A Survey of Spatial Deformation from a User-Centered Perspective

JAMES GAIN
University of Cape Town
and
DOMINIQUE BECHMANN
LSIIT, University of Strasbourg

The spatial deformation methods are a family of modeling and animation techniques for indirectly reshaping an object by warping the surrounding space, with results that are similar to molding a highly malleable substance. They have the virtue of being computationally efficient (and hence interactive) and applicable to a variety of object representations.

In this article we survey the state of the art in spatial deformation. Since manipulating ambient space directly is infeasible, deformations are controlled by tools of varying dimension—points, curves, surfaces and volumes—and it is on this basis that we classify them. Unlike previous surveys that concentrate on providing a single underlying mathematical formalism, we use the user-centered criteria of versatility, ease of use, efficiency and correctness to compare techniques.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric Algorithms*

General Terms: Algorithms, Human Factors, Performance

Additional Key Words and Phrases: Free-form deformation, spatial deformation, warping

ACM Reference Format:

Gain, J. and Bechmann, D. 2008. A survey of spatial deformation from a user-centered perspective. ACM Trans. Graph, 27, 4, Article 107 (October 2008), 21 pages. DOI = 10.1145/1409625.1409629 http://doi.acm.org/10.1145/1409625.1409629

1. INTRODUCTION

Spatial deformations, first introduced by Barr [1984] and Sederberg and Parry [1986], deform an object by warping its ambient space. While this may at first seem convoluted and indirect, it has some considerable advantages: spatial deformations, because they operate on points sampled from an object, are independent of the underlying representation of the embedded object; by controlling the size and shape of the ambient space it is possible to control the scope of the deformation making it local or global, as required; unlike physical simulations (such as mass-spring, finite element, and boundary element methods) spatial deformation bypasses complex intra-object interactions with consequent efficiency benefits. In short, spatial deformations are a class of highly interactive, powerful and intuitive modeling techniques.

Spatial deformation can be formulated as a mapping $\mathcal{F} : \Re^3 \mapsto \Re^m \mapsto \Re^3$, from world space, X, through a local parameter space, U, defined by the deformation tools of a particular technique, to deformed world space, $\widetilde{X}$. The mapping $\mathcal{F}$ is achieved by two functions, the embedding and deformation functions, $\mathcal{E}(X) = U$ and $\mathcal{D}(U) = \widetilde{X}$. The composition of $\mathcal{D}$ and $\mathcal{E}$ constitutes spatial deformation:

$$\mathcal{F}(X) = \mathcal{D} \circ \mathcal{E}(X) = \mathcal{D}(\mathcal{E}(X)) = \mathcal{D}(U) = \widetilde{X}. \quad (1)$$

Some deformation methods exploit higher-dimensional spaces. In particular, Bechmann and Dubreuil [1993] use $\Re^4 \mapsto \Re^4$ to animate deformations in both space and time. Equation (1) is an adaptation of the formalism first presented by Bechmann [1994] in her survey of spatial deformation methods. In fact, most previous surveys [Bechmann 1994; Blanc et al. 1994; Bechmann 1998; Gain 2000; Milliron et al. 2002] concentrate on deriving mathematical frameworks that enhance an understanding and categorization of spatial deformations. An alternative (first suggested by Bechmann [1994] and applied by Gain [2000]) is to use extrinsic factors for evaluation, such as versatility, ease of use, efficiency and correctness. With few exceptions (notably Milliron et al. [2002]) these factors have been the driving force behind advances in spatial deformation.

The case for the representation independence of spatial deformation needs some clarification. It is not that a representation is maintained intact through the deformation process. While some deformations, such as Free-Form Deformation, can be applied natively to parametric surfaces [Nimscheck 1995], the results are so degree

Author's address: J. Gain, email: jgain@cs.uct.ac.za.

elevated as to be unusable. Rather, the representation is sampled and it is the point samples themselves that are deformed. This is ideal for a triangle mesh representation, where the vertices can be used as samples directly, but even here issues of over and under-sampling can become a concern.

Implicit surfaces represent a special case. They are typically either converted to a polygon mesh using marching cubes or rendered directly using ray casting. In either case, the efficient inside/outside test of the implicit formulation can be exploited. In order to retain this efficiency, a ray in deformed space ($\widetilde{X}$) is sampled, then undistorted (using the inverse of the deformation function $\mathcal{F}^{-1}$) and finally tested against the undeformed implicit object, to determine the ray hit. Of course, this requires the existence of an inverse, which is not always guaranteed, especially in the case of self-intersecting deformations.

In this article, we classify techniques by the dimension of their user-controlled manipulators (an approach first introduced by Bechmann [1998]). Thus, current contributions to the field can be grouped under four headings:

0D Point-Based Deformations. Under these schemes the user provides a set of point displacements, each comprising a point along with its intended motion and region of influence. The space which covers all the regions of influence and incidentally incorporates the deformable object is then warped to match these displacement constraints. Some variants locate derivative frames at constraint points to enable additional bending, twisting and scaling.

1D Curve-Based Deformations. Here deformations are controlled by one or more curves, which may affect either the entire space or a specified enclosing volume. This can be envisaged as distorting space to map from source to destination versions of curves.

2D Surface-Based Deformations. This category consists of deformations initiated by user-defined changes to a surface. These surfaces can be bivariate patches modified by repositioning control points, meshes of any shape and topology modified by direct manipulation of their vertices, or static surfaces encoded in a distance or vector field representation modified by sweeping them through space.

3D Volume-Based Deformations. Univariate curves can be extended to bivariate patches and from there to trivariate hyperpatches. These hyperpatches demarcate a volume whose shape is altered by repositioning control points. Techniques which transfer the resulting hyperpatch distortion to an embedded solid are among the earliest forms of spatial deformation.

At their most atomic, nearly all deformations are controlled by positioning and repositioning control points (and possibly by means of simple handles for specifying scaling, tangent orientation, deformation regions, etc.) but this is too microscopic to serve as a basis for classification. Rather, manipulators are the true conceptual and mathematical foundation of deformations: conceptual in the sense that users regard manipulators as the primary means of controlling deformation, with the result that a failure in understanding a particular dimension of manipulators (as is sometimes the case with volume-based deformations [Hsu et al. 1992]) can lead to unforeseen results, and mathematical in that manipulators are integral to the definition of a deformation scheme. Further, the dimensionality of manipulators is often the most telling difference (or conversely the strongest source of commonality) between competing schemes. These dimensions (point-, curve-, surface- and volume-based deformations) thus serve as a sound basis for classification.

Another way to group deformation schemes is by application domain. Although deformations have found diverse application in 2D warping, physical simulation and CAD, this inevitably boils down to the differences between their application in modeling and animation. Although there are some interesting distinctions: for instance resampling (covered in section 6.4) is more important in modeling than animation and, conversely, linking a deformation to an underlying skeleton [Chadwick et al. 1989; Singh and Kokkevis 2000; Capell et al. 2002] more applicable to animation than modeling, many deformation schemes are equally applicable in either domain.

Using our chosen classification scheme we compare the mathematical basis, algorithm, and user interface of each deformation technique. This is followed by an evaluation section in which all the techniques are assessed according to the following criteria:

Versatility. Ideally spatial deformations should exhibit both range (produce a wide variety of distortions) and power (achieve shape modifications in a direct and straightforward fashion). To this end, a comparison is made of control over the position, size and shape of the deformation boundary offered by each technique. Within this boundary, flexibility in specifying the continuity and density of the deformation function is also discussed.

Ease of Use. The HCI tenets of interface simplicity, consistency and flexibility are applied to evaluate the interfaces of deformation schemes according to how effectively the user is shielded from the complexities of the underlying mechanism, the degree of practice and prior experience needed to attain modeling proficiency and the extent to which they match attributes of real world free-form shape design. This comparison is done in a discursive manner deferring a detailed HCI-style experimental evaluation to later work.

Efficiency. Interactive feedback is crucial to the success of spatial deformation. Delays should not hinder exploratory design. We use an operation counting approach to quantify the computation cost of different methods.

Correctness. Neither the internal validity nor the external realism of a shape should be compromised by the application of spatial deformation methods. Self-intersection is a case in point: it contravenes the two-manifold property and is also physically impossible. The circumstances under which spatial deformations violate the validity of an embedded object are considered in detail.

Sections 2, 3, 4, and 5 present, respectively, volume-based, surface-based, curve-based, and point-based deformation methods. Section 6 compares these methods in terms of their versatility, ease of use, efficiency and correctness. In this section we also examine issues common to all deformation approaches, such as volume conservation, topology alteration, remeshing and self-intersection. Finally, we conclude in Section 7 with a brief discussion of future trends in spatial deformation.

2. VOLUME-BASED DEFORMATION

Volume-based deformations (as shown in Figure 1) use a lattice of control points to manipulate shape.

2.1 Bézier Free-Form Deformation

The term free-form deformation (FFD) was originally coined by Sederberg and Parry [1986] and applies to techniques that bind the distortion of a hyperpatch to an embedded solid. As first outlined by Bezier [1972], FFD imposes a parametric hyperpatch onto a portion of world co-ordinate space and links distortions in the hyperpatch to points sampled from an object. The hyperpatch can be visualized as a block of pliable jelly capable of twisting, bending and bulging under the influence of a set of control points.

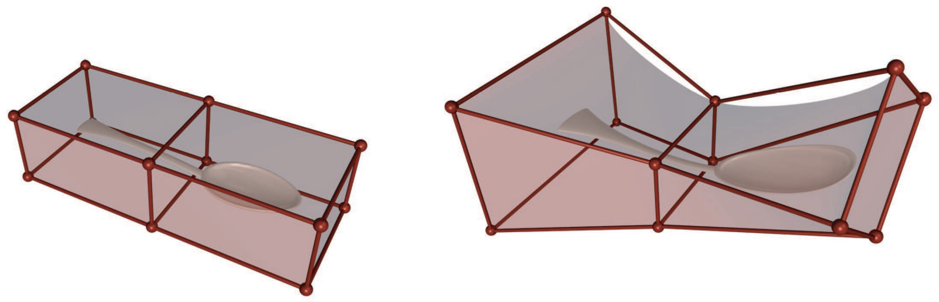
Fig. 1. Volume-based deformation. [left] Before deformation. [right] After deformation.

Mathematically, the deformation tool is defined as a trivariate polynomial tensor product volume. This is a straightforward extension of one-dimensional curves to three dimensions. A curve's control polygon, which indicates the univariate adjacency of control points, generalizes to a control lattice. Here lattice edges show the trivariate relationship between control points in three axial directions.

A user initiates deformation by moving lattice control points, and consequently reshaping the hyperpatch. Any sample point lying inside the lattice is deformed according to the hyperpatch deformation.

Sederberg and Parry [1986] use a hyperpatch defined with Bernstein polynomials. Points X within the hyperpatch are formulated as a sum of control points, P, weighted by polynomial basis functions, $\mathcal{N}_r^s$, with index r and degree s, as follows:

$$\begin{aligned}\mathcal{H}(U) &= \mathcal{H}(u, v, w)\\ &= \sum_{i=0}^{\ell} \sum_{j=0}^{m} \sum_{k=0}^{n} \mathcal{N}_i^{\ell}(u) \cdot \mathcal{N}_j^{m}(v) \cdot \mathcal{N}_k^{n}(w) \cdot P_{i,j,k} = X, \quad (2)\end{aligned}$$

where $u, v, w \in [0, 1]$ are parametric coordinates of a point within the hyperpatch. Early research by Davis and Burton [1991] and Kalra et al. [1992] proposed the use of rational Bernstein polynomials, which introduce an additional degree of freedom (weight) to the three existing freedoms (position) of control points. In practical terms, an increase in relative weight causes a control point to act as an attractor of object vertices and conversely a weight decrease causes repulsion. The benefits of weight parameters are debatable since a user might find the increased degrees of freedom bewildering. There is also a small computation overhead associated with rational Bernstein polynomials.

To deform an object, the FFD algorithm proceeds in four stages:

(1) *The deformation tool is created.* The user sets the initial position of a hyperpatch's control points (defines $\mathcal{H}$). In terms of the earlier jelly metaphor, the liquid jelly is poured into a handmade mold.

(2) *The object is linked to the deformation tool.* Sample points of the object (with Cartesian coordinates $X = (x, y, z)$) that fall within the initial hyperpatch are assigned parametric (u, v, w) coordinates relative to the lattice (Equation (2)). Metaphorically, the shape being deformed is placed within the jelly.

(3) *The deformation tool is modified.* A number of control points are displaced by the user (P becomes $\widetilde{P}$), with a resulting distortion of the hyperpatch ($\mathcal{H}$ becomes $\widetilde{\mathcal{H}}$). In the rational version, the weight of control points could also be altered. This equates to flexing the jelly.

(4) *The object is deformed.* The main idea behind FFD is that the parametric coordinates of sample points within the lattice are invariant despite deformation, it is the corresponding Cartesian coordinates that change. New Cartesian coordinates for object points are obtained from the post-deformation version of Equation (2) ($\widetilde{X} = \widetilde{\mathcal{H}}(U) = \widetilde{\mathcal{H}}(u, v, w)$). This is applied repeatedly to all of the sample points of the object to produce a deformed object. By analogy, the inset shape is warped along with its cocooning jelly.

In terms of Equation (1), FFD consists of a transition through hyperpatch parameter space by combining an embedding function, $\mathcal{E}(X) = \mathcal{H}^{-1}(X) = U$, with a deformation function, $\mathcal{D}(U) = \widetilde{\mathcal{H}}(U) = \widetilde{X}$.

Due to the properties of Bernstein polynomials, the displacement of a control point influences the entire space inside the lattice. To simplify embedding the object within the lattice (step (2)), the lattice is set in a cuboid configuration.[1] Thus, the boundary shape is limited to a cuboid volume. The shape and continuity of the deformation within this boundary is linked to the degree (l, m, n in Equation (2)) of the Bernstein polynomials. Global deformations are obtained when the whole object lies inside the volume. Local deformations are possible only when a limited part of the object lies inside the volume. In the latter case, the object intersects with the volume and thus continuity must be controlled on the boundaries.

Controlling deformations by moving lattice control points, while producing sculpted results, is both cumbersome and counterintuitive. Indeed, it is a remarkably awkward interface to the control of deformation.

Bézier FFD does suffer from shortcomings in versatility and ease-of-use but the extreme simplicity of its mathematical formalism has made it very popular. Its disadvantages have provided impetus for extensions in lattice topology and basis functions.

2.2 Free-Form Deformation with Local Control

Parametric surfaces have evolved to cope with issues such as irregular patch topologies and a lack of local control. Because of their close connection, hyperpatches suffer from many of the same shortcomings. Fortunately, the solutions applicable in the bivariate case generalize well to the trivariate. For instance, altering a single control point affects the entire hyperpatch and providing finer control by increasing the number of control points along an axis raises

[1]Strictly it is a parallelepiped since the $\vec{U}, \vec{V}, \vec{W}$ axes do not have to be orthogonal but, in practice, a cuboid is invariably used.

the degree of the hyperpatch and hence its computation cost. These problems are solved by substituting piecewise basis functions.

