ELSEVIER

Computer-Aided Design 33 (2001) 389–402

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Surface design using hand motion with smoothing

J. Dorman[a], A. Rockwood[b,*]

[a]*Silicon Graphics Inc., USA*

[b]*Mitsubishi Electric Research Laboratories, 18375 Highland Estates Drive, Colorado Springs, CO 80908, USA*

Abstract

We describe elements used in a system to manipulate a polygon-based surface object using a data glove (made by Fifth Dimension Inc.). A direct-manipulation method is used that allows for placing new displacements on previous ones. Good results are obtained for manipulation at any incidence angle to the surface by using a new linear transformation on the selected vertices, in order to properly calculate their displacement magnitudes. The design of the manipulation system emphasizes intuitiveness and speed.

A method for smoothing selected portions of the object is also presented. The smoothing tool does not change regions of the surface where vertex coordinates may be expressed as a second-order implicit function i.e. the method has quadratic precision. It operates by finding a least-squares fit, followed by a relaxation phase where vertices are moved near the surface approximation. The smoother operates on a set of disjoint points in space, no knowledge of connectivity is required by the surface smoother.

1. Motivation and background

The act of creating 3D objects interactively suggests an intuitive hand-based interface and a modeling environment that is robust, predictable, and fast. There is no commercially produced sculpting package that is easy enough for a novice to use immediately. These other packages require constant menu selection and mode changing. Often the interaction involves only keyboard/mouse input. A characteristic of directly manipulating a faceted object is that it often becomes jagged. Gouraud shading masks this feature in a rendering, by forming an interpolant across the triangle mesh. We provide a smoothing method to reduce these geometric discontinuities with quadratic precision.

1.1. Overview

We have developed a simple intuitive computer graphics sculpting system that incorporates glove input. This user interface provides direct control of the part of the object to be manipulated; it visually cues the user to the exact points on the object that will respond to the glove motion. The system is designed for crafting objects of an artistic or free-form nature. Our system is conducive to 3D back-of-the-napkin design. With little training, users were able to produce rough forms, in a broad spectrum of styles, in a matter of minutes.

For many years computer-assisted 3D modeling has been used in some fields, such as shipbuilding, car-design, and entertainment. With the low cost and ubiquity of computer hardware, plus the cultural acceptance of computers, 3D modeling should be made available to many more individuals.

One use of such a modeling system would be for studying 3D eye–hand coordination in child development. Recent studies distribute real clay balls to children to mold; these are later analysed for development trends. A virtual world would provide efficiency, provide numerical classification, and allow quantitative comparisons such as by placing one piece inside another and doing Boolean operations on them.

Another example comes from the medical community. Medical experts often need assistance from application experts when it comes to modeling. It would be efficient if a physician could work independently to design, or artistically craft, familiar geometry. A vascular surgeon could use empirical knowledge to add plaque to a carotid artery, starting with either a typical baseline artery or using scanning equipment to acquire the carotid of a specific individual. The manipulated form could then be built with a rapid prototyping machine for tactile use in a classroom, or for real and virtual flow studies to gain further general knowledge or to determine whether there exists a need for surgery. Validation studies should be used for tuning the simulation to account for simulation deficiencies such as faceted wall geometry. The important contribution here is in allowing experts with real-world experience to be directly in control of the sculpting process.

* Corresponding author. Tel.: +1-719-495-1247.
E-mail address: rockwood@enuxsa.eas.asu.edu (A. Rockwood).

0010-4485/01/$ - see front matter
PII: S0010-4485(00)00130-5

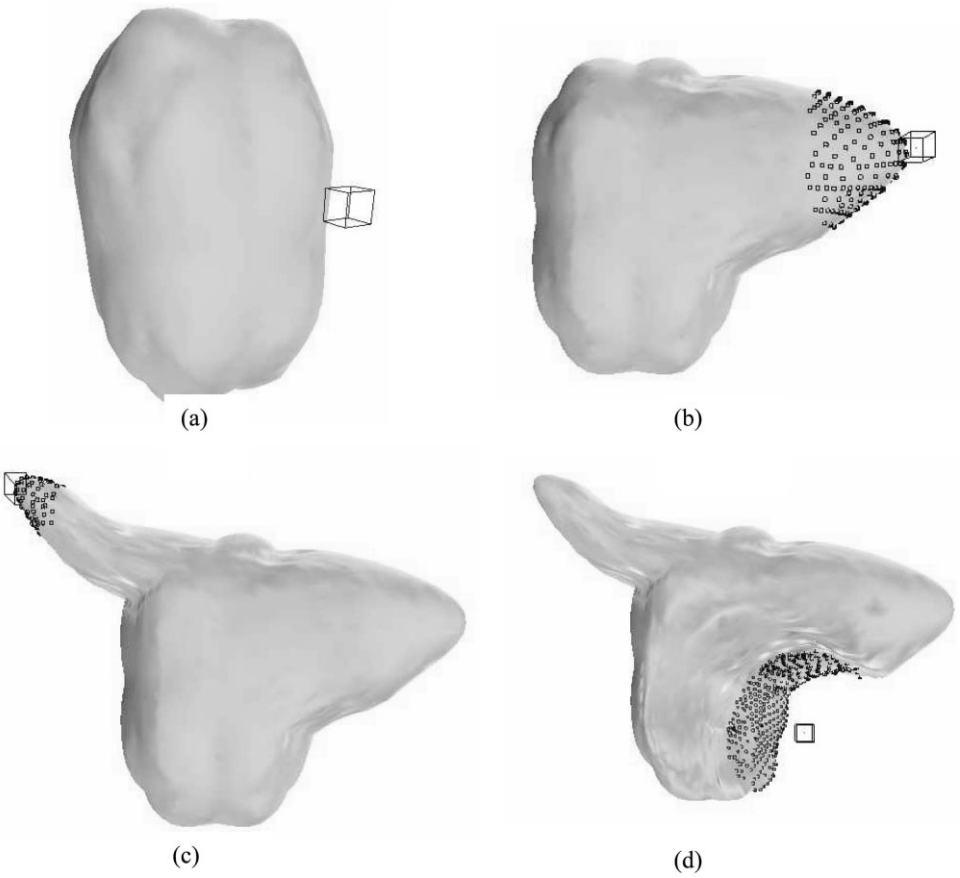


Fig. 1. (a) The default initial object, (b) the object is pulled, (c) a smaller amount is grabbed and pulled, (d) the object is pushed. The hand's position is shown as a cube, the portion of material grabbed (the vertices affected), are highlighted in black.

Artists are naturally accustomed to using their hands to touch their work during 3D design; they develop hand related artistic techniques and physical skills. Our system's use of eye–hand coordinated motion to control the design process accommodates these artistic talents. This leads to a greater acceptance of the technology, increasing the desire for artists to become involved in computer aided projects such as sculpting a dragon's head, or quickly filling a scene with many 3D forms for the movie industry.

