applications 17(5)* (September/October), 66–77.

LAZARUS, F., COQUILLART, S., AND JANCÈNE, P. 1994. Axial deformations: an intuitive deformation technique. In *Computer-Aided Design*, vol. 26(8), 607–613.

LLAMAS, I., KIM, B., GARGUS, J., ROSSIGNAC, J., AND SHAW, C. 2003. Twister: A space-warp operator for the two-handed editing of 3d shapes. In *SIGGRAPH*, vol. 22(3) of *ACM Transactions on Graphics, Annual Conference Series*, ACM, 663–668.

MACCRACKEN, R. A., AND JOY, K. I. 1996. Free-form deformations with lattices of arbitrary topology. In *Proceedings of SIGGRAPH'96*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 181–188.

MASON, D., AND WYVILL, G. 2001. Blendeforming: Ray traceable localized foldover-free space deformation. In *Proceedings of Computer Graphics International (CGI)*, 183–190.

MEI, C., POPESCU, V., AND SACKS, E. 2005. The occlusion camera. In *Eurographics 2005*, vol. 24(3).

MOCCOZET, L., AND MAGNENAT-THALMANN, N. 1997. Dirichlet free-form deformation and their application to hand simulation. In *Computer Animation'97*, 93–102.

PAULY, M., KEISER, R., KOBBELT, L., AND GROSS, M. 2003. Shape modeling with point-sampled geometry. In *Proceedings of SIGGRAPH'03*, vol. 22(3), ACM, 641–650.

RUTHERFORD, A. 1990. *Vectors, Tensors and the Basic Equations of Fluid Mechanics*. Dover.

SCHEIN, S., AND ELBER, G. 2004. Discontinuous free form deformations. In *Proceedings of Pacific Graphics*, IEEE, 227–236.

SEDERBERG, T., AND PARRY, S. 1986. Free-form deformation of solid geometric models. In *Proceedings of SIGGRAPH'86*, ACM Press / ACM SIGGRAPH, vol. 20(4) of *Computer Graphics Proceedings, Annual Conference Series*, ACM, 151–160.

SINGH, K., AND FIUME, E. 1998. Wires: a geometric deformation technique. In *Computer graphics, Proceedings of SIGGRAPH'98*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 405–414.

SINGH, K., AND KOKKEVIS, E. 2000. Skinning Characters using Surface Oriented Free-Form Deformations". In *Graphics Interface*, 35–42.

WEISSTEIN, E. Lagrange multipliers. From Mathworld – A Wolfram Web Ressource http://mathworld.wolfram.com/LagrangeMultipliers.html.