Technical Report

UCAM-CL-TR-499
ISSN 1476-2986

Number 499

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Enhancing spatial deformation for virtual sculpting

James Edward Gain

August 2000

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted June 2000 by the author for the degree of Doctor of Philosophy to the University of Cambridge, St John's College.

Pages ii–iv and vi have been removed from this Technical Report to save space; they contained only a formal declaration relating to the PhD submission or were blank.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/TechReports/

ISSN 1476-2986

Abstract

The task of computer-based free-form shape design is fraught with practical and conceptual difficulties. Incorporating elements of traditional clay sculpting has long been recognised as a means of shielding a user from the complexities inherent in this form of modelling. The premise is to deform a mathematically-defined solid in a fashion that loosely simulates the physical moulding of an inelastic substance, such as modelling clay or silicone putty. Virtual sculpting combines this emulation of clay sculpting with interactive feedback.

Spatial deformations are a class of powerful modelling techniques well suited to virtual sculpting. They indirectly reshape an object by warping the surrounding space. This is analogous to embedding a flexible shape within a lump of jelly and then causing distortions by flexing the jelly. The user controls spatial deformations by manipulating points, curves or a volumetric hyperpatch. Directly Manipulated Free-Form Deformation (DMFFD), in particular, merges the hyperpatch- and point-based approaches and allows the user to pick and drag object points directly.

This thesis embodies four enhancements to the versatility and validity of spatial deformation:

1. We enable users to specify deformations by manipulating the normal vector and tangent plane at a point. A first derivative frame can be tilted, twisted and scaled to cause a corresponding distortion in both the ambient space and inset object. This enhanced control is accomplished by extending previous work on bivariate surfaces to trivariate hyperpatches.
2. We extend DMFFD to enable curve manipulation by exploiting functional composition and degree reduction. Although the resulting curve-composed DMFFD introduces some modest and bounded approximation, it is superior to previous curve-based schemes in other respects. Our technique combines all three forms of spatial deformation (hyperpatch, point and curve), can maintain any desired degree of derivative continuity, is amenable to the automatic detection and prevention of self-intersection, and achieves interactive update rates over the entire deformation cycle.
3. The approximation quality of a polygon-mesh object frequently degrades under spatial deformation to become either oversaturated or undersaturated with polygons. We have devised an efficient adaptive mesh refinement and decimation scheme. Our novel contributions include: incorporating fully symmetrical decimation, reducing the computation cost of the refinement/decimation trigger, catering for boundary and crease edges, and dealing with sampling problems.
4. The potential self-intersection of an object is a serious weakness in spatial deformation. We have developed a variant of DMFFD which guards against self-intersection by subdividing manipulations into injective (one-to-one) mappings. This depends on three novel contributions: analytic conditions for identifying self-intersection, and two injectivity tests (one exact but computationally costly and the other approximate but efficient).

Acknowledgements

Two individuals have been instrumental to this work. My supervisor, Neil Dodgson, has steered me through difficult times with his insightful comments and thorough proofreading, and the "subversive" ideas provided by Malcolm Sabin have helped to broaden my research.

The members of the Rainbow Graphics Group have ensured that working in the Computer Laboratory is, in about equal measure, stimulating, fruitful and entertaining.

Thanks also are due to my proofreading team: Neil Dodgson, Malcolm Sabin, Jonathan Pfautz and Ingrid van Eck-Gain. Tony Polichroniadis helped me by providing Yoda with his "skin".

I acknowledge the financial support of the Association of Commonwealth Universities, St. John's College and the Computer Laboratory at the University of Cambridge.

I am thankful to my far-flung family who have always offered enthusiastic support. In particular, the value that my parents placed on education set me on this path in the first place.

Finally, my deepest gratitude is to Ingrid. Without her unstinting financial, intellectual and emotional support this thesis would never have been completed.

Trademark Information

The following are registered trademarks: Open Inventor, SGI Octane, Polhemus, Plasticine and Yoda.

Contents

Chapter 1

Introduction

1.1 Motivation

Free-form modelling is a significant subdiscipline of computer graphics concerned with the computer-based design of three-dimensional shapes. A primary consideration is the aesthetics of the final shape, rather than functional aspects such as aerodynamics, volume or stress response. The field has diverse applications, ranging from exploratory design (rapidly sketching a rough initial model) to visual effects (the creation of animated characters). There are two broad categories of free-form design: surface modelling, which manipulates the three-dimensional skin of an object (typically represented by parametric patches or a polygon mesh) and solid modelling, which deals with the complete, unambiguous encoding of volumetric objects, where both the interior and surface are regarded as important.

Unfortunately, the task of free-form modelling is beset with practical and conceptual usability difficulties to such an extent that many designers still rely heavily on traditional non-automated techniques. The visual effects industry is a case in point. In films such as "Godzilla" [Martin 1998], "A Bug's Life" [Vaz 1999] and "Star Wars: Episode I" [Duncan, Martin and Vaz 1999], the animated characters are modelled in three stages: a clay maquette (preliminary model) is sculpted by hand, this is scanned to produce a digital point set, and an interpolating mesh is extracted. This process, which requires time, skill and relatively expensive equipment, is illustrated in figure 1.1. It would be more productive, especially if a model required modification, for the entire cycle to be computerised.

Virtual sculpting has long been touted [Parent 1977; Sederberg and Parry 1986] as a natural and intuitive solution to the complex task of free-form modelling. The intention is to link the familiar physical action of moulding and manipulating clay with the difficult (and often unfamiliar) process of computerised shape design. This is achieved by loosely emulating the deformation of an inelastic substance, such as modelling clay or silicone putty. The accurate simulation of clay sculpting must be balanced against the need for interactive feedback.

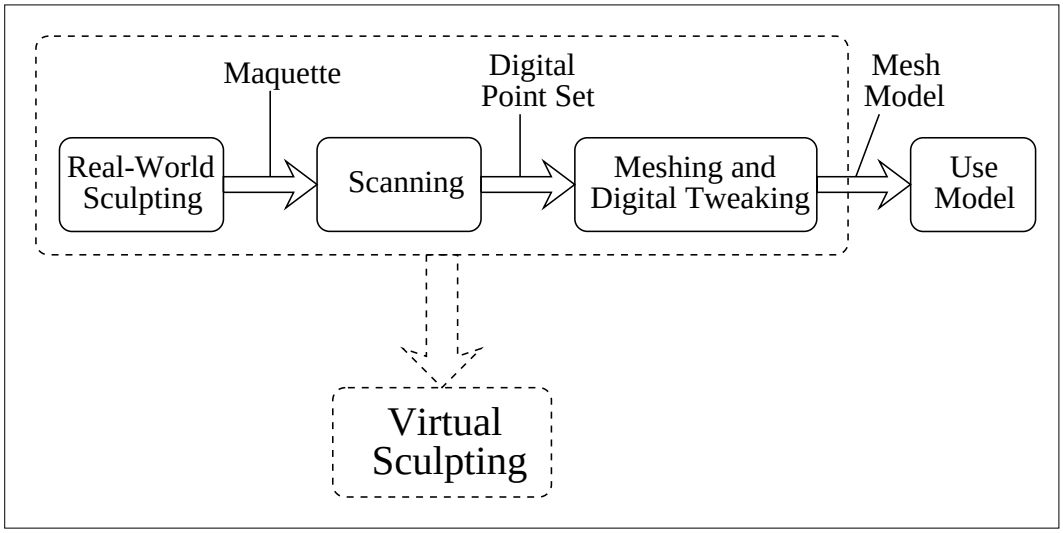


Figure 1.1: **The Free-Form Modelling Process. Virtual sculpting offers an alternative to the process of physical sculpting, digital scanning and meshing currently employed in the visual effects industry.**

The virtual sculpting approach is motivated in part by the success of paint programs in two-dimensional design. Rossignac [1994] attributes their popularity to predictability, use of natural gestures and immediate response. They also employ painting or sketching as a cohesive user-interface metaphor. These characteristics can valuably be extended to encompass three-dimensional design. Another impetus is the maturation of three-dimensional input and output devices such as position trackers and stereoscopic displays. The development of direct manipulation techniques can now be matched by the user-interface equipment. Finally, virtual sculpting can shield a designer from the internal mathematical representation of objects. This enables an artist to rapidly master an unfamiliar modelling environment.

There is a considerable demand for free-form models in computer graphics, and virtual sculpting is potentially a powerful strategy for meeting this need.

1.2 Prior Art

The concept of virtual sculpting is over two decades old. Parent [1977] discusses "*an attempt at the creation of a sculptor's studio-like environment, in which the 'sculptor' can create complex 3D objects in the computer, as if moulding a piece of clay.*" Since its inception a variety of techniques have been developed in support of virtual sculpting.

Decay Functions. Parent [1977] introduced the use of decay functions in surface modelling. Here, the user repositions a single key vertex and a decreasing proportion of its movement is

distributed across the mesh. The displacement decays as a simple power function of the surface distance, which is measured by the expedient of counting the edges in the shortest path between a given vertex and the key vertex. Allan, Wyvill and Witten [1989] dramatically extend decay functions by: employing a Euclidean measure of surface distance; defining a variety of decay shapes (cone, bell, cusp, random, sine); and allowing vertices to be directly bound to the key's movement or anchored in place. Finally, Bill and Lodha [1994] devise superellipsoidal tools (ranging from rounded boxes and cylinders, to ellipses and spheres), which can simultaneously push or pull many key vertices.

Boolean Set Operations. Combining polygon-mesh objects by means of the regularised boolean set operations (union, intersection and difference) is a staple of solid modelling. The carving or merging of shapes can be simulated, with the limitations that set operations on complex objects are costly and the results tend to have sharp rather than rounded edges. Parent [1977] and Naylor [1990] adapt the boolean set operations to virtual sculpting. They accept the lack of smoothness and overcome much of the computation cost by ensuring that one of the argument objects (the tool) is a simple convex polyhedron.

Spatial Deformation. The premise behind spatial deformation is to indirectly affect an object by warping its ambient space. A correspondence must be created between the enclosing space and inset object so that distortions are transferred from space to object. Bézier [1978] and Sederberg and Parry [1986] achieve this by imposing a hyperpatch (three-dimensional parametric function) onto co-ordinate space. A lattice of control points enables deformation of the hyperpatch and indirectly any embedded object whose vertices have been linked to it. Borrel and Bechmann [1991] and Lazarus, Coquillart and Jancéne [1994] replace the control point mechanism with deformations that interpolate the motion of arbitrary points and curves. Spatial deformation has the virtues of relative efficiency, independence from the underlying object representation, and either localised or global volumetric effect.

Voxels. A free-form solid can be approximately represented as a collection of cube-shaped volume elements (voxels). These are an extension of two-dimensional picture element squares (pixels) to three-dimensions. An object is stored as a voxmap which encodes values at the interstices of a regular lattice (with 1.0 representing the interior and 0.0 the exterior). Sculpting tools can be applied to locally modify the voxmap and consequently the shape that it represents, much as "paintbrush" tools affect a pixmap in traditional 2D paint programs. Galyean and Hughes [1991] develop subtractive (material removing), additive (material accreting) and averaging (smoothing) tools, which are centred on a single point and aligned with the object voxmap. Wang and Kaufman [1995] support more complex carving and sawing tools, which can be rotated with respect to the object. The principle difficulties with voxel sculpting are the aliasing artefacts arising from the discrete nature of voxels (typically reduced with low-pass filters) and the high cost of rendering (generally overcome with local incremental updates). Voxel models have large computation and storage overheads but support highly complex topologies.

Parametric Patches. Boolean set operations can be considered the primary technique within solid modelling [Mäntylä 1988, p. 263]. Similarly, free-form parametric patches are the most mature and widespread entities in surface modelling [Farin 1997, ch. 15-17]. Here, an object is

constructed quilt-like (piecewise) from triangular (three-sided) or rectangular (four-sided) Bézier or B-spline patches, which join along their edges. Care must be taken to ensure smoothness (continuity) along the joins between patches. The user interacts with the object through a net of control points which either approximate or interpolate the surface. Brewer and Anderson [1977], for instance, support the use of Overhauser cubics in interactive 3D design because the resulting surface passes through (interpolating) rather than nearby (approximating) the defining control points. Unfortunately, as explained by Farin [1997, p. 317], parametric patches are not suited to objects with non-planar topology. This is a serious limitation. It is impossible, for example, to model even a simple C^1 continuous sphere from rectangular patches without introducing degenerate patches.

Subdivision Surfaces. The topology limitations of parametric patches are overcome by recursive subdivision surfaces. An arbitrary open or closed polyhedron (the control net) is iteratively subdivided, introducing additional faces, edges, and vertices, according to a set of refinement rules. Doo and Sabin [1978] and Catmull and Clark [1978] devised eponymous subdivision schemes, which respectively generate uniform biquadratic and bicubic B-spline surfaces, except in the neighbourhood of certain extraordinary points. Depending on the scheme, each n-valent ($n \neq 4$) face or vertex is responsible for an extraordinary point on the final surface. Subdivision surfaces are capable of complex topology and regions of variable detail. They combine well with polygon-mesh sculpting (decay functions, set operations, and spatial deformation), which is useful in designing the initial control mesh. DeRose, Kass and Truong [1998] have demonstrated the suitability of subdivision surfaces for modelling and animation in a high-end production environment.

Physically-Based Deformation. An accurate physical simulation of inelastic deformation has the potential for unprecedented realism in virtual sculpting. Terzopoulos and Fleischer [1988] describe an appealing scenario. They envisage: *"users, aided by stereoscopic and haptic input-output devices, carving 'computer plasticine' and applying simulated forces to it in order to create free-form shapes interactively."* There is a rich body of theory to draw on in this quest for dynamically-driven sculpting. Recently, mass-spring [Dachille, Qin, Kaufman and El-Sana 1999], finite element [Bro-Nielsen and Cotin 1996] and boundary element [James and Pai 1999] methods have been employed in real-time deformation. The challenge, met thus far with only mixed success, according to James and Pai [1999], is to combine speed and low latency with physical accuracy.

1.3 Requirements

A set of requirements for effective virtual sculpting can be synthesized from several sources. Nielsen [1994] provides a list of heuristics obtained from the study of 249 usability problems. These form part of heuristic evaluation [Molich and Nielsen 1990] in which a panel of experts analyse the usability of a system. The cognitive dimensions framework [Green 1989] is a more general (but less precise) technique aimed at establishing a working vocabulary for trade-off

discussions. Green derives a set of dimensions (usability attributes) from cognitive psychology. Both of these evaluation methods are heavily reliant on the principles of direct manipulation [Schneiderman 1983], which shifts the focus from commands to the object of the user's action. In all three cases, the guiding principles represent absolutes that are not always attainable. Specifically, improving one attribute is likely to have a negative impact on others. Direct manipulation, heuristic evaluation and cognitive dimensions together suggest several guidelines for exploratory design. We group these under four headings:

1. **Ease of Use.** This covers the ease with which users attain and retain proficiency in virtual sculpting. In striving for ease of use, unnecessary conceptual difficulties in understanding and remembering system operations must be avoided. There are a number of ways in which such system complexities can be reduced:

   (a) **Closeness of Mapping.** The solution, embodied by the sculpting system, should be closely matched to the real-world problem of free-form shape design. This allows artists to directly apply their experience with physical modelling to the unfamiliar task of virtual sculpting.

   (b) **Simplicity.** A simple uncluttered interaction mechanism promotes ease of use. It is important to structure operations in a streamlined fashion, avoid unnecessary detail and create components whose purpose can be readily inferred.

   (c) **Consistency.** It is worth enforcing conventions throughout the system so that similar operations are expressed in a uniform way. Conversely, to avoid confusion, dissimilar operations should not have the same form.

   (d) **Flexibility.** The sculpting system should adapt to the familiarity of the user. Experienced users may want to tailor frequent operations and develop shortcuts. On the other hand, novice users must first learn the core functionality.

   For the purposes of this thesis, ease of use is evaluated in a discursive and qualitative manner. A detailed experimental evaluation is deferred to future work.

2. **Efficiency.** Interactive feedback is crucial to the success of virtual sculpting. The principle of visibility demands that users are rapidly informed through visual (but perhaps also auditory and tactile) feedback as to the effects of their actions. The response-time of the system should be such that delays do not hinder creative exploratory design. We deem a rate of ten updates per second (10 Hz) to be the lowest acceptable limit on interactive response for design purposes. This threshold is chosen for comparison purposes only, it does not represent a limitation on system behaviour. The performance of different sculpting algorithms is measured relative to this interactivity threshold in two ways:

   (a) **Operation Counting.** For any given algorithm the number of floating point operations of each type (addition/subtraction, multiplication, division, square root, sine/cosine) are calculated. The operation counts for each type are then scaled by a relative cost and included in the total complexity (C):

$$C = \sum_{\forall \text{ op types}(i)} \text{count}_i \times \text{cost}_i$$

Addition/Subtraction	Multiplication	Division	Square Root	Sine/Cosine
1.0	1.614	16.068	31.094	26.687

Table 1.1: **Relative Floating Point Operation Costs. These weights are derived from timing experiments on an SGI Octane** 195 **MHz R**10000**.**

Table 1.1 shows the operation costs (normalised with respect to addition/subtraction) for an SGI Octane. As a rough guide the SGI Octane is capable of approximately 14.5×10^7 f.p. additions per second. Although operation counting does not consider loop, comparison, function call, memory access, and ancillary overheads, it is useful in comparing the relative performance of algorithms without requiring actual machine implementation.

(b) **Execution Timing.** Accurate timing (typically with millisecond precision) can be achieved by polling the system clock before and after executing an algorithm. Atypical timing fluctuations can be accounted for by averaging the timings over many algorithm executions. In this dissertation, execution timings are taken on an SGI Octane with a 195MHz R10000 processor.

3. **Versatility.** The user should be able to convert intentions and aesthetic requirements into a designed result with ease and precision. This requires sculpting operations that are varied (produce a wide range of different shapes) and powerful (achieve major shape modifications in a direct and straightforward fashion). Truly versatile sculpting operations also display the following characteristics:

   (a) **Fluidity.** The sculpting system should not resist shape modifications, be they widespread or local. Instead, changes should be easy to specify and instigate, allowing a smooth and swift transition between diverse shapes.

   (b) **Reversibility.** An 'undo' operation must always be supported. It speeds up design and encourages experimentation by providing users with an assurance that no mistake is irrevocable.

   (c) **Order Independence.** Sculpting operations should be allowed in any order. This frees the user from having to plan a series of modelling steps in advance.

4. **Validity.** Neither the internal mathematical correctness nor the external realism of a shape should be compromised by the application of virtual sculpting methods. Self-intersection is a case in point: it contravenes the manifold object representation and is also physically infeasible. It is unreasonable to expect the user to identify errors. Rather, the system should automatically recognise and recover from breaches in validity. Better yet, sculpting operations could be structured to prevent errors from occurring in the first place.

1.4 Thesis

The research embodied by this dissertation is focused on the use of spatial deformation in virtual sculpting. This technique was selected, in preference to others, due to its balance of efficiency and modelling power. Spatial deformation, as highlighted by Sederberg and Parry [1986], effectively combines surface and solid modelling, acts independently of the underlying object representation, allows both local and global distortions, and is capable of interactive performance on relatively complex objects.

The objectives of this thesis can now be stated in terms of the four earlier requirements:

Research Objectives: To enhance in a balanced fashion the ease of use, efficiency, versatility and validity of spatial deformation within a virtual sculpting context.

In effect, these requirements are woven into the very fabric of our research. They underpin our critical comparison of spatial deformation methods, form a basis for evaluating the improvements undertaken in this thesis, and provide inspiration for future work.

1.5 Dissertation Structure

This dissertation documents progress towards the stated research objectives, as follows:

- *Chapter 2 [Literature Survey]* examines previous contributions to the field of spatial deformation. The various deformation methods are characterised as hyperpatch-, point- or curve-based, depending on their interaction mechanism. Hyperpatch-based deformations define the shape of a parametric volume and any embedded object by means of a lattice of control points; point-based deformations conform to the motion of sample points selected and then repositioned by the user; and curve-based deformations similarly interpolate the motion of a set of user-defined curves. The different schemes are also compared and contrasted. Finally, the overall shortcomings of spatial deformation are identified.

- *Chapter 3 [Foundations]* outlines the fundamentals of Directly Manipulated Free-Form Deformation (DMFFD) [Hsu, Hughes and Kaufman 1992], a technique which melds hyperpatch- and point-based spatial deformation, balances efficiency with modelling power, and provides an effective research foundation. DMFFD is presented in two tiers: Free-Form Deformation alters the position and derivative vectors of sample points on an object to match the distortions of an encompassing volumetric hyperpatch, and Direct Manipulation converts point manipulations, provided by the user as a set of spatial positions and their intended displacements, into interpolating hyperpatch alterations.

- In *Chapter 4 [Derivative-Frame Manipulation]*, a new manipulation mechanism is devised for DMFFD. This builds on work by Fowler [1992] on the derivative manipulation of bivariate surfaces. We link derivative-frame changes to a hyperpatch. The normal vector and tangent plane at a point, represented graphically by an arrow and planar rectangle, can be tilted, scaled and twisted to bring about a corresponding transformation of the ambient hyperpatch and inset object.

- *Chapter 5 [Adaptive Refinement and Decimation]* details an adaptive meshing scheme designed to counteract the degradation in mesh quality caused by spatial deformation. The characteristics of successful subdivision are established and employed in the evaluation of previous approaches to refinement under deformation. Two previous schemes, due to Greissmair and Purgathofer [1989] and Nimscheck [1995], are combined and extended into a novel refinement method, which incorporates fully symmetrical decimation, reduces the cost of testing for refinement, caters for boundary and crease edges, and deals with sampling issues.

- In *Chapter 6 [Curve Manipulation]*, we improve the versatility of DMFFD by developing a manipulation technique which maps source curves onto destination curves through contortions of the underlying hyperpatch. The chapter begins by presenting background theory on blossoming (a compact, geometrically intuitive means of defining Bézier and B-spline curves), trivariate functional composition (embedding curves within a hyperpatch), and degree reduction (decreasing the degree of Bézier curves). Then a novel means of binding curve control points through composition and degree reduction to hyperpatch control points is presented. At the expense of a small measure of approximation error this technique merges hyperpatch-, point- and curve-based deformation, achieves interactive update rates, maintains any degree of continuity, and is amenable to the automatic detection and prevention of self-intersection.

- *Chapter 7 [Preventing Self-Intersection]* is devoted to overcoming the validity problems brought on by self-intersection. The theoretical conditions for determining the self-intersection status of a hyperpatch mapping are documented. From these, an exact (necessary and sufficient) but computationally costly test for FFD self-intersection is derived. Subsequently, a fast but approximate (merely sufficient) self-intersection test is promulgated. These novel developments lead to a powerful variant of DMFFD, which guards against self-intersection by subdividing manipulations.

- Finally, in *Chapter 8 [Future Work]* and *Chapter 9 [Conclusions]*, a series of potential improvements to the ease of use and versatility of spatial deformation are proposed for future study, and the content and contribution of this thesis are summarised.

Chapter 2

A Survey of Spatial Deformation Techniques

2.1 Introduction

The spatial deformation methods are a family of highly interactive, powerful and intuitive modelling techniques, which indirectly deform an object by warping its ambient space. This is analogous [Sederberg and Parry 1986] to setting a flexible shape inside a lump of jelly with the same consistency and then flexing this jelly, resulting in a corresponding distortion of the embedded shape. All spatial deformations exhibit the following characteristics:

- **Mathematical Formalism.** Spatial deformation can be formulated as a mapping $\mathcal{F} : \Re^3 \mapsto \Re^m \mapsto \Re^3$, from world space, $X = (x, y, z)$, through a local parameter space, U, to deformed world space, $\widetilde{X} = (\widetilde{x}, \widetilde{y}, \widetilde{z})$. The parameter space U, of dimension m, is defined by the deformation tools of a particular technique. 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:

  $$\mathcal{F}(X) = \mathcal{D} \circ \mathcal{E}(X) = \mathcal{D}(\mathcal{E}(X)) = \mathcal{D}(U) = \widetilde{X} \tag{2.1}$$

  The spatial deformation mapping can be generalised to higher dimensions $\mathcal{F} : \Re^n \mapsto \Re^m \mapsto \Re^n$. In particular, Bechmann and Dubreuil [1993] exploit $\Re^4$ to animate deformations in both space and time. Equation 2.1 is a generalisation of the formalism first presented by Bechmann [1994] in her survey of spatial deformation methods.

  The encompassing formalism of eqn. 2.1 remains somewhat abstract but it will be made more concrete in the survey of techniques that follows.

- **Algorithm.** The deformation of an object can be split into four generic steps:

1. The user defines the initial embedding state of a set of deformation tools with consideration given to the original shape of the object. These tools, be they curves or constraint points, provide handles for manipulating ambient space.
2. A collection of sample points taken from the object's surface are embedded in the tool parameter space by repeated application of the function $\mathcal{E}(X) = U$.
3. The tools are altered by the user into a deforming configuration. The difference between the deformed and embedded states of the toolset defines a distortion of ambient space.
4. The altered position of every sample point is calculated by using the previously determined parameters (U) and the deformation function $\mathcal{D}(U) = \tilde{X}$.

- **Representation Independence.** These deformations, because they operate only on sampled points, are independent of the underlying representation of the embedded object. A spatial deformation can be applied as long as surface points can be extracted from an object. For example, Constructive Solid Geometry [Parry 1986], superquadrics [Güdükbay and Özguç 1990], and polygon-mesh models [Sederberg and Parry 1986] can all undergo spatial deformation. There is a proviso here: according to Bechmann [1994], a parametric or implicit surface formulation is lost when deforming sample points. This is true in general, but Nimscheck [1995] discovered a composition method which retains the parametric representation of Bézier curves, albeit with a significantly raised degree, under Free-Form Deformation. In practice objects are converted into a triangle-mesh format prior to deformation. This representation is amenable to polygon scan conversion rendering [Foley *et al.* 1991, pp. 92-99] and stays valid after any number of spatial deformations, unlike meshes with higher order polygons (having more than three edges) which may not remain planar.

- **Variable Scope.** A spatial deformation can be either global or local in scope. In practical terms, either the entire object or only a portion falls within the demarcated deformed space. An important consideration for local deformations is the maintenance of a desired degree of surface continuity between the deformed and undeformed portions of the object.

- **Sequential Application.** The shape of an object can be gradually refined through a series of spatial deformations. This has some similarities to the sequential application of affine transformations (rotation, scaling, shear, etc.): both are $\Re^3 \mapsto \Re^3$ mappings where ordering is significant. If two deformations ($\mathcal{F}_1$ and $\mathcal{F}_2$) affect the surface of an object and have overlapping influence then a different ordering ($\mathcal{F}_1 \circ \mathcal{F}_2$ or $\mathcal{F}_2 \circ \mathcal{F}_1$) will generally produce different results. In other respects they are dissimilar. For instance, a sequence of spatial deformations, unlike affine transformations, cannot, in general, be composed into a single operation.

- **Topological Invariance.** 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). In seeming contradiction of this property, Aubert and Bechmann [1997b] 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. However, even though the $\Re^3$ object may develop holes and split into separate components, this is as a consequence of the extrusion and cross-sectioning process. The actual spatial deformation in $\Re^4$ remains topology preserving.

Chapter Structure

Current contributions to the field of spatial deformation can be grouped under three headings:

1. **Hyperpatch-based Deformations.** Univariate curves can be extended to bivariate surfaces and from there to trivariate hyperpatches. These hyperpatches demarcate a volume whose shape is altered by repositioning control points. The term Free-Form Deformation (FFD) is given in the literature to any technique which binds the distortion of a hyperpatch to an embedded solid.
2. **Point-based Deformations.** Under these schemes the user provides a set of constraints, 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 constraints.
3. **Curve-based Deformations.** This category consists of interpolating deformations which are controlled by the motion of one or more curve constraints each consisting of a source and destination curve which may affect either the entire space or a specified enclosing volume.

Under these headings, the mathematical basis, algorithm and user interface of each deformation technique is discussed. This is followed by an evaluation section in which all the techniques are assessed according to the following criteria (from chapter 1):

1. **Ease of Use.** The interfaces to these deformation schemes are compared 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 modelling proficiency and the extent to which they match attributes of real world sculpting.
2. **Efficiency.** The computation cost of the different methods are quantified according to the metric presented in chapter 1. Memory consumption is also considered insofar as it impacts on performance.
3. **Versatility.** 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.
4. **Correctness.** The circumstances under which spatial deformations violate the validity of an embedded object are considered in detail.

2.2 Types of Spatial Deformation

2.2.1 Hyperpatch-Based

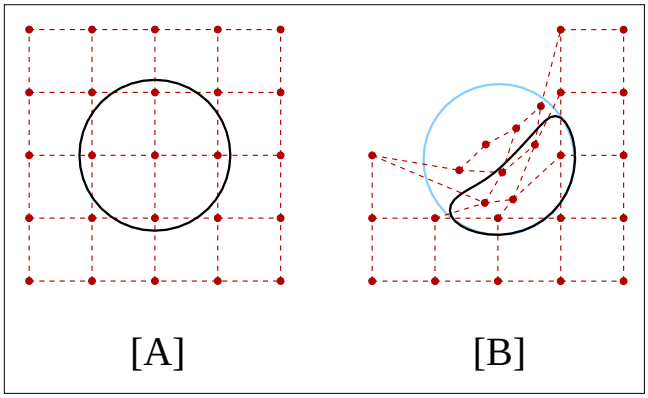


Figure 2.1: **Two-Dimensional Hyperpatch-Based Deformation. [A] Pre-deformation: a lattice of control points (red) surrounds an object (black). [B] Post-deformation: changes to the lattice induce a deformation of the object.**

Hyperpatch-based spatial deformation, originally outlined by Bézier [1978] and since termed Free-Form Deformation (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 visualised as a pliable plastic block [Sederberg and Parry 1986] capable of twisting, bending and bulging under the influence of a set of control points. Mathematically, it 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, generalises to a control lattice. Here lattice edges show the trivariate relationship between control points in three axial directions. A user initiates deformation by moving these control points, and consequently reshaping the hyperpatch.

Points 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:

$$\mathcal{H}(U) = \mathcal{H}(u,v,w) = \sum_{i=0}^{a+\ell-1} \sum_{j=0}^{b+m-1} \sum_{k=0}^{c+n-1} \mathcal{N}_i^\ell(u) \cdot \mathcal{N}_j^m(v) \cdot \mathcal{N}_k^n(w) \cdot P_{i,j,k} = X \qquad (2.2)$$

Equation 2.2 provides a mapping from a parallelepiped domain, e.g. $u, v, w \in [0,1]$, in the $\Re^3$ parameter space of the hyperpatch, to conventional world co-ordinate space with $3 \cdot (a+\ell) \cdot (b+m) \cdot (c+n)$ degrees of control freedom available to the user. Now, consider two specific configurations of control points, pre-deformation P and post-deformation $\widetilde{P}$, and the corresponding instantiations of eqn. 2.2, $\mathcal{H}(U) = X$ and $\widetilde{\mathcal{H}}(U) = \widetilde{X}$. In accordance with the encompassing formalism (eqn. 2.1), hyperpatch-based deformation 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, $\mathcal{D}(U) = \widetilde{\mathcal{H}}(U) = \widetilde{X}$.

With this background in place, the FFD algorithm proceeds in four stages:

1. 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 mould.
2. Object vertices which fall within the undistorted hyperpatch are assigned parametric (u, v, w) co-ordinates ($\mathcal{E} = \mathcal{H}^{-1}$ is applied). Metaphorically, the shape being deformed is placed within the jelly.
3. A number of control points are displaced by the user (P becomes $\widetilde{P}$), with a resulting distortion of the hyperpatch. This equates to flexing the jelly.
4. The post-deformation version of eqn. 2.2 ($\mathcal{D} = \widetilde{\mathcal{H}}$) is applied repeatedly to all of the parametrised object vertices to produce a deformed image of the object. By analogy, the inset shape is warped along with its cocooning jelly.

In the following sections the original development of Free-Form Deformation is presented in detail. Also examined are the extensions to FFD: different polynomial bases, complex pre-deformation lattices (a broadened range of preliminary control point positions and connections), adaptively refining deformed polygon meshes for rendering, and improved user interfaces (intuitive means of controlling the displacement of control points).

Free-Form Deformation

Free-Form Deformation (FFD) in its seminal form [Bézier 1978; Sederberg and Parry 1986] relies on a Bernstein polynomial basis and a parallelepiped pre-deformation lattice.

A Bernstein basis allows the following substitution in eqn. 2.2:

$$\mathcal{N}_r^s(t) = \mathcal{B}_r^s(t) = \binom{s}{r}(1-t)^{(s-r)}t^r \qquad a = 1, b = 1, c = 1$$

The FFD lattice has $(l+1)\cdot(m+1)\cdot(n+1)$ control points and defines a trivariate Bézier hyperpatch. The properties of Bézier curves [Farin 1997, pp. 36-39], such as affine invariance, convex hull containment and endpoint interpolation, carry over to Bézier hyperpatches. Designers are thus able to exploit previous modelling experience.

The lattice is originally arranged in a parallelepiped tied to a local origin (O) with control points evenly distributed along the axes of a local co-ordinate frame ($\vec{U}, \vec{V}, \vec{W}$). Under these conditions, the linear precision property of Bézier curves [Farin 1997, p. 48] applies: If the control points are uniformly spaced along a line segment from P to Q then the Bézier curve simply generates

the linearly parametrised straight line between P and Q. The initial hyperpatch is thus a parallelepiped volume and the pre-deformation function is reduced to an affine transformation between co-ordinate systems.

$$\mathcal{H}(u,v,w) = O + u\vec{U} + v\vec{V} + w\vec{W} \qquad u,v,w \in [0,1] \tag{2.3}$$

The hyperpatch parameters ($\mathcal{H}^{-1}(X) = (u,v,w)$) can be evaluated using linear algebra:

$$u = \frac{\vec{V}\times\vec{W}\cdot(X-O)}{\vec{V}\times\vec{W}\cdot\vec{U}},\ v = \frac{\vec{U}\times\vec{W}\cdot(X-O)}{\vec{U}\times\vec{W}\cdot\vec{V}},\ w = \frac{\vec{U}\times\vec{V}\cdot(X-O)}{\vec{U}\times\vec{V}\cdot\vec{W}} \tag{2.4}$$

From these details the cost of the embedding and deformation functions for Bézier FFD can be derived (refer to figure 2.1).

Sederberg and Parry [1986] also present the conditions necessary to maintain continuity under different circumstances. Continuity of a certain order (C^k) can be maintained across the hyperpatch boundary between the deformed and undeformed parts of an object if k planes of boundary control points are undistorted. Secondly, continuity between adjoining lattices can be ensured if k planes of control points are forced to colinearity.

Finally, Sederberg and Parry [1986] define a family of volume preserving Free-Form Deformations which constrain the Jacobian of $\mathcal{F}(X)$ to unity ($\mathbf{det}(\mathbf{J}) = 1$).

FFD suffers from shortcomings in versatility and ease-of-use. Controlling deformations by moving lattice control points, while producing sculpted results, is both cumbersome and counter-intuitive. This is the motivation behind a number of user-interface improvements which culminate in constraint-based spatial deformation.

The boundary shape of FFD is limited to the image of a parallelepiped, continuity control is awkward, and the deformation within the boundary is not fully general. These disadvantages provided impetus for extensions in lattice topology and basis functions.

If the solid being modelled is approximated by a polygon mesh the adequacy of this approximation may fail under the contortions of FFD. It is then necessary to refine the mesh structure subject to the object's curvature after FFD. This work falls under the heading of rendering.

Different Polynomial Bases

A Bernstein basis for FFD implies that the image of the control lattice is a Bézier hyperpatch. Such a formulation is hampered by the same disadvantages as Bézier curves. Altering a single control point affects the entire hyperpatch and providing finer control by increasing the number of control points along an axis raises the degree of the hyperpatch and hence its computation cost. These problems are solved by substituting other basis functions.

Method	Embedding	Deformation	Total
Bernstein	130.31	672.74	803.05
Rational Bernstein	130.31	888.24	1018.55
Uniform B-Spline	138.15	679.22	817.37
NURBS	181.51	1417.6	1599.11

Table 2.1: **Computation Cost for Embedding and Deformation of a Single Sample Point under Different FFD Bases.**

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.

The hyperpatch equation (eqn. 2.2) becomes:

$$\mathcal{H}(u,v,w) = \frac{\sum_{i=0}^{\ell}\sum_{j=0}^{m}\sum_{k=0}^{n} \mathcal{B}_i^{\ell}(u)\cdot\mathcal{B}_j^{m}(v)\cdot\mathcal{B}_k^{n}(w)\cdot w_{i,j,k}P_{i,j,k}}{\sum_{i=0}^{\ell}\sum_{j=0}^{m}\sum_{k=0}^{n} \mathcal{B}_i^{\ell}(u)\cdot\mathcal{B}_j^{m}(v)\cdot\mathcal{B}_k^{n}(w)\cdot w_{i,j,k}} \tag{2.5}$$

$w_{i,j,k}$ is a scalar weight associated with the control point $P_{i,j,k}$. In theoretic terms, eqn. 2.5 is a projection from $\Re^4$, where control points are encoded as homogeneous co-ordinates $[w_{i,j,k}\cdot x_{i,j,k},\ \ w_{i,j,k}\cdot y_{i,j,k},\ \ w_{i,j,k}\cdot z_{i,j,k},\ \ w_{i,j,k}]$, onto $\Re^3$, characterised by $\big[x,y,z,w\big]\mapsto\big[\frac{x}{w},\frac{y}{w},\frac{z}{w}\big]$. Equation 2.5 reduces to a conventional Bézier hyperpatch if all weights are set uniformly to unity. 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 arguable since a user might find the increased degrees of freedom ($4\cdot(\ell+1)\cdot(m+1)\cdot(n+1)$) bewildering. There is also a small computation overhead associated with rational Bernstein polynomials, as tabulated in figure 2.1.

The introduction of a rational basis does not circumvent the problems of global support. For this reason 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 defined over a parallelepiped block of the trivariate domain). Cells are controlled by a limited subset of control points, e.g. $4\times4\times4=64$ control points in the case of a B-spline which is cubic in each $\vec{U}$, $\vec{V}$, $\vec{W}$ direction. B-spline curves have an associated knot vector whose non-decreasing elements mark the joins between segments and control the shape and domain of the basis functions. Uniform B-splines are so named because the knots are equally spaced.

Apart from local control over deformation there are two other advantages of this form of FFD:

1. The degree of the basis is independent of the number of control points. If a given lattice

has $(a+\ell)\cdot(b+m)\cdot(c+n)$ control points, derived from the indices of summation in eqn. 2.2, then the number of cells is $a \cdot b \cdot c$. A unit increment in the upper summation bound for an axis, say $a+\ell-1 \mapsto a+\ell$ along $\vec{U}$, introduces an extra slice of $b \cdot c$ cells, but does not change the degree or computation cost.

2. Continuity conditions on the boundary do not adversely impact the flexibility of deformation. The control lattice can be pictured as a kind of cube-shaped onion with layers of control points extending outwards from the core. If the outermost k layers, where $k \leq min(\ell, m, n)$, remain static and effectively beyond user control, then C^{k-1} continuity is assured [Bartels, Beatty and Barsky 1983, pp. 27-29]. These shells of "phantom" control points can be added seamlessly and invisibly to any user-defined lattice.

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 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 bases. For instance a tricubic B-spline based FFD requires the computation shown in figure 2.1.

Lamousin and Waggenspack [1994] combine the two strands of rational and piecewise bases. In their work FFD is built on non-uniform rational B-splines (NURBS), which have the local control properties of uniform 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. Lamousin and Waggenspack [1994] claim that "*the nonuniform NFFD lattice precludes direct use of previously proposed [linear embedding] methods.*" As a consequence they employ a costly numerical search, separately in $\vec{U}$, $\vec{V}$, and $\vec{W}$, to find the local cell co-ordinates of a point. Lamousin and Waggenspack favour a parallelepiped 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 inbetween). 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.

The development so far has highlighted the sensitive balance between versatility and efficiency. NURBS-based FFD, for instance, should only be used if either its features of weighting and non-uniform cells are vital to the application context, or real-time response is not critical.

Generalised Lattice Topologies

Free-Form Deformation in its original form is limited to pre-deformation lattices which induce a linearly parametrised hyperpatch. As a consequence, the embedding function $\mathcal{E}(X) = U$ is elegantly simplified at the expense of restricting the shape of both the deformation boundary and its interior density. There are two approaches to generalizing the topology of lattices. These

tradeoff versatility against efficiency.

Extended Free-Form Deformation (EFFD) [Coquillart 1990] allows a user to generate complex pre-deformation lattices through editing operations. The control points of a parallelepiped lattice with a Bernstein basis can be moved or merged. For instance, a cylindrical lattice is created from a paralellepiped 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 EFFD lattices can be formed by joining control points from the boundary of separate lattices. There are restrictions on this process:

- An EFFD lattice may not intersect itself as this complicates sample point parametrization. Unique co-ordinates in hyperpatch parameter space cannot be established if cells, or their interior, overlap.
- 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.

The complexity of EFFD hyperpatches forfeits the linear precision property of Bézier curves and consequently a numerical search procedure is necessary for object point embedding ($\mathcal{H}^{-1}(X)$). A sample point is located within a particular piecewise cell by using the convex hull property of Bézier curves [Farin 1997, pp. 38-39]. The exact (u, v, w) co-ordinates are then evaluated by the Newton-Raphson method for non-linear systems of equations [Press *et al.* 1992, pp. 379-383] with special consideration given to degenerate cells and a lack of convergence. As is to be expected, this is considerably more costly than previous embedding functions. 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 re-edit lattices, the process remains time-consuming and exacting.

MacCracken and Joy [1996] employ a subdivision strategy to cope with generalised lattices. The user specifies a lattice of arbitrary topology (bar self-intersection) and trivariate subdivision recursively refines this lattice towards the hyperpatch. Historically, bivariate subdivision schemes generate successively finer control nets which 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. By contrast, basis evaluation methods such as NURBS enforce a restrictive topology in forming and joining surface patches. Motivated by these topology benefits, MacCracken and Joy [1996] extend bivariate Catmull-Clark subdivision to trivariate volumes. 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 user constructs a lattice by borrowing from a template library, extruding a two-dimensional mesh, or placing and linking individual control points. These design tools ensure a valid non-intersecting lattice composed of closed cells bounded by polygonal faces. Next the user "freezes" the lattice ($\mathcal{L}_1$) by specifying its position and orientation relative to the embedded object.

2. Volumetric Catmull-Clark subdivision is 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 non-hexahedral "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 after 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 co-ordinates with a trilinear approximation.

3. As with all Free-Form Deformation techniques, the user then moves the control points of the "frozen" lattice into a deformed configuration ($\widetilde{\mathcal{L}}_1$).

4. 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 the pre- and post-deformation lattices, $\mathcal{L}_r$ and $\widetilde{\mathcal{L}}_r$, are structurally equivalent. The parametrisation "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 EFFD in two respects:

- The user is no longer forced to evolve lattices from a hexahedral-celled arrangement. Problem specifics, such as the desired deformation boundary and internal shape, can be used to directly determine the lattice.
- The continuity problems associated with merged control points are avoided.

Unfortunately, a subdivision approach is also more costly than EFFD. Steps (2) and (4) above rely on a recursive procedure whose computation cost is difficult to quantify but, as a rough guide, at each level of refinement a vertex V, belonging to a cell C, and of valence n (where valence is the number of edges in C incident on V) spawns a new cell with $2n$ 4-sided faces. 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.

In summary, these schemes address the issue of versatility to the detriment of efficiency. Further, they do not improve the ease-of-use of FFD.

Rendering Polygon-Mesh Objects

A polygon-mesh representation can only be considered an approximation of a curved, smoothly-varying object. The dichotomy is that while FFD produces smoothly moulded deformations it is applied most frequently to faceted polygon-mesh objects, especially when interactivity is a consideration. The primary difficulty is that an initially adequate approximation often fails under FFD, as areas of high detail and variation become sparsely covered, while areas of low detail and near planarity are oversaturated with polygons.

Three solutions are presented in the literature.

1. Parry [1986] develops tessellation of triangulated primitives in a Constructive Solid Geometry (CSG) environment. One edge in every triangle is heuristically marked as a "long side" (confusingly this may not be the longest edge in the triangle). Adjacent triangles can be halved at the midpoint of their shared edge provided this edge is classified as a "long side" by both triangles. A recursive subdivision of the surrounding mesh may be required to meet this condition. Parry's technique is triangle-centric. Candidate triangles are drawn off a stack and subdivision is triggered by a combination of (a) the triangle's projected screen area and (b) the disparity between the triangle's normal and that of its neighbours.
2. Greissmair and Purgathofer [1990] also act on a triangle-element polygon mesh and subdivide triangles midway along the common edge but, in contrast to Parry's refinement, this scheme is edge-based. All edges are tested according to the divergence of the midpoint after FFD and those that fail are placed on a refinement heap, ordered by decreasing edge length. Later, this heap is emptied by repeatedly extracting and subdividing edges.
3. Nimscheck [1995] adapts an advancing front finite-element mesh generation algorithm to the rendering problem. The input is a completely general polygon mesh with arbitrary sided convex and concave polygons and any number of holes piercing the object. The output is a triangulated and toleranced approximation of the object under FFD.

One measure of refinement quality is the aspect ratio equality of inserted triangles. In this way slivers (thin wedge-shaped triangles) are avoided. All three solutions cope with this issue by giving careful consideration to the length of edges. Another advantage is their broad applicability to other forms of deformation.

The first two approaches are efficient but lack symmetry. Refinement of detailed areas is provided but not decimation of flat regions. This implies that under a sequence of deformations the object's complexity will grow (or at least remain constant) even if its overall curvature decreases. Further, these techniques can only be applied to triangle-mesh objects and an initial tessellation of more complex polygon-mesh objects is necessary.

These deficiencies are avoided by Nimscheck, at the expense of efficiency, and consequently his technique is unsuitable for real-time applications [Nimscheck, p. 119]. There is certainly room

for a method which synthesizes the best aspects of the current refinement solutions and this is the purpose behind chapter 5.

User-Interface Improvements

As previously discussed, modifying an FFD lattice is a remarkably awkward interface to the control of deformation. It exposes the underlying mathematics of FFD to the user, and both inhibits exact specification of a particular deformation, and confuses the user with a proliferation of freedoms of control (number of control points).

A partial remedy to this situation is presented by Kuriyama and Tachibana [1995]. This demonstrates how the *dragger*, *engine* and *filter* components of the *Open Inventor* 3D graphics libraries [Wernecke 1994] can be instantiated and then linked to form tools. The user is able tie the motion of several control points to a single parameter and replace a lattice manipulation with a representative geometric object (e.g. a vice) intended to make the manipulation function clear. This solution is not wholly satisfactory as no toolsets are presented merely a complicated means of designing limited tools.

Davis and Burton [1991] develop high-level tools on a foundation of rational Bézier FFD. These tools act on aggregates of control points and provide taper, twist, bend and stretch operations along arbitrary axes and with local control. Objects can also be moulded to follow a curved path by aligning planes of a lattice with the control points of a Bézier curve.

Finally, the awkwardness of FFD interaction is the primary inspiration behind the point- and curve-based styles of spatial deformation.

Applications

Since its inception FFD has found widespread use in computer animation and solid modelling.

Piecewise cubic Bernstein FFD has been employed by Chadwick, Haumann and Parent [1989] in the “Critter” system for the layered construction of deformable animated characters, to simulate the bulges and ripples of muscles, creasing at joints and simple facial deformation.

Animated Free-Form Deformation [Coquillart and Jancéne 1991] illustrates the use of EFFD in key-framed computer animation. This animation is defined by moving in discrete time-steps either the lattice itself or the object within the lattice. The paper also outlines how metamorphic techniques and EFFD may be combined by applying a metamorphosis to a lattice.

Kalra *et al.* [1992] utilise rational Bernstein FFD to model a set of facial muscle actions. These are simple atomic facial motions such as “lip corner pull” and “eyebrow raise” that can be combined to simulate complex changes in facial expression. The work goes on to describe

an expression editor based on this principle.

Faloutsos, Van de Panne and Terzopoulos [1997] create a framework for Dynamic Free-Form Deformations which couple Lagrangian dynamics with a hierarchy of global and local deformations. The user first specifies a set of deformation modes (e.g. shear, bend) that tie a control point displacement matrix to a single amplitude parameter. Secondly, the user assigns $4 - 10$ discrete mass points to the embedded object. Finally, the authors solve the Lagrangian dynamics, which relate the kinetic energy of the mass point distribution, the potential energy of the amplitude parameters relative to their rest state, and external forces such as ground contact and gravity, in order to compute the state of the system at a step forward in time. The user can drive the simulation by keyframing the external and internal forces. The system can imbue inanimate objects such as tables and teapots with stylised cartoon-like movement.

In the free-form solid modelling arena, Güdükbay and Özgüç [1990] describe the sequential combination of FFD and regular tapering, twisting and bending deformations (section 2.2.3). In their system objects are represented as superquadrics (generalised ellipsoids and hyperbloids with arbitrary positive exponents) or Bézier surfaces and the user interface is a rudimentary combination of menu and keyboard input. This work places emphasis on synthesis rather than novelty.

As already mentioned Davis and Burton [1991] explore tool-based FFD in an interactive modelling environment. They believe that: "*FFD truly provides a lump-of-clay model manipulation capability in an interactive environment. Such a capability is valuable in creating and manipulating 3D models.*"

2.2.2 Point-Based

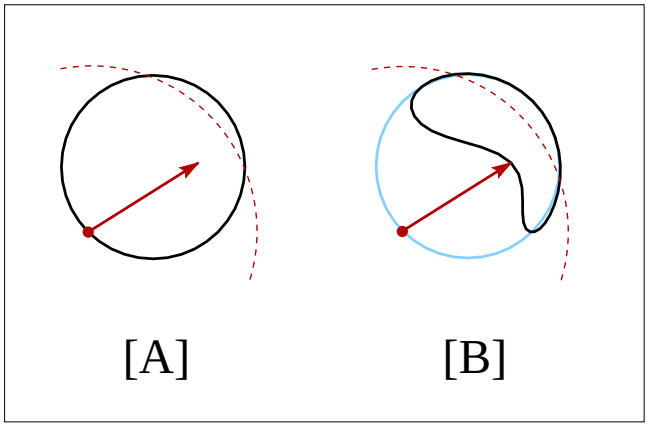


Figure 2.2: **Two-Dimensional Point-Based Deformation. [A] Pre-deformation: a point constraint (red vector) with a region of influence (dashed arc) relative to a deformable object (black). [B] Post-deformation: the point constraint has been imposed on the object.**

Controlling deformations by moving lattice control points, while producing sculpted results, tends to be cumbersome and counter-intuitive [Hsu, Hughes and Kaufman 1992]. Specifying even simple deformations requires a good working knowledge of spline theory. While a deformation "follows" lattice displacement, exact control over a given object point is difficult. Also, the proliferation of control points can easily confuse the user with an inordinate number of control freedoms. Further, 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.

Point-based spatial deformation, first presented by Borrel and Bechmann [1991] and Hsu, Hughes and Kaufman [1992], replaces the awkward lattice interface with direct manipulation. 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.

Point-based deformations are specified by a set of r constraints, each consisting of a constraint point (C_i), a vector displacement (ΔC_i), and a region of influence. As described by Bechmann [1994], these constraints are bound into a deformation function:

$$\mathcal{F}(X) = X + \mathbf{M} \cdot \mathcal{G}(\mathcal{E}(X)) = \widetilde{X} \tag{2.6}$$

Here $\mathcal{E}$ is a parametrising function ($\mathcal{E} : \Re^n \mapsto \Re^{r \times p}$) that maps a point to the local frame of each constraint, $\mathcal{G}$ is an extrusion function ($\mathcal{G} : \Re^{r \times p} \mapsto \Re^m$) which specifies how the deformation density tails off around each constraint and $\mathbf{M}$ is a matrix, of size $n \times m$, encapsulating a linear transformation ($\mathcal{T} : \Re^m \mapsto \Re^n$) combining the interaction of all constraints. Equation 2.6 can be recast in terms of the global spatial deformation formalism (section 2.1) by substituting $\mathcal{D} = \mathbf{M} \cdot \mathcal{G}$.

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

1. The user contributes a constraint point (C_i), vector displacement (ΔC_i), and region of influence for each of r constraints.

2. In eqn. 2.6, $\mathbf{M}$ is a matrix of unknowns, whose elements are instantiated so as to satisfy the user-defined constraints. The deformation of all constraint points by their corresponding displacement vectors is encoded in the following set of equations:

$$\Delta C_i = \mathbf{M} \cdot [\mathcal{G}_i(\mathcal{E}_i(C_i))] \qquad \forall\, i = 1, \ldots, r \tag{2.7}$$

   where $\mathcal{G}_i$, $\mathcal{E}_i$ are parametrising and extrusion functions particular to a constraint. In full matrix notation this can be expressed as:

$$\Delta \mathbf{C} = \mathbf{M}\mathbf{X} \Rightarrow (\Delta \mathbf{C})^T = \mathbf{X}^T \mathbf{M}^T \tag{2.8}$$

   This is a system of linear equations, which can be solved to find the matrix of unknowns ($\mathbf{M}^T$), either by standard numerical methods if $\mathbf{X}$ is square and of full rank, or via the pseudo-inverse if it is not.

3. Once the matrix $\mathbf{M}$ has been resolved, all sample points are successively embedded and deformed using eqn. 2.6.

Three categories of point-based spatial deformation are examined in the next sections: Directly Manipulated Free-Form Deformation (DMFFD), which effectively melds the hyperpatch- and point-based approaches; simple radial deformation, an elegantly minimal solution that owes much to the field of scattered data interpolation; and the Deformation of Geometric Models Editor (DOGME), a highly flexible system that encompasses volume preservation and topology alteration.

