Preventing Self-Intersection under Free-Form Deformation

James E. Gain and Neil A. Dodgson

Abstract—Free-Form Deformation (FFD) is a versatile and efficient modeling technique which transforms an object by warping the surrounding space. The conventional user-interface is a lattice of movable control points but this tends to be cumbersome and counterintuitive. Directly Manipulated Free-Form Deformation (DMFFD) allows the user to drag object points directly and has proven useful in an interactive sculpting context. A serious shortcoming of both FFD and DMFFD is that some deformations cause self-intersection of the object. This is unrealistic and compromises the object's validity and suitability for later use. An in-built self-intersection test is thus required for FFD and its extensions to be truly robust. In this paper, we present the following novel results: a set of theoretical conditions for preventing self-intersection by ensuring the injectivity (one-to-one mapping) of FFD, an exact (necessary and sufficient) injectivity test which is accurate but computationally costly, an efficient but approximate injectivity test which is a sufficient condition only, and a new form of DMFFD which acts by composing many small injective deformations. The latter expands the range of possible deformations without sacrificing the speed of the approximate test.

Index Terms—Free-form deformation, direct manipulation, self-intersection, space homeomorphism.

1 Introduction

Free-form Deformation (FFD) [7], [29] and its extensions form a class of highly intuitive, versatile, and efficient free-form modeling and animation tools. They address the task of computer-assisted shape design and modification by warping the ambient space containing an embedded object. However, a major drawback which is often overlooked in the literature is the self-intersection of objects under FFD. In this paper, we present a novel solution to this problem by developing two self-intersection tests: the first exact, although inefficient, and the second approximate, but efficient. Finally, we seamlessly incorporate this latter test into Directly Manipulated Free-Form Deformation (DMFFD) [19], [8], providing a robust and interactive modeling tool which prevents self-intersection.

Free-Form Deformation employs a hyperpatch (the three-dimensional analog of a two-dimensional parametric patch) to demarcate a deformable portion of world coordinate space. A lattice of control points governs the shape of this hyperpatch. The FFD process is analogous [29] to setting a deformable shape inside a block of jelly of the same consistency and then flexing this jelly, resulting in a corresponding distortion of the inset shape. Unfortunately, specifying deformations by moving lattice control points has proven awkward [19]. Instead, the Direct Manipulation extensions to Free-Form Deformation (DMFFD) replace the cumbersome lattice interface with a set of user-defined constraints, each consisting of a point within the hyperpatch (usually on the object's surface) and its intended motion. The lattice changes necessary to satisfy these constraints are then calculated.

FFD in its seminal form [7], [29] utilized a Bézier hyperpatch with the associated lattice restricted to an initial regular parallelepiped configuration. Later research expanded the range of FFD by generalizing the topology of the initial lattice [11], [22] and coupled adaptive meshing schemes to FFD in order to address degradation in the quality of objects under deformation [16], [26], [14]. The Free-Form Deformation methods can be characterized as lattice-based spatial deformations. There are, however, two other types of control mechanism. Curve-based deformations [5], [10], [21], [31] bind spatial distortions to the motion of curves and point-based spatial deformations [19], [8], [9], [4], of which DMFFD is an instance, are controlled by direct point manipulation.

A serious weakness common to all forms of spatial deformation is the potential for self-intersection of an object. This interpenetration of portions of the object's surface is problematic for a number of reasons. First, it is highly counterintuitive. No real-world solids can contort in this fashion without rupturing and, as a consequence, it is unlikely to fulfill the intentions of the modeler. Second, self-intersection contravenes the correctness of the affected object. For instance, some boundary representation systems support only 2-manifold solids (where any point and neighborhood on the manifold is topologically equivalent to a disk in the plane) and these are invalidated by interpenetrating faces [13]. It also compromises certain applications, notably rendering, which often assumes that only the outside of a polyhedron is visible, and texturing algorithms that rely on a "single sheet" property. The importance of an absence of self-intersection is evidenced by its inclusion in the ISO STEP Standard for CAD data

- *J.E. Gain is with the Collaborative Visual Computing Laboratory, Computer Science Department, University of Cape Town, Private Bag RONDEBOSCH, 7701, South Africa. E-mail: jgain@cs.uct.ac.za.*
- *N.A. Dodgson is with the Rainbow Graphics Group, Computer Laboratory, University of Cambridge, New Museums Site, Pembroke Street, Cambridge, CB2 3QG, UK. E-mail: Neil.Dodgson@cl.cam.ac.uk.*

Manuscript received 14 May 1999; revised 24 Aug. 2000; accepted 27 Oct. 2000.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 109812.

transfer [17]. Finally, the prevention of self-intersection is required for the existence of an inverse. This has implications for building a workable (memory and computation efficient) "undo" operation.

The focus of this paper is on automatically detecting and preventing self-intersection under FFD. We begin with a brief survey of related work. This is followed by a background explanation of the notation and mechanism of Free-Form Deformation and Direct Manipulation. We then present a series of novel results:

- a set of theoretical conditions for preventing self-intersection by ensuring the injectivity (one-to-one mapping) of the FFD hyperpatch,
- an exact (necessary and sufficient) injectivity test of the FFD Jacobian which is accurate but computationally costly,
- an efficient but approximate injectivity test which is a sufficient condition only and may, as a consequence, falsely reject valid deformations,
- a new form of DMFFD which acts by composing many small injective deformations. This expands the range of possible deformations and enhances realism without sacrificing the speed of the approximate test.

2 Related Work

Self-intersection has been largely overlooked in the spatial deformation literature. Perhaps this is because, in the case of FFD, the lattice provides an indication to experienced users of the degree of deformation. However, the lattice is not always a transparent guide to self-intersection. A lattice with overlapping faces does not necessarily imply self-intersection. Conversely though, a lattice without overlap does provide a good indication that there is no hyperpatch self-intersection. Nevertheless, even the limited feedback provided by an FFD lattice is not available in curve and point-based spatial deformation.

There are several passing references to the dangers of self-intersection in the spatial deformation literature [11], [21], [22], but only Borrel and Rappoport [9] embark on a concerted investigation. They identify a "space-tearing" phenomenon (similar to Fig. 3b) in connection with Simple Constrained Deformation (ScoDef), a radial point-based technique. The problem is ameliorated by a method of duplicating point manipulations, but this places the onus of identifying and correcting self-intersection on the user.

Contemporary work by Joy and Duchaineau [20] presents an analysis similar to our efficient sufficient injectivity test (Section 7). They also employ a conic-hull hodograph to identify possible zeros in the Jacobian of deformation. However, their research is focused on the boundary determination and rendering of trivariate solids rather than self-intersection prevention.

Self-intersection has garnered much attention in the context of offset curves and surfaces [18]. For instance, Maekawa et al. [23] develop necessary and sufficient conditions for preventing the self-intersection of tubular pipe surfaces which are offsets from a rational curve.

There has been comprehensive research into establishing perturbation bounds on vertices and control points which preserve the topological form of polyhedral 2-spheres [33], rectilinear finite polyhedra [1], and objects composited from Bézier curves and surfaces [2]. This vein of research relies (like our own) on conditions for ensuring a space homeomorphism which, inter alia, prevents self-intersection. These results are extensible to FFD and would provide limits on the displacement of lattice control points. In contrast, our methods are more directly applicable to other spatial deformation techniques (e.g., direct manipulation).