Greissmair and Purgathofer [1989] and Comninos [1989] opt for uniform B-spline basis functions, which have a piecewise nature and hence local support. Just as B-spline curves may be divided into piecewise segments (on a subinterval of the univariate domain), so too a B-spline hyperpatch may be broken into cells. Each cell is defined over a cuboid of the trivariate domain. Cells are controlled by a limited subset of control points, for example, $4 \times 4 \times 4 = 64$ control points in the case of a B-spline which is cubic in each $\vec{U}$, $\vec{V}$, $\vec{W}$ direction. The parametrization of control points is slightly complicated by their placement within a specific cell rather than the lattice frame as a whole. This is trivially determined since the initial lattice is again evenly spaced and the linear precision property implies a similar regular subdivision of the cells. The slight increase in parametrization cost is more than offset by the curtailed order of the basis functions.

Lamousin and Waggenspack [1994] combine the two strands of rational and piecewise bases. In their work FFD is built on nonuniform rational B-splines (NURBS), which have the local control properties of B-splines and the weighting associated with a rational formulation. Further, the knot vector is no longer constrained to a regular initial spacing. Some areas of the hyperpatch can have a higher concentration of knots than others and the locality of a deformation can vary across the lattice. This increased versatility has a trade-off in efficiency. As a consequence they employ a costly numerical search, separately in $\vec{U}$, $\vec{V}$, and $\vec{W}$, to find the local cell coordinates of a point. Lamousin and Waggenspack favor a cuboid lattice with unevenly spaced control points and an open uniform knot vector (with end-knot multiplicities one greater than the basis degree and knots evenly distributed in between). Linear parametrization is forfeit in this case but it need not be in general. A straightforward alternative is to have the user mark out the cell joints (domain knots) along each axis and from this automatically generate a lattice satisfying linear parametrization.

NURBS-based FFD does not allow truly variable lattice spacing since it enforces a tartan-like pattern of cells. For instance, cells adjacent in the $\vec{U}$ direction are restricted to the same v and w dimensions. Not so with rational T-spline FFD [Song and Yang 2005], which allows greater variability across neighboring cells and better multiresolution behavior. T-splines [Sederberg et al. 2003] are so named because, in the bivariate case, they allow, with some restrictions, a control mesh topology with T-junctions (i.e., 3-valence rather than 4-valence interior control points). In the trivariate case, this means 4, 5, and 6-valence control points. Song and Yang [2005] generate the lattice automatically by octree subdivision that clusters around the object surface. This requires far fewer control points than B-spline or NURBS FFD of equivalent power. Unfortunately, locating sample points in this lattice requires up to 15 conjugate gradient iterations. T-spline FFD forfeits any notion of interactivity, requiring minutes to embed even relatively simple objects with a few thousand vertices.

Due to the piecewise nature of splines, displacing a control point influences only a limited cluster of surrounding cells in the hyperpatch. Unfortunately, the shape and size of this region of influence depends on cell dimensions and spline degree and is difficult to predict and modify because it requires an expert knowledge of the underlying mathematics.

A piecewise basis enables local deformations with little complication in ensuring geometric continuity across cells. This does not preclude global deformations which can be achieved with a single cell. Unfortunately, a piecewise basis offers no improvement in ease-of-use. In fact it exacerbates screen clutter. NURBS and T-Spline FFD could be used if control point weighting or nonuniform cells are vital to the application context.

2.3 Free-Form Deformation with a Noncuboid Lattice

Free-form deformation in its original form is limited to lattices which induce a linearly parametrized hyperpatch. As a consequence, the initial lattice is cuboid shaped, which is a serious restriction on the both the deformation boundary and its interior density.

Extended Free-Form Deformation [Coquillart 1990] expands the range of initial lattice shapes by supporting user editing. The control points of an initially cuboid lattice can be moved or merged. For instance, a cylindrical lattice is created from a cuboid by collapsing one of the six faces into a central spine and then rotating and merging corresponding control points from the faces to either side. Additionally, composite lattices can be formed by joining control points from the boundary of separate lattices. Each individual lattice defines a Bézier hyperpatch as a trivariate Bernstein polynomial tensor product. Composite lattices define piecewise Bézier volumes.

There are four major issues with EFFD:

Restricted Topology. Within the limits of EFFD editing it is not possible to radically alter lattice topology. Relocating control points distorts cells without changing their overall cuboid configuration and widespread merging of control points is problematic.

Continuity. If C^1 continuity is to be maintained when merging control points then care must be taken to align the tangent vectors originating from the newly melded points. These considerations extend to the merging of more than two control points and also to higher continuity constraints. However, as lattice complexity increases and, in the face of degenerate cells caused by merging multiple control points, it becomes increasingly difficult to maintain anything more than positional continuity.

Parametrization. The complexity of noncuboid hyperpatches forfeits the linear precision property of Bézier curves and consequently a numerical search procedure is necessary for object point embedding ($\mathcal{H}^{-1}(X)$). The lattice may not intersect itself as this complicates sample point parametrization. Unique coordinates in hyperpatch parameter space cannot be established if cells, or their interior, overlap. As is to be expected, this is considerably more costly than previous embedding functions.

User Intervention. EFFD is also limited by the painstaking nature of lattice creation. Despite automatic control point alignment under certain types of merging, lattice creation tools such as extrusion of 2D lattices, and the ability to catalogue and reedit lattices, the process remains time-consuming and exacting.

In short, ease of use and efficiency are damaged without achieving truly general lattice arrangements. Nevertheless, EFFD did serve to point the way for later improvements.

In order to both generalize lattice topology and address the continuity problems of EFFD, Bechmann et al. [1996] propose the use of piecewise tetrahedral Bézier volumes. An initial volume is composed of one or more tetrahedral Bézier hyperpatches with control points evenly distributed with respect to a tetrahedron. This permits calculation of the parametric coordinates of a sample point within the lattice without resorting to Newton-Raphson iteration. Theoretically, assembling sets of tetrahedra enables a finite element like definition and control of spatial density without requiring the merging of control points. Thus, C^1 continuity can be maintained with no major difficulties.

An even greater variety of lattice arrangements and hence deformation shapes can be achieved by generalizing subdivision surfaces.

Historically, bivariate subdivision schemes generate successively finer control nets that converge to the surface. Each subdivision step has two components: (a) topology alteration, in which additional control points are connected into the net, and (b) positioning of these control points in relation to adjacent areas of the control net. Subdivision surfaces afford a complex and varied control net topology. Motivated by these topology benefits, MacCracken and Joy [1996] allow the user to specify a lattice of arbitrary topology (bar self-intersection) and use a trivariate version of Catmull-Clark subdivision [Catmull and Clark 1978] to recursively refine this lattice towards the hyperpatch. The resulting free-form deformation with lattices of arbitrary topology has an algorithmic structure which adheres broadly to other methods but differs in the specifics.

(1) *The deformation tool is created.* The user constructs a lattice ($\mathcal{L}_1$) and specifies its position and orientation relative to the embedded object.
(2) *The object is linked to the deformation tool.* Volumetric Catmull-Clark subdivision is then executed recursively and leads, after r cycles, to a lattice ($\mathcal{L}_r$) composed predominantly of hexahedral cells with, depending on the initial lattice topology, a number of nonhexahedral "extraordinary" cells. If a sample point falls within a given cell prior to deformation, it will remain in the same relative position in the corresponding cell through deformation. This is an approximation but it tends to exactness as the cell volume tends to zero. Hence, for parametrization purposes, recursion must continue until the subdivided cells are sufficiently small. Sample points are "tagged" by finding the cell to which they belong and then determining their local coordinates with a trilinear approximation.
(3) *The deformation tool is modified.* As with all free-form deformation techniques, the user then moves the control points of the lattice ($\mathcal{L}_1$) into a deformed configuration ($\widetilde{\mathcal{L}}_1$).
(4) *The object is deformed.* The deformed lattice is subdivided r times, giving rise to a sequence $\{\widetilde{\mathcal{L}}_1, \widetilde{\mathcal{L}}_2, \ldots, \widetilde{\mathcal{L}}_r\}$. The most refined versions of both lattices, $\mathcal{L}_r$ and $\widetilde{\mathcal{L}}_r$, are structurally equivalent. The parametrization "tags" are used to recover the deformed position of all sample points, by transferring the corresponding cell location into world space.

Subdivision-driven free-form deformation improves on extended and tetrahedral free-form deformation since the user is no longer forced to evolve lattices from restrictive (hexahedral or tetrahedral) initial arrangements. Merging control points is avoided and, but for exception points, C^2 continuity is assured.

Unfortunately, a subdivision approach is also more costly. Each vertex in the subdivided lattice is a non-trivial combination of control points in the previous lattice. The embedding procedure, where sample points are located within a specific cell, further increases the computation overhead. This algorithm is also memory intensive since each level of refinement nearly triples the size of the lattice data structure.

With subdivision surfaces the correspondence between control mesh and limit surface is clarified by rendering them in the same space. This is not possible with subdivision-based FFD. Further, the user manipulates the initial lattice of arbitrary topology rather than the intermediate or final refinements, making the connection between hyperpatch density, deformation boundary and control points even more difficult to visualize.

Configuring the initial lattice can be laborious, especially if it requires altering lattice connectivity. Often a user is most interested in a lattice that conforms to the object's overall shape. This is recognized by Ono et al. [2002] who automatically generate an octree-like lattice with the outermost control points shifted towards the object surface. The user can control the level of octree subdivision on a cell-by-cell basis, thereby enabling multiresolution control over FFD. After automatic lattice creation and user-driven subdivision, the technique proceeds along similar lines to subdivision-based FFD.

In theory, extended, tetrahedral, and subdivision-based FFD allow noncuboid initial lattice configurations that better fit the embedded object and enable finer control over the influence of a given control point. In practice, it can be difficult to predict the shape of the region of influence around a control point because of the interaction between cell shape and spline basis. These schemes thus address the issue of versatility to the detriment of efficiency and ease of use.

2.4 Summary

Free-Form Deformation in its various forms has found widespread application in animation: by key-framing the position of lattice control points over time [Coquillart and Jancène 1991], imposing a mass-spring model on the vertices and edges of the lattice for dynamics-based animation [Faloutsos et al. 1997], and simulating the muscle and tissue layer under geometric skin in character [Chadwick et al. 1989] and facial [Kalra et al. 1992] animation. Also, a two-dimensional version of FFD has been applied to image warping [Lee et al. 1995].

Free-Form Deformation and its extensions exhibit all the generality of parametric curves in representing the boundary and internal shape of a deformable region but, unfortunately, in extending from the bivariate to trivariate case there is a proliferation in control points that makes FFD difficult to use in practice.

3. SURFACE-BASED DEFORMATION

Surface-based deformations (as shown in Figure 2) use parametric patches, surface meshes, or vector fields to manipulate shape.

3.1 Parametric Patch Tools

The real difficulty with using surfaces to control deformation lies in finding a way to attach sample points. Feng et al. [1996] solve this by using an initially flat bivariate B-spline patch $\mathcal{S}(u, v)$ and projecting samples onto the plane of the patch. To deform a collection of samples from an object, the user alters the parametric patch by moving its control points (thereby defining $\widetilde{\mathcal{S}}(u, v)$).

In detail, the algorithm works as follows:

(1) *The deformation tool is created.* The control points of the initial patch $\mathcal{S}(u, v)$ are laid out in the XZ-plane $Y = 1$. The patch is planar but need not necessarily be rectangular. This defines a deformable region bounded by the four edges of the patch in X and Z and extending infinitely in Y.
(2) The object is linked to the deformation tool. For each sample point of the object $X = (x, y, z)$, parametric coordinates ($U = (u, v, w)$) are obtained by computing the projection X_p of X onto the surface along its normal. Then $(u, v) = \mathcal{S}^{-1}(X_p)$ is calculated by numerical search (since linear precision can only be relied upon if the patch is rectangular) and w is assigned as the distance from X to X_p. So, we have:

$$X = \mathcal{S}(u, v) + w \cdot \mathcal{N}(u, v), \tag{3}$$

where $\mathcal{N}(u, v)$ is the normal vector at point (u, v) of the surface $\mathcal{S}$.

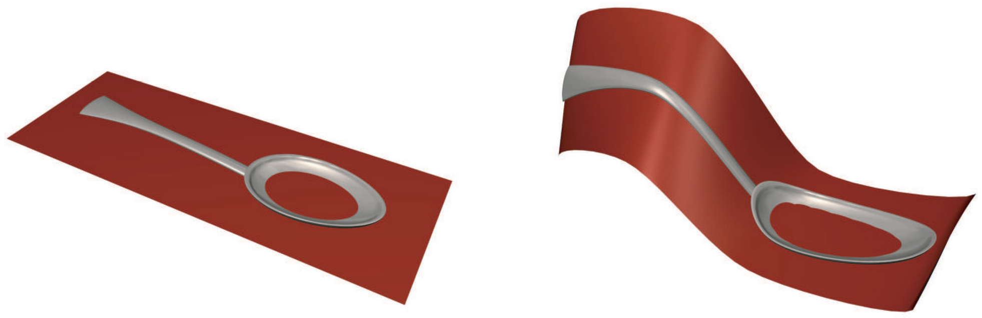
Fig. 2. Surface-based deformation. [left] Before deformation. [right] After deformation.

(3) *The deformation tool is modified.* Control points P are displaced to positions $\widetilde{P}$ by the user, forming a new distorted, potentially nonplanar, patch $\widetilde{\mathcal{S}}$.

(4) *The object is deformed.* The assigned parametric coordinates of the sample points relative to the patch are invariant under deformation, so that new Cartesian coordinates of a sample can be obtained by applying:

$$\widetilde{X} = \widetilde{\mathcal{S}}(u, v) + w \cdot \widetilde{\mathcal{N}}(u, v), \tag{4}$$

Where $\widetilde{\mathcal{N}}(u, v)$ is the normal vector of the deformed surface $\widetilde{\mathcal{S}}(u, v)$. A variant of this model retains $\mathcal{N}(u, v)$ in place of $\widetilde{\mathcal{N}}(u, v)$ in the previous formula.

Through control of the so-called shape surface $\mathcal{S}$, it is quite possible to globally or locally bend or twist an object. However, to obtain tapering effects the authors use the y component of a height surface $\mathcal{H}(u, v)$ to modulate the w parameter.

This approach works well for bends, twists, and tapers, which can be either local or global depending on a suitable construction of the shape and height surfaces. In fact, for this subset of tasks it is more versatile than Barr's regular global deformations [Barr 1984] (covered in Section 4.1). On the other hand, complex deformations requiring precise displacements are next to impossible. The deformation boundary is an infinite extrusion along Y of the four bounding edges of the shape surface in XZ.

The interface is clumsy, relying as it does on the manipulation of two separate surfaces that must interact in creating the final deformation. In fact, a single surface would be preferable, perhaps with an extra height-scaling value attached to control points of the shape surface. Also, the reduction in degrees of freedom and consequently interface crowding (roughly from $3n^3$ for a hyperpatch to $6n^2$ for the patches) is not that impressive for smaller tools. Efficiency-wise, the embedding step suffers under the burden of numerical iteration, even if it is in two dimensions rather than three.