1.2. Previous work

Alciatore and Wohlers [1] describe free-form stretching of a mesh as a manipulation paradigm. They use a cubic distribution for spreading out the discrete input displacement vector. They outline a contact detection method based on vertex connectivity. The method identifies a primary vertex closest to the tail of the discrete displacement vector. This primary vertex is always displaced. Other vertices to be displaced are found by expanding outwardly from this primary vertex using vertex connectivity information. These are candidate vertices that are then tested for proximity to the displacement vector using an Euclidean distance calculation.

Bryson [3] uses a spatially-weighted transformation to directly manipulate vertices in space. He introduces several paradigms for direct manipulation. We provide an extension to the pushing paradigm. Our contribution allows creating forms having an infinite number of bumps placed one on top of another, and so is not limited to only placing isolated bump displacements onto the surface. We accomplish this by mapping each vertex so it may be handled as if the surface were flat. Fowler and Bartels [9] also provide a framework to deform smooth bumps. Borrel and Rappoport [2] produced similar work that is presented in a more rigorous mathematical abstraction. Some other free-form direct manipulation techniques have been presented by Hsu et al. [10], also by Yamashita and Fukui [15].

1.3. An example session from our system

Fig. 1(a) shows the rendering of a triangular mesh object. This virtual object exists in the virtual space in front of the user. The user wears a data glove whose virtual space position is shown as a wireframe cube along with the object on the computer monitor. By opening and closing fingers in the glove, the user controls how large a proportion of the object near the virtual glove is affected by subsequent hand movements. The user may then distend the surface by moving the data glove in front of him [Fig. 1(b)]. When the glove is

opened wide, the chunk is released. The user may then grab a smaller chunk of material and pull that [Fig. 1(c)]. Alternatively we may push instead of pull [Fig. 1(d)]. The amount of material grabbed may be changed dynamically during the push, or pull. Self-intersections are allowed and controlled by the user.

2. Basic technique

Our focus is on maintaining an intuitive interface and a fast frame update rate. We use a polygonal model and an inference-based interface that is fast, but inappropriate for engineering parts that have to meet specific requirements of shape, dimensions, strength, etc. We desire to put together an enjoyable modeling package that allows abstraction, and intuitive expression through natural metaphors. The system has unique interactive characteristics; existing language must be used to communicate its feel, and so we say it is similar to sculpting a clay-like material. Sometimes this is inadequate, e.g. the material changes its volume when sculpted. One must balance novelty with consideration for a user's prior experiences so that the system will be intuitive.

2.1. Construction of initial surface

The start of our design session uses an initial object consisting of a polyhedral mesh formed by sampling a closed spline-surface object. The object mesh represents a piece-wise linear reconstruction of the continuous spline-based input object. The default spline surface object in Fig. 1(a) is made up of a set of cubic Bernstein–Bezier triangular patches (192 patches for this particular object) that form a rough ellipsoid. A static phase samples the spline surface to form an interpolating triangular mesh. Each triangular spline patch is decomposed into a set of triangular facets using a recursive algorithm to form triangles that interpolate the patch uniformly in parameter space. Each level of recursion forms a set of triangles that is nested in the parameter space corresponding to the preceding level. We will only manipulate vertices of the most detailed level. We maintain the data structure for the more-coarse triangles to allow low-resolution renderings of the final product (Fig. 13) and for extensibility. The spline characteristics of the input object are discarded at this point.

Each triangular Bernstein–Bezier patch of degree n (b^n) is tessellated into 4^k triangles, where k equals the maximum static resolution space desired. We form this tessellation by recursive subdivision, adding triangles of each level of tessellation to the data structure. The zero resolution space representation for b^n interpolates the spline at its three corner points $\mathbf{b}_{n,0,0}$, $\mathbf{b}_{0,n,0}$, and $\mathbf{b}_{0,0,n}$. We then subdivide this triangle by instanciating three new vertices as edge bisectors in the domain. They average the super-triangle's adjacent edge vertices in parameter space. These three new vertices and the three vertices of the super-triangle form four new

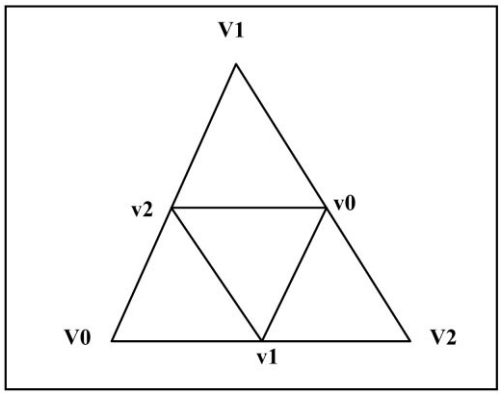


Fig. 2. The 1:4 split in parameter space of the static tessellation.

triangles that are nested in the parameter space of the super-triangle. This recursive subdivision in parameter space forms a tree structure of nested interpolating triangle facets for each input spline patch. Vertices of subdivision, unlike triangles, do not always belong to only one nesting space due to vertex sharing at nesting space boundaries.

No subdivision surface of infinite resolution such as that described by Catmull [4], Doo [5], or Dyn [7] is created. We subdivide as part of forming a nested hierarchy, not for forming a procedural surface. Nesting allows great extensibility. It may be employed in schemes to speed rendering, contact detection, and dynamic splitting of triangles. The static nested hierarchy possesses a 1:4 split as shown in Fig. 2. We form a forest with trees of equal static height built up through subdivision. The static nesting is homogeneous for the entire object—every region of the object has a common maximum static nesting depth. External consistency [8] is ensured through the subdivision geometry.

2.2. Updating the surface tessellation—dynamic splitting

As the vertices are manipulated, the triangle facets tend to grow in size giving poor quality triangles. In order to increase detail during manipulation, and to be able to produce unlimited reshaping of any surface region, new vertices are inserted via dynamic splitting. We split triangles along an edge midpoint. The edge-sharing neighbor is split as well, using a 1:2 split as shown in Fig. 3. This dynamic splitting method prevents holes, is local, and is fast. It allows for local regions of very high detail and therefore is non-homogeneous. In the static phase (setup), all splitting is done in a triangular spline's domain. In the dynamic phase (manipulation), all splitting is done in Euclidean Coordinate space.