Directly Manipulated Free-Form Deformation

In conventional Free-Form Deformation the alteration in sample points is dictated in accordance with eqn. 2.2 by the position of lattice control points. By contrast Directly Manipulated Free-Form Deformation (DMFFD) determines the lattice configuration from a selection of sample points and their movement. To carry out this principle DMFFD is broken into three steps: the user provides a number of constraints (samples), each composed of a point, C_i, and its intended motion, ΔC_i; the lattice control points, $P_{i,j,k}$, 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 eqn. 2.2 and recasting the result in matrix notation:

$$\widetilde{\mathcal{H}}(\mathcal{E}(C_i)) = \sum_{i=0}^{a+\ell-1} \sum_{j=0}^{b+m-1} \sum_{k=0}^{c+n-1} \mathcal{N}_i^{\ell}([\mathcal{E}(C_i)]_u) \cdot \mathcal{N}_j^{m}([\mathcal{E}(C_i)]_v) \cdot \mathcal{N}_k^{n}([\mathcal{E}(C_i)]_w) \cdot \widetilde{P}_{i,j,k}$$

$$\Rightarrow \quad C_i + \Delta C_i = \sum_{i=0}^{a+\ell-1} \sum_{j=0}^{b+m-1} \sum_{k=0}^{c+n-1} \mathcal{N}_{i,j,k}^{\ell,m,n}(\mathcal{E}(C_i)) \cdot (P_{i,j,k} + \Delta P_{i,j,k}) \qquad \forall i = 1, \ldots, r$$

$$\Rightarrow \quad \mathbf{C} + \Delta\mathbf{C} = \mathbf{B}(\mathbf{P} + \Delta\mathbf{P})$$

$$\Rightarrow \quad \Delta\mathbf{C} = \mathbf{B}\Delta\mathbf{P} \tag{2.9}$$

Each row of eqn. 2.9 is an unrolling of eqn. 2.2 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. The weight entries in column j of $\mathbf{B}$ are matched to the control point alteration in row j of $\Delta\mathbf{P}$. Normally FFD evaluates the alteration in constraint points, $\Delta\mathbf{C}$, 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 points, $\mathbf{C}$, and their intended motion, $\Delta\mathbf{C}$, and the new control point positions, $\mathbf{P} + \Delta\mathbf{P}$, are found. In mathematical terms, given $\mathbf{B}$ and $\Delta\mathbf{C}$ a solution $\Delta\mathbf{P}$ is found for the system of linear equations $\mathbf{B}\,\Delta\mathbf{P} = \Delta\mathbf{C}$. This process is very well defined if $\mathbf{B}$ is square ($r = s$) and non-singular. Unfortunately, $\mathbf{B}$ is generally underdetermined ($r < s$) and sometimes (rarely) overdetermined ($r > s$). In these instances, a solution (or the closest approximation) must rely on a formulation known as the pseudo-inverse [Peters and Wilkinson 1970] represented as $\Delta\mathbf{P} = \mathbf{B}^{+}\Delta\mathbf{C}$. For interactive performance this is best solved by exploiting the sparse structure of the basis matrix.

The region of influence of individual constraint points is implicitly dependent on the initial lattice (P). A hyperpatch with compact cells will contain the constrained deformation within a small volume, while expansive cells will induce correspondingly wide-ranging deformations. A more detailed treatment of DMFFD appears in chapter 3.

Simple Radial Deformations

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 influence (r_i) centred 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] develop Simple Constrained Deformation (*Scodef*), the first instance of the radial deformation technique. Scodef has the following useful properties:

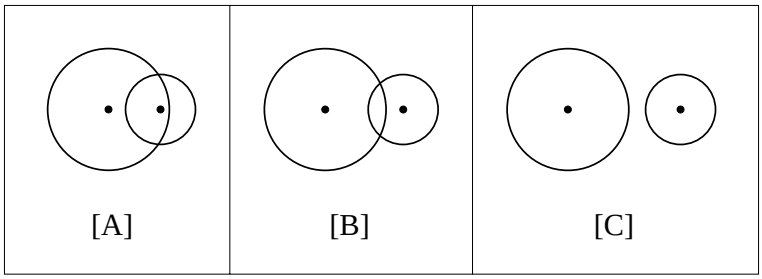


Figure 2.3: **Radial Constraint Relationships. Two point constraints and their circular regions of influence: (A) Non-disjoint, (B) Partially disjoint, (C) completely Disjoint.**

1. 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 centred at C_i.

2. 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.

3. 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.

In terms of the mathematical formalism of point-based spatial deformation, each Scodef constraint has a parametrising function, $\mathcal{E}_i : \Re^3 \mapsto \Re$, a scalar extrusion function, $\mathcal{G}_i : \Re \mapsto \Re$, and a column in the mapping matrix, $\mathbf{M}_i : \Re \mapsto \Re^3$, associated with it.

The parametrising function, $\mathcal{E}_i(Q) = \|Q - C_i\|/r_i = u$, has a scalar domain and returns $0 \leq u \leq 1$ if Q is within (and $u > 1$ if it is outside) the sphere of radius r_i around C_i.

The extrusion function, $\mathcal{G}_i(u)$, is a normalised B-spline basis function. This implies that the maximum scalar contribution of $\mathcal{G}_i(\mathcal{E}_i(Q))$ occurs when $Q = C_i$ and this contribution tapers to zero as Q moves a distance r_i away from C_i.

The column vector $\mathbf{M}_i$ stores a modified displacement vector which accounts for the effect of nearby constraints. If C_i is completely independent of other constraints then $\mathbf{M}_i = \Delta C_i$. In general, the $3 \times r$ matrix $\mathbf{M}$ is created so as to satisfy the r constraints:

$$[\Delta C_i] = \mathbf{M} \cdot [\mathcal{G}_i(\mathcal{E}_i(C_j))] \qquad \forall\ i = 1, \ldots, r;\ j = 1, \ldots, r$$

As per section 2.2.2, these equations are combined into a single matrix system $\Delta\mathbf{C} = \mathbf{MX}$. Here $\Delta\mathbf{C}$ is an $3 \times r$ matrix storing the displacement vectors ΔC_i in each column, $\mathbf{M}$ is a $3 \times r$ matrix of unknowns, and $\mathbf{X}$ is an $r \times r$ matrix with element (i, j) holding the ith extrusion function applied to the jth constraint ($\mathbf{X}_{(i,j)} = \mathcal{G}_i(\mathcal{E}_i(C_j))$). The matrix $\mathbf{X}$ quantifies the interrelationship between constraints. This can be recast as a more familiar system of fully determined linear equations as: $(\Delta\mathbf{C})^T = \mathbf{X}^T\mathbf{M}^T$. The solution to this system depends on the relationship between constraint points.

Constraint points can be classified, with respect to each other, by the degree to which their spheres of influence overlap. Two constraint points (refer to figure 2.3) are completely disjoint if their spheres of influence do not intersect ($\|C_i - C_j\| > r_i + r_j$), and non-disjoint if either constraint point lies within the sphere of influence of the other ($\|C_i - C_j\| \leq max(r_i, r_j)$). Otherwise, they are partially disjoint. If all constraints are either partially or completely disjoint then there is no need to explicitly solve eqn. 2.8 because $\mathbf{X}$ is then the identity matrix (since constraints influence only themselves) and therefore $\Delta\mathbf{C} = \mathbf{M}$. A numerical solution is only necessary where non-disjoint constraints exist. In practice, eqn. 2.8 is partitioned into non-disjoint and disjoint components which are then solved separately.

Borrel and Rappoport [1994] go on to identify a "space-tearing" phenomenon 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 into eqn. 2.8 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 neighbouring constraints. To accommodate k duplicated constraints eqn. 2.8 is modified as follows: k columns and k rows are added to $\mathbf{X}^T$ and $\mathbf{M}^T$ respectively.

$$\mathbf{X}^T = \begin{bmatrix} \mathcal{G}_1(\mathcal{E}_1(C_1)) & \dots & \mathcal{G}_r(\mathcal{E}_r(C_1)) & \mathcal{G}_{r+1}(\mathcal{E}_{r+1}(C_1)) & \dots & \mathcal{G}_{r+k}(\mathcal{E}_{r+k}(C_1)) \\ \vdots & & \vdots & \vdots & & \vdots \\ \mathcal{G}_1(\mathcal{E}_1(C_r)) & \dots & \mathcal{G}_r(\mathcal{E}_r(C_r)) & \mathcal{G}_{r+1}(\mathcal{E}_{r+1}(C_r)) & \dots & \mathcal{G}_{r+k}(\mathcal{E}_{r+k}(C_r)) \end{bmatrix}$$

$(\Delta\mathbf{C})^T$ remains unchanged. This is no longer a fully determined system of linear equations. It is now underdetermined ($\mathbf{X}^T$ has dimensions $r \times (r + k)$) and must be solved, in a least squares sense, by the pseudo-inverse.

Ruprecht, Nagel and Müller [1995] adapt advances in scattered data interpolation to radial deformation. They recommend a distance-measuring parametrisation $\mathcal{E}_i(Q) = \|Q - C_i\| = u$ and an extrusion founded on Hardy's multiquadrics:

$$\mathcal{G}_i(u) = (u^2 + r_i^2)^{\frac{\mu}{2}}, \qquad \mu \neq 0$$

Setting $\mu \leq -2$ supplies a constraint contribution which, even though unbounded (nowhere zero), is still fairly localised (tends rapidly to zero beyond r_i). Ruprecht, Nagel and Müller [1995] also propose that the radii of influence be set automatically as the distance to the nearest neighbouring constraint point, i.e. $r_i = min_{i \neq j}(\mathcal{E}_i(C_j))$. This provides a reasonable default which dampens "space-tearing" behaviour. In other respects this method is identical to Scodef.

The main advantage of radial deformations are their sheer simplicity and consequent efficiency. As will be shown in the evaluation it compares very favourably 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. This is a feature which radial deformation cannot emulate in this formulation, restricted as it is to spherical boundaries.

Deformation of Geometric Models Editor

DOGME (Deformation of Geometric Models Editor) [Borrel and Bechmann 1991; Bechmann 1994] is the most general form of point-based spatial deformation and subsumes the other methods presented in this survey. DOGME is characterised by the deformation function in its full generality: $\mathcal{F}(X) = X + \mathbf{M} \cdot \mathcal{G}(\mathcal{E}(X))$. This function can be instantiated in a variety of ways.

Firstly, different extrusion functions ($\mathcal{G}_i$) can be applied. For instance, Bechmann [1994] suggests that, in addition to the standard B-spline basis, extrusion functions could be designed to simulate specific materials such as clay, wood and stone, but does not go into details. More conventionally, Borrel and Bechmann [1991] employ power basis polynomials for global deformation.

Secondly, the composition vector ($\mathbf{X}_i$), whose elements combine different extruded parametrizations, can take one of three forms:

- *Simple Form.* A single radial function $f_i(g_i(Q)) : \Re^3 \mapsto \Re$ is associated with each constraint and it quantifies the influence of that constraint on the point Q. The vector $\mathbf{X}_i$ has r entries: $[\mathbf{X}_i]_j = \mathcal{G}_j(\mathcal{E}_j(Q)), j = 1, \ldots, r$. The simple form is identical to Scodef.

- *Simple Product Form.* Bechmann and Dubreuil [1995] make provision for convex polyhedral boundaries by imposing a system of local co-ordinate axes at each constraint. At its simplest, three orthogonal axes with range markings specify a parallelepiped region of influence around the constraint point at their origin, but the boundary may become as complex as a convex prismatic polyhedron with the introduction of additional non-orthogonal axes. The ith constraint now has a_i axes and an associated range and embedding function ($r_{i,k}$ and $\mathcal{G}_{i,k}(\mathcal{E}_{i,k}(X))$ over $i = 1, \ldots, r;\ k = 1, \ldots, a_i$). The elements of $\mathbf{X}_i$ are assembled using an axis-wise product:

$$[\mathbf{X}_i]_j = \prod_{k=1,\ldots,a_i} \mathcal{G}_{j,k}(\mathcal{E}_{j,k}(Q)) \qquad i = 1, \ldots, r$$

  Understandably, this form is more computationally costly than Scodef.

- *Tensor Product Form.* Borrel and Bechmann [1991] proposed this form of DOGME in order to mimic the behaviour, but not the control mechanism, of FFD. Deformation is particularised along each world co-ordinate axis according to a B-spline knot sequence. The parametrising function becomes redundant (unless an orientation other than the world co-ordinate system is required), the extrusions are B-spline basis functions, which depend

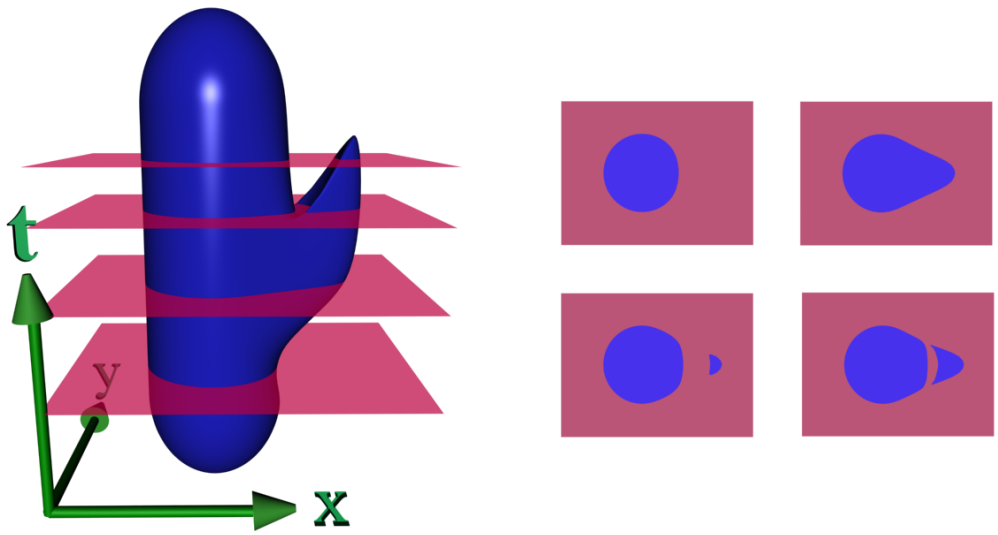


Figure 2.4: **Topology Changes Induced by Cross-Sectioning. [Left] A cylinder with an extrusion is cross-sectioned at four t-intervals. [Right] The cross-sections produce two-dimensional shapes that change topology with increasing t (clockwise from top-left).**

on a degree and knot sequence, and the vector $\mathbf{X}_i$ obeys a tensor product composition. The extrusion functions for each of the $\vec{X}$, $\vec{Y}$ and $\vec{Z}$ axes are $\mathcal{G}_i^{\vec{X}}(x) = \mathcal{N}_i^{\ell}(x)$, $\mathcal{G}_j^{\vec{Y}}(y) = \mathcal{N}_j^{m}(y)$, $\mathcal{G}_k^{\vec{Z}}(z) = \mathcal{N}_k^{n}(z)$ where $i = 0, \ldots, a+\ell-1$; $j = 0, \ldots, b+m-1$; $k = 0, \ldots, c+n-1$. The composition vector, $\mathbf{X}_i$ is simply a flattening of a product of extrusions, $[\mathbf{X}_i] = \mathit{flatten}(\mathcal{G}_i^{\vec{X}}(x) \cdot \mathcal{G}_j^{\vec{Y}}(y) \cdot \mathcal{G}_k^{\vec{Z}}(z))$ and has $(a+\ell) \times (b+m) \times (c+n)$ entries. The composition matrix $\mathbf{X}^T$ has dimension $r \times (a+\ell) \cdot (b+m) \cdot (c+n)$ and is, in general, no longer fully determined. The matrix of unknowns $\mathbf{M}^T$ must be found with the pseudo-inverse. If $\mathbf{M}^T$ is visualised as containing the changes in a set of virtual control points then the equivalence of tensor product DOGME and DMFFD becomes evident.

Thirdly, DOGME is a generalised mapping from and to n-dimensional space ($\mathcal{F} : \Re^n \mapsto \Re^n$). Common choices are $\Re^2$ for image warping and $\Re^3$ for object deformation but Bechmann and Dubreuil [1993] advocate $\Re^4$ for computer animation. This application is particularly appropriate since the time parameter is seamlessly incorporated into the DOGME formulation as the fourth dimension. A three-dimensional snapshot of an evolving object is taken by holding time constant.

There have been several significant extensions to DOGME since its inception.

Bechmann and Dubreuil [1995] augment DOGME with high-level animation tools which take the form of "orders" such as "move", "turn", "roll on a flat surface" and "bounce along a flat surface". The user inputs parameters, for example the starting and finishing locations and times, and these are automatically decomposed into a set of rotations and simple product constraints in $\Re^4$.

The topology of objects undergoing spatial deformation is innately invariant (as explained in section 2.1). However, Aubert and Bechmann [1997b] circumvent this property by exploiting the fact that, even though the topology of an n-dimensional object is unaffected by a spatial deformation in $\Re^n$, the same does not necessarily apply to its $(n-1)$ dimensional cross-section at a specific time interval. This is evidenced by figure 2.4 where a cylinder, formed by sweeping an ellipse through time, is deformed by a constraint whose displacement has a time component. The resulting cross-sections separate gradually into two parts. Aubert and Bechmann's [1997b] space-time algorithm operates by firstly extruding a 3D object linearly along the time axis, then administering a set of space-time point constraints, and finally taking a cross section at time t_0 by intersecting the deformed 4D object with a hyperplane of constant time $t = t_0$. Predictably, this is very expensive in terms of both memory and computation. The algorithm also focuses on the mechanics rather then the means of control. There is scope in this area for research into efficient and precise control over topology changes.

Volume maintenance is an aspect of real-world sculpting that is not supported by spatial deformation. Aubert and Bechmann [1997a] propose a remedy for the DOGME system which iteratively approaches perfect volume preservation while still matching the user's point constraints. To begin with, the volume variation of a triangle-mesh is measured by summing the tetrahedral displacement volumes for every facet. A single tetrahedron is constructed by connecting all the vertices in a pre-deformation triangle to a single selected post-deformation vertex. This volume variation is reduced by using the underdetermined nature of tensor product DOGME. A new solution matrix $\mathbf{M}^T$ can be selected from the infinity available by adding the matrix $(\mathbf{I} - \mathbf{X}^+\mathbf{X})\xi$, where ξ is any $m \times 3$ matrix. Aubert and Bechmann [1997a] search for a matrix ξ which minimised the norm ($\|\xi\| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{3}\xi_{ij}^2}$) and the volume variation. Typically this "minimum deviation" search converges to a volume change of less than 0.01% within 4 - 6 iterations.

2.2.3 Curve-Based

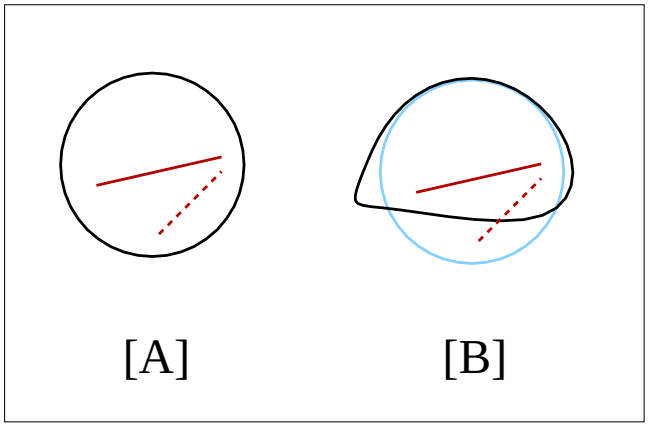


Figure 2.5: **Two-Dimensional Curve-Based Deformation. [A] Pre-deformation: a curve constraint (red) with source (dashed) and destination (solid) components inside a deformable object (black). [B] Post-deformation: the curve constraint has been imposed on the object.**

An even higher level of usability and control is afforded by curve-based spatial deformation. In lieu of lattice or point-constraint interaction, the user places and then manipulates a set of characteristic curves. The surrounding space and embedded object are warped to follow the curves' distortions. Curve-based deformations are interpolating in the sense that a sample point on an initial curve will deform onto the corresponding final curve.

There is some historical precedent to the usefulness of curves in traditional, as opposed to computer-assisted, sculpting and design. When sculpting in wax or clay an armature, consisting of a curved wire skeleton, is often used to provide support and shape definition to a model. Also paper-based architectural and automotive design is heavily reliant on profile curves and cross sections. The ubiquity of curves carries over to computer-based modelling so that designers generally have a thorough practical understanding of the properties and behaviour of curves. Such familiarity benefits the predictability and ease of use of curve-based deformation. Curves are also fundamentally more versatile than isolated points: they can control the interrelationship between point and derivatives along a thread of space.

A given curve-based deformation is defined by a set of r constraints, each having a source, $\mathcal{S}_i(t)$, and destination curve, $\widetilde{\mathcal{S}}_i(t)$. Often these constraints incorporate a means of scaling and twisting around the curve constraint and an enclosing volume which demarcates the deformation boundary. The specifics of these latter controls are dependent on the particular curve-based deformation method.

In general terms, curve-based deformation proceeds in four stages:

1. The user stipulates the embedding component of all r curve constraints, laying out the

source curves, $\mathcal{S}_i$, as spines within the object or feature curves across the surface, and optionally specifying an accompanying volume of influence. The early regular global [Barr 1984] and generalised de Casteljau [Chang and Rockwood 1994] curve schemes limit the user to a single source curve aligned with one of the co-ordinate axes and have no mechanism for localising deformations.

2. Sample points from the object are attached to the source curves by means of an embedding function $\mathcal{E}(X) = U$. Sometimes this is merely an affine co-ordinate transformation identical to FFD parametrisation, but in the case of Axial Deformation [Lazarus, Coquillart and Jancène 1994] and Wires [Singh and Fiume 1998] a local co-ordinate frame is established at the closest location on each source curve and the sample point in question is located within these frames.

3. The destination curves, $\widetilde{\mathcal{S}}_i$, and orthogonal controls (scale and twist) are provided by the user.

4. Collectively, the embedding and deformation components of the curve constraints define a deformation function $\mathcal{D}(U) = \widetilde{X}$, which is administered to all the embedded samples. If there is more than a single curve constraint, as is possible in Wires, then care must be taken to blend the deformation contributions of individual constraints.

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 stylised 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 flavour 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:

$$\mathcal{F}(X) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} X \qquad \theta = \mathcal{R}(z) \tag{2.10}$$

The other regular global deformations have a couple of attributes in common with this example.

1. They have no embedding function mapping co-ordinates to an intermediate tool-centred space, i.e. $\mathcal{E}(X) = X$, and the transformation is entirely captured by the deformation function $\mathcal{D}(X) = \widetilde{X}$.

2. They are, at heart, differential affine transformations. As is obvious from eqn. 2.10 twisting consists of differential rotation. Similarly, 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 co-ordinate axis deformations are extremely efficient but their utility is limited. The distortions are stylised rather than free-form and centred around a single straight axis (the source curve).

A Generalised de Casteljau Approach

Chang and Rockwood [1994] set forth an efficient free-form curve-based deformation method which generalises the de Casteljau algorithm for Bézier curve evaluation. The repeated linear interpolation used to move from the control polygon to a point on the curve is extended to full affine transformation. The user manipulates Bézier control points, P_i, each augmented by two vector "handles", S_i and T_i, that generate tilt and twist around the controlling Bézier skeleton.

The univariate de Casteljau algorithm [Farin 1997, p. 35] evaluates a Bézier curve of degree n, with control points P_i, by recursively engraving the parameter value u onto the control polygon

$$\mathcal{P}_i^r(u) = (1-u)P_i^{r-1} + uP_{i+1}^{r-1} \qquad \begin{cases} r = 1, \ldots, n \\ i = 0, \ldots, n-r \end{cases} \tag{2.11}$$

where $\mathcal{P}_i^0(u) = P_i$ and $\mathcal{P}_0^n(u)$ is the point on the curve at u.

A generalised trivariate de Casteljau algorithm is obtained by raising the rank of the control point matrix of eqn. 2.11 from linear interpolation to full affine transformation in homogenous form. Rather than a single scalar parameter u being recursively engraved onto a sequence of progressively shorter control polygons, three-dimensional sample point parameters, $U = (u, v, w)$, are engraved on a chain of control volumes. At the first level of recursion the volumes are parallelepipeds demarcated by the control polygon and "handles".

$$\mathcal{P}_i^r \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix} = \Phi[P_i^{r-1}, P_{i+1}^{r-1}, S_i^{r-1}, T_i^{r-1}] \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} \qquad \begin{cases} r = 1, \ldots, n \\ i = 0, \ldots, n-r \end{cases} \tag{2.12}$$

$$\Phi[P, Q, R, S] = \begin{bmatrix} (Q_x - P_x) & S_x & T_x & P_x \\ (Q_y - P_y) & S_y & T_y & P_y \\ (Q_z - P_z) & S_z & T_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

$$\mathcal{P}_i^0 = P_i, \qquad S_i^r = \begin{cases} S_i & \text{if } r = 0 \\ 0 & \text{otherwise} \end{cases}, \qquad T_i^r = \begin{cases} T_i & \text{if } r = 0 \\ 0 & \text{otherwise} \end{cases}$$

The deformation of a sample point with parameter space co-ordinates $U = (u, v, w)$ is $\mathcal{D}(U) = \mathcal{P}_0^n(U)$ from eqn. 2.12. Chang and Rockwood [1994] do not dwell on the embedding function $\mathcal{E}(X) = U$ but, presumably, a mapping similar to FFD is sufficient.

Generalised de Casteljau deformation allows primary control over the cube of parameter space $u, v, w \in [0, 1]$ by means of a Bézier source curve along $\vec{U}$ (degree n) and vector "handles" in $\vec{V}$ and $\vec{W}$ (linear). Bends and undulations in the $\vec{U}$ axis are effected with the control points, while tilting and twisting $\vec{V}$ and $\vec{W}$ is the province of the "handles". Similar end-results can be attained with a Bézier FFD hyperpatch which is linear in $\vec{V}$ and $\vec{W}$ and of degree n in $\vec{U}$, but without the same economy of specification and control. Another marked disparity is that generalised de Casteljau deformation has a global effect on parameter space, although some of the extrapolation effects when $u < 0$ or $u > 1$ may be undesirable. Generalised de Casteljau deformations allow greater freedom than global co-ordinate axis deformations but they are still restricted to a single control axis (source curve) with global influence.

Axial Deformations

The Axial Deformation (AxDf) modelling tool, expounded by Lazarus, Coquillart and Jancène [1994], forces an object to follow the motion of a single 3D curve. AxDf improves on generalised 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 generalised cylinder centred on the curve, can be used to localise the deformation.

Under AxDf, a local orthogonal co-ordinate frame is associated with key points on the controlling curve. The Frenet frame, consisting of the tangent, normal and binormal, might seem a natural choice but it is undesirable because the normal may be undefined (where curvature vanishes), flip direction (at inflection points) or rotate wildly around the curve. Instead, AxDf employs a rotation minimising orthogonal frame which can be geometrically constructed [Klok 1986] 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 non-existent ($d > r_{max}$).

Only one curve point influences a given object point and this is determined by the embedding function. A vertex V_i is attached to the closest point ($\mathcal{S}(t_i)$) on the curve by storing the co-ordinates of V_i in the local frame of $\mathcal{S}(t_i)$. The curve is warped and V_i is imparted a deformation by projecting from the parametric co-ordinates in the new local frame at $\widetilde{\mathcal{S}}(t_i)$, as mitigated by the zone of influence.

To elaborate, the AxDf algorithm has 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 generalised cylinders which taper the deformation.

2. The source curve is recursively subdivided into a line segment approximation. The Bishop and Klok co-ordinate frame and the zone of influence interpolants are then constructed for each line segment. All object vertices (V_i) are parametrised with respect to the approximated curve by establishing the closest point on the curve $\mathcal{S}(t_i)$, the co-ordinates (u_i, v_i, w_i) of V_i within the frame at $\mathcal{S}(t_i)$ and a zone of influence scaling factor r_i^* calculated as:

$$r_i^* = \mathcal{R}_i(\|V_i - \mathcal{S}(t_i)\|) = \mathcal{R}_i(d) = \begin{cases} 1 & if \quad d < r_{min,i} \\ \mathcal{I}(\frac{d - r_{min,i}}{r_{max,i} - r_{min,i}}) & if \quad r_{min,i} < d < r_{max,i} \\ 0 & if \quad d > r_{max,i} \end{cases}$$

where $\mathcal{I}(d) = r, d \in [0,1], r \in [1,0]$ is an interpolating function.

3. The embedding 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 co-ordinate frames are calculated for the deformed curve $\widetilde{\mathcal{S}}$. A vertex V_i is deformed by projecting from its co-ordinates $U_i = (u_i, v_i, w_i)$, in the new frame at $\widetilde{\mathcal{S}}(t_i)$, back to world co-ordinate space as $\widetilde{V}_i$. One subtlety is that this deformation must be moderated by the zone of influence. Thus the final deformation is:

$$\mathcal{F}(V_i) \quad = \quad V_i + r_i^* \cdot (\widetilde{V}_i - V_i)$$

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 parametrisation. Apart from a considerable computation burden, it has some continuity implications since adjacent sample point do not necessarily attach to adjacent curve points.

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. Singh and Fiume [1998] liken Wires to a sculptor's armature, whereby a wire skeleton guides the surrounding clay.

As with other schemes, Wires deformation takes place in two steps:

1. The object's vertices are parametrised relative to a bundle of n wires.

2. The wires are manipulated and the object's vertices warp accordingly.

Each wire i has a host of defining parameters. $\mathcal{S}_i$ 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_P) that minimises the distance between the sample point (P) and the source curve ($\mathcal{S}_i(t)$). A density function $\mathcal{Z}(P, \mathcal{S}) = \Gamma(\|P - \mathcal{S}_i(t_P)\|/r)$, familiar from implicit function and scattered data interpolation literature, ascribes a weighting to the deformation of P 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:

1. 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 $\mathcal{S}_i$.

2. A domain curve $\mathcal{M}_i$ determines the radius of influence for points according to $r = \|\mathcal{S}_i(t_P) - \mathcal{M}_i(t_M)\|$, where t_M is the parameter value of the point on $\mathcal{M}_i$ closest to P. 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 co-ordinate frames of AxDf. The deformation contribution of a wire $\mathcal{D}_i(P)$ is a function of the difference between $\mathcal{S}_i(t_P)$ and $\widetilde{\mathcal{S}}_i(t_P)$ as modulated by the density weighting function $\mathcal{Z}(P, \mathcal{S}_i)$ and is separated into scaling, rotation and translation components. The radial scaling value (s) contributes uniform scaling about $\mathcal{S}_i(t_P)$. Then the angle θ between the tangent vectors $\mathcal{S}_i'(t_P)$ and $\widetilde{\mathcal{S}}_i'(t_P)$ imparts a corkscrew rotation by $\theta \cdot \mathcal{Z}(P, \mathcal{S}_i)$. Finally, the translation $(\widetilde{\mathcal{S}}_i(t_P) - \mathcal{S}_i(t_P)) \cdot \mathcal{Z}(P, \mathcal{S}_i)$ 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:

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

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

2.3 Evaluation

2.3.1 Ease of Use

An intuitive and uncomplicated means of user interaction is crucial to effective sculpting 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 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. In contrast, constrained spatial deformations implement direct manipulation of either points or curves. 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, and curve constraints have been likened by Singh and Fiume [1998] to a sculptor's armature. 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. There are two paths to improving matters. The first route involves reformulating the interface to spatial deformation so as to reduce the number of controls. For instance, FFD on a uniform cubic B-spline basis with only 8 cells has 375 degrees of freedom (3 co-ordinates $\times\ 5 \cdot 5 \cdot 5$ control points), whereas DMFFD, with an identical lattice foundation, offers only 6 degrees of freedom per constraint (3 co-ordinates $\times$ (1 constraint point + 1 constraint displacement)). Broadly speaking, spatial deformations can be ranked by decreasing degrees of control freedom, as follows: hyperpatch, then curve and finally point-based deformation. The second route, as recommended by Schneiderman [1983], is to enable 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. For example, having the control weights of NURBS FFD set to unity saves the user from wastefully adjusting weights. In terms of a control hierarchy, Wires has two modes for creating a zone of influence. Locators are a simple, uncluttered way of designing a generalised influence cylinder. At the next level of sophistication, domain curves supply convoluted boundaries.

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 behaviour of actual modelling clay. The disparity between real and virtual sculpting is marked:

- **Volume Conservation**. Despite moulding and manipulation, inelastic substances, such as modelling clay and putty, generally retain the same volume. For instance, a localised indentation will induce outward bulging more globally. Aubert and Bechmann [1997a] devise volume preserving deformation tools. Despite being more realistic, such tools may prevent local control: a small deformation can have global implications. Also volume preservation is deleterious from an efficiency standpoint.

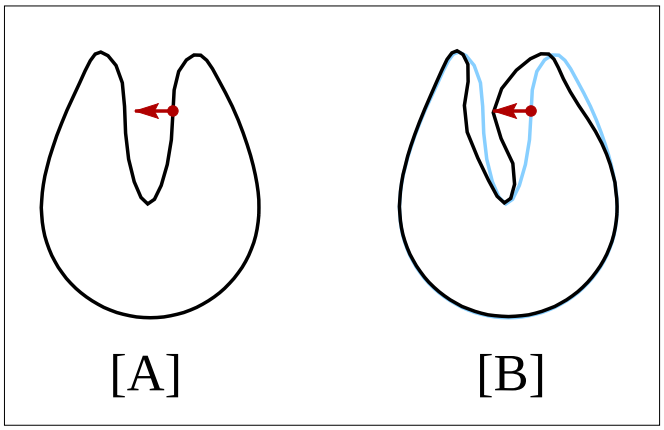


Figure 2.6: **Transmission Effects. The effect of a point manipulation is transmitted across the external space between two extrusions. [A] Pre-deformation. [B] Post-deformation.**

- **Transmission Effects**. Spatial deformation, in contrast to real clay, makes no distinction between the interior and exterior of an object. The kinds of transmission across the empty space between separated surfaces shown in figure 2.6 may baffle user’s expectations and are worth correcting.
- **Gravity**. Real clay, unlike its computer-generated counterpart, is susceptible to gravity. Thin extrusions will tend to sag and the mutable underside often conforms to a hard supporting surface. Gravity is not worth simulating since its effects are detrimental rather than desirable.
- **Self-Intersection**. Another jarring behaviour of spatial deformation is the interpenetration of an object’s surface. Apart from being physically unrealistic, this has validity implications, which are explored in section 2.3.4.
- **Topology Alteration**. Artists often employ the friability of clay to pierce holes, tear away unwanted pieces and smoothly join separately sculpted components. Aubert and

Method	Page No.	Local Deformation	Precise Displacement	Constraint Independence	Boundary Shape
FFD	13	√	×	-	parallelepiped
EFFD	16	√	×	-	free-form
Arbitrary Lattice Topology	17	√	×	-	free-form
DMFFD	23	√	√	×	parallelepiped
Radial	24	√	√	√	sphere
DOGME (Simple Product)	27	√	√	√	symmetric polyhedron
Regular Global	31	×	×	×	none
Generalised de Casteljau	32	×	×	×	none
Axial	33	√	√	×	generalised cylinder
Wires	34	√	√	√	generalised cylinder

Table 2.2: **A Comparison of the Versatility of Space Deformation Methods. The dashed (-) entries indicate that the corresponding method is not constraint-based.**

Bechmann [1997b] support these genus transformations but their system is tailored to animation with no real-time response requirement. In order to effectively implement "cut", "copy" and "paste", simple and efficient topology alteration is mandated.

- **High-Level Tools**. Sculptors manipulate clay primarily with their hands but adjuncts, such as a potter's wheel, are also employed. By comparison, only relatively unsophisticated point and curve tools are available in spatial deformation. Developing more advanced tools would reduce some of the dissonance between physical and computer-assisted modelling and also benefit productivity and ease-of-use.

2.3.2 Versatility

The developments reviewed in this chapter 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 threefold:

1. **Locality**. Certain schemes, namely regular global and generalised de Casteljau deformation, affect the whole span of world co-ordinate space and are only suited to warping entire objects. Fortunately, the majority allow local 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.

2. **Constraints**. 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, with varying success, by the constrained point or curve-based deformation techniques. With the exception of the regular global and generalised de Casteljau deformations, which are limited by enforced axis alignment, these constraint tools enable precise positioning of any selected element. They can be further ranked according to whether multiple independent constraints are supported. Radial deformation, DMFFD, simple product DOGME and Wires all allow any number of constraints and, with the exception of DMFFD, where the region of influence around directly manipulated points is determined by the hyperpatch substrate, there is no interdependence of constraint parameters.

3. **Boundary Control**. Many techniques focus on extending the range of permissible boundary configurations. Apart from regular global and generalised de Casteljau deformation which have no boundary at all, the most limiting shapes belong to FFD and DMFFD (a parallelepiped whose aspect ratio is governed by the number and spread of control points along orthogonal axes) and radial deformation (a union of spheres of varying radii centred on constraints). Greater flexibility is furnished by simple product DOGME (a union of symmetric polyhedra each marked out by a collection of paired half-planes enclosing a constraint). Axial deformation and Wires separate between deformed and undeformed zones with one (or in the case of Wires, several) generalised cylinders. However the pinnacle of a truly free-form boundary is only attained by EFFD and deformation with lattices of arbitrary topology.

The relationship between the different spatial deformation methods and these versatility enhancements are catalogued in table 2.2. Versatility is not without its price. The next section will demonstrate that the most powerful schemes (arbitrary lattice topology deformations, DOGME and Wires) are also among the least efficient.

2.3.3 Efficiency

The computation cost of a particular spatial deformation technique is a crucial factor in determining its suitability for interactive sculpting. 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.

Figure 2.8 supplies an operation count, calculated according to the method of section 1.3, for the spatial deformation techniques reviewed in this chapter. Their overall efficiency is dependent on the number of object vertices (n) and, where multiple constraints are supported, on the number of constraints (r). Certain simplifying assumptions have been made in calculating these computation costs. DOGME is evaluated in its simple product form with five axes per constraint;

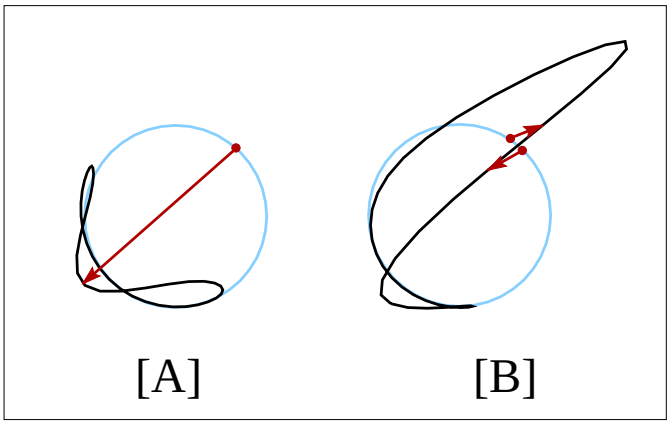


Figure 2.7: **Self-Intersection Caused by Point-based Deformation. [A] A single overextending point constraint. [B] Two nearly singular point constraints with conflicting motion vectors.**

seven levels of recursive subdivision (128 line segments) are used in calculating the Bishop-Klok frames for Axial Deformation; and in Wires it is assumed that the shortest distance from point to curve can be established to sufficient accuracy in twenty search iterations. Entries in figure 2.8 are ranked by increasing operation count, from most efficient (Regular Global Deformation) to least efficient (AxDf). Deformation with lattices of arbitrary topology is notably absent from this table. Its computation cost is affected by many variables (e.g. number and type of cells, levels of subdivision, cell parametrisation accuracy) and it is not amenable to this form of analysis. It is likely, however, that its performance will be the worst of all.

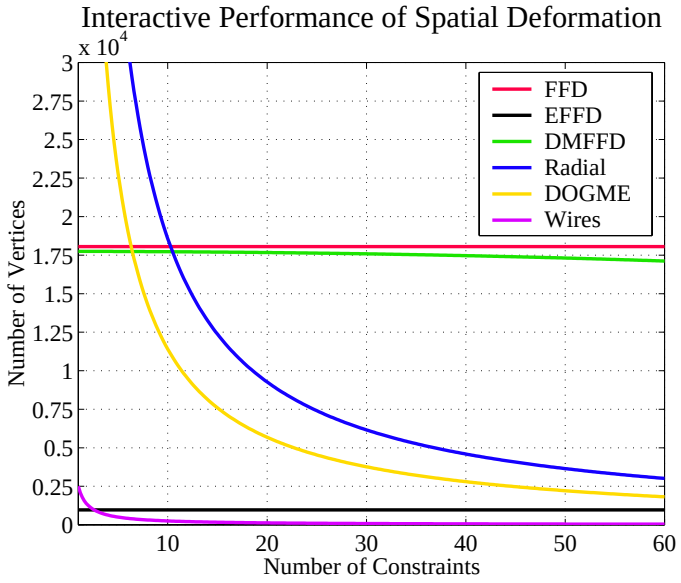
The behaviour of selected techniques at the 145 million additions per second threshold, which roughly translates (see section 1.3) to an interactive ten updates per second, is graphed in figure 2.8. In constraint dependent techniques the non-disjoint case is chosen for comparison purposes. The graph in figure 2.8 demonstrates that the behaviour with respect to the number of constraints varies considerably. Conventional Bézier and Extended FFD are constraint independent. In contrast both the constraint setup and point deformation of Radial, DOGME and Wires deformations are highly constraint dependent and evince a marked decline in the number of deformable vertices with an increase in constraints. DMFFD is a sound compromise with point deformation (but obviously not constraint setup) separated by the hyperpatch from constraint concerns.

2.3.4 Correctness

Topological invariance is a property of spatial deformation. One implication is that certain spatial deformations cause self-intersection of the deformed object. Figure 2.7 reveals two such cases in the context of DMFFD. The first is caused by an elongated constraint displacement reaching beyond the area of effect of the associated constraint point, and the second shows two nearly

singular constraints pulling in opposite directions. Self-intersection invalidates an object for the purposes of some 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. Many sources [Coquillart 1990; Lazarus, Coquillart and Jancène 1994; MacCracken and Joy 1996] mention self-intersection in passing but only Borrel and Rappoport [1994] consider it in detail. Unfortunately, their solution places the onus of identifying and correcting "space-tearing" on the user.

Another problem is that a polygon-mesh object exposed to repeated 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. The candidate schemes proposed to remedy this problem are inadequate for the purposes of interactive sculpting. Nimscheck [1995] presents an adaptation of finite element meshing which produces an optimal triangulation to within a given approximation tolerance, but is too computationally costly for interactive purposes. Parry [1986] and Greissmair and Purgathofer [1989] outline efficient adaptive triangle-mesh refinement methods but do not provide for decimation so that nothing prevents the mesh from becoming wastefully oversaturated with triangles.



Method		Constraint Set-up	Point Deformation	Total Cost
[1] Regular Global		-	118	$118n$
[2] Generalised de Casteljau		-	242	$242n$
[3] FFD	(Bézier)	-	803	$803n$
[4] DMFFD	(disjoint)	$1104r$	817	$1104r + 817n$
	(non-disjoint)	$0.436r^3 + 100r^2$ $+1020r + 43$	817	$0.436r^3 + 100r^2$ $+1020r + 43 + 817n$
[5] Radial	(disjoint)	$57r^2 - 57r$	$57r + 18$	$57r^2 - 57r$ $+57rn + 18n$
	(non-disjoint)	$0.436r^3 + 86r^2$ $+49r + 43$	$78r$	$0.436r^3 + 86r^2$ $+49r + 43 + 78rn$
	(duplicated)	$3.05r^3 + 174r^2$ $-21r + 43$	$114r$	$3.05r^3 + 174r^2$ $-21r + 43 + 114rn$
[6] DOGME	(disjoint)	$49r^2 - 49r$	$49r + 75$	$49r^2 - 49r$ $+49rn + 75n$
	(non-disjoint)	$0.436r^3 + 158r^2$ $-23r + 43$	$127r$	$0.436r^3 + 158r^2$ $-23r + 43 + 127rn$
[7] EFFD		-	15020	$15020n$
[8] Wires	(disjoint)	-	$5498r + 228$	$5498rn + 228n$
	(non-disjoint)	-	$5776r + 19$	$5776rn + 19n$
[9] AxDf		64137	5687	$64137 + 5687n$

Figure 2.8: **Computation Cost of Different Spatial Deformation Techniques. [Bottom] The techniques reviewed in this chapter are ranked by increasing cost and expressed as a function of the number of constraints (r) and object vertices (n). The constraint setup is executed only once but point deformation is applied to every deformable vertex. [Top] The number of vertices deformable at a rate of 10 updates per second is plotted against the number of constraints for six representative techniques.**

Chapter 3

Foundations

3.1 Introduction

Directly Manipulated Free-Form Deformation (DMFFD) [Hsu, Hughes and Kaufman 1992] merges the hyperpatch- and point-based approaches to spatial deformation and enables the user to select and move object points with the surrounding surface conforming smoothly. Broadly speaking DMFFD is a two tier process:

1. Direct Manipulation: the user provides a number of constraints (each a selected point and its desired motion) which are evaluated to find a constraint-satisfying distortion of the volume (hyperpatch) enclosing the target object.

2. Free-Form Deformation: the distortion of the volumetric hyperpatch is transferred to the object.

This can be likened to embedding a deformable shape within a lump of see-through jelly (FFD) and then fastening hooks into the jelly as a means of moulding (Direct Manipulation).

As evidenced in chapter 2, Directly Manipulated Free-Form Deformation embodies a useful balance between ease-of-use, versatility and efficiency. The point constraint mechanism shields the user from the underlying complexities of FFD and also promotes a more direct 'hands-on' style of interaction. In terms of versatility, even though DMFFD allows precise positioning and variable effect for constraint points, the dimensions of their parallelepiped zone of influence cannot be set independently but depend on the FFD hyperpatch substrate. However, this makes the object embedding and deformation procedures independent of the number of constraints, with obvious efficiency benefits. It is this effective compromise between ease-of-use, versatility and efficiency that recommends DMFFD as a foundation for further exploration of spatial deformation.

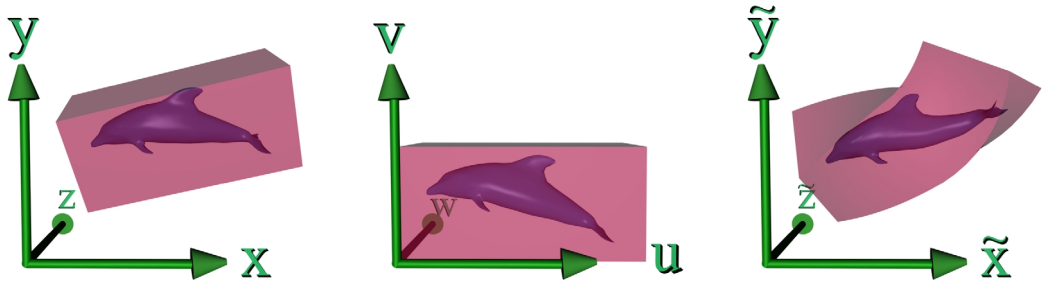


Figure 3.1: **Free-Form Deformation Mappings. A dolphin encased in an FFD hyperpatch (transparent red) is deformed. [Left] Undeformed space $X = (x, y, z)$, [middle] local hyperpatch space $U = (u, v, w)$, and [right] deformed space $\widetilde{X} = (\widetilde{x}, \widetilde{y}, \widetilde{z})$.**

Chapter Structure

In this chapter we develop a complete spatial deformation framework. The principles of Free-Form Deformation are introduced, and their application to tangents and normals, as well as point positions, is considered. This is followed by a description of Direct Manipulation and the complexities involved in matching the user-defined constraints. At every stage of development attention is focused on improving the overall efficiency.

3.2 Free-Form Deformation

Free-Form Deformation (FFD) [Bézier 1978; Sederberg and Parry 1986], like all spatial deformation methods, is a modelling tool which warps the space enclosing an object and thereby transforms the object indirectly. FFD achieves this by first placing a hyperpatch (parametric volume) within world co-ordinate space and then attaching object vertices to the hyperpatch. In this way hyperpatch distortions are transmitted to the object. A helpful analogy is embedding a pliable shape in a block of transparent jelly. Squeezing, twisting, bending, or otherwise moulding the jelly, will result in a corresponding distortion of the inset shape (as in figure 3.1).

The medium of deformation is a parametric hyperpatch, the trivariate volume analogue of a univariate curve. A tensor product approach [Farin 1997, pp. 236-240] is one means of extending from univariate curves, $\mathcal{C}(u) = \sum_{i=0}^{a+\ell-1} \mathcal{N}_i^\ell(u) \cdot P_i$, to bivariate surfaces, $\mathcal{S}(u, v) = \sum_{i=0}^{a+\ell-1} \sum_{j=0}^{b+m-1} \mathcal{N}_i^\ell(u) \cdot \mathcal{N}_j^m(v) \cdot P_{i,j}$. A surface is formed by an evolving curve (in u) swept through space (along v). The next step up in dimension is from bivariate surfaces to trivariate volumes (or hyperpatches). In this instance, the volume is defined by a surface (in u and v) which changes as it moves (along w). A curve's control polygon, which indicates the adjacency of control points, generalises 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

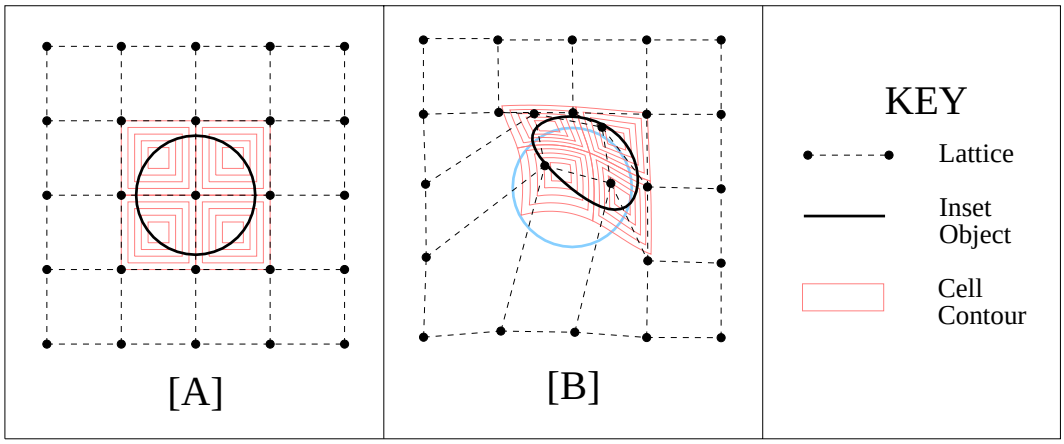


Figure 3.2: **Stages of Free-Form Deformation. [A] Pre-deformation: the initial hyperpatch is configured and the object embedded. [B] Post-deformation: the user repositions control points and the object is deformed.**

decomposed into cells (each covering a block of the trivariate domain). The position of points within a hyperpatch, $\mathcal{H}(u,v,w)$, is governed by a weighted sum of lattice control points, P, and basis functions, $\mathcal{N}_i^n$, with index i and degree n:

$$\mathcal{H}(u,v,w) = \sum_{i=0}^{a+\ell-1} \sum_{j=0}^{b+m-1} \sum_{k=0}^{c+n-1} \mathcal{N}_i^\ell(u) \cdot \mathcal{N}_j^m(v) \cdot \mathcal{N}_k^n(w) \cdot P_{i,j,k} \tag{3.1}$$

where the hyperpatch has $a \times b \times c$ cells and degree $\ell \times m \times n$.

As indicated in chapter 2, Free-Form Deformation can be formulated as a mapping $\mathcal{F} : \Re^3 \mapsto \Re^3 \mapsto \Re^3$, from world co-ordinate space, $X = (x,y,z)$, through the local parameter space of the hyperpatch, $U = (u,v,w)$, to deformed world space, $\widetilde{X} = (\widetilde{x},\widetilde{y},\widetilde{z})$. This is illustrated by figure 3.1 If the hyperpatch control points have pre-deformation, P, and post-deformation, $\widetilde{P}$, arrangements, then this engenders two instantiations of eqn. 3.1: $\mathcal{H}(U) = X$ and $\widetilde{\mathcal{H}}(U) = \widetilde{X}$. Now, FFD is the composition of an embedding, $\mathcal{E}(X) = \mathcal{H}^{-1}(X)$, and deformation function, $\mathcal{D}(U) = \widetilde{\mathcal{H}}(U)$:

$$\mathcal{F}(X) = \mathcal{D} \circ \mathcal{E}(X) = \widetilde{\mathcal{H}}(\mathcal{H}^{-1}(X)) = \widetilde{\mathcal{H}}(U) = \widetilde{X} \tag{3.2}$$

FFD can be broken down into four steps (demonstrated in figure 3.2):

1. The initial (pre-deformation) configuration of the hyperpatch is set by the user ($\mathcal{H}$ is defined).

2. Object vertices are embedded in the pre-deformation hyperpatch by assigning parametric $U = (u,v,w)$ co-ordinates ($\mathcal{E} = \mathcal{H}^{-1}$ is applied).

3. The user repositions a number of control points ($P \mapsto \widetilde{P}$), producing a distorted (post-deformation) version of the hyperpatch ($\mathcal{H} \mapsto \widetilde{\mathcal{H}}$).
4. The parametrised object vertices are all fed into the post-deformation hyperpatch equation ($\mathcal{D} = \widetilde{\mathcal{H}}$) to generate a deformed image of the object.

Each of these four stages has parallels in the jelly metaphor: liquid jelly is poured into a mould, the deformable shape is set inside, the jelly is flexed, and the inset shape warps accordingly.

The development up to this point has glossed over several specifics:

- In this work we favour uniform cubic B-spline basis functions. The cubic polynomial degree offers a reasonable compromise between the computational efficiency of lower degrees and the increased derivative continuity and flexibility (true space curves, points of inflection) of higher degrees. The B-spline basis allows multiple cells and the associated local control, so that individual cells can be distorted without affecting the entire hyperpatch. Lastly, the uniformly spaced knot sequence means that all basis functions are simple translates of each other. In such a case, any parametric co-ordinate within a particular cell can be affinely projected into the unit domain ($u, v, w \in [0, 1]$). The basis function values ($\mathcal{N}_i^3(u), \mathcal{N}_j^3(v), \mathcal{N}_k^3(w)$) are then found by direct polynomial evaluation rather than the more costly Cox-de Boor recursion [Farin 1997, pp. 153-156] which would be mandated by a non-uniform knot sequence.
- There is, in general, no closed form solution for the inverse of the hyperpatch equation and embedding object vertices usually requires iterative root finding. However, if the lattice is arranged as a parallelepiped, anchored at O, and with control points evenly distributed along the axial vectors $\vec{U}, \vec{V}, \vec{W}$, then the linear precision property of B-splines [Farin 1997, p. 158] holds. Vertex embedding is thus reduced to a simple linear co-ordinate transformation into the hyperpatch system, with origin O and axes $\vec{U}, \vec{V}, \vec{W}$ (refer to section 2.2.1). Typically, FFD permits the hyperpatch arbitrary orientation relative to the world co-ordinate axes. If the hyperpatch cells are cubes ($\|\vec{U}\|/a = \|\vec{V}\|/b = \|\vec{W}\|/c$) then this variable orientation does not increase the range of DMFFD, since the hyperpatch is hidden under several layers of indirection. Instead, it unnecessarily complicates the evaluation of local hyperpatch co-ordinates. So, in this work, the $\vec{U}, \vec{V}, \vec{W}$ axes are always aligned parallel to the world co-ordinate axes $\vec{X}, \vec{Y}, \vec{Z}$. The hyperpatch dimensions can be encoded in a single vector S whose components correspond to the span along each axis. If necessary the object can be rotated into a suitable alignment with the hyperpatch.
- There may be cells whose control points are unaltered and which do not perturb object points which fall within them. Instead of applying FFD only to be returned the original point unchanged, a Boolean index, which flags cells with altered control points, can be consulted. In this way, if a control point is moved, then the $(\ell + 1) \times (m + 1) \times (n + 1) = 4 \times 4 \times 4 = 64$ cells that it influences are marked in the index.
- There are two alternatives for parametrising object vertices. Each vertex can be embedded immediately prior to deformation or the entire object can be embedded and then undergo

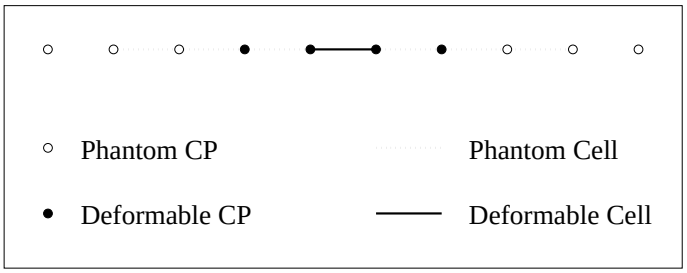


Figure 3.3: **Phantom and Deformable Zones along a Single Hyperpatch Axis. A single deformable cell is enclosed on either side by three phantom cells intended to ensure C^2 continuity across the deformation boundary.**

deformation. While the latter approach more than doubles the vertex storage cost of the object, it has a couple of advantages:

1. Object vertices that are unaltered do not need to be embedded in a subsequent FFD with the same base-state lattice.
2. This approach to FFD is effectively a form of space-partitioning and is useful in enhancing the efficiency of later developments.