Using conventional B-spline patches restricts the deformation boundary to a rectangular topology in the XZ plane. To get around this, Feng et al. [2005] substitute subdivision surfaces, thereby allowing patches with variable topology. This is directly analogous to the extension of FFD to lattices of arbitrary topology and offers the same tradeoff of efficiency for versatility.

In conclusion, parametric patch tools are better for bends, twists, and tapers but worse for more general manipulations than volume-based deformations.

3.2 Star-Convex Tools

A star-convex object is one which contains in its interior a region such that any ray originating from that region intersects the object's surface only once. Otherwise concave objects such as stellated structures often have this property. From the point of view of deformation this is useful because it admits a unique polar coordinate parametrization of sample points [Jin and Li 2000].

As always, the four stage process applies:

(1) *The deformation tool is created.* A source surface $\mathcal{S}$, that is star convex and has a center O located inside the defining core, is specified by the user. The surface can have any representation: parametric, implicit or mesh, so long as it supports ray-surface intersection tests.

(2) *The object is linked to the deformation tool.* A ray is constructed, originating at O, passing through a sample point X, and intersecting the tool surface at P. The direction of the ray, U, and the ratio of distances from O to X and O to P are recorded.

(3) *The deformation tool is modified.* A star convex destination surface $\widetilde{\mathcal{S}}$, with center $\widetilde{O}$, is defined.

(4) *The object is deformed.* To deform a given sample, a ray is shot from $\widetilde{O}$ along the same direction as before (U) and the intersection $\widetilde{P}$ with the destination tool $\widetilde{S}$ is found. Next, the deformed sample ($\widetilde{X}$) is placed the same relative distance between origin and surface.

A straightforward variant replaces the tool center with an axis line to cope better with cylindrical star convex objects. As Jin and Li [2000] readily admit, the versatility of their approach is limited. Apart from the obvious restriction of star convex tools, the deformation itself is linear in each polar direction. This has two implications: the technique is global, since there is no attempt to taper deformation with distance, and there is no variation in internal density such as would be afforded by FFD. On the positive side, the approach is not as inefficient as it might at first appear. Although two ray-surface intersections are required per sample point, these can be accelerated by light buffer methods. Further, any underlying higher derivative discontinuities (C^1 or C^2) in the source or target surfaces will reflect in the deformation. A cube, for example, will induce 8 corresponding seams and corners in a deformed object. This could be a way of introducing crease edges into an otherwise smooth model but it is likely to be difficult to control.

Decaudin's [1996] Shape Merging Deformation can be viewed as a precursor to the star convex method in that it also relies on a polar parametrization. A single central point (C) is used to position a simple shape ($\mathcal{S}$), such as a sphere or ellipsoid. Then object space is expanded to make room for this shape. This induces a spherical (or ellipsoidal) bulge or dent depending on whether C lies inside or outside the object. An important departure from Jin and Lee's approach, where both the source and destination tools are surfaces, is that the initial deformation tool (C) is a single point.

The resulting deformation is smooth (it depends only on the continuity of the shape $\mathcal{S}$), local but not bounded (in that deformation tails off rapidly away from $\mathcal{S}$ but is never zero), volume conserving (for bulges, object volume is increased only by the volume of $\mathcal{S}$ and for dents, not at all), but again rather limited in versatility.

This general approach sacrifices versatility for the sake of user interface simplicity but other surface-based methods are, in the end, less limiting.

3.3 Triangle Mesh Tools

Another tack is to use the flexing of a relatively simple triangle mesh to dictate deformation of a more complex object. One might legitimately ask what really differentiates this from a subdivision surface tool, since they both rely on a control mesh? The answer is linear precision. The expectation is that a sample lying on a triangle will remain there as the triangle distorts. Even with interpolating subdivision schemes this is not guaranteed, except at mesh vertices.

The first instance of a triangle-mesh tool is t-FFD [Kobayashi and Ootsubo 2003]. Its key attributes are that tool triangles: (a) contribute in a weighted average to the deformation of a nearby sample, (b) have a spherical volume of influence with linear density, and (c) are allowed a very general arrangement.

t-FFD has the following stages:

(1) *The deformation tool is created.* An initial set of m triangles $\mathcal{S}$ is placed by the user. There are very few restrictions on this placement: $\mathcal{S}$ can be connected into a mesh, exist as a soup of disconnected triangles, and even self-intersect. Only degenerate triangles with coincident vertices are disallowed.

(2) *The object is linked to the deformation tool.* Every triangle is assigned its own coordinate system, with one vertex as the origin O_i, the two incident edges as axes U_i and V_i and the triangle normal as the third axis W_i. This is not an orthogonal coordinate system except in the case of right-angled triangles. A sample point is parametrised in $\Re^{4m}$ with $U = (u_i, v_i, w_i)$ coordinates local to the frame of each of m triangles and a weight value k_i, which decays linearly in a sphere of influence radiating from the triangle centroid out to a distance determined by triangle size. This particular distance metric is used for efficiency reasons but does have some adverse side effects.

(3) *The deformation tool is modified.* The triangle set is modified, either directly by repositioning vertices to produce $\widetilde{\mathcal{S}}$ or indirectly by selecting and moving points not on $\mathcal{S}$ and then reverse-engineering the configuration of $\widetilde{\mathcal{S}}$ necessary to induce the specified motion (in this guise it is actually a type of point-based deformation). In either case the triangles must be nondegenerate.

(4) *The object is deformed.* Changes in triangles induce a change in points attached to their coordinate frames. Thus, $\widetilde{X}_i = \widetilde{O}_i + u_i\widetilde{U} + v_i\widetilde{V} + w_i\widetilde{W}$. Finally, the effects of individual triangles are weighted, blended and normalized to produce a deformed sample: $\widetilde{X} = \sum_i k_i\widetilde{X}_i / \sum_i k_i$.

The effects of t-FFD vary from local to global depending on how extensively $\mathcal{S}$ is altered in deriving $\widetilde{\mathcal{S}}$. For instance, if $\mathcal{S}$ is a triangle mesh, the deformation induced by moving a single vertex is bounded by the spheres of influence associated with the triangles incident on that vertex.

A cause for concern is that local deformations must be completely ringed by stationary triangles, otherwise extreme (C^0) discontinuities occur where the object crosses the spherical limit of a triangle's influence. This presents no difficulty if $\mathcal{S}$ is restricted to a triangle mesh that roughly tracks the object's surface. A disconnected triangle soup is less manageable because a user must determine by inspection whether or not the object strays outside overlapping spheres of influence.

The last point worth raising is the computation cost of t-FFD, which results from having to attach each sample point to multiple triangles. The authors report that a deformation, dominated by the parametrization step, typically requires several seconds. However, a variety of deformations can still be explored interactively as long as they arise from the same initial triangle configuration. This is a common argument [Singh and Fiume 1998] in support of schemes with a lengthy embedding and short deformation.

A way around the flaws in smoothness and continuity of t-FFD is offered by Ju et al. [2005]. They extend mean value coordinates (MVC) to closed triangular meshes. Under this formulation an object sample X has a coordinate value w_i with respect to every vertex p_i of a control mesh $\mathcal{S}$. If there are m vertices in $\mathcal{S}$ then the parametrization space U is $\Re^m$. A key feature of mean value coordinates is that they display linear precision, which means that the position of a sample can be recovered by an affine combination $X = \sum_i w_i \cdot p_i / \sum_i w_i$ of mesh vertices weighted by mean value coordinates. One implication is that a rigid body transformation of the control mesh will induce a matching rigid body transformation of the embedded object. Further, the resulting interpolation is continuous everywhere and smooth in the interior of the control mesh. The same certainly cannot be claimed of t-FFD. However, as a consequence of the MVC parametrization this is strictly a global deformation method, although vertex contributions can become almost vanishingly small.

MVC deformation requires a closed triangle mesh tool and is best when this also encloses the deformable object. This is obviously a more restrictive tool than the triangle soup available in t-FFD but it does free the user from having to worry about deformation coverage and continuity. We foresee MVC deformation being applied to character animation, where it will obviate the need for a predefined hierarchy of bones and joints. As to expense, the computation of mean value coordinates typically requires several seconds and is on a par with t-FFD, but deformation, since it is merely an m-element weighted sum, can be highly interactive.

3.4 Field-Based Tools

From a user's point of view most deformation schemes require two steps: setting up an initial (source) tool and then modifying it into a final (destination) form, thereby implicitly deforming an object. Linking source tool to object is often computationally expensive because, for a given object sample, it requires searching for the closest position on the source tool or attaching to every element of the source tool. Sweepers [Angelidis et al. 2004] and Warp Sculpting [Gain and Marais 2005] seek to address both of these issues and do so in similar ways. Kil et al. [2006] use the same principles but concentrate on user interaction.

Both methods emulate physical sculpting, where a tool, such as a chisel, stylus or hand, is employed by an artist to mould the shape of an object. In implementation terms a tool surface is swept through space, altering the object along its trajectory. Although tool movement is dynamic, the tools themselves have a fixed shape. This allows their conversion to a sampled distance field representation during an off-line preprocess, which in turn enables fast online reconstruction of the distance from a point sample to the tool's surface.

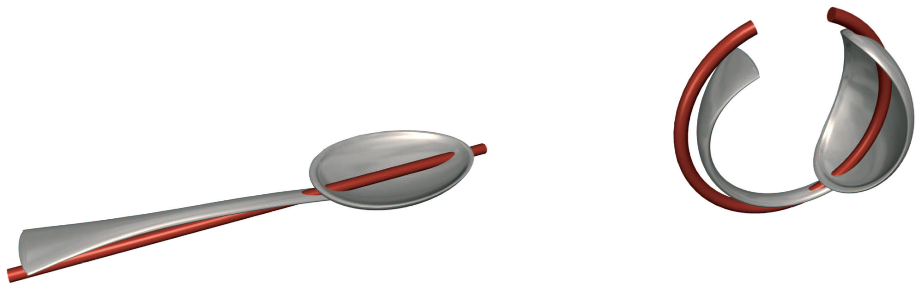

Fig. 3. Curve-based Deformation. [left] Before deformation. [right] After deformation.

The techniques proceed as follows: the user interactively specifies the initial and final position and orientation of the tool; then, for a given sample X, the shortest distance to the tool's surface $d = \phi(X)$ in its initial configuration is evaluated by interpolating distance field values; next, a decay function $w = w(d)$ is applied to taper deformations from a value of 1 on or inside the tool to 0 at or beyond a region of influence; finally, the tool's rigid body transformation, represented as an affine transformation $\mathbf{M}$, is scaled by the decay value and applied to the sample.

The point of departure for the two schemes is the representation of the tool's trajectory. Warp Sculpting decomposes $\mathbf{M}$ into a minimal screw motion and applies decay to the resulting translation vector $\vec{T}$ and rotation angle α (thus $\widetilde{X} = \mathbf{R}(w \cdot \alpha)X + w \cdot T$), while Sweepers directly decays the affine transformation using Alexa's fractional matrix operator $\odot$ (thus $\widetilde{X} = w \odot \mathbf{M}X$). This has implications for the subsequent analysis and overall efficiency of the two techniques.

One way in which the illusion of realistic sculpting breaks down is when an object clings to a tool as it moves away. Gain and Marais [2005] solve this by exploiting the gradient of the distance field, which always points away from the tool surface. They introduce two devices: a gradient toggle, to switch off deformation if no samples lie on the leading edge of the tool, and a gradient swathe, to moderate the decay value depending on whether samples are in front (full decay), behind (zeroed decay) or to the side (interpolated decay) of the tool surface.

Of the two methods Sweepers has the more elegant formulation leading to a simpler implementation. However, unlike Warp Sculpting, its foldover prevention relies on an unproven conjecture. Using a direct implementation of Alexa's matrix operations makes Sweepers about an order of magnitude slower than Warp Sculpting. However, this is explicitly addressed in later work [Angelidis et al. 2006] by using closed forms of the logs of certain simplified matrix transformations, thereby providing a more efficient formulation for single tools tracing out simple paths.

The key limitation of these schemes is that individual surfaces must have a static shape. The authors argue in favor of this by appealing to a sculpting metaphor, in which hard tools are used to shape soft clay. The restriction is partially overcome by support for multiple tools with overlapping regions of effect and independent movement trajectories that can simulate tools as complicated as an articulated hand [Gain and Marais 2005]. These methods are the most efficient of all surface-based deformations, capable of achieving real-time updates of object with tens of thousands of sample point. Warp Sculpting and Sweepers are also two of a handful of schemes that successfully prevent foldover during the compression and expansion of deformable space and hence also in the embedded object (an issue that is more fully discussed in Section 6.4). They do so by breaking large tool movements into a succession of equivalent smaller steps.

Scalar and vector fields have applicability beyond defining the influence of a tool; they can dictate the deformation more directly. In Vector Field Based Deformation (VFBD) [von Funck et al. 2006; 2007] the trajectory of sample points is determined by the path-line integration of a time-based vector field. By construction the vector field is divergence free, which means that a deformed shape cannot self-intersect (since the path lines do not cross) and its volume is preserved (to within the limits of surface sampling).

From the user's standpoint, VFBD supports an impressive variety of interaction styles: bending and twisting around a user-defined axis, pinching and dragging portions of a shape, localized etching of deformations into a surface, and full tool-based sculpting. It is even possible to paint influence regions onto the shape itself [von Funck et al. 2007]. The only significant limitation is that, while translation and rotation are supported, it is not possible to scale a tool, since the resulting vector-field would diverge.

Overall, the benefits of such an approach are considerable: deformations are provably C^1 continuous, foldover free, volume preserving and localized. On the downside, numerical path line integration is processor intensive. Based on results reported by Von Funck et al. [2006] for a CPU implementation on a 2.6GHz processor, it is only possible to interactively deform between 400 and 600 samples. Contrast this with distance-field deformation, which is two orders of magnitude faster. Better performance can be obtained with a GPU implementation since VFBD is "embarrassingly" parallel, but the same applies to most spatial deformation methods.

3.5 Summary

Surface-based tools are on the whole less versatile, especially when it comes to controlling internal deformation density, but more manageable than volume-based tools. In achieving this, there have been some sacrifices in efficiency, especially among triangle mesh methods and vector field deformation. Distance field tools run counter to this trend. Limiting tools to static shapes enables significant preprocessing with consequent acceleration of both the embedding and deformation steps, making this one of the more efficient approaches.

Creating and modifying a surface tool can be burdensome for the user, especially for relatively straightforward deformations. For instance, specifying a simple bend or twist by first creating a control surface and then altering it, is almost a modeling task in its own right, and unnecessary when curve-based deformations are available.

4. CURVE-BASED DEFORMATION

Curve-based deformations (as shown in Figure 3) use axial or parametric curves to manipulate shape.