3 Free-Form Deformation

Free-Form Deformation is a modeling tool which warps the space surrounding an object and thereby transforms the object indirectly. This is achieved by imposing a parametric hyperpatch onto a portion of world coordinate space and linking distortions in the hyperpatch to object vertices.

FFD can be formulated as a mapping, $\mathcal{F} : \Re^3 \mapsto \Re^3 \mapsto \Re^3$, from world space, $X = (x, y, z)$, through the local parameter space of the hyperpatch, $U = (u, v, w)$, to deformed world space, $\tilde{X} = (\tilde{x}, \tilde{y}, \tilde{z})$. This is achieved by two functions: the embedding and deformation functions, $F(U) = X$ and $\tilde{F}(U) = \tilde{X}$. The composition of $\tilde{F}$ and F^{-1} constitutes FFD: $\mathcal{F}(X) = \tilde{F}(F^{-1}(X)) = \tilde{F}(U) = \tilde{X}$.

The medium of deformation is a hyperpatch, usually defined as a trivariate polynomial tensor product volume. This is a straightforward extension of one-dimensional curves to three dimensions. The curve's control polygon, which indicates the adjacency of control points, generalizes to a control lattice. Likewise, just as curves may be divided into piecewise segments (on a subinterval of the univariate domain), so, too, a hyperpatch may be broken into cells (each defined over a parallelepiped block of the trivariate domain). Deformation is achieved by linking distortions of the hyperpatch to object vertices. A lattice of control points, P, weighted by polynomial basis functions, B^s_r, with index r and order s, governs points within the hyperpatch, $F(U) = F(u, v, w)$, as follows:

$$F(U) = \sum_{i=1}^{a+l} \sum_{j=1}^{b+m} \sum_{k=1}^{c+n} B^l_i(u) \cdot B^m_j(v) \cdot B^n_k(w) \cdot P_{ijk}. \qquad (1)$$

With this background in place, FFD proceeds in three stages:

1. Object vertices which fall within the undistorted hyperpatch are assigned parametric u, v, w coordinates (F^{-1} is applied). In terms of the earlier metaphor, the shape being deformed is set inside a block of jelly.
2. A number of control points are displaced (P become $\tilde{P}$), with a consequent distortion of the hyperpatch. This equates to flexing the jelly.
3. The deformed version of (1) is applied repeatedly to all of the parametrized object vertices to produce a deformed version of the object. So, by the analogy, the inset shape is warped along with its cocooning jelly.

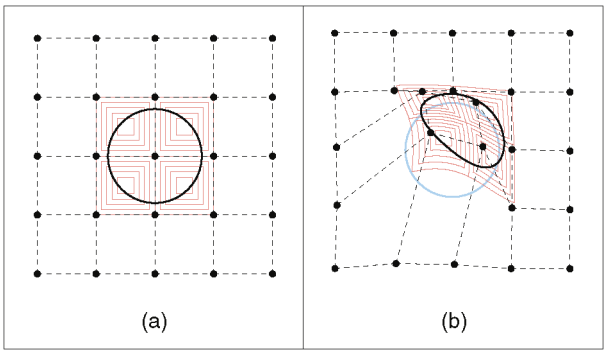

Fig. 1. Two-dimensional Free-Form Deformation with a uniform cubic B-spline basis. (a) Predeformation: A regular initial lattice with four cells (in red) and an embedded circle. (b) Postdeformation: The user repositions control points and the cells and object are deformed.

In the interests of computational efficiency and presentation brevity, we adopt a uniform B-spline basis and a regular axis-aligned initial lattice with control points evenly spaced in a parallelepiped configuration. Now, due to the linear precision property of B-splines [12], the embedding of vertices is immediate ($X \equiv U$ and F^{-1} is the identity function). This allows us to drop the embedding function from further consideration and we need consider only the deformation function. Fig. 1 provides an example of this type of Free-Form Deformation.

4 Direct Manipulation

Controlling deformations by moving lattice vertices while producing sculpted results tends to be cumbersome and counterintuitive [19]. Specifying even simple deformations requires a good working knowledge of splines and FFD. While a deformation "follows" control point displacement, exact control of a given object point is difficult. Also, the lattice tends to clutter the screen and obscure the object being created. Even worse, some lattice control points may be hidden within the object. It would be preferable if the user could drag object points directly and have the surrounding surface conform smoothly. This is the intention behind the Direct Manipulation extensions to Free-Form Deformation (DMFFD) [19], [8]. For instance, pushing or pulling a single object point will create either a dimple or a mound in the object's surface. More complex manipulation can be achieved by simultaneously moving several points and calculating the lattice changes required to induce these effects.

DMFFD can be broken down into three steps: The user provides a number of constraints, each composed of a point and its intended motion, the lattice control points are altered to meet these constraints, and this new lattice is applied through standard FFD to the original object (see Fig. 2).

To achieve this, some mathematical foundations must first be established. The deformation form of (1), applied to multiple object points, can be concisely expressed using matrix notation as:

$$\mathbf{B}\tilde{\mathbf{P}} = \tilde{\mathbf{F}}, \tag{2}$$

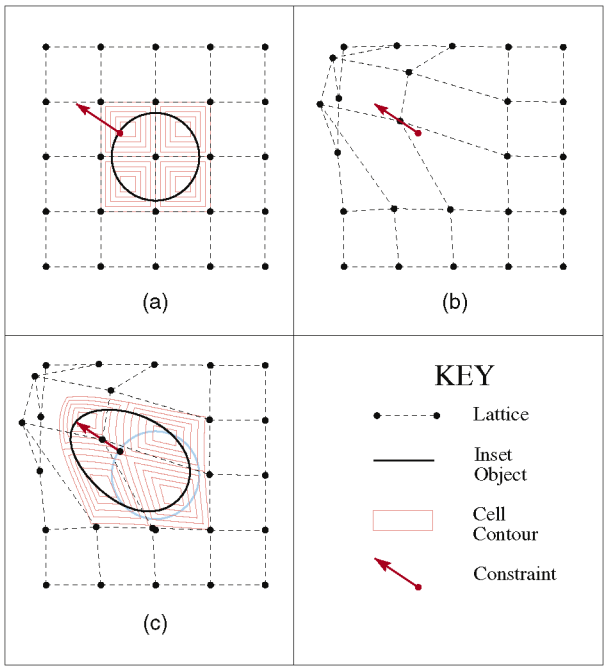

Fig. 2. Two-Dimensional Directly Manipulated Free-Form Deformation. (a) Setup: A single point and vector constraint on a circle within an FFD lattice. (b) Lattice alteration: Control points are repositioned to satisfy the constraint. (c) Deformation: The altered lattice is applied via FFD to the circle.

where $\tilde{\mathbf{F}}$ is an $r \times 3$ matrix with each row holding the deformed coordinates of an object point. $\tilde{\mathbf{P}}$ is a $c \times 3$ matrix of repositioned control point coordinates. It is formed by cycling through $\tilde{\mathbf{F}}$ and placing in $\tilde{\mathbf{P}}$, without duplication, all control points that influence each object point under consideration. This means that a control point is only included if it contributes a nonzero weight to at least one object point. $\mathbf{B}$ is an $r \times c$ matrix of tensor product basis functions, with the weight entries in column j of $\mathbf{B}$ matched to the control point in row j of $\tilde{\mathbf{P}}$. A particular (i, j) entry of $\mathbf{B}$ is zero if the control point in row j of $\tilde{\mathbf{P}}$ does not affect the object point in row i of $\tilde{\mathbf{F}}$. Essentially, the basis summation in (1) is expanded, reordered, interspersed with zeros, and fitted into a row of $\mathbf{B}$ [15].