- The fringes of the lattice may, without careful attention, produce anomalous continuity degradation. To prevent this, the lattice is partitioned into two shells. At the centre is the deformable zone, with any number of control points and their corresponding cells. Around this lies the 'phantom' zone, which is affected by the movement of control points in the deformable zone and is three cells, or static control points, thick (as outlined by Bartels, Beatty and Barsky [1983, pp. 28-29] and displayed in figure 3.3).

3.3 Deformation of Tangents and Normals

It is often necessary to transform not only the position of object vertices but also their first derivative properties (tangents and normals). These provide a deeper indication of an object's true shape and are essential to rendering algorithms such as Gouraud and Phong shading [Foley *et al.* 1991, pp. 736-739]. The deformation of tangent and normal vectors also underpins direct derivative-frame manipulation (chapter 4) and curvature-based meshing of deformed objects (chapter 5). One obvious method for computing the deformed normal at a vertex is to consider the neighbouring mesh and average the normals of the faces incident on the vertex. This is only an approximation which, unfortunately, depends on the accuracy of the post-deformation polygon mesh. For this reason, a precise (but slightly less efficient) analytic approach is pursued. This Free-Form Deformation of derivatives is best described in three stages, moving from the general to the particular: first the general transformation rules for tangents and normals [Spain 1960, pp. 4-7], then their adaptation by Nimscheck [1995, pp. 68-70] to FFD, and finally the implications of the axis-aligned B-spline hyperpatch favoured in this work.

Provided with a generic function of the form $\mathcal{G}(U) = X : \Re^3 \mapsto \Re^3$ the contravariant and covariant transformation rules [Spain 1960, pp. 4-7] supply a mapping from $U = (u, v, w)$ space to $X = (x, y, z)$ for tangents ($T_U \mapsto T_X$) and normals ($N_U \mapsto N_X$).

$$\text{General Contravariant Transformation:} \quad T_X = \mathbf{J} \cdot T_U \tag{3.3}$$

$$\text{General Covariant Transformation:} \quad N_X = \mathbf{det}(\mathbf{J}) \cdot \mathbf{J}^{-T} \cdot N_U \tag{3.4}$$

$\mathbf{J}$ represents the Jacobian matrix of the function $\mathcal{G}$ and it embodies the derivative distortion at a point as it is mapped from U to X. This type of derivative analysis plays a pivotal role later in the investigation of self-intersection (chapter 7). The Jacobian matrix, $\mathbf{J}$, is composed of component-wise partial derivatives of the associated transformation, $\mathcal{G}$:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathcal{G}_x(U)}{\partial u} & \frac{\partial \mathcal{G}_x(U)}{\partial v} & \frac{\partial \mathcal{G}_x(U)}{\partial w} \\ \frac{\partial \mathcal{G}_y(U)}{\partial u} & \frac{\partial \mathcal{G}_y(U)}{\partial v} & \frac{\partial \mathcal{G}_y(U)}{\partial w} \\ \frac{\partial \mathcal{G}_z(U)}{\partial u} & \frac{\partial \mathcal{G}_z(U)}{\partial v} & \frac{\partial \mathcal{G}_z(U)}{\partial w} \end{bmatrix}$$

Note that $\mathcal{G}_x(U) = x$, $\mathcal{G}_y(U) = y$, and $\mathcal{G}_z(U) = z$ are the component partitions of $\mathcal{G}(U) = X$.

The contravariant and covariant rules have been applied to regular global deformations [Barr 1984], DOGME [Borrel and Bechmann 1991], Scodef [Borrel and Rappoport 1994], and FFD [Nimscheck 1995]. Since FFD relies on composition and inversion, so too must the derivative transformation rules. FFD can be summarised as the mapping:

$$X \xrightarrow{\mathcal{H}^{-1}} U \xrightarrow{\widetilde{\mathcal{H}}} \widetilde{X}$$

If $\mathbf{J}$ and $\widetilde{\mathbf{J}}$ are the non-singular Jacobian matrices of $\mathcal{H}$ and $\widetilde{\mathcal{H}}$ respectively, then the tangent ($T_X \mapsto T_{\widetilde{X}}$) and normal ($N_X \mapsto N_{\widetilde{X}}$) mappings from pre-deformation (X) to post-deformation ($\widetilde{X}$) space are:

$$T_X \xrightarrow{\mathbf{J}^{-1}} T_U \xrightarrow{\widetilde{\mathbf{J}}} T_{\widetilde{X}}$$

$$N_X \xrightarrow{\mathbf{det}(\mathbf{J})^{-1}\mathbf{J}^T} N_U \xrightarrow{\mathbf{det}(\widetilde{\mathbf{J}})\widetilde{\mathbf{J}}^{-T}} N_{\widetilde{X}}$$

As mentioned by Nimscheck [1995, p. 70] scaling factors such as the determinants ($\mathbf{det}(\mathbf{J})^{-1}$, $\mathbf{det}(\widetilde{\mathbf{J}})$) can be discarded because the direction of tangents and normals is important, rather than their magnitude. This idea can be carried even further by evaluating the adjoint of a Jacobian matrix ($\mathbf{J}^\star$) rather than the full inverse ($\mathbf{J}^{-1}$), since they differ by only a scalar coefficient ($\mathbf{J}^{-1} = \mathbf{det}(\mathbf{J})^{-1}\mathbf{J}^\star$) [Glassner 1990, pp. 470-471]. Furthermore, the adjoint is numerically more stable as the determinant approaches zero ($\mathbf{det}(\mathbf{J}) \to 0$). This leads to the FFD transformation rules:

$$\text{Contravariant FFD Transformation:} \quad T_{\widetilde{X}} = \mathbf{J}^\star \cdot \widetilde{\mathbf{J}} \cdot T_X \tag{3.5}$$

$$\text{Covariant FFD Transformation:} \quad N_{\widetilde{X}} = \mathbf{J}^T \cdot \widetilde{\mathbf{J}}^{\star T} \cdot N_U \tag{3.6}$$

In the specific case of Free-Form Deformation with a uniform cubic B-spline basis and an axis-aligned evenly distributed lattice, implementation of these transformations is particularly straightforward. The embedding Jacobian matrix, $\mathbf{J}$, is simply an identity matrix, $\mathbf{I}$. The columns of the deformation Jacobian matrix, $\widetilde{\mathbf{J}}$, are given by:

$$\frac{\partial \widetilde{\mathcal{H}}}{\partial u} = \sum_{i=1}^{a+2} \sum_{j=0}^{b+2} \sum_{k=0}^{c+2} \mathcal{N}_i^2(u) \cdot \mathcal{N}_j^3(v) \cdot \mathcal{N}_k^3(w) \cdot \left[\widetilde{P}_{i,j,k} - \widetilde{P}_{i-1,j,k} \right] \tag{3.7}$$

$$\frac{\partial \widetilde{\mathcal{H}}}{\partial v} = \sum_{i=0}^{a+2} \sum_{j=1}^{b+2} \sum_{k=0}^{c+2} \mathcal{N}_i^3(u) \cdot \mathcal{N}_j^2(v) \cdot \mathcal{N}_k^3(w) \cdot \left[\widetilde{P}_{i,j,k} - \widetilde{P}_{i,j-1,k} \right] \tag{3.8}$$

$$\frac{\partial \widetilde{\mathcal{H}}}{\partial w} = \sum_{i=0}^{a+2} \sum_{j=0}^{b+2} \sum_{k=1}^{c+2} \mathcal{N}_i^3(u) \cdot \mathcal{N}_j^3(v) \cdot \mathcal{N}_k^2(w) \cdot \left[\widetilde{P}_{i,j,k} - \widetilde{P}_{i,j,k-1} \right] \tag{3.9}$$

These transformations will be particularly useful in derivative-frame manipulation (chapter 4) and adaptive mesh refinement and decimation based on the behaviour of deformed normals (chapter 5).

3.4 Direct Manipulation

While Free-Form Deformation has many desirable qualities, such as representation independence, variable scope (either local or global), and smooth sculpted results, the user interface is flawed. As mentioned by Hsu, Hughes and Kaufman [1992], interacting with a three-dimensional lattice of control points can be awkward, time-consuming and confusing, especially for novice users. In the case of univariate curves, the control polygon has proved to be an effective and uncluttered control mechanism. Unfortunately, the generalisation to trivariate hyperpatches damages this ease of use. The display can easily be swamped by the mass of control points and sections of the control lattice often obscure the target object or are hidden within it. Furthermore, achieving anything beyond the most basic deformations requires an intimate understanding of B-splines.

It would be preferable for the user to pick and drag hyperpatch points directly and have the attached solid conform like malleable clay. This is the purpose of Directly Manipulated Free-Form Deformation (DMFFD) [Hsu, Hughes and Kaufman 1992; Borrel and Bechmann 1991]. For example, pushing or pulling a single point on an object's surface will cause either an indentation or extrusion. The richness and complexity of the deformation can be enhanced simply by adding more constraints.

Figure 3.4 demonstrates the different stages underlying DMFFD:

1. The user specifies a set of r constraints, each comprising a point (C_i) and its intended motion (ΔC_i).

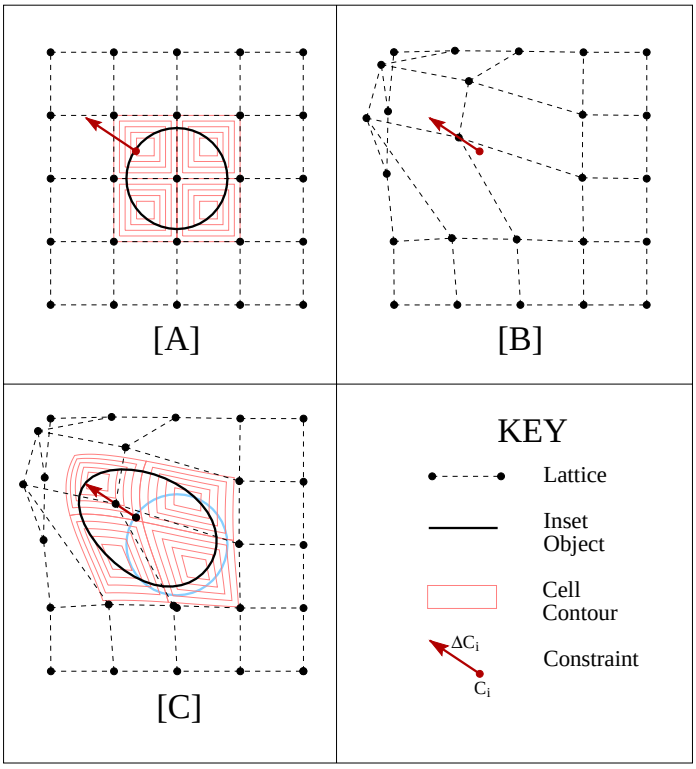


Figure 3.4: **Stages of Direct Manipulation. [A] Constraint and hyperpatch set-up, [B] constraint satisfaction, and [C] object transformation through Free-Form Deformation.**

2. The lattice control points are altered to satisfy these constraints.

3. This new lattice is applied through standard FFD to the original object.

To achieve this, the constraints are substituted into eqn. 3.1 and the results are bundled into a system of linear equations:

$$\begin{aligned} C_i + \Delta C_i &= \sum_{i=0}^{a+\ell-1} \sum_{j=0}^{b+m-1} \sum_{k=0}^{c+n-1} \mathcal{N}_{i,j,k}^{\ell,m,n}(\mathcal{E}(C_i)) \cdot (P_{i,j,k} + \Delta P_{i,j,k}) \qquad \forall i = 1, \ldots, r \\ \mathbf{C} + \Delta\mathbf{C} &= \mathbf{B}(\mathbf{P} + \Delta\mathbf{P}) \\ \Delta\mathbf{C} &= \mathbf{B}\Delta\mathbf{P} \end{aligned} \tag{3.10}$$

where: $\Delta\mathbf{C}$ is an $r \times 3$ matrix with each row holding the constraint displacement vectors, ΔC_i. $\Delta\mathbf{P}$ is an unknown $s \times 3$ matrix capturing the change in lattice control points. It is structured by cycling through ΔC_i and placing in $\Delta\mathbf{P}$, without duplication those control points that influence the constraint under consideration. This means that a control point change is only included if it contributes a non-zero weight to at least one constraint. $\mathbf{B}$ is an $r \times s$ matrix of tensor product

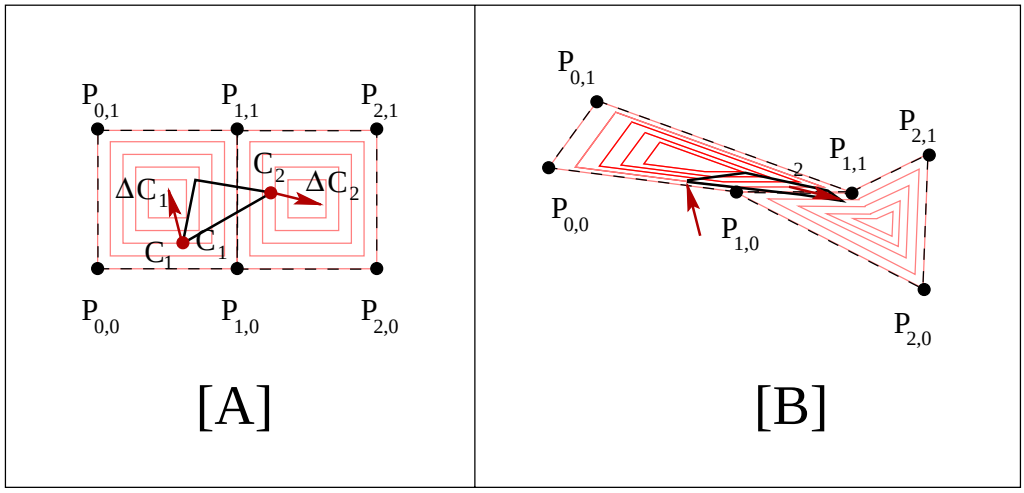


Figure 3.5: **Simple Constraint Example. A triangular object with two point constraints ($[C_1, \Delta C_1]$ and $[C_2, \Delta C_2]$) in separate cells of a linear B-spline hyperpatch with control points $P_{i,j}$. [A] Pre-deformation and [B] post-deformation.**

basis functions evaluated at the constraint points. The weight entries in column j of $\mathbf{B}$ are matched to the control point alteration in row j of $\Delta\mathbf{P}$. It follows that a particular (i, j) element of $\mathbf{B}$ is zero if the control point in row j of $\Delta\mathbf{P}$ has no influence (due to the local control of B-splines) on the constraint point C_i.

Example 3.1 (A simple linear constraint system) *The intricacies of eqn. 3.10 can be clarified by an example. Figure 3.5 shows a two-dimensional linear B-spline lattice in a pre-deformation state with two constraints, $[C_1, \Delta C_1]$ and $[C_2, \Delta C_2]$, inset. Once embedded the constraints have hyperpatch-space co-ordinates: $\mathcal{E}(C_1) = (u_1, v_1)$ and $\mathcal{E}(C_2) = (u_2, v_2)$. The constraints are bound into eqn. 3.10 as:*

$$\mathbf{B} = \begin{bmatrix} \mathcal{N}^1_{0,0}(\mathcal{E}(C_1)) & \mathcal{N}^1_{1,0}(\mathcal{E}(C_1)) & 0 & \mathcal{N}^1_{0,1}(\mathcal{E}(C_1)) & \mathcal{N}^1_{1,1}(\mathcal{E}(C_1)) & 0 \\ 0 & \mathcal{N}^1_{1,0}(\mathcal{E}(C_2)) & \mathcal{N}^1_{2,0}(\mathcal{E}(C_2)) & 0 & \mathcal{N}^1_{1,1}(\mathcal{E}(C_2)) & \mathcal{N}^1_{2,1}(\mathcal{E}(C_2)) \end{bmatrix} \quad (3.11)$$

$$\begin{aligned} \mathcal{N}^1_{j,k}(\mathcal{E}(C_i)) &= \mathcal{N}^1_j(u_i) \cdot \mathcal{N}^1_k(v_i) \\ \Delta\mathbf{P} &= \begin{bmatrix} \Delta P_{0,0} & \Delta P_{1,0} & \Delta P_{2,0} & \Delta P_{0,1} & \Delta P_{1,1} & \Delta P_{2,1} \end{bmatrix}^T \\ \Delta\mathbf{C} &= \begin{bmatrix} \Delta C_1 & \Delta C_2 \end{bmatrix}^T \end{aligned}$$

The second stage of direct manipulation reverses the normal course of FFD. A constraint-matching set of lattice changes ($\Delta\mathbf{P}$) are found, given that $\mathbf{B}$ and $\Delta\mathbf{C}$ are known, by solving the system of linear equations $\mathbf{B}\Delta\mathbf{P} = \Delta\mathbf{C}$. This process is well-defined if $\mathbf{B}$ is square ($r = s$) and non-singular (an inverse, denoted by $\mathbf{B}^{-1}$, exists). Unfortunately, $\mathbf{B}$ is generally underdetermined ($r < s$) and sometimes (rarely) overdetermined ($r > s$). In the former case there are more unknowns than constraints and an infinity of solutions. In the latter case there are more constraints than unknowns and there is no exact solution, since not all of the equations can be satisfied.

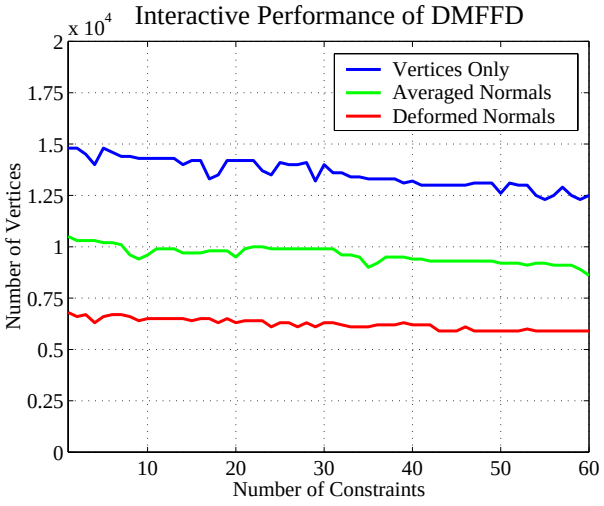


Figure 3.6: **Interactive Performance of Directly Manipulated Free-Form Deformation. The number of vertices deformable at a rate of 10 updates per second is plotted against the number of point manipulation for: (a) deformation of the vertices alone, (b) deformation of the vertices and averaging of normals, and (c) deformation of both vertices and normals.**

To solve for $\Delta\mathbf{P}$ we rely on a formulation known as the pseudo-inverse [Peters and Wilkinson 1970], represented by $\mathbf{B}^{+}$, which extends the definition of the inverse. In the underdetermined situation it minimises 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. In the overdetermined case the closest solution with the smallest residual error norm ($\|\mathbf{B}\Delta\mathbf{P} - \Delta\mathbf{C}\|$) is sought.

The relative performance of three varieties of Directly Manipulated Free-Form Deformation at the threshold of ten updates per second is portrayed in figure 3.6. The number of directly manipulated points is plotted against the number of deformable vertices. The highest level is attained by only deforming position and ignoring surface normals (≈ 12500 vertices at 10 Hz). The performance drops slightly (≈ 9000 vertices at 10 Hz) if the normals are found by averaging the orientation of surrounding faces, and is further decreased (≈ 6500 vertices at 10 Hz) if the normals undergo the FFD transformation described in section 3.3.

3.5 An Efficient Pseudo-Inverse Solution

Evaluating the alteration of lattice control points via the pseudo-inverse is the single costliest computation in Directly Manipulated Free-Form Deformation. So, a careful consideration of the competing pseudo-inverse solutions is advisable. Only the underdetermined case is explored here, since interactive sculpting presupposes a rapid sequence of relatively simple deformations.

Pseudo-Inverse Method	Computation Cost
[1] Greville's Method	$2rs^2$
[2] Householder Factorisation	$r^2s - \frac{1}{3}r^3$
[2] Method of Normal Equations	$\frac{1}{2}r^2s + \frac{1}{6}r^3$

Table 3.1: **Computation Cost of the Pseudo-Inverse Solution Methods. The number of multiplications and divisions required for different calculations of the pseudo-inverse (B^+) of an $r \times s$ matrix (B), with lower order terms discarded.**

Hence, we limit the constraints to 64 ($r < 64$), which is the minimum number of non-zero weighted control points ($s \geq 64$). Three solution schemes appear in the numerical analysis literature and their relative efficiency is displayed in table 3.1.

1. **Greville's Method** [Greville 1960]. Borrel and Bechmann [1991] advocate Greville's derivation for the recursive decomposition of the pseudo-inverse. This is the least efficient approach, but it does have two redeeming attributes [Greville 1960]. Firstly, adding or removing columns in $\mathbf{B}$ does not force a complete recalculation of the pseudo-inverse. Unfortunately, direct manipulation is row oriented in this respect. Secondly, the algorithm is independent of the relative dimensions of r and s and can be applied unchanged in both underdetermined and overdetermined scenarios.

2. **Householder Factorisation** [Golub and Van Loan 1989, ch. 5]. This method is preferred by numerical analysts [Golub and Van Loan 1989, pp. 230-231; Press *et al.* 1992, pp. 674] and is based on an orthogonal decomposition of $\mathbf{B}$, which leads to an effective numerical scheme for least squares solutions. Indeed, Householder factorisation has much to recommend it. The algorithm is both stable and accurate (applicable to a broader, more poorly-conditioned class of matrices), and remarkably compact (since most intermediate steps overwrite entries in the basis matrix $\mathbf{B}$).

3. **Method of Normal Equations** [Golub and Van Loan 1989, pp. 224-225]. The underdetermined pseudo-inverse has the explicit solution:

$$\Delta\mathbf{P} = \mathbf{B}^+\Delta\mathbf{C} = \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}\Delta\mathbf{C}$$

   Hsu, Hughes and Kaufman [1992] employ a naïve construction of the inverse $(\mathbf{B}\mathbf{B}^T)^{-1}$. A more fruitful approach is to exploit the structural properties of $\mathbf{A} = \mathbf{B}\mathbf{B}^T$: $\mathbf{A}$ is symmetric ($\mathbf{A} = \mathbf{A}^T$) and non-negative definite ($x^T\mathbf{A}x \geq 0$). These are requirements for Choleski Decomposition [Press *et al.* 1992, pp. 96-98], a fast technique for solving fully-determined linear systems and implicitly building the inverse. The underdetermined evaluation of $\Delta\mathbf{P}$ proceeds in three steps:

   (a) $\mathbf{A} = \mathbf{B}\mathbf{B}^T$ (The lower triangle is formed by matrix multiplication and the remainder follows by symmetry).

   (b) $\mathbf{D} = \mathbf{A}^{-1}\,\Delta\mathbf{C}$ (by Choleski Factorisation).

   (c) $\Delta\mathbf{P} = \mathbf{B}^T\mathbf{D}$ (by matrix multiplication).

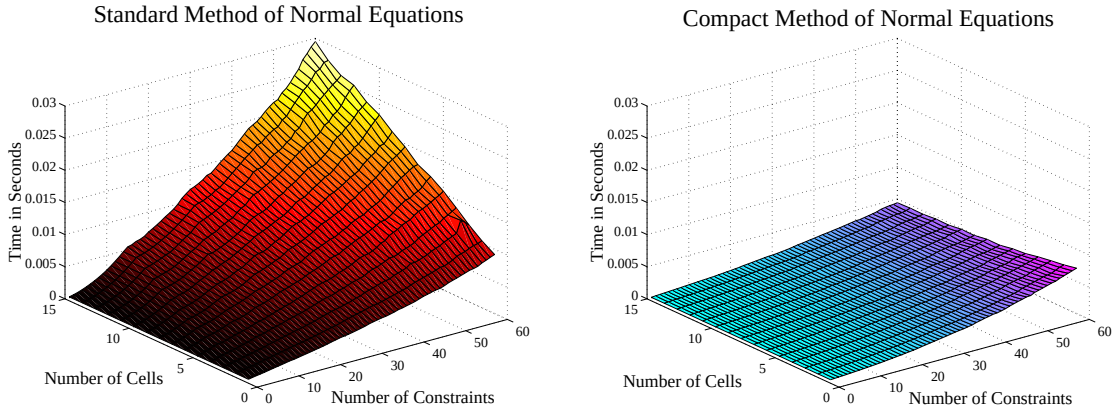


Figure 3.7: **The Impact of a Compact Basis Matrix on the Pseudo-Inverse Calculation. Graphs of two versions of the normal equation method for constraint satisfaction across multiple cells. Ordinary [left] and improved (compact) [right].**

	Constraint Points	**Cells**	s	**Ordinary**	**Compact**
[1]	40	7	160	0.00987 s	0.00268 s
[2]	60	15	288	0.02948 s	0.00442 s
[3]	60	1	64	0.00918 s	0.00706 s

Table 3.2: **Comparison of Normal Equation Methods at Selected Points. [1] Middle, [2] extreme corner, and [3] near right corner. Timings taken on an SGI Octane** 195 **MHz R**10000**.**

On the face of it, Householder factorisation, although losing out marginally in terms of computation overheads, is the appropriate choice for DMFFD. However, both the space consumption and efficiency of the method of normal equations are open to substantial improvement [Gain 1996, ch. 3; Gain and Dodgson 1999b]. This enhancement is based on the sparse nature of $\mathbf{B}$. It is achieved by noting that each row of $\mathbf{B}$ holds 64 non-zero B-spline coefficients, which are weights for a contiguous $4 \times 4 \times 4$ block of lattice control points. The remaining row entries are zero filled. The matrix $\mathbf{B}$ can be replaced by a sparse matrix structure β, which for each row stores the base $I = (i, j, k)$ control point index and the 64 tensor product weights of the corresponding lattice cell.

Steps (a) and (c) of the method of normal equations must be altered to accommodate β. Originally each (i, j) entry of $\mathbf{A} = \mathbf{B}\mathbf{B}^T$ was formed by multiplying corresponding entries in row i and j of $\mathbf{B}$ and summing across the row (i.e. standard matrix multiplication). Instead we find which control points are common to the lattice cubes of row i and j and sum the multiplied weights in the overlapping region. Altering step (c) to take the compact representation into account is straightforward.

Example 3.2 (The compact basis for a simple system) *The constraint system in example 3.1 and figure 3.5 is open to enhancement. A compact structure, β, replaces the equivalent matrix* $\mathbf{B}$ *in equation 3.11:*

$$\beta = \left[\begin{array}{c|cccc} (0,0) & \mathcal{N}^1_{0,0}(\mathcal{E}(C_1)) & \mathcal{N}^1_{1,0}(\mathcal{E}(C_1)) & \mathcal{N}^1_{0,1}(\mathcal{E}(C_1)) & \mathcal{N}^1_{1,1}(\mathcal{E}(C_1)) \\ (1,0) & \mathcal{N}^1_{1,0}(\mathcal{E}(C_2)) & \mathcal{N}^1_{2,0}(\mathcal{E}(C_2)) & \mathcal{N}^1_{1,1}(\mathcal{E}(C_2)) & \mathcal{N}^1_{2,1}(\mathcal{E}(C_2)) \end{array}\right] \tag{3.12}$$

Notice how the zero entries in $\mathbf{B}$ *are made redundant by storing the base index in β. To form entry $\mathbf{A}_{(1,0)}$ from β we note in figure 3.5 that the cells indexed by $(0,0)$ and $(1,0)$ have only $P_{1,0}$ and $P_{1,1}$ in common. Thus:*

$$\mathbf{A}_{(1,0)} = \mathcal{N}^1_{1,0}(\mathcal{E}(C_1)) \times \mathcal{N}^1_{1,0}(\mathcal{E}(C_2)) + \mathcal{N}^1_{1,1}(\mathcal{E}(C_1)) \times \mathcal{N}^1_{1,1}(\mathcal{E}(C_2))$$

The evaluation of direct manipulation now has a worst case cost of $32r^2 + \frac{1}{6}r^3$, which occurs when all constraints lie within the same cell. This improves as constraints are scattered across cells (i.e. efficiency is proportional to the sparsity of $\mathbf{B}$). Further, there is considerable saving in space since $\mathbf{B}$ need never be built explicitly (we have saved $r(s-64)$ floating point storage units). In general the evaluation speed improves with the spread of cells, whereas before it degraded exponentially in both space and time in proportion to the number of cells (see figure 3.7).

It is useful to compare the two methods of Normal Equations at some sample graph points. This is done in table 3.2, where $[1]$ is from the middle of the graph, $[2]$ from the extreme corner, and $[3]$ the near right corner.

Gain [1996, ch. 3] covers a more complete comparison of pseudo-inverse solutions in the context of DMFFD. The compact method of normal equations forms the basis for the implementation of DMFFD in this work.

3.6 Conclusion

Directly Manipulated Free-Form Deformation is a point-based spatial deformation tool constructed in two tiers. At the base is Free-Form Deformation, which operates on the point and normal vectors of an object suspended within a deformable hyperpatch. Above this, direct manipulation acts as an intermediary between the user and the FFD hyperpatch by allowing the imposition of constraints, each composed of a point within the hyperpatch and its intended motion. The FFD hyperpatch is conformed to meet the constraints by efficiently solving a sparse system of linear equations. In this way a point-based pick and drag interface is implemented.

DMFFD strikes a reasonable balance between efficiency, versatility and ease-of-use. Although, each of the components detailed in this chapter contributes a computation overhead, the composite system is still capable of interactive updates ($\geq$ 10 deformations per second) for moderately large objects (with $\leq$ 6500 vertices). On the versatility front, the influence of constraints on the surrounding object can be expanded or reduced by simply resizing the

hyperpatch cells. Also ‘pinning’ constraints (with zero motion) can be used to reshape the deformation boundary. Additionally, the direct manipulation interface improves ease-of-use. Perhaps most importantly, Directly Manipulated Free-Form Deformation provides a firm foundation for exploring further enhancements to the versatility and correctness of spatial deformation.

Chapter 4

Derivative-Frame Manipulation

4.1 Introduction

Directly Manipulated Free-Form Deformation enables a user to interactively pick and drag multiple points on and around an object, which immediately induces a conforming deformation. This can be likened to pulling and prodding clay using a set of thin metal rods with tiny pincers. While this point-based interaction is better than the complex and indirect hyperpatch-based interface of basic Free-Form Deformation, the versatility and ease-of-use remain restricted. Expanding the types of constraints beyond point positioning is one avenue of improvement. A useful first step is the development of direct derivative-frame manipulation. The normal vector and tangent plane at a surface point constitutes a derivative frame and can be represented by an arrow and planar rectangle. This derivative frame is open to tilting, twisting and scaling with DMFFD extended to correspondingly distort the surrounding object.

Chapter Structure

This chapter details the development of derivative-frame manipulation. Firstly, the process of binding partial derivative changes at a point into the linear constraint system of DMFFD is described, and secondly, a user interface for defining these partial derivatives is presented. Fowler [1992] develops a constraint mechanism for the derivative manipulation of bivariate surfaces. Our contribution lies in the extension of these constraints to trivariate hyperpatches and the mechanism of deriving constraints from changes made by the user to a derivative frame. An outline of derivative-frame manipulation for DMFFD appears in Gain and Dodgson [1999b].

4.2 First Derivative Constraints

DMFFD provides control over the position of selected constraint points within an FFD hyperpatch. The intention of derivative-frame manipulation is to extend the constraint mechanism to also encompass control of the first derivatives. Derivative-frame manipulation will enable a user to modify the partial derivatives at a point while pinning its position. Each derivative-frame manipulation has several parameters: an orthonormal frame originating at a point, C, and consisting of normal, binormal and tangent vectors, $D = (\vec{N}, \vec{B}, \vec{T})$, and the intended frame alteration, specified by a local scaling, $S = (s_{\vec{N}}, s_{\vec{B}}, s_{\vec{T}})$, and rotation, by θ about the vector $\vec{R} = (r_{\vec{N}}, r_{\vec{B}}, r_{\vec{T}})$, of the derivative frame. Direct manipulation proceeds by incorporating the derivative-frame constraints into a system of linear equations (possibly alongside conventional point constraints) and then, as before, solving for the control point changes and executing Free-Form Deformation. The technique developed here is an extension of work by Fowler [1992] on the derivative manipulation of bivariate surfaces.

DMFFD requires that the derivative-frame manipulation be related to a hyperpatch through a set of linear constraints. The partial hyperpatch derivatives provide a first step towards this goal and the next section will demonstrate the connection between these hyperpatch derivatives and any given derivative frame.

Section 3.3 shows that the partial derivative of the hyperpatch with respect to u ($\frac{\partial \widetilde{\mathcal{H}}}{\partial u}$) can also be expressed as a hyperpatch with lower degree in u ($l \mapsto l-1$) and control vectors taken from the $\vec{U}$-oriented lattice edges. The other two partial hyperpatch derivatives ($\frac{\partial \widetilde{\mathcal{H}}}{\partial v}$ and $\frac{\partial \widetilde{\mathcal{H}}}{\partial w}$) are similar. This does not directly relate tensor product weights to individual control points as is required for a linear constraint system. One solution is to return to the original hyperpatch equation and take the derivative of the appropriate basis function.

$$\frac{\partial \widetilde{\mathcal{H}}}{\partial u} = \sum_{i=0}^{a+2} \sum_{j=0}^{b+2} \sum_{k=0}^{c+2} \frac{d\mathcal{N}_i^3(u)}{du} \cdot \mathcal{N}_j^3(v) \cdot \mathcal{N}_k^3(w) \cdot P_{i,j,k} \tag{4.1}$$

$$\frac{\partial \widetilde{\mathcal{H}}}{\partial v} = \sum_{i=0}^{a+2} \sum_{j=0}^{b+2} \sum_{k=0}^{c+2} \mathcal{N}_i^3(u) \cdot \frac{d\mathcal{N}_j^3(v)}{dv} \cdot \mathcal{N}_k^3(w) \cdot P_{i,j,k} \tag{4.2}$$

$$\frac{\partial \widetilde{\mathcal{H}}}{\partial w} = \sum_{i=0}^{a+2} \sum_{j=0}^{b+2} \sum_{k=0}^{c+2} \mathcal{N}_i^3(u) \cdot \mathcal{N}_j^3(v) \cdot \frac{d\mathcal{N}_k^3(w)}{dw} \cdot P_{i,j,k} \tag{4.3}$$

Another alternative is to algebraically manipulate eqns. 3.7- 3.9 into a suitable control point form. This latter alternative is more appropriate for non-uniform knot sequences where evaluation requires Cox-de Boor recursion and polynomial coefficients are not immediately available.

The change in partial derivatives ($\Delta C^{\langle u \rangle}$, $\Delta C^{\langle v \rangle}$, $\Delta C^{\langle w \rangle}$) at a point ($C$) can be combined,

together with the condition that C remain stationary into a single subsystem of linear equations.

$$\begin{bmatrix} \mathcal{N}(\mathcal{E}(C)) \\ \mathcal{N}^{\langle u \rangle}(\mathcal{E}(C)) \\ \mathcal{N}^{\langle v \rangle}(\mathcal{E}(C)) \\ \mathcal{N}^{\langle w \rangle}(\mathcal{E}(C)) \end{bmatrix} \Delta\mathbf{P} = \begin{bmatrix} 0 \\ \Delta C^{\langle u \rangle} \\ \Delta C^{\langle v \rangle} \\ \Delta C^{\langle w \rangle} \end{bmatrix} \tag{4.4}$$

The rows of this system are a conversion of eqns. 3.1, 4.1, 4.2 and 4.3 to matrix form. Each basis vector encodes the flattening of a lattice structure of tensor product weights:

$$\begin{aligned}
\mathcal{N}(\mathcal{E}(C)) &= \mathcal{N}(u_C, v_C, w_C) = \mathit{flatten}(\mathcal{N}_i^3(u_C) \cdot \mathcal{N}_j^3(v_C) \cdot \mathcal{N}_k^3(w_C)) \\
\mathcal{N}^{\langle u \rangle}(\mathcal{E}(C)) &= \mathcal{N}^{\langle u \rangle}(u_C, v_C, w_C) = \mathit{flatten}(\frac{d\mathcal{N}_i^3(u_C)}{du} \cdot \mathcal{N}_j^3(v_C) \cdot \mathcal{N}_k^3(w_C)) \\
\mathcal{N}^{\langle v \langle}(\mathcal{E}(C)) &= \mathcal{N}^{\langle v \rangle}(u_C, v_C, w_C) = \mathit{flatten}(\mathcal{N}_i^3(u_C) \cdot \frac{d\mathcal{N}_j^3(v_C)}{dv} \cdot \mathcal{N}_k^3(w_C)) \\
\mathcal{N}^{\langle w \rangle}(\mathcal{E}(C)) &= \mathcal{N}^{\langle w \rangle}(u_C, v_C, w_C) = \mathit{flatten}(\mathcal{N}_i^3(u_C) \cdot \mathcal{N}_j^3(v_C) \cdot \frac{d\mathcal{N}_k^3(w_C)}{dw})
\end{aligned}$$

These weights are matched to the unknown control point changes in $\Delta\mathbf{P}$.

A complete system of constraint equations $\mathbf{B}\Delta\mathbf{P} = \Delta\mathbf{C}$ is built by appending point constraints (a single row for each point) and derivative-frame constraints (four rows for each derivative frame) to $\mathbf{B}$ and $\Delta\mathbf{C}$. After this, DMFFD continues, as described in chapter 3, to configure the lattice in accordance with the constraint system and invoke FFD with this new lattice.

It is possible to manipulate second (and even higher) derivatives (as is done in a surface context by Georgiades and Greenberg [1992]) but this suffers from two main disadvantages:

1. As pointed out by Fowler [1992] such manipulations are not as geometrically intuitive and do not allow the creation of tools which are easy to understand and manipulate.
2. First derivatives require 4 rows in the constraint system (one to fix the point position and 3 for the partial derivatives). This can be contrasted against the 13 rows required for second derivatives (4 to fix position and first derivatives and 9 for the second order partial derivatives). This means that fewer constraints of this kind can be included simultaneously before reaching overdetermination and each second derivative constraint contributes to impeding interactivity.

4.3 User Interaction

An interface based on textual input of the partial derivative constraints goes against the spirit of interactive sculpting and is likely to be painstaking and difficult to predict. For this reason we develop a means of directly manipulating the first derivative frame at a point.

Section 3.3 demonstrates the relevance of the Jacobian matrix in considering derivative distortion at a point. The Jacobian matrix of deformation, $\widetilde{\mathbf{J}}$, embodies the final state of the partial derivatives. Thus, if a derivative constraint (C, $\Delta C^{\langle u \rangle}$, $\Delta C^{\langle v \rangle}$, $\Delta C^{\langle w \rangle}$) is satisfied we expect:

$$\widetilde{\mathbf{J}}(C) = \left[\begin{array}{ccc} (C^{\langle u \rangle})^T & (C^{\langle v \rangle})^T & (C^{\langle w \rangle})^T \end{array} \right] \tag{4.5}$$

$$\Delta\widetilde{\mathbf{J}}(C) = \widetilde{\mathbf{J}}(C) - \mathbf{I} = \left[\begin{array}{ccc} (\Delta C^{\langle u \rangle})^T & \Delta C^{\langle v \rangle})^T & \Delta C^{\langle w \rangle})^T \end{array} \right] \tag{4.6}$$

Eqn. 4.6 captures the change in partial derivatives between undeformed (identity transform, $\mathbf{I}$) and deformed (deformation Jacobian matrix, $\widetilde{\mathbf{J}}$) states.

$\widetilde{\mathbf{J}}$ can also be regarded as a 3×3 affine transformation matrix combining scaling, rotation and shear operations. This interpretation is the key to extracting the constraints from a derivative-frame manipulation. The individual changes to a derivative frame can be concatenated into a single transformation, $\mathbf{T}$, which aligns the frame with the hyperpatch, $\mathbf{D}^{-1}$, scales and rotates relative to the hyperpatch axes, $\mathbf{S}$ and $\mathbf{R}$, and then returns to the original frame orientation, $\mathbf{D}$:

$$\widetilde{\mathbf{J}}(C) = \mathbf{T} = \left(\mathbf{DRSD}^{-1}\right) \tag{4.7}$$

$$\mathbf{D} = \left[\begin{array}{ccc} \vec{N}^T & \vec{B}^T & \vec{T}^T \end{array} \right] \qquad \mathbf{S} = \left[\begin{array}{ccc} s_{\vec{N}} & 0 & 0 \\ 0 & s_{\vec{B}} & 0 \\ 0 & 0 & s_{\vec{T}} \end{array} \right]$$

$$\mathbf{R} = \left[\begin{array}{ccc} r_{\vec{N}}^2 + c_\theta(1 - r_{\vec{N}}^2) & r_{\vec{N}} r_{\vec{B}}(1 - c_\theta) + r_{\vec{T}} s_\theta & r_{\vec{T}} r_{\vec{N}}(1 - c_\theta) + r_{\vec{B}} s_\theta \\ r_{\vec{N}} r_{\vec{B}}(1 - c_\theta) + r_{\vec{T}} s_\theta & r_{\vec{B}}^2 + c_\theta(1 - r_{\vec{B}}^2) & r_{\vec{B}} r_{\vec{T}}(1 - c_\theta) - r_{\vec{N}} s_\theta \\ r_{\vec{T}} r_{\vec{N}}(1 - c_\theta) - r_{\vec{B}} s_\theta & r_{\vec{B}} r_{\vec{T}}(1 - c_\theta) + r_{\vec{N}} s_\theta & r_{\vec{T}}^2 + c_\theta(1 - r_{\vec{T}}^2) \end{array} \right]$$

where $s_\theta = \sin(\theta)$ and $c_\theta = \cos(\theta)$.

The user interactively inputs a derivative constraint by first selecting a surface point (C) on the deformable object. The derivative frame, with an arrow and rectangle for the normal vector and tangent plane, is then displayed at the picked point. The user provides the rotation and scaling parameters, S and $(\theta, \vec{R})$, perhaps by spinning a virtual trackball [Glassner 1990, pp. 462-463]. From this information the derivative constraints ($C, \Delta C^{\langle u \rangle}, \Delta C^{\langle v \rangle}, \Delta C^{\langle w \rangle}$) are obtained using equations 4.6 and 4.7. Interactive feedback can be supplied by constantly updating the derivative frame and re-applying DMFFD.

A variety of effects can be achieved through different transformations of the derivative frame.

- **Tilting the tangent plane.** This induces a crest or bank in the surface.
- **Rotation around the normal vector.** The surface beneath assumes a vortex-like twist.
- **Scaling the derivative frame.** An increase or decrease in the scaling factor will cause the surface to swell or deflate accordingly.

4.4 Conclusion

An example of derivative-frame manipulation is shown in figure 4.1. By transforming the first derivative frame, an initial object undergoes successive tilting, scaling and twisting around the constraint point. Of course some or all of these effects can be combined within a single deformation. Derivative-frame manipulation has no physical correspondence with any one clay sculpting tool but it is still valuable. The interaction mechanism is relatively simple and easy-to-use and the end-results are useful and predictable. The computation overheads are not substantial since vertex embedding and deformation are unaffected. The only impact is on the constraint solution, since each derivative-frame constraint, instead of one row, occupies four. In order to maintain an underdetermined system fewer derivative frame constraints (less than 16 within a single cell) are allowed. On the other hand derivative-frame constraints expand the range of easily achievable deformations and combine seamlessly with point constraints (the position of the derivative frame need not be pinned) and curve constraints (as will be explored in chapter 6). In summary, derivative-frame manipulation is an effective first step in enhancing the versatility of Directly Manipulated Free-Form Deformation.

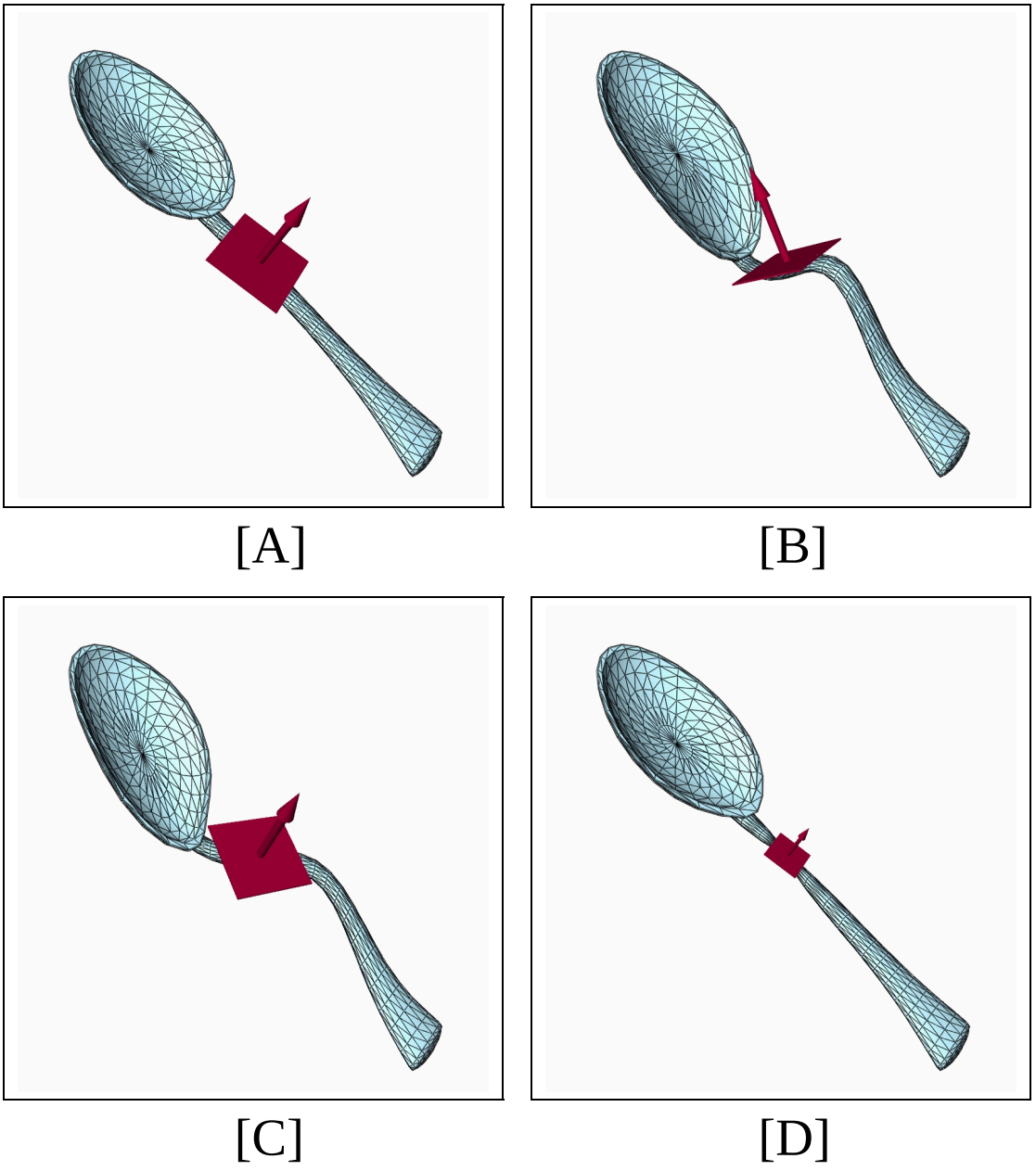


Figure 4.1: **Spoon Bending. Manipulating a derivative frame (red) consisting of a normal vector (arrow) and tangent plane (rectangle): [A] initial state, [B] tilting, [C] twisting and [D] shrinking.**

Chapter 5

Adaptive Refinement and Decimation

5.1 Introduction

The most convenient format for storing, manipulating and displaying a deformable object is the polygon-mesh representation [Hoscheck and Lasser 1993, p. 489]. Rendering by polygon scan-conversion requires that a continuous object be approximated by a polygon mesh. An early conversion to polygon-mesh form (*prior* to deformation) can be motivated by two concerns: a clean separation between any mesh modification and the underlying representation (be it B-spline, implicit or CSG), and the ability to efficiently apply a possibly lengthy sequence of deformations to the object in an interactive modelling context. However, rendering polygon-mesh objects which have undergone spatial deformation is problematic. The primary difficulty is that this approximation degrades under deformation, as previously flat (and sparsely covered) regions become sharply undulating. Conversely, highly curved areas may be warped to near planarity and be wastefully oversaturated with polygons. These problems can be avoided by coupling an adaptive polygon-mesh refinement and decimation scheme to FFD. Another issue is that the vertices of high-order polygons with four or more edges will typically be shifted out of planar alignment by deformation. This problem is easily rectified by insisting on a triangle-element mesh.

Chapter Structure

This chapter presents an adaptive refinement and decimation scheme [Gain and Dodgson 1999a] that is based on the research of Greissmair and Purgathofer [1989] and Nimscheck [1995, ch. 6]. While this scheme is tailored to repeated interactive DMFFD, it remains applicable to other spatial deformation methods. Our contributions are as follows:

1. We incorporate a complementary decimation process which merges faces in nearly planar

areas and thereby reduces the polygon-mesh complexity (number of vertices, edges and faces).

2. We trigger refinement or decimation by testing the deviation between surface normals at edge endpoints (adapted from Nimscheck [1995, pp. 111-115]). As will be shown, this has efficiency advantages over Greissmair and Purgathofers' [1989] midpoint displacement condition.

3. We identify and develop measures to ameliorate the serious sampling problems associated with refinement and decimation under FFD.

The remainder of this chapter is divided into four sections: a means of ranking subdivision schemes is extracted from the meshing literature and adapted to the requirements of spatial deformation; previous approaches to subdivision in the context of FFD are discussed and compared; our improvements on the previous methods are presented; and the new algorithms are vetted in terms of the previously established metrics of subdivision quality.

5.2 Measuring Subdivision Quality

It becomes necessary, when choosing among existing subdivision schemes or devising new ones, to assess both the performance of the subdivision algorithm and the quality of the resulting mesh. The goal is a method which accepts any type of input mesh, executes in real-time, and generates a triangle mesh that closely approximates the 'true' shape with the minimum number of triangles and without artefacts. We focus in turn on each of the attributes that constitute an ideal subdivision scheme within the practical context of spatial deformation.

5.2.1 Type of Input Mesh

Historically, the adaptive subdivision of height field meshes (scalar elevations associated with a two-dimensional domain) is prevalent [Brown 1998, pp. 14-28]. This is driven by the real-time demand for locally detailed terrain in flight simulation. Attention has also been devoted recently to non-manifold approximations [Ronfard and Rossignac 1996; Popović and Hoppe 1997] which alter topology (e.g. the number of holes) during refinement or decimation and allow reduction of any model to a minimal form, such as a tetrahedron. However, interactive sculpting acts on solids whose topological features are significant and worth preserving. Manifold meshing schemes [Schroeder, Zarge and Lorensen 1992; Hoppe *et al.* 1993; Klein, Liebich and Straßer 1996] satisfy this requirement.

The restriction of objects to a manifold form is widely adopted in computer graphics [Foley *et al.* 1991, p. 542]. The neighbourhood of any point on the surface of a manifold object is homeomorphic to an open disc in $\Re^2$ [Hoffmann 1989, p. 38]. This property is contravened,

for instance, if more than two faces adjoin at a common edge because the neighbourhood of any point on that edge is a sheaf which cannot be flattened into a single continuous disc. A manifold surface may exist with or without boundary. The latter correctly embodies a single closed continuous (C^0) surface. The former has a boundary whose point neighbourhoods are homeomorphic to a half-disc. While manifolds with boundaries expose both sides of a surface and are thus less representative of true solids, they are nevertheless widely exploited in computer graphics. For instance, the pervasive Utah teapot has a boundary around the rim of its spout. In the interest of enabling both solid and surface modelling, adaptive subdivision under deformation should support both types of manifold.

Also at issue are the geometric qualities of the input mesh. Forcing the initial mesh to be sampled from a smooth (C^1 or C^2 continuous) object is unnecessarily limiting. The user may want to mould a surface with sharp creases, which might typically be the product of a separate modelling process such as Constructive Solid Geometry (CSG) [Foley *et al.* 1991, pp. 557-558]. This should be allowed as it advances versatility and generality.

5.2.2 Interactive Response

As always with interactive sculpting the impact of an extension on the overall efficiency is a key concern. Many otherwise excellent subdivision schemes fail on this count. Either, they are simply too computationally costly (e.g. Delaunay triangulation [Hoscheck and Lasser 1993, pp. 402-408]) or they supply different resolutions in real time, but only if the model is static and available for preprocessing (e.g. Progressive Simplicial Complexes [Popović and Hoppe 1997]).

5.2.3 Close Approximation

The primary purpose of subdivision after FFD is obtaining a closer approximation to the 'true' surface of the deformed object. Two issues need resolving before the quality of such an approximation can be established. Firstly, a means of measuring the difference between the 'true' and approximating surface is needed. The Haussdorf distance, $\mathcal{K}(S_1, S_2)$, which is a standard measure of the consistency of two surfaces (S_1 and S_2) [Klein, Liebich and Straßer 1996], is useful in this regard.

$$\mathcal{K}(S_1, S_2) = max(max_{Y \in S_2} min_{X \in S_1} \|X - Y\|, max_{Y \in S_1} min_{X \in S_2} \|X - Y\|) \qquad (5.1)$$

Roughly speaking, this max-min metric considers the shortest distance from a point on the first surface, $X \in S_1$, to any point on the second, $Y \in S_2$, and selects the largest of all the shortest distances. Then in an inverted fashion the maximum of the distances from a point on S_2 to the closest point on S_1 is found. The Haussdorf distance is the largest of the two results.