One must govern dynamic splitting conservatively. Splitting too much leaves too many vertices in an area of low fidelity. This over-refinement also causes inherent surface noise. We want to split only where there is a good likelihood that the sub-triangles will be manipulated out of the common plane that they share. We do this using the following method: if a vertex is moved, it sets a flag indicating the

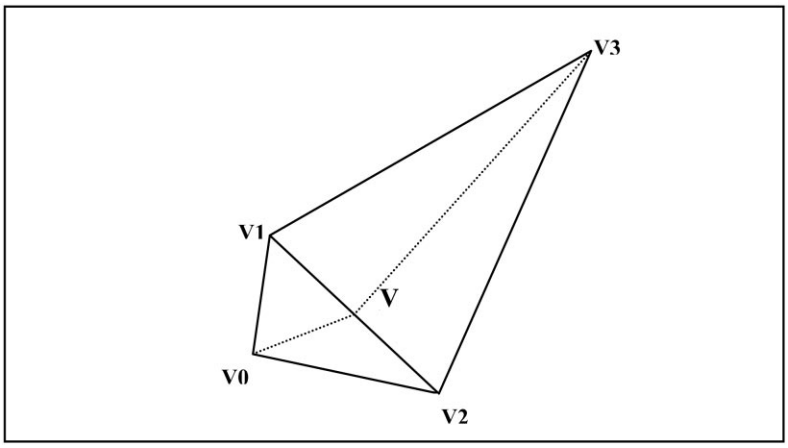


Fig. 3. The dynamic 1:2 splitting of a triangle pair in Euclidean space.

nesting space corresponding with the vertex has been manipulated. A master frame-cycle clock is kept, and each displaced vertex has its private time-stamp set to the current frame time. All nesting spaces whose manipulated flag is set are screened for its leaf-node triangles having one or more of its three vertices moved within a certain number of frame-cycles. This improves the likelihood that newly split triangles will be manipulated out of their common plane since some of the triangles' vertices are in the process of being displaced. These triangles are further subdivided if its longest edge is greater than a tolerance size. The triangle is either left untouched or is subdivided. If the triangle is subdivided, it is split into two new nested triangles by instanciating a new vertex. This vertex is placed at the midpoint of the longest edge. The other triangle, that shares the longest edge, must be found. It too must be subdivided in the data structure using the same new vertex. The edge sharing neighbor triangle must be in the same space or a neighbor nesting space of the triangle being split. During a dynamic split, four new triangles and one new vertex are instanciated.

Since dynamic splitting is intertwined with deformation, when determining what tolerance to use when considering if a triangle's longest edge is too long, consideration is given to the current scale of the deformation. Deformation scale is modeled as being proportionate to the amount of material grabbed and so we use an edge length tolerance proportional to the standard deviation of the current frame's Gaussian distribution used to distribute the discrete displacement vector of the user's glove.

2.3. Glove input

Using the system involves wearing an instrumented glove that allows the user to grab the virtual surface. The surface can be pulled or pushed, inward or outward, perpendicular to or in an oblique direction to the surface tangent. Closing or opening the hand defines a larger or smaller region of the surface to manipulate. In each frame, a Fifth Dimension Inc. glove's average finger bend defines a spherical amount of the object grabbed, called the sphere-of-influence, which in turn determines what part of the object will be affected. A Polhemus 3D tracker on the glove provides a position change, called the displacement vector; the vector's tail is at the previous frame's hand position and its tip is at the current frame's hand position. The plane perpendicular to the displacement vector and containing its tail is called the displacement plane.

The data glove provides five finger input signals. The tracker provides three coordinate signals. These eight signals are each preconditioned with a first-order lag as described by Dorman [6]. The first-order lag provides a unity blend of the current measured value and previous frame's calculated value. This filters out spurious components, as well as smoothing the user's motion. Ultimately, we want hand position at the end of the previous and current frames, we use these values to calculate a displacement vector for the current frame, and we calculate a measure of "hand closure" for the current frame's sphere-of-influence's radius.

2.4. Selecting vertices to be moved

Influence detection is the act of determining which vertices of the object are to be affected by the deformation. Distance calculation is key to influence detection. An exhaustive testing method with two levels of fast distance calculation is used. The second level (level two) of distance calculation is also used in calculating a weighting factor for those vertices that are influenced.

The first level of contact detection (level one) finds all vertices that are within a hand centered cube with side equal to the sphere-of-influence's diameter. This is implemented by using a less-than-or equal to condition on the distance from a vertex to the glove point and the sphere-of-influence's radius. For glove point **g** and

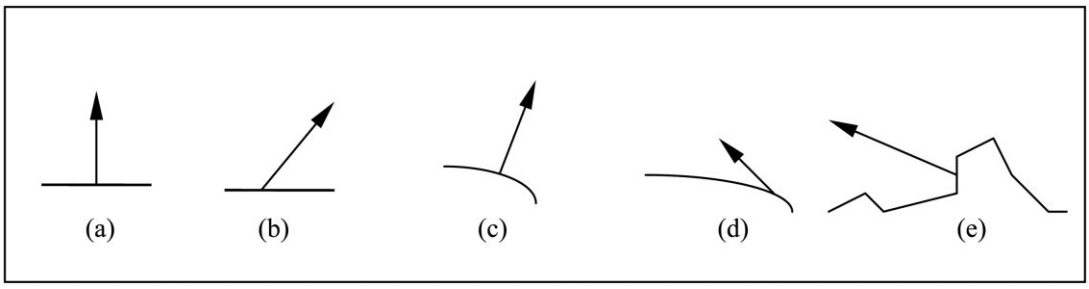


Fig. 4. Five surface manipulation cases (a) planar surface, normal displacement, (b) planar surface, skew displacement, (c) smooth surface, normal displacement, (d) smooth surface, skew displacement, (e) piece-wise linear reconstructed surface, skewed displacement not based at a vertex.

vertex **v** we calculate

$$\text{dist}(\mathbf{v},\ \mathbf{g}) = \text{maximum}(\text{abs}(v_x - g_x),\ \text{abs}(v_y - g_y),\ \text{abs}(v_z - g_z)) \quad (1)$$

abs(x) is the absolute value function for the scalar value x, and maximum() provides the maximum value of its scalar arguments. This value must be less-than-or-equal to the current sphere-of-influence's radius for level two testing to occur. Level one testing finds all vertices that are within the smallest axis aligned cube that encloses the sphere-of-influence. Vertices within this cube are a workable subset of the object's vertices. This subset contains all the vertices that are within the sphere-of-influence.

Level two testing calculates the exact Euclidean distance squared from the vertex to the glove point, for each vertex within the subset found in level one testing. The distance squared must be less than the frames current sphere-of-influence's radius squared for the vertex to be moved. Since our system is fast, the displacement vector is small and no problem has been observed for two nearby vertices, one inside, the other just outside the sphere-of-influence.

2.5. Weighted displacement of selected vertices

A weighting distribution governs how the lumped displacement vector is applied to the vertices. So for each vertex grabbed a weighting factor is calculated. This factor ranges from zero to one and is a function of the glove point position, vertex position, and the displacement vector. A vertex at the center of the glove is given a weight of one; this decreases to zero for vertices outside the sphere-of-influence's perimeter. Each vertex is displaced in the same direction for that frame. We use a Gaussian shaped weighting of the discrete displacement. The displacement vector's angle of incidence, as well as the surface's deviation from planar, causes this simple manipulation to require the addition of a transformation before proper weighting may be given to individual vertices.