Both $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{F}}$ can be separated into undeformed and delta components as $\tilde{\mathbf{P}} = \mathbf{P} + \Delta\mathbf{P}$ and $\tilde{\mathbf{F}} = \mathbf{F} + \Delta\mathbf{F}$, respectively. This allows us to recast (2) so as to focus on the change in control and object points under FFD.

$$\begin{aligned} &\mathbf{B}\tilde{\mathbf{P}} = \tilde{\mathbf{F}} \\ &\Rightarrow \mathbf{B}(\mathbf{P} + \Delta\mathbf{P}) = \mathbf{F} + \Delta\mathbf{F} \\ &\Rightarrow \mathbf{B}\,\Delta\mathbf{P} = \Delta\mathbf{F}. \end{aligned} \tag{3}$$

Normally, FFD evaluates the alteration in object points, $\Delta\mathbf{F}$, by multiplying the basis matrix of spline weights, $\mathbf{B}$, and the list of control point changes, $\Delta\mathbf{P}$, but direct manipulation reverses this. The user specifies a selection of object points, $\mathbf{F}$, and their intended motion, $\Delta\mathbf{F}$, and the new control point positions, $\mathbf{P} + \Delta\mathbf{P}$, are found. In mathematical terms, we seek to solve for $\Delta\mathbf{P}$ in the system of linear equations $\mathbf{B}\,\Delta\mathbf{P} = \Delta\mathbf{F}$, given $\mathbf{B}$ and $\Delta\mathbf{F}$.

This process is very well defined if $\mathbf{B}$ is square ($r = c$) and nonsingular, in which case, $\Delta\mathbf{P}$ can be solved explicitly

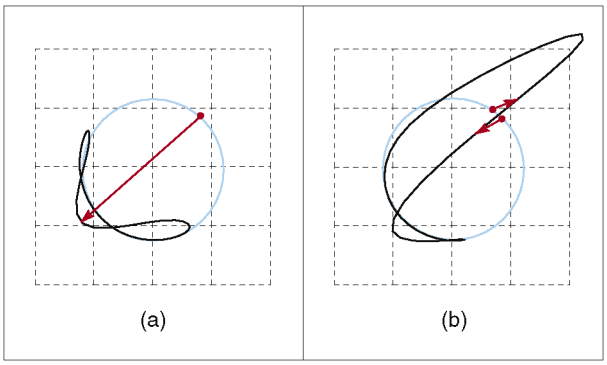

Fig. 3. Self-intersection under two-dimensional DMFFD. (a) Overextension: An elongated constraint vector reaching beyond the area of effect of its associated point. (b) Overconstraint: Two nearly coincident constraint points are wrenched in opposite directions.

as $\Delta\mathbf{P} = \mathbf{B}^{-1}\Delta\mathbf{F}$. Unfortunately, $\mathbf{B}$ is seldom a square matrix and the system is thus either underdetermined or overdetermined. In the former case, there are more unknowns than equations ($r < c$) and an infinite number of solutions. In the latter case, there are more equations than unknowns ($r > c$) and there is no exact solution since not all of the constraints can be met. Only the underdetermined case is considered here since interactive sculpting presupposes a rapid sequence of relatively simple deformations. Hence, the number of direct manipulation vectors (r) is less than $l \cdot m \cdot n$, which is the minimum number of control points ($c \geq l \cdot m \cdot n$).

To solve for $\Delta\mathbf{P}$, we rely on a formulation known as the pseudoinverse [27], represented by $\mathbf{B}^{+}$, which extends the definition of the inverse. In the underdetermined situation, it minimizes the sum of squares (or norm) of the solution matrix ($\|\Delta\mathbf{P}\|$). This corresponds roughly to finding the smallest overall change in the control points consistent with a valid solution. The pseudoinverse can be explicitly evaluated according to $\mathbf{B}^{+} = \mathbf{B}^{T}(\mathbf{B}\mathbf{B}^{T})^{-1}$ [27]. This leads to an overall solution for (3):

$$\Delta\mathbf{P} = \mathbf{B}^{T}(\mathbf{B}\mathbf{B}^{T})^{-1}\Delta\mathbf{F}. \tag{4}$$

For interactive performance, this is best solved [15] by exploiting the sparse structure of the basis matrix and using Choleski Factorization [28].

5 Injectivity Analysis

This work addresses a flaw in both Free-Form Deformation and its Direct Manipulation extensions. Certain lattice distortions cause self-intersection of an embedded object. Fig. 3 shows two such cases in the context of DMFFD, which is particularly prone to this problem because large deformations may be instigated by relatively small constraint movements.

Hyperpatch self-intersection is a necessary but not sufficient condition for object self-intersection. Thus, self-intersection within an FFD hyperpatch is required to cause self-intersection of the object, but a self-intersecting hyperpatch does not always produce a self-intersecting object.

This paper focuses on ensuring that FFD does not introduce self-intersection. It is implicitly assumed that the predeformation object is free from self-intersection.

There are two approaches to the detection of self-intersection, each with its strengths and concomitant weaknesses. A space-based test would predict the self-intersection of an FFD hyperpatch by analyzing its associated lattice. Such a test is independent of both the object's representation and geometry. Once a particular FFD is established as injective, it can be applied to an object with any internal description (be it implicit, CSG, or B-Rep) or shape (no matter how convoluted) without fear of self-intersection. The alternative is to check for intersection on a polygon-by-polygon basis. This naive object-based test suffers from numerous failings. It is computationally prohibitive, cannot detect the complete inversion of an object, can only be applied after deformation, requires a polygon-mesh and is subject to error caused by the approximation inherent in this representation [15]. Given these considerations an analytic lattice-based self-intersection test is preferable.

Self-intersection requires that at least two points in the initial space map under FFD to a single point in deformed space. In particular, the inset object becomes self-intersecting if two or more points on its boundary prior to deformation are forced by FFD to coincide. An injective (one-to-one) mapping means that every predeformation point is transformed to a unique and separate postdeformation position. Injectivity thus implies non-self-intersection. The following theorem provides a set of requirements for injectivity (and, more broadly, homeomorphism) of FFD:

Theorem 1. *Let $\mathcal{F}$ be a spatial deformation function of the form $\mathcal{F} : X \mapsto \tilde{X}$; $X, \tilde{X} \subset \Re^n$ and $\mathbf{J}$ be the Jacobian matrix of $\mathcal{F}$. $\mathcal{F}$ is a homeomorphism (injective, onto, and invertible) from X to $\tilde{X}$ iff*

1. *$\mathcal{F}$ has continuous first partial derivatives,*
2. *$det(\mathbf{J}) > 0$.*

Theorem 1 relies on two coupled theorems: the Inverse Function Theorem [32], which gives requirements for local injectivity in the neighborhood of a point, and a result, due to Meisters and Olech [24], which extends this to an entire closed bounded domain.

The first condition in Theorem 1 is easily met by ensuring at least C_1 continuity of the basis functions in $\tilde{F}$. B-splines in particular are $n - k - 1$ times continuously differentiable on an order n basis at a knot of multiplicity k [12] and $n - k \geq 2$ is therefore required. Maintaining this continuity at the join between the FFD hyperpatch boundary and surrounding space can be achieved by a shell of static or "phantom" control points [6]. This condition on the boundary of FFD, together with $det(\mathbf{J}) > 0$, enables the full space homeomorphism of $\mathcal{F} : \Re^n \mapsto \Re^n$ advocated by many authors [33], [1], [23]. If this seam continuity is not enforced, then the injective domain is restricted to the hyperpatch into which the entire object must be embedded.