Secondly, the 'true' surface must somehow be found. This is a trivial consideration for surface simplification by pure decimation, as researched by Schroeder, Zarge and Lorensen [1992]; and

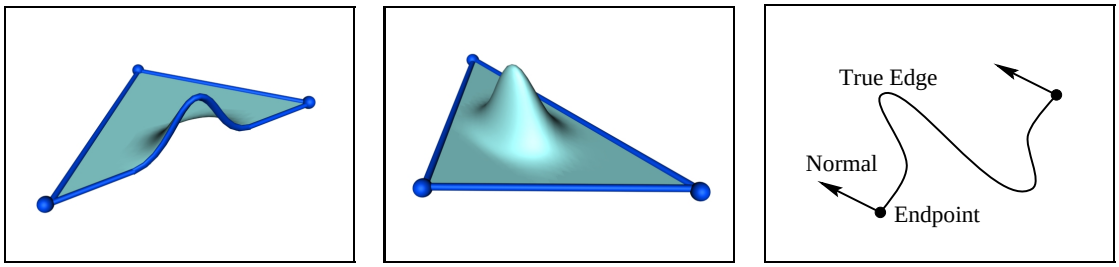


Figure 5.1: **Sampling Issues. [left] Deformation local to an edge with vertices unaffected. [middle] Deformation local to a triangle with vertices *and* edges unaffected. [right] Despite equality of the normals an edge may still require refinement.**

Klein, Liebich and Straßer [1996], because the initial mesh can be taken as 'true'. The task is far more complex in the case of spatial deformation. Consider a curved surface prior to any distortion. This can be precisely represented by an infinitely and uniformly subdivided mesh, S_∞. The 'true' surface is the image of this mesh under deformation, $\widetilde{S}_\infty$. It is obviously infeasible to infinitely subdivide a mesh, so instead we use $\widetilde{S}_r$, where the degree of subdivision, r, is high and $\widetilde{S}_r$ is sufficiently close to optimal ($\mathcal{K}(\widetilde{S}_r, \widetilde{S}_\infty) < \widetilde{\varepsilon}_r$).

An approximate upper bound on the Haussdorf distance $\widetilde{\varepsilon}_r$ can be obtained as follows. Assume that for the uniformly subdivided mesh in its pre-deformation state, S_r, we know the maximum edge length, l_r, and are given the pre-deformation approximation error, $\varepsilon_r = \mathcal{K}(S_r, S_\infty)$. If S_r interpolates a simple analytic surface, such as a sphere segment, then ε_r is easy enough to derive. Now, a bound on the post-deformation error is:

$$\widetilde{\varepsilon}_r \quad \leq \quad \varepsilon_r + \left(\frac{l_r}{s}\right)^2 \cdot \rho \qquad l_r < s \tag{5.2}$$

where s is the minimum cell extent relative to the hyperpatch axes and ρ is the magnitude of the largest control point displacement, $\rho = max(\|\Delta P_{i,j,k}\|)$.

This result depends on the convex hull property of B-splines [Farin 1997, pp. 158-159] and the quadratic convergence rate of B-spline subdivision [Dahmen 1986]. An initial bound, ρ, on the error introduced during deformation is provided by the convex hull of the control point displacements. As the cells of the deformed hyperpatch are subdivided, Dahmen [1986] stipulates that the control lattice will converge quadratically towards the actual hyperpatch. In this process the deformation error bound which is linked to the convex hull of the lattice displacements will also reduce quadratically. If the edges of a subdivided mesh are limited to the length l_r then the relative size of subdivided pre-deformation cells capable of containing triangles from this mesh is $\frac{l_r}{s}$. Once this degree of subdivision is known, an error bound, $\widetilde{\varepsilon}_r$, for the deformed mesh, $\widetilde{S}_r$, can be established with eqn. 5.2.

We prefer the global Haussdorf distance to more local error metrics, such as those proposed by Garland and Heckbert [1997] and Erikson and Manocha [1999], because of the serious sampling issues inherent in spatial deformation. If there is a mismatch between the scope of

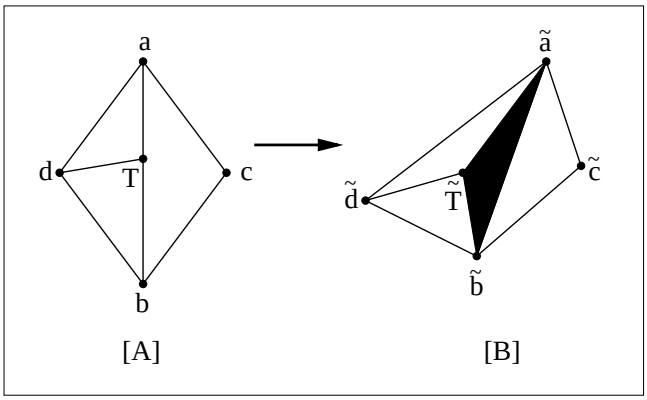


Figure 5.2: **A T-Vertex Induced Gap. A Vertex which lies on, but does not segment, an edge [A] may result in a gap (black triangle) after deformation [B].**

a deformation and the length of mesh edges, then distortion of edge endpoints is not a reliable guide to subdivision. One eventuality (figure 5.1[left]) is that the inner section of an edge may be deformed without involving the endpoints. Even worse (figure 5.1[middle]), a deformation may take place within a triangle and leave all edges and vertices unaffected. A separate but related difficulty occurs if testing for refinement or decimation involves the surface normals. The endpoint normals of an edge may not diverge significantly even though the edge undulates inbetween (figure 5.1[right]). In general, sampling theory dictates that such situations cannot be completely avoided since pathological cases can always be constructed. However, the problem can be curtailed by adjusting the sampling density for different deformation situations.

5.2.4 Limiting the Number of Triangles

Increasing the density of the mesh over regions with high curvature, in order to faithfully capture sharp fluctuations and fine detail, is only half the story. If mesh coverage is not symmetrically reduced in flat areas then under a sequence of deformations the object's complexity may grow without bound even if its overall curvature decreases. Adaptive refinement needs to be balanced by decimation since an overly-detailed nearly-planar triangulation is worse than superfluous. Each unnecessary element contributes to storage, rendering and deformation costs. If adaptive refinement of a polygon mesh is regarded as containing the inevitable approximation error within an upper limit, then the reciprocal decimation can be viewed as establishing a lower bound.

5.2.5 Preventing Artefacts

There are two distinct artefacts which can be introduced through careless subdivision. The first, *T-vertices* tend to induce unforseen gaps in an otherwise continuous mesh [Nimscheck 1995, pp. 79-80]. A T-vertex coincides with the edge of an adjacent triangle but does not subdivide it. For

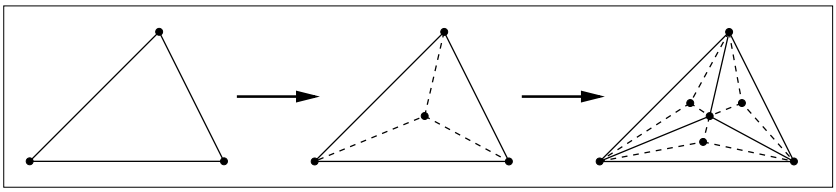


Figure 5.3: **Slivers Caused by Ternary Subdivision. Subdividing triangles at their centrepoint leads after only a few iterations to thin ill-formed triangles.**

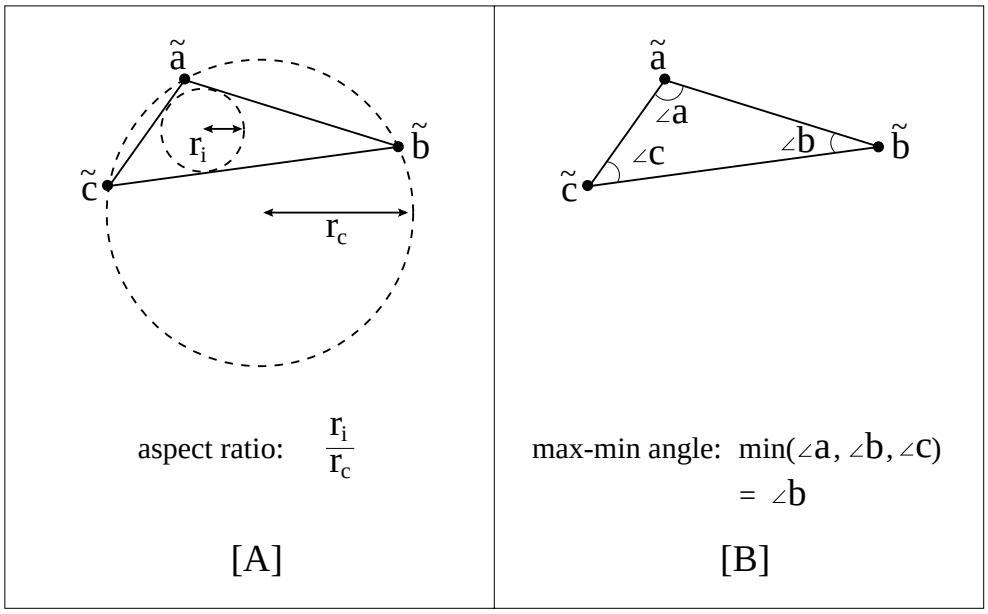


Figure 5.4: **Two Measures of Triangle Quality. [A] Aspect Ratio: the ratio of the inscribed radius (r_i) to the circumscribed radius (r_c). [B] Max-Min Angle: the size of the smallest angle in the triangle.**

instance, in figure 5.2[A] the vertex T is an element of triangle aTd and bdT, but even though it lies on the segment ab it does not belong to acb. Deformation will invariably displace the T-vertex away from its parent edge with a resulting tear in the fabric of the mesh (figure 5.2[B]).

A less overt problem is the creation of *slivers*. These thin elongated triangles tend to cause visual discontinuities in interpolated shading [Foley *et al.* 1991, pp. 734-739], numerical inaccuracies in finite element modelling [Brown 1998, p. 20], and difficulties in form factor calculation for radiosity [Baum, Rushmeier and Winget 1989]. Ternary subdivision, in which an inserted vertex is connected with radiating edges to the corners of its enclosing triangle, is notorious for outputting slivers. Figure 5.3 illustrates serious sliver formation after only two levels of ternary subdivision.

With regard to slivers there are two widely-accepted measures of triangle quality, ranging from triangles collapsed into a line segment in the worst case to equilateral triangles in the best. The *max-min angle* criterion [Schumaker 1993] considers the size of the smallest interior angle (figure 5.4[B]). The closer this lies to a maximum value of $\frac{\pi}{3}$ radians the more equilateral the triangle. The Delaunay triangulation is nearly optimal in this respect as it attempts to maximise the minimum angle over the entire mesh. A popular alternative metric is the *aspect ratio* [Nimscheck 1995, p. 81] defined as the ratio of the inscribed and circumscribed circles (figure 5.4[A]). As the aspect ratio approaches one half the triangle becomes more equilateral.

5.3 Subdivision Schemes Specific to Spatial Deformation

There have been three previous approaches to subdivision in the context of Free-Form Deformation: Parry's "long side", Greissmair and Purgathofers' "worst edge" and Nimscheck's advancing-front subdivision. Table 5.1 shows which desirable subdivision features are supported by these schemes.

Criterion	Parry	Greissmair and Purgathofer	Nimscheck
Allows Boundaries	×	×	√
Handles Creases	√	√	√
Efficient	√	√	×
Addresses Sampling	×	×	√
Provides Decimation	×	×	×
No T-vertices	√	√	√
Good Aspect Ratio	√	√	√

Table 5.1: **A Comparison of the Three Subdivision Schemes Specific to Spatial Deformation.**

5.3.1 Parry's "Long Side" Subdivision

Parry [1986, pp. 46-55] can be credited with the first FFD-specific subdivision method. His focus is on the adaptive subdivision of deformed primitives (blocks, spheres and cylinders) within a Constructive Solid Geometry (CSG) system. The solid primitives are represented by an initially regular triangle mesh.

Parry's adaptive subdivision begins by identifying one edge in every triangle as a "long side". This terminology is somewhat misleading. The "long side" is not necessarily the longest edge. It merely indicates which edge in the triangle is due for subdivision. All triangles in the object are placed into a stack from which candidates are drawn until the stack is empty. Subdivision of a candidate triangle is prompted by a combination of screen- and object-space criterion. Both

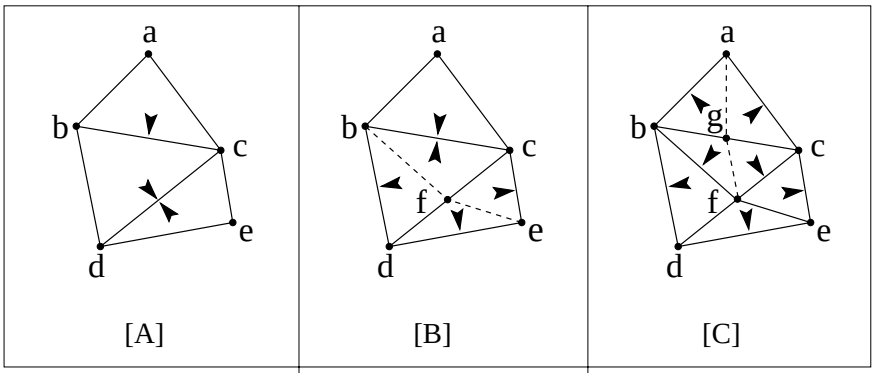


Figure 5.5: **Parry's Recursive Search and Midpoint Subdivision. [A] The original mesh with "long sides" marked by arrow heads. Edge bc is the subdivision target but it is not marked by both adjacent triangles (acb and bcd). [B] A suitable edge cd is found after a recursive search and then subdivided. [C] This in turn allows the original target bc to be subdivided.**

(a) the horizontal and vertical screen extent of the projected triangle, and (b) the divergence between the candidate's normal and those of its three edge-adjacent neighbours are tested to see if they exceed user-defined thresholds. If either of these conditions are met then the candidate and the adjacent triangle sharing its "long side" are subdivided in a section-halving fashion at the midpoint of the "long side". However, this is only allowed if the adjacent triangle also classifies the common edge as its "long side".

As demonstrated by figure 5.5 this requirement can be attained by recursively subdividing the surrounding mesh. With each section-halving the unmarked edges in the two pre-subdivision triangles become "long-sides" for the four post-subdivision triangles. In this way the edge at the root of the recursion will eventually be tagged as a "long side" by the incident triangle. At the end of each subdivision the newly formed triangles are pushed onto the candidate stack. The only danger is that, as evidenced by figure 5.6, cycles within the mesh will prevent the recursive search from terminating. These cycles may occur if care is not taken at the initial marking stage by, wherever possible, assigning pairs of "long side" labels to the same edge.

The main purpose of Parry's "long side" labelling convention is to ensure a reasonable aspect ratio among the subdivided fragments. Section-halving is used to guarantee that no T-vertices are introduced. Parry [1986, p. 54] handles discontinuities in the input mesh, such as the join between the top and sides of a block primitive, by disabling curvature testing across the pertinent edges. Taken as a whole, this scheme is fairly efficient.

On the downside, Parry's adaptive subdivision does not provide decimation or allow for manifold meshes with boundaries. Further, the screen-space subdivision test of a triangle's projected extent depends on specific viewing parameters and is inappropriate for view-independent modelling. Finally, the only consideration given to sampling issues is the insistence on a fine-grained initial mesh [Parry 1986, p. 53], but this is hardly sufficient.

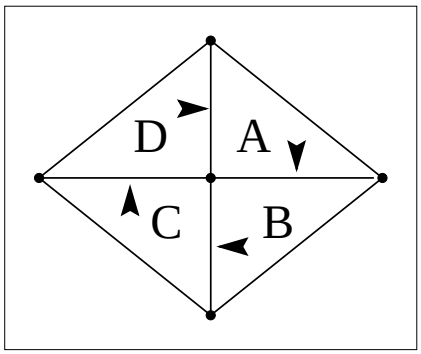


Figure 5.6: **A "Long Side" Cycle. A recursive subdivision search will cycle indefinitely through the triangles A, B, C and D without finding a double marking.**

5.3.2 Greissmair and Purgathofers' "Worst Edge" Subdivision

Greissmair and Purgathofer [1989], like Parry [1986], refine a triangle-element polygon mesh with section-halving but in other respects their FFD subdivision strategy is dissimilar. They examine edges rather than faces and trigger subdivision based on the deviation of a midpoint instead of either screen size or the angle between face normals.

Greissmair and Purgathofer [1989] process a post-deformation triangle mesh by running every edge through a midpoint deviation test. Those that fail are inserted into a refinement heap ordered by decreasing edge length, so that the "worst edge" lies on top. After this, edges are repeatedly drawn off the heap and split at the midpoint through section-halving of the incident triangles. The post-deformation position of the new vertex ($\widetilde{m}$) of a split edge ($\widetilde{ab}$) is simply the deformed image of the pre-deformation midpoint ($\widetilde{m} = \mathcal{F}(m) = \mathcal{F}(midpoint(ab))$). The four new edges spawned by section-halving are then tested for their midpoint deviation and placed in the heap as necessary. Extraction and subdivision of edges continues until the heap is emptied.

The midpoint deviation test, demonstrated in figure 5.7, examines the post-deformation images of the midpoint, $\widetilde{m}$, and incident triangles, $\widetilde{abd}$ and $\widetilde{acb}$. The distances, d_1 and d_2, from the deformed midpoint, $\widetilde{m}$, to the plane of the deformed triangles, $\widetilde{abd}$ and $\widetilde{acb}$, are measured and if their sum exceeds a threshold value ($d_1 + d_2 > \varepsilon$) then the edge ab is targeted for subdivision. A simpler option is to consider the difference between the deformed midpoint of edge ab ($\widetilde{m} = \mathcal{F}(midpoint(ab))$ and the midpoint of the deformed edge $\widetilde{ab}$ ($\widetilde{e} = midpoint(\widetilde{ab})$). Unfortunately, unlike the previous metric, this will trigger subdivision for deformations which lie in the plane of the surface and Greissmair and Purgathofer [1989] consider this undesirable.

Greissmair and Purgathofers' technique has many of the same strengths and weaknesses as Parry's "long side" subdivision. The "worst edge" ordering of the heap means that longer edges are subdivided first and a balanced aspect ratio is promoted and, as already mentioned, section-halving guards against T-vertex formation. The problem of creases (discontinuities) in the input mesh is not an issue because subdivision depends on the displacement of vertices rather

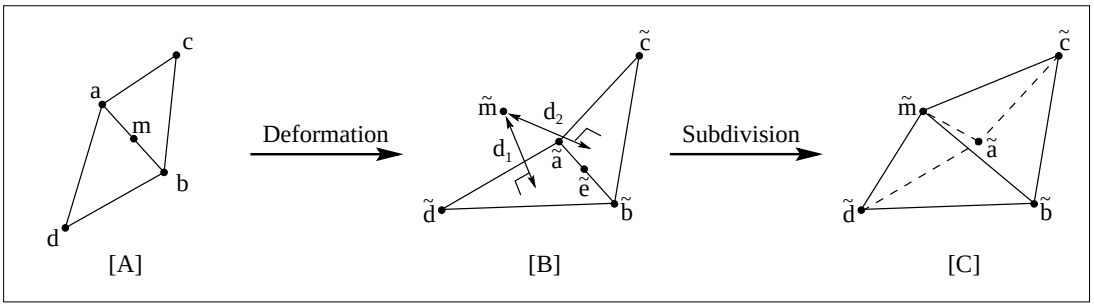


Figure 5.7: **"Worst Edge" Testing and Subdivision. [A] Two pre-deformation triangles adjoining edge ab with midpoint m. [B] The post-deformation distances from $\widetilde{m}$ to the triangles $\widetilde{acb}$ and $\widetilde{abd}$ is measured. [C] If necessary four new triangles which include $\widetilde{m}$ are introduced.**

than the angle of normal vectors. Greissmair and Purgathofers' method is the less efficient of the two. This is solely due to the extra Free-Form Deformation demanded by the midpoint deviation test. For example, an object with n edges, even if it undergoes no subdivision, still requires n midpoint calculations and deformations. The actual subdivision procedure is faster since no recursion is needed.

On the weaknesses front, Greissmair and Purgathofer dismiss sampling issues summarily as being too expensive to deal with. Also, like Parry [1986], they do not explore decimation or boundaries in the manifold mesh.

5.3.3 Nimscheck's Advancing-Front Subdivision

Domain discretisation (or meshing) is a deeply explored topic in finite element analysis with much relevance to the problem of subdivision under FFD. Nimscheck [1995] recognises this and adapts an advancing-front meshing algorithm to FFD. His strategy is highly effective. A completely general polygon mesh with many-sided convex or concave polygons is submitted as input and a triangulated approximation, whose normal and tangent vectors are within a given error tolerance of the true object, is returned as output. The biggest drawback is the high degree of computation involved.

As a preliminary, Nimscheck's algorithm fragments the edges of the initial object. An edge is recursively halved based on either its post-deformation length or the disparity between its endpoint tangents. Afterwards each polygon boundary will, depending on the coarseness of the starting mesh, consist of a chain of many short edges. In the advancing-front stage each polygon is processed in turn. Its edges are stored on an "active front" heap that is sorted by increasing length. In diametric contrast to Greissmair and Purgathofers' method the shortest edge is always first on the heap. The actual triangulation takes place in a two-dimensional domain. Edges are successively drawn off the heap until none remain (the polygon is tessellated). Each extracted

edge forms the base of a new triangle whose apex is either an existing vertex in the "active front" heap or inserted to build an isosceles triangle. The apex selection process shrinks the resulting triangle in accordance with the divergence after deformation of local tangents and normals. It also ensures that new triangles have a good aspect ratio and do not overlap or pass near existing triangles. Next, any new edges are inserted into the "active front" heap and the algorithm either continues on with the subsequent iteration or terminates.

Nimscheck's advancing-front subdivision is only deficient in two respects: decimation of an overly-detailed initial mesh is not enabled and the computation costs are exorbitant. For example, a subdivision which introduces only 322 triangles executes in 2.3 seconds on an SGI Indy workstation [Nimscheck 1995, p. 119], which is well beyond the interactivity cutoff.

On the positive side, because subdivision is on a per polygon basis there is no difficulty with creases or manifold boundaries. Nimscheck's scheme avoids T-vertices, allows complex input polygons, which may even have interior holes, and carefully addresses sampling problems by adjusting the length of subdivided edges in tune with the scope of deformations.

5.4 An Adaptive Refinement and Decimation Algorithm

It is now possible, with these foundations in place, to present our curvature-based adaptive refinement and decimation algorithm. This can be visualised as stacking refinements in successive layers, which later decimations will remove in reverse order. It is assumed that the initial input mesh prior to any deformation is a reasonably close approximation to the "true" object in the sense that adjacent vertex normals in curved areas do not diverge too greatly. At the same time there should be no oversaturation of polygons because our algorithm is designed not to decimate the base mesh configuration.

As a preliminary, all vertices and edges in the original (undeformed) polygon mesh are tagged to indicate that they are at the base level of the refinement hierarchy. After each successive deformation the algorithm scans all edges and categorises a subset of these as candidates for either refinement or decimation. This testing takes place in post-deformation space but the subsequent re-meshing must be done in pre-deformation space. The edge test can be regarded as a trigger condition for refinement or decimation.

If either
(a) at least one of the endpoints of an edge has been deformed and the normals at these endpoints diverge after FFD by more than a certain threshold angle (θ_{max}) *or*
(b) the edge is longer than a certain maximum length ($l_{\mathcal{H}}$ if neither endpoint is within the hyperpatch or $l_{\mathcal{C}}$ otherwise),
then the edge is placed on the refinement heap. Conversely, *if both*
(a) the angle between deformed endpoint normals is below a decimation threshold (θ_{min}) *and*
(b) the edge is shorter than the defined maximum length ($l_{\mathcal{H}}$ or $l_{\mathcal{C}}$ as appropriate),
then the edge's endpoints are marked as possible candidates for vertex-centred decimation.

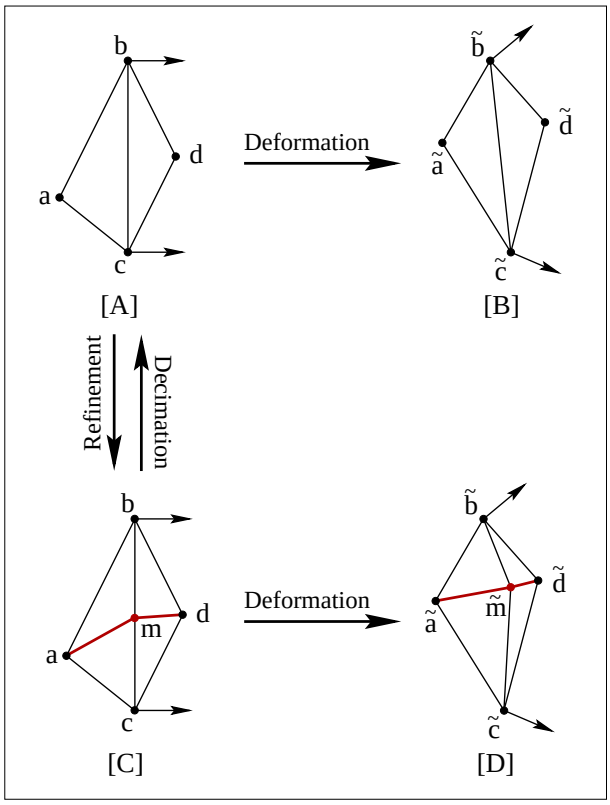


Figure 5.8: **Edge Splitting and Merging. Refinement: an initial edge bc in [A] is tested in post-deformation space [B] and if so indicated split at the midpoint [C] and then deformed [D]. Decimation: An initial vertex m in [C] is tested in post-deformation space [D] and if it passes, the surrounding topology is merged $[A]$ and deformed [B]. Edges and vertices with higher refinement levels are in red.**

The refinement stage (adapted from Greissmair and Purgathofer [1989]) repeatedly extracts edges from the refinement heap until none remain. This heap is ordered by decreasing edge length. This dictates an order of subdivision that tends to reduce the number of ill-formed (sliver) triangles by splitting long edges first. Each extracted edge, bc, is subdivided according to figure 5.8 and the four new edges am, bm, cm, dm are tested for refinement and if necessary inserted into the refinement heap. The only subtlety is that a refinement level, r, is assigned to the vertex m and the two new edges, am, dm. This level is set at one more than the previous highest level amongst the vertices a, b, c, d: $r = max(a_{level}, b_{level}, c_{level}, d_{level}) + 1$ The remaining edges, bm, cm, are assigned the same level as their source edge bc.

Decimation proceeds by grouping candidate vertices according to their refinement level. These groups are processed in descending order so that later refinements are removed first. A candidate vertex, m, is decimated only if it has:
(a) exactly four incident edges which all pass the decimation test *and*
(b) all four adjacent vertices at a lower level than m.

		Endpoint Normal	Midpoint Distance
Algorithm	Pre-Trigger	$\varepsilon = 1 - \widetilde{n}_\ell \cdot \widetilde{n}_r$	$\widetilde{m} = \mathcal{F}(midpoint(e))$ $m^\star = midpoint(\widetilde{e})$ find normals ($\widetilde{f}_\ell$, $\widetilde{f}_r$) of faces incident on $\widetilde{e}$ $d = \widetilde{m} - m^\star$ $d_1 = \lvert\widetilde{f}_\ell \cdot d\rvert$ $d_2 = \lvert\widetilde{f}_r \cdot d\rvert$ $\varepsilon = d_1 + d_2$
	Post-Trigger	$\widetilde{m} = \mathcal{F}\ (midpoint(e))$	-
Number of Operations	Pre-Trigger	$3a + 3m$	$255a + 320m + 15d + 2s$
	Post-Trigger	$225a + 296m + 6d$	0
Total Cost	Pre-Trigger	6.21	629.81
	Post-Trigger	553.18	0

Table 5.2: **An Efficiency Comparison of the Endpoint Normal and Midpoint Distance Tests. The calculations which take place during and after an edge-test are considered separately. e is the edge prior to FFD, and $\widetilde{e}$ is the post-FFD version with normal endpoints $\widetilde{n}_\ell$ and $\widetilde{n}_r$.**

In this case decimation proceeds by exactly reversing a previous refinement. Figure 5.8 is also a pictorial representation of decimation in a vertex neighbourhood. The vertex m and its two incident edges am and dm at the same refinement level are removed.

There are a number of intricacies pertaining to this algorithm that are worth examining.

5.4.1 Triggering Refinement/Decimation

Our test of edges for refinement/decimation suitability has two elements. The first condition is Nimscheck's [1995] endpoint normal divergence. If $\widetilde{n}_\ell$ and $\widetilde{n}_r$ are the unit normals at the left and right vertices of an edge, then the expression $\varepsilon = 1 - \widetilde{n}_\ell \cdot \widetilde{n}_r$ is an efficient substitute for the (small) angle, θ between $\widetilde{n}_\ell$ and $\widetilde{n}_r$. The property $\varepsilon = 1 - cos(\theta)$ holds. The use of normals is economical since they must, in any event, be available for rendering purposes. This endpoint normal divergence test is chosen over Greissmair and Purgathofers' [1989] midpoint distance

test because of its efficiency advantages (as shown in table 5.2). In particular, the midpoint deformation is shifted from before the trigger to afterwards and this becomes significant when there is a high volume of non-triggering edge tests. The only question that remains is how to evaluate the endpoint normals prior to subdivision. There are two possibilities:

1. The normal at a vertex is averaged from the normals of the surrounding triangles.
2. The attitude of the normal is tracked through deformation using the covariant FFD transformation rule (section 3.3).

This represents a classic trade-off between efficiency and accuracy. The former option is faster but the latter has greater fidelity. A third option, not explored in this dissertation is to extract the normal from one or more cubic curves interpolating the central and neighbouring vertices.

The second trigger condition is the length of the candidate edge, len. This is designed to ameliorate the sampling problems exemplified in figure 5.1. At a coarse level, the test ensures that no triangle can intersect the hyperpatch without at least one of its component vertices doing so, by subdividing edges that are longer than the shortest side of the hyperpatch ($len > l_{\mathcal{H}}$). At a finer level, the test guarantees that triangles within the hyperpatch do not span the bulk of a local deformation ($len > l_{\mathcal{C}}$). In practice $l_{\mathcal{C}}$ is set to the minimum cell extent. This two layered approach is a careful compromise between oversampling and undersampling.

The following pseudocode encapsulates the complete refinement/decimation trigger:

```
 RefDecTest(θ_max , θ_min , l_H , l_C)
Input:  θ_max (refinement angle), θ_min (decimation angle),
        l_H (hyperpatch length), l_C (cell length).
Output:  'Refine', 'Decimate' or 'Pass Over'.

IF e.len > l_H THEN
  RETURN 'Refine'
ELSE
  IF InsideHyperpatch(e.vert_ℓ) OR InsideHyperpatch(e.vert_r) THEN
    IF e.len > l_C THEN
      RETURN 'Refine'
    ELSE
      ε = 1 − ñ_ℓ · ñ_r
      IF ε > 1 − cos(θ_max) THEN
        RETURN 'Refine'
      ELSE
        IF ε < 1 − cos(θ_min) THEN
          RETURN 'Decimate'
RETURN 'Pass Over'
```

5.4.2 Splitting Edges

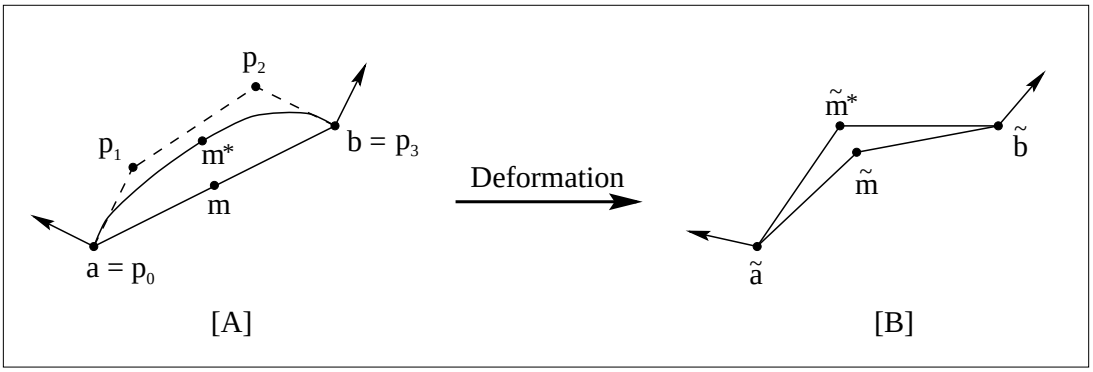


Figure 5.9: **Cubic Bézier Midpoint Interpolation. [A] Pre-deformation: a Bézier curve is constructed from the endpoints and their normals. A midpoint is found on the Bézier curve ($m^\star$) rather than the edge (m). [B] Post-deformation: $\widetilde{m}^\star$ may provide a smoother approximation than $\widetilde{m}$.**

During section-halving subdivision the location of the new vertex, $\widetilde{m}$, is conventionally [Parry 1986; Greissmair and Purgathofer 1989] the deformed image of the midpoint, m, which halves the original edge, ab. The endpoint normals, n_a and n_b, of edge ab are not involved in any way. Another route, which follows the implied curve of the object by considering the normals, is to select the halfway point, $m^\star = \mathcal{C}(0.5)$, of a cubic Bézier curve, $\mathcal{C}(t)$, laid over ab (figure 5.9). The first and last control points of this curve are set at the edge endpoints ($P_0 = a$, $P_3 = b$). The interior curve control points (P_1 and P_2) are positioned a distance $\frac{1}{3}\|ab\|$ along the endpoint tangents, t_a and t_b. Each tangent lies in the plane spanned by its associated normal (n) and edge (ab), and is also orthogonal to n. This cubic Bézier interpolation represents another tradeoff of efficiency against accuracy.

5.4.3 Boundaries and Creases

An edge on the boundary of a manifold mesh has only a single incident triangle and demands special consideration. All edges and vertices on the boundary are marked. On encountering a boundary edge the refinement loop creates a bounding vertex at the midpoint and splits only one triangle rather than two. Likewise, decimation is modified to seek boundary vertices with three incident edges and, if so indicated, reverse a single triangle subdivision.

Another concern is the impact of discontinuous features, like creases and corners, on adaptive meshing. A single normal vector assigned to a crease vertex will inevitably initiate unwarranted fragmentation of one or more of the radiating edges. Consider the vector at the corner of a cube (figure 5.10[left]). It must form an angle of at least $\frac{1}{4}\pi$ with one of the normals belonging to an abutting face. In most cases severe refinement will be signalled as a matter of course, even with little or no deformation. Clearly this is objectionable, but it can be resolved by associating

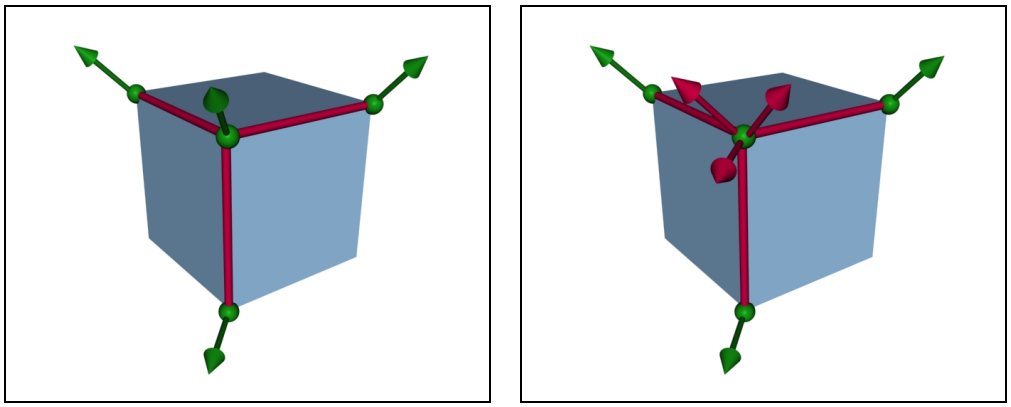


Figure 5.10: **Vertex Normals on Crease Edges. [Left] A single normal is assigned to every vertex (green) and leads to unwanted normal divergence. [Right] The corner vertex has normals associated with every incident edge (red) and this prevents divergence.**

normals with every edge leading into a crease vertex (figure 5.10[right]). A pre-deformation endpoint normal can be calculated on demand by averaging the triangle normals on either side of the pertinent edge. Once deformed this normal can be employed in refinement and decimation tests on the edge.

5.4.4 Implementation Details

In our implementation the triangle mesh is encoded by a winged-edge data structure [Baumgart 1972; Foley *et al.* 1991, pp. 545-546]. Here, pointers relate edges to their endpoints, incident triangles and bounding edges. All the topological and geometric information necessary for refinement is immediately available to hand. Finding the edges radiating from a vertex for the purposes of decimation requires slightly more indirect pointer following. The winged-edge structure favours access over mesh updates. It obviates the need for searching through the vertex, edge or face lists but complicates edge splitting and merging with the intricacies of adding and rerouting pointers. The design of a streamlined data structure, tailored to the specific tasks involved in refinement and decimation, would be a useful future enhancement.

A straightforward refinement heap with a binary tree structure is also employed. A potential improvement would be to select a heap that was optimised for frequent heap operations as established by profiling the algorithm. For example, Greissmair and Purgathofer [1989] proposed a pagoda heap.

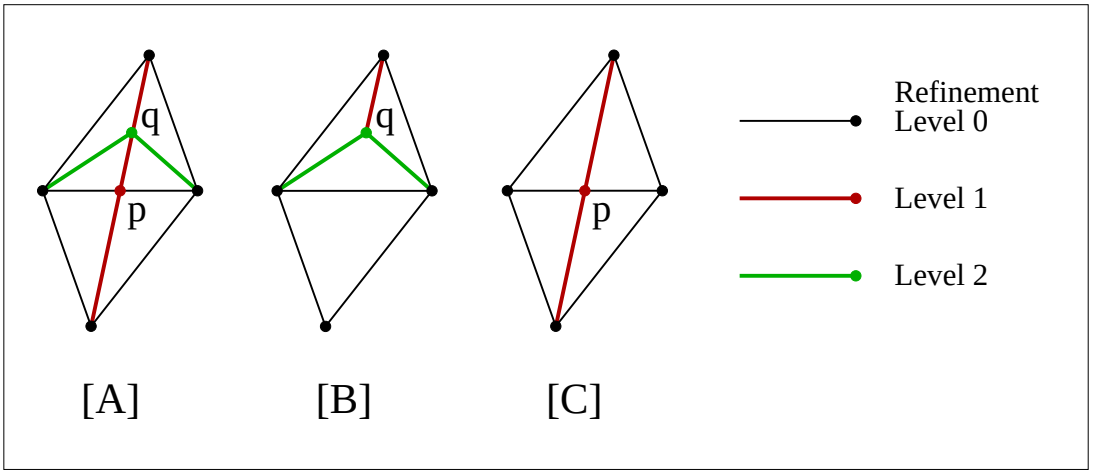


Figure 5.11: **The Importance of Ordered Decimation. [A] Pre-decimation. [B] Illegal decimation of vertex p prevents further decimation. [C] Legal decimation of vertex q allows later decimation of p.**

5.4.5 Decimation

The decimation procedure is restricted to vertices with a very specific surrounding topology produced during the refinement stage and is applied in a specific order, which is controlled by the refinement level labelling of edges and vertices. These limitations are justified for several reasons:

1. No decimation of vertices belonging to the original base mesh is allowed since the surrounding vertices cannot have a lower refinement level ($r \not< 0$). The refinement/decimation algorithm is consequently restricted to overlaying topological detail. This ensures that important aspects in the original topology (such as feature edges) are maintained.

2. The topology restrictions have considerable efficiency benefits. The alternative is a scheme which allows any number of edges incident on the decimation vertex (rather than just four). For example, Schroeder, Zarge and Lorensen [1992] achieve this by removing the vertex and all incident edges and then triangulating the resulting gap. The problem is thus reduced to tessellating an arbitrary polygon, but this represents a significant additional overhead.

3. The ordering constraint ensures that refinement is fully reversible. Ideally, if a region is distorted from planar to convoluted and back to planar over a sequence of Free-Form Deformations, it should return to its original topology. Given the topology restrictions imposed on decimation, this is only possible (as illustrated by figure 5.11) if refinements are placed in a stack from which only the top element can be removed by decimation.

5.5 Evaluation

Figures 5.12 and 5.13 demonstrate both the necessity of adaptive refinement and decimation, and the efficacy of our technique. In the first case, a slightly curved patch has undergone an extruding deformation and a range of subsequent refinements, controlled by a normal deviation angle (α). There may be a scattering of subdivided sliver triangles but the average aspect ratio remains high across all depths of refinement. Another measure of mesh quality is the Haussdorf distance of various refinements from the true surface, S_∞. We approximate this ideal with a 65536 element triangle mesh, S_8, that can be bounded to within $\mathcal{K}(\widetilde{S}_8, \widetilde{S}_\infty) < 0.0026$ by utilising eqn. 5.2 (with $l_8 = 0.0201, s = 0.769, \rho = 3.829, \varepsilon_8 = 3.3 \times 10^{-6}$). The approximation quality improves in a stair-step fashion with increasing refinement. The decrease in Haussdorf distance is not always consistent because the max-min locale does not always coincide with the region of greatest normal divergence. Figure 5.12 also displays the exponential relationship between the birth of new subdivided triangles and the decrease in refinement angle. The asymptote of this graph can be shifted left or right by adjusting the coarseness or fineness of the pre-deformation mesh and the severity of the deformation, but the overall relationship will remain intact. More reassuring is the execution economy of refinement (1000 new triangles spawned in less than 0.02 seconds) and its linear relationship with the amount of subdivision.

To evaluate the action of decimation a second flattening deformation is imposed on the peak of the rightmost mesh in figure 5.12. Notice that without decimation the centre of the nearly level plateau is now wastefully oversaturated with polygons. As the normal deviation angle (α) is relaxed, previous refinements are peeled away until the original topology is reinstated. This process is exceptionally cheap as topological structure is discarded without needing any geometric calculation. In this instance, because the decimation is instituted over an almost planar area there is no appreciable degradation in the Haussdorf distance.

This appraisal is based on covariant normal deformation and linear midpoint interpolation. Evaluating midpoints with cubic Bézier interpolation (section 5.4.2) has little impact on either efficiency or accuracy in this instance. We have found that the Bézier midpoint solution only improves perceptual smoothness when the preliminary mesh is particularly jagged (coarse). Averaging the triangle normals (section 5.4.1) also proved ineffective because infinite refinement can arise if the refinement angle approaches or exceeds the inherent deviation of normals in the initial model.

5.6 Conclusion

In this chapter we have extended and improved Greissmair and Purgathofers' [1989] adaptive refinement algorithm by:

- incorporating an efficient, fully symmetrical decimation scheme that maintains the

underlying initial topology,

- addressing sampling problems caused by a disparity between the scope of a deformation and the size of individual faces in the polygon-mesh object,
- reducing the computation cost of the refinement/decimation trigger condition.
- allowing boundaries and creases in the manifold input mesh and hence catering for a broader class of models.

In addition, like previous schemes, our algorithm prevents degenerate T-vertices that inject unexpected and undesirable gaps into the mesh and, wherever possible, avoids slivers by striving for equilateral triangles. We also cater for discontinuities in mesh smoothness at creases, folds and corners. The overall efficiency of our algorithm is such that any amount of decimation and a reasonable degree of refinement (< 1000 subdivisions) can be achieved in real-time. If more detailed models are prescribed, as in high-end animation, then more extreme adaptive refinement can be relegated to a post-processing stage.

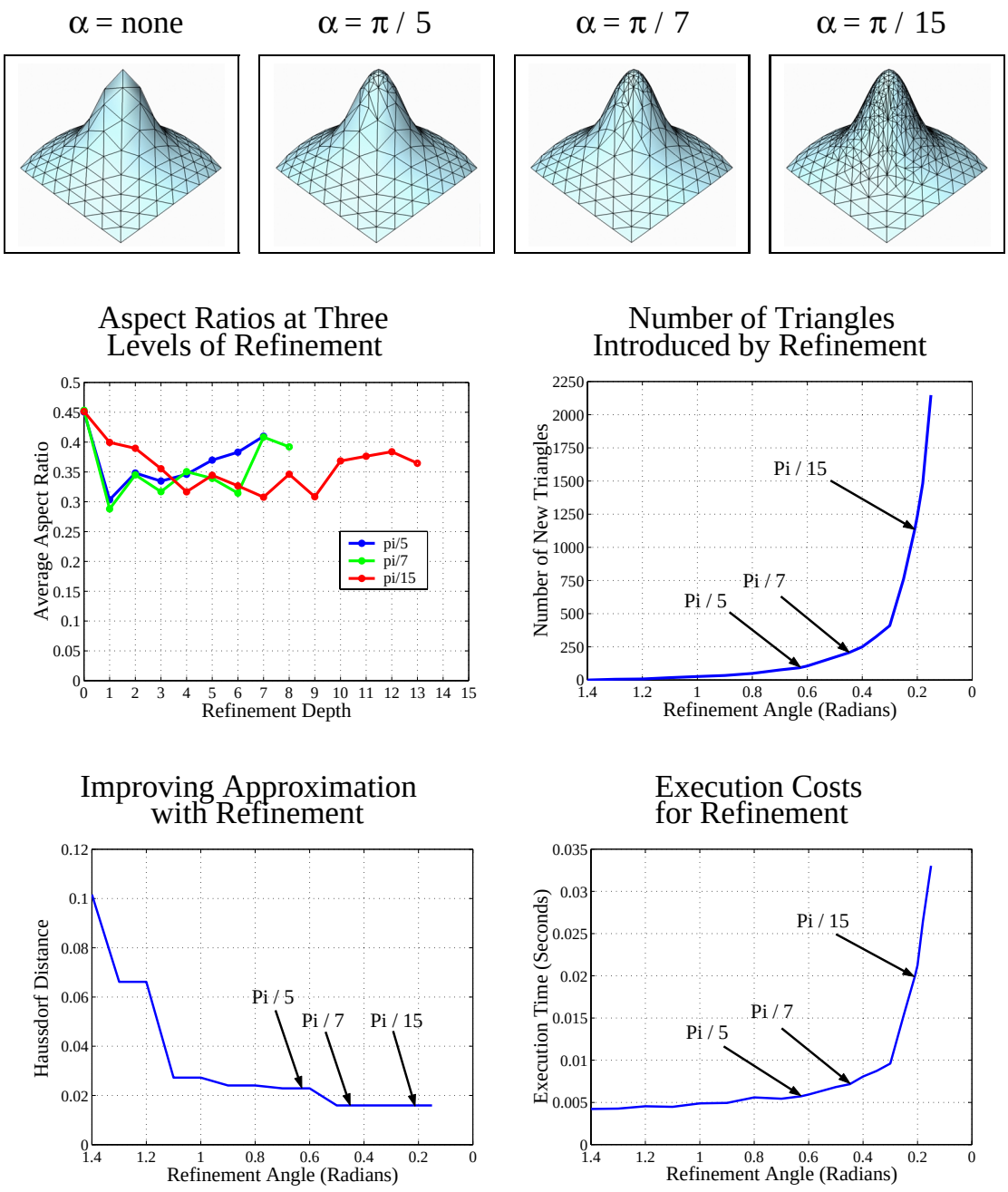


Figure 5.12: **Evaluation of Refinement. [Images] An extruding deformation is refined at four normal deviation angles (α). [Graphs] Behaviour of the average aspect ratio, number of new triangles, approximation quality, and execution cost with respect to refinement at all levels.**

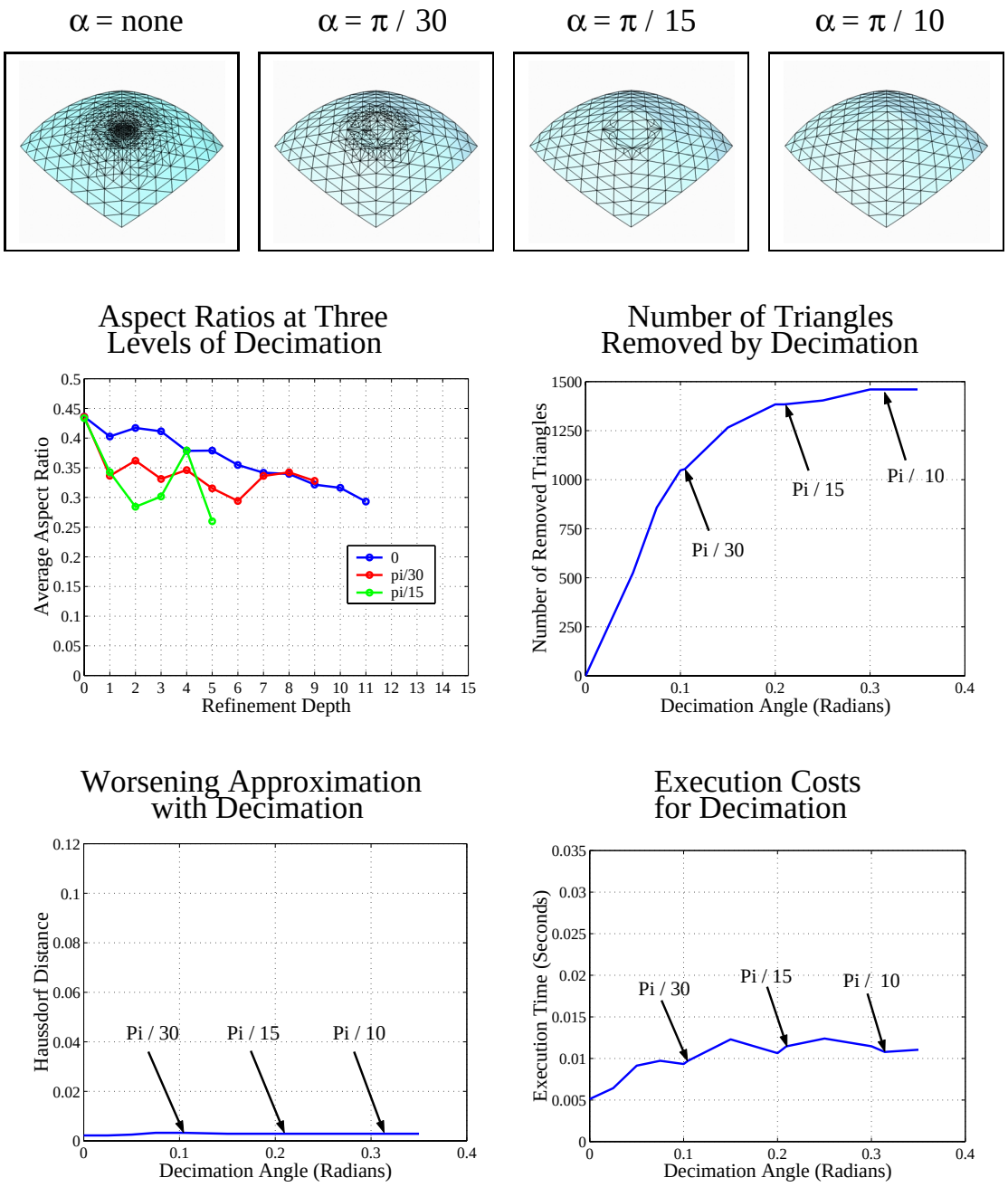


Figure 5.13: **Evaluation of Decimation. [Images] A flattening deformation is decimated at four normal deviation angles (α). [Graphs] Behaviour of the average aspect ratio, number of new triangles, approximation quality, and execution cost with respect to decimation at all levels.**

Chapter 6

Curve Manipulation

6.1 Introduction

Directly Manipulated Free-Form Deformation merges the hyperpatch- and point-based approaches to spatial deformation and allows the user to pick and drag object points directly using an FFD hyperpatch substrate. However, the third and final mode of interaction, curve-based manipulation, is not supported. This is unfortunate, because directing spatial distortion through the motion of curves is a boon to ease of use and versatility. Many designers are comfortable with specifying profile curves and sculptors often employ wire armatures to support and give structure to clay models [Singh and Fiume 1998]. Curve-directed shape design thus has a familiar physical precedent. In addition, curves, as geometric entities, exist at a higher level than points. Their power lies in controlling not only discrete points but also a strand of intermediate positions.

The curve-based category of spatial deformations [Barr 1984; Chang and Rockwood 1994; Lazarus, Coquillart and Jancène 1994; Singh and Fiume 1998] is extensively covered in section 2.2.3. In broad terms these deformations interpolate a number of curve constraints, each with an explicit or implicit source curve, destination curve and enclosing volume of influence. The recent, more general schemes, attach object vertices to the closest reference point on the source curve. A deformation is then imparted by the disparity between the reference point on the source and destination curves and moderated by the zone of influence. While the actual deformation may be computationally inexpensive, the preliminary closest distance parametrisation is costly.

In this chapter, we extend the constraint mechanism of DMFFD to encompass curves 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. Firstly, the source Bézier curve is segmented at cell boundaries in the undistorted hyperpatch. Secondly, the resulting curve fragments are fed to

a composition process, which binds the embedded control points to the hyperpatch through a set of weighting coefficients. Thirdly, since the enormous degree elevation inherent in composition is problematic, the weighting coefficients undergo a series of repeated degree reductions. A complete system of constraints is extracted which DMFFD can process as before. Finally, we compare this composition procedure with a manipulation formed by simply sampling a set of constraint points on the curve.

The theory of trivariate hyperpatch composition and degree reduction is prior art. Our contribution lies in adapting these techniques to DMFFD so that a curve manipulation can be converted into a compact linear constraint system.

Chapter Structure

This material is covered in three sections: background theory on blossoming, composition and degree reduction is reviewed before examining and then evaluating the curve manipulation algorithm in detail.

Our approach has several advantages over previous curve-based spatial deformation techniques:

1. **Combining interaction methods.** Curve-directed DMFFD, as a synthesis of the three strands of spatial deformation, allows the seamless combination of point, derivative-frame and curve constraints within a single deformation.

2. **Efficiency.** By avoiding Euclidean distance parametrisation (used in other curve-based methods), curve-directed DMFFD achieves interactive update rates (≥ 10 frames per second) over the entire deformation cycle, for reasonable numbers of curves (< 10) and moderately sized objects (< 6000 vertices). This makes it highly suitable for interactive sculpting.