2.6. Mapping vertices to the weighting function space

The following definitions are introduced to describe the Gaussian weighting function used to calculate the displacement vector for each of the selected vertices:

v	a vertex in R^3 with coordinates v_x, v_y, and v_z
ab	the vector from point **a** to point **b**
vect1*·*vect2	the dot product between vector ***vect1*** and vector ***vect2***

We use a linear transformation to map each vertex into the domain of the weighting function. This changes the manipulation of a non-planar surface, with a displacement that is not in the surface normal's direction, into the simple case where the surface is planar and the displacement vector is along the plane's normal (see Fig. 4).

In Fig. 5, three different mappings to the Gaussian's abscissa are presented. Method 1 is described by Alciatore [1], and by Bryson [3]; they use Euclidean straight-line distance from a selected point on the surface closest to the glove to the point in question, as the abscissa value. In method 2, a linear Euclidean distance from the glove point to a vertex is the abscissa used. We use method 3, in which the length of a vector from the glove point to a vertex is reduced by projecting the vector onto the displacement plane. Method 3 is significantly more predictable than either method 1 or method 2 and allows for the pleasing and intuitive manipulation presented. Fig. 6 shows how the glove displacement vector affects the selected vertices.

2.7. Initializing the look-up table

We use an array of 64 Gaussian function tables of increasing standard deviation. Each function table has 1024 entries corresponding to abscissa values from zero up to four standard deviations. The Gaussian domains are cutoff sharply at four standard deviations. A value of four provides a function value acceptably close to the Gaussian's asymptotic value of zero. This provides good spatial-blending of displaced vertices with adjacent stationary vertices. The 64 standard deviation values were chosen empirically during a system

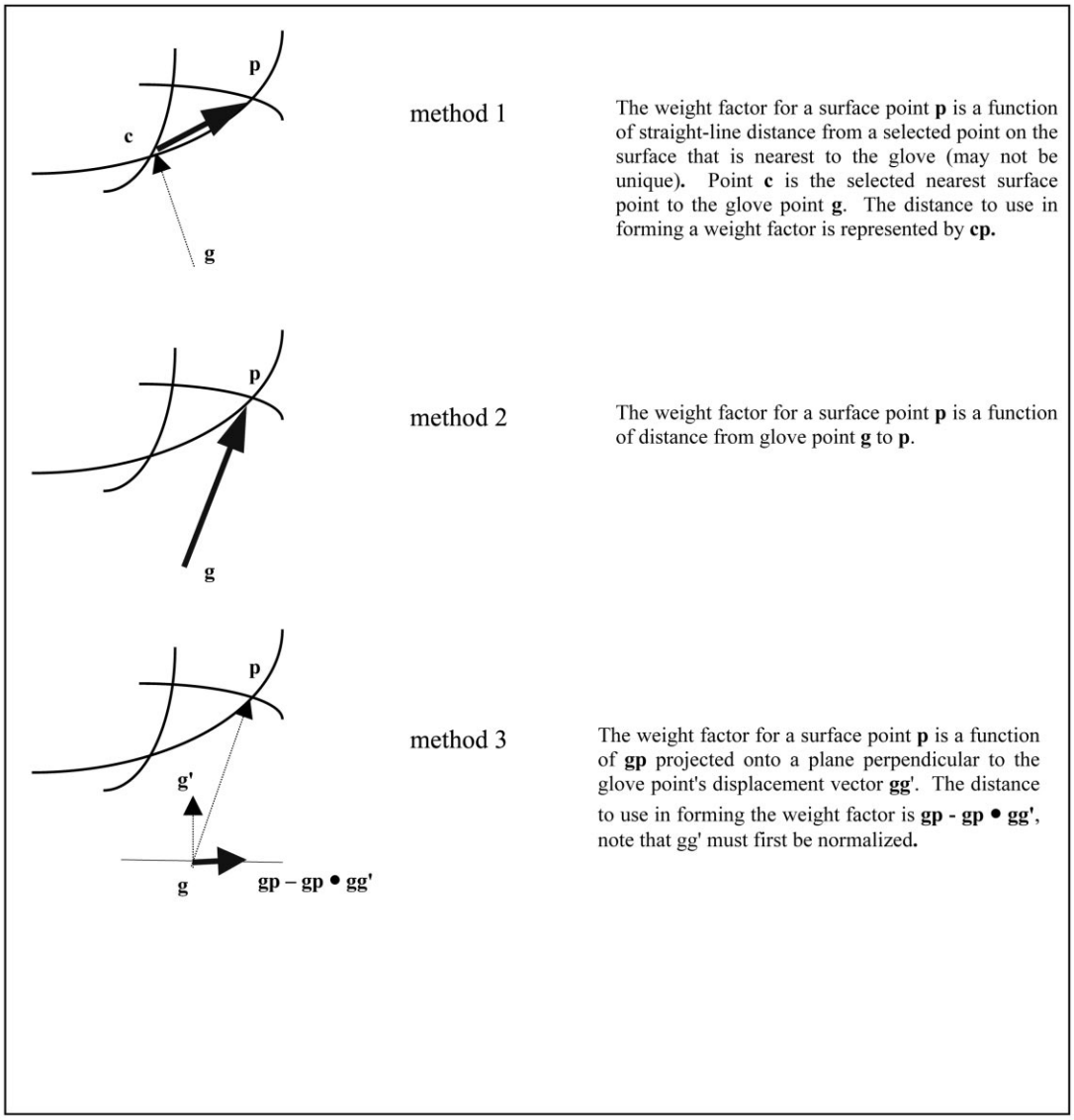


Fig. 5. Three ways of calculating the weight function for the displacement of a vertex at **p**.

tuning session to be a linear distribution ranging from 0.009 to 196. Along with each table we store its corresponding standard deviation, standard deviation squared, four standard deviations, and four standard deviations squared.

2.8. Calculating parameter values for the frame

The five filtered finger bend values are averaged and linearly mapped to the set of standard deviations that we use in the tables in order to provide the frame's current Gaussian standard deviation. The sphere-of-influence's radius is set to four standard deviations, equivalent to the finite symmetric domain of the current frame's Gaussian function.

An average finger bend tolerably close to zero implies a fully open hand and release of the object—a weight of zero for all vertices. We check for this and do not calculate any vertex displacements or perform dynamic splitting. The maximum finger bend value that the glove delivers implies grabbing as much of the object as is possible i.e. using the system's maximum standard deviation. This provides a good intuitive metaphor of grabbing a piece of the object's material and releasing it by closing and opening the hand's fingers respectively.