The second condition relies on the positivity of the FFD Jacobian, $det(\mathbf{J})$. The Jacobian at a point provides a measure

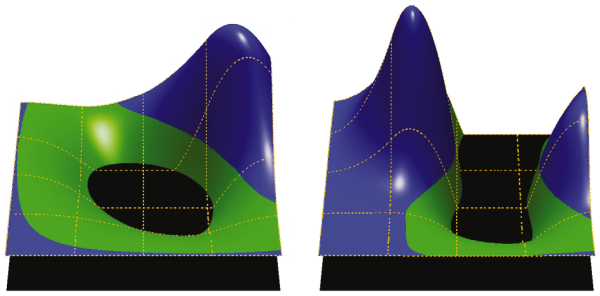

Fig. 4. Two Jacobians which are represented as height fields above a two-dimensional domain and correspond to the self-intersecting deformations in Fig. 3. The peaks and troughs show areas of expansion (blue) and contraction (green). Injectivity is broken where the Jacobians drop below the black ground plane.

of the local distortion. If a two-dimensional domain is envisaged as an elasticated sheet, then the Jacobian indicates the existence and magnitude of expansion ($det(\mathbf{J}) > 1$), contraction ($0 < det(\mathbf{J}) < 1$), or foldover ($det(\mathbf{J}) < 0$) caused by contorting the sheet. A deformation is perfectly volume preserving if the Jacobian is uniformly unity ($det(\mathbf{J}) = 1$) [29]. In Theorem 1, the positivity restriction (rather than the more conventional prevention of singularity, $det(\mathbf{J}) \neq 0$) avoids the inversion of the domain that occurs with uniformly negative Jacobians. Fig. 4 plots two deformation Jacobians as height fields on a planar domain. These correspond to the self-intersecting direct manipulations in Fig. 3. Note the dark regions where the Jacobians fall below the plane $det(\mathbf{J}) = 0$ and which signal that the deformations are folding back upon themselves.

This theorem is applicable to all forms of FFD: It is independent of dimension (it can be applied with equal ease to planar or volume warping), basis (be it Bernstein or B-spline, rational, or nonrational), and lattice topology.

6 A Necessary and Sufficient Injectivity Test

The Jacobian of Free-Form Deformation is a mapping of the form $det(\mathbf{J}) : \Re^3 \mapsto \Re$, which associates a scalar distortion value, d, with each point in the hyperpatch domain, U. A necessary and sufficient FFD injectivity test should establish within machine precision whether these values reach or drop below zero anywhere in the domain, thereby breaking the injectivity conditions of Theorem 1.

The Jacobian of FFD ($det(\mathbf{J})$) takes the form:

$$det(\mathbf{J}) = det \begin{bmatrix} \frac{\partial \tilde{F}_x}{\partial u} & \frac{\partial \tilde{F}_x}{\partial v} & \frac{\partial \tilde{F}_x}{\partial w} \\ \frac{\partial \tilde{F}_y}{\partial u} & \frac{\partial \tilde{F}_y}{\partial v} & \frac{\partial \tilde{F}_y}{\partial w} \\ \frac{\partial \tilde{F}_z}{\partial u} & \frac{\partial \tilde{F}_z}{\partial v} & \frac{\partial \tilde{F}_z}{\partial w} \end{bmatrix}. \quad (5)$$

The injectivity test can be decomposed into two stages: The deformation Jacobian is converted into a conventional trivariate tensor product hyperpatch with scalar control points and this is then recursively subdivided into subcells until, by examining the signs of the refined control scalars, either a negative subcell is encountered or all subcells are found to be positive. This algorithm can now be examined in more detail:

1. A tensor product expression for the deformation Jacobian can be obtained by substituting the partial derivatives of (1) into (5) and regrouping terms:

$$\begin{aligned} det(\mathbf{J}) = & \sum_{d=2}^{a+l} \sum_{e=1}^{a+l} \sum_{f=1}^{a+l} B_d^{l-1}(u) \cdot B_e^l(u) \cdot B_f^l(u) \cdot \\ & \sum_{i=2}^{b+m} \sum_{j=1}^{b+m} \sum_{k=1}^{b+m} B_i^{m-1}(v) \cdot B_j^m(v) \cdot B_k^m(v) \cdot \\ & \sum_{r=2}^{c+n} \sum_{s=1}^{c+n} \sum_{t=1}^{c+n} B_r^{n-1}(w) \cdot B_s^n(w) \cdot B_t^n(w) \cdot \phi, \end{aligned} \quad (6)$$

where

$$\phi = det \begin{bmatrix} \tilde{P}_{djt}^{\langle u \rangle} & \tilde{P}_{eis}^{\langle v \rangle} & \tilde{P}_{fkr}^{\langle w \rangle} \end{bmatrix} \quad (7)$$

and

$$\begin{aligned} \tilde{P}_{djt}^{\langle u \rangle} &= (\tilde{P}_{d,j,t} - \tilde{P}_{d-1,j,t}) \\ \tilde{P}_{eis}^{\langle v \rangle} &= (\tilde{P}_{e,i,s} - \tilde{P}_{e,i-1,s}) \\ \tilde{P}_{fkr}^{\langle w \rangle} &= (\tilde{P}_{f,k,r} - \tilde{P}_{f,k,r-1}). \end{aligned}$$

The components along each axis of (6) are the products of three B-spline basis functions (e.g., $B^{l-1}(u) \otimes B^l(u) \otimes B^l(u)$).

2. Equation (6) is particularly unmanageable since it is costly to evaluate and standard operations such as subdivision cannot be applied directly. There is a useful set of recurrence relations, discovered by Mørken [25], that allows the algebraic product of two B-spline functions to be represented under certain linear combinations as a single higher order B-spline function. This can be applied repeatedly to simplify the B-spline products in (6) and obtain a more tractable trivariate tensor product form,

$$det(\mathbf{J}) = \sum_{o=1}^{\hat{o}} \sum_{p=1}^{\hat{p}} \sum_{q=1}^{\hat{q}} B_o^{\hat{l}}(u) \cdot B_p^{\hat{m}}(v) \cdot B_q^{\hat{n}}(w) \cdot \varrho_{opq}, \quad (8)$$

where $\hat{o} = a(2l-1) + l - 3$, $\hat{p} = b(2m-1) + m - 3$, $\hat{q} = c(2n-1) + n - 3$, $\hat{l} = 3(l-1)$, $\hat{m} = 3(m-1)$, and $\hat{n} = 3(n-1)$. Each control scalar, ϱ_{opq}, is a combination of determinant scalars, ϕ, and product B-splines.

Details of the discrete product bases and their evaluation, as well as how new product knot sequences are formed from the constituent multiplicand sequences, can be found in [25]. It is worth noting, however, that, although the order of the B-spline functions is substantially increased (from l, m, and n to $3(l-1)$, $3(m-1)$, and $3(n-1)$, respectively), this does not apply to the knot continuity, which remains C_{l-2}, C_{m-2}, and C_{n-2}. This almost complete degeneracy results from large knot multiplicities and this simplifies the conversion to multi-Bézier form in the next stage. Forming the control scalars ϱ_{opq} in (8) is very costly, even if the

discrete product bases are amenable to preevaluation, as they are in the uniform B-spline FFD case.