3. **Curing self-intersection.** Curve-directed DMFFD is open to the automatic detection and prevention of self-intersection (covered in chapter 7). This does not hold true for other curve-based methods, such as Wires [Singh and Fiume 1998] and Axial Deformation [Lazarus, Coquillart and Jancène 1994], which often introduce a seam of parametrisation discontinuity.

4. **Continuity.** The constrained deformation can maintain any desired degree of derivative continuity simply by raising the degree of the underlying FFD lattice.

6.2 Background

6.2.1 Polar Forms

The Polar Form (or Blossom) of a degree n polynomial function is an equivalent representation with n variables all of single degree. That is, if $\mathcal{F}(u)$ is a polynomial mapping $\mathcal{F}(u) : \Re \mapsto \Re^d$ of degree n, then there exists a unique blossom $\mathcal{F}(u_1, \ldots, u_n) : \Re^n \mapsto \Re^d$. Polar forms are particularly effective at encoding affine (linear) interpolations which underpin the de Casteljau and de Boor algorithms for evaluating Bézier and B-spline curves respectively, and they are consequently achieving widespread popularity. Seidel [1993] and Farin [1997] both give succinct introductions to the application of polar forms in geometric design.

The polar form of a function $\mathcal{F}$ has the following properties [Nimscheck 1995, pp. 8-9]:

1. It produces $\mathcal{F}$ on the diagonal (when all arguments are set equal) $\mathcal{F}(u) = \mathcal{F}(u, \ldots, u)$. Blossom arguments are often repeated and a shorthand notation has been developed

   $$\mathcal{F}(u^{\langle i \rangle}, v^{\langle j \rangle}) \quad = \quad \mathcal{F}(\overbrace{u, \ldots, u}^{i}, \overbrace{v, \ldots, v}^{j})$$

   where the superscripts $\langle i \rangle$ and $\langle j \rangle$ denote the number of repetitions of argument u and v respectively. A similar notation will be adopted to represent repeated knots.

2. It is perfectly symmetric. The arguments can be reshuffled into any order without changing the value. $\mathcal{F}(u_1, \ldots, u_n) = \mathcal{F}(u_{i_1}, \ldots, u_{i_n})$ where $\{i_1, \ldots, i_n\}$ is any permutation of $\{1, \ldots, n\}$.

3. It is multiaffine (affine in each argument with the others held constant). A single variable function $\mathcal{F}(u)$ is considered affine if $\mathcal{F}(\sum_i \alpha_i u_i) = \sum_i \alpha_i \mathcal{F}(u_i)$ where $\sum_i \alpha_i = 1$. In the blossoming context this can be applied independently to any argument

   $$\mathcal{F}(u_1, \ldots, \sum_i \alpha_i u_{j_i}, \ldots, u_n) = \sum_i \alpha_i \cdot \mathcal{F}(u_1, \ldots, u_{j_i}, \ldots, u_n); \quad \sum_i \alpha_i = 1$$

4. For a polynomial in monomial form, $\mathcal{F}(u) = \sum_{i=0}^{n} a_i u^i$, there is a unique blossom equivalent [Seidel 1993]:

   $$\mathcal{F}(u_1, \ldots, u_n) = \sum_{i=0}^{n} a_i \binom{n}{i}^{-1} \sum_{\substack{\mathcal{S} \subseteq \{1, \ldots, n\} \\ |\mathcal{S}| = i}} \prod_{j \in \mathcal{S}} u_j$$

   Example 6.1 *This implies that the cubic polynomial $\mathcal{F}(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$ has the equivalent blossom*

   $$\mathcal{F}(u_1, u_2, u_3) = a_0 + \frac{a_1}{3}(u_1 + u_2 + u_3) + \frac{a_2}{3}(u_1 u_2 + u_2 u_3 + u_3 u_1) + a_3(u_1 u_2 u_3)$$

   It is useful to test the blossoming properties on this example

- $\mathcal{F}(u,u,u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3 = \mathcal{F}(u)$
- $\mathcal{F}(u_1,u_2,u_3) = \mathcal{F}(u_1,u_3,u_2) = \mathcal{F}(u_3,u_1,u_2) = \mathcal{F}(u_3,u_2,u_1) = \mathcal{F}(u_2,u_1,u_3) = \mathcal{F}(u_2,u_3,u_1)$
- *Let* $u_2 = \frac{1}{3}w_1 + \frac{2}{3}w_2$ *then*

$$\begin{aligned}\mathcal{F}(u_1, \frac{1}{3}w_1 + \frac{2}{3}w_2, u_3) &= a_0 + \frac{a_1}{3}(u_1 + (\frac{1}{3}w_1 + \frac{2}{3}w_2) + u_3) + \\ \frac{a_2}{3}(u1(\frac{1}{3}w_1 + \frac{2}{3}w_2) &\quad +(\frac{1}{3}w_1 + \frac{2}{3}w_2)u_3 + u_3u_1) + a_3(u_1(\frac{1}{3}w_1 + \frac{2}{3}w_2)u_3) \\ &= \frac{1}{3}\mathcal{F}(u_1,w_1,u_3) + \frac{2}{3}\mathcal{F}(u_1,w_2,u_3)\end{aligned}$$

5. If three blossoms differ only in a single variable then they have a meaningful geometric relationship: they are collinear. Furthermore, if each blossom represents a point then they are spaced along the straight line in the ratio of the free variables (e.g. $\mathcal{F}(u,v,0), \mathcal{F}(u,v,t)$ and $\mathcal{F}(u,v,1)$ are distributed in the ratio $0:t:1$).

Blossoms can be applied to the representation of B-splines. They are especially effective at highlighting and compactly encoding geometric relationships. A B-spline curve with knot sequence $\{t_0,\ldots,t_{\ell+2n-2}\}$ and control points P_i, can be expressed in the standard form:

$$\mathcal{C}(u) = \sum_{i=0}^{a+\ell-1} \mathcal{N}_i^\ell(u) \cdot P_i$$

The B-spline curve can be evaluated in a numerically stable fashion using de Boor recurrence on the control points [Farin 1997, pp. 147-150]. One alternative is to use an equivalent polar form. The blossom of the control points capture local knot sequence information:

$$P_i = \mathcal{P}(t_i,\ldots,t_{i+\ell-1})$$

Intermediate points in the recursive process of evaluating the curve at a specific parameter value rely on an affine combination of surrounding knot values. If $t_i \le u < t_{i+1}$ then the multiaffine invariance property of blossoms allows the recursion [Nimscheck 1995, p. 21]:

$$\begin{aligned}&\mathcal{P}(\overbrace{t_{i-(\ell-k-j-1)},\ldots,t_i}^{\ell-k-j}, u^{\langle k\rangle}, \overbrace{t_{i+1},\ldots,t_{i+j}}^{j}) = \\ \frac{t_{i+j+1}-u}{t_{i+j+1}-t_{i-(\ell-k-j)}}&\mathcal{P}(\overbrace{t_{i-(\ell-k-j)},\ldots,t_i}^{\ell-k-j+1}, u^{\langle k-1\rangle}, \overbrace{t_{i+1},\ldots,t_{i+j}}^{j}) + \\ \frac{u-t_{i-(\ell-k-j)}}{t_{i+j+1}-t_{i-(\ell-k-j)}}&\mathcal{P}(\overbrace{t_{i-(\ell-k-j-1)},\ldots,t_i}^{\ell-k-j}, u^{\langle k-1\rangle}, \overbrace{t_{i+1},\ldots,t_{i+j+1}}^{j+1})\end{aligned} \tag{6.1}$$

where $\mathcal{P}(u^{\langle\ell\rangle})$ is a point at parameter value u on the curve. This can be validated by noting that the coefficients form a partition of unity:

$$\frac{t_{i+j+1}-u}{t_{i+j+1}-t_{i-(\ell-k-j)}} + \frac{u-t_{i-(\ell-k-j)}}{t_{i+j+1}-t_{i-(\ell-k-j)}} = 1$$

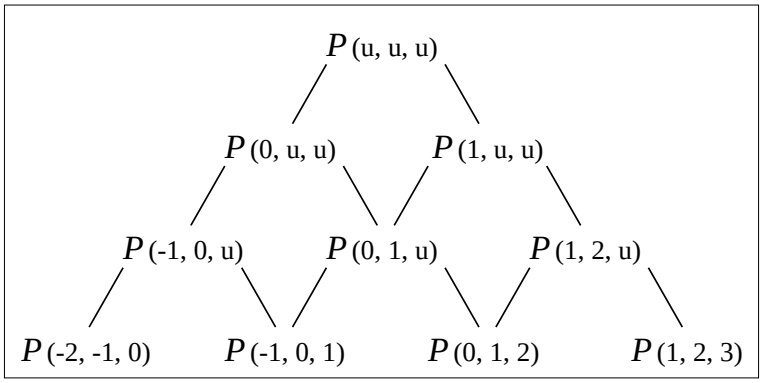


Figure 6.1: **Pyramid of Polar Forms. The recursive generation of intermediate blossoms in the evaluation of a uniform cubic B-spline at u.**

and by employing multiaffine invariance.

The evaluation of a cubic B-spline with a uniform knot vector and single segment, $\{-2,-1,0,1,2,3\}$, at $u = \frac{1}{2}$ is shown schematically in figure 6.1 and geometrically in figure 6.2.

Tensor product B-spline surfaces and hyperpatches have corresponding polar forms which rely on the separability of axes in the tensor product formulation. Thus a B-spline hyperpatch:

$$\mathcal{H}(u,v,w) = \sum_{i=0}^{a+\ell-1} \sum_{j=0}^{b+m-1} \sum_{k=0}^{c+n-1} \mathcal{N}_i^{\ell}(u) \cdot \mathcal{N}_j^{m}(v) \cdot \mathcal{N}_k^{n}(w) \cdot P_{i,j,k}$$

with knot vectors $\{\bar{u}_0,\ldots,\bar{u}_{a+2\ell-2}\}$, $\{\bar{v}_0,\ldots,\bar{v}_{b+2m-2}\}$, $\{\bar{w}_0,\ldots,\bar{w}_{c+2n-2}\}$ has a blossom found by polarising along each axis separately:

$$P_{i,j,k} = \mathcal{P}(\bar{u}_i,\ldots,\bar{u}_{i+\ell-1};\bar{v}_j,\ldots,\bar{v}_{j+m-1};\bar{w}_k,\ldots,\bar{w}_{k+n-1})$$

All the polar form properties hold in this instance: it produces $\mathcal{H}(u,v,w)$ on the tridiagonal, and is symmetric and multiaffine within each axial grouping. Evaluation proceeds by affine interpolation along each axis separately, first collapsing the blossom to an isoparametric surface with $u = const : P(u^{\langle\ell\rangle};\bar{v}_j,\ldots,\bar{v}_{j+m-1};\bar{w}_k,\ldots,\bar{w}_{k+n-1})$, then an isoparametric line $u,v = const : P(u^{\langle\ell\rangle};v^{\langle m\rangle};\bar{w}_k,\ldots,\bar{w}_{k+n-1})$ and finally a point in the hyperpatch at $u,v,w : P(u^{\langle\ell\rangle};v^{\langle m\rangle};w^{\langle n\rangle})$.

Many other geometric operations (subdivision, degree elevation and taking derivatives) can be expressed in polar form and we will be using it extensively for trivariate composition and curve constraint extraction.

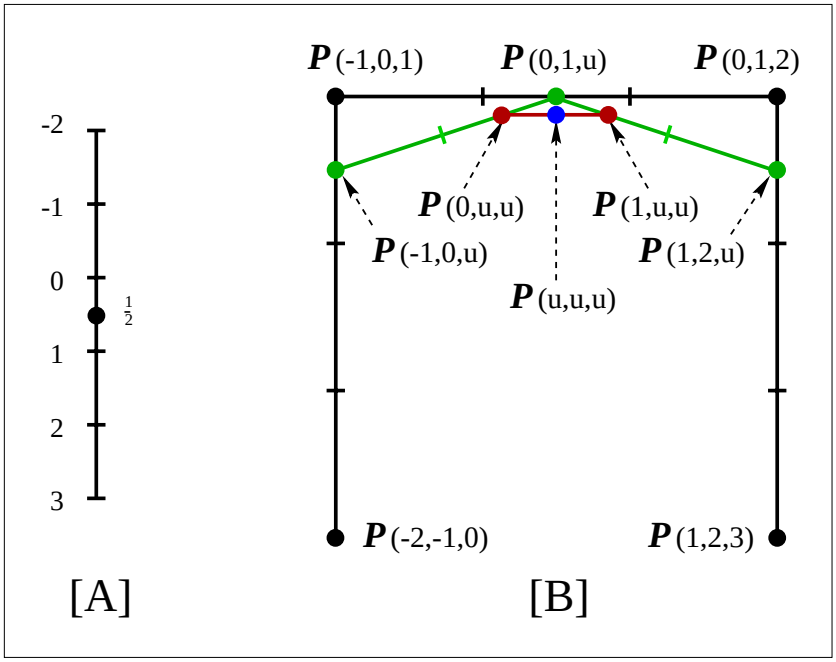


Figure 6.2: **B-Spline Polar Evaluation Using de Boor Recurrence. [A] The B-spline knot sequence and evaluation parameter $u = \frac{1}{2}$. [B] Repeated affine interpolation (each embedding a portion of the knot vector onto an arm of the control polygon) leads to a point $\mathcal{P}(u,u,u)$ at parameter $u = \frac{1}{2}$ on the uniform B-spline curve.**

6.2.2 Univariate Composition

Our development of curve manipulation is founded, at a fundamental level, on univariate composition which expresses an embedded function in terms of an enclosing curve. Later this will be extended to attaching an inset curve to a trivariate hyperpatch. The composition of a B-spline curve, $\mathcal{C}(u) : \Re \mapsto \Re^3$, and a Bézier function, $\mathcal{F}(t) : \Re \mapsto \Re$ results in a Bézier curve taking the form: $\mathcal{F}^\star(t) = \mathcal{C} \circ \mathcal{F} = \mathcal{C}(\mathcal{F}(t)) : \Re \mapsto \Re^3$. Conceptually the composition process embeds $\mathcal{F}$ within $\mathcal{C}$ to supply a Bézier curve, $\mathcal{F}^\star$, coincident with $\mathcal{C}$ but generally of different degree (combining the degrees of $\mathcal{F}$ and $\mathcal{C}$), polynomial basis (migrating from B-spline to Bézier), range ($t \subset u$) and parametrisation (speed along the curve).

There are three paths to solving this composition problem, all of which rely on subdividing the Bézier function, $\mathcal{F}$, at t-parameter values which correspond to joins between segments of the B-spline curve, $\mathcal{C}$. A particular subdivided function, $\mathcal{F}_s$, can be composed with $\mathcal{C}$ in one of three ways:

- Both $\mathcal{C}$ and $\mathcal{F}_s$ are converted to monomial (power) form. Once the composition, which is now relatively straightforward [Nimscheck 1995, pp. 39-41], has been constructed, the result is returned to a Bézier format. Bézier [1978] employed this strategy in composing

curves with two-dimensional FFD patches. Unfortunately, as demonstrated by Farouki [1991] such conversions are numerically unstable.

- The conventional technique [Nimscheck 1995, p. 58] is to segment the B-spline curve, $\mathcal{C}$, into multiple Bézier curves via repeated knot insertion and then apply composition entirely within a Bézier context [DeRose 1988]. This has the overhead of B-spline to Bézier conversion.
- As noted by DeRose, Goldman, Hagen and Mann [1993], since a B-spline polar form exists, it is possible to directly embed a Bézier function into a B-spline curve. This is the approach that we adopt.

The univariate composition process is provided with a B-spline curve, $\mathcal{C}(u)$, in polar form on the knot sequence $\{\bar{u}_0, \ldots, \bar{u}_{a+2\ell-2}\}, \bar{u}_i \in \Re, \bar{u}_i \leq \bar{u}_{i+1}$

$$\mathcal{C}(u) = \sum_{i=0}^{a+\ell-1} \mathcal{N}_i^\ell(u) \cdot \mathcal{P}(\bar{u}_i, \ldots, \bar{u}_{i+\ell-1})$$

and a Bézier function, $\mathcal{F}_s(t)$, which is local to a single segment of $\mathcal{C}$ ($\bar{u}_i \leq t < \bar{u}_{i+1}$) and has scalar coefficients

$$\mathcal{F}_s(t) = \sum_{j=0}^{p} \mathcal{B}_j^p(t) \cdot a_j$$

The composed Bézier curve, $\mathcal{F}_s^\star$, then has a degree, $p\ell$, which is the product of its component degrees, and control points, $P^\star$, that are, as derived by DeRose *et al.* [1993], convex combinations of the B-spline blossoms, $\mathcal{P}$, evaluated at the Bézier coefficients, a.

$$\begin{aligned}
\mathcal{F}_s^\star(t) &= \sum_{k=0}^{p\ell} \mathcal{B}_k^{p\ell}(t) \cdot P_k^\star \\
P_k^\star &= \sum_{\forall (i)_\ell \in {}^\ell\mathcal{S}_k^p} \mathcal{Y}_k^p((i)_\ell) \cdot \mathcal{P}(a_{i_1}, \ldots, a_{i_\ell}) && (6.2) \\
{}^\ell\mathcal{S}_k^p &= \{(i)_\ell : i_1, \ldots, i_\ell \in \{0, \ldots, p\}, i_1 + \cdots + i_\ell = k\} && (6.3) \\
\mathcal{Y}_k^p((i)_\ell) &= \mathcal{Y}_k^p(i_1, \ldots, i_\ell) = \frac{\binom{p}{i_1} \cdots \binom{p}{i_\ell}}{\binom{p\ell}{k}} && (6.4)
\end{aligned}$$

Equation 6.2 is heavily reliant on multi-indices, $(i)_\ell \in {}^\ell\mathcal{S}_k^p$. These are ℓ-arrangements in which order is important and repetition is allowed, and whose elements fall in the range $\{0, \ldots, p\}$ and together sum to k. DeRose [1988] proves the veracity of this formula through induction using the multivariate product (which is where the combinatorial constant $\mathcal{Y}_k^p$ is introduced) and recursive definition of Bernstein polynomials. It is worth noting that both the blossom, $\mathcal{P}$, and combinatorial constant, $\mathcal{Y}$, are symmetric so that only a single permutation of the multi-index need actually be instantiated.

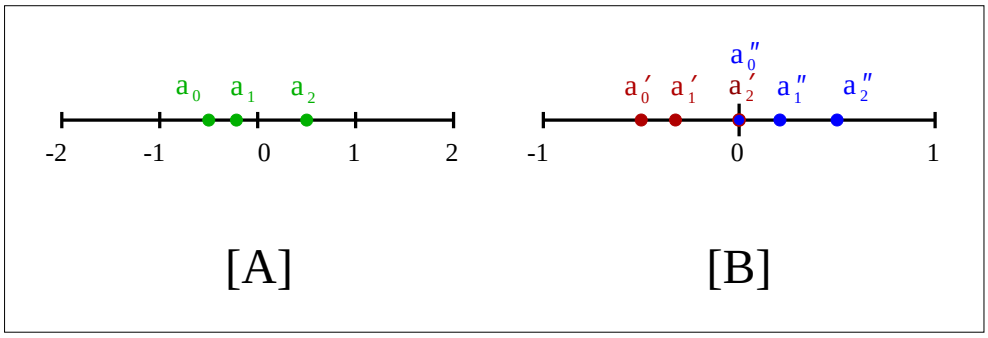


Figure 6.3: **Bézier Subdivision. A Bézier curve with control points, a_i, is subdivided at the join between B-spline segments into two curves whose control points are a'_i and a''_i.**

Certain operations can be regarded as special cases of composition. For instance, composition of a B-spline curve with a constant function is equivalent to de Boor evaluation. In fact, in a sense composition is a generalisation of de Boor recurrence.

Example 6.2 (Composition of a quadratic B-spline curve and a constant Bézier function) *If a B-spline curve, $\mathcal{C} = \sum_{i=0}^{2} \mathcal{N}_i^2 \cdot \mathcal{P}(\bar{u}_i, \bar{u}_{i+1})$ and constant function $\mathcal{F}(t) = a_0$ are composed, then the result is effectively $\mathcal{C}(a_0)$:*

$$\mathcal{F}^\star(t) = \sum_{k=0}^{0} \mathcal{B}_k^0(t) \sum_{(i)_\ell=(0,0)} \mathcal{Y}_0^0((0,0)) \cdot \mathcal{P}(a_0, a_0) = \mathcal{P}(a_0, a_0) = \mathcal{C}(a_0)$$

Example 6.3 (A quadratic B-spline curve composed with a quadratic Bézier function) *A more complex example of univariate composition will help foster geometric insight. Given a uniform quadratic B-spline curve with two segments ($\ell = 2, a = 2$), which is defined over the knot vector $\{-2, -1, 0, 1, 2\}$, and whose first segment control polygon appears in figure 6.4; and also given a quadratic Bézier function ($p = 2$) with control points $\{a_0 = -0.5, a_1 = -0.25, a_2 = 0.5\}$, then the composition is formed in the following stages:*

1. *The Bézier function, $\mathcal{F}$, spans a segment boundary of the B-spline curve, $\mathcal{C}$, at $u = 0$. To ensure that the embedded function lies within a segment, the equation $\mathcal{F}(t) = 0$ is solved to obtain $t_s = 0.618$, the t-parameter value of $\mathcal{F}$ at the u-segment boundary. $\mathcal{F}$ is then split into $\mathcal{F}'$ and $\mathcal{F}''$ at t_s through de Casteljau subdivision [Farin 1997, pp. 53-56]. The control points of the subdivided functions are $\{a'_0 = -0.5, a'_1 = -0.3455, a'_2 = 0.0\}$ and $\{a''_0 = 0.0, a''_1 = 0.2135, a''_2 = 0.5\}$ as shown in figure 6.3.*

2. *The polar forms $\mathcal{P}(a'_{i_1}, a'_{i_2})$, $\mathcal{P}(a''_{i_1}, a''_{i_2})$; $i_1, i_2 \in \{0, 1, 2\}$ are generated via recursive affine interpolation of the B-spline control points. By exploiting symmetry, only a subset of blossoms need be evaluated. In composing $\mathcal{F}'$, the control points $\mathcal{P}(-2, -1)$, $\mathcal{P}(-1, 0)$ and $\mathcal{P}(0, 1)$ are a starting point from which the following distinct blossoms are derived (as in figure 6.4):*

$$\mathcal{P}(a'_0, a'_0), \mathcal{P}(a'_0, a'_1), \mathcal{P}(a'_0, a'_2), \mathcal{P}(a'_1, a'_1), \mathcal{P}(a'_1, a'_2), \mathcal{P}(a'_2, a'_2)$$

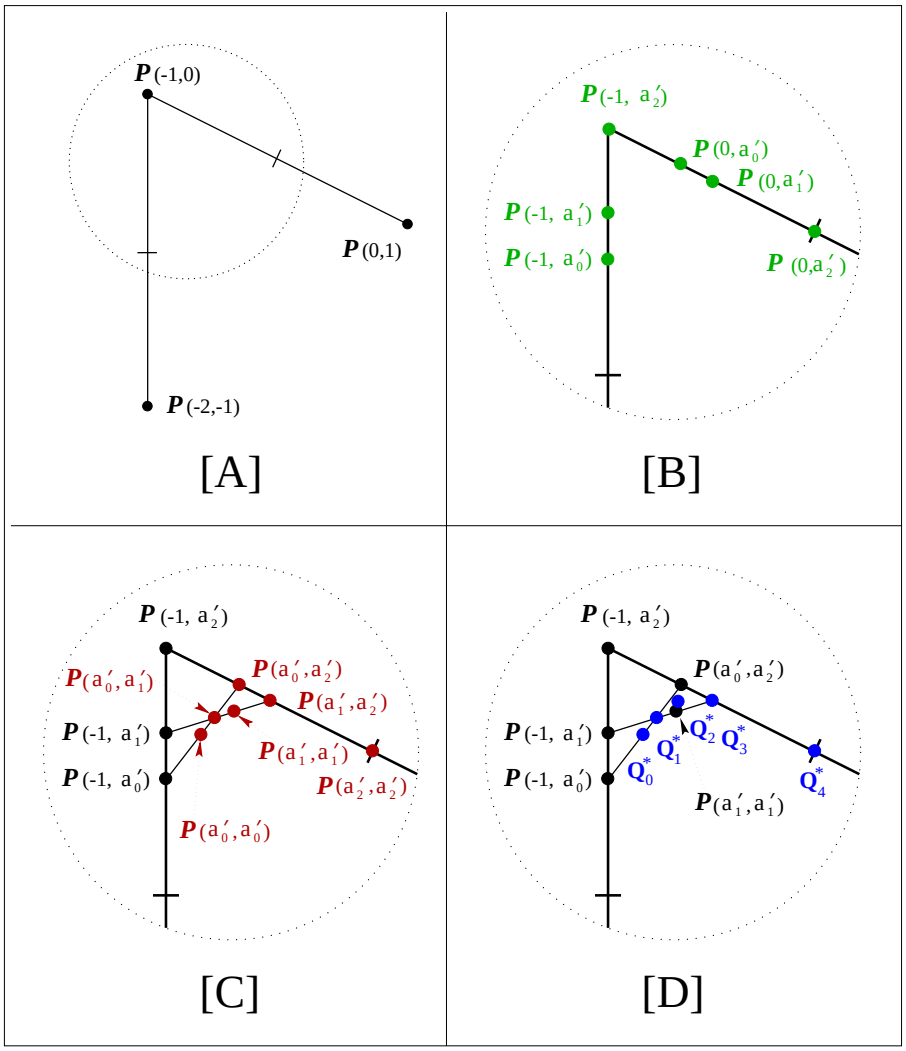


Figure 6.4: **Univariate Composition. Embedding a quadratic Bézier function into a quadratic B-spline curve. [A] The B-spline control polygon. [B] First level, introducing a'_0, a'_1, a'_2 into the polar form. [C] Second level of polar evaluation. [D] Deriving the composed control points, $Q^\star_i$, as convex combinations of the final polar forms.**

Obtaining the blossoms of $\mathcal{F}''$ follows a similar path, with a''_j in place of a'_j and beginning from $\mathcal{P}(-1,0)$, $\mathcal{P}(0,1)$ and $\mathcal{P}(1,2)$. Notice how a bi-symmetric polar form such as $\mathcal{P}(a'_0,a'_1) = \mathcal{P}(a'_1,a'_0)$ can be reached in either of two ways:

$$\begin{array}{ccc} & \mathcal{P}(a'_1,a'_0) & \\ \mathcal{P}(-1,a'_0) & & \mathcal{P}(0,a'_0) \\ \mathcal{P}(-2,-1) & \mathcal{P}(-1,0) & \mathcal{P}(0,1) \end{array} \qquad \textit{or} \qquad \begin{array}{ccc} & \mathcal{P}(a'_0,a'_1) & \\ \mathcal{P}(-1,a'_1) & & \mathcal{P}(0,a'_1) \\ \mathcal{P}(-2,-1) & \mathcal{P}(-1,0) & \mathcal{P}(0,1) \end{array}$$

The number of symmetric forms matches the number of different derivation pyramids.

3. *The composed control points are convex combinations of the Bézier coefficient-inscribed blossoms, as stipulated by eqn. 6.2. Thus $\mathcal{F}'^{\star} = \mathcal{C} \circ \mathcal{F}'$ is a degree $\ell \cdot p = 2 \cdot 2 = 4$ Bézier curve with control points $Q^{\star}$:*

$$\begin{aligned}
Q_0^{\star} &= 1 \cdot \mathcal{P}(a'_0, a'_0) = \mathcal{P}(a'_0, a'_0) \\
Q_1^{\star} &= 2 \cdot \frac{1}{2} \cdot \mathcal{P}(a'_0, a'_1) = \mathcal{P}(a'_0, a'_1) \\
Q_2^{\star} &= 2 \cdot \frac{1}{6} \cdot \mathcal{P}(a'_0, a'_2) + \frac{2}{3} \cdot \mathcal{P}(a'_1, a'_1) = \frac{1}{3} \cdot \mathcal{P}(a'_0, a'_2) + \frac{2}{3} \cdot \mathcal{P}(a'_1, a'_1) \\
Q_3^{\star} &= 2 \cdot \frac{1}{2} \mathcal{P}(a'_1, a'_2) = \mathcal{P}(a'_1, a'_2) \\
Q_4^{\star} &= 1 \cdot \mathcal{P}(a'_2, a'_2) = \mathcal{P}(a'_2, a'_2)
\end{aligned}$$

6.2.3 Trivariate Composition

The principles of univariate composition (embedding a Bézier function into a B-spline curve) extend smoothly to trivariate composition (embedding a Bézier curve into a B-spline hyperpatch). An FFD hyperpatch on the domain u, v, w with corresponding degrees ℓ, m, n; knot sums a, b, c; and knot vectors $\{\bar{u}_0, \ldots, \bar{u}_{a+2\ell-2}\}$, $\{\bar{v}_0, \ldots, \bar{v}_{b+2m-2}\}$, $\{\bar{w}_0, \ldots, \bar{w}_{c+2n-2}\}$, has the polar form:

$$\begin{aligned}
\mathcal{H}(u, v, w) = \sum_{i=0}^{a+\ell-1} \sum_{j=0}^{b+m-1} \sum_{k=0}^{c+n-1} \quad & \mathcal{N}_i^{\ell}(u) \cdot \mathcal{N}_j^{m}(v) \cdot \mathcal{N}_k^{n}(w) \cdot \\
& \mathcal{P}(\bar{u}_i, \ldots, \bar{u}_{i+\ell-1}; \bar{v}_j, \ldots, \bar{v}_{j+m-1}; \bar{w}_k, \ldots, \bar{w}_{k+n-1})
\end{aligned}$$

The Bézier curve due for embedding has degree p and is represented by:

$$\mathcal{G}(t) = \sum_{r=0}^{p} \mathcal{B}_r^p(t) \cdot Q_r$$

The trivariately composed curve $\mathcal{G}^{\star}(t) = \mathcal{H} \circ \mathcal{G}$ is of elevated degree $p(\ell + m + n)$.

$$\mathcal{G}^{\star}(t) = \sum_{d=0}^{p(\ell+m+n)} \mathcal{B}_d^{p(\ell+m+n)}(t) \cdot Q_d^{\star}$$

The composed control points $Q_d^{\star}$ can be constructed from repeated univariate composition (eqn. 6.2) and the multiplication of Bernstein bases (eqn. 7.8) [Nimscheck 1995, pp. 48-51]:

$$\begin{aligned}
Q_d^{\star} = \sum_{e+f+g=d} & \sum_{\forall (i)_\ell \in {}^{\ell}\mathcal{S}_e^p} \sum_{\forall (j)_m \in {}^{m}\mathcal{S}_f^p} \sum_{\forall (k)_n \in {}^{n}\mathcal{S}_g^p} \\
& \Big[\mathcal{X}(e, f, g) \cdot \mathcal{Y}_e^p((i)_\ell) \cdot \mathcal{Y}_f^p((j)_m) \cdot \mathcal{Y}_g^p((k)_n) \cdot \\
& \mathcal{P}([Q_{i_1}]_u, \ldots, [Q_{i_\ell}]_u; [Q_{j_1}]_v, \ldots, [Q_{j_m}]_v; [Q_{k_1}]_w, \ldots, [Q_{k_n}]_w) \Big] \qquad (6.5)
\end{aligned}$$

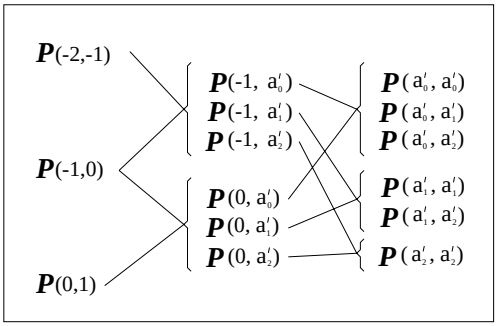


Figure 6.5: **Re-using Partial Polar Evaluations. A minimal evaluation scheme for example 6.3.**

$$\mathcal{X}(e,f,g) \quad = \quad \frac{\binom{p\ell}{e}\binom{pm}{f}\binom{pn}{g}}{\binom{p(\ell+m+n)}{e+f+g}} \tag{6.6}$$

This consists of successively "engraving" the u, v, w components of the Bézier control points $Q_r = \left([Q_r]_u, [Q_r]_v, [Q_r]_w\right)$ onto the $\vec{U}, \vec{V}, \vec{W}$-oriented edges of the lattice and forming convex combinations of the results. The cost of evaluating eqn. 6.5 is unnecessarily high and can be reduced in several ways.

1. As in univariate composition, the polar form and all of the combinatorial constants are symmetric (the arguments can be reordered without affecting the value) and only one permutation need be evaluated. By enforcing a non-decreasing ordering on multi-index elements, re-evaluations on permutations of the same multi-index are avoided. The set of unique multi-indices, $\mathcal{I}$, replaces the set of permutable multi-indices, $\mathcal{S}$, in eqn. 6.5:

$$\begin{aligned} {}^{\ell}\mathcal{I}_e^p \quad = \quad & \Big\{(i)_\ell : i_1, \ldots, i_\ell = 0^{\langle r_0 \rangle}, 1^{\langle r_1 \rangle}, \ldots, p^{\langle r_p \rangle}; \\ & 0r_0 + 1r_1 + 2r_2 + \cdots + pr_p = e\Big\} \end{aligned} \tag{6.7}$$

   For example, the elements of ${}^{3}\mathcal{I}_4^3$ in lexicographic order are:

$$\{(0,1,3),(0,2,2),(1,1,2)\}$$

   The number of permutations of a particular multi-index must also be accounted for. Combinatorial mathematics [Grimaldi 1989, p. 6] supplies a count of the permutations of a multi-index:

$$\mathcal{Z}_e((i)_\ell) = \frac{e!}{r_0!r_1!\cdots r_p!} \tag{6.8}$$

   The elements of the set ${}^{\ell}\mathcal{I}_e^p$ can be generated in increasing lexicographic order using an iterative algorithm, which reduces elements on the right and distributes the reduction leftwards while ensuring a non-decreasing ordering.

```
NextMultiIndex(ℓ, p, i_1,...,i_ℓ)
(Given a multi-index {i_1,...,i_ℓ} which is an element of ℓI_e^p,
 provide, if possible, the next element in lexicographic
 order.  Return FALSE if no further multi-indices remain,
 otherwise TRUE.)

 fin ← FALSE; notlast ← TRUE
 decr ← ℓ; incr ← ℓ − 1
 WHILE NOT fin DO
   IF i_decr − i_incr > 1 THEN
     i_decr ← i_decr − 1; i_incr ← i_incr + 1
     fin ← TRUE
   ELSE
     IF i_decr − i_incr = 1 THEN
       incr ← incr − 1
     ELSE (i_decr − i_incr = 0)
       incr ← incr − 1; decr ← decr − 1
     IF incr = 0 THEN
       fin ← TRUE; notlast ← FALSE
 RETURN notlast
```

DeRose *et al.* [1993] advocate a similar implementation strategy.

2. If the hyperpatch is defined over a uniform knot vector then, for a particular setting of the embedding curve and hyperpatch degrees, the combinatorial constants, $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$, are amenable to precalculation and premultiplication. Only the polar evaluation and combination, which depends on particular control point configurations, is not open to preprocessing.

3. As DeRose *et al.* [1993] point out, it is vital to reuse partial polar evaluations. For instance, the polar forms $\mathcal{P}(a'_0, a'_0)$, $\mathcal{P}(a'_0, a'_1)$ and $\mathcal{P}(a'_0, a'_2)$ in example 6.3 are affine interpolations of $\mathcal{P}(-1, a'_0)$ and $\mathcal{P}(0, a'_0)$ which should only be calculated once. With this in mind, a minimal evaluation scheme for the polar forms in example 6.3 would be that shown in figure 6.5.

Equation 6.5 can be replaced by an updated version of trivariate composition which takes these improvements into consideration:

$$
\begin{aligned}
Q^\star_d \;=\; & \sum_{e+f+g=d} \;\; \sum_{\forall (i)_\ell \,\in\, {}^\ell\mathcal{I}^p_e} \;\; \sum_{\forall (j)_m \,\in\, {}^m\mathcal{I}^p_f} \;\; \sum_{\forall (k)_n \,\in\, {}^n\mathcal{I}^p_g} \\
& \left[\mathcal{X}(e,f,g)\cdot \mathcal{Y}_e((i)_\ell)\cdot \mathcal{Y}_f((j)_m)\cdot \mathcal{Y}_g((k)_n)\cdot \mathcal{Z}_e((i)_\ell)\cdot \mathcal{Z}_f((j)_m)\cdot \mathcal{Z}_g((k)_n)\cdot \right. \\
& \left. \mathcal{P}([Q_{i_1}]_u, \ldots, [Q_{i_\ell}]_u ; [Q_{j_1}]_v, \ldots, [Q_{j_m}]_v ; [Q_{k_1}]_w, \ldots, [Q_{k_n}]_w)\right] \qquad (6.9)
\end{aligned}
$$

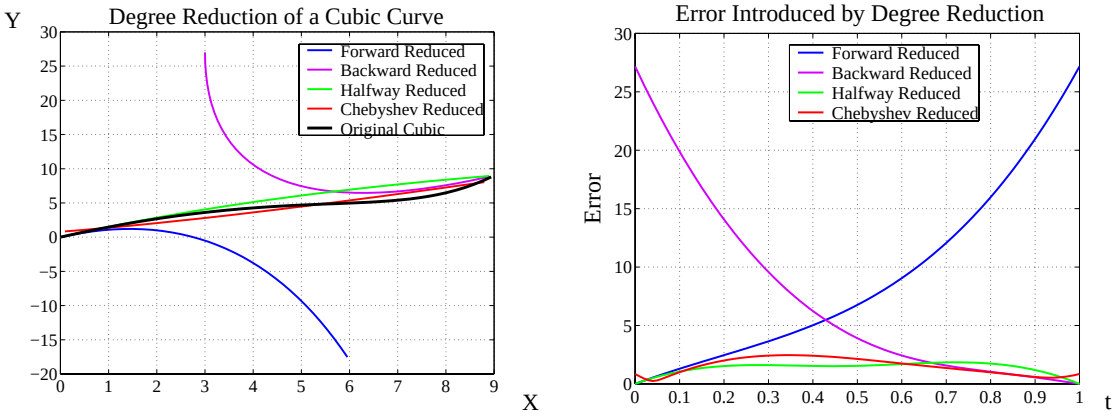


Figure 6.6: **Effect of Different Varieties of Degree Reduction. [Left] A cubic curve with inflection point (black) is reduced to quadratic degree using the forward, backward, halfway and Chebyshev interpolants. [Right] The error between the initial cubic and the different degree reductions is graphed.**

6.2.4 Degree Reduction

The act of embedding a curve within a hyperpatch through trivariate composition is innately and massively degree elevating ($p \mapsto p(\ell + m + n)$). In a curve manipulation setting this gives rise to ill-conditioning and an increased computation burden. Fortunately, repeated degree reduction can be administered to approximate a curve of high degree with one of substantially lower degree.

It is helpful to regard degree reduction as reversing a previous degree elevation. A Bézier curve with raised degree, but the same overall shape, can be obtained by piecewise linear interpolation of the original control points, $\widehat{Q}_i$, at parameter values, $i/(\widehat{n}+1)$. The control points, Q_i, of the $(n = \widehat{n}+1)$ degree Bézier curve are thus related to the originating $\widehat{n}$-degree curve by [Farin 1997, p. 58]:

$$Q_i = \frac{i}{n}\widehat{Q}_{i-1} + \frac{n-i}{n}\widehat{Q}_i \qquad i = 0, 1, \ldots, n \tag{6.10}$$

Typically the degree elevation formula is wielded to gain flexibility without discarding a previous design. A modeller might build a parabola with roughly the desired shape but then realize that an inflection point was required, necessitating a shift from quadratic to cubic degree. According to Farin [1997, p. 65] degree elevation also has its uses in data transfer between different modelling systems with different degree requirements.

Working from the assumption that the control points, Q_i, of a given curve are generated by degree elevation, it is possible to obtain from eqn. 6.10 both forward and backward recursive

expressions for the degree reduced control points:

$$\overrightarrow{Q}_i = \frac{nQ_i - i\,\overrightarrow{Q}_{i-1}}{n-i} \qquad i = 0, 1, \ldots, n-1 \tag{6.11}$$

$$\overleftarrow{Q}_{i-1} = \frac{nQ_i - (n-i)\,\overleftarrow{Q}_i}{i} \qquad i = n, n-1, \ldots, 1 \tag{6.12}$$

There are advantages in linearly combining these two formulae into a single control point approximation:

$$\widehat{Q}_i = (1-\lambda_i)\,\overrightarrow{Q}_i + \lambda_i\,\overleftarrow{Q}_i \qquad i = 0, 1, \ldots, n-1 \tag{6.13}$$

Farin [1997, pp. 67-71] mentions four different settings for the interpolation parameter, λ_i:

$$\begin{aligned}
\text{Forward:} \quad & \lambda_i = 0 \\
\text{Backward:} \quad & \lambda_i = 1 \\
\text{Halfway:} \quad & \lambda_i = \begin{cases} 0 & i \le \frac{n}{2} \\ 1 & i > \frac{n}{2} \end{cases} \\
\text{Chebyshev:} \quad & \lambda_i = \frac{1}{2^{2n-1}} \sum_{j=0}^{i} \binom{2n}{2j}
\end{aligned} \tag{6.14}$$

If the working assumption that the Q_i are arrived at by degree elevation holds true, then all four λ_i settings achieve an identical and error-free degree reduction. In general this is not the case and a certain amount of error intrudes. For example, a cubic with an inflection point can only be approximated by a quadratic. The approximation quality of the degree reduction depends on the choice of interpolation parameter, λ_i.

The forward ($\lambda_i = 0$) and backward ($\lambda_i = 1$) interpolations reduce to equations 6.11 and 6.12 respectively. These formulae are unstable extrapolations, which rapidly accumulate catastrophic error as they progress. As illustrated by figure 6.6, forward and backward reduction most closely approximate the initial curve near the beginning of their cycles, towards, respectively, the left and right endpoints, where little or no error is contained in the $\overrightarrow{Q}_{i-1}$ and $\overleftarrow{Q}_i$ terms. This suggests that favouring forward reduction on the left-hand side and backward reduction towards the right, would be advantageous. Indeed, the halfway step function interpolation behaves well near both endpoints, although less decently between. Fortunately, an optimal least-error degree reduction can be obtained from approximation theory [Press *et al.* 1992, pp. 190-194]. The initial curve can be expressed on a Chebyshev rather than Bernstein basis:

$$\mathcal{C}_n(t) = \sum_{i=0}^{n} \mathcal{B}_i^n(t) \cdot Q_i \equiv \sum_{i=0}^{n} \mathcal{T}_i^n(t) \cdot R_i$$

By removing the final term, $\mathcal{T}_n^n(t) \cdot R_n$, from the polynomial summation, a curve, $\mathcal{C}_{n-1}(t)$, of reduced degree, is obtained. The procedure, termed Chebyshev economisation, is guaranteed to introduce the least possible deviation between $C_n(t)$ and $C_{n-1}(t)$. Rather than circuitously

converting between Bernstein and Chebyshev bases, Eck [1993] stipulates the Chebyshev interpolation parameter that achieves the same end. The only (slight) drawback is that, as can be seen in figure 6.6, the endpoints of the original are not interpolated by its Chebyshev reduced counterpart. However, this side-effect is not a serious issue in the curve manipulation context.

6.3 Curve Manipulation Algorithm

With supporting material on polar forms, composition and degree reduction in place, the time has come to address the specific task of embedding a degree p Bézier curve, $\mathcal{G}(t)$, into a B-spline hyperpatch, $\mathcal{H}(u, v, w)$, of degree $\ell \times m \times n$, so that changes in the curve's control points ($\mathbf{Q} \mapsto \mathbf{Q} + \Delta\mathbf{Q}$) will induce a matching distortion of the lattice ($\mathbf{P} \mapsto \mathbf{P} + \Delta\mathbf{P}$). From each curve, with $p + 1$ control point constraints ($Q_r, \Delta Q_r, r = 0, \ldots, p$), supplied by the user, a system of linear equations is derived that relates weighting coefficients, $\mathbf{W}$, and unknown lattice control point changes, $\Delta\mathbf{P}$, to the movement of extracted curve constraints, $\Delta\mathbf{Q}^\star$:

$$[W_i]\,\Delta\mathbf{P} = [\Delta Q_i^\star] \Rightarrow \mathbf{W}\Delta\mathbf{P} = \Delta\mathbf{Q}^\star \tag{6.15}$$

This linear system can be extracted from the source curve, $\mathcal{G}(t) : Q_r$, and destination curve, $\widetilde{\mathcal{G}}(t) : Q_r + \Delta Q_r$, in four phases. Firstly, as composition can only be applied within a single hyperpatch cell, the source curve is segmented at cell boundaries. Secondly, the resulting curve segments, $\mathcal{G}_s(t) : Q_{s,r}$, are trivariately composed with the hyperpatch in a process which binds the control points, $Q_{s,r}$, to the lattice through a set of weighting coefficients, $\mathbf{W}_s$. Thirdly, the weighting coefficients undergo a sequence of degree reductions so as to compensate for the difficulties with the huge degree elevation implicit in composition. Finally, the displacement constraints, $\Delta\mathbf{Q}^\star$ are elicited from $\mathcal{G}_s(t)$ and $\widetilde{\mathcal{G}}_s(t)$.

The final number of constraints (rows in eqn. 6.15) is difficult to predict. It is increased by segmentation and composition from $p+1$ (the number of curve control points) to $(q+1)\cdot(p(\ell+m+n)+1)$ (with q being the segmentation count) and decreased to some extent by degree reduction. This behaviour will be analysed in depth in section 6.3.3.

A particular curve manipulation need not stand alone. It can be bundled alongside other point, derivative-frame and curve manipulations into a complete constraint system, which can be solved and then imposed on an object through conventional DMFFD techniques (as specified in chapter 3). The only proviso (considered later) is that care must be taken to avoid redundancies and inconsistencies. We now consider the separate stages of curve manipulation in further detail.

6.3.1 Cell-Wise Segmentation

It is necessary to subdivide the Bézier source curve, $\mathcal{G}(t)$, at cell boundaries in the pre-deformation hyperpatch, $\mathcal{H}(u, v, w)$, because, as mentioned in section 6.2.2, composition can only be carried out locally within a single cell. In conventional Directly Manipulated

FFD the pre-deformation lattice is arranged in a parallelepiped layout across an orthogonal co-ordinate frame $(O, \vec{U}, \vec{V}, \vec{W})$. Locating cell boundary intersections reduces, in this instance, to locating all t-parameter values, $\{t_u\}$, $\{t_v\}$, $\{t_w\}$, where the component Bézier functions, $\mathcal{G}_u(t), \mathcal{G}_v(t), \mathcal{G}_w(t)$, coincide with the domain knots, $\{\bar{u}_{\ell-1}, \ldots, \bar{u}_{a+\ell-1}\}$, $\{\bar{v}_{m-1}, \ldots, \bar{v}_{b+m-1}\}$, $\{\bar{w}_{n-1}, \ldots, \bar{w}_{c+n-1}\}$ along the $\vec{U}$, $\vec{V}$, $\vec{W}$ axes. Individually, this is a test of whether (and at what parameter values) a given Bézier function spans certain points on the real line. Once complete, the individual intersection lists $\{t_u\}$, $\{t_v\}$, $\{t_w\}$ are combined into a sequence $\{t_s\}$ sorted in ascending order. Then the de Casteljau subdivision algorithm [Farin 1997, pp. 53-56] is employed to successively segment the source curve, $\mathcal{G}(t)$, at the junctures where the curve intersects a cell-wall, $\{t_s\}$, thereby producing curve fragments, $\mathcal{G}'_s(t)$, which are each contained within a cell. The same intersection values $\{t_s\}$ are also used to perform a segmentation of $\widetilde{\mathcal{G}}(t)$ into $\widetilde{\mathcal{G}}_s(t)$.

If the source curve is of sufficiently low degree (linear, quadratic or cubic) then the knot intersections along a given axis, say $\vec{U}$, can be determined analytically as the roots of a simple algebraic equation [Press *et al.* 1992, pp. 183-185]. A particular domain knot, $\bar{u}$, can be discarded from consideration if it lies outside the range of the curve coefficients, $\bar{u} < \min([Q_r]_u)$ or $\bar{u} > \max([Q_r]_u)$. No intersection is possible unless the knot falls within the convex hull of the curve control points. The domain knots that remain are each in turn checked algebraically for their intersection with the component Bézier function.

Linear Equation :
$$(\sum_{r=0}^{1} \mathcal{B}_r^1(t) \cdot Q_{r,u}) - \bar{u} = 0$$

Algebraic Solution :
$$t_0 = \frac{\bar{u} - Q_{0,u}}{Q_{1,u} - Q_{0,u}}$$

Quadratic Equation :
$$(\sum_{r=0}^{2} \mathcal{B}_r^2(t) \cdot Q_{r,u}) - \bar{u} = 0$$

Algebraic Solution :
$$a = Q_{0,u} - 2Q_{1,u} + Q_{2,u}$$
$$q = Q_{0,u} - Q_{1,u} - \mathrm{sgn}(Q_{1,u} - Q_{0,u}) \cdot \sqrt{Q_{1,u}^2 + Q_{0,u}(Q_{1,u} - Q_{2,u}) + \bar{u}a}$$
$$t_0 = \frac{q}{a}$$
$$t_1 = \frac{Q_{0,u} - \bar{u}}{q}$$

Cubic Equation :
$$(\sum_{r=0}^{3} \mathcal{B}_r^3(t) \cdot Q_{r,u}) - \bar{u} = 0$$

Algebraic Solution :
$$a = Q_{3,u} - 3Q_{2,u} + 3Q_{1,u} - Q_{0,u}$$
$$b = 3Q_{2,u} - 6Q_{1,u} + 3Q_{0,u}$$
$$c = 3Q_{1,u} - 3Q_{0,u}, d = Q_{0,u} - \bar{u}$$
$$q = \frac{1}{9}(\frac{1}{a}(\frac{b^2}{a} - 3c))$$
$$r = \frac{1}{54}(\frac{1}{a}(2\frac{b^3}{a^2} - 9\frac{bc}{a} + 27d))$$

$$
\begin{array}{ll}
\text{IF} & r^2 < q^3 \\
& \theta = arccos(\frac{r}{\sqrt{q^3}}) \\
& t_0 = -2\sqrt{q}cos(\frac{\theta}{3}) - \frac{1}{3}\frac{b}{a} \\
& t_1 = -2\sqrt{q}cos(\frac{\theta+2\pi}{3}) - \frac{1}{3}\frac{b}{a} \\
& t_0 = -2\sqrt{q}cos(\frac{\theta-2\pi}{3}) - \frac{1}{3}\frac{b}{a}
\end{array}
$$

A particular root, $t_{\bar{u}}$, is retained if it is unique and within the curve domain $0 < t_{\bar{u}} < 1$. The uniqueness condition avoids spurious roots where the curve lies on the divide between cells (root denominator is zero) or touches but does not cross a cell wall (multiple equal roots).

In cases of higher degree, $p > 3$, the algebraic root finding must be replaced by a numerical search. A recursive de Casteljau subdivision can be performed on the source curve until those repeatedly halved segments which lie across a cell wall can be adequately approximated by a linear function. The problem of establishing the intersection values defaults at this base level to analytic linear root finding. Generally though, such high degree curves ($p > 3$) are not required in curve-based sculpting.

It is worth highlighting that the positioning of the lattice dramatically affects the number of cell-wall segmentations. This suggests that a strategy of shifting the lattice origin so as to limit the number of curve-cell intersections would be beneficial. The exact ramifications are left for future consideration.

6.3.2 Extracting Weights through Trivariate Composition

It is possible, as described in section 6.2.3 and by DeRose [1988] and Nimscheck [1995, pp. 48-51], for curves to undergo Free-Form Deformation through the agency of functional composition. The composition of a degree $\ell \times m \times n$ deformed hyperpatch, $\widetilde{\mathcal{H}}$, and a degree p pre-deformation curve segment, $\mathcal{G}_s(t)$, defines a deformed image, $\widetilde{\mathcal{G}}^{\star}_s(t) = \widetilde{\mathcal{H}} \circ \mathcal{G}_s(t)$, that can be expressed as a Bézier curve of degree $p(\ell + m + n)$

$$\widetilde{\mathcal{G}}^{\star}_s(t) = \sum_{d=0}^{p(\ell+m+n)} \mathcal{B}_d^{p(\ell+m+n)}(t) \cdot \widetilde{Q}^{\star}_{s,d}$$

The composed control points, $\widetilde{Q}^{\star}_{s,d}$, are, as embodied in equation 6.9, a complicated affine combination of hyperpatch control points, $\widetilde{P}$, compactly represented using polar form notation.

The goal here is not to deform a curve but instead to enable a curve's motion to direct the overall deformation.