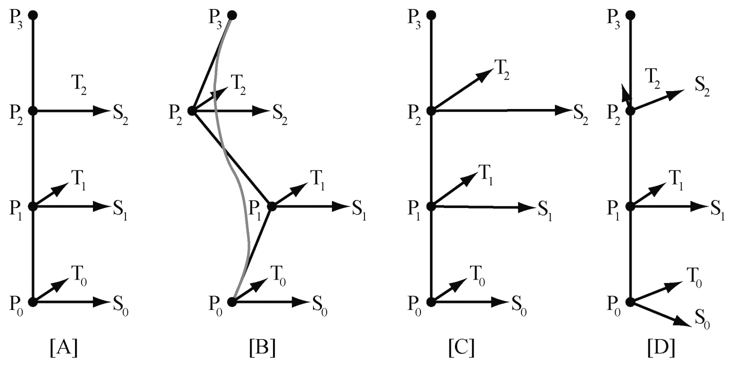

Fig. 4. Generalized de Casteljau Tools. [A] No deformation. [B] S-shaped deformation. [C] Tapering. [D] Twisting.

4.1 Regular Global Deformations

The earliest type of spatial deformation to be extensively adopted in computer graphics were Barr's [1984] regular global deformations. He introduced the stylized deformations: scale, taper, twist and bend, which alter the entire world space in alignment with linear curves (the co-ordinate axes). In addition, the inverse, normal and tangent transformation rules are developed for each flavor of deformation. Barr [1984] intended these deformations to be arranged in a hierarchy that gradually refined the object's shape. The user's involvement is limited to specifying a small set of parameters. For instance, in the case of twisting around the z-axis the user supplies a function $\theta = \mathcal{R}(z)$ that controls the degree of angular twist along z. The function $\mathcal{R}$ then drives the following matrix-based transformation of object points:

$$\mathcal{F}(X) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} X, \ \theta = \mathcal{R}(z). \tag{5}$$

The regular global deformations are, at heart, differential affine transformations. As is obvious from Equation (5), twisting consists of differential rotation. Tapering is essentially differential scaling. Linear bends are somewhat more involved. The unbent regions of the axis are given a rigid body rotation and translation, while the bent section is differentially translated and rotated.

Global coordinate axis deformations such as this are extremely efficient but their utility is limited. The distortions are stylized rather than free-form and centered around a single straight axis (the source curve).

4.2 Generalized de Casteljau Deformation

Chang and Rockwood [1994] introduced a curve-based deformation scheme that can be viewed as a restricted version of Free-Form Deformation. The original coverage of Generalized de Casteljau Deformation is somewhat terse and fuller details are provided by Bechmann and ElKouhen [2001], especially in its application to animation.

An interpretation of the de Casteljau algorithm [Farin 1997] is that it deforms points in the segment [0, 1] (the parametric domain) into points on a Bézier curve. This observation inspired Chang and Rockwood [1994] to extend the interpolation scheme into a fully-fledged deformation algorithm by expanding its domain to points in the unit cube $[0, 1]^3$ of $\Re^3$. In common with other methods the Generalized de Casteljau deformation tool has both an initial and deformed version. In this case, the deformed tool is a unique Bézier curve of degree n defined by $n + 1$ user-specified control points, P_i. In addition, each segment of the Bézier control polygon is given two user-controllable axes, called handles S_i and T_i. Together $P_{i+1} - P_i$, S_i and T_i define a local co-ordinate system associated with each control point P_i (except, obviously, for the last P_n). As with Bézier FFD, the initial tool is implicitly defined. It is a straight-line curve with control points uniformly spaced along [0, 1] of the x-axis and unit-vector handles aligned with the y- and z-axes. Of course, a preliminary affine mapping will allow a parallelepiped domain positioned at an arbitrary origin.

A deformed object is warped by Generalized de Casteljau deformation in correspondence with the Bézier curve but modified by the orientation and the length of the associated handles. Let us illustrate this process with some examples (see Figure 4). If the deformed curve is a line segment along $x = [0, 1]$ with orthogonal unit handles, then no deformation occurs (Figure 4[A]). If an s-shaped curve is defined but the handles remain in their original orientation and of unit length then the deformed object will describe a corresponding s-shape with no tapering or twisting (Figure 4[B]). If the curve remains unchanged but the handles are stretched progressively along the control polygon, then tapering will result (Figure 4[C]). Likewise, if the handles are twisted about the principal axis, then a twist will be introduced in the deformed object (Figure 4[D]). Unlike Regular Global Deformation these effects can be freely combined in a single deformation tool.

Chang and Rockwood [1994] directly generalize the de Casteljau algorithm, but Bechmann and ElKouhen [2001] provide an explanation that is easier to grasp. In order to work on a sample point X in $\Re^3$, the first step consists of building a control polygon of n control points Q_i,

$$Q_i = (1 - u)P_i + uP_{i+1} + vS_i + wT_i, \tag{6}$$

where (u, v, w) are local coordinates of a point X inside the unit cube $[0, 1]^3$.

The subsequent step simply consists of evaluating the Bézier curve of degree $n - 1$, with n control points Q_i at u:

$$\widetilde{X} = \sum_{i=0}^{n-1} \mathcal{N}_i^{n-1}(u) \cdot Q_i. \tag{7}$$

The domain of Generalized de Casteljau deformation is an issue. Deformation is both global and linear in v and w, so any value for these coordinates is valid. Not so with u: if the curve control points P_i are not carefully restricted, discontinuities will be introduced at $u = 0, 1$. As with FFD, this is avoided by a change of basis. Mikita [1996] substitutes de Boor for de Casteljau recurrence, making for curve-based deformations that are both continuous and local along the u-axis (although v and w remain global).

Further extensions are possible. The S handle can be dropped in favor of bivariate u, v control points, replacing the curve tool with a Generalized de Casteljau triangular [Mikita 1996] or rectangular [Bechmann and ElKouhen 2001] patch. Each control point $Q_{i,j}$ now has a single (T) handle, which controls scaling and tilting in a direction normal to the patch, and in this respect serves much the same purpose as the height surface in the Parametric Patch tool of Feng et al. [1996] (see Section 3.1). The final obvious extension is to allow control points in u, v and w, with no handles. In doing so we have re-derived Free-Form Deformation. Indeed, one attraction of Generalized de Casteljau deformation is that it offers a curve (1D), surface (2D) and volume (3D) tool under the same unifying formulation, a fact that is exploited by Bechmann and ElKouhen [2001].

4.3 Axial Deformation

Another scheme that relies on a single controlling curve is Axial Deformation (AxDf), introduced by Lazarus et al. [1994]. AxDf improves on Generalized de Casteljau deformation in two respects: the source curve can assume any shape not just a straight line segment and a zone of influence, set up by the user as a generalized cylinder centered on the curve, can be used to localize the deformation, but in consequence the elegant and easily generalized formalism of de Casteljau is lost.

Under AxDf, a local orthogonal coordinate frame is associated with key points on the controlling curve. To avoid the well-known problems with Frenet frames, AxDf employs a rotation minimizing orthogonal frame [Klok 1986], which can be geometrically constructed using a line segment approximation of the curve.

Two scalar zone of influence parameters (r_{min} and r_{max}) are also linked to points on the curve. The deformation of an object point as a function of distance (d) from a curve point is either full ($d < r_{min}$), attenuated ($r_{min} \leq d \leq r_{max}$) or nonexistent ($d > r_{max}$).

Only a single curve point influences a given object point. A sample point X is attached to the closest point ($\mathcal{S}(t)$) on the curve by storing the coordinates of X relative to the local frame of $\mathcal{S}(t)$. The curve is warped and X is imparted a deformation by projecting from the parametric coordinates in the new local frame at $\widetilde{\mathcal{S}}(t)$, as mitigated by the zone of influence.

To elaborate, the AxDf algorithm has the usual four phases:

(1) The user stipulates the initial shape of the source curve ($\mathcal{S}$) and the zone of influence parameters (r_{min} and r_{max}) at selected intervals. The markers (r_{min} and r_{max}) are interpolated to form inner and outer generalized cylinders which taper the deformation.

(2) The source curve is recursively subdivided into a line segment approximation. The Bishop and Klok coordinate frame and the zone of influence interpolants are then constructed for each line segment. All object sample points (X) are parametrised with respect to the approximated curve by establishing the closest point on the curve $\mathcal{S}(t)$. Then the coordinates (u, v, w) of X within the frame at $\mathcal{S}(t)$ and a weighting factor $r^* \in [0, 1]$ that tails off with distance between X and $\mathcal{S}(t)$ are calculated. A twist factor can also be associated with each line segment.

(3) The curve $\mathcal{S}$ is reshaped by the user, with standard curve-editing techniques, into a deformed curve $\widetilde{\mathcal{S}}$.

(4) The deformation of the curve is transmitted to the object. As before, local coordinate frames are calculated for the deformed curve $\widetilde{\mathcal{S}}$. A sample point X is deformed by projecting from its coordinates (u, v, w), in the new frame at $\widetilde{\mathcal{S}}(t)$, back to world coordinate space as $\widetilde{X}$. One subtlety is that this deformation must be moderated by the zone of influence.

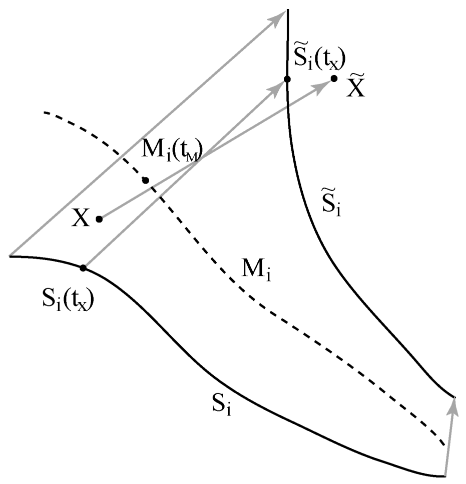

Fig. 5. Wires tools. An initial Wires curve ($\mathcal{S}_i$) with domain ($\mathcal{M}_i$) is displaced by the user to curve $\widetilde{\mathcal{S}}_i$. In so doing the point X, with closest positions $\mathcal{S}_i(t_X)$ and $\mathcal{M}_i(t_M)$ (on $\mathcal{S}_i$ and $\mathcal{M}_i$, respectively), is deformed to position $\widetilde{X}$. Movement during deformation is indicated in grey.

If AxDf is used to define a backbone for figure animation then, for the sake of realism, the curve may twist and flex but should not alter in length. This can be elegantly accomplished by enforcing arc-length-preserving Axial Deformations [Peng et al. 1997].

AxDf allows an arbitrary source curve and limits deformation to a zone of influence but these versatility enhancements are balanced by the disadvantages of closest point parametrization. Apart from a considerable computation burden, it has some continuity implications since adjacent sample point do not necessarily attach to adjacent curve points.

4.4 Wires

The Wires technique [Singh and Fiume 1998] represents the pinnacle of curve-based spatial deformation. It shares the closest Euclidean distance embedding of AxDf but substitutes domain curves for zone of influence cut-off radii and allows multiple interacting control curves. With Wires an object's sample points are parametrised relative to a bundle of n wires. The wires are manipulated and the object's sample points warp accordingly. Singh and Fiume [1998] liken Wires to a sculptor's armature, whereby a wire skeleton guides the surrounding clay.

Each wire i has a host of defining parameters (as shown in Figure 5). $\mathcal{S}_i$ and $\widetilde{\mathcal{S}}_i$ are free-form parametric curves in their embedded and deformed states. The embedding function works by finding the univariate parameter value (t_X) that minimizes the distance between the sample point (X) and the source curve ($\mathcal{S}_i(t)$). A density function $\mathcal{Z}(X, \mathcal{S}) = \Gamma(\|X - \mathcal{S}_i(t_X)\|/r)$, familiar from implicit function and scattered data interpolation literature, ascribes a weighting to the deformation of X according to a radius of influence r. Singh and Fiume [1998] recommend the C^1 function:

$$\Gamma : \Re^+ \mapsto [0, 1], \qquad \Gamma(x) = \begin{cases} (x^2 - 1)^2 & \text{if } 0 < x < 1 \\ 0 & \text{if } x > 1. \end{cases}$$

The radius of influence, r, can be determined by a combination of two methods:

—The user specifies "locators" along the curve. These tie scalars to particular points on the curve and intermediate values at other positions are found by interpolation. Apart from the radius of influence, r, these locators are useful for attaching radial scaling values, s, and rotational twist angles to $\mathcal{S}_i$.

—A domain curve $\mathcal{M}_i$ determines the radius of influence for points according to $r = \|\mathcal{S}_i(t_X) - \mathcal{M}_i(t_M)\|$, where t_M is the parameter value of the point on $\mathcal{M}_i$ closest to X. This radius only applies for points that fall heuristically on the same "side" of $\mathcal{S}_i$ as $\mathcal{M}_i$. Points that lie on the side opposite $\mathcal{M}_i$ are locator controlled. There is some blending between the two methods to ensure continuity.

Wires does away with the local coordinate frames of AxDf. The deformation contribution of a wire $\mathcal{D}_i(X)$ is a function of the difference between $\mathcal{S}_i(t_X)$ and $\widetilde{\mathcal{S}}_i(t_X)$ as modulated by the density weighting function $\mathcal{Z}(X, \mathcal{S}_i)$ and is separated into scaling, rotation and translation components. The radial scaling value (s) contributes uniform scaling about $\mathcal{S}_i(t_X)$. Then the angle θ between the tangent vectors $\mathcal{S}'_i(t_X)$ and $\widetilde{\mathcal{S}}'_i(t_X)$ imparts a corkscrew rotation by $\theta \cdot \mathcal{Z}(X, \mathcal{S}_i)$. Finally, the translation $(\widetilde{\mathcal{S}}_i(t_X) - \mathcal{S}_i(t_X)) \cdot \mathcal{Z}(X, \mathcal{S}_i)$ is added. The separation of components in this fashion allows the user selective control over aspects such as radial scaling and rotational twist along the curve.

The contribution of the individual wires are merged according to:

$$\mathcal{F}(X) = P + \frac{\sum_{i=1}^{n} \mathcal{D}_i(X)\|\mathcal{D}_i(X)\|^m}{\sum_{i=1}^{n} \|\mathcal{D}_i(X)\|^m}.$$

If $m = 0$, the deformation is a simple average of $\mathcal{D}_i(X)$, but as m increases it converges to $max(\mathcal{D}_i(X))$.

Unfortunately, distance parametrizing curve-based tools, such as AxDf and Wires have a fundamental weakness. As a consequence of attaching sample points to the nearest point on a source curve, they have (unless the source is straight) a seam of parametrization discontinuity along the medial axis of a bend, where the closest reference point jumps from one arm of the bend to the other. This can result in "space-tearing" artefacts (C^{-1} discontinuities) where the object buckles or rips along the medial axis.

Of course, the problem of space foldover, of which space tearing is one instance, applies across the range of tool dimensions, but curves suffer most acutely from this weakness. General solutions to the foldover problem will be considered in Section 6.4, but in the context of curves Milliron et al. [2002] correct for it by attaching object samples to multiple points on the source curve, which in the limit becomes an integration along the curve. The mapping from source to target curve is no longer exact and the computation cost is significantly greater, but at least continuity can now be guaranteed.