3. The algorithm next focuses on each cell of the deformation Jacobian (8) in turn. A cell under consideration is converted from B-spline to Bézier form using repeated insertion of domain knots [12]. This is useful because Bézier curves (and, hence, hyperpatches) interpolate their endpoints and thus have a tighter convex hull than their uniform B-spline counterparts. Since the final stage of the injectivity test relies on recursively shrinking the convex hull toward the hyperpatch, this conversion improves the starting conditions.
4. Recursive subdivision proceeds by successively refining the control scalars of each Jacobian Bézier cell into eight subcells. The de Casteljau algorithm [12] is applied repeatedly to split the cell along the u, v, and then w coordinate axes. The subdivided control scalars converge quadratically toward the Jacobian hyperpatch. The recursive subdivision search space is an Octree, with each node (cell) spawning eight children (subcells). The leaves of the Octree are subcells whose control scalars are either all negative or all positive, which, by the convex hull property of B-splines [12], implies that the subcell itself has the same uniformity of sign over its domain. The injectivity test reports "success" if all leaf nodes in the recursive subdivision Octree are positive and terminates with "failure" immediately upon generating a negative node. A recursive subdivision procedure is preferred to iterative search because of its robustness in locating the absolute (rather than merely local) minimum.

   This necessary and sufficient test is capable of precisely separating injective and noninjective deformations, but its computation burden is high. An SGI Octane ($\mathrm{R10000} \times 195\mathrm{MHz}$) requires at least 37 s to test the injectivity of a single-celled hyperpatch, even without any recursive subdivision of the Jacobian control scalars (in Step 4). If 10 updates per second is considered reasonably interactive, then, even without the additional overhead of FFD, this test is roughly two orders of magnitude too slow.

7 An Efficient Sufficient Injectivity Test

Our necessary and sufficient injectivity test, while exact, is computationally costly and thus not suitable for use in an interactive context. We modify the precise algorithm to produce a weak sufficient test. This sacrifices the full range of injective deformations for improved performance by classifying some valid deformations as self-intersecting.

The positivity of the Jacobian control scalars ($\phi > 0$) in (6) is, by the positivity of B-splines [12], a sufficient condition for injectivity of FFD. This test can be optimized by: 1) exploiting the local control of B-splines, which allows the determinant scalars to be evaluated in $l \cdot m \cdot n$ blocks and 2) precalculating the partial derivative vectors ($\tilde{P}^{\langle u \rangle}$, $\tilde{P}^{\langle v \rangle}$, and $\tilde{P}^{\langle w \rangle}$) over the entire lattice. Unfortunately, this test remains inefficient.

A geometric interpretation of the control scalars (ϕ) is the key to improving this situation. Each determinant (ϕ) represents the signed volume of a parallelepiped whose edges are formed from the three component vectors ($\tilde{P}^{\langle u \rangle}, \tilde{P}^{\langle v \rangle}, \tilde{P}^{\langle w \rangle}$). The volume's sign is determined by the orientation of one vector relative to the plane formed by the other two. This relationship is expressed in the following vector equation:

$$\begin{aligned} \phi &= det\left[\, \tilde{P}^{\langle u \rangle} \quad \tilde{P}^{\langle v \rangle} \quad \tilde{P}^{\langle w \rangle} \,\right] \\ &= (\tilde{P}^{\langle u \rangle} \times \tilde{P}^{\langle v \rangle}) \cdot \tilde{P}^{\langle w \rangle}. \end{aligned} \tag{9}$$

The cross product ($\tilde{n} = \tilde{P}^{\langle u \rangle} \times \tilde{P}^{\langle v \rangle}$) produces a vector normal to the plane defined by its arguments ($\tilde{P}^{\langle u \rangle}, \tilde{P}^{\langle v \rangle}$). The dot product ($\phi = \tilde{n} \cdot \tilde{P}^{\langle w \rangle}$) is negative or positive, depending on whether the angle between $\tilde{n}$ and $\tilde{P}^{\langle w \rangle}$ is either obtuse ($> \frac{\pi}{2}$) or acute ($< \frac{\pi}{2}$). In other words, ϕ is positive if $\tilde{P}^{\langle w \rangle}$ falls on the same side of the orientation plane as $\tilde{n}$, zero if it lies in the plane, and negative if it lies on the opposite side.

Now, rather than expensively testing ϕ for all cell-wise combinations of partial derivative vectors, enclosing convex hulls are formed around the derivative vectors for each axis. These hulls are used in an approximate test for the sign of ϕ. The algorithm proceeds as follows:

1. All the partial derivative vectors for each axis are calculated by taking the difference of adjacent lattice control points along the u, v, w axes. By basing these derivative vectors at a fixed position in Euclidean space (the origin), a hodograph [12] is formed. Next, the angle of deviation of each hodograph element from its associated axis is found and stored. For efficiency reasons, the intermediate value, $t = \tan^2\theta$, is used in place of the angular axis deviation, θ. If a u-oriented derivative is $\tilde{P}^{\langle u \rangle} = (a, b, c)$, then $t^{\langle u \rangle} = (b^2 + c^2)/a^2$ (assuming $a > 0$). As soon as $a \leq 0$ is encountered, the test is halted with a negative ϕ result relative to the v and w axes. The other angular deviation metrics, $t^{\langle v \rangle}$ and $t^{\langle w \rangle}$, can be evaluated in a corresponding fashion.
2. For each cell (C) in the deformed hyperpatch:
   a. The maximum angle measure, $m^{\langle u \rangle}, m^{\langle v \rangle}, m^{\langle w \rangle}$, among the derivative vectors local to C is found for each axis. Once converted into angles ($\theta = \arctan\sqrt{m}$), these define half-cones centered on the appropriate axis, which represent convex hulls bounding the cell derivatives.
   b. Equation (9) can now be recast in terms of these conic-hulls.
      - Cross Product: As long as two conic hulls ($\theta^{\langle u \rangle}, \theta^{\langle v \rangle}$) bounding orthogonal axes (u, v) do not overlap (i.e., $\theta^{\langle u \rangle} + \theta^{\langle v \rangle} < \frac{\pi}{2}$), then a bound for the divergence of their cross product ($\theta^{\times}$) from the third axis (w) can be found using spherical trigonometry [30]:

$$\theta^{\times} = \arcsin\left(\sqrt{\sin^2\theta^{\langle u \rangle} + \sin^2\theta^{\langle v \rangle}}\right). \tag{10}$$

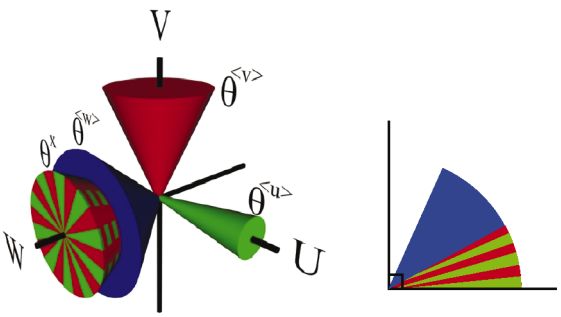

Fig. 5. The three-dimensional conic-hull hodograph of an injective FFD cell. Left: Bounding cones for the lattice edges ($\theta^{\langle u\rangle}, \theta^{\langle v\rangle}, \theta^{\langle w\rangle}$) and a cross product ($\theta^{\times}$) are established. Right: The two w-axis bounds ($\theta^{\langle w\rangle}, \theta^{\times}$) sum to less than 90 degrees.