To this end, each composed control point constitutes a constraint, separable into the product of a vector of weighting coefficients and a matrix of lattice control points:

$$\begin{aligned} \widetilde{Q}^{\star}_{s,d} &= \sum_{i=0}^{a+\ell-1} \sum_{j=0}^{b+m-1} \sum_{k=0}^{c+n-1} w_{s,d,(i,j,k)} \cdot \widetilde{P}_{i,j,k} \\ \widetilde{Q}^{\star}_{s,d} &= \mathbf{W}_{s,d}\widetilde{\mathbf{P}} \\ Q^{\star}_{s,d} + \Delta Q^{\star}_{s,d} &= \mathbf{W}_{s,d}(\mathbf{P} + \Delta\mathbf{P}) \\ \Delta Q^{\star}_{s,d} &= \mathbf{W}_{s,d}\Delta\mathbf{P} \end{aligned} \tag{6.16}$$

To achieve this, equation 6.9 is modified so as to generate the weighting vector component, $\mathbf{W}_{s,d}$, rather than the composed control point, $\widetilde{Q}^{\star}_{s,d}$. The lattice blossom, $\mathcal{P} : \Re^{\ell\times m\times n} \mapsto \Re^3$, which encodes an affine combination of hyperpatch control points, is replaced by a weighting blossom, $\mathcal{W} : \Re^{\ell\times m\times n} \mapsto \Re^{(a+\ell)\times(b+m)\times(c+n)}$, that returns a vector of affine combination coefficients divorced from their associated control points. Each blossom, $\mathcal{W}$, represents the flattening of a three-dimensional lattice-like structure with scalar values. Each element of $\mathcal{W}$ has a matching control point so that a particular lattice blossom, $\mathcal{P}(u_1,\ldots,u_\ell;v_1,\ldots,v_m;w_1,\ldots,w_n)$ can be recovered by straightforward matrix multiplication:

$$\mathcal{P}(u_1,\ldots,u_\ell;v_1,\ldots,v_m;w_1,\ldots,w_n) = [\mathcal{W}(u_1,\ldots,u_\ell;v_1,\ldots,v_m;w_1,\ldots,w_n)]\,\mathbf{P}$$

At the base level, the weights ascribed to individual control points, $P_{\bar{\imath},\bar{\jmath},\bar{k}}$, have a unit value in the appropriate position and zero elsewhere:

$$\begin{aligned} \mathcal{W}(\bar{u}_{\bar{\imath}},\ldots,\bar{u}_{\bar{\imath}+\ell-1};\bar{v}_{\bar{\jmath}},\ldots,\bar{v}_{\bar{\jmath}+m-1};w_{\bar{k}},\ldots,w_{\bar{k}+n-1}) = \\ \mathit{flatten}(\delta(\bar{\imath},i)\cdot\delta(\bar{\jmath},j)\cdot\delta(\bar{k},k)) \end{aligned} \tag{6.17}$$

where δ is the Kronecker delta function ($\delta(x,y) = 1$ when $x = y$ and 0 otherwise) and the $\mathit{flatten}$ operation unravels a three-dimensional structure into a vector. Equation 6.17 follows directly from the identity:

$$\begin{aligned} \mathcal{P}(\bar{u}_{\bar{\imath}},\ldots,\bar{u}_{\bar{\imath}+\ell-1};\bar{v}_{\bar{\jmath}},\ldots,\bar{v}_{\bar{\jmath}+m-1};w_{\bar{k}},\ldots,w_{\bar{k}+n-1}) = \\ \mathcal{W}(\bar{u}_{\bar{\imath}},\ldots,\bar{u}_{\bar{\imath}+\ell-1};\bar{v}_{\bar{\jmath}},\ldots,\bar{v}_{\bar{\jmath}+m-1};w_{\bar{k}},\ldots,w_{\bar{k}+n-1})\mathbf{P} = P_{\bar{\imath},\bar{\jmath},\bar{k}} \end{aligned}$$

Another aspect of this formulation is that the B-spline basis vector $\mathbf{B}(C) = \mathit{flatten}(\mathcal{N}^{\ell}_{i}(C_u)\cdot\mathcal{N}^{m}_{j}(C_v)\cdot\mathcal{N}^{n}_{k}(C_w))$ belonging to a point, C, within the hyperpatch (see section 3.4) can be recovered by evaluating the weighting blossom on the tridiagonal $\mathbf{B}(C) = \mathcal{W}(C_u^{\langle\ell\rangle}, C_v^{\langle m\rangle}, C_w^{\langle n\rangle})$. A similar relationship exists between Cox-de Boor recurrence for B-spline bases [Farin 1997, pp. 153-156] and weighting blossoms, $\mathcal{W}$, as there is between de Boor recurrence and lattice blossoms, $\mathcal{P}$.

We are now in a position to "engrave" the control points of the segmented source curve, $Q_{s,r} = (u_{s,d}, v_{s,d}, w_{s,d}); r \in \{0,\ldots,p\}$, onto the weighting blossom, thereby producing the composed

weights, $\mathbf{W}_{s,d}, d \in \{0, \ldots, p(l+m+n)\}$:

$$\begin{aligned}\mathbf{W}_{s,d} \quad = \quad & \sum_{e+f+g=d} \quad \sum_{\forall (i)_\ell \in {}^{\ell}\mathcal{I}_e^p} \quad \sum_{\forall (j)_m \in {}^{m}\mathcal{I}_f^p} \quad \sum_{\forall (k)_n \in {}^{n}\mathcal{I}_g^p} \\ & \Big[\mathcal{X}(e,f,g) \cdot \mathcal{Y}_e^p((i)_l) \cdot \mathcal{Y}_f^p((j)_m) \cdot \mathcal{Y}_g^p((k)_n) \cdot \mathcal{Z}_e((i)_\ell) \cdot \mathcal{Z}_f((j)_m) \cdot \mathcal{Z}_g((k)_n) \cdot \\ & \mathcal{W}(u_{s,i_1}, \ldots, u_{s,i_\ell}; v_{s,j_1}, \ldots, v_{s,j_m}; w_{s,k_1}, \ldots, w_{s,k_n})\Big] \end{aligned} \tag{6.18}$$

with $\mathcal{I}$, $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$ defined in equations 6.7 , 6.6 , 6.4 , 6.8. In addition to all the efficiency measures discussed in section 6.2.3 the computation cost of equation 6.18 can be cut back by two further measures. Due to the compact support of B-splines, only $(\ell+1)\cdot(m+1)\cdot(n+1)$ elements in any weighting vector are non-zero and actually require calculation. Furthermore, the axial independence of the tensor product formulation can be exploited by independently evaluating the $\ell+1$, $m+1$ and $n+1$ elements of $\mathcal{W}(u_{s,i_1}, \ldots, u_{s,i_\ell})$, $\mathcal{W}(v_{s,j_1}, \ldots, v_{s,j_m})$ and $\mathcal{W}(w_{s,k_1}, \ldots, w_{s,k_n})$ with eqn. 6.1. The B-spline polar recurrence formula (eqn. 6.1) remains valid despite changes in the base case and length of vectors. The axial weights are combined to supply the complete weighting blossom as follows:

$$\begin{aligned}&\mathcal{W}(u_{s,i_1}, \ldots, u_{s,i_\ell}; v_{s,j_1}, \ldots, v_{s,j_m}; w_{s,k_1}, \ldots, w_{s,k_n}) = \\ &\qquad flatten([\mathcal{W}(u_{s,i_1}, \ldots, u_{s,i_\ell})]_i\, [\mathcal{W}(v_{s,j_1}, \ldots, v_{s,j_m})]_j\, [\mathcal{W}(w_{s,k_1}, \ldots, w_{s,k_n})]_k)\end{aligned}$$

In the curve constraint system of eqn. 6.16 both $\mathbf{W}_{s,d}$ and $\Delta Q^\star_{s,d}$ must be extracted from the source and destination segments, $\mathcal{G}_s$ and $\widetilde{\mathcal{G}}_s$, before a constraint satisfying solution to the unknown lattice changes, $\Delta\mathbf{P}$, can be found. The derivation of the weighting coefficient, $\mathbf{W}_{s,d}$, has been scrutinised but the composed control point changes, $\Delta Q^\star_{s,d}$, have yet to be discussed. As a preliminary the degree (p) of both $\mathcal{G}_s$ and $\widetilde{\mathcal{G}}_s$ must be raised to match the degree ($p(\ell+m+n)$) of the composed system through $p(\ell+m+n-1)$ applications of eqn. 6.10. Now, the composed control point changes are simply the difference between the control points of the degree raised segments, $\mathcal{G}^\star_s$ and $\widetilde{\mathcal{G}}^\star_s$:

$$\Delta Q^\star_{s,d} = \widetilde{Q}^\star_{s,d} - Q^\star_{s,d}; \qquad d = \{0, \ldots, p(\ell+m+n)\}$$

If the unsegmented source curve, $\mathcal{G}(t)$, is of degree p and has q intersections with cell walls in the degree $\ell \times m \times n$ undistorted hyperpatch, then the ensuing system of linear equations will contain $(q+1)\cdot(p(\ell+m+n)+1)$ constraints ($s = 0, \ldots, q$ and $d = 0, \ldots, p(\ell+m+n)$ in eqn. 6.16).

6.3.3 Constraint Reduction

The trivariate composition process has proven to be dangerously degree elevating ($p \mapsto p(\ell+m+n)$). For example, directly manipulating a cubic Bézier curve contained within a single cell of a tricubic hyperpatch necessitates $3(3+3+3)+1 = 28$ constraints. This high degree has several damaging implications for solving the constraint system:

1. The computation overhead is large, since, as demonstrated in section 3.5, the pseudo-inverse calculation is $O(r^3)$ in the number of constraints, r.

2. If any of the curve segments, $\mathcal{G}_s(t)$, are parallel to one or more of the hyperpatch axes, $\vec{U}$, $\vec{V}$, $\vec{W}$, then much of the degree elevation is redundant. For instance a composed curve parallel to the $\vec{V}$-$\vec{W}$ plane has degenerate degree $p(m+n)$, since effectively $\ell = 0$ is constant. Similarly, a curve aligned with $\vec{W}$ and orthogonal to $\vec{U}$ and $\vec{V}$ is only raised to degree pn through what amounts to univariate composition. Only curves which lie to some extent diagonal to $\vec{U}$, $\vec{V}$, $\vec{W}$ are truly of degree $p(\ell+m+n)$ under composition.

3. If the source curve is short relative to the cell dimensions, the composition will give rise to a large number of closely clustered constraints, in which the rows of $\mathbf{W}\Delta\mathbf{P} = \Delta\mathbf{Q}$ are insufficiently differentiated. The consequent ill-conditioning of the linear constraint system leads to a $\Delta\mathbf{P}$ which is either highly inaccurate or unsolvable.

Fortunately, repeated degree reduction (eqn. 6.13) with Chebyshev interpolation (eqn. 6.14) of both the weighting vectors, $\mathbf{W}_s$, and alteration in composed control points, $\Delta\mathbf{Q}_s$, circumvents all of these difficulties. The drawback is that degree reduction tends to introduce an approximation error. A useful measure of this error, ε_c, is the maximum disparity between the actual deformed location of a point on the source curve, $\mathcal{F}(\mathcal{G}(t))$, and its intended position on the destination curve, $\widetilde{\mathcal{G}}(t)$, in proportion to the overall displacement:

$$\varepsilon_c = max\left(\frac{\left\|\mathcal{F}(\mathcal{G}(t)) - \widetilde{\mathcal{G}}(t)\right\|}{\left\|\widetilde{\mathcal{G}}(t) - \mathcal{G}(t)\right\|}\right); \quad t \in [0,1] \tag{6.19}$$

The challenge is to balance the extent of degree reduction, motivated by computation cost, redundancy and constraint conditioning concerns, against the percentage approximation error.

We have identified four representative curve manipulation test cases suitable for experimenting with constraint reduction. Different versions of each test type (I-IV) apply for manipulation of linear, quadratic or cubic curves and each type representative is carefully chosen to supply the worst error behaviour.

- **Type I (Volumetric).** The endpoints of the source curve are set at diagonally opposite corners of a hyperpatch cell, with the curve inbetween demonstrating, without degeneracy, the full characteristics of its degree. A type I cubic ($p = 3$) is a true space curve with inflection point, quite dissimilar to a quadratic ($p = 2$) parabolic arc, which in turn is distinct from a line segment ($p = 1$). The expectation is that this class of curve will be the least amenable to repeated error-free degree reduction.

- **Type II (Planar).** The source curve is collapsed with respect to one of the hyperpatch axes and is trapped within a plane parallel to the two remaining axes. Apart from this limitation, a type II curve spans two of the cell's dimensions from bottom left to top right and again exhibits as much degree behaviour as allowed. As previously observed, a composed type II curve vanishing in $\vec{U}$ should be reducible to $p(m+n)$ degree with no discernible error.

- **Type III (Axial).** The source curve is aligned with a single axis and orthogonal to the others. The curve spans the cell from one side to the other and its degree is manifest in the linear, quadratic or cubic parametrisation along its course. We predict that a $\vec{W}$ axis curve will allow degree reduction to pn without error.

- **Type IV (Nearly Vanishing).** It is entirely plausible that cell-wise segmentation will supply a tiny curve fragment that cuts across the corner of a cell. So this eventuality must be considered. It is probable that the ill-conditioning of type IV constraint will only be surmounted with much degree reduction. In more extreme cases, where the curve is one minuscule segment of a larger whole, its constraints can be completely discarded without significantly impacting the overall error.

The destination curve for each test case is selected to distort every part of the source curve appropriately with its degree.

	Linear (10 constraints)				Quadratic (19 constraints)				Cubic (29 constraints)			
	I	II	III	IV	I	II	III	IV	I	II	III	IV
Solvable	10	7	5	4	16	12	4	4	24	15	5	4
$\varepsilon_c < 0.1\%$	6	6	4	2	7	7	-	-	12	10	-	-
$\varepsilon_c < 1\%$	6	4	3	2	6	6	4	4	10	8	4	4
$\varepsilon_c < 5\%$	4	4	3	2	5	5	3	3	7	6	4	4

Table 6.1: **Bounds on the Approximation Error for Linear, Quadratic and Cubic Curve Constraint Reduction within a Tricubic Hyperpatch. The first row tabulates the maximum number of constraints for which the system remains solvable. The other three rows display the minimum constraints consistent with certain error bounds. The dashed entries (-) indicate that a particular upper error limit is not attainable.**

The outcome of constraint reduction on linear, quadratic and cubic curve manipulations of each representative type (I-IV) is summarised in table 6.1. Four levels of constraint reduction are tabulated: the minimum required to overcome ill-conditioning (and ensure solvability) and the maximum extent that contains the approximation error within 0.1%, 1% and 5%. The predictions about the behaviour of each test type under constraint reduction are, in the main, borne out by table 6.1.

In practice, a particular level of degree reduction is selected by forming the $\vec{U}$, $\vec{V}$, $\vec{W}$ component-wise convex hulls of the source curve control points, finding the closest type-match by examining the axial extent of each convex hull, and then cross-referencing this type with the desired approximation error. We believe that an error of less than one pixel discrepancy in a hundred, $\varepsilon_c < 1\%$, is tolerable in a sculpting application because, as mentioned in chapter 1, design aesthetics and interactivity override absolute accuracy.

6.3.4 Combining Different Manipulations

One of the advantages of curve-directed DMFFD is that it enables the combined manipulation of curves, points and derivative frames within a single deformation. By placing derivative frames along a curve and specifying pinned curve manipulations which have no displacement ($\Delta Q_r = 0$) the locator and domain curve features of Wires [Singh and Fiume 1998] can be emulated. On the other hand, unlike Wires, DMFFD provides for arbitrarily located point and derivative-frame manipulations free from any curve attachment. It can be argued that Wires might match this flexibility with curves of zero length or by borrowing from point-based radial methods (section 2.2.2), which are also reliant on shortest distance parametrisation.

In concrete terms, different DMFFD manipulations can be merged by appending extra rows to the system of linear constraint equations, but care must be taken to prevent redundancies and inconsistencies. A derivative-frame manipulation, previously specified in chapter 4 by an orthogonal frame, $D = (\vec{N}, \vec{B}, \vec{T})$, at an origin, C, with intended scaling, $S = (s_{\vec{N}}, s_{\vec{B}}, s_{\vec{T}})$, and rotation, by θ about $\vec{R} = (r_{\vec{N}}, r_{\vec{B}}, r_{\vec{T}})$, is a pertinent case in point. If the frame is located on a source curve, $C = \mathcal{G}(t_C)$, and the tangents of the curve and frame coincide, $\vec{T} = \frac{d\mathcal{G}}{dt}(t_C)/\|\frac{d\mathcal{G}}{dt}(t_C)\|$, as they should, then the user's control over the derivative frame is restricted to twisting and scaling around the axis of the curve. Given a derivative frame on a curve manipulation, $(\mathcal{G}, \widetilde{\mathcal{G}})$, at parameter value, t_C, the user if responsible for: (a) completing the orthonormal frame, by supplying the normal, $\vec{N}$, and binormal, $\vec{B}$, vectors, and (b) specifying a rotation, θ, about the tangent, $\vec{T}$, and a scaling of the tangent-orthogonal plane, $S = (s_{\vec{N}}, s_{\vec{B}})$. The curve manipulation implicitly dictates a translation of the origin, $\Delta C = \widetilde{\mathcal{G}}(t_C) - \mathcal{G}(t_C)$, and an alignment of the tangent, consisting of a rotation by $\alpha = \cos^{-1}(\vec{T} \cdot (\frac{d\widetilde{\mathcal{G}}}{dt}(t_C)/\|\frac{d\widetilde{\mathcal{G}}}{dt}(t_C)\|))$ about $\vec{R} = \vec{T} \times \frac{d\widetilde{\mathcal{G}}}{dt}(t_C)$ and a scaling of $\vec{T}$ by $s_{\vec{T}} = \|\frac{d\widetilde{\mathcal{G}}}{dt}(t_C)\|/\|\frac{d\mathcal{G}}{dt}(t_C)\|$. The complete affine transformation of the curve-attached derivative frame is:

$$\mathbf{T} = \left(\mathbf{D} \cdot \mathbf{R}(\alpha, \mathbf{D}^{-1}\vec{R}) \cdot \mathbf{R}(\theta, \mathbf{D}^{-1}\vec{T}) \cdot \mathbf{S}(s_{\vec{N}}, s_{\vec{B}}, s_{\vec{T}}) \cdot \mathbf{D}^{-1}\right)$$

where $\mathbf{R}(\phi, \vec{L})$ represents a rotation matrix by ϕ about $\vec{L}$, and $\mathbf{S}(u, v, w)$ is a scaling matrix with factors u, v, w relative to the $\vec{U}, \vec{V}, \vec{W}$ axes. The derivative constraints are then derived as per chapter 4 except that, in order to avoid duplication, the positional constraint, ΔC (the first row in eqn. 4.4) is dropped.

Another area of concern is with intersecting curve manipulations. If two source curves intersect at a point then, for the sake of consistency in the constraint system, their destination curves should similarly intersect. A simple means of ensuring this integrity is to split the two curve manipulations into four with one endpoint of each meeting at the intersection point.

6.4 Evaluation

A deeper understanding of curve manipulation through composition and degree reduction can be attained by comparison with other curve-based deformations. For this purpose, two alternatives are selected:

1. **Wires.** This embodies the leading Euclidean distance method. A closest point on Bézier curve routine [Glassner 1990, pp. 607-611] is at the core of our minimal implementation of Wires [Singh and Fiume 1998], which supports multiple source and destination curves but not locators or domain curves. The lack of these features does not impede its evaluation relative to curve-directed DMFFD.

2. **Curve-Sampled DMFFD.** The second basis of comparison is a straightforward alternative to curve composition which operates within the framework of DMFFD by creating point manipulations from discrete samples on the curve (as opposed to composing curve control points). A set of direct point manipulations ($C_i, \Delta C_i$) can easily be constructed from samples on a source and destination curve ($\mathcal{G}(t)$ and $\widetilde{\mathcal{G}}(t)$) at regularly distributed parameter intervals, such that $C_i = \mathcal{G}(t_i)$, $\Delta C_i = \widetilde{\mathcal{G}}(t_i) - \mathcal{G}(t_i)$. Obviously, such a solution is only an approximation whose coarseness depends on the number of samples. However, the door to approximation has been forced open by the necessity for degree reduction of composed curve constraints and it is worth considering under a different guise.

Figure 6.8 demonstrates the outcome of curve-directed DMFFD at varying levels of complexity. Figure 6.8[A] shows a single cubic curve constraint and changes in its region of influence effected by adjusting the number and size of cells in the hyperpatch. Complex and realistic examples, which combines point, derivative-frame and curve manipulation of more involved models, are provided by figure 6.8[B], and figures 6.9 - 6.11. Both Wires and curve-sampled DMFFD are capable of emulating this power and flexibility. The only reservation is that Wires, as presented by Singh and Fiume [1998], does not support manipulations which are not attached to curves.

Unlike Wires, both curve composed and curve sampled DMFFD inject approximation error. The degree reduction of composed curves makes use of Chebyshev economisation and thus evinces the optimal least-error for a given number of constraints, c. If c constraints are derived by simply sampling at c locations along the source and destination curves then the error will inevitably be greater. Table 6.2 captures the number of samples necessary to attain $\varepsilon_c < 0.1\%$, $\varepsilon_c < 1\%$ and $\varepsilon_c < 5\%$ error bounds on the different types of curve manipulations. This can be directly contrasted with table 6.1.

The largest discernible difference in the two approaches occurs with type I and II cubic curve manipulations, where damping the error to an equal level requires from 2 to 4 more constraints in the sampled case. The additional constraints only become a concern when approaching

	Linear (10 constraints)				Quadratic (19 constraints)				Cubic (29 constraints)			
	I	II	III	IV	I	II	III	IV	I	II	III	IV
Solvable	10	7	4	4	15	12	4	4	21	17	4	5
$\varepsilon_c < 0.1\%$	6	6	4	3	7	8	-	-	16	13	-	-
$\varepsilon_c < 1\%$	6	4	4	2	6	6	-	-	14	10	4	-
$\varepsilon_c < 5\%$	4	4	3	2	5	5	3	3	9	9	3	4

Table 6.2: **Bounds on Approximation Error for Point Sampling Along Linear, Quadratic and Cubic Curves. This table corresponds directly to table 6.1.**

overdetermination. Curve composition, for example, enables 6 type I cubic curve manipulations within a single cell at $\varepsilon_c < 1\%$ before forcing the constraint system into overdetermination, as contrasted with 4 for curve sampling. Additionally, for type III and IV quadratic and cubic curves, certain error bounds, specifically $\varepsilon_c < 1\%$, are simply not attainable with curve sampling. It is debatable whether the advantages of allowing more manipulations before overdetermination and achieving a lower overall error in degenerate situations are sufficient to overcome the conceptual complexity, and hence implementation difficulty, of curve composition. Without even considering computation cost, the sheer simplicity of curve sampling goes a long way to counterbalancing its slight deficiencies in approximation.

As revealed by the graphs in figure 6.7, which plot execution time on a logarithmic scale, both curve-directed and curve-sampled DMFFD offer significant improvements in speed over comparable curve-based deformation techniques (represented in our timing experiments by Wires). Typically, schemes which rely on a closest Euclidean distance parametrisation force the user to wait while curves are attached to an object (Wires with embedding) but then allow interactive deformation (Wires with deformation only) as these curves are reshaped. On the other hand, even though a single Wires curve may map to several composed segments (e.g. 7 in the case of the closely localised curve manipulation of figure 6.8A), the entire deformation cycle of curve-directed DMFFD is roughly an order of magnitude faster and satisfies interactivity (≥ 10 frames per second) for reasonable numbers of curve segments (< 10) and moderately large objects (< 7000 vertices). Curve sampling exhibits, as can be seen in figure 6.7, even better performance. This makes either form of curve-based DMFFD more suitable for interactive free-form modelling, where a set of curves may be used once and then discarded.

Another strength of curve-directed DMFFD, explained in chapter 7, is the ability to automatically detect self-intersection and break constraints into small injective steps. The same is not true of distance parametrising curve-based methods such as Wires. These have a seam of parametrisation discontinuity along the axis of a bend where the closest reference point jumps from one arm of the bend to the other. This negates the continuity conditions which underpin self-intersection analysis.

The hyperpatch substrate of curve-directed FFD allows C^k, $k = min(\ell, m, n) - 1$, derivative continuity to be maintained throughout the deformation. Wires, on the other hand, allows only C^1 continuity.

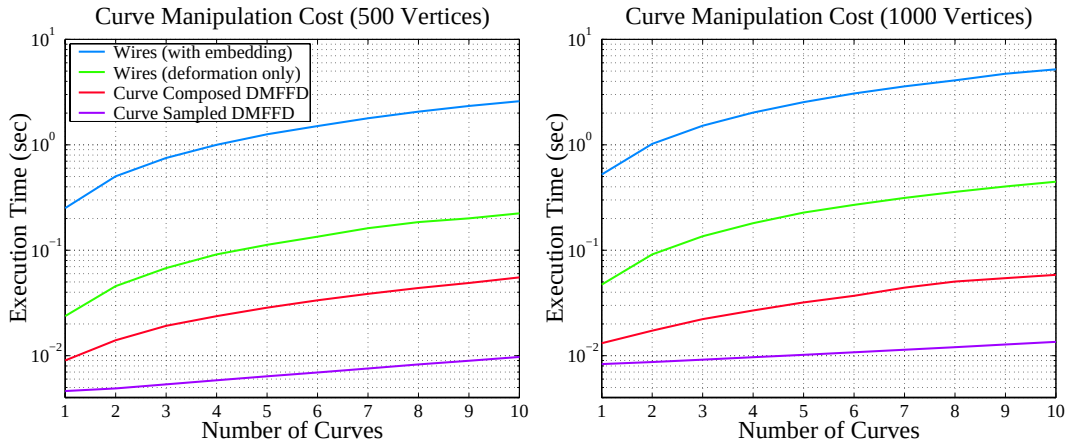


Figure 6.7: **An efficiency comparison of Curve-Directed DMFFD and Wires. These graphs plot the number of curves against the execution times on a logarithmic scale for the deformation of** 500 **and** 1000 **vertices. All methods were executed on an SGI Octane** 195 **MHz R**10000**.**

6.5 Conclusion

Curve-directed Directly Manipulated Free-Form Deformation maps source curves onto destination curves through a process of composition and degree reduction and this approach has several notable strengths:

- The fields of hyperpatch-, point- and curve-based spatial deformation are merged. In practical terms, extra rows can be joined to a constraint system to allow the combined manipulation of curves, points and derivative frames within a single deformation.
- It operates approximately an order of magnitude faster than equivalent distance parametrising methods, such as Wires [Singh and Fiume 1998] and Axial Deformations [Lazarus, Coquillart and Jancène 1994]. If the expensive process of attaching an object to a wire is discounted, then the execution costs of the curve composed and distance parametrising approaches are roughly comparable, since a single composed curve may consist of several curve segments. The principle efficiency advantage of curve-directed DMFFD is the ability to interactively specify source curves, so that a given set of curves may be used only a few times and then replaced with no deterioration in performance.
- Unlike distance parametrisations, curve-directed DMFFD is open to the automatic detection and prevention of self-intersection. The technicalities involved in achieving this are the subject of the next chapter.
- Higher orders of deformation continuity (above C^1) are attainable with curve-directed DMFFD simply by increasing the hyperpatch degree (e.g. a tricubic $3 \times 3 \times 3$ degree hyperpatch is C^2).

These benefits have been gained at the expense of introducing approximation error. Points which lie along the source curve prior to deformation may deviate from the intended destination curve following deformation. Fortunately, tight bounds can be placed on the magnitude of approximation error so that the fidelity of interactive sculpting is not damaged. Curve-sampled DMFFD, which generates point-manipulations by sampling the source and destination curves, achieves similar results to curve-directed DMFFD. Although somewhat more error prone, curve sampling is even faster (exceeding Wires by two orders of magnitude) and has all the advantages of a DMFFD formulation described above. In short, the superiority of curve composition over curve sampling is not clear cut. Nevertheless, curve manipulation with DMFFD, either through composition or sampling, represents an approach to interactive modelling which, at the expense of absolute accuracy in tracking a curve, greatly improves versatility, efficiency and correctness.

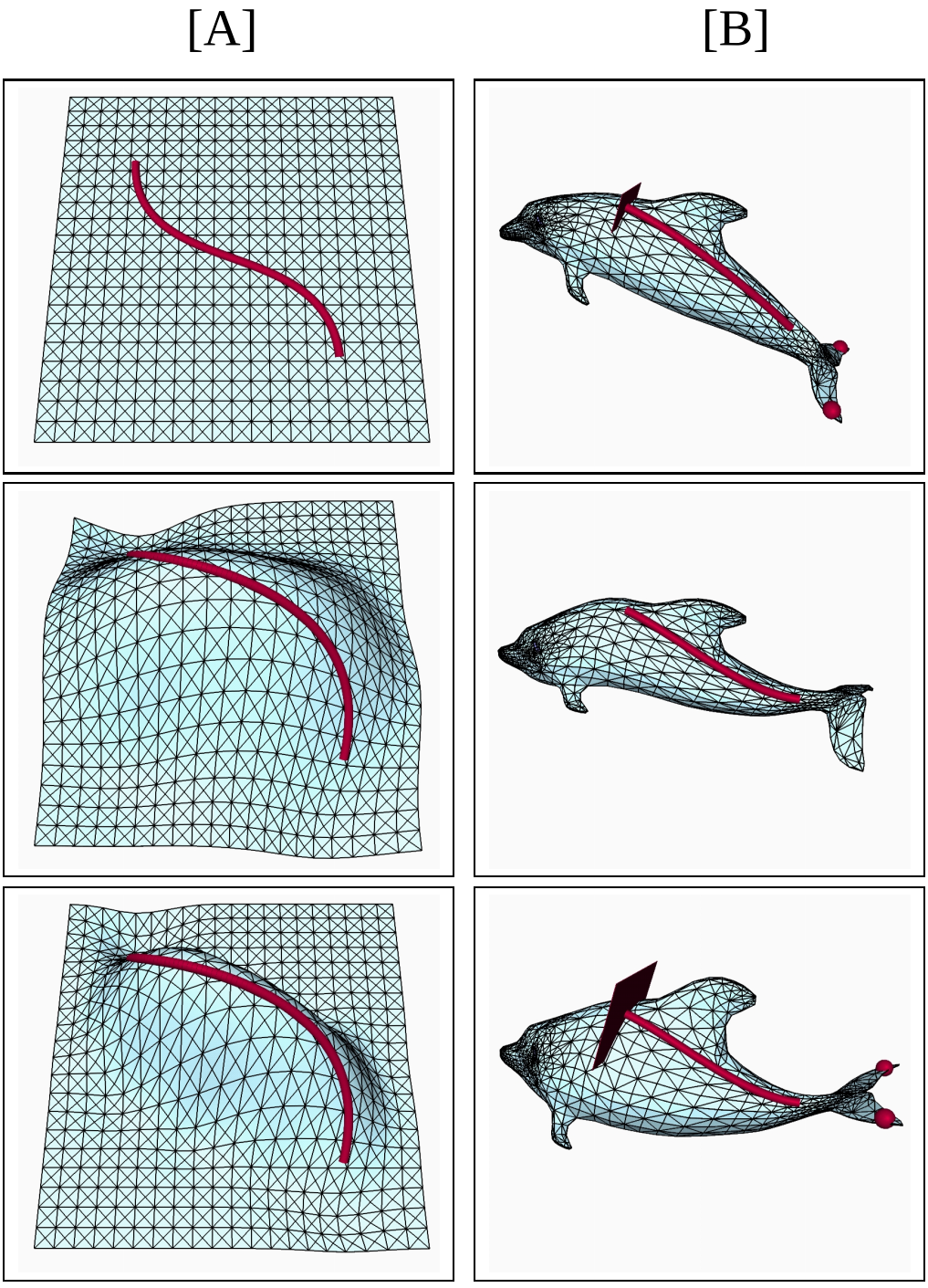


Figure 6.8: **Curve Manipulation. [A] Altering the region of influence of a single cubic curve (red) on planar object (blue): [top] source position, [middle] broad influence, [bottom] narrow influence. [B] A more complex manipulation combining point (two spheres), derivative-frame (rectangle) and curve constraints: [top] source position, [middle] curve alone, [bottom] all constraints combined.**

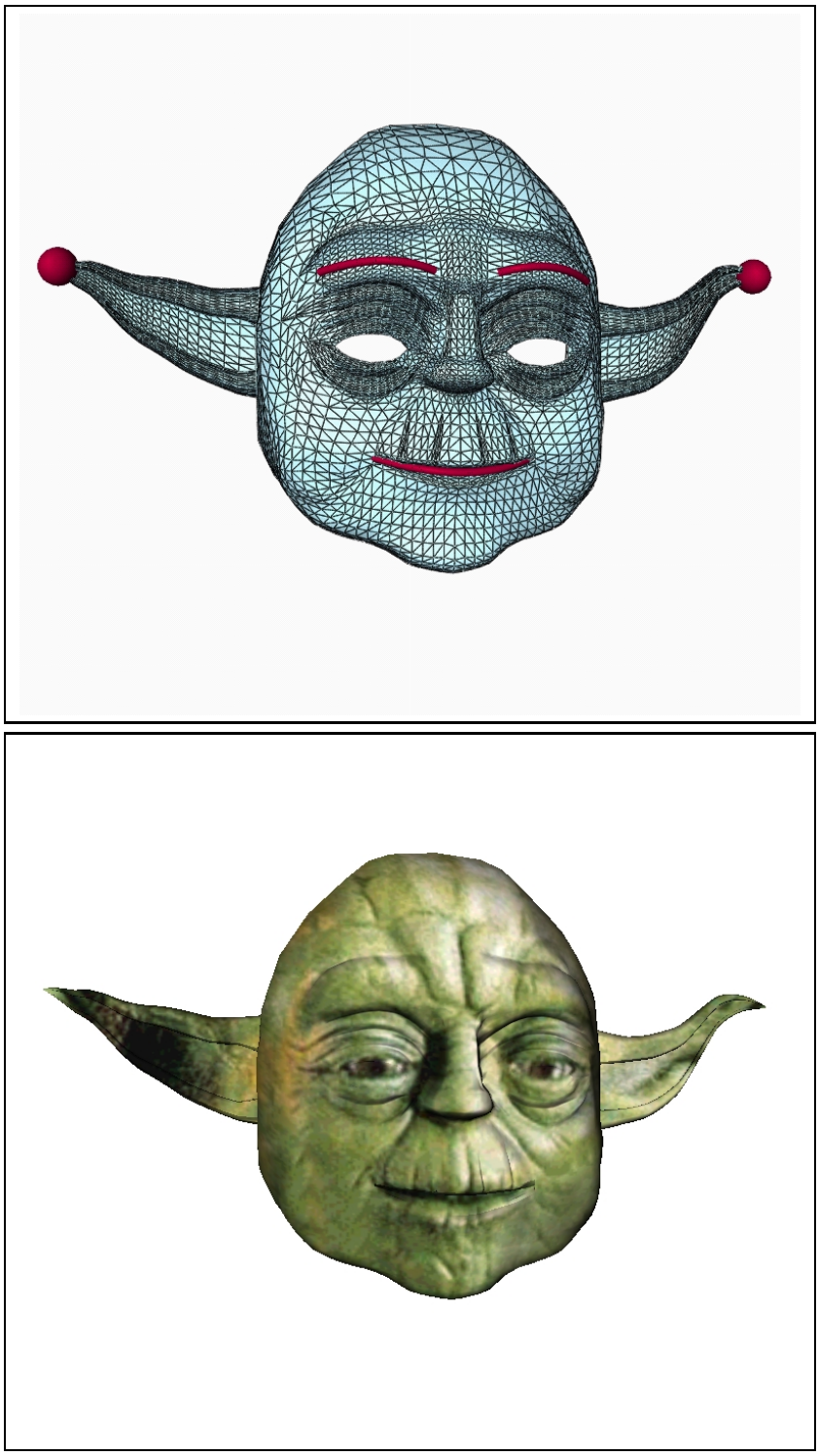


Figure 6.9: **Yoda (Neutral). [Top] Three quadratic source curves and two points are placed on the polygon-mesh model. [Bottom] A texture map is wrapped over the pre-deformation model.**

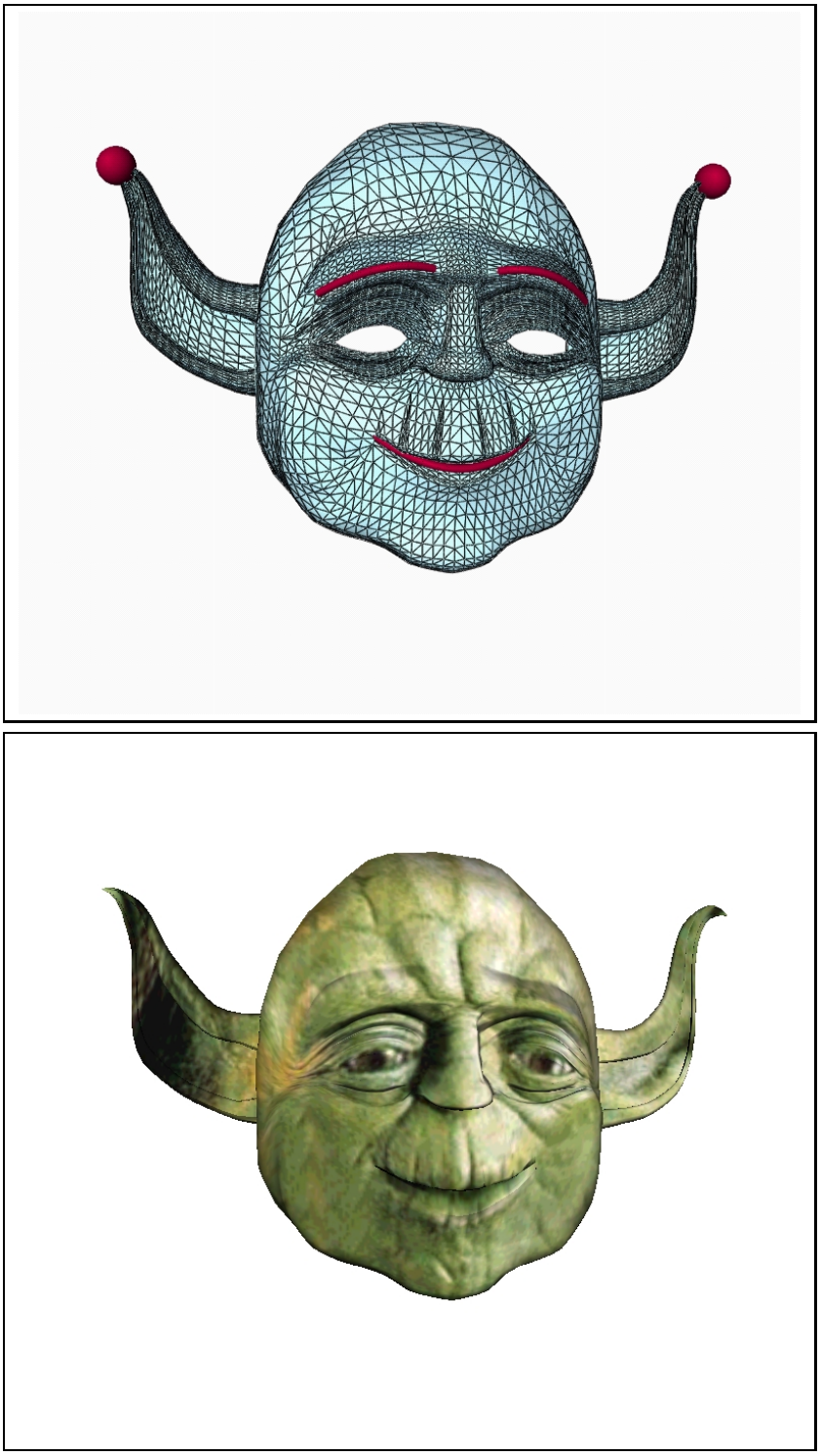


Figure 6.10: **Yoda (Happy). The ear tips and endpoints of the mouth curve are dragged upwards. The brow curves are pinched downwards at the outer corners.**

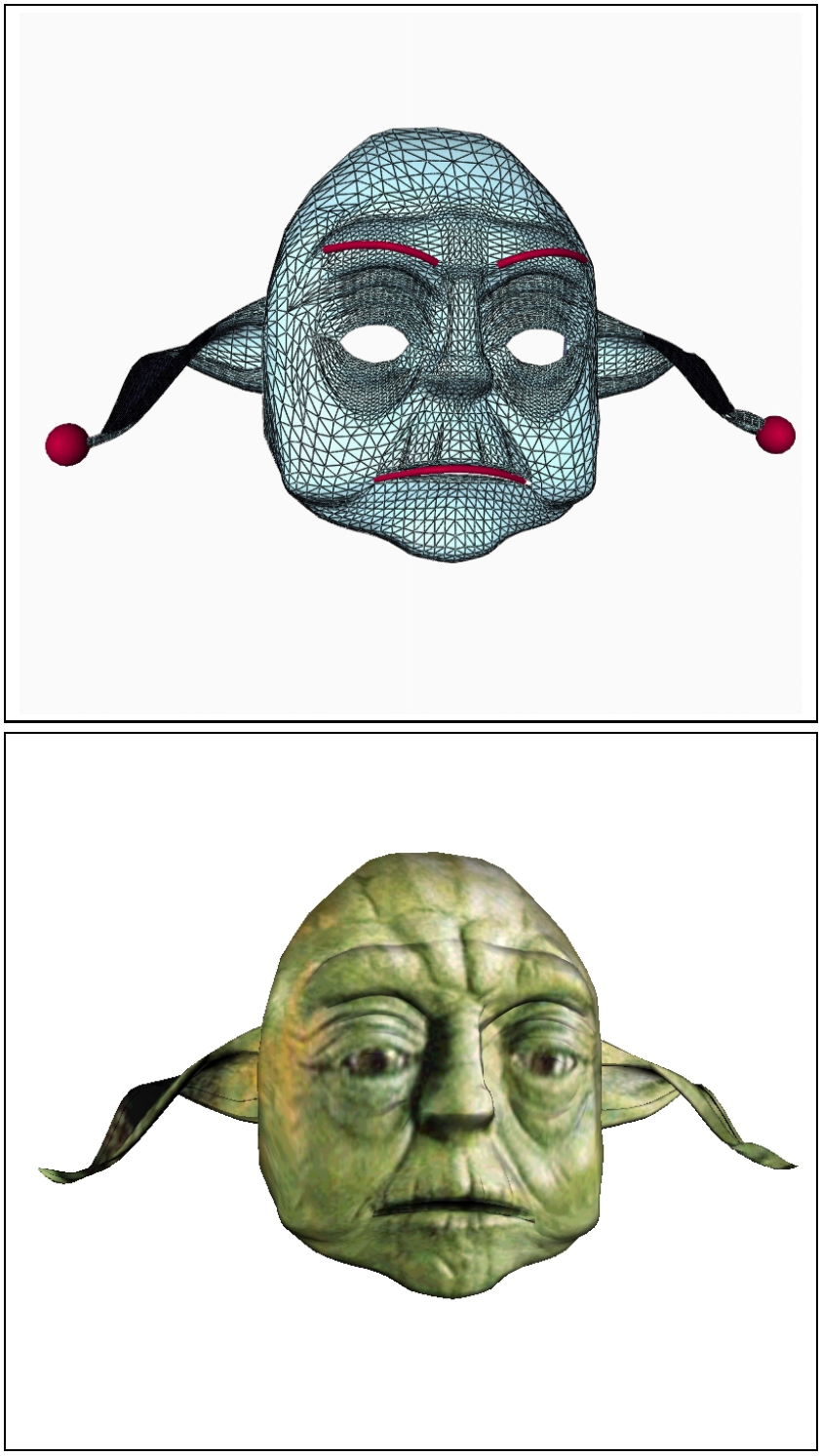


Figure 6.11: **Yoda (Sad). The ear tips and endpoints of the mouth curve sag downwards. The brow curves are raised slightly.**

Chapter 7

Preventing Self-Intersection

7.1 Introduction

A serious weakness common to all forms of spatial deformation is the potential for self-intersection of an object. Formally, the injectivity (or one-to-one mapping) is broken, as many points across the original object may map to a single point under deformation. This interpenetration of portions of the object's surface is problematic for a number of reasons. Firstly, it is highly counterintuitive. No real-world solids can contort in this fashion without rupturing and, as a consequence, self-intersection is unlikely to fulfil the intentions of the modeller. Secondly, it contravenes the correctness of the affected object. For instance, many boundary representation systems support only manifold solids (where any point and neighbourhood on the manifold is topologically equivalent to a disk in the plane) and these are invalidated by interpenetrating faces [Hoffmann 1989, pp. 58-61]. 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 [Watt 1989, p. 237]. The importance of ensuring that models do not self-intersect is evidenced by the inclusion of this condition in the ISO STEP Standard for CAD data transfer [Hardwick *et al.* 1996]. 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.

Unfortunately, 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.

A separate issue in the context of Free-Form Deformation is the relationship between hyperpatch

and object self-intersection. Self-intersection within a hyperpatch is required for self-intersection of the object. The converse does not always hold since an object's surface can skirt the areas of hyperpatch self-intersection. Thus hyperpatch self-intersection is a necessary but not sufficient condition for object 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 analysing 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 self-intersection on a polygon-by-polygon basis. This naïve, $O(n^2)$, object-based test can be improved by exploiting spatial coherence. Baraff and Witkin [1998] cluster polygons into a hierarchy of axis-aligned bounding boxes, and an absence of overlap between boxes allows a trivial rejection of lower levels in the hierarchy. In this context the lattice embedding provides an initial axis-aligned spatial subdivision at no additional cost and allows the immediate identification of deformed boxes. However, the computation costs of this method remain prohibitive and it suffers from numerous other failings: the complete inversion of an object, where all outward facing surface normals are turned inwards cannot be detected; the particular FFD under examination must be applied prior to the test, thereby increasing the computation burden; and the test is specific to a polygon-mesh representation.

Both tests are subject to error caused by the approximation inherent in the polygon-mesh representation. With reference to figure 7.1, a sparse mesh may track an object so poorly that it intersects itself where the true surface does not. The reverse is also possible: an object might self-intersect at high but not low resolutions. These failings are reduced by the adaptive mesh refinement scheme described in chapter 5.

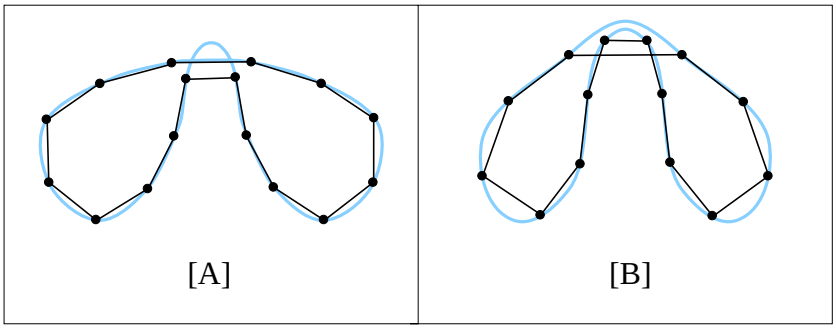


Figure 7.1: **Self-Intersection Error Due to Polygon-Mesh Approximation. [A] The 'true' surface (blue) self-intersects but not the mesh (black). [B] The 'true' surface does not intersect but the mesh does.**

The most important characteristic of any self-intersection test is that it correctly identifies a lack of injectivity, thereby guaranteeing the correctness of any deformed object. The opposite is not

as crucial. If certain deformations are falsely identified as self-intersecting this merely limits the range of allowable FFD. Another criterion is the computation cost of the chosen test. Given these considerations, an analytic hyperpatch-based self-intersection test is preferable.

Chapter Structure

The focus of this chapter is on automatically detecting and preventing self-intersection under Free-Form Deformation. We begin with a summary of previous contributions. A set of theoretical conditions for preventing self-intersection by ensuring the injectivity (one-to-one mapping) of the FFD hyperpatch follows. We then present a series of novel results:

- An exact (necessary and sufficient) injectivity test utilising 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 variant on the direct manipulation of points, curves and first derivatives which acts by concatenating many small injective deformations. This expands the range of valid deformations and enhances realism without sacrificing the speed of the approximate test.

7.2 Previous Work

There are several passing references to the dangers of self-intersection in the spatial deformation literature [Coquillart 1990; Lazarus, Coquillart and Jancène 1994; MacCracken and Joy 1996; Aubert and Bechmann 1997a] but only Borrel and Rappoport [1994] embark on a concerted investigation. They identify a "space-tearing" phenomenon in connection with Scodef, a simple radial technique reviewed in section 2.2.2. The problem (similar to figure 7.3[B]) occurs when two constraints with different displacements approach singularity and it is ameliorated by duplicating constraints. This effectively links two radii to each constraint: one controls the area of effect and the other determines the degree of influence on nearby constraints. Decreasing the latter radius among problematic constraints will reduce and eventually eliminate space-tearing. However, this places the onus of identifying and correcting self-intersection on the user.

Self-intersection has garnered much attention in the context of offset curves and surfaces [Hoffmann 1989, p. 227; Hoscheck and Lasser 1993, pp. 604-605]. These offset constructs typically lie a constant distance along the normal of a curve or surface and have many uses in engineering, for example in defining the shell of a ship or automobile. Self-intersection of offsets causes collision problems in the numerically controlled steering of milling tools and must be avoided. Unfortunately, this research has no direct relevance to spatial deformation.

In considering parametric curves and surfaces more generally, Goodman and Unsworth [1994] give sufficient but not necessary conditions for the injectivity under warping of Bézier triangles and tensor product patches that are restricted to a planar ($\Re^2$) domain. Their results, while valuable, are not extensible to three-dimensional deformations.

Andersson, Peters and Stewart [1998] provide a comprehensive analysis of the self-intersection of objects composited from Bézier curves and surfaces. They develop both exact (necessary and sufficient) and relaxed (merely sufficient) injectivity conditions for joined Bézier curves and triangles and, more generally, curvilinear simplicial complexes (patch-based objects). However, in these cases, unlike FFD, the domain (univariate $\Re$ or bivariate $\Re^2$) and range (embedding in $\Re^3$) do not match. It is therefore necessary to develop a body of self-intersection theory applicable to FFD.

7.3 Injectivity Analysis

Certain spatial distortions cause self-intersection (or even eversion) of an embedded object. Figure 7.3 shows three 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. These effects arise from a foldover in deformable space. In this section, the theoretical conditions for avoiding such foldovers are presented.

Self-intersection requires that at least two points in the initial space map to a single point in the deformed space. In particular, the inset object becomes self-intersecting if two or more points on its boundary prior to deformation are warped to coincide. An *injective* (one-to-one) mapping means that every pre-deformation point is transformed to a unique and separate post-deformation position. Injectivity thus implies non-self-intersection.

Definition 7.1 ([Clapham 1996, p. 196] Injective Mapping) *A mapping $\mathcal{F} : X \mapsto \widetilde{X}$ is injective (one-to-one) if, whenever x_1 and x_2 are distinct elements of X, their images $\mathcal{F}(x_1) = \widetilde{x}_1$ and $\mathcal{F}(x_2) = \widetilde{x}_2$ are distinct elements of $\widetilde{X}$. So $\mathcal{F}$ is injective if $\widetilde{x}_1 = \widetilde{x}_2$ implies that $x_1 = x_2$.*

As a precursor to the existence of a continuous inverse, spatial deformation should form an *onto* mapping, which projects directly from the domain to the codomain.

Definition 7.2 ([Clapham 1996, p. 196] Onto Mapping) *A mapping $\mathcal{F} : X \mapsto \widetilde{X}$ is onto if every element of the codomain $\widetilde{X}$ is the image under $\mathcal{F}$ of at least one element of the domain X. So $\mathcal{F}$ is onto if the range $\mathcal{F}(X)$ fills the whole of $\widetilde{X}$.*

These two mappings (injective and onto) combine to allow inversion.

Definition 7.3 ([Clapham 1996, p. 145] Inverse Mapping) *Let $\mathcal{F} : X \mapsto \widetilde{X}$ be a bijective mapping (both injective and onto). The inverse mapping $\mathcal{F}^{-1} : \widetilde{X} \mapsto X$ of $\mathcal{F}$ is defined as follows: for $\widetilde{x} \in \widetilde{X}$, $\mathcal{F}^{-1}(\widetilde{x})$ is the unique element $x \in X$ such that $\mathcal{F}(x) = \widetilde{x}$.*

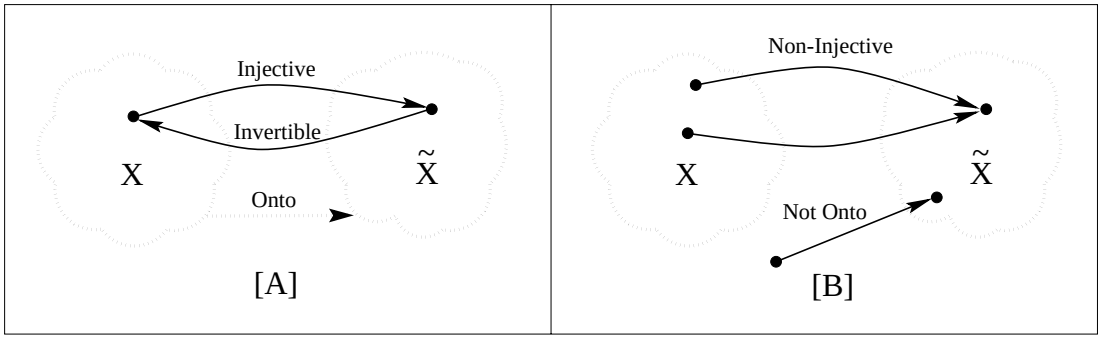


Figure 7.2: **Topological Concepts - Injective, Onto and Invertible Mappings. [A] A homeomorphic mapping between X and $\widetilde{X}$. [B] Non-homeomorphic counterexamples.**

All these qualities (injective, onto and invertible) are components of homeomorphism, which enables a space to be stretched, twisted or contracted but not overlapped.

Definition 7.4 ([Parker 1997, p. 111] Homeomorphic Mapping) *A continuous mapping $\mathcal{F} : X \mapsto \widetilde{X}$ is a homeomorphism if it is injective, onto and has a continuous inverse.*

These topological concepts are illustrated alongside their counterexamples in figure 7.2

The following theorem provides a set of requirements for the homeomorphism (and hence self-intersection prevention) of spatial deformation:

Theorem 7.1 *Let $\mathcal{F}$ be a spatial deformation function of the form $\mathcal{F} : X \mapsto \widetilde{X}$; $X, \widetilde{X} \subset \Re^n$ and* **J** *be the Jacobian matrix of $\mathcal{F}$. $\mathcal{F}$ is homeomorphic iff*

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

Theorem 7.1 relies on two coupled theorems: the inverse function theorem [Munem and Foulis 1984, p. 406-407], which gives requirements for local injectivity in the neighbourhood of a point, and a result [Meisters and Olech 1963], which extends this to an entire closed bounded domain. Theorem 7.1 is applicable to all forms of spatial deformation: it is independent of dimension (it can be applied with equal facility to planar or volume warping) and interaction mechanism (it examines the deformable space directly).

Our focus now turns to Free-Form Deformation. If the notation of section 3.2 is adopted, then FFD is a composition of embedding ($\mathcal{H}^{-1}(X) = U$) and deformation ($\widetilde{\mathcal{H}}(U) = \widetilde{X}$) functions, mapping from undeformed space (X) to deformed space ($\widetilde{X}$). The Jacobian matrices **J** and $\widetilde{\mathbf{J}}$ belong to $\mathcal{H}$ and $\widetilde{\mathcal{H}}$ respectively. The composed FFD function $\mathcal{F} = \widetilde{\mathcal{H}} \circ \mathcal{H}^{-1}$ is homeomorphic if both components satisfy the requirements of theorem 7.1 (i.e. if both $\mathcal{H}$ and $\widetilde{\mathcal{H}}$ are homeomorphisms).