2.9. Visual display of glove and selected vertices

In the absence of tactile cues, visual cues are a reasonable substitute to the designer. They increase the predictability and thereby the control of the deformation. Fig. 1 shows the visual cues used during manipulation. The glove point's position in space is shown only as a non-intrusive cube object. All finger-bend feedback is conveyed by the set of grabbed vertices' glyphs in relation to the glove point and

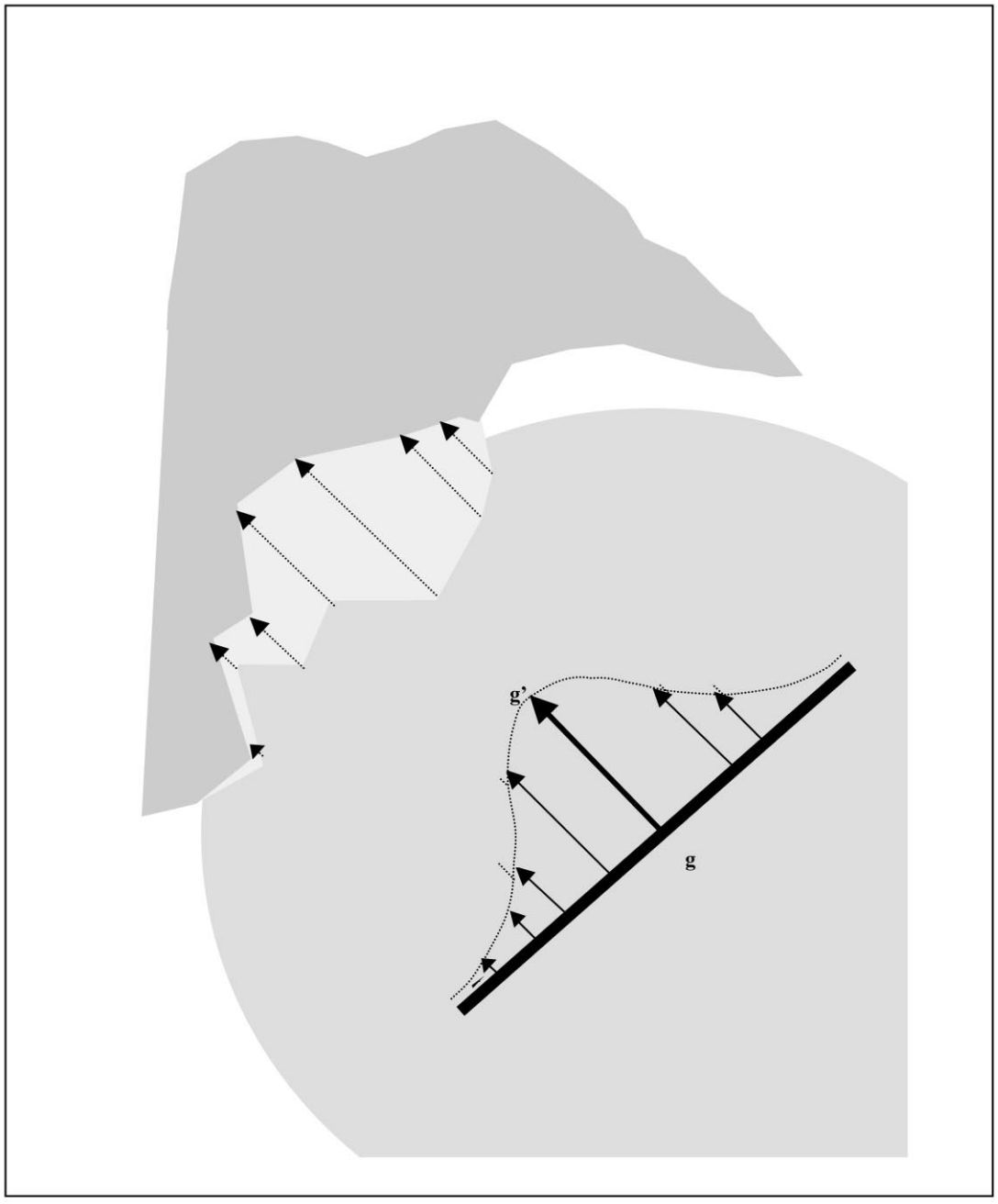


Fig. 6. The effect of the displacement vector field on the polygon surface—the vector field caused by the glove displacement vector $\mathbf{gg}'$ shapes the object in the upper left. The glove's sphere-of-influence is shown in the lower right in orange. The middle section shown in teal is the portion of the original surface that is displaced during the frame.

the object. These glyphs are small visual cues placed one at each selected vertex. This gives the impression of a highlighted section of the object that is currently held in the hand.

2.10. Ways to speed-up the calculations

Our system must be fast in order to be used by common single-processor personal computers. For speed we use a polygonal surface model, a fast two-phase contact detection scheme, avoidance of trigonometric functions, tabulated data for costly functions using direct integer array indexing to access the table data, and an efficient rendering management scheme.

A polygon has its vertices' positions explicitly stored. Using explicit vertex position benefits deformation speed since a vertex position may be updated by simply adding a displacement vector. A spline methodology often requires a minimization of a system of equations. A system using

Finding the quadratic approximating surface

1) form the set of vertex points P = {$\mathbf{p}_i$: i = 1, 2, ..., n; n > 10} which are to be approximated
2) fill row i of C with the coordinate values for point $\mathbf{p}_i$
3) calculate U = C^t C
4) perform a Cholesky decomposition on U
5) set a_i = cofactorSign(i) det($cofactor_i$)
6) normalize using $a_i = a_i / a_{10}$

Fig. 7. The algorithm for finding the overdetermined implicit surface.

constraints that require constantly finding eigenvalues is implicitly slower than one that uses simple vector addition.

The Gaussian table size is large enough so that interpolation of values is unnecessary. The tables contain Gaussian values of the square root of the input values; they can thus be accessed directly with the square of the abscissa. We calculate the index to access the table data using the integer value of [1024 $\text{dist}^2(\mathbf{gp} - \mathbf{gp}\cdot\mathbf{gg}')$]. We use the same table data for weighting vertices in our smoothing method.

We use a static number of nesting spaces (input spline patches) to allow for an array of pointers into linked lists of triangles. The surface manipulation each frame is encapsulated in a known subset of nesting spaces. We render each nesting space in its own OpenGL display list, thereby limiting the display lists that need to be created each frame. The 12 line segments of the glyph that indicate a selected vertex and the glove point, besides satisfying our concerns of not obscuring the scene and still being highly visible, are also quickly rendered in OpenGL.

3. Smoothing the object

A characteristic of directly manipulating a faceted object is that it often becomes jagged. The faceted nature of a polyhedral object is more noticeable, especially when physically handled, for an object with a rough surface (discontinuous surface tangent) compared to a smooth feeling object. This has been observed with physical renditions of objects made by our system. One may wish to minimize the "faceted feel" of the object, so we provide a smoothing method to reduce these geometric discontinuities with second-order precision.