- Dot Product: As long as two conic hulls around the same axis ($\theta^{\times}, \theta^{\langle w\rangle}$) diverge by less than 90 degrees ($\theta^{\times} + \theta^{\langle w\rangle} < \frac{\pi}{2}$), the dot product of any two vectors within those conic hulls will be positive and, hence, satisfy ϕ-injectivity.

3. The test halts (and reports noninjectivity) immediately on failing the conic hull test for any cell. Otherwise, the test continues until all cells have been processed (and injectivity is reported).

The conic hull hodograph of a ϕ-injectivite FFD is shown in Fig. 5 and a pseudocode version of the algorithm appears in the Appendix. If a faster (but less accurate) ϕ-test is desired, then, in Step 2, a single set of conic hulls can be found for the entire hyperpatch, rather than a set for every cell. Conversely, greater accuracy (at the expense of efficiency) can be achieved in two ways: 1) by recursively subdividing hyperpatch cells and applying the conic hull tests more locally and 2) by creating conic hulls which are not centered on the hyperpatch axes and bound the derivative vectors more tightly.

It is worth noting that the conic-hull hodograph injectivity test on a hyperpatch cell is $O(l \cdot m \cdot n)$ and well over three orders of magnitude faster than the original $O(l^3 \cdot m^3 \cdot n^3)$ ϕ-test from (7).

8 Adaptive Subdivision of Direct Manipulation

The conic-hull hodograph injectivity test, while efficient, severely limits the range of allowable Free-Form Deformations. However, it can be incorporated in DMFFD to produce an efficient and versatile variant which circumvents this problem. In principle, a set of direct manipulation constraints $(\mathbf{F}, \Delta\mathbf{F})$ is broken into shorter injective steps. This is a recursive procedure (as detailed in the Appendix), which, on failure of the conic-hull hodograph test, splits every constraint in the set into two adjoining pieces. A constraint, with point ($\mathbf{F}$) and displacement vector ($\Delta\mathbf{F}$) parts, is split into two halves $(\mathbf{F}, \frac{1}{2}\Delta\mathbf{F})$ and $(\mathbf{F} + \frac{1}{2}\Delta\mathbf{F}, \frac{1}{2}\Delta\mathbf{F})$, which join head to tail and are tested for injectivity separately. A maximum recursion depth can be set based on interactivity considerations and, if this is exceeded, then the direct manipulation is classified as illegal and self-intersecting. Otherwise, adaptive subdivision replaces a single ϕ-failing direct manipulation with an ordered collection of ϕ-injective direct manipulations which are applied independently and in sequence to collectively achieve the original constraints.

The advantages of this technique are as follows:

- The results are guaranteed to be injective (non-self-intersecting) if classified as such.
- The principle of adaptive subdivision is extensible to other point-based spatial deformation techniques.
- The range of DMFFD is expanded. The behavior of deformations is altered so that some pathological constraints (e.g., the overextended or overconstrained deformations of Fig. 3) no longer cause self-intersection.
- The deformation results are intuitive and, as can be seen in Fig. 7, mirror some of the properties of highly elastic modeling clay. A problem with the basic

Type of Deformation

Number of Vertices	1S	1M	1L	2S	2M	2L
145	0.0127s	0.0269s	0.0522s	0.0253s	0.0366s	0.0492s
545	0.0234s	0.0472s	0.0882s	0.0411s	0.0588s	0.0772s
2113	0.0623s	0.1244s	0.2204s	0.1039s	0.1486s	0.1937s

[1] Type 1 and 2 deformations are constraint scaled versions of Fig. 7[1C] and 7[2C] respectively. S = $\frac{1}{2}$ length constraint vectors, M = full constraint vectors, L = 1 $\frac{1}{2}$ length constraint vectors.
[2] Red entries represent update rates of less than 10 Hz.
[3] Timings taken on an SGI Octane (R10000 x 195 Mhz)
[4] The execution times for figures 7[1C] and 7[2C] are marked in blue

Fig. 6. Timings for a range of injective deformations applied to a polygon-mesh object at different resolutions.

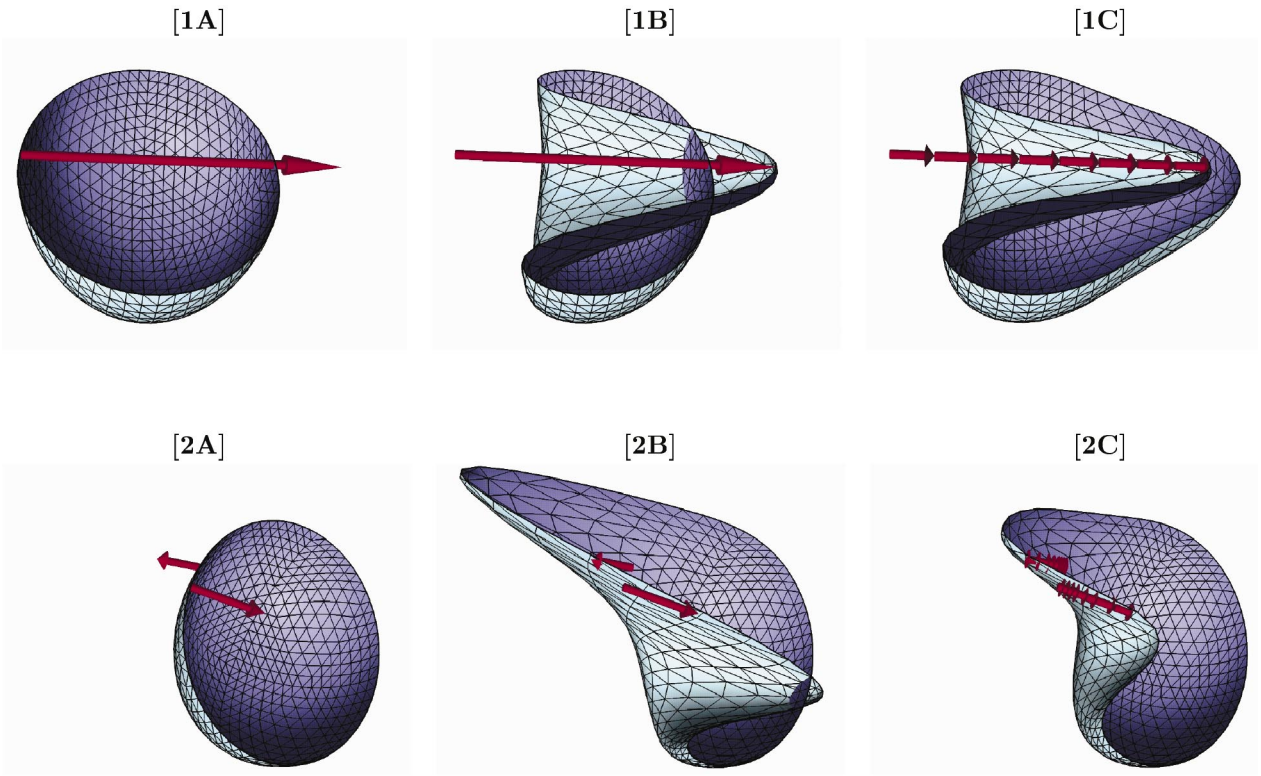

Fig. 7. Comparisons of injective and noninjective DMFFD. (1A) and (2A) Direct manipulation constraints (red) on a squashed hemisphere (blue). (1B) and (2B) Conventional noninjective DMFFD applied to these constraints causes self-intersection. (1C) and (2C) Injective DMFFD applied to adaptively subdivide these constraints prevents self-intersection.