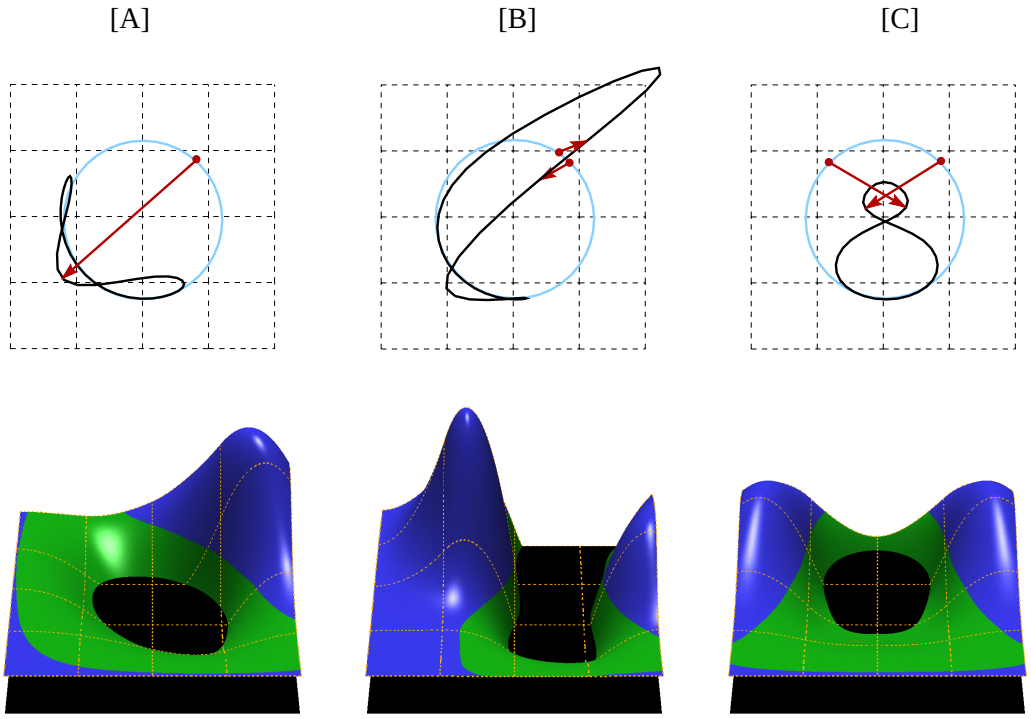


Figure 7.3: **Self-Intersection under Two-Dimensional DMFFD. [A] Overextension: an elongated constraint vector reaching beyond the area of effect of its associated point. [B] Overconstraint: two nearly coincident constraint points are wrenched in opposite directions. [C] Intersection: the displacement paths of two constraints cross. [Top] Object deformation showing a pre-deformation circle (light blue), its self-intersecting post-deformation image (black), and the instigating constraints (red). [Bottom] Jacobians represented as height fields above a two-dimensional domain. The peaks and troughs show areas of expansion (dark blue) and contraction (green). Injectivity is broken where the Jacobians drop below the black ground plane.**

The first condition in theorem 7.1 is easily met by ensuring at least C^1 continuity of the basis functions in $\mathcal{H}$ and $\widetilde{\mathcal{H}}$. B-splines in particular are $n-k$ times continuously differentiable on a degree n basis at a knot of multiplicity k [Farin 1997, p. 151] and $n-k \geq 1$ 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 (as advocated by Bartels, Beatty and Barsky [1983, pp. 27-29] and discussed in section 3.2). 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 Jacobians, $det(\mathbf{J}) > 0$ and $det(\widetilde{\mathbf{J}}) > 0$. The Jacobian at a point provides a measure of the local distortion. With a regular parallelepiped preliminary lattice the matching embedding Jacobian matrix has positive elements on the

diagonal and zero elsewhere and the determinant is guaranteed to be positive ($det(\mathbf{J}) > 0$). The deformation Jacobian, $det(\widetilde{\mathbf{J}})$, assumes the role of discriminator between the absence and presence of hyperpatch self-intersection. If a two-dimensional domain is envisaged as an elasticated sheet, then the Jacobian indicates the existence and magnitude of expansion ($det(\widetilde{\mathbf{J}}) > 1$), contraction ($0 < det(\widetilde{\mathbf{J}}) < 1$) or foldover ($det(\widetilde{\mathbf{J}}) < 0$) caused by contorting the sheet. As mentioned by Sederberg and Parry [1986] a deformation is perfectly volume preserving if the Jacobian is uniformly unity ($det(\widetilde{\mathbf{J}}) = 1$). In theorem 7.1 the positivity restriction (rather than the more conventional prevention of singularity, $det(\widetilde{\mathbf{J}}) \neq 0$) avoids the eversion of the domain that occurs with uniformly negative Jacobians. Figure 7.3 plots three deformation Jacobians which correspond to self-intersecting direct manipulations. Note the dark regions where the Jacobians fall below the plane $det(\widetilde{\mathbf{J}}) = 0$ and which signal that the deformations are folding back upon themselves.

Theorem 7.1 lays down necessary and sufficient conditions for establishing the self-intersection status of an FFD hyperpatch. We turn now to the practical implications of this theory.

7.4 Products of B-Splines

There is a useful set of recurrence relations, discovered by Mørken [1991], 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. Formation of the deformation Jacobian, $det(\widetilde{\mathbf{J}})$, induces many pairwise products and Mørken's formulae are pivotal in amalgamating them to provide a simpler, more manageable tensor product equation.

Assuming that we have two B-spline functions, $\mathcal{Q}'(u) = \sum_{j_1=0}^{s_1+n_1-1} \mathcal{N}_{j_1}^{n_1}(u) \cdot c'_{j_1}$ and $\mathcal{Q}''(u) = \sum_{j_2=0}^{s_2+n_2-1} \mathcal{N}_{j_2}^{n_2}(u) \cdot c''_{j_2}$ (with degrees n_1, n_2, knot sums s_1, s_2, and nature $\mathcal{Q}', \mathcal{Q}'' : \Re \mapsto \Re$), and the product, $\mathcal{Q} = \mathcal{Q}' \times \mathcal{Q}''$, is represented by $\mathcal{Q}(u) = \sum_{i=0}^{s+n-1} \mathcal{N}_i^n(u) \cdot c_i$; then Mørken's product recurrence relations can be used to evaluate the control coefficients of the product (c) given the coefficients of its multiplicands (c', c'').

Before these recurrence relations can be presented, the construction of the product knot vector (and, incidentally, the associated variables n and s) must be investigated. If two terms with degree n_1 and n_2 are multiplied together then the result has degree $n_1 + n_2$ ($u^{n_1} \cdot u^{n_2} = u^{n_1+n_2}$). It follows that the degree of the product function is $n = n_1 + n_2$. Let τ' and τ'' be the knot sequences linked to $\mathcal{Q}'$ and $\mathcal{Q}''$. The sum of domain knots, s_1 and s_2, reduced by multiple knots, corresponds to the number of domain intervals [Farin 1997, pp. 143-144]. From the combination of τ' and τ'' the knot sequence (t) and sum (s) of the product can be built. If τ' and τ'' share the same knot (k) with respective multiplicities m_1 and m_2 then $\mathcal{Q}'$ and $\mathcal{Q}''$ will have discontinuities of the $n_1 - m_1 + 1$ and $n_2 - m_2 + 1$ derivatives at that knot. Since the product cannot have a higher continuity at a knot join than either of its constituent functions, the implication is that $\mathcal{Q}$ is less than $min(n_1 - m_1 + 1, n_2 - m_2 + 1)$ times continuously differentiable at knot k. Hence t must have a knot at k of multiplicity $m = n - min(n_1 - m_1 + 1, n_2 - m_2 + 1) + 1 =$

$max(n_2 + m_1, n_1 + m_2)$. Now, the knot sequence t of $\mathcal{Q}$ is constructed as follows:

1. All knots from τ' and τ'' are placed, without repetition, in t.
2. The multiplicity, m, of a knot k in t must satisfy:

$$m = \begin{cases} max(n_2 + m_1, n_1 + m_2) & if\ k \in \tau', \tau'' \\ n_1 + m_2 & if\ k \in \tau'', k \notin \tau' \\ n_2 + m_1 & if\ k \in \tau', k \notin \tau'' \end{cases} \tag{7.1}$$

3. s can be extracted from t in the usual way by summing the domain knot multiplicities.

The control points of the product are generated by the identity:

$$c_i = \sum_{j_1=0}^{s_1+n_1-1} \sum_{j_2=0}^{s_2+n_2-1} c'_{j_1} \cdot c''_{j_2} \cdot \Gamma_{j_1,j_2,n_1,n_2}(i) \tag{7.2}$$

Γ denotes Mørken's product recurrence relations. These bear a strong resemblance to knot insertion formulae. This is not a chance occurrence: product recursion acts by separately inserting elements of the product knot sequence into the multiplicand sequences and blending. The knot sequences τ', τ'' and t are thus implicit parameters at every level of recursion.

Product Recurrence Relations (Γ)
Recursive Case: ($n_1 > 0, n_2 > 0$)

$$\begin{aligned} \Gamma_{j_1,j_2,n_1,n_2}(i) \quad = \quad & \frac{n_1}{n_1+n_2} \left\{ \frac{t_{i+n_1+n_2-1} - \tau'_{j_1-1}}{\tau'_{j_1+n_1-1} - \tau'_{j_1-1}} \cdot \Gamma_{j_1,j_2,n_1-1,n_2}(i) \right. \\ & \left. + \frac{\tau'_{j_1+n_1} - t_{i+n_1+n_2-1}}{\tau'_{j_1+n_1} - \tau'_{j_1}} \cdot \Gamma_{j_1+1,j_2,n_1-1,n_2}(i) \right\} \\ & + \frac{n_2}{n_1+n_2} \left\{ \frac{t_{i+n_1+n_2-1} - \tau''_{j_2-1}}{\tau''_{j_2+n_2-1} - \tau''_{j_2-1}} \cdot \Gamma_{j_1,j_2,n_1,n_2-1}(i) \right. \\ & \left. + \frac{\tau''_{j_2+n_2} - t_{i+n_1+n_2-1}}{\tau''_{j_2+n_2} - \tau''_{j_2}} \cdot \Gamma_{j_1,j_2+1,n_1,n_2-1}(i) \right\} \end{aligned} \tag{7.3}$$

Base Case: ($n_1 = 0$ or $n_2 = 0$)

$$\Gamma_{j_1,j_2,0,n_2}(i) = \alpha_{j_1,0,\tau',t}(i) \cdot \alpha_{j_2,n_2,\tau'',t}(i) \tag{7.4}$$

$$\Gamma_{j_1,j_2,n_1,0}(i) = \alpha_{j_1,n_1,\tau',t}(i) \cdot \alpha_{j_2,0,\tau'',t}(i) \tag{7.5}$$

n, n_1, n_2 are the degrees and t, τ', τ'' are the knot sequences of the B-spline product and its two multiplicands. The discrete B-splines (α) are defined by the following equations:

Discrete B-Spline Recurrence Relations (α)
Recursive Case: ($n > 0$)

$$\alpha_{j,n,\tau,t}(i) = \frac{t_{i+n-1} - \tau_{j-1}}{\tau_{j+n-1} - \tau_{j-1}} \cdot \alpha_{j,n-1,\tau,t}(i) + \frac{\tau_{j+n} - t_{i+n-1}}{\tau_{j+n} - \tau_j} \cdot \alpha_{j+1,n-1,\tau,t}(i) \tag{7.6}$$

Base Case: ($n = 0$)

$$\alpha_{j,0,\tau,t}(i) = \begin{cases} 1 & if\ \tau_{j-1} \leq t_i < \tau_j \\ 0 & elsewhere \end{cases} \tag{7.7}$$

It is helpful to demonstrate the evaluation of B-spline products with some examples:

Example 7.1 (The product of two Bézier functions) *Consider two cubic Bézier functions, both defined over the domain interval $u \in [0, 1]$. In a B-spline formulation they have associated knot sequences $\tau' = \tau'' = \{0\langle 3\rangle, 1\langle 3\rangle\}$. The numbers in angle braces indicate the multiplicity of the preceding non-simple knot (i.e. $\tau' = \tau''$ could be rewritten in expanded notation as $\{0, 0, 0, 1, 1, 1\}$). Since $n_2 + m_1 = n_1 + m_2 = 6$ at all knots in both τ' and τ'', the product has knots $t = \{0\langle 6\rangle, 1\langle 6\rangle\}$, and is of degree $n = n_1 + n_2 = 6$. The product is therefore also a Bézier function (of degree 6 on the domain $u \in [0, 1]$). In the special case of Bernstein bases the product coefficients have an explicit form:*

$$c_i = \sum_{\forall j_1 + j_2 = i} c'_{j_1} \cdot c''_{j_2} \cdot \frac{\binom{n_1}{j_1}\binom{n_2}{j_2}}{\binom{n}{i}}$$

This relies on the fact [Farin 1997, p. 79] that the product of two Bernstein bases is:

$$\mathcal{B}_i^m(u) \cdot \mathcal{B}_j^n(u) = \frac{\binom{m}{i}\binom{n}{j}}{\binom{m+n}{i+j}} \cdot \mathcal{B}_{i+j}^{m+n}(u) \tag{7.8}$$

Example 7.2 (The product of two uniform B-spline functions) *In this second example we examine the product of two uniform B-spline functions. The first multiplicand ($\mathcal{Q}'$) is linear ($n_1 = 1$) and has knot sequence $\tau' = \{1, 2\}$ (single knot multiplicities are implied) and the second is quadratic ($n_2 = 2$) with knots $\tau'' = \{0, 1, 2, 3\}$. These are both single segment functions ($s_1 = s_2 = 1$) on the domain $u \in [1, 2]$. The product, $\mathcal{Q} = \mathcal{Q}' \times \mathcal{Q}''$, has degree $n = 3$ over $t = \{0\langle 2\rangle, 1\langle 3\rangle, 2\langle 3\rangle, 3\langle 2\rangle\}$. Multiplication has introduced several superfluous knots. This is because the product recurrence relations apply to individual B-spline bases, which exist over a wider domain than the function. However, we are only interested in the product over the same interval as its arguments, $u \in [1, 2]$. The extreme knots of t can be discarded to obtain $t = \{1\langle 3\rangle, 2\langle 3\rangle\}$. Assuming that $\mathcal{Q}'$ and $\mathcal{Q}''$ have coefficients $\{0, 2\}$ and $\{0, 2, 1\}$, the control coefficients of the product are derived as follows:*

$$c_0 = \sum_{j_1=0}^{1}\sum_{j_2=0}^{2} c'_{j_1} \cdot c''_{j_2} \cdot \Gamma_{j_1,j_2,1,2}(0) = 0 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2} + 0 \cdot 0 + 0 \cdot 0 + 4 \cdot 0 + 2 \cdot 0 = 0$$

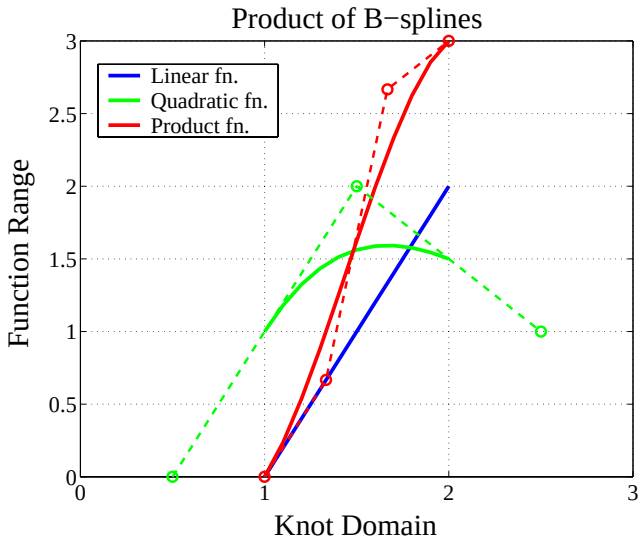


Figure 7.4: **The Product of B-Spline Functions. Linear (blue) and quadratic functions (green) are multiplied to produce a cubic function (red).**

$$c_1 = \sum_{j_1=0}^{1}\sum_{j_2=0}^{2} c'_{j_1} \cdot c''_{j_2} \cdot \Gamma_{j_1,j_2,1,2}(1) = 0\cdot 0 + 0\cdot\frac{2}{3} + 0\cdot 0 + 0\cdot\frac{1}{6} + 4\cdot\frac{1}{6} + 2\cdot 0 = \frac{4}{6}$$

$$c_2 = \sum_{j_1=0}^{1}\sum_{j_2=0}^{2} c'_{j_1} \cdot c''_{j_2} \cdot \Gamma_{j_1,j_2,1,2}(2) = 0\cdot 0 + 0\cdot\frac{1}{6} + 0\cdot\frac{1}{6} + 0\cdot 0 + 4\cdot\frac{2}{3} + 2\cdot 0 = 2\frac{2}{3}$$

$$c_3 = \sum_{j_1=0}^{1}\sum_{j_2=0}^{2} c'_{j_1} \cdot c''_{j_2} \cdot \Gamma_{j_1,j_2,1,2}(3) = 0\cdot 0 + 0\cdot 0 + 0\cdot 0 + 0\cdot 0 + 4\cdot\frac{1}{2} + 2\cdot\frac{1}{2} = 3$$

Figure 7.4 shows the multiplicands and the product functions with control points plotting the de Boor ordinates (c, c', c'') on the vertical axis against the Greville abscissae (ξ, ξ', ξ'') on the horizontal axis. Each Greville abscissa is an average of the knots, $\xi = \frac{1}{n}(t_i + \cdots t_{i+n-1})$ [Farin 1997, p. 144].

7.5 A Necessary and Sufficient Injectivity Test

The Jacobian of Free-Form Deformation is a mapping of the form $det(\widetilde{\mathbf{J}}) : \Re^3 \mapsto \Re$, 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 d-values reach or drop below zero anywhere in the domain, thereby breaking the second homeomorphism condition of theorem 7.1.

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 by making use of Mørken's [1991] product of B-splines, and this is then recursively subdivided into sub-cells until, by examining the signs of the refined control scalars, either a negative sub-cell is encountered or all sub-cells are found to be positive. This algorithm can now be examined in more detail.

7.5.1 Derivation of the Deformation Jacobian

The deformation Jacobian of FFD in $\Re^3$, $det(\widetilde{\mathbf{J}})$, is the determinant of a 3×3 matrix of partial derivatives:

$$det(\widetilde{\mathbf{J}}) = det \begin{bmatrix} \frac{\partial \widetilde{\mathcal{F}}_x}{\partial u} & \frac{\partial \widetilde{\mathcal{F}}_x}{\partial v} & \frac{\partial \widetilde{\mathcal{F}}_x}{\partial w} \\ \frac{\partial \widetilde{\mathcal{F}}_y}{\partial u} & \frac{\partial \widetilde{\mathcal{F}}_y}{\partial v} & \frac{\partial \widetilde{\mathcal{F}}_y}{\partial w} \\ \frac{\partial \widetilde{\mathcal{F}}_z}{\partial u} & \frac{\partial \widetilde{\mathcal{F}}_z}{\partial v} & \frac{\partial \widetilde{\mathcal{F}}_z}{\partial w} \end{bmatrix} \tag{7.9}$$

A tensor product expression for the deformation Jacobian can be obtained by substituting the partial derivatives of equations 3.7 - 3.9 into equation 7.9 and regrouping terms:

$$\begin{aligned} det(\widetilde{\mathbf{J}}) = & \sum_{d=1}^{a+\ell-1} \sum_{e=0}^{a+\ell-1} \sum_{f=0}^{a+\ell-1} \mathcal{N}_d^{\ell-1}(u) \cdot \mathcal{N}_e^{\ell}(u) \cdot \mathcal{N}_f^{\ell}(u) \cdot \\ & \sum_{i=1}^{b+m-1} \sum_{j=0}^{b+m-1} \sum_{k=0}^{b+m-1} \mathcal{N}_i^{m-1}(v) \cdot \mathcal{N}_j^{m}(v) \cdot \mathcal{N}_k^{m}(v) \cdot \\ & \sum_{r=1}^{c+n-1} \sum_{s=0}^{c+n-1} \sum_{t=0}^{c+n-1} \mathcal{N}_r^{n-1}(w) \cdot \mathcal{N}_s^{n}(w) \cdot \mathcal{N}_t^{n}(w) \cdot \phi \end{aligned} \tag{7.10}$$

where

$$\phi = det \begin{bmatrix} \widetilde{P}_{d,j,t}^{\langle 100 \rangle} & \widetilde{P}_{e,i,s}^{\langle 010 \rangle} & \widetilde{P}_{f,k,r}^{\langle 001 \rangle} \end{bmatrix} \tag{7.11}$$

and $\langle \alpha\beta\gamma \rangle$ denotes the difference operator.

$$\begin{aligned} \widetilde{P}_{d,j,t}^{\langle 100 \rangle} &= (\widetilde{P}_{d,j,t} - \widetilde{P}_{d-1,j,t}) \\ \widetilde{P}_{e,i,s}^{\langle 010 \rangle} &= (\widetilde{P}_{e,i,s} - \widetilde{P}_{e,i-1,s}) \\ \widetilde{P}_{f,k,r}^{\langle 001 \rangle} &= (\widetilde{P}_{f,k,r} - \widetilde{P}_{f,k,r-1}) \end{aligned}$$

The components along each axis of equation 7.10 are the products of three B-spline basis functions (e.g. $\mathcal{N}^{\ell-1}(u) \otimes \mathcal{N}^{\ell}(u) \otimes \mathcal{N}^{\ell}(u)$). Equation 7.10 is particularly unmanageable since

it is costly to evaluate and standard operations such as subdivision cannot be applied directly. This is where Mørken's product formulae come into their own. Equations 7.3 - 7.7 can be applied repeatedly to simplify the B-spline products in equation 7.10 and obtain a more tractable trivariate tensor product form.

$$det(\widetilde{\mathbf{J}}) = \sum_{o=0}^{\widehat{o}} \sum_{p=0}^{\widehat{p}} \sum_{q=0}^{\widehat{q}} \mathcal{B}_o^{\widehat{\ell}}(u) \cdot \mathcal{B}_p^{\widehat{m}}(v) \cdot \mathcal{B}_q^{\widehat{n}}(w) \cdot \varrho_{o,p,q} \tag{7.12}$$

where the upper limits of summation are $\widehat{o} = a(2\ell+1) + \ell - 3$, $\widehat{p} = b(2m+1) + m - 3$, $\widehat{q} = c(2n+1) + n - 3$ with degrees $\widehat{\ell} = 3\ell - 1$, $\widehat{m} = 3m - 1$ and $\widehat{n} = 3n - 1$. Assuming that the original hyperpatch knots are simple and evenly spaced at unit intervals, it follows that the knot sequences of the product obey the structure:

$$t_u = \{-1\langle \ell - 2\rangle,\ 0\langle 2\ell+1\rangle, \ldots,\ a\langle 2\ell+1\rangle,\ a+1\langle \ell-2\rangle\} \tag{7.13}$$

The sequences t_v, t_w are found by substituting b, m and c, n for a, ℓ in eqn. 7.13. For a single celled tricubic hyperpatch on the domain $u, v, w \in [0, 1]$ the relevant values are $\widehat{o}, \widehat{p}, \widehat{q} = 8$; $\widehat{\ell}, \widehat{m}, \widehat{n} = 8$; $t_u, t_v, t_w = \{-1, 0\langle 7\rangle, 1\langle 7\rangle, 2\}$.

Each of the control scalars, $\varrho_{o,p,q}$, is a combination of determinant scalars, ϕ, and product bases, Γ:

$$\begin{aligned}
\varrho_{o,p,q} \quad = \quad & \sum_{d=0}^{a+\ell-2} \sum_{\alpha=0}^{a(\ell+1)+\ell-1} \Gamma_{d,\alpha,\ell-1,2\ell}(o) \cdot \sum_{e=0}^{a+\ell-1} \sum_{f=0}^{a+\ell-1} \Gamma_{e,f,\ell,\ell}(\alpha) \\
& \left(\sum_{i=0}^{b+m-2} \sum_{\beta=0}^{b(m+1)+m-1} \Gamma_{i,\beta,m-1,2m}(p) \cdot \sum_{j=0}^{b+m-1} \sum_{k=0}^{b+m-1} \Gamma_{j,k,m,m}(\beta) \right. \\
& \left(\sum_{r=0}^{c+n-2} \sum_{\gamma=0}^{c(n+1)+n-1} \Gamma_{r,\gamma,n-1,2n}(q) \cdot \sum_{s=0}^{c+n-1} \sum_{t=0}^{c+n-1} \Gamma_{s,t,n,n}(\gamma) \right. \\
& \left. \left. det \begin{bmatrix} \widetilde{P}_{d+1,j,t}^{\langle 100\rangle} & \widetilde{P}_{e,i+1,s}^{\langle 010\rangle} & \widetilde{P}_{f,k,r+1}^{\langle 001\rangle} \end{bmatrix} \right) \right)
\end{aligned} \tag{7.14}$$

The indices of summation (d, i, r) of the derivative bases in eqn. 7.10 have been shifted down by removing the first elements from the hyperpatch knot vectors.

It is worth noting that although the degree of the component B-spline functions are substantially increased ($\ell \mapsto 3\ell - 1$, $m \mapsto 3m - 1$, $n \mapsto 3n - 1$) this does not apply to the knot continuity which remains $C^{\ell-1}$, C^{m-1} and C^{n-1}. This almost complete degeneracy results from large knot multiplicities and it simplifies the conversion to multi-Bézier form in the next phase. Deriving the control scalars $\varrho_{o,p,q}$ in equation 7.12 is very costly, even if the discrete product bases are amenable to pre-evaluation as they are in the uniform B-spline case.

7.5.2 Sign Test by Recursive Subdivision

The algorithm next focuses on each cell of the deformation Jacobian (eqn. 7.12) in turn. A cell under consideration is converted from B-spline to Bézier form using repeated insertion of domain knots until they are all of full multiplicity ($\hat{\ell}, \hat{m}, \hat{n}$). A standard univariate knot insertion algorithm [Farin 1997, pp. 143-147] can be iterated in a tensor product fashion along each axis. This is useful because Bézier curves (and hence hyperpatches) interpolate their endpoints and thus have a tighter convex hull than their lower multiplicity B-spline counterparts. Since the final stage of the injectivity test relies on recursively shrinking the convex hull towards the hyperpatch, this conversion improves the starting conditions.

Recursive subdivision proceeds by successively refining the control scalars of each Jacobian Bézier cell into eight sub-cells. The de Casteljau algorithm [Farin 1997, pp. 34-46, 53-56] is applied repeatedly to split the cell in half along the u, v, and then w co-ordinate axes. The subdivided control scalars converge quadratically [Dahmen 1986] towards the Jacobian hyperpatch. The recursive subdivision search space is an octree, with each node (cell) spawning eight children (sub-cells). The leaves of the octree are sub-cells whose control scalars are either all negative or all positive, which by the convex hull property of Bézier curves implies that the sub-cell 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.

The alternative is an iterative search technique which, starting from a heuristically selected point in the domain of the Jacobian, converges over several iterations to a minimum. Unfortunately, there is no guarantee that the minimum located is absolute, rather than local. A local minimum may be positive while the absolute minimum is negative, causing an incorrect injectivity classification. So, a recursive subdivision procedure is preferred because of its robustness in locating the absolute minimum.

7.5.3 Performance

The necessary and sufficient test described in this section is capable of precisely separating injective and non-injective deformations, but its computation burden is high. An SGI Octane ($\mathrm{R}10000 \times 195$MHz) requires at least 37s to test the injectivity of a single-celled hyperpatch even without any recursive subdivision of the Jacobian control scalars. If ten updates per second is considered reasonably interactive then, even without the additional overhead of FFD, this test is roughly three orders of magnitude too slow.

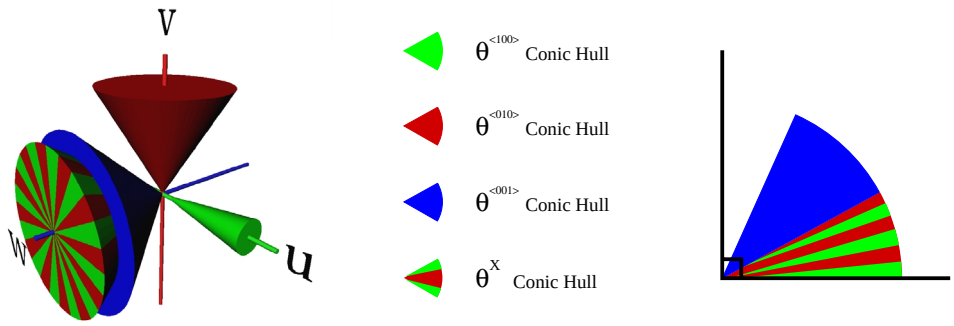


Figure 7.5: **The Three-Dimensional Conic-Hull Hodograph of an Injective FFD. [Left] Bounding cones for the lattice edges (red, green and blue) and a cross product (striped) are established. [Right] The two w-axis bounds sum to less than** 90 **degrees.**

7.6 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 curtail the precise algorithm to produce a weak sufficient test. This sacrifices the full range of injective deformations for improved performance by classing some valid deformations as self-intersecting.

The positivity of the Jacobian control scalars ($\phi > 0$) in equation 7.10 is, by the convex hull property of B-splines [Farin 1997, pp. 158-159], a sufficient condition for injectivity of FFD. The evaluation of ϕ can be optimised by: (a) exploiting the local control of B-splines, which allows the determinant scalars to be evaluated in $\ell+1 \cdot m+1 \cdot n+1$ blocks and (b) pre-calculating the partial derivative vectors ($\widetilde{P}^{\langle 100 \rangle}$, $\widetilde{P}^{\langle 010 \rangle}$ and $\widetilde{P}^{\langle 001 \rangle}$) over the entire lattice. Unfortunately, this test remains expensive.

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. The volume's sign is determined by the orientation of the third vector relative to the plane formed by the other two. This relationship is expressed in the following vector equation:

$$\begin{aligned} \phi &= det \begin{bmatrix} \widetilde{P}^{\langle 100 \rangle} & \widetilde{P}^{\langle 010 \rangle} & \widetilde{P}^{\langle 001 \rangle} \end{bmatrix} \\ &= (\widetilde{P}^{\langle 100 \rangle} \times \widetilde{P}^{\langle 010 \rangle}) \cdot \widetilde{P}^{\langle 001 \rangle} \end{aligned} \tag{7.15}$$

The cross product ($\widetilde{n} = \widetilde{P}^{\langle 100 \rangle} \times \widetilde{P}^{\langle 010 \rangle}$) produces a vector normal to the plane defined by its arguments ($\widetilde{P}^{\langle 100 \rangle}, \widetilde{P}^{\langle 010 \rangle}$). The dot product ($\phi = \widetilde{n} \cdot \widetilde{P}^{\langle 001 \rangle}$) is negative or positive depending on whether the angle between $\widetilde{n}$ and $\widetilde{P}^{\langle 001 \rangle}$ is either obtuse ($> \frac{\pi}{2}$) or acute ($< \frac{\pi}{2}$).

The geometric algorithm proceeds by constructing a partial derivative hodograph for each axis. A hodograph [Kim 1993; Farin 1997, p. 49] is formed by basing the derivative control vectors

at a fixed position in Euclidean space (usually the origin). Each hodograph is then enclosed by a conic hull which bounds the partial derivative vectors. In specific terms, for each derivative vector, the deviation angle relative to its particular axis is used to establish bounds on the maximum deviation ($\theta^{\langle 100\rangle}, \theta^{\langle 010\rangle}, \theta^{\langle 001\rangle}$). Equation 7.15 can now be recast in terms of these conic-hulls.

- Cross Product: As long as two conic hulls ($\theta^{\langle 100\rangle}, \theta^{\langle 010\rangle}$) bounding orthogonal axes (u, v) do not overlap (i.e. $\theta^{\langle 100\rangle} + \theta^{\langle 010\rangle} < \frac{\pi}{2}$) then a bound for the divergence of their cross product ($\theta^{\times}$) from the third axis (w) can be found using spherical trigonometry [Sederberg and Meyers 1988]:

$$\theta^{\times} = \arcsin(\sqrt{\sin^2\theta^{\langle 100\rangle} + \sin^2\theta^{\langle 010\rangle}}) \tag{7.16}$$

- Dot Product: As long as two conic hulls around the same axis ($\theta^{\times}, \theta^{\langle 001\rangle}$) diverge by less than ninety degrees ($\theta^{\times} + \theta^{\langle 001\rangle} < \frac{\pi}{2}$), the dot product of any two vectors within those conic hulls will be positive, and hence satisfy ϕ-injectivity.

The conic hull hodograph of a ϕ-injective FFD is shown in figure 7.5 and the algorithm appears below. For efficiency purposes, angular axis deviation (θ) is established using the metric $t = \tan^2\theta$. Once the maximum deviation metric has been found for each axis ($t^{\langle 100\rangle}, t^{\langle 010\rangle}, t^{\langle 001\rangle}$), the actual angles ($\theta^{\langle 100\rangle}, \theta^{\langle 010\rangle}, \theta^{\langle 001\rangle}$) can be derived. This is followed by the cross and dot product test.

```
AdjustBounds(t, a, b, c)
(Adjust the current conic hull measure (t = tan²θ) to incorporate
  a given vector (a, b, c) relative to the axis (1, 0, 0))

  IF a ≤ 0 THEN
    RETURN 'fail'
  e ← (b² + c²)/a²
  IF e > t THEN
    t ← e

InjectiveTest(ΔP)
(Establish the conic hull for each lattice direction given the
  control point changes (ΔP))

  t^⟨100⟩ ← 0,  t^⟨010⟩ ← 0,  t^⟨001⟩ ← 0
  FOR all u-directed lattice edges ē = P̃^⟨100⟩ DO
    AdjustBounds(t^⟨100⟩, ē_i, ē_j, ē_k)
  FOR all v-directed lattice edges ē = P̃^⟨010⟩ DO
    AdjustBounds(t^⟨010⟩, ē_j, ē_k, ē_i)
```

```
FOR all w-directed lattice edges ē = P̃^⟨001⟩ DO
  AdjustBounds(t^⟨001⟩, ē_k, ē_i, ē_j)
θ^⟨100⟩ ← arctan(√t^⟨100⟩)
θ^⟨010⟩ ← arctan(√t^⟨010⟩)
θ^⟨001⟩ ← arctan(√t^⟨001⟩)

(Test the conic hulls against each other)
IF (θ^⟨100⟩ + θ^⟨010⟩ ≥ π/2) THEN
  RETURN 'non-injective'
ELSE
  θ^× ← arcsin(√(sin²θ^⟨100⟩ + sin²θ^⟨010⟩))
  IF (θ^× + θ^⟨001⟩ ≥ π/2) THEN
    RETURN 'non-injective'
  ELSE RETURN 'injective'
```

It is worth noting that the range of deformations allowed by the ϕ-test is further reduced by the approximations inherent in this algorithm. However, the conic-hull hodograph injectivity test is $O((a+\ell)\cdot(b+m)\cdot(c+n))$ and well over three orders of magnitude faster than the original $O((a+\ell)^3\cdot(b+m)^3\cdot(c+n)^3)$ ϕ-test from equation 7.12.

7.7 Adaptive Subdivision of Direct Manipulation

The conic-hull hodograph test for injectivity, while efficient, is only weakly sufficient and restricts the range of allowable Free-Form Deformations beyond what is required to inhibit self-intersection. However, it can be incorporated into DMFFD to yield an effective variant which circumvents this problem. In principle, a deformation comprising any mixture of point, derivative and curve manipulations, is broken, where possible, into shorter injective steps. This is implemented as a recursive procedure which, on failure of the conic-hull hodograph test, splits every direct manipulation in a set into two adjoining pieces. If successful this 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 splitting mechanism is dependent on the type of manipulation:

1. **Point Manipulations** $(C, \Delta C)$ A manipulation, with point (C) and displacement vector (ΔC) parts, is split into two halves, $(C, \frac{1}{2}\Delta C)$ and $(C+\frac{1}{2}\Delta C, \frac{1}{2}\Delta C)$, that join head to tail.

2. **Curve Manipulations** $(Q_i, \Delta Q_i, i = 0, \ldots, k)$ The decomposition of point manipulations extends effortlessly to curves. A degree k curve, with control points (Q_i) and their motion (ΔQ_i), is subdivided into $(Q_i, \frac{1}{2}\Delta Q_i, i = 0, \ldots, k)$ and $(Q_i + \frac{1}{2}\Delta Q_i, \frac{1}{2}\Delta Q_i, i = 0, \ldots, k)$, as expected.

3. **Derivative Manipulations** $(C, D, S, \theta, \vec{R})$ In the derivative manipulation situation a frame $(D = (\vec{N}, \vec{B}, \vec{T}))$ of orthogonal normal, tangent and binormal vectors at a position (C) and with intended scaling, $S = (s_{\vec{N}}, s_{\vec{B}}, s_{\vec{T}})$, and rotation, θ about $\vec{R} = (r_{\vec{N}}, r_{\vec{B}}, r_{\vec{T}})$, that encode an affine transformation, $\mathbf{T}$, is subdivided into two derivative frame manipulations: $\mathbf{T_l} = (C, D, \sqrt{S}, \frac{1}{2}\theta$ about $\vec{R})$ and $\mathbf{T_r} = (C, \mathbf{T_l} \cdot D, \sqrt{S}, \frac{1}{2}\theta$ about $\vec{R})$. Conceptually the original derivative-frame transformation is achieved in two stages, each with half the original scaling and rotation.

At a particular depth (d) of recursive subdivision the current set of manipulations (M) undergo conic-hull hodograph testing. On failing (not ϕ-injective) the aforementioned splitting procedures are invoked and recursion addresses the two manipulation fragments (M_ℓ and M_r). On succeeding (ϕ-injective) the recursive subdivision is terminated and the relevant lattice changes $(\Delta\mathbf{P})$ passed back. 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, a sequence of lattice changes (L) is returned and these are applied in order, with the lattice being reset inbetween. The injective subdivision process is embodied in the following pseudo-code:

```
 InjectSubdiv(M, L, d)
(A recursive procedure which subdivides a set of direct
  manipulations (M) into injective steps and returns this
  as a sequence of lattice changes (L)

  IF d < maximum recursion depth THEN
    ΔP ← Pseudo-Inverse(cnst)
    IF InjectiveTest(ΔP) = 'injective' THEN
      Append ΔP to L
    ELSE
      HalfSplit(M, M_ℓ, M_r)
      InjectSubdiv(M_ℓ, L, d+1)
      InjectSubdiv(M_r, L, d+1)
  ELSE
    HALT and RETURN 'non-injective'
```

7.8 An Undo Operation

The ability to reverse a series of undesirable operations is central to the tenets of human-computer interaction through direct manipulation [Schneiderman 1983] and exploratory design [Hoffmann 1989, pp. 19-21]. Both creative freedom and design fluency will probably be stunted if deformation errors cannot be undone, since modellers are less likely to attempt risky innovations if they are forced to work with painstaking care.

Given a history of deformations leading, in sequence, to the final design state, an 'undo' operation can be implemented in one of three ways.

1. The deformation history can be re-applied, starting from the initial object but stopping short of its final state. This option is infeasibly slow if the deformation history is lengthy.

2. Another route is to retain copies of the object's recent topology and geometry. The 'undo' operation would involve backtracking to one of these copies. Unfortunately, the storage consumed by polygon-mesh objects is typically substantial. As a consequence, only a few object iterations can be held at any one time, thereby limiting the extent of reversibility. Hoffmann [1989, p. 19] mentions a combination of object caching and forward traversal which "checkpoints" an object's structure at regular intervals. If an object snapshot is taken once in every t interactive deformations (each consuming $< 0.1s$) then any instance in the deformation history can be reached within $0.1 \cdot t$ seconds.

3. The final alternative is to traverse backwards through deformation history by inverting FFD ($\mathcal{F}^{-1}(\widetilde{X}) = \mathcal{H}(\widetilde{\mathcal{H}}^{-1}(\widetilde{X})) = \mathcal{H}(U) = X$). In general, a continuous inverse, $\mathcal{F}^{-1}$, is only certain to exist if $\mathcal{F}$ is a homeomorphism satisfying theorem 7.1. This stipulation is always obeyed by injectively subdivided DMFFD.

Finding the inverse of a deformed object point, $\widetilde{V}$, is realized as follows. A closed form expression for the deformation function inverse ($\mathcal{D}^{-1} = \widetilde{\mathcal{H}}^{-1}$) does not exist and so must be computed through numerical search. The hyperpatch co-ordinates, $U_{\widetilde{V}}$, of the deformed point, $\widetilde{V}$, are evaluated through Newton-Raphson root finding [Press *et al.* 1992, pp. 379-383] on the equation $\widetilde{\mathcal{H}}(U_{\widetilde{V}}) - \widetilde{V} = 0$. Convergence from a heuristically selected initial position, $U_S = \mathcal{H}^{-1}(\widetilde{V})$, towards the solution $U_{\widetilde{V}}$, is quadratic. The solution can be bracketed by the maximum control point displacement ($max(\Delta\mathbf{P})$) in the (rare) case of divergent behaviour. Once the hyperpatch co-ordinates, $U_{\widetilde{V}}$, have been established, the embedding function supplies the pre-deformation object point directly ($V = \mathcal{H}(U_{\widetilde{V}})$). Obviously the inverse mapping is not as rapid as its forward counterpart but it is sufficient to support a reasonable 'undo' facility for any degree of backtracking.

7.9 Evaluation

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

[1] Type 1 and 2 deformations are constraint scaled versions of figures 7.8[A] 7.8[B] respectively.
S = $\frac{1}{2}$ length vectors, M = full length vectors, L = 1 $\frac{1}{2}$ length 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 7.8[A] and [B] are marked in blue .

Figure 7.6: **Cost of Injective Subdivision. Timings for a range of injective deformations applied to a polygon-mesh object at different resolutions.**

The various outcomes of injective subdivision are contrasted in figure 7.8 and 7.9 against standard DMFFD. Once injectively subdivided a particular, potentially self-intersecting, direct manipulation is either partitioned into injective fragments (figure 7.8[A]-[B], 7.9[B]) or designated as irrevocably self-intersecting (figure 7.9[A]). Injective DMFFD exhibits the following general behaviour:

- The results are guaranteed to be injective (non-self-intersecting) and continuously invertible (amenable to 'undoing') if classified as such.
- As evidenced by figure 7.8[A],[B] and 7.9[B] the behaviour of injective DMFFD is more intuitive and better mirrors the properties of highly elastic modelling clay. One problem with conventional DMFFD is overconstraint: in order to maintain continuity, certain portions of the hyperpatch may be distorted far more than the constraint points. Another difficulty is overextension: local control may be abused if constraints overreach their immediate area of influence. These problems are either significantly reduced or altogether eliminated by injective subdivision.
- In the timings of figure 7.6 the algorithm displays complex behaviour that is highly dependent on the magnitude and type of deformation. The performance degrades linearly

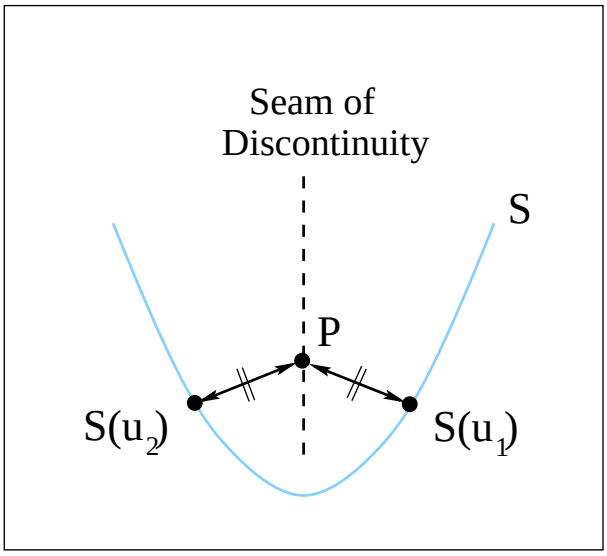


Figure 7.7: **Discontinuity in Shortest Distance Embedding. A seam of discontinuity along which sample points are equidistant from two locations on the source curve, $\mathcal{S}$.**

with an increase in object points and injective subdivisions and quadratically with the number of rows in the constraint matrix ($\Delta\mathbf{C}$). These factors are further coupled in a subtle fashion to the type of constraint (point, derivative or curve), conflict (overconstrained, overextended or crossing) and magnitude of constraint displacement. However, as long as reasonable bounds are placed on the object size, number of direct manipulation constraints and recursive subdivision levels, the technique remains suitable for interactive sculpting.

The techniques outlined in this chapter are focused on detecting and preventing self-intersection under DMFFD. However, some of the results, specifically the injectivity conditions of theorem 7.1 and the process of adaptively subdividing and then concatenating small injective deformations (section 7.7), are directly applicable to other, particularly point-based, spatial deformation methods. Unfortunately, distance parametrising curve-based approaches, such as Axial Deformation [Lazarus, Coquillart and Jancène 1994] and Wires [Singh and Fiume 1998], contain a fundamental flaw in this respect. These attach sample points (P) to the nearest point on a source curve ($\mathcal{S}$) and they consequently have (unless the source is straight) a seam of parametrisation discontinuity along the axis of a bend where the closest reference point jumps from one arm of the bend to the other (figure 7.7). This negates the first continuity condition of theorem 7.1. Fortunately, curve-directed FFD has no such drawback and as demonstrated in figure 7.9[B] is amenable to injective subdivision.

7.10 Conclusion

The principle contribution of this chapter is a robust variant of Directly Manipulated Free-Form Deformation which prevents self-intersection by concatenating 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 preventing self-intersection. The technique is suitable for interactive use, behaves with greater physical realism akin to manipulating highly malleable modelling 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 highly efficient. This test sacrifices the full range of injective deformations for improved performance and may falsely reject some valid deformations.

As mentioned in section 7.9, much of this research is more broadly applicable and, where feasible, extension to cover other types of spatial deformation would thus be an area of fruitful future research.

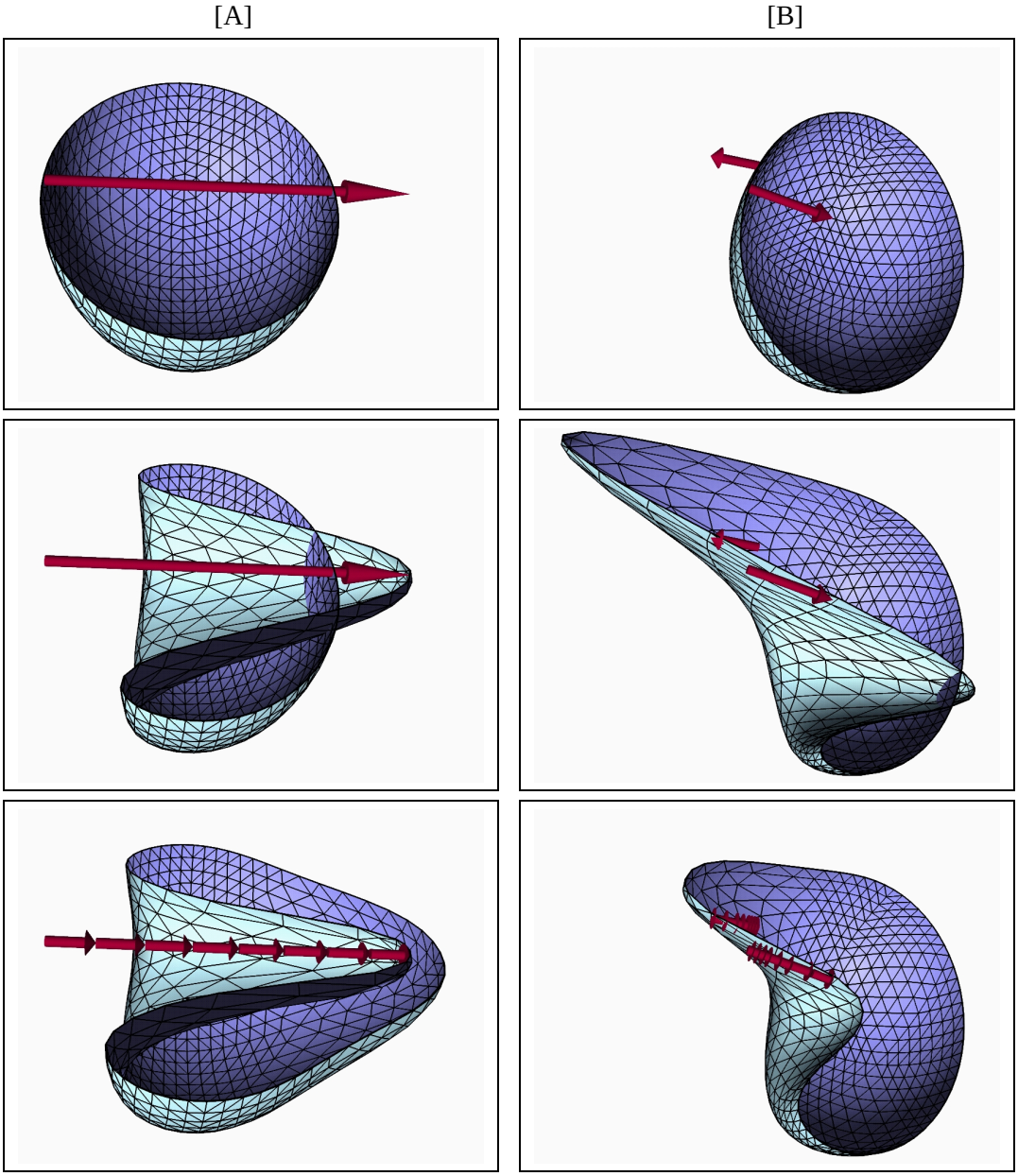


Figure 7.8: **Comparisons of Injective and Non-Injective DMFFD. [Top] Direct Manipulation constraints (red) on a squashed hemisphere (blue). [Middle] Conventional non-injective DMFFD applied to these constraints causes self-intersection. [Bottom] Injective DMFFD applied to adaptively subdivide these constraints prevents self-intersection. [A] Overextension. [B] Overconstraint.**

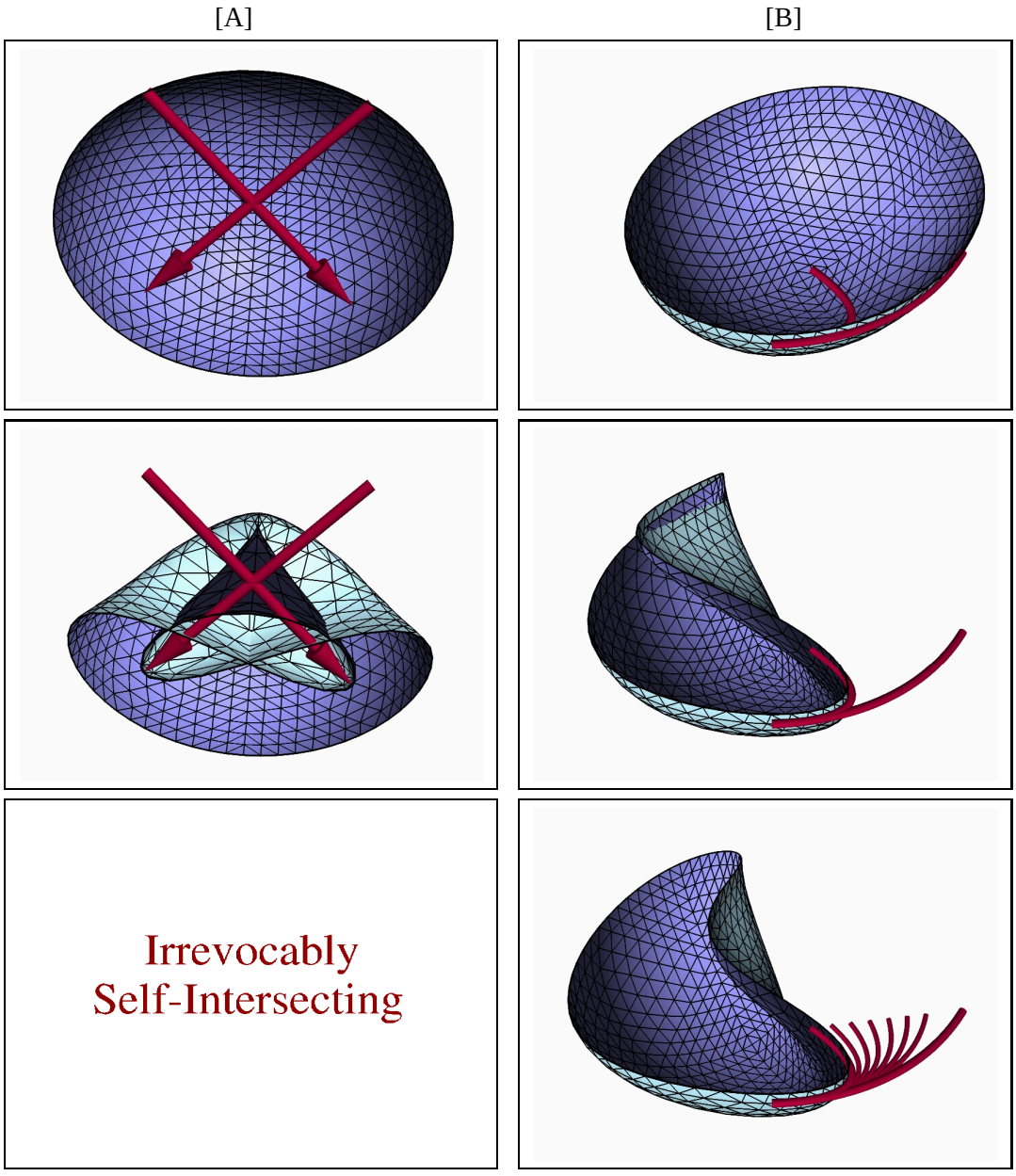


Figure 7.9: **Further Comparisons of Injective and Non-Injective DMFFD. [Top] Direct Manipulation constraints. [Middle] Conventional non-injective DMFFD. [Bottom] Results of Injective DMFFD. [A] Intersecting constraints cause irrevocable self-intersection. [B] A curve constraint.**

Chapter 8

Future Work

8.1 Introduction

Over the course of this thesis several incremental improvements to spatial deformation have been effected. Motivated by the wider context of virtual sculpting, enhancements have been made to the versatility (by enabling the manipulation of curves and derivative frames) and validity (through the prevention of self-intersection, and adaptive refinement and decimation) of Directly Manipulated Free-Form Deformation. Nevertheless, the spatial deformation approach to virtual sculpting is open to improvement in many areas. Taken as a whole, this thesis describes a cohesive sculpting system, which is useful as a testbed for future developments in versatility and ease of use.

Chapter Structure

In this chapter six worthwhile avenues for future research are outlined. The issues involved in constructing a virtual environment, which fosters a close mapping between computer-based (virtual) and physical (real) sculpting, are of particular significance. This has three aspects: developing software (a virtual tool-set), selecting hardware (user interface issues) and testing the resulting system (usability studies).