Our approach to smoothing involves these three stages:

1. the user selects a region of the object which is to be smoothed;
2. an approximating quadric implicit surface is found;
3. the vertices are relaxed towards that surface.

The method described by Pratt [13] is used to find an approximating surface. It is a non-iterative method that solves an overdetermined linear system, to find function coefficients. This method was chosen over a least squares approach using Euclidean distance because it has only static space requirements for any number of points and easily handles incremental addition and deletion of points.

3.1. Smoothing overview

The vertex relaxation process involves displacing vertices by following the gradient of a scalar field. The polynomial function f defines a scalar field in R^3—a field function. The field function has associated isosurfaces defined by the set $\{\mathbf{p} \in R^3 : f(\mathbf{p}) = c\}$ where $c \in R$. The equation $f = 0$ represents the implicit surface of the weighted least squares approximation of the surface to be smoothed. The second order polynomial surface

$$f(x, y, z) = a_1 + a_2x + a_3y + a_4z + a_5xy + a_6yz + a_7xz + a_8x^2 + a_9y^2 + a_{10}z^2 = 0 \quad (2)$$

requires specification of 10 coefficients $A = \{a_i \in R : i = 1, 2, \ldots, 10\}$.

3.2. Selecting a region to smooth

The user selects a region using the same interface as that used to manipulate vertices. A set is formed containing all vertices within the user defined sphere-of-influence. This group is used as an unconnected set of points to calculate a surface that best approximates this set. We use best fit with algebraic distance and the quadratic normalization basis $(x^2 + y^2 + z^2)$ used by Pratt [13]. This is close to the least possible sum of vector distances of all points $\mathbf{p}_i$ to the quadric surface that approximates the selected points.

Placing vertices near the approximating surface

1) form the set of vertex points P = {$\mathbf{p}_i$: i = 1, 2, ..., n; n > 10} that are to be smoothed

2) find the implicit surface that best fits P

3) for each point $\mathbf{p}_i$ in P

4) calculate weightedTol = (1.0 – Gauss(dist($\mathbf{g}$, $\mathbf{p}_i$)) maxTolIncrease + minTol

5) while (abs(f($\mathbf{p}_i$)) > weightedTol and have not looped excessively)

6) w = - tuningFac f($\mathbf{p}_i$) / |∇f |

7) (coord($\mathbf{p}_i$)$_j$ += w ∇_jf) for j = x, y, z

Fig. 8. Our algorithm for relaxing vertices.

3.3. Procedure for finding the quadric in the overdetermined case

The procedure we use (Fig. 7) is a direct method (non-iterative) based on the work by Pratt [13], and by Lawson [11]. The point $\mathbf{p}$ may be expressed in terms of a set of basis functions. We use the polynomial with basis b_1, b_2, …, b_{10} where $b_1 = 1$, $b_2 = x$, $b_3 = y$, $b_4 = z$, $b_5 = 2xy$, $b_6 = 2xz$, $b_7 = 2zy$, $b_8 = x^2 - y^2$, $b_9 = y^2 - z^2$, $b_{10} = x^2 + y^2 + z^2$. For proper conditioning of the linear system used to find $\{a_i\}$, we assume there are more than 10 points to be approximated. Matrix $\mathbf{C}$ is of dimensions $m \times 10$ where m is the number of points to approximate. Matrix $\mathbf{U}$ is found as the product of $\mathbf{C}$ transpose ($\mathbf{C}^t$) and $\mathbf{C}$ and is of dimensions 10×10. The Cholesky decomposition finds the positive definite upper triangular matrix $\mathbf{V}$ $\mathbf{U} = \mathbf{V}^t\mathbf{V}$. Positive definite means that all diagonal elements are non-negative and all eigenvalues are positive [14]. Using $\mathbf{V}$ we solve the linear system for the set $\{a_i\}$ by the equation given in step 5 where cofactorSign(i) $= (-1)i + 1 :$ $i =$ 1, 2, …, 10 and det(cofactor$_i$) is the determinate of the cofactor sub-matrix corresponding to row and column i of $\mathbf{V}$. We must normalize the set of coefficients by scaling using the last coefficient as shown in step 6.

3.4. Vertex relaxation

After the 10 coefficients have been found, the vertices must be moved towards the discovered surface. The important task is to decide where on the implicit surface we place these points. These points appear unconnected but they hold a correspondence with vertices of a mesh. We desire that the relaxation method minimally affect the underlying topology. For example it would be poor to cause any self-intersections in the polyhedral surface. We do not simply find a point on the surface that is closest to a vertex; this will not work. The relaxation method moves a vertex along a streamline from an initial position to a final one that is near or on the fitted surface. We may picture the 10 coefficients of our surface fit as forming a field in space, often, but not always, with positive field values on one side of the surface, negative values on the opposite side. We let each vertex flow through this spatial-field to a point on the quadric surface. We take steps in the negative gradient direction using the magnitude $f_{\text{current}}/|\nabla f|$tuning factor. Variable f_{current} is the current potential value of the point during the iterative relaxation process and $|\nabla f|$ is the scalar gradient value at the point's current field position. We use a magnitude that is proportionate to f_{current} since we want large steps when the vertex is far from $f = 0$ and small steps when the vertex is near $f = 0$. We also want small steps when the vertex is at a point where f varies rapidly (large gradient), and large steps when the vertex is at a point where $f$varies slowly (small gradient), and so we use a magnitude that is inversely proportionate to $|\nabla f|$. The tuning factor balances these two contributions to step size; we use a value of 0.5, which usually produces an acceptable convergence time of under 30 iterations.

3.5. Procedure for placing vertices near the approximating surface

We assume the user has already selected a section of the surface to be smoothed, as implied in Section 3.2; this is step 1 in Fig. 8. The factor weightedTol in step 4 is the tolerance used to determine if $\mathbf{p}_i$ is close enough to the smooth surface. It is at least as large as minTol and may be as large as maxTolIncrease + minTol. We use a weighted distribution to form the approximating surface based on a vertex's proximity to the glove. We use a Gaussian weighting distribution to cause points near the glove point $\mathbf{g}$ to move very close to $f = 0$; points further away from $\mathbf{g}$ to tend to remain near their original position. We do this by reducing the tolerance for vertices near the glove. We move vertices against ∇f using the weighting factor w that,