DMFFD algorithm is that, in order to maintain continuity, certain portions of the hyperpatch may be distorted far more than the constraint points. This problem is significantly reduced by our technique.

- The timings in Fig. 6 show that the algorithm exhibits complex behavior that is highly dependent on the magnitude and type of deformations. However, as long as the number of direct manipulation constraints and recursive subdivision levels are within reasonable bounds, this technique can be applied in an interactive sculpting context.

Andersson et al. [2] consider cusp formation as well as singularities and self-intersection. In the context of injective DMFFD, cusps may form if the depth of the recursive subdivision is particularly great. We highlight this as an area for future consideration.

At each level of recursion, our injective DMFFD technique consistently splits every constraint in half. This works well in practice. As an alternative, the constraints could be split in different proportions at different depths of recursion or even within a single constraint set. In this way, the number of lattice changes (L) and, hence, deformation steps could be reduced. Also, certain extreme direct manipulations, for example, two constraints intersecting midway, could be prevented from causing self-intersection. The challenge, left for future research, is to develop an efficient algorithm for determining the optimal splitting proportions among constraints.

9 Conclusion

The principle contribution of this paper is a robust variant of Directly Manipulated Free-Form Deformation which prevents self-intersection by composing many small injective (one-to-one) deformations. This process entails decomposing direct manipulation constraints into injective steps. Injective DMFFD has advantages in efficiency, intuitivity, and versatility beyond guaranteeing non-self-intersection. The technique is suitable for interactive use, behaves with greater physical realism akin to manipulating highly malleable modeling putty, and expands the range of valid deformations by preventing overextension, where constraint points are dragged beyond their volume of influence, and overconstraint, where small constraint motion generates wild hyperpatch distortion.

Injective DMFFD depends on three novel and independently useful developments, namely:

- A set of conditions for enforcing injective deformations and, hence, preventing self-intersection. Although formulated in terms of FFD, this theory is applicable to all forms of spatial deformation.
- A precise (necessary and sufficient) FFD injectivity test which is accurate to within machine precision. This test is too computationally costly for inclusion in three-dimensional interactive sculpting, but it is suitable for two-dimensional interactive image warping.
- An approximate (merely sufficient) FFD injectivity test that relies on a geometric construction and is

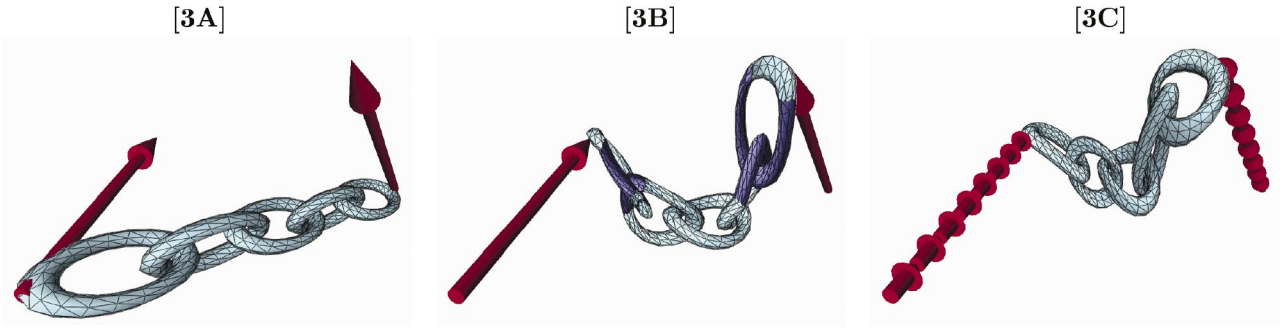

Fig. 8. A more complex comparison of injective and noninjective DMFFD. (3A) Direct manipulation constraints (red) on a rubbery chain object (blue) with separate non-overlapping links. (3B) Conventional noninjective DMFFD causes self-intersection. (3C) Injective DMFFD prevents self-intersection.

highly efficient. This test sacrifices the full range of injective deformations for improved performance and may falsely reject some valid deformations.

The research outlined in this paper is focused on detecting and preventing self-intersection under lattice-based spatial deformation. However, some of our results, specifically the injectivity conditions of Theorem 1 and the process of adaptively subdividing and then concatenating small injective deformations, are directly applicable to curve and point-based techniques, as well as certain forms of image warping [3]. The extension of these developments to cover all types of spatial deformation would thus be an area of fruitful future research.

APPENDIX

FindDeviation(t, a, b, c)
(Calculate the axial deviation of a vector)

IF $a \leq 0$ THEN
  RETURN "fail"
$t \leftarrow (b^2 + c^2)/a^2$

InjectiveTest($\Delta\mathbf{P}$)
(Test hyperpatch injectivity by forming three-dimensional conic-hull hodographs)

FOR all $\tilde{P}_{ijk}^{\langle u \rangle} = (a, b, c)$ DO
  FindDeviation($t_{ijk}^{\langle u \rangle}, a, b, c$)
FOR all $\tilde{P}_{ijk}^{\langle v \rangle} = (a, b, c)$ DO
  FindDeviation($t_{ijk}^{\langle v \rangle}, b, c, a$)
FOR all $\tilde{P}_{ijk}^{\langle w \rangle} = (a, b, c)$ DO
  FindDeviation($t_{ijk}^{\langle w \rangle}, c, a, b$)
FOR each lattice cell C
  FOR all $t^{\langle u \rangle}, t^{\langle v \rangle}, t^{\langle w \rangle}$ in C DO
    $m^{\langle u \rangle} = max(t^{\langle u \rangle})$, $m^{\langle v \rangle} = max(t^{\langle v \rangle})$
    $m^{\langle w \rangle} = max(t^{\langle w \rangle})$,
  $\theta^{\langle u \rangle} \leftarrow \arctan(\sqrt{m^{\langle u \rangle}})$, $\theta^{\langle v \rangle} \leftarrow \arctan(\sqrt{m^{\langle v \rangle}})$
  $\theta^{\langle w \rangle} \leftarrow \arctan(\sqrt{m^{\langle w \rangle}})$
  IF $(\theta^{\langle u \rangle} + \theta^{\langle v \rangle} \geq \frac{\pi}{2})$ THEN
    RETURN "noninjective"
  ELSE
    $\theta^{\times} \leftarrow \arcsin(\sqrt{\sin^2\theta^{\langle u \rangle} + \sin^2\theta^{\langle v \rangle}})$
    IF $(\theta^{\times} + \theta^{\langle w \rangle} \geq \frac{\pi}{2})$ THEN
      RETURN "noninjective"
  RETURN "injective"

InjectSubdiv($\mathbf{F}$, $\Delta\mathbf{F}$, L, d)
(Recursively subdivide (up to depth d) a set of direct manipulations $(\mathbf{F}, \Delta\mathbf{F})$ *into injective steps. If successful, the procedure returns a sequence of lattice changes (L))*

IF $d <$ maximum recursion depth THEN
  $\Delta\mathbf{P} = \mathbf{B}^{+}\Delta\mathbf{F}$
  IF InjectiveTest($\Delta\mathbf{P}$) = "injective" THEN
    Append $\Delta\mathbf{P}$ to L
  ELSE
    InjectSubdiv($\mathbf{F}$, $\frac{1}{2}\Delta\mathbf{F}$, L, $d+1$)
    InjectSubdiv($\mathbf{F} + \frac{1}{2}\Delta\mathbf{F}$, $\frac{1}{2}\Delta\mathbf{F}$, L, $d+1$)
ELSE
  STOP and RETURN "noninjective"

ACKNOWLEDGMENTS

This work was supported by the British Commonwealth Scholarship Commission and St. John's College, University of Cambridge. The authors are grateful to Malcolm Sabin for his "subversive" ideas, which helped to strengthen this research.