4.5 Bender

The trend in curve-based deformation has been towards generality: multiple curves with variable radii of influence and sophisticated control over scaling and twisting. In contrast, Llamas et al. [2005] argue for simplicity and efficiency over generality in support of interactive styling. This philosophy is embodied by Bender, in which a user drives deformation by controlling the position and orientation of the endpoints of a stretchable virtual ribbon, using Polhemus trackers held in each hand. Because of the emphasis accorded to computational efficiency, Bender is capable of deforming 70, 000 or more point samples at interactive update rates. As with star-convex tools this is viewed as a complement to, rather than a replacement for, existing schemes.

Bender deformations are derived solely from the initial and final coordinate frames of a curve's endpoints (see Figure 6), as provided by the user who manipulates two 6DOF trackers equipped with buttons to activate deformation. First, a curve is fitted to the endpoint positions and tangents. Rather than use conventional Hermite interpolation Llamas et al. [2005] opt for a biarc, consisting of two circular arcs joined with C^1 continuity. By distributing the twist of the endpoint normals along the chosen biarc, a coordinate frame $\mathcal{C}_s$ can be attached to the biarc at any parametric location $s \in [0, 1]$. A biarc is favored because, in the first instance, if a point X lies closer to the biarc than the radius of curvature then there are at most two local minima in the distance function and, in the second instance, the s-parameter value of these minima may be calculated very cheaply. From the closest parameter values, starting and ending coordinate frames ($\mathcal{C}_x$ and $\widetilde{\mathcal{C}}_x$) can be obtained. A rigid-body screw motion that interpolates these frames is used to deform the sample point X, with a simple radial weighting function that defines a tubular region of influence around the initial curve used to scale the degree of deformation with distance. A noteworthy aspect of Bender is that the deformation provided by the two arms of the biarc (the two local closest point minima) is blended, thereby preventing the seam of parametrization discontinuity and subsequent space-tearing that is prevalent in Wires.

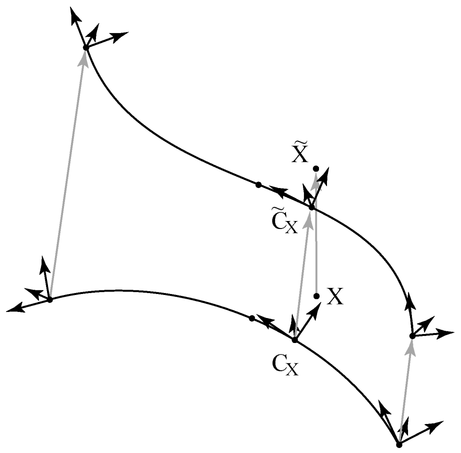

Fig. 6. Bender tools. Frames at either end of an initial tool are interpolated to find a frame $\mathcal{C}_x$ at the position closest to the sample point (X). The user alters the endpoint frames and $\mathcal{C}_x$ becomes $\widetilde{\mathcal{C}}_x$, thereby deforming X to $\widetilde{X}$. Deformation movement is indicated in grey.

Within the context of a single curve tool and a two-handed user interaction style the spatial deformation solution provided by Bender has the virtues of efficiency, usability and an appealing simplicity.

4.6 Skinning Frameworks

A variant of curve-based spatial deformation, character skinning, is widely applied in the field of figure animation. In this domain it is standard to animate the surface of a model (the skin) by attaching individual vertices to an underlying stick-figure skeleton (the rig). From a user perspective, the rig is composed of bone segments linked at joints. From a mathematical perspective, this is merely a hierarchy of coordinate frames. Strictly speaking, skinning frameworks are thus an instance of point- (and frame-) based rather than curve-based deformation, but in placing them here we uphold the user's view.

In general, skinning frameworks deal with the problem of how to attach vertices to the underlying skeleton in such a way that

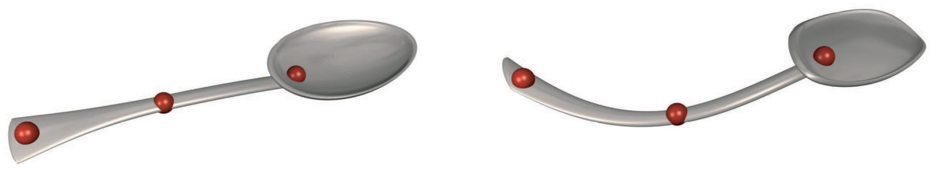

Fig. 7. Point-based deformation. [left] Before deformation. [right] After deformation.

physically plausible deformation results when the skeleton is repositioned. A simple and efficient approach is embodied by Skeletal Subspace Deformation (SSD), which takes a vertex in a canonical rest pose (X), embeds it in the local frame of each bone in its rest configuration ($\mathcal{E}_i(X) = \mathbf{G}_i^{-1}X$) and then transforms from these local spaces using a new articulation of the bone frames ($\mathcal{D}_i(U) = \widetilde{\mathbf{G}}_i U$). These per bone transformations are combined using an affine weight for each vertex-bone pairing (w_i):

$$\widetilde{X} = \sum_i w_i \widetilde{\mathbf{G}}_i \mathbf{G}_i^{-1} X.$$

Vertex transformations are rigid with respect to individual bones but their linear combination is not. As a consequence SSD has some well-publicized flaws. Unrealistic volume loss can occur at joints—the picturesquely named "collapsing elbow" and "candy wrapper" effects.

The shortcomings of SSD blending are generally overcome in one of three ways: by training weights over a set of examples [Sloan et al. 2001], by increasing the number of parameters associated with a vertex to bone pairing [Wang and Phillips 2002; Merry et al. 2006] or by redefining the nature of the underlying skeleton (such as with quaternions [Kavan et al. 2007] or by introducing extra intermediate joints [Mohr and Gleicher 2003]).

This is a rich vein of research, deserving of a survey in its own right, but there are lessons here for other deformation techniques, particularly those that seek to combine the influence of multiple manipulators. With Distance Field Tools and Wires, pinching artifacts are somewhat disguised by the nonlinearity of the deformation but they can still be noticeable and objectionable.

4.7 Summary

Curve-based deformations are most suited to twists, bends, and tapers about an axis, introducing ridge and valley features on a surface, or deforming an object along a skeleton curve. Volumetric and surface tools should take precedence for more general deformation tasks.

One often overlooked benefit of using curves as deformation tools is their sheer ubiquity. Designers generally have a thorough practical understanding of the properties and behavior of curves and this enhances their usability. The usability benefits of curves are well recognized by the CAD community, who make significant use of profile curves and cross sections. In fact, curve-based deformation has been used explicitly for feature definition in CAD [Guillet and Léon 1998; Pernot et al. 2002].

On the downside, deformation schemes which support fully general source curves (rather than simple axes) run the danger of space tearing. While this can be overcome, current solutions tend to be either overly expensive (Milliron's Wires [Milliron et al. 2002]) or restrictive (Bender [Llamas et al. 2005]).

5. POINT-BASED DEFORMATION

Point-based deformations (as shown in Figure 7) use freely positioned points to manipulate shape.

5.1 Constraint-Based Deformation (DOGME)

Shortly after FFD [Sederberg and Parry 1986] was first promulgated, Borrel and Bechmann [1991] developed a constraint-based method called DOGME (Deformation of Geometric Models Editor) that replaces the awkward lattice user-interface of FFD with direct point manipulation. In a similar process to the evolution of FFD, DOGME has inspired many variants and extensions [Bechmann and Dubreuil 1993; Borrel and Rappoport 1994; Bechmann and Dubreuil 1995; Aubert and Bechmann 1997b; Raffin et al. 2000; Bechmann and Gerber 2003].

The intention is to enable a user to drag object points and have the surrounding surface conform smoothly. For instance, pushing or pulling a single object point will create either a dimple or a mound in the object's surface, and more complex manipulation can be achieved by simultaneously moving several points.

The deformation of an object is defined by the displacement of points called constraints. In particular, a constraint can be applied to a sample point on an object. It then becomes trivial to achieve exact displacement of sample points. The model can exactly satisfy an arbitrary number of constraints, unless two conflicting constraints are applied to the same unique point, in which case a best approximation is calculated. For points lying within the regions of influence of several constraints, a weighted combination of each constraint is applied.

Point-based deformation can be broken down into two operations:

(1) *The constraints are created.* The user selects a set of r points without restriction ($\mathcal{C}_i$) and provides a displacement ($\Delta\mathcal{C}_i$) and region of influence for each point (thereby defining $\mathcal{F}$). The displacement paths can be linear or curvilinear [Raffin et al. 2000; Bechmann and Gerber 2003].

(2) *The object is deformed.* DOGME is characterized by the deformation function [Borrel and Bechmann 1991]:

$$\mathcal{D}(X) = X + \mathbf{M} \cdot \mathcal{F}(X) = \widetilde{X}, \tag{8}$$

where $\mathcal{F}$ is a function ($\mathcal{F} : \Re^n \mapsto \Re^m$) which specifies how the deformation density tails off around each constraint and $\mathbf{M}$ is a matrix of unknowns, of size $n \times m$, whose elements are instantiated so as to satisfy the user-defined constraints ($\Delta\mathcal{C}_i = \mathbf{M} \cdot \mathcal{F}(\mathcal{C}_i), \forall i = 1, \dots, r$). To solve for the matrix of unknowns ($\mathbf{M}$), standard numerical methods are available if the system is square and of full rank, or the pseudo-inverse if it is not [Boullion and Odell 1971; Greville 1960]. Once the matrix $\mathbf{M}$ has been resolved, all sample points are successively embedded and deformed using Equation (8).

To obtain local deformation, B-splines functions are used in the formulation of $\mathcal{F}$, but any other function having a limited range could be used. When DOGME is utilized for local deformations, the user can choose between regions of constraint influence that are spherical (with simple form B-splines), cuboid (with simple product form B-splines), polyhedral (with multiple axes in the simple product form), or even generalized voxel volumes [Bechmann and Gerber 2003]. The extent of regions are controlled by a radius linked to the range of the appropriate B-spline function. The shape of the deformation within these regions is, in cross-section, an image of the B-spline but any function with a limited range could be substituted if this is considered undesirable. By using polynomial functions with unlimited range, DOGME can also achieve global deformations such as affine and nonaffine transformations.

To obtain regions of influence, we introduce DOGME in its full generality [Bechmann and Gerber 2003], where $\mathcal{F}$ is surrounded by two parametrizing functions $\mathcal{H}$ and $\mathcal{G}$, such that:

$$\mathcal{D}(X) = X + \mathbf{M} \cdot \mathcal{H}(\mathcal{F}(\mathcal{G}(X))) = \widetilde{X}. \qquad (9)$$

Where, for instance, $\mathcal{G}$ could map a point to the local frame of each constraint [Bechmann and Dubreuil 1995]) and $\mathcal{H}$ could define curvilinear instead of linear displacements [Bechmann and Gerber 2003].

To simplify subsequent discussion, we assume deformations in $\Re^3$ ($n = 3$) and m equivalent to the number of constraints, unless otherwise mentioned. This model could be instantiated in a variety of ways, where the parametrizing function $\mathcal{F}$ takes one of three broad forms:

—*Simple Form*. In its simplest form $\mathcal{F}$ transforms points from $\Re^3$ to $\Re^m$:

$$\mathcal{F}(X) = (f_1(X), f_2(X), \ldots, f_i(X), \ldots, f_m(X))^t. \qquad (10)$$

This offers the greatest simplicity and efficiency of all point-based spatial deformations. A single function $f_i(X) : \Re^3 \mapsto \Re$ is associated with each constraint and it quantifies the influence of that constraint on the point X. In this approach, deformations are determined by an arbitrary number of constraint points with an associated displacement. When f_i is a B-spline function, each region of influence is a sphere centered on a constraint point and controlled through its radius of influence. The result when applied to an object is a collection of smooth, possibly overlapping bumps. The Simple Dogme Form is sometimes known as a radial deformation because sample points are parametrised solely by their distance from the constraint points and deformations thus radiate uniformly in all directions. This class of deformation will be covered in more detail in Section 5.4.

—*Simple Product Form*. Here, each component of $\mathcal{F}$ is expressed as a simple product. So, for three axes: $f_i(X) = f_i^1(x) f_i^2(y) f_i^3(z)$. The ith constraint now has a_i axes each with an associated function and its range. At its simplest, when f_i^j is a B-spline function, each region of influence is a cuboid centered on a constraint point and controlled through three radii of influence. By imposing (using $\mathcal{G}$) a system of local coordinates using three orthogonal axes with range markings at each constraint, the region of influence remains cuboidal, but does not have to be parallel to the global axis. With the introduction of additional nonorthogonal axes, the boundary of the region of influence around a constraint may become as complex as a convex prismatic polyhedron [Bechmann and Dubreuil 1993; 1995] still controlled through a radius of influence. Understandably, this is more computationally costly than the simple form, but it remains applicable to an interactive setting.

—*Tensor Product Form*. In this category, deformation is particularized along each world coordinate axis according to a set of functions: $\mathcal{F}(X) = f^1(x) \otimes f^2(y) \otimes f^3(z)$ (where $\otimes$ denotes the tensor product operator). The extent of deformation relates directly to the range of the component functions. For instance, B-spline functions provide localized deformations with a cuboid region of influence just as with the simple product form. Sometimes, the tensor product form is underconstrained (when m in Equation (8) is greater than the number of constraints), allowing an infinity of solutions that satisfy the constraints. This can be exploited in pseudo-inverse solutions [Boullion and Odell 1971; Greville 1960] to introduce optimization criteria such as minimum volume variation [Aubert and Bechmann 1997b], which allows deformations that preserve volume while still obeying the user's constraints. Understandably, this is considerably more computationally costly than the simple product form. With a particular choice of functions, tensor product DOGME can be made equivalent, as proven by Bechmann [1994], to Directly Manipulated FFD [Hsu et al. 1992] (more on which in the next section). Thus, if desired, DOGME can mimic the behavior, if not the control mechanism, of FFD.

Recently there has been some cross-pollination of DOGME with other manipulators. So, points can be made to traverse curvilinear paths both in the simple [Raffin et al. 2000] and product [Bechmann and Gerber 2003] forms. In addition, DOGME can accommodate curve-based [Bechmann and Gerber 2003] and, potentially surface-based manipulators within the same formalism.

The simple and simple product forms of DOGME are among the most efficient of all deformation schemes and a good choice if real-time response is an overriding requirement. This is less true of the tensor product form, especially with the addition of optimization criteria such as minimum volume variation [Aubert and Bechmann 1997b].

Constraint-based (DOGME) deformation offers a useful compromise among ease of use, versatility, and efficiency. A particular strength is that it enables fine control over the boundary and internal shape of the regions of influence surrounding constraints without suffering the attendant overheads of schemes such as FFD with non-cuboid lattices. In addition, DOGME holds promise for combining the different dimensions of manipulators (points, curves, surfaces and volumes) into one formalism.

5.2 Directly Manipulated FFD

In conventional Free-Form Deformation the alteration in sample points is dictated by the position of lattice control points. By contrast Directly Manipulated Free-Form Deformation (DMFFD) [Hsu et al. 1992] determines the lattice configuration from a selection of constraint points and their movement. To carry out this principle DMFFD is broken into three steps: the user provides a number of constraints, each composed of a point, C_i, and its intended motion, ΔC_i; the lattice control points, $P_{i,j,k}$ (in Equation (2)), are altered, $\Delta P_{i,j,k}$, to meet these constraints; and this new lattice is applied through standard FFD to the original object.