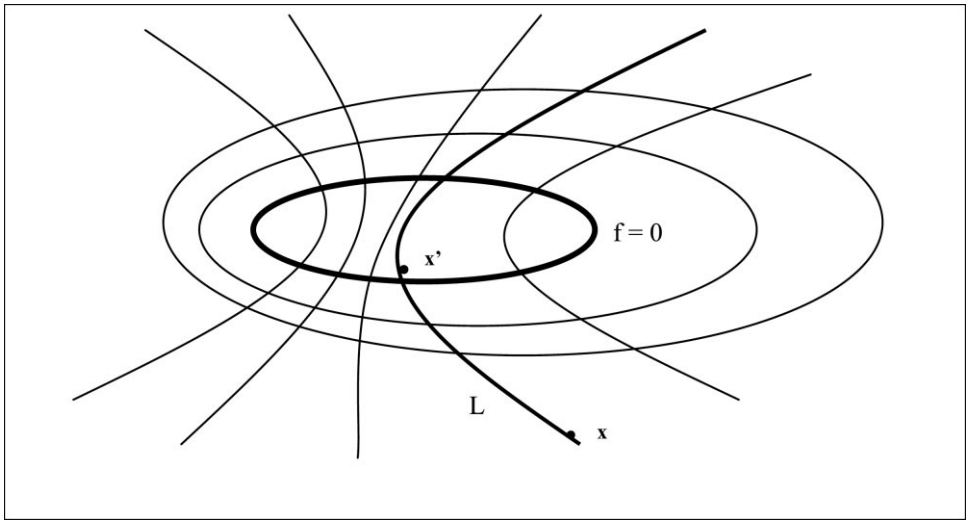


Fig. 9. Vertex **x** is relaxed onto approximating surface $f = 0$ by placing **x** at the first reached intersection point of $f = 0$ and gradient line L.

for the reasons discussed in Section 3.4, is based on the empirical factor tuningFac, the algebraic distance of the vertex from the smooth surface, and the magnitude of the function gradient at the vertex.

Fig. 9 depicts the vector field that is the gradient to the solution polynomial function *f*. A general method of relaxing vertices would be to place each vertex at the first reached intersection point of the gradient line and $f = 0$, for the gradient line that intersects the vertex. The gradient line L intersects vertex **x**, L intersects $f = 0$ closest at $\mathbf{x}'$, and so to relax **x** place it at $\mathbf{x}'$.

3.6. Surface change during smoothing

During smoothing the surface either expands or shrinks as a result of relaxing vertices onto the approximating surface. Any one particular surface likely either shrinks or expands, but the method is not predisposed either way. This is due to our calculating the approximating surface so that the original vertex positions are on either side of the final surface. Other polygonal object smoothing methods, such as subdivision described by Peters [12], either always expands or always shrinks the surface according to the specifics of the method. Also, our method differs from this and other subdivision algorithms, such as those by Dyn [7] and by Doo [5] in that our method moves existing vertices and does not rely on adding new vertices to smooth the geometry.

3.7. Maintaining the surface mesh during relaxation

Vertex relaxation displaces vertices towards the quadric $f = 0$ according to the vector field **L**.

$$\mathbf{L} = \nabla f = (\sigma f/\sigma x, \sigma f/\sigma y, \sigma f/\sigma z) \tag{3}$$

$$\mathrm{curl}(\mathbf{L}) = (a_6 - a_6, a_7 - a_7, a_5 - a_5) \tag{4}$$

$$= (0, 0, 0) \tag{5}$$

Since the curl(**L**) = (0,0,0) the field has no vortex regions that may trap a vertex preventing convergence towards the quadric and the flow field does not twist and so does not adversely disturb the mesh connectivity due to twisting of facet edges. Using higher than a second order surface would likely introduce undesirable twisting of facet edges.

3.8. Smoothing method deficiencies

Two deficiencies with the smoothing method have been identified, one concerns the quality of the quadric fit and the other has to do with the connectivity of the input set of points. The smoothing method requires that the input set of points be well approximated by a quadric surface. No monitoring of the quality of the implicit surface in approximating the input set of points is made. It is incumbent upon the user to judge when a quadric surface will be an acceptable approximation. This is done interactively by properly selecting input points.

It would be possible for the system to calculate a quality of fit metric. This metric might simply be the average algebraic distance of a point to the approximating surface. If this metric is too large, then, the system could respond by not performing the relaxation phase. Two disconnected pieces of the surface selected for smoothing may or may not have a good quality of fit metric. Ultimately user discretion is needed during the design process.

The system does not detect nor correct the second deficiency. Remember that the smoothing method operates on a disjoint set of points. The points do however represent a connected mesh of vertices. There is no guarantee that the polygon edges formed from these vertices do not cross or

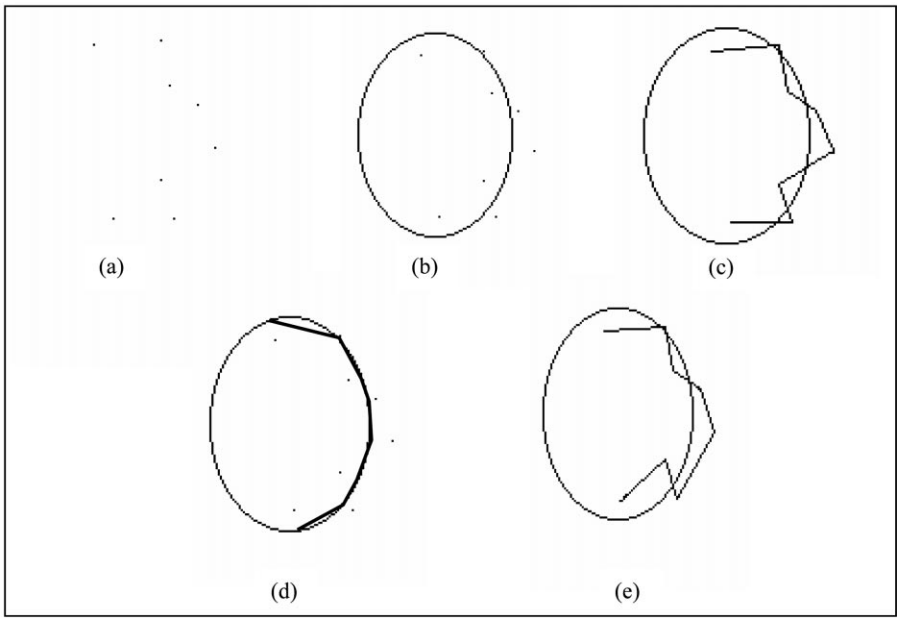


Fig. 10. (a) A point cloud in R^2, (b) a least squares fit is formed, (c) one possible connectivity associated with good smoothing results, (d) a piece-wise linear reconstruction of relaxed points is formed using the previous connectivity, (e) a different possible connectivity associated with poor smoothing results; the piece-wise linear reconstruction will be self-intersecting.