REFERENCES

[1] L.-E. Andersson, S.M. Dorney, T.J. Peters, and N.F. Stewart, "Polyhedral Peturbations that Preserve Topological Form," *Computer Aided Geometric Design*, vol. 12, no. 8, pp. 785-799, Dec. 1995.
[2] L.-E. Andersson, T.J. Peters, and N.F. Stewart, "Selfintersection of Composite Curves and Surfaces," *Computer Aided Geometric Design*, vol. 15, pp. 507-527, May 1998.
[3] N. Arad, N. Dyn, D. Reisfeld, and Y. Yeshurin, "Image Warping by Radial Basis Functions: Application to Facial Expressions," *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 2, pp. 161-172, Mar. 1994.
[4] F. Aubert and D. Bechmann, "Animation by Deformation of Space-Time Objects," *Computer Graphics Forum (Proc. Eurographics '97)*, vol. 16, no. 3, pp. 57-66, Sept. 1997.
[5] A.H. Barr, "Global and Local Deformations of Solid Primitives," *Computer Graphics (Proc. SIGGRAPH '84)*, vol. 18, no. 3, pp. 21-30, July 1984.

[6] R.H. Bartels, J.C. Beatty, and B.A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling.* Los Altos, Calif.: Morgan Kaufmann, 1987.

[7] P. Bézier, *Numerical Control: Mathematics and Applications.* Wiley, 1972.

[8] P. Borrel and D. Bechmann, "Deformation of *n*-Dimensional Objects," *Int'l J. Computational Geometry and Applications,* vol. 1, no. 4, pp. 427-453, 1991.

[9] P. Borrel and A. Rappoport, "Simple Constrained Deformations for Geometric Modelling and Interactive Design," *ACM Trans. Graphics,* vol. 13, no. 2, pp. 137-155, Apr. 1994.

[10] Y.-K. Chang and A.P. Rockwood, "A Generalized de Casteljau Approach to 3D Free-Form Deformation," *Computer Graphics (Proc. SIGGRAPH '94),* pp. 257-260, July 1994.

[11] S. Coquillart, "Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling," *Computer Graphics (Proc. SIGGRAPH '90),* vol. 24, no. 4, pp. 187-196, Aug. 1990.

[12] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide,* fourth ed. San Diego, Calif.: Academic Press, 1997.

[13] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice,* second ed. New York: Addison-Wesley, 1991.

[14] J.E. Gain and N.A. Dodgson, "Adaptive Refinement and Decimation under Free-Form Deformation," *Proc. Eurographics UK '99,* pp. 7-18, Apr. 1999.

[15] J. Gain, "Enhancing Spatial Deformation for Virtual Sculpting," Technical Report 499, Computer Laboratory, Univ. of Cambridge, U.K., Aug. 2000.

[16] J. Greissmair and W. Purgathofer, "Deformation of Solids with Trivariate B-Splines," *Proc. Eurographics '89,* pp. 137-148, Sept. 1989.

[17] M. Hardwick, D.L. Spooner, T. Rando, and K.C. Morris, "Sharing Manufacturing Information in Virtual Enterprises," *Comm. ACM,* vol. 39, no. 2, pp. 46-54, Feb. 1996.

[18] C. Hoffmann, *Geometric and Solid Modeling: An Introduction.* San Mateo, Calif.: Morgan Kaufmann, 1989.

[19] W.M. Hsu, J.F. Hughes, and H. Kaufman, "Direct Manipulation of Free-Form Deformations," *Computer Graphics (Proc. SIGGRAPH '92),* vol. 26, no. 2, pp. 177-184, July 1992.

[20] K. Joy and M. Duchaineau, "Boundary Determination for Trivariate Solids," *Proc. Pacific Graphics '99,* pp. 82-91, Oct. 1999.

[21] F. Lazarus, S. Coquillart, P. Jancène, "Axial Deformations: An Intuitive Deformation Technique," *Computer-Aided Design,* vol. 26, no. 8, pp. 607-613, Aug. 1994.

[22] R. MacCracken and K.I. Joy, "Free-Form Deformations with Lattices of Arbitrary Topology," *Computer Graphics (Proc. SIGGRAPH '96),* pp. 181-188, Aug. 1996.

[23] T. Maekawa, N.M. Patrikalakis, T. Sakkalis, and G. Yu, "Analysis and Application of Pipe Surfaces," *Computer-Aided Geometric Design,* vol. 15, no. 5, pp. 437-458, May 1998.

[24] G. Meisters and C. Olech, "Locally One-to-One Mappings and a Classical Theorem on the Schlicht Functions," *Duke Math. J.,* vol. 30, pp. 63-80, Mar. 1963.

[25] K. Mørken, "Some Identities for Products and Degree Raising of Splines," *Constructive Approximation,* vol. 7, no. 2, pp. 195-208, Apr.-June 1991.

[26] U.M. Nimscheck, "Rendering for Free-Form Deformations," Technical Report 381, Computer Laboratory, Univ. of Cambridge, U.K., Oct. 1995.

[27] B. Noble, *Applied Linear Algebra.* Englewood Cliffs, N.J. : Prentice Hall, 1969.

[28] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing,* second ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[29] T.W. Sederberg and S.R. Parry, "Free-Form Deformation of Solid Geometric Models," *Computer Graphics (Proc. SIGGRAPH '86),* vol. 20, no. 4, pp. 151-160, Aug. 1986.

[30] T.W. Sederberg and R.J. Meyers, "Loop Detection in Surface Patch Intersections," *Computer Aided Geometric Design,* vol. 5, no. 2, pp. 161-171, July 1988.

[31] K. Singh and E. Fiume, "Wires: A Geometric Deformation Technique," *Computer Graphics (Proc. SIGGRAPH '98),* pp. 405-414, July 1998.

[32] M.R. Spiegel, *Theory and Problems of Advanced Calculus.* McGraw-Hill, 1987.

[33] N.F. Stewart, "Sufficient Condition for Correct Topological Form in Tolerance Specification," *Computer Aided Design,* vol. 25, no. 1, pp. 39-48, Jan. 1993.

James Gain received the BSc (Hons) and MSc degrees in computer science from Rhodes University, South Africa, in 1994 and 1996, respectively. In 2000, he obtained the PhD degree, entitled "Enhancing Spatial Deformation for Virtual Sculpting," from the University of Cambridge. He is currently a senior lecturer in the Computer Science Department at the University of Cape Town, South Africa, and a member of its Collaborative Visual Computing Laboratory.

Neil Dodgson received the BSc degree in physics and computer science from Massey University, New Zealand, in 1988 and the PhD degree from the University of Cambridge in 1992. He is currently a university lecturer in the Computer Laboratory at Cambridge, where he coleads a research group in computer graphics and image processing. Dr. Dodgson is a fellow of Emmanuel College, Cambridge, a member of the IEE, and a Chartered Engineer.