To achieve this, the r constraints are bound into a system of linear equations by applying Equation (2) and recasting the result in matrix notation:

$$\Delta \mathbf{C} = \mathbf{B} \Delta \mathbf{P}. \qquad (11)$$

Each row of Equation (11) is an unrolling of Equation (2) into matrix form. $\Delta\mathbf{C}$ is an $r \times 3$ matrix holding the constraint displacement vectors, $\Delta\mathbf{P}$ is an unknown $s \times 3$ matrix capturing the change in lattice control points, and $\mathbf{B}$ is an $r \times s$ matrix of tensor product basis functions evaluated at the constraint points. As with DOGME, Equation (11) can be solved using the pseudo-inverse, although for efficiency's sake the sparsity of the basis matrix ($\mathbf{B}$) should be exploited [Gain 2000]. Hu et al. [2001] present a simpler and more direct approach. Approximating behavior occurs when the system is overconstrained (too many constraints act on the same control point). Lee et al. [1995] work around this by recursively inserting additional control points.

The real issue with DMFFD is that, while the computation of the deformed hyperpatch is transparent to the user, the properties of the resulting deformation are strongly linked to it. One consequence is that the region of influence of constraint points cannot be varied independently; they are implicitly dependent on the initial lattice (P). It is a simple task to automatically generate a hyperpatch for a given single radius of influence. A hyperpatch with compact cells will contain the constrained deformation within a small volume, while expansive cells will induce correspondingly wide-ranging deformations. However, finer control over the boundary shape and internal density of a constraint's influence volume would require editing the initial lattice into a noncuboid configuration, thereby forfeiting much of the efficiency and ease-of-use of DMFFD.

By manipulating Equation (11) it is possible to introduce other types of constraints. For instance, by taking partial derivatives with respect to the hyperpatch axes it is possible to manipulate the derivative frame at constraint points [Gain and Dodgson 1999b]. A variety of effects can be induced by affine transformations of these derivative frames, including tilting, twisting, and swelling in the object undergoing deformation. This is similar in spirit to locators on Wires.

The constraint mechanism of DMFFD can also be extended to encompass curves [Gain 2000] through a process of functional composition and degree reduction. Curve manipulation requires several additional phases to derive a system of linear constraints relating changes in a curve's control points to an FFD lattice. First, a source Bézier curve is segmented at cell boundaries in an undistorted hyperpatch. Secondly, the resulting curve fragments are fed to a functional composition process, which binds the embedded curve control points to the hyperpatch through a set of weighting coefficients. Thirdly, since the enormous degree elevation inherent in composition [Nimscheck 1995] is problematic, the weighting coefficients undergo a series of repeated degree reductions, which introduces a bounded approximation error. The end result is a complete system of constraints which DMFFD can process as before. By avoiding Euclidian distance parametrization, curve-directed DMFFD enables interactive update rates over the entire deformation cycle and the automatic detection and prevention of self-intersection (discussed further in Section 6.4) at the expense of absolute accuracy in tracking a curve.

DMFFD is unique in providing a synthesis of $0D$, $1D$, and $3D$ manipulators, allowing a seamless combination of point, curve, derivative-frame and lattice manipulation within a single deformation. Unfortunately, due to the limitations in functional composition [Nimscheck 1995] this cannot be extended to surface-based tools. The biggest shortcoming of DMFFD is its lack of independent control over the influence region of constraints. Nevertheless, DMFFD is also one of the more computationally inexpensive deformation methods applicable where region-of-influence control is not an overriding concern.

5.3 Dirichlet FFD

The idea behind Dirichlet FFD [Moccozet and Magnenat-Thalmann 1997] is to automatically generate regions of influence and internal spatial density from a set of constraint points without requiring any user intervention. A Delaunay triangulation is used to create an underlying volumetric structure for the deformation. In this sense Dirichlet FFD bridges volumetric and point-based manipulation: the control mechanism is a set of points and their displacement but the underlying foundation is a cluster of Bézier simplex volumes determined by a Delaunay triangulation. The Bézier simplices are an extension of Farin's [1990] use of Dirichlet surfaces, which explains the naming of this technique. In the discussion that follows we assume a familiarity with Delaunay triangulations and Voronoi diagrams [Boissonnat 1984].

The process of deformation can be subdivided into the conventional four steps:

(1) *The deformation tool is created.* The user places an initial set of control points, which will be approximated (or possibly interpolated) during deformation. The convex hull of this set demarcates the overall bounds of the deformation. A Voronoi diagram and the associated Delaunay triangulation are computed over the control points.

(2) *The object is linked to the deformation tool.* For each sample point X, the Voronoi diagram and the associated Delaunay triangulation are (locally) recomputed with X inserted. By overlapping the original and refined triangulations the natural neighbors and ultimately the natural (Sibson) coordinates (U) of a sample can be determined.
Next, a multivariate Bézier simplex [Farin 1990] is constructed with the natural neighbors of X serving as Bézier endpoints. Depending on the chosen degree (and hence continuity) of the associated Bernstein polynomials, other internal control points will be introduced by a process of degree elevation.
A sample point X within the associated multivariate Bézier simplex can be represented as a sum of control points, P, weighted by the polynomial Bernstein functions, $\mathcal{B}_I^m$, with multi-index I and degree m, as follows:
$$\mathcal{H}(U) = \sum_{I=m} \mathcal{B}_I^m \cdot P_I = X, \tag{12}$$
where U are the Sibson coordinates of X.

(3) *The deformation tool is modified.* The user moves control points ($P \Rightarrow \widetilde{P}$), resulting in a distortion of the attendant Bézier simplices ($\mathcal{H} \Rightarrow \widetilde{\mathcal{H}}$). It is worth noting that only the initial set of control points (which define simplex endpoints) are accessible to the user. Moccozet [1996] provides rules for how the interior control points are modified. As with Rational FFD (Section 2.2), weights can be associated with control points of the multivariate Bézier simplex if desired. Finally, for continuity reasons, control points on the convex hull should remain fixed.

(4) *The object is deformed.* For any sample point X, its Sibson coordinates over the associated multivariate Bézier simplex remains invariant under deformation. This allows deformed world space coordinates to be obtained for all sample points by repeatedly applying Equation (12) ($\widetilde{X} = \widetilde{\mathcal{H}}(U)$).

The region of influence associated with a control point in the initial set is the union of its incident Delaunay spheres in the initial Delaunay triangulation. These regions are overlapping, thereby ensuring smooth transmission of the deformation between adjacent simplices. This has the effect of expanding or contracting the region of influence according to point density, automatically enabling a seamless combination of fine and broad scale deformations. On the other hand, it might be difficult for a user to predict this region during control point configuration and this detracts from its usability. The actual shape of the deformation is determined by the degree of the underlying multivariate Bézier simplex.

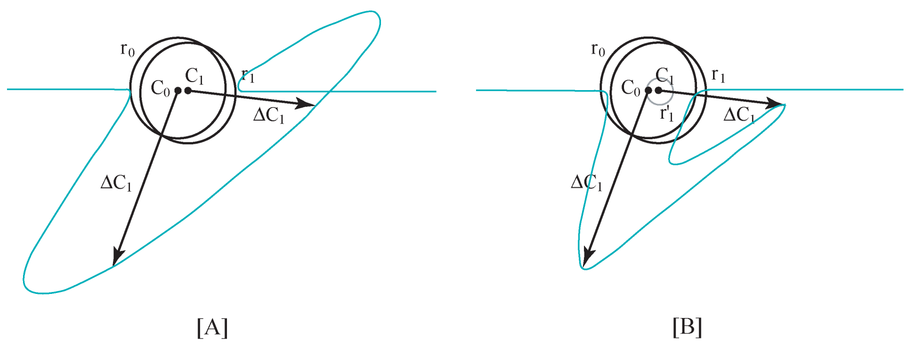

Fig. 8. Scodef tools. [A] Two nearly coincident constraints (with positions, C, displacements, ΔC, and radii of effect, r) can cause space tearing. [B]. Space tearing is avoided by introducing a radius of influence (r').

A real stumbling block with Dirichlet FFD is the computation overhead: Delaunay triangulations, Sibson coordinate evaluations and Multivariate Bézier simplices do not come cheap. For this reason Moccozet and Magnenat-Thalmann [1997] rightly restrict their application of Dirichlet FFD to off-line animation.

Dirichlet FFD does have some obvious virtues in terms of versatility. Unlike Directly Manipulated FFD, it offers variable regions of influence around constraints but these are an implicit consequence of control point configuration, which can hamper ease of use.

5.4 Simple Radial Deformation

Radial deformations offer the greatest simplicity and efficiency of all the point-based spatial deformations. In these schemes deformations are determined by an arbitrary number of constraints, each consisting of a spherical radius of effect (r_i) centered on a constraint point (C_i) with an associated displacement (ΔC_i). The result when applied to an object is a collection of smooth, possibly overlapping bumps. The radial deformations are so named because sample points are parametrised solely by their distance from the constraint points and deformations thus radiate uniformly in all directions.

Borrel and Rappoport [1994] developed Simple Constrained Deformation (*Scodef*), an early instance of the radial deformation technique. Scodef has the following useful properties:

—Unlike some other radial methods, the deformation effect is local. The deformation is bounded by a union of the spheres of radius r_i each centered at C_i.

—All constraint parameters (C_i, ΔC_i, r_i) are completely independent. In contrast, DMFFD has regions of influence bound to the lattice rather than individual constraints.

—The deformation image is that of a scaled B-spline. It is this B-spline basis which principally differentiates Scodef from other radial deformation techniques.

Another of Borrel and Rappoport's contributions is to identify a "space-tearing" phenomenon (as illustrated in Figure 8) which occurs when two constraints with different displacements approach singularity (the distance between them is significantly less than their radii). This causes certain regions of space in the vicinity of these constraints to overshoot the constraint displacements and may result in the self-intersection of an affected object.

Space-tearing is allayed by introducing redundancy in the form of duplicated constraints. Now, two radii are associated with each constraint. The larger radius demarcates an area of effect, while the smaller determines the degree of influence on neighboring constraints.

Inspired by Scodef, Crespin [1999] proposes a deformation model that combines the influence of multiple constraints using blending techniques from the implicit function literature. Each constraint consists of a point and associated frame, to add twist and tilt to the translations inherent in Scodef. Furthermore, Crespin's method supports both global and local modes of deformation.

Ruprecht et al. [1995] adapt advances in scattered data interpolation to radial deformation. They recommend a distance-measuring parametrization and a weighting function based on Hardy's multiquadrics. They also propose setting the radii of influence automatically as the distance to the nearest neighboring constraint point. In this respect it has something of the flavor of Dirichlet FFD.

It is worth noting that the above schemes are actually instantiations of simple form Dogme. For instance, setting f_i to a B-spline or Hardy multiquadric in Equation (10) will produce, respectively, SCODEF and Scattered Data Deformation.

Starting from the same premise as Bender, that is, two-handed deformation controlled interactively by 6 degree-of-freedom magnetic trackers, Llamas et al. [2003] develop Twister. They employ the two positioned coordinate frames to directly generate point and rotation constraints rather than interpolating a virtual ribbon between them. There are several implementation choices that differentiate Twister from other radial schemes: it is limited to at most two interacting constraints; it adds coordinate frame manipulation and the attendant twist effects on top of conventional point manipulation (in a similar manner to derivative-frame DMFFD); it squashes one side of each sphere of influence as constraints approach each other. This is quite different from the radius adjustment supported by Scattered Data Deformation since it actually alters the shape of the spheres of influence and does so only in the region between the conflicting constraints.

The main advantage of radial deformations are their sheer simplicity and consequent efficiency. In fact, Botsch and Kobbelt [2005] demonstrate that it is possible to program radial deformations on the GPU with consequent performance benefits. As will be shown in the evaluation it compares very favorably to other point-based spatial deformation techniques, especially if all constraints are disjoint. The area of effect can vary between constraints in radial deformations, unlike DMFFD which is dependent on a uniformly subdivided lattice substrate. On the other hand DMFFD allows different subdivisions of the lattice (and thus independent boundary control) along different axes.

5.5 Summary

Even though point-based deformation is of lower dimensionality than curve-based deformation, it paradoxically offers greater freedom because points are no longer limited to a one-dimensional strand-like relationship. Point-based deformation also offers a gamut of tradeoffs between efficiency and versatility. At one extreme, Simple Form Dogme or Simple Radial deformations are highly efficient but with limited control over the region of influence around constraints and, at the other extreme, Simple Product or Tensor Product Dogme offers individualized control over the density and shape of influence regions but with an attendant processing cost.

6. EVALUATION

6.1 Versatility

The developments reviewed in this article have steadily augmented the expressive power of spatial deformation, in terms of both the variety of possible deformations and their means of control. The improvements in versatility have been fivefold:

Locality. Certain schemes, based on Bernstein polynomials, affect the whole span of world coordinate space and are only suited to warping entire objects. Fortunately, the majority allow locally bounded deformations and the flexibility to design variable-scale features. Key to this enhancement is the enforcement of continuity across the deformation boundary. This enables a seamless blending of the distorted and undistorted areas of an object.

Precise Displacements. One deficiency of hyperpatch-based deformations is the lack of exact control over individual object vertices. The precise displacement of a given point in the hyperpatch is, without a deep insight into the underlying mathematics, arduous in the extreme. This difficulty is overcome partially by surface- and curve-based techniques and completely by point-based deformations. They can be further ranked according to whether multiple independent constraints are supported.

Boundary Control. Many techniques focus on extending the range of permissible boundary configurations. The most limiting shape is a cuboid whose aspect ratio is governed by the number and spread of control points along orthogonal axes. The pinnacle of generality in this respect is attained by FFD with lattices of arbitrary topology and Dogme with general voxel volumes of influence [Bechmann and Gerber 2003].

Combining Manipulators. There has been some early research into allowing users to combine manipulators of varying dimension when specifying a single deformation. Generalized de Casteljau Deformation [Mikita 1996; Bechmann and ElKouhen 2001] can configure its single manipulator into curve, surface or volumetric forms, and both DOGME [Bechmann and Gerber 2003] and DMFFD [Gain 2000] allow multiple simultaneous point and curve constraints. However, there is as yet no deformation framework that supports an unconstrained combination of manipulators of any dimension.

Volume Conservation. Inelastic substances, such as modeling clay and putty, generally retain the same volume as they are manipulated. This has ease-of-use implications since a user new to spatial deformation, reasoning by physical analogy, will probably expect such behavior. To mimic this effect Aubert and Bechmann [1997b], and subsequently Hirota et al. [1999] and Angelidis et al. [2004], devise volume-preserving deformation tools. Shape Merging [Decaudin 1996] incorporates an early version of volume conservation that works when the manipulator is placed outside the object, while prior work by Rappoport et al. [1995] is specific to tensor-solids.