1. **A Virtual Toolset.** Individually specifying large numbers of direct manipulations can become time consuming and frustrating. Deformation tools, which automatically generate direct manipulations from contact points between the tool and deformable object, would alleviate these difficulties. We envisage tools which mimic real-world sculpting devices, such as throwing wheels, smoothing spatulas or a sculptor's hands.

2. **User Interface Issues.** This thesis has placed much emphasis on the mechanics of direct manipulation but little attention has been paid to designing an effective user interface. Sculpting is a highly interactive, three-dimensional task which would benefit from an interface with spatial position tracking, stereoscopy, multi-modal feedback and head-coupled display.

3. **Usability Studies.** Up to this point, developments in the ease of use of spatial deformation have been motivated by general and established tenets of human-computer interaction. More specific insight into current usability and future possibilities could be garnered from formal trials with end-users. Such subjects would fall between three extremes: modellers fluent in current design software, artists who sculpt purely in traditional materials, and users with neither form of experience.

4. **Multiresolution DMFFD.** A limitation of DMFFD is that the deformation boundary and its interior density around a direct manipulation are determined by the underlying hyperpatch. With a uniform B-spline basis all cells in the hyperpatch assume identical dimensions. Even a non-uniform basis only allows an uneven distribution along the hyperpatch axes. These limitations could be avoided by directly manipulating extended FFD [Coquillart 1990] or adopting a multiresolution B-spline basis [Mrose 1997].

5. **Preventing Transmission Effects.** Spatial deformation fails to make any distinction between the inside and outside of a deformable object so that, perhaps disconcertingly, warping can be transmitted across the external space between separated surfaces. In the interests of matching real-world sculpting this behaviour should be prevented.

6. **Topology Alteration.** The topological invariance property of spatial deformation (examined in section 2.1) disallows the creation and dissolution of holes as well as the separation and merging of component objects. Incorporating these operations would expand the diversity of possible shapes, and also mimic the perforation and rejoining of physical clay.

8.2 A Virtual Toolset

The direct manipulation of points, curves and derivative frames as a means of controlling spatial deformation is not an unqualified success. Designing a collection of direct manipulations contributing to a single deformation can be both time consuming and frustrating for the user who must "pick" the initial position of each manipulation and "drag" it into a final configuration. For instance, flattening part of an object might require 20-30 point manipulations with initial positions distributed across the surface and displacement vectors enforcing a final planar alignment.

One means of automatically generating larger manipulations might be through object-independent sculpting tools. There has been some research [Davis and Burton 1991; Kuriyama and Tachibana 1995] into designing tools for FFD. Hsu, Hughes and Kaufman

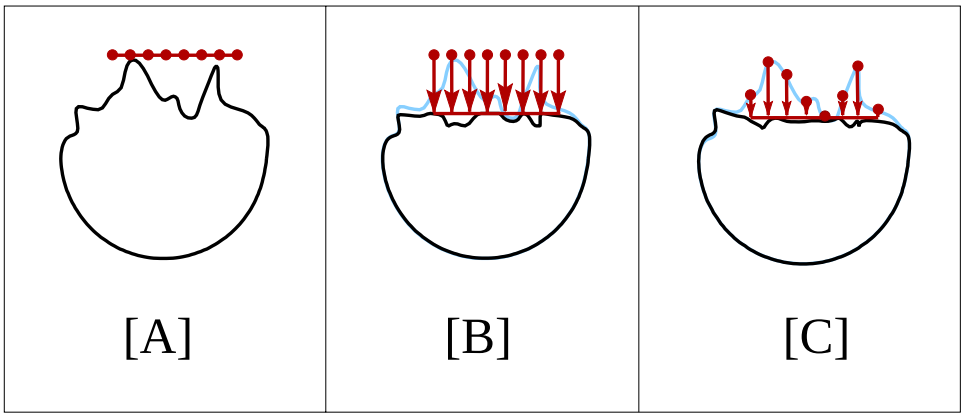


Figure 8.1: **Tool-Based Deformation. [A] A flattening tool (red) and ragged shape (black). [B] Direct point manipulations are derived from the tool's motion. [C] Alternatively, manipulations are only instigated when the tool and shape intersect.**

[1992] suggest that similar extensions are necessary for multiple point direct manipulation. We envisage each tool as an arrangement of points and curves. This construct would be positioned relative to a deformable object and then moved through a series of rotations, scalings and translations. At each stage, a set of direct manipulations could be derived from the tool's initial position and transformation. In the proximity of a deformable surface, the tool would behave like a repulsive field (figure 8.1[B]). If this indirect behaviour is undesirable then direct manipulations could be initiated only after contact between the surface and tool (figure 8.1[C]). The emphasis should be on supporting a clay sculpting analogy by mirroring real-world tools. From this perspective we outline a selection of possibilities:

- **Imprinting.** Altering sculpting tools purely through rigid body translations and rotations, so that the relative layout of the component points and curves remains consistent, allows a user to emulate the effects on clay of imprinting or extruding an immutable shape. For example, in figure 8.2, a letter-shaped tool is stamped into a deformable object, leaving a printed indentation. A given sculpting tool can be linked to a specific size of hyperpatch cell so that scaling relative to the tool's centre of mass also affects the cell dimensions. In this way global and local deformations can be defined in a natural manner by scaling-up or scaling-down the tool and consequently its associated lattice.

- **Volume Locking.** "Locking" or "freezing" portions of an object within a specified volume so as to prevent further deformations is a valuable capability. The difficulties are: (a) to ensure continuity across the "lock" boundary, and (b) to mediate when deformations intersect the locked volume. A static tool which establishes a wall of "pinned" (zero displacement) point manipulations around the locked volume would resolve these issues.

- **Symmetric Spins.** Throwing clay on a wheel is a technique used in pottery to create circular symmetric vessels such as pots and vases. This could be simulated in spatial deformation by gradually rotating a tool around a user-specified axis.

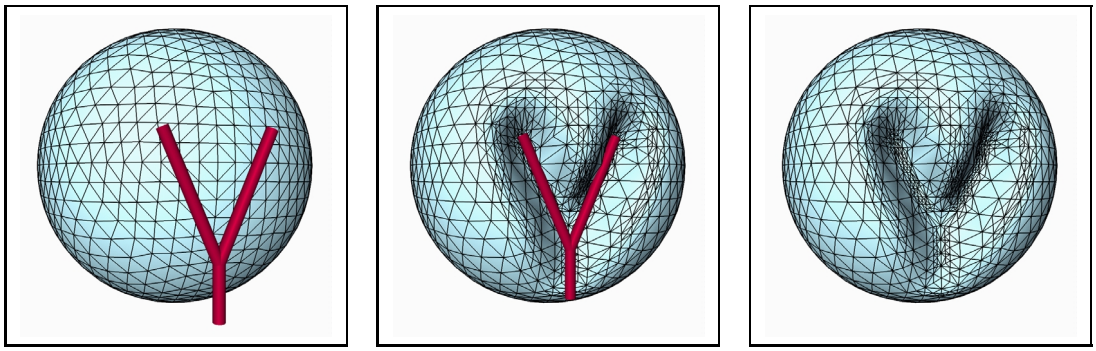


Figure 8.2: **Imprinting the Letter 'Y'. [Left] A spherical object and 'Y'-shaped tool, [middle] the tool is pushed into the object and [right] an imprint is retained.**

- **Collars.** Often a tendril-like extrusion or intrusion will either be too narrow or too wide in cross-section. A collaring tool, made up of a closed circular set of curves placed around or within the tendril, could be shrunk to pinch it inwards or expanded to widen it outwards.
- **Hand-Based Moulding.** Kneading, pinching and moulding modelling putty or clay is a familiar physical process extending back to childhood for many people. For this reason deformations which simulate the action of a user's hands and fingers (the tools) on clay (the modelled object) are ideal. As sculpting tools the hands are marvellously versatile. They are capable of dynamically altering as the fingers are flexed and also acting effectively in concert. To match this, the external reaction to other tools and the internal relationship between the constituent points and curves of a virtual hand must be adaptable. However, the true challenge lies in enabling a user to control the virtual hand in a straightforward and intuitive manner.

8.3 User Interface Issues

This thesis has focused on the interaction mechanism and subsequent behaviour of spatial deformation. Of equal importance in a complete virtual sculpting environment is the interface between the user and these spatial deformation techniques. A suitable user interface should match the input and output devices to the nature of the task [Preece *et al.* 1994, pp. 221-224] in order to facilitate rapid, intuitive and accurate design. The system response, as measured by the latency (or lag) between the user initiating an event and the results being displayed, is critical to 3D task performance [Arthur, Booth and Ware 1993], and should be degraded to the least possible extent by the choice of interface devices. There are four features which could effectively support the inherently three-dimensional nature of virtual sculpting.

1. **Three-Dimensional Tracking.** Controlling the motion of tools and deformable objects with 3D tracking devices has distinct advantages over conventional input (keyboard, mouse): the six degrees of motion freedom (position and orientation) map naturally to

the six degrees of tracker control. Polhemus trackers, which are typical of this type of hardware, link a magnetic transmitter to one or more sensors capable of determining their location and orientation relative to the transmitter.

2. **Stereoscopy.** A stereoscopic display enables binocular depth cues by presenting a separate image to each eye. This allows for finer discernment of an object's exact shape and position relative to any tools than perspective projection alone. There are two broad classes of stereoscopic displays: immersive devices (e.g. head-mounted displays), which totally replace the real view with computer-generated images, and fish-tank devices (e.g. shutterglasses), which often operate with modified workstation displays and provide a limited viewing area likened by Arthur, Booth and Ware [1993] to " *looking at a fish tank inside which there is a virtual world.*" Fish-tank devices are generally more suited to virtual sculpting as they can be used for longer periods without fatigue in a desktop environment and enable easy collaboration.

3. **Multi-Modality.** While sight is generally pre-eminent, other senses (hearing and touch) can contribute to the verisimilitude of virtual sculpting. Auditory feedback would be useful as a secondary cue to contact between a tool and object. More important is haptic feedback (touch), which is called upon in the control of fine motor tasks such as clay sculpting. In fact, a clay sculptor might conceivably rate tactile sensation as equal to sight in importance. Haptic devices which provide force feedback during virtual contact between a tool and object, although relatively crude compared to the tactile sensitivity involved in traditional sculpting, are nevertheless worthwhile. Brooks *et al.* [1990] report a two-fold performance improvement from the haptic augmentation of a molecular docking application and we expect a similar benefit for virtual sculpting. Several haptic sculpting systems have already been developed. These are either active [Chai, Luecke and Edwards 1998; Masse 1998], where the input device responds with mechanical counteracting forces, or inactive [Murakami and Nakajima 1994; Kameyama 1997], where pressure against a foam or rubber surface is detected but without providing mechanical response.

4. **Head-Coupled Display.** If a 3D tracker is attached to the user's head then the viewpoint for perspective and stereo transformations can be adjusted as the head shifts, allowing motion induced depth cues. Arthur, Booth and Ware [1993] report that head-coupled display is even more helpful than stereo in a 3D shape identification task but admit that rotating objects in the scene has the same benefits. Allowing the user to interactively control the axis and speed of rotation would be an effective means of achieving this in virtual sculpting.

We imagine a virtual sculpting system which supports two-handed interaction (perhaps with spatially-tracked haptic data-gloves for the left and right hands) and head-coupled fish-tank stereoscopy. Foot-based input devices such as the 'mole' (footmouse) [Preece *et al.* 1994, pp. 234-235] would permit simultaneous control over secondary parameters (e.g. area of effect) while the user's hands are otherwise occupied. In short, there is much to be gained from selectively matching the real world in a virtual sculpting environment.

8.4 Usability Studies

No formal experiments have been undertaken to evaluate the usability of the versatility and validity improvements made in this dissertation. Rather, our research has been motivated by established principles in human-computer interaction, particularly direct manipulation [Schneiderman 1983], and a desire to emulate some useful and familiar features of traditional sculpting. However, a comprehensive usability study of virtual sculpting with spatial deformation would be worthwhile. Preece *et al.* [1994, pp. 603-604] identify two categories of user-centred system evaluation: formative studies guide a user interface design by providing feedback, and summative studies allow judgements as to the success of a completed interface. Both apply in the case of virtual sculpting. Evaluating spatial deformation against other computer-based free-form design techniques and also traditional sculpting would enable both an assessment of its relative merits and point the way to future advancements.

At a low level, controlled laboratory experiments could be set up to compare the speed and accuracy of virtual sculpting with other design methods, or judge the effectiveness of different combinations of user interface mechanisms (position tracking, haptics, stereoscopy and head-coupled display) and spatial deformation techniques (point, derivative-frame, curve and tool interaction). For example, a sample of inexperienced users could each be trained for an equal period in traditional clay sculpting, virtual spatial deformation sculpting and the use of representative commercial modelling software. Their performance in recreating specific free-form shapes would then be analysed from a record of the time taken, visual or software logging of the type and frequency of deformation operations, and the accuracy with which the target was matched. However, such training experiments are time consuming and less suited to examining some fundamental issues, such as support for a creative design process (iterative refinement, backtracking and the exploration of alternatives), which features the users most favour or dislike, and formative, as opposed summative, judgements.

At a high level, the qualitative opinion of users can be elicited through structured interviews and questionnaires. It is more appropriate, for these forms of evaluation, to choose subjects with experience in traditional sculpting and/or modelling software, as their opinions are likely to be more informed.

8.5 Multiresolution DMFFD

There are two principal limitations which arise from the hyperpatch foundations of Directly Manipulated Free-Form Deformation:

1. The deformation boundary and internal density associated with a point, derivative-frame or curve manipulation are determined by the parallelepiped distribution of the pre-deformation lattice. Coupled with a uniform B-spline hyperpatch basis this dictates

that all cells are parallelepiped, have the same dimensions, and uniform density. A non-uniform basis allows the dimensions and density, but not the parallelepiped shape, to vary among cells. However, this variation is tied to the hyperpatch axes. For instance, the length of cells may change along the $\vec{U}$ axis but all cells in the same $\vec{V}$-$\vec{W}$ plane will have identical lengths. In short, the diversity of boundary shapes (beyond parallelepiped) and internal densities enabled by extended FFD [Coquillart 1990] and arbitrary lattice-topology deformations [MacCracken and Joy 1996], as examined in section 2.2.1, are not supported by DMFFD in its current formulation.

2. The region of influence of a manipulation, be it a point, derivative frame or curve, depends on its location in the hyperpatch and, more particularly, the dimensions of the cells within which it falls. Unlike with other point- and curve-based spatial deformation techniques, such as Scodef [Borrel and Rappoport 1994], DOGME [Bechmann 1994] and Wires [Singh and Fiume 1998], it is currently impossible under DMFFD to independently set the size, shape and density of the deformation region for individual manipulations.

The first limitation can be circumvented by making use of extended FFD [Coquillart 1990]. If the restriction of a linear parametrisation is lifted, then the lattice control points need not be configured as a regular parallelepiped, and variation in the boundary shape and internal parametrisation density of the pre-deformation hyperpatch cells is practicable. As a consequence, the embedding of sample and constraint points, $\mathcal{E}(X) = U$, necessitates an interactive search. With the exception of this embedding process and the initial lattice control point positions, P, the direct point manipulation process remains unaltered. In particular, the fundamental point constraint relationship (eqn. 3.10) still holds true:

$$C_i + \Delta C_i \quad = \quad \mathbf{B}_i(\mathcal{E}(C_i)) \cdot (\mathbf{P} + \Delta\mathbf{P}) \qquad \forall i = 1, \ldots, r$$

Adjusting curve-directed manipulation (chapter 6) and the automatic prevention of self-intersection (chapter 7) to compensate for an extended FFD hyperpatch is only slightly more involved. The greatest hurdle is likely to be finding a fast and intuitive means of configuring the pre-deformation lattice.

Mrose [1997] investigates the melding of B-splines and wavelets for multiresolution curve and surface design. Wavelets allow the compact and efficient representation of functions combining multiple scales. This is achieved through self-similar oscillatory functions generated by dilation and translation operations [Stollnitz, DeRose, Salesin 1996, pp. 11-16]. Wavelets exhibit, as do other successful multiresolution methods, both adaptation and filtering of details: changes made at a coarse level are reflected at all finer scales (higher resolutions) and conversely small-scale detail is transmitted through approximation to broader scales (lower resolutions). Mrose [1997, pp. 1-2] asserts that the main strength of multiresolution design is its support of iterative refinement. At first, only the rough shape of a curve or surface is specified but ever finer details are gradually incorporated, until the final design is achieved. For the purposes of DMFFD, a wavelet B-spline basis would permit different, but interacting, local lattice resolutions to be associated with individual manipulations. This would overcome the limiting interdependence of boundary regions among manipulations.

By combining extended lattice control point positioning and a wavelet B-spline basis, a multiresolution DMFFD technique can be developed which surmounts both of the aforementioned limitations in the current approach, but not without considerably reducing performance.

8.6 Preventing Transmission Effects

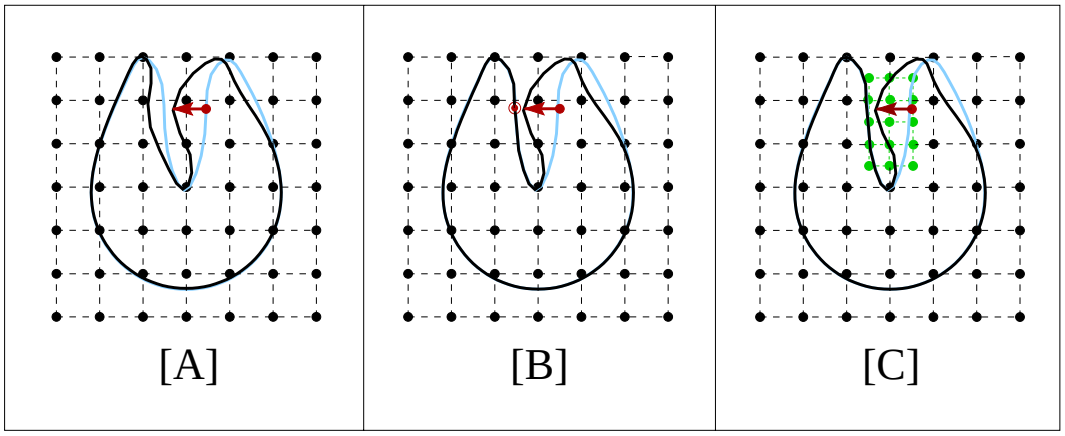


Figure 8.3: **Preventing Unwanted Spatial Transmission. [A] A point manipulation on one limb of an object affects another nearby. [B] A pin (static point manipulation) is inserted to block deformation. [C] Increasing the local lattice resolution contains the deformation.**

Current spatial deformation techniques do not distinguish between the exterior and interior of a deformable solid. Warping is transmitted equally inside and outside an object. This accords perfectly with the metaphor of distorting a lump of jelly containing an object of the same consistency, but may confound users' expectations of clay-like behaviour. By way of illustration, figure 8.3[A] shows how (perhaps disconcertingly) a deformation is spread between two extrusions separated by empty space. Ideally, there should be little interaction between separated surfaces unless they are brought into contact, in which case they should conform to avoid self-intersection, or merge through topology alteration. We forsee at least two ways of preventing unwanted transmission effects:

1. Static point manipulations with no displacement can be placed on the secondary surface in the path of the deformation. In figure 8.3[B], the transmission effect is limited by pinning the leftmost extrusion at a point closest to the deformation source. The challenge lies in automatically determining where and when to place such pinning manipulations.

2. Multiresolution DMFFD (proposed in section 8.5) would permit regions of higher lattice density to cushion against the transmission of deformation. In figure 8.3[C], a zone of small cells is interposed between the extrusions so as to localise the deformation. This approach is likely to carry substantial overheads.

8.7 Topology Alteration

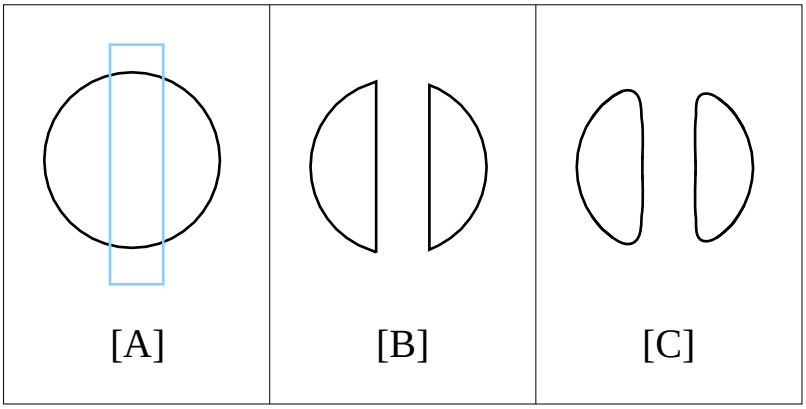


Figure 8.4: **Topology Alteration using Set Operations and Mesh Smoothing. [A] A deformable object (black) and overlaying rectangle (light blue). [B] The rectangle is subtracted leaving sharp corners. [C] Creasing is removed by local mesh subdivision.**

Sculptors often employ the friability of clay to pierce holes, tear away unwanted pieces and smoothly join separately sculpted components. In virtual sculpting, these topology altering operations expand the diversity of possible shapes and promote the reuse of models. A designer would no longer need to determine the number, and approximate size and distribution of holes prior to sculpting but could instead introduce them at an appropriate juncture in the creation process. Also, key shape features might be separated from an object or borrowed from a model library, duplicated as necessary, and attached appropriately. For example, in modelling an octopus, with the minimum of time and effort, the designer would create a single tentacle, replicate it seven times, and join all eight instances in different poses to the body of the octopod. Component separation, duplication and merging thus serve a similar function to "cut", "copy" and "paste" operations in two-dimensional sketching.

Unfortunately, spatial deformations are topologically invariant in that the number and relationship of edges, faces, and holes of an inset object remain constant. Aubert and Bechmann [1997b] work around this restriction by operating in a higher-dimensional space: a three-dimensional object is extruded into $\Re^4$, the resulting space-time construct undergoes a four-dimensional spatial deformation, and a cross-section of constant time is taken to produce a topology altered object. Section 2.2.2, and in particular figure 2.4, provide further details. Aubert and Bechmanns' space/time deformation has the advantage that it fits seamlessly within the spatial deformation paradigm but, due to the huge computational cost of four-dimensional extrusion, deformation and cross-sectioning, its applicability is limited to computer animation. Instead, we propose two topology alteration strategies more suited to interactive virtual sculpting:

1. The subtractive ($A - B$) and additive ($A \cup B$) set operations of constructive solid geometry (CSG) either remove material where one object, B, overlaps another, A, or combine

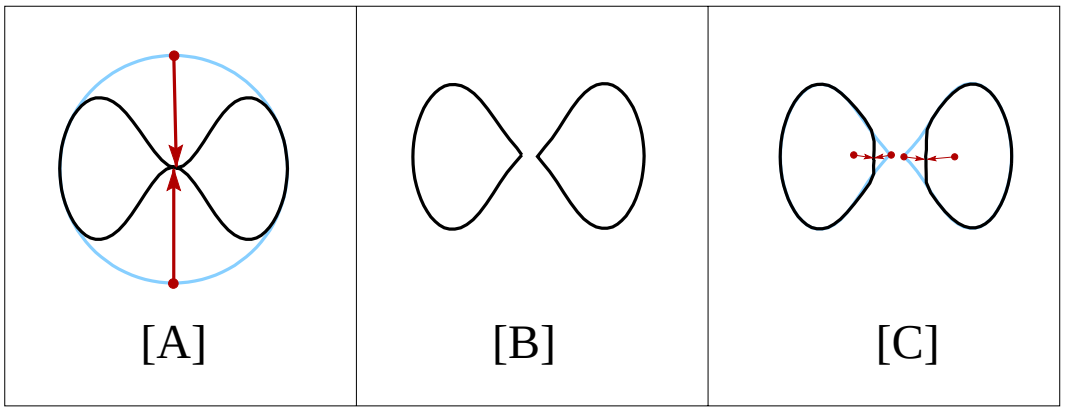


Figure 8.5: **Topology Alteration using Spatial Deformation. [A] Opposite sides of a circle are dragged towards a common point. [B] The mesh is restructured to cause component separation. [C] A set of pinching deformations removes any creases.**

the two solids, A and B, into a seamless whole. Along with intersection ($A \cap B$), these operations are a staple of most solid modelling systems [Mäntylä 1988, ch. 15; Hoffmann 1989, ch. 3]. Hole creation can be achieved by subtracting a cylinder from the object (object − cylinder), component separation by subtracting a disc (object − disc), and component merging through simple union (part $A \cup$ part B). Set operations on solids tend to introduce sharp creases at the intersection of the component surfaces. This crease effect could be allayed by a local smoothing subdivision [Peters and Reif 1997] of the surrounding surface, with particular care required to maintain C^1 continuity between the original and subdivided mesh regions. Figure 8.4 is a speculative example of how a subtractive set operation and mesh smoothing might be combined to bring about component separation.

2. Alternatively, topology alteration can be implemented by combining direct manipulation and mesh restructuring. Firstly, a singularity is formed by directly manipulating surfaces towards a common point. Secondly, the surface meshes are stitched together at the singularity to create a minute hole, separation or join. Finally, the new feature is widened and flattened by a set of expanding manipulations orthogonal to those employed in originally forming the singularity. Figure 8.5 demonstrates this process in action, with the component separation of a simple two-dimensional object. Particularly noteworthy are the two pinching manipulations, which, in the last step, smooth out the discontinuous peaks leading into the singularity.

Chapter 9

Conclusions

This dissertation has focused on improving the versatility and validity of spatial deformation in a virtual sculpting context. Incorporating elements from traditional sculpting into computer-based shape design has long been regarded [Parent 1977] as valuable. An artist's physical familiarity with moulding clay, wax or putty can then be transferred to an unfamiliar computer-based free-form modelling environment, benefitting ease of learning and later use. Spatial deformations, which act by warping the volume surrounding an embedded object, are a class of modelling techniques well suited to virtual sculpting. They are powerful (independent of any particular object representation, and variable in scope, allowing both global and local modifications); intuitive (deformations can be directly specified by displacing points or curves); and efficient (smooth sculpted results can be achieved interactively). Nevertheless, spatial deformation is not a panacea for the difficulties of virtual sculpting. Four areas of improvement in spatial deformation have been addressed by this thesis and six more were proposed for future investigation.

We began the body of this dissertation by categorising spatial deformations according to their interaction mechanism. Hyperpatch-based deformations, commonly referred to as Free-Form Deformations (FFD), enclose the object in a trivariate parametric volume, which can be altered by repositioning control points within a connected lattice. Point-based deformations enable a user to manipulate arbitrary points, and thereby instigate a conforming distortion of the neighbouring space and embedded object. Curve-based deformations interpolate the motion of a set of curves supplied by the user. We founded our research on Directly Manipulated Free-Form Deformation (DMFFD) because of its effective balance between ease of use, versatility and efficiency. DMFFD combines the hyperpatch- and point-based methods of spatial deformation in a two step process: lattice control points are altered to match point manipulations provided by the user and the hyperpatch distortions defined by these lattice changes are imparted through standard FFD to the object.

We have outlined a complete DMFFD framework by explaining:

- The mechanism of Free-Form Deformation as applied to sample points and their first derivative properties (tangents and normals).
- The conversion of user-defined point manipulations (each consisting of a position and intended displacement) into a system of constraint equations.
- How FFD lattice changes are matched to the direct manipulations through an efficient pseudo-inverse solution of the constraint system.

While DMFFD provides a useful development framework, we have endeavoured, wherever possible, to indicate how our research can be extended to other spatial deformation techniques.

The contributions made in this thesis fall into two categories:

1. **Versatility:** Conventional DMFFD supports the direct manipulation of multiple points with a 'pick' and 'drag' style of interaction. This is analogous to prodding and pulling at clay, using a set of slender rods capable of fastening to the surface with tiny pincers. Specifying even simple deformations can become frustratingly time-consuming if they involve many such point manipulations. We have made DMFFD more manageable by extending the interaction mechanism in two directions:

   (a) **Derivative-Frame Manipulation.** We have shown how the normal vector and tangent plane at a point, represented graphically by an arrow and planar rectangle, can be tilted, scaled and twisted to cause a corresponding distortion of the underlying space and associated object. Fowler [1992] developed a method for manipulating the derivatives of a bivariate parametric surface. We have extended this to trivariate hyperpatches so that a set of four linear constraint equations can be extracted from the transformation of a derivative frame. The resulting interaction mechanism is relatively simple and easy to use, combines seamlessly with point and curve manipulations, and gives rise to useful and predictable deformations.

   (b) **Curve Manipulation.** The curve-based spatial deformations (surveyed in section 2.2.3) function by tracking a number of curve manipulations, each comprising an explicit or implicit source curve, destination curve and enclosing volume of influence. Curve manipulation is a boon to ease of use and versatility. Many designers are familiar with controlling curves and, as geometric entities, their power lies in covering not only discrete points but also a strand of positions inbetween. We have devised a curve-directed DMFFD technique which maps source curves onto destination curves by contorting the underlying hyperpatch. Trivariate functional composition [DeRose *et al.* 1993; Nimscheck 1995, ch. 3] allows the Free-Form Deformation of a Bézier curve to be expressed as a resulting Bézier curve, albeit of substantially raised degree. Our contribution lies in linking curve control points through composition to hyperpatch control points, so that a system

of curve interpolating constraint equations can be derived. We have found that the degree elevation implicit in composition leads to unnecessarily large, ill-conditioned constraint systems, and so apply a sequence of Chebyshev economising degree reductions [Eck 1993]. The resulting systems are compact and solvable but prone to some approximation error in interpolating the intended destination curves. Fortunately, tight bounds can be placed on the magnitude of this error so that the fidelity of interactive sculpting is not damaged. Our approach has distinct advantages over previous curve-based spatial deformation techniques: all three types of spatial deformation are merged to enable the combined manipulation of points, derivative frames and curves; self-intersection can be automatically detected and prevented; by avoiding costly Euclidean distance parametrisation [Lazarus, Coquillart and Jancène 1994; Singh and Fiume 1998] interactive update rates can be achieved over the entire deformation cycle; and any desired degree of deformation continuity can be attained. None of the previous curve-based spatial deformation techniques can match these benefits. However, we discovered that generating point manipulations by sampling the source and destination curves, has the same advantages and achieves similar results to curve-directed DMFFD using composition and degree reduction. This curve sampling is somewhat faster but also introduces more error. In general, DMFFD with curve manipulation, either by sampling or composition, achieves, at the expense of absolute accuracy in tracking curves, considerably improved versatility.

2. **Validity:** Under certain circumstances spatial deformation impairs the physical realism and mathematical correctness of an embedded object. It is worth identifying these problems, cataloguing the situations in which they arise, and seeking preventative remedies. We address, in this thesis, two key validity issues:

   (a) **Adaptive Refinement and Decimation.** The most convenient representation for deformable objects is a triangle-element polygon mesh. Unfortunately, the approximation quality of a mesh frequently degrades under deformation, as previously flat (and sparsely covered) regions become sharply undulating. Conversely, highly curved areas may be warped to near planarity and be wastefully oversaturated with polygons. To combat these effects we have devised an adaptive mesh refinement and decimation scheme, which is independent of any particular approach to spatial deformation and builds on the work of Greissmair and Purgathofer [1989] and Nimscheck [1995]. Our contributions are as follows: incorporating an efficient, fully symmetrical decimation scheme that maintains the underlying topology; reducing the computation cost of the refinement/decimation trigger condition by testing the deviation between surface normals at edge endpoints (adapted from Nimscheck [1995]); catering for boundaries and creases in the manifold input mesh and hence supporting a broader class of models; and dealing with sampling problems caused by a disparity between the scope of a deformation and the size of individual mesh faces. The overall efficiency of our algorithm is such that almost any amount of decimation and a reasonable degree of refinement (< 1000 subdivisions) can be achieved in real time.

(b) **Preventing Self-Intersection.** A serious weakness common to all forms of spatial deformation is the potential for self-intersection of an object. This is highly counterintuitive, invalidates the popular manifold representation, compromises subsequent applications, notably texturing and rendering, and obstructs analytic 'undo' operations used to reverse deformations. At a fundamental level, a breakdown in the injectivity (one-to-one mapping) during spatial warping can lead to interpenetration of portions of the object's surface.

Our research in this area has lead to a new injective variant of DMFFD, which guards against self-intersection by subdividing point, curve and derivative-frame manipulations into short injective steps. This depends on three novel and independently useful developments. Firstly, a set of theoretical conditions for ensuring that spatial deformation constitutes a homeomorphic mapping, and hence cannot cause self-intersection. Secondly, an exact (necessary and sufficient) test of FFD injectivity, which is accurate to within machine precision but computationally costly. Thirdly, an approximate (merely sufficient) FFD injectivity test that relies on a geometric construction and is highly efficient. These advances are incorporated into injective DMFFD to produce a technique, which apart from preventing self-intersection, behaves with greater physical realism, expands the range of valid deformations, and is suitable for interactive use.

We have also suggested future research into: virtual sculpting tools, which would generate direct manipulations when contacted with the deformable object; user-interface issues, such as the choice of input/output devices appropriate to the three-dimensional interactive nature of virtual sculpting; usability studies as to the efficacy of various interaction methods and interface devices; multiresolution DMFFD, intended to overcome restrictions on the boundary shape and internal density of deformations enforced by regular hyperpatches; preventing transmission effects, which occur when warping bridges the external space between separated surfaces; and topology alteration, which would permit the separation and merging of components, and the creation of holes.

Together the four enhancements of spatial deformation made in this thesis constitute a virtual sculpting solution, which integrates the hyperpatch-, point- and curve-based forms of manipulation, automatically refines and decimates a deformed mesh as required, and guards effectively against self-intersection.

References

Allan, J., Wyvill, B. and Witten, I. [1989] "A Methodology for Direct Manipulation of Polygon Meshes", *New Advances in Computer Graphics (Proceedings of CG International '89)*, Springer Verlag, Tokyo, Jun. 1989, pp. 451-469.

Andersson, L., Peters, T. and Stewart, N. [1998] "Selfintersection of Composite Curves and Surfaces", *Computer Aided Geometric Design*, Vol. 15, No. 5, May 1998, pp. 507-527.

Arthur, K., Booth, K. and Ware, C. [1993] "Evaluating 3D Task Performance for Fish Tank Virtual Worlds", *ACM Transactions on Information Systems*, Vol. 11, No. 3, Jul. 1993, pp. 239-265.

Aubert, F. and Bechmann, D. [1997a] "Volume-Preserving Space Deformation", *Computers and Graphics*, Vol. 21, No. 5, Sep.-Oct. 1997, pp. 625-639.

Aubert, F. and Bechmann, D. [1997b] "Animation by Deformation of Space-Time Objects", *Computer Graphics Forum (Eurographics '97)*, Vol. 16, No. 3, Sep. 1997, pp. 57-66.

Baraff, D. and Witkin, A. [1998] "Large Steps in Cloth Simulation", *Computer Graphics (SIGGRAPH '98)*, Annual Conference Series, Publication of ACM SIGGRAPH, Jul. 1998, pp. 43-54.

Barr, A. [1984] "Global and Local Deformations of Solid Primitives", *Computer Graphics (SIGGRAPH '84)*, Vol. 18, No. 3, Jul. 1984, pp. 21-30.

Bartels, R., Beatty, J. and Barsky, B. [1983] *An Introduction to the Use of Splines in Computer Graphics*, report no. UCB/CSD/83/136, Computer Science Division, University of California, Berkeley, U.S.A., Aug. 1983.

Baum, D., Rushmeier, H. and Winget, J. [1989] "Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors", *Computer Graphics (SIGGRAPH '89)*, Vol. 23, No. 3, Jul. 1989, pp. 325-334.

Baumgart, B. [1972] "Winged-edge Polyhedron Representation", report no. STAN-CS-320, Computer Science Department, Stanford University, Palo Alto, U.S.A., 1972.

Bechmann, D. and Dubreuil, N. [1993] "Animation through Space and Time Based on a Space Deformation Model", *The Journal of Visualization and Computer Animation*, Vol. 4, No. 3, Jul.-Sep. 1993, pp. 165-184.

Bechmann, D. [1994] "Space Deformation Models Survey", *Computers and Graphics*, Vol. 18, No. 4, Jul. 1994, pp. 571-586.

Bechmann, D. and Dubreuil, N. [1995] "Order-controlled Free-Form Animation", *The Journal of Visualization and Computer Animation*, Vol. 6, No. 1, Jan.-Mar. 1995, pp. 11-32.

Bézier, P. [1978] "General distortion of an ensemble of biparametric patches", *Computer Aided Design*, Vol. 10, No. 2, Mar. 1978, pp. 116-120.

Bill, J. and Lodha, S. [1994] *Computer Sculpting of Polygonal Models using Virtual Tools*, report no. UCSC-CRL-94-27, Baskin Center for Computer Engineering and Information Sciences, University of California, Santa Cruz, U.S.A., Jul. 1994.

Borrel, P. and Bechmann, D. [1991] "Deformation of n-Dimensional Objects", *SMA '91: Proceedings of the First Symposium on Solid Modeling Foundations and CAD/CAM Applications*, ACM Press, Jun. 1991, pp. 351-370.

Borrel, P. and Rappoport, A. [1994] "Simple Constrained Deformations for Geometric Modeling and Interactive Design", *ACM Transactions on Graphics*, Vol. 13, No. 2, Apr. 1994, pp. 137-155.

Brewer, J. and Anderson, D. [1977] "Visual Interaction with Overhauser Curves and Surfaces", *Computer Graphics (SIGGRAPH '77)*, Vol. 11, No. 2, Jul. 1977, pp. 132-137.

Bro-Nielsen, M. and Cotin, S. [1996] "Real-time volumetric deformable models for surgery simulation using finite elements and condensation", *Computer Graphics Forum (Eurographics '96)*, Vol. 15, No. 3, Aug. 1996, pp. 57-66.

Brooks, P., Ouh-Young, M., Batter, J. and Kilpatrick, P. [1990] "Project GROPE - Haptic Displays for Scientific Visualization", *Computer Graphics (SIGGRAPH '90)*, Vol. 24, No. 4, Aug. 1990, pp. 177-185.

Brown, P. [1998] *Selective Mesh Refinement for Rendering*, PhD thesis, Computer Laboratory, University of Cambridge, U.K., Feb. 1998.

Catmull, E. and Clark, J. [1978] "Recursively generated B-Spline surfaces on arbitrary topological meshes", *Computer-Aided Design*, Vol. 10, No. 6, Nov. 1978, pp. 350-355.

Chadwick, J., Haumann, D. and Parent, R. [1989] "Layered Construction for Deformable Animated Characters", *Computer Graphics (SIGGRAPH '89)*, Vol. 23, No. 3, Jul. 1989, pp. 243-252.

Chai, Y., Luecke, G. and Edwards, J. [1998] "Virtual Clay Modeling Using the ISU Exoskeleton", *IEEE Virtual Reality Annual International Symposium (VRAIS '98)*, Annual Conference Series, Publication of the IEEE, Jul. 1998, pp. 76-80.

Chang, Y-K. and Rockwood, A. [1994] "A Generalized de Casteljau Approach to 3D Free–Form Deformation", *Computer Graphics (SIGGRAPH '94)*, Annual Conference Series, Publication of ACM SIGGRAPH, Jul. 1994, pp. 257-260.

Clapham, C. [1996] *Concise Dictionary of Mathematics* [2nd edition], Oxford University Press, Oxford, 1996.

Comninos, P. [1989] "Fast Bends or Fast Free-Form Deformation of Polyhedral Data", *Computer Graphics '89*, Blenheim Online, Middlesex, Nov. 1989, pp. 225-242.

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

Coquillart, S. and Jancéne, P. [1991] "Animated free-form deformation: An interactive animation technique", *Computer Graphics (SIGGRAPH '91)*, Vol. 25, No. 4, Jul. 1991, pp. 23-26.

Dachille, F., Qin, H., Kaufman, A. and El-Sana, J. [1999] "Haptic Sculpting of Dynamic Surfaces", *1999 Symposium on Interactive 3D Graphics*, Publication of ACM SIGGRAPH, Apr. 1999, pp. 103-110.

Dahmen, W. [1986] "Subdivision algorithms converge quadratically", *Journal of Computational and Applied Mathematics*, Vol. 16, No. 2, Oct. 1986, pp. 145-158.

Davis, O. and Burton, R. [1991] "Free-Form Deformation as an Interactive Modeling Tool", *Journal of Imaging Technology*, Vol. 17, No. 4, Aug. 1991, pp. 181-187.

DeRose, T. [1988] "Composing Bézier Simplexes", *ACM Transactions on Graphics*, Vol. 7, No. 3, Jul. 1988, pp. 198-221.

DeRose, T., Goldman, R., Hagen, H. and Mann, S. [1993] "Functional Composition Algorithms Via Blossoming", *ACM Transactions on Graphics*, Vol. 12, No. 2, Apr. 1993, pp. 113-135.

DeRose, T., Kass, M. and Truong, T. [1998] "Subdivision Surfaces in Character Animation", *Computer Graphics (SIGGRAPH '98)*, Annual Conference Series, Publication of ACM SIGGRAPH, Jul. 1998, pp. 85-94.

Doo, D. and Sabin, M. "Behaviour of recursive division surfaces near extraordinary points", *Computer-Aided Design*, Vol. 10, No. 6, Nov. 1978, pp. 356-360.

Duncan, J., Martin, K. and Vaz, M. "Heroes' Journey", *Cinefex*, No. 78, Jul. 1999, pp. 76-145.

Eck, M. [1993] "Degree reduction of Bézier curves", *Computer Aided Geometric Design*, Vol. 10, No. 3-4, Aug. 1993, pp. 237-251.

Erikson, C. and Manocha, D. [1999] "GAPS: General and Automatic Polygonal Simplification", *1999 Symposium on Interactive 3D Graphics*, Publication of ACM SIGGRAPH, Apr. 1999, pp. 79-88.

Faloutsos, P., Van de Panne, M. and Terzopoulos, D. [1997] "Dynamic Free-Form Deformations for Animation Synthesis", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 3, No. 3, Jul.-Sep. 1997, pp. 201-214.

Farin, G. [1997] *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide* [4th edition], Academic Press, San Diego, 1997.

Farouki, R. [1991] "On the stability of transformations between power and Bernstein polynomial forms", *Computer Aided Geometric Design*, Vol. 8, No. 1, Feb. 1991, pp. 29-36.

Foley, J., van Dam, A., Feiner, S. and Hughes, J. [1991] *Computer Graphics: Principles and Practice* [2nd edition], Addison-Wesley, New York, 1991.

Fowler, B. [1992] "Geometric Manipulation of Tensor Product Surfaces", *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, Publication of ACM SIGGRAPH, Mar. 1992, pp. 101-108.

Gain, J. [1996] *Virtual Sculpting: An Investigation of Directly Manipulated Free-Form Deformation in a Virtual Environment*, MSc thesis, Department of Computer Science, Rhodes University, South Africa, Feb. 1996.

Gain, J. and Dodgson, N. [1999a] "Adaptive Refinement and Decimation under Free-Form Deformation" Proceedings of Eurographics UK '99, Cambridge, U.K., Apr. 1999, pp. 7-18.

Gain, J. and Dodgson, N. [1999b] "Enhancing the Efficiency and Versatility of Directly Manipulated Free-Form Deformation", *Technical Sketch, Conference Abstracts and Applications, SIGGRAPH '99*, Aug. 1999, p. 240.

Galyean, T. and Hughes, J. [1991] "Sculpting: An Interactive Volumetric Modeling Technique", *Computer Graphics (SIGGRAPH '91)*, Vol. 25, No. 4, Jul. 1991, pp. 267-274.

Garland, M. and Heckbert, P. [1997] "Surface Simplification Using Quadric Error Metrics", *Computer Graphics (SIGGRAPH '97)*, Annual Conference Series, Publication of ACM SIGGRAPH, Aug. 1997, pp. 209-216.

Georgiades, P. and Greenberg, D. [1992] "Locally Manipulating the Geometry of Curved Surfaces", *IEEE Computer Graphics and Applications*, Vol. 12, No. 1, Jan. 1992, pp. 54-64.

Glassner, A. [1990] *Graphics Gems*, Academic Press, Ltd., London, 1990.

Golub, G. and van Loan, C. [1989] *Matrix Computations* [2nd edition], John Hopkins University Press, Baltimore, Maryland, 1989.

Goodman, T. and Unsworth, K. [1994] "Injective Bivariate Maps", report no. CS 94/02, Department of Mathematics and Computer Science, University of Dundee, U.K., Dec. 1994.

Green, T. [1989] "Cognitive dimensions of notations", *People and Computers V*, Cambridge University Press, Cambridge, pp. 443-460.

Greissmair, J. and Purgathofer, W. [1989] "Deformation of Solids with Trivariate B-Splines", *Proceedings of Eurographics '89,* Eurographics Association, Sep. 1989, pp. 137–148.

Greville, T. [1960] "Some Applications of the Pseudoinverse of a Matrix", *SIAM Review,* Vol. 2, No. 1, Jan. 1960, pp. 15-22.

Grimaldi, R. [1989] *Discrete and Combinatorial Mathematics: an Applied Introduction* [2nd edition], Addison-Wesley, Reading, Mass., 1989.

Güdükbay, U. and Özgüç [1990] "Free-Form Solid Modeling using Deformations", *Computers and Graphics,* Vol. 14, No. 3/4, Jul.-Dec. 1990, pp. 491-500.

Hardwick, M., Spooner, D., Rando, T. and Morris, K. [1996] "Sharing Manufacturing Information in Virtual Enterprises", *Communications of the ACM*, Vol. 39, No. 2, Feb. 1996, pp. 46-54.

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

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W. [1993] "Mesh Optimization", *Computer Graphics (SIGGRAPH '93),* Annual Conference Series, Publication of ACM SIGGRAPH, Aug. 1993, pp. 19-26.

Hoscheck, J. and Lasser, D. [1993] *Fundamentals of Computer Aided Geometric Design,* A.K. Peters Ltd., Wellesley, Mass., 1993.

Hsu, W., Hughes, J. and Kaufman, H. [1992] "Direct manipulation of free-form deformations", *Computer Graphics (SIGGRAPH '92),* Vol. 26, No. 2, Jul. 1992, pp. 177-184.

James, D. and Pai, D. [1999] "ArtDefo: Accurate Real Time Deformable Objects", *Computer Graphics (SIGGRAPH '99),* Annual Conference Series, Publication of ACM SIGGRAPH, Aug. 1999, pp. 65-72.

Kalra, P., Mangili, A., Magnenat Thalmann, N. and Thalmann, D. [1992] "Simulation of Facial Muscle Actions Based on Rational Free Form Deformations", *Computer Graphics Forum (Eurographics '92),* Vol. 11, No. 3, Sep. 1992, pp. 59-69.

Kameyama, K. [1997] "Virtual Clay Modeling System", *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '97)*, Annual Conference Series, Publication of the ACM, Sep. 1997, pp. 197-200.

Kim, D-S. [1993] "Hodograph approach to geometric characterization of parametric cubic curves", Computer-Aided Design, Vol. 25, No. 10, Oct. 1993, pp. 644-654.

Klein, R., Liebich, G. and Straßer, W. [1996] "Mesh Reduction with Error Control", *Proceedings of IEEE Visualization '96*, IEEE Computer Society, Nov. 1996, pp. 311-318.

Klok, F. [1986] "Two moving co-ordinate frames for sweeping along a 3D trajectory", *Computer Aided Geometric Design*, Vol. 13, 1986, pp. 217-229.

Kuriyama, S. and Tachibana, K. [1995] "Interactive 3D Widgets for Free-Form Deformation", *Pacific Graphics '95*, Aug. 1995, pp. 320-329.

Lamousin, H. and Waggenspack, W. [1994] "NURBS-Based Free-Form Deformations", *IEEE Computer Graphics and Applications*, Vol. 14, No. 6, Nov. 1994, pp. 59-65.

Lawson, C. and Hanson, R. [1974] *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliff, New Jersey, 1974.

Lazarus, F., Coquillart, S. and Jancène, P. [1994] "Axial deformations: an intuitive deformation technique", *Computer-Aided Design*, Vol. 26, No. 8, Aug. 1994, pp. 607-613.

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

Mäntylä, M. [1988] *An Introduction to Solid Modeling*, Computer Science Press, Rockville, Maryland, 1988.

Martin, K. [1998] "The Sound and the Fury", *Cinefex*, No. 74, Jul. 1998, pp. 84-106.

Masse, T. [1998] "A Tangible Goal for 3D Modeling", *IEEE Computer Graphics and Applications*, Vol. 18, No. 3, May-Jun. 1998, pp. 62-65.

Meisters, G. and Olech, C. [1963] "Locally One-to-one Mappings and a Classical Theorem on the Schlicht Functions", *Duke Mathematical Journal*, Vol. 30, No. 1, Mar. 1963, pp. 63-80.

Molich, R. and Nielsen, J. [1990] "Improving a human-computer dialogue", *Communications of the ACM*, Vol. 33, No. 3, Mar. 1990, pp. 338-348.

Mørken, K. [1991] "Some Identities for Products and Degree Raising of Splines", *Constructive Approximation*, Vol. 7, No. 2, Apr.-Jun. 1991, pp. 195-208.

Mrose, S. [1997] *Multiresolution Curve and Surface Design: Theory and Application of Spline Wavelets*, PhD Thesis, Computer Laboratory, University of Cambridge, U.K., Jul. 1997.

Munem, M. and Foulis, D. [1984] *Calculus with Analytic Geometry* [2nd edition], Worth Publishers, New York, 1984.

Murakami, T. and Nakajima, N. [1994] "Direct and Intuitive Input Device for 3-D Shape Deformation", *Human Factors in Computing Systems (CHI '94)* Annual Conference Series, Publication of the ACM, Apr. 1994, pp. 465-470.

Naylor, B. [1990] "Sculpt - An Interactive Solid Modeling Tool", *Proceedings of Graphics Interface '90*, Morgan Kaufmann Publishers, May 1990, pp. 138-148.

Nielsen, J. [1994] "Enhancing the explanatory power of usability heuristics", *Proceedings of ACM CHI '94*, ACM Press, Apr. 1994, pp. 152-158.

Nimscheck, U. [1995] *Rendering for Free-Form Deformations*, PhD thesis, report no. TR381, Computer Laboratory, University of Cambridge, U.K., Oct. 1995.

Parent, R. [1977] "A System for Sculpting 3-D Data", *Computer Graphics (SIGGRAPH '77)*, Vol. 11, No. 2, Jul. 1977, pp. 138-147.

Parker, S. [1997] *McGraw-Hill Dictionary of Mathematics*, McGraw-Hill Book Company, New York, 1997.

Parry, S. [1986] *Free-Form Deformations in a Constructive Solid Geometry Modeling System*, PhD thesis, Department of Civil Engineering, Brigham Young University, Utah, U.S.A., Apr. 1986.

Peters, J. and Reif, U. [1997] "The Simplest Subdivision Scheme for Smoothing Polyhedra", *ACM Transactions on Graphics*, Vol. 16, No. 4, Oct. 1997, pp. 420-431.

Peters, G. and Wilkinson, J. [1970] "The least squares problem and pseudo-inverses", *The Computer Journal*, Vol. 13, No. 3, Aug. 1970, pp. 309-316.

Popović, J. and Hoppe, H. [1997] "Progressive Simplicial Complexes", *Computer Graphics (SIGGRAPH '97)*, Annual Conference Series, Aug. 1997, pp. 217-224.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. and Carey, T. [1994] *Human-Computer Interaction*, Addison-Wesley Longman Limited, Harlow, Essex, 1994.

Press, W., Teukolsky, S., Vetterling, W. and Flannery, B. [1992] *Numerical Recipes in C: The Art of Scientific Computing* [2nd edition], Cambridge University Press, Cambridge, 1992.

Ronfard, R. and Rossignac, J. [1996] "Full-range approximation of triangulated polyhedra", *Computer Graphics Forum (EUROGRAPHICS '96)*, Vol. 15, No. 3, Aug. 1996, pp. 67-76.

Rossignac, J. [1994] "Introduction to the Special Issue on Interactive Sculpting", *ACM Transactions on Graphics*, Vol. 13, No. 2, Apr. 1994, pp. 101-102.

Ruprecht, D., Nagel, R. and Müller, H. [1995] "Spatial Free-Form Deformation with Scattered Data Interpolation Methods", *Computers and Graphics*, Vol. 19, No. 1, Jan.-Feb. 1995, pp. 63-71.

Schneiderman, B. [1983] "Direct Manipulation: A Step Beyond Programming Languages", *IEEE Computer*, Vol. 16, No. 8, Aug. 1983, pp. 57-69.

Schroeder, W., Zarge, J. and Lorensen, W. [1992] "Decimation of Triangle Meshes", *Computer Graphics (SIGGRAPH '92)*, Vol. 26, No. 2, Jul. 1992, pp. 65-70.

Schumaker, L. [1993] "Triangulations in CAGD", *IEEE Computer Graphics and Applications*, Vol. 13, No. 1, Jan. 1993, pp. 47-52.

Sederberg, T. [1983] "Implicit and Parametric Curves and Surfaces for Computer Aided Geometric Design", PhD thesis, Purdue University, West Lafayette, Indiana, 1983.

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

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

Seidel, H-P. [1993] "An Introduction to Polar Forms", *IEEE Computer Graphics and Applications*, Vol. 13, No. 1, Jan. 1993, pp. 38-46.

Singh, K. and Fiume, E. [1998] "Wires: A Geometric Deformation Technique," *Computer Graphics (SIGGRAPH '98)*, Annual Conference Series, Publication of ACM SIGGRAPH, Jul. 1998, pp. 405-414.

Spain, B. [1960] *Tensor Calculus* [3rd edition], Oliver and Boyd, Edinburgh, 1960.

Stollnitz, E., DeRose, T. and Salesin, D. [1996] *Wavelets for Computer Graphics: Theory and Applications*, Morgan Kaufmann Publishers, San Francisco, California, 1996.

Terzopoulos, D. and Fleischer, K. [1988] "Modeling Inelastic Deformation: Viscoelasticity, Plasticity and Fracture", *Computer Graphics (SIGGRAPH '88)*, Vol. 22, No. 4, Aug. 1988, pp. 269-278.

Vaz, M. [1999] "A Bug's Life: an Entomological Epic" *Cinefex*, No. 76, Jan. 1999, pp. 41-50, 133-140.

Wang, S. and Kaufman, A. [1995] "Volume Sculpting", *1995 Symposium on Interactive 3D Graphics*, ACM Press, Apr. 1995, pp. 151-156.

Watt, A. [1989] *Fundamentals of Three-Dimensional Computer Graphics*, Addison-Wesley, Wokingham, England, 1989.

Wernecke, J. [1994] *The Inventor Mentor: Programming Object Oriented 3D Graphics with Open Inventor, Release 2*, Addison-Wesley, Reading, Mass., 1994.