This is a challenging problem, which explains the paucity of research results in this area. In general, there are two solution strategies. The first, as exemplified by Aubert and Bechmann [1997b] and Hirota et al. [1999], is to optimize the free parameters of a deformation to approach volume conservation, while the second [Decaudin 1996; Angelidis et al. 2004; von Funck et al. 2006] builds volume conservation directly into the deformation model.

Topology Alteration. Spatial deformations are topology preserving since the number and relationship of edges, faces and holes of an embedded solid remain constant. This precludes a sphere (genus 0) being transformed into a coffee mug (genus 1). However, certain topology alterations are highly desirable from a user perspective. Aubert and Bechmann [1997a] achieve topology changes by extruding a three-dimensional object into $\Re^4$, deforming the resulting space-time construct, and then forming an $\Re^3$ cross-section of constant time. Even though the actual spatial deformation in $\Re^4$ remains topology preserving, the same does not apply to the $\Re^3$ cross sections, which may develop holes and split into separate components. An alternative to this is Discontinuous Free-Form Deformation [Schein and Elber 2004] which introduces C^{-1} discontinuities by inserting duplicate knots into an FFD lattice, thereby creating space tearing along fault planes in the hyperpatch. There is a link between foldovers and topology alteration. In a mathematical sense, both of them break the homeomorphism of a deformation. One strategy for ensuring realism would be to replace physics-defying self-intersections with topology alteration, but this has yet to be attempted. Rather (as discussed in Section 6.4), self-intersection is usually prevented entirely.

The relationship between the different spatial deformation methods and these versatility enhancements are catalogued in Table I.

Versatility subsumes aspects of both deformation power and range. The question of deformation "power" asks: what features make deformations more expressive by reducing the time and effort required to achieve a given shape? The question of deformation "range" is equally important: what features expand the variety of shapes attainable under deformation? The features of locality and topology alteration extend the range of deformations; without locality a deformation must always affect the entire object and without topology alteration the genus of an object can never be altered. Other features such as precise displacements, boundary control, combining manipulators and volume conservation improve power but not range because their effects can be replicated by less versatile deformations given enough time and effort.

Versatility is not without its price. The study of efficiency will demonstrate that the most powerful schemes, DOGME and Wires, are also among the least efficient. Nevertheless, they remain applicable in an interactive setting.

6.2 Ease of Use

An intuitive and uncomplicated means of user interaction is crucial to effective deformation but it cannot be conceived as a cosmetic afterthought. Rather, as has been amply demonstrated in this survey, the interaction mechanism is fundamentally interwoven with the construction of a spatial deformation technique. A couple of

Table I. A Comparison of the Versatility of Spatial Deformation Methods

Under Manipulator Type: V, S, C, P stand for volume, surface, curve and point-based deformations. Multiple Tools refers to whether more than a single manipulator of the same type can be applied simultaneously. The Conserve Volume and Foldover Free columns provide references for those techniques with variants that preserve object volume or prevent self-intersections, respectively. [H99] = [Hirota et al. 1999], [D96] = [Decaudin 1996], [AB97b] = [Aubert and Bechmann 1997b], [A+04a] = [Angelidis et al. 2004], [A+04b] = [Angelidis et al. 2004], [GD01] = [Gain and Dodgson 2001], [GM05] = [Gain and Marais 2005], [MW01] = [Mason and Wyvill 2001], [vF+06] = [von Funck et al. 2006].

Method	Manip. Type	Local Def.	Precise Displace.	Multiple Tools	Conserve Volume	Foldover Free	Boundary Shape
Bézier FFD	V	no	no	no			cuboid
FFD with local control	V	yes	no	no	[H99]	[GD01]	cuboid
FFD with noncuboid lattices	V	yes	no	no			free-form
Star-Convex tools	S	no	no	no	[D96]		none
Parametric Patch tools	S	no	no	no			rectangle in XZ unbounded in Y
Triangle Mesh Tools	S	yes	yes	yes			union of spheres
Field-based Tools	S	yes	no	yes	[vF+06]	[A+04a] [GM05] [vF+06]	free-form
Regular Global	C	no	no	no			none
Generalized de Casteljau	C | S | V	no	no	no			bounded on some axes
Axial	C	yes	no	no			generalized cylinder
Wires	C	yes	no	yes			generalized cylinder
Constraint-based (Product Dogme)	P & C	yes	yes	yes	[AB97b]		cuboid, convex polyhedron or voxel volume
DMFFD	P & C	yes	yes	yes		[GD01]	cuboid
Simple Radial (Simple Form Dogme)	P & C	yes	yes	yes	[A+04b]	[A+04b] [MW01]	sphere
Dirichlet FFD	P	yes	yes	yes			convex polyhedron

advancements in ease of use have been inspired by the tenets of human-computer interaction.

Direct Manipulation. Schneiderman [1983] has experimentally proven that direct operation on objects is preferable to indirect access. The control points endemic to hyperpatch-based methods offer only an indirect approximating mode of interaction. Surface- and Curve-based deformation are an improvement but it is only point-based deformations that truly implement direct manipulation. A further benefit of direct manipulation is the similarity with physical sculpting. Directing the motion of points on an object's surface can be compared to prodding and pulling at clay with a set of metal pins. Even so, the considerable potential of direct manipulation has barely been touched.

Reduced Degrees of Freedom. A proliferation in the degrees of control freedom, even though it may increase versatility, detracts from conceptual clarity and ease of use. To improve matters, Schneiderman [1983] recommends enabling the graceful expansion of a user's knowledge of the deformation system by supporting a layered approach. The user can be shielded from unnecessary complexity by setting sensible defaults and establishing a hierarchy of increasingly sophisticated control tools. Table II lists the typical degrees of freedom for different deformation schemes.

However, this vein of research is still viable. Rossignac [1994] highlights parallels between sketching programs and interactive sculpting. Two-dimensional design is a mature field, and it could be worthwhile transferring some of its successful elements to three-dimensional sculpting. In particular, an analytic "undo" facility would be a useful aid to productivity. Currently, reversing a series of mistakes requires the memory intensive storage of object images captured at every stage. Also "cut," "copy," and "paste" operations that act topologically to separate, duplicate, and merge object features would be worth implementing. Further inspiration can be derived from the behavior of actual modeling clay [Angelidis et al. 2004; Gain and Marais 2005].

In fact, such sculpting tools (encompassing Sweepers, Warp Sculpting, Bender, and Twister) represent a significant departure in their interaction style. Conventional deformations presume a measured CAD approach, where a complex manipulator with many degrees of freedom is carefully configured to produce a single desired deformation. In contrast, a sculpting approach builds complex shape changes from the successive application of simple manipulators, often interactively controlled by a user's gestures as tracked by 6DOF input devices.

These tools appeal to our everyday familiarity with physical sculpting and as a result they can be easier to learn and more intuitive to predict. However, they tend not to integrate as seamlessly with existing modeling and animation pipelines. Furthermore, where deformations overlap, the outcome of applying tools in series or in parallel can be markedly different. Since sculpting tools offer only the former option, they can produce only a subset of the

Table II. Degrees of Control Freedom for Different Spatial Deformation Methods
V, S, C, P stand for volume, surface, curve, and point based deformations. In many cases the DOF are doubled because the user can control both the initial and final form of a manipulator.

Method	Manip. Type	DOF	Details of Derivation
Bézier FFD	V	201	9 (initial lattice) $+3$ (pos) $\times 64$(CP of tricubic lattice)
FFD with local control	V	201	as above
FFD with noncuboid	V	384	$2 \times 3 \times 64$ (initial & final CP specified)
Star-Convex tools	S	14	$3 + 3 + 1$ (position, orientation and scale) $\times 2$
Parametric Patch tools	S	57	6 (initial rectangle) $+3$ (pos) $\times 16$ (CP of bicubic surface)
Triangle Mesh Tools	S	$18T$	3×3 (pos of vertices) $\times T$ (triangles) $\times 2$
Field-based Tools	S	13	1 (radius of effect) $+6$ (coordinate frame) $\times 2$
Regular Global	C	4	4 (coefficients of cubic input fn.)
Generalized de Casteljau	C | S | V	39	9 (initial frame) + 3 (pos) $\times$(4 (CP) $+6$ (S, T handles))
Axial	C	$24 + 2M$	3 (pos) $\times 4$ (CP of cubic curve) $\times 2$ $+M$ (influence markers) $\times 2$ (inner & outer)
Wires	C	$(48 + 2L)C$	C (wires) $\times$(2 (domain curves) $\times 3$ (pos) $\times 4$ (CP of cubic curve) $\times 2 + L$ (locators) $\times 2$)
Constraint-based (Simple Product Dogme)	P & C	$9P$	P (points) $\times$(3 (pos) $\times 2 + 3$ (cuboid region))
DMFFD	P & C	$12 + 6P$	3 (pos) + 3 (orient) + 3 (length of axes) + 3 (num cells) + 3 (pos) $\times P$ (points) $\times 2$
Simple Radial (Simple Form Dogme)	P & C	$7P$	P (points) $\times$(1 (influence) $+3$ (pos) $\times 2$)
Dirichlet FFD	P	$6P$	P (points) $\times 3$ (pos) $\times 2$

deformations available to other techniques. This is a textbook case of trading off ease of use and versatility.

6.3 Efficiency

The computation cost of a particular spatial deformation technique is a crucial factor in determining its suitability for interactive applications. While processor speed is increasing, such that a method unsuitable at present may well achieve interactive feedback in the near future, so too are end-user's expectations regarding the complexity and detail of models. The relative performance of spatial deformation techniques will remain important.

The graph in Figure 9 demonstrates that the behavior with respect to the number of manipulators (variously, lattice control points, triangles, curves, or points, depending on the technique) differs significantly. Seven representative deformation models are compared: conventional Bézier, Directly Manipulated and Extended FFD are largely constraint independent. In contrast the deformation costs of DOGME (evaluated in its simple product form with five axes per constraint), Radial deformation (Scodef), Wires and t-FFD are highly manipulator dependent and evince a marked decline in the number of deformable vertices with an increase in constraints. The performance of t-FFD on this range of manipulators is somewhat misleading, since, in practice, at least an order of magnitude more manipulators are used.

Figure 9 bundles together both the embedding and deformation stages. For some deformation methods (particularly curve-based schemes) these can be rather asymmetric. As pointed out by Singh and Fiume [1998] such schemes are best suited to an exploratory style of interaction with a single (slow) embedding that links the mesh to the initial manipulator followed by several (faster) deformations exploring different final configurations of the manipulator.

6.4 Correctness

As already mentioned, topological invariance is a property of spatial deformation. One implication is that certain spatial deformations can cause self-intersection of the deformed object. Self-intersection invalidates an object for the purposes of many applications (notably rendering), and there is a need to determine analytically exactly when self-intersection will occur and what restrictions are necessary to prevent it. The first work on preventing self-intersection [Gain and Dodgson 2001] concentrated on necessary and sufficient conditions for ensuring injectivity (a one-to-one mapping between pre- and postdeformation spaces) of FFD and DMFFD. Some recent deformation methods, such as Sweepers [Angelidis et al. 2004], Warp Sculpting [Gain and Marais 2005], Blendeforming [Mason and Wyvill 2001] and Vector-Field Based Deformation [von Funck et al. 2006; 2007], have inbuilt foldover prevention. This is typically achieved by breaking a self-intersecting deformation into multiple injective steps.

Another problem is that a polygon-mesh object exposed to repeated or large scale spatial deformation may exhibit a jagged silhouette and the disappearance of smaller features. In these cases, the polygon coverage in areas of high curvature is insufficient and the polygon mesh is no longer a close approximation to the true surface. Parry [1986] and Greissmair and Purgathofer [1989] outline efficient triangle-mesh refinement methods that adapt to curvature introduced during deformation. Later work by Gain and Dodgson [1999a] also incorporates triangle decimation which prevents a mesh from becoming wastefully oversaturated with triangles.

7. CONCLUSION

In this survey we have examined spatial deformations from the user's perspective, by concentrating on factors such as versatility,

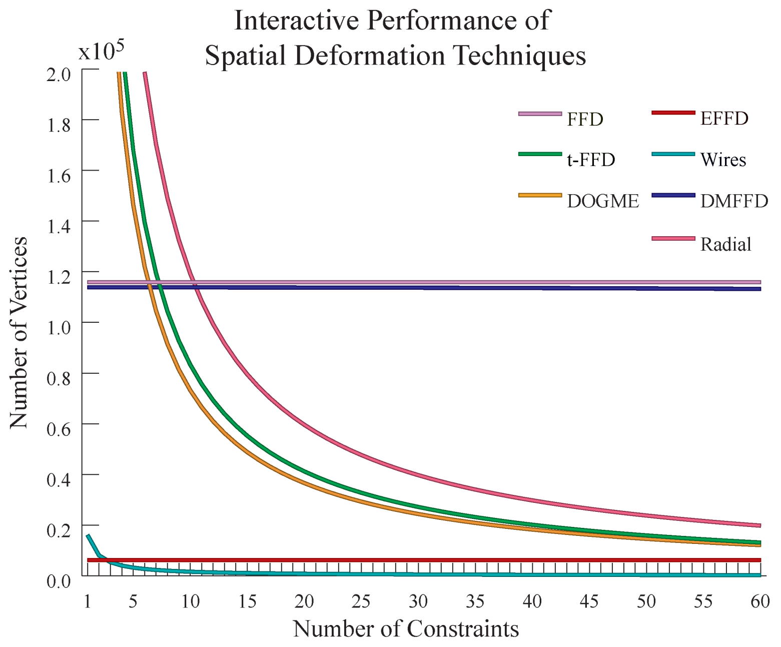

Fig. 9. The number of vertices deformable at 10 updates per second is plotted against the number of manipulators for seven representative techniques. Graphs are produced by counting the number of floating point arithmetic operations as a function of the number of vertices (y-axis) and manipulators (x-axis), with division weighted ten times as heavily as multiplication/addition. The performance of an Intel Core 2 Duo with 2.33 GHz processor on the Whetstone Benchmark is used to calibrate the number of floating point operations per second. Although issues such as memory management and pipelining are ignored, this does provide comparative performance guidelines.

ease of use, efficiency and correctness. While the ultimate choice of deformation technique must depend on the particular application and end-user requirements, we can make some general recommendations.

The most powerful tools in each manipulator category are: FFD with lattices of arbitrary topology (volumes), mean value coordinate deformation (surfaces), wires (curves) and constraint-based DOGME (points). On the other hand, for usability we recommend field-based tools and simple radial formulations (such as Twister). Finally, ease of implementation should not be neglected, particularly when rapid development is a necessity or deformation is just one component of a larger project. In this regard, it is difficult to beat FFD and simple radial deformation for sheer simplicity.

Of course, effective deformation is by no means a solved problem. We foresee fruitful future research in at least three areas:

(1) *Feature preservation.* We would like to see recent developments in volume preservation and foldover prevention extended to the full gamut of deformations. In particular, the notion of preservation should encompass other geometric attributes, such as curve length and surface area. The latter is particularly useful for models that are not watertight (such as two-manifolds with boundary) since they have no inherent notion of volume.

(2) *Improving interaction.* There have been some recent developments in deformation interaction. Here, researchers have progressed by designing an interaction style and then inventing a corresponding deformation to support it. However, little attention has been paid to reconfiguring the interfaces of existing deformation tools to improve their usability.

(3) *Multidimensional multiresolution manipulators.* Currently, no framework encompasses the simultaneous application of manipulators of any dimension. We envisage a system that automatically extracts multi-dimensional features from an object across multiple resolutions, be it a corner point, a feature curve, or the object itself at the lowest level of detail, and then makes these available to a user as persistent handles for deformation. This is in the spirit of the sketched curves in FiberMesh [Nealen et al. 2007], but applied to deformation and extended to multi-dimensional manipulators.

ACKNOWLEDGMENTS

We would like to thank Neil Dodgson and Malcolm Sabin for their help and advice.

REFERENCES

ANGELIDIS, A., CANI, M.-P., WYVILL, G., AND KING, S. 2004. Swirling sweepers: Constant-volume modeling. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications (PG'04)*. IEEE Computer Society, 10–15.

ANGELIDIS, A., WYVILL, G., AND CANI, M.-P. 2004. Sweepers: Swept user-defined tools for modeling by deformation. In *Proceedings of the International Conference on Shape Modeling and Applications (SMI'04)*. IEEE Computer Society, 63–73.

ANGELIDIS, A., WYVILL, G., AND CANI, M.-P. 2006. Sweepers: Swept deformation defined by gesture. *Graph. Models 68*, 1, 2–14.

AUBERT, F. AND BECHMANN, D. 1997a. Animation by deformation of space-time objects. *Proceedings of the Computer Graphics Forum 16*, 3, 57–66.

AUBERT, F. AND BECHMANN, D. 1997b. Volume preserving space deformation. *Comput. Graph. 21*, 5, 625–639.

BARR, A. H. 1984. Global and local deformations of solid primitives. *Comput. Graph. 18*, 3, 21–30.

BECHMANN, D. 1994. Space deformation models survey. *Comput. Graph. 18*, 4, 571–586.

BECHMANN, D. 1998. Multidimensional free-form deformation tools. In *Proceedings of EUROGRAPHICS*'. State of the Art Reports (STARs).

BECHMANN, D., BERTRAND, Y., AND THERY, S. 1996. Continuous free-form deformation. In *Proceedings of Computer Networks and ISDN Systems (Compugraphics'96)*. 157–171.

BECHMANN, D. AND DUBREUIL, N. 1993. Animation through space and time based on a space deformation model. *J. Visualis. Comput. Animat. 4*, 3, 165–184.

BECHMANN, D. AND DUBREUIL, N. 1995. Order controlled free-form animation. *J. Visualis. Comput. Animat. 5*, 1, 11–32.

BECHMANN, D. AND ELKOUHEN, M. 2001. Animating with the multidimensional deformation tool. In *Proceedings of the Workshop on Computer Animation and Simulation (CAS'01)*.

BECHMANN, D. AND GERBER, D. 2003. Arbitrary shaped deformations with dogme. *Vis. Comput. 19*, 2-3, 175–186.

BEZIER, P. 1972. *Numerical Control: Mathematics and Applications*. Wiley.

BLANC, C., GUITTON, P., AND SCHLICK, C. 1994. A methodology for description of geometrical deformations. In *Proceedings of Pacific Graphics*. 49–61.

BOISSONNAT, J.-D. 1984. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph. 3*, 4, 266–286.

BORREL, P. AND BECHMANN, D. 1991. Deformation of n-dimensional objects. *Int. J. Computat. Geom. Appl. 1*, 4, 427–453.

BORREL, P. AND RAPPOPORT, A. 1994. Simple constrained deformations for geometric modelling and interactive design. *ACM Trans. Graph. 13*, 2, 137–155.

BOTSCH, M. AND KOBBELT, L. 2005. Real-time shape editing using radial basis functions. *Comput. Graph. Forum 24*, 3, 611–621.

BOULLION, T. L. AND ODELL, P. L. 1971. *Generalized Inverse Matrix*. Wiley Interscience, New York.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIC, Z. 2002. Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph. 21*, 3, 586–593.

CATMULL, E. AND CLARK, J. 1978. Recursively generated b-spline surfaces on arbitrary topological meshes. *Comput.-Aid. Des. 10*, 6, 350–355.

CHADWICK, J. E., HAUMANN, D. R., AND PARENT, R. E. 1989. Layered construction for deformable animated characters. *Comput. Graph. 23*, 3, 243–252.

CHANG, Y. AND ROCKWOOD, A. P. 1994. A generalized de Casteljau approach to 3D free–Form deformation. *Comput. Graph.* 257–260.

COMNINOS, P. 1989. Fast bends or fast free-form deformation of polyhedral data. In *Computer Graphics*. Blenheim Online Publications, Middlesex, UK, 225–242.

COQUILLART, S. 1990. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. *Comput. Graph. 24*, 4, 187–196.

COQUILLART, S. AND JANCÈNE, P. 1991. Animated free-form deformation: An interactive animation technique. *Comput. Graph. 25*, 4, 23–26.

CRESPIN, B. 1999. Implicit free-form deformations. In *Implicit Surfaces*. 17–23.

DAVIS, O. R. AND BURTON, R. P. 1991. Free-form deformation as an interactive modeling tool. *J. Imag. Tech. 17*, 4, 181–187.

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

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 1997. Dynamic free–form deformation for animation synthesis. *IEEE Trans. Visualiz. Comput. Graph. 3*, 3, 201–209.

FARIN, G. 1990. Surfaces over dirichlet tesselations. *Comput.-Aid. Geom. Des. 7*, 281–292.

FARIN, G. 1997. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, 4th ed. Academic Press.

FENG, J., MA, L., AND PENG, Q. 1996. A new free-form deformation through the control of parametric surfaces. *Comput. Graph. 20*, 4, 531–539.

FENG, J., SHAO, J., JIN, X., PENG, Q., AND FORREST, R. 2005. Multuresolution free-form deformations with subdivision surfaces of arbitrary topology. *Vis. Comput. 22*, 1, 28–42.

GAIN, J. AND MARAIS, P. 2005. Warp sculpting. *IEEE Trans. Visualiz. Comput. Graph. 11*, 2, 217–227.

GAIN, J. E. 2000. Enhancing spatial deformation for virtual sculpting. Ph.D. thesis, University of Cambridge.

GAIN, J. E. AND DODGSON, N. 1999a. Adaptive refinement and decimation under free-form deformation. In *Proceedings of Eurographics*, 7–18.

GAIN, J. E. AND DODGSON, N. 1999b. Enhancing the efficiency and versatility of directly manipulated free-form deformation. In *Proceedings of SIGGRAPH (Conference on Abstracts and Applications)*. ACM Press, 240.

GAIN, J. E. AND DODGSON, N. A. 2001. Preventing self-intersection under free–form deformation. *IEEE Trans. Visualiz. Comput. Graph. 7*, 4, 289–298.

GREISSMAIR, J. AND PURGATHOFER, W. 1989. Deformation of solids with trivariate B-Splines. *Comput. Graph. Forum*, 137–148.

GREVILLE, T. 1960. Some applications of the pseudo-inverse of a matrix. *SIAM Revue II*, 15–22.

GUILLET, S. AND LÉON, J.-C. 1998. Parametrically deformed free-form surfaces as part of a variational model. *Comput.-Aid. Des. 30*, 8, 621–630.

HIROTA, G., MAHESHWARI, R., AND LIN, M. C. 1999. Fast volume-preserving free form deformation using multi-level optimization. In *Proceedings of the 5th ACM Symposium on Solid Modeling and Applications (SMA'99)*. ACM Press, 234–245.

HSU, W. M., HUGHES, J. F., AND KAUFMAN, H. 1992. Direct manipulation of free-form deformations. *Comput. Graph. 26*, 2, 177–184.

HU, S.-M., ZHANG, H., TAI, C.-L., AND SUN, J.-G. 2001. Direct manipulation of ffd : efficient explicit solutions and decomposible multiple point constraints. *Vis. Comput. 17*, 370–379.

JIN, X. AND LI, Y. F. 2000. Three-dimensional deformation using directional polar coordinates. *J. Graph. Tools 5*, 2, 15–24.

JU, T., SCHAEFFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 3, 561–566.

KALRA, P., MANGILI, A., MAGNENAT THALMANN, N., AND THALMANN, D. 1992. Simulation of facial muscle actions based on rational free form deformations. *Comput. Graph. Forum 11*, 3, 59–69.

KAVAN, L., COLLINS, S., ŽÁRA, J., AND O'SULLIVAN, C. 2007. Skinning with dual quaternions. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (13D'07)*. ACM, New York, NY, 39–46.

KIL, Y. J., RENZULLI, P. A., KREYLOS, O., HAMANN, B., MONNO, G., AND STAADT, O. G. 2006. 3d warp brush modeling. *Comput. Graph. 30*, 4, 610–618.

KLOK, F. 1986. Two moving coordinates frames for sweeping along a 3d trajectory. *Comput.-Aid. Geom. Des. 3*, 217–229.

KOBAYASHI, K. G. AND OOTSUBO, K. 2003. t-ffd: free-form deformation by using triangular mesh. In *Proceedings of the 8th ACM Symposium on Solid Modeling and Applications (SM '03)*. ACM Press, 226–234.

LAMOUSIN, H. J. AND WAGGENSPACK, JR., W. N. 1994. Nurbs-based free-form deformations. *IEEE Comput. Graph. Appl. 14*, 6, 59–65.

LAZARUS, F., COQUILLART, S., AND JANCÈNE, P. 1994. Axial deformations : an interactive deformation technique. *Comput.- Aid. Des. 26*, 8, 607–613.

LEE, S.-Y., CHWA, K.-Y., SHIN, S. Y., AND WOLBERG, G. 1995. Image metamorphosis using snakes and free-form deformations. *Comput. Graph.* 439–448.

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. *Comput. Graph. 22*, 3, 663–668.

LLAMAS, I., POWELL, A., ROSSIGNAC, J., AND SHAW, C. D. 2005. Bender: a virtual ribbon for deforming 3d shapes in biomedical and styling applications. In *Proceedings of the ACM Symposium on Solid and Physical Modeling (SPM'05)*. ACM Press, 89–99.

MACCRACKEN, R. AND JOY, K. I. 1996. Free-form deformations with lattices of arbitrary topology. *Comput. Graph.*, 181–188.

MASON, D. AND WYVILL, G. 2001. Blendeforming: Ray traceable localized flodover-free space deformation. *Comput. Graph. Int.* 183–190.

MERRY, B., MARAIS, P., AND GAIN, J. 2006. Animation space: A truly linear framework for character animation. *ACM Trans. Graph. 25*, 4, 1400–1423.

MIKITA, M. 1996. 3d free-form deformation: basic and extented algorithms. In *Proceedings of the 12th Spring Conference on Computer Graphics*. 183–191.

MILLIRON, T., JENSEN, R. J., BARZEL, R., AND FINKELSTEIN, A. 2002. A framework for geometric warps and deformation. *ACM Trans. Graph. 21*, 1, 20–51.

MOCCOZET, L. 1996. Hand modeling and animation for virtual humans. Ph.D. thesis, Université de Genève.

MOCCOZET, L. AND MAGNENAT-THALMANN, N. 1997. Dirichlet free-form deformations and their application to hand simulation. In *Proceedings of the Conference on Computer Animation*. IEEE Computer Society, 93–102.

MOHR, A. AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. In *ACM SIGGRAPH Papers*. ACM, New York, NY, 562–568.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph. 26*, 3, 41–.

NIMSCHECK, U. 1995. Rendering for free-form deformations. Ph.D. thesis, University of Cambridge.

ONO, Y., CHEN, B.-Y., NISHITA, T., AND FENG, J. 2002. Free-form deformation with automatically generated multiresolution lattices. In *Proceedings of IEEE Cyberworlds*. IEEE Computer Society, 472–479.

PARRY, S. R. 1986. Free-form deformations in a constructive solid geometry modeling system. Ph.D. thesis, Department of Civil Engineering, Brigham Young University.

PENG, Q., JIN, X., AND FENG, J. 1997. Arc-length-based axial deformation and length preserving deformation. In *Proceedings of the Conference on Computer Animation*. IEEE Computer Society, 87–93.

PERNOT, J.-P., GUILLET, S., LÉON, J.-C., GIANNINI, F., CATALANO, C. E., AND FALCIDIENO, B. 2002. A shape deformation tool to model character lines in the early design phases. In *Proceedings of Shape Modeling International*. 165–174.

RAFFIN, R., NEVEU, M., AND JAAR, F. 2000. Curvilinear displacement of free-form-based deformation. *Vis. Comput. 16*, 38–46.

RAPPOPORT, A., SHEFFER, A., AND BERCOVIER, M. 1995. Volume-preserving free-form solid. In *Proceedings of the 3rd ACM Symposium on Solid Modeling and Applications (SMA'95)*. ACM, New York, NY, 361–372.

ROSSIGNAC, J. 1994. Introduction to the special issue on interactive sculpting. *ACM Trans. Graph. 13*, 2.

RUPRECHT, D., NAGEL, R., AND MULLER, H. 1995. Spatial free-form deformation with scattered data interpolation methods. *Comput. Graph. 19*, 63–72.

SCHEIN, S. AND ELBER, G. 2004. Discontinuous free-form deformations. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications (PG '04)*. IEEE Computer Society, 227–236.

SCHNEIDERMAN, B. 1983. Direct manipulation: a step beyond programming languages. *IEEE Comput. 16*, 8, 57–69.

SEDERBERG, T., ZHENG, J., BAKENOV, A., AND NASRI, A. 2003. T-splines and t-nurccs. *ACM Trans. Graph. 22*, 3, 477–484.

SEDERBERG, T. W. AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. *Comput. Graph. 20*, 4, 151–160.

SINGH, K. AND FIUME, E. 1998. Wires: A geometric deformation technique. *Comput. Graph.*, 405–414.

SINGH, K. AND KOKKEVIS, E. 2000. Skinning characters using surface-oriented free-form deformations. In *Proceedings of Graphies Interface*. 35–42.

SLOAN, P.-P. J., CHARLES F. ROSE, I., AND COHEN, M. F. 2001. Shape by example. In *Proceedings of the Symposium on Interactive 3D Graphics (I3D'01)*. ACM, New York, NY, 135–143.

SONG, W. AND YANG, X. 2005. Free-form deformation with weighted t-spline. *Vis. Comput. 21*, 3 (Apr.), 139–151.

VON FUNCK, W., THEISEL, H., AND SEIDEL, H.-P. 2006. Vector field based shape deformations. *ACM Trans. Graph. 25*, 3, 1118–1125.

VON FUNCK, W., THEISEL, H., AND SEIDEL, H.-P. 2007. Explicit control of vector field based shape deformations. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications (PG'07)*. IEEE Computer Society, Washington, DC, 291–300.

WANG, X. C. AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (ESCA'02)*. ACM, New York, NY, 129–138.

Received July 2006; revised November 2007, June 2008; accepted June 2008