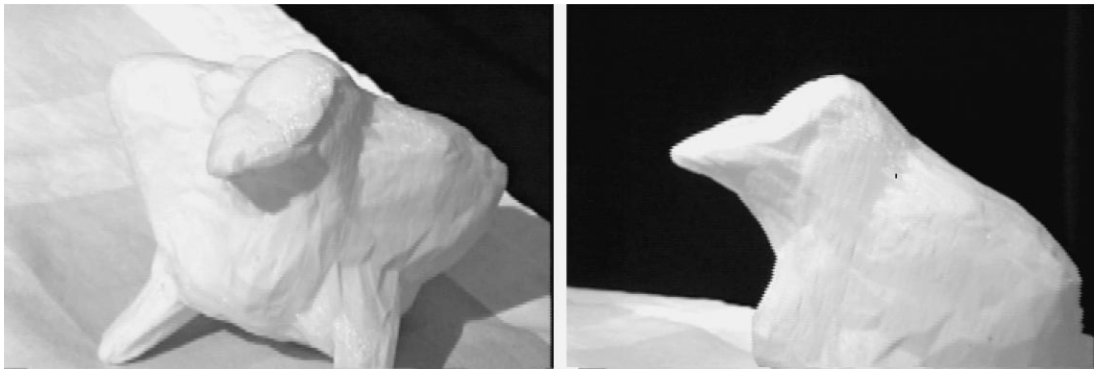


Fig. 11. Two photographs of a plastic stereo-lithography bird-object created using the system.

overlap. Fig. 10 outlines two possible scenarios. Fig. 10(a) shows a point cloud, in R^2. Fig. 10(b) shows an approximating quadric curve to the point set. Fig. 10(c) shows one possible connectivity of the point set. This connectivity will be associated with good smoothing results. Fig. 10(d) shows the linear curve segment after vertex relaxation. In Fig. 10(e) a different connectivity of the same point set is shown. This however, is associated with poor smoothing results. Relaxation would produce the same final vertex positions as the previous connectivity. These poor results are due to part of the curve being folded over during the relaxation phase. We have not tried to implement the detection and correction of this phenomenon. The curve in Fig. 10(e) is actually a poor candidate for quadric approximation, although the average algebraic distance relaxed is small. The case of two vertices lying on the same streamline is the boundary case of this second deficiency.

4. Results

All these objects depicted started from the default ellipsoid object; we have also tested and used the system with initial objects that are topologically different from the sphere, such as a torus. Fig. 11 shows two photographs of a plastic stereo-lithography "bird" created using our system. In Fig. 12, six objects are presented, the objects in the center row show smoothing. Fig. 13 shows a rabbit, chicken, and a fanged dog. Dynamic splitting during deformation produces striations on the surface. These striations follow contours that tend to accent the object's form. They are due to triangles stretching and becoming slender in the direction of the pull (or push), and then splitting. The system's smoothing tool minimizes striations of the original surface.

We would like to incorporate a second glove and 3D tracker for the user's second hand to use for object

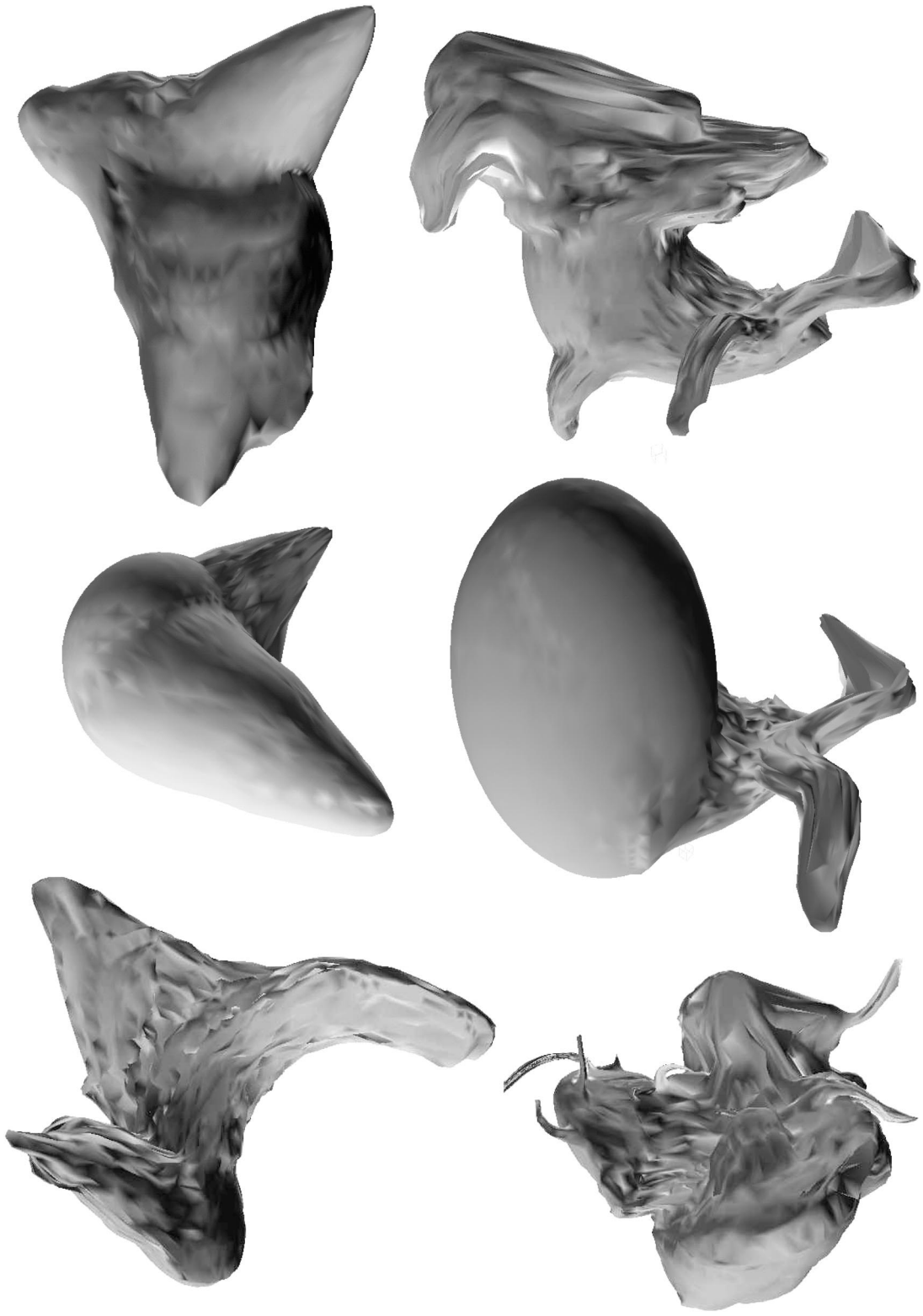


Fig. 12. Six sculptures produced by the system.

translation and rotation. This would add the metaphor of holding the object in one hand while using the other hand to sculpt. To grab hold of the object, the second hand would simply close its fingers, then the object would track the hand's translation and rotation. Opening the hand would release the object, which then remains fixed in space. The only burden placed on the program every frame would be to poll the glove and use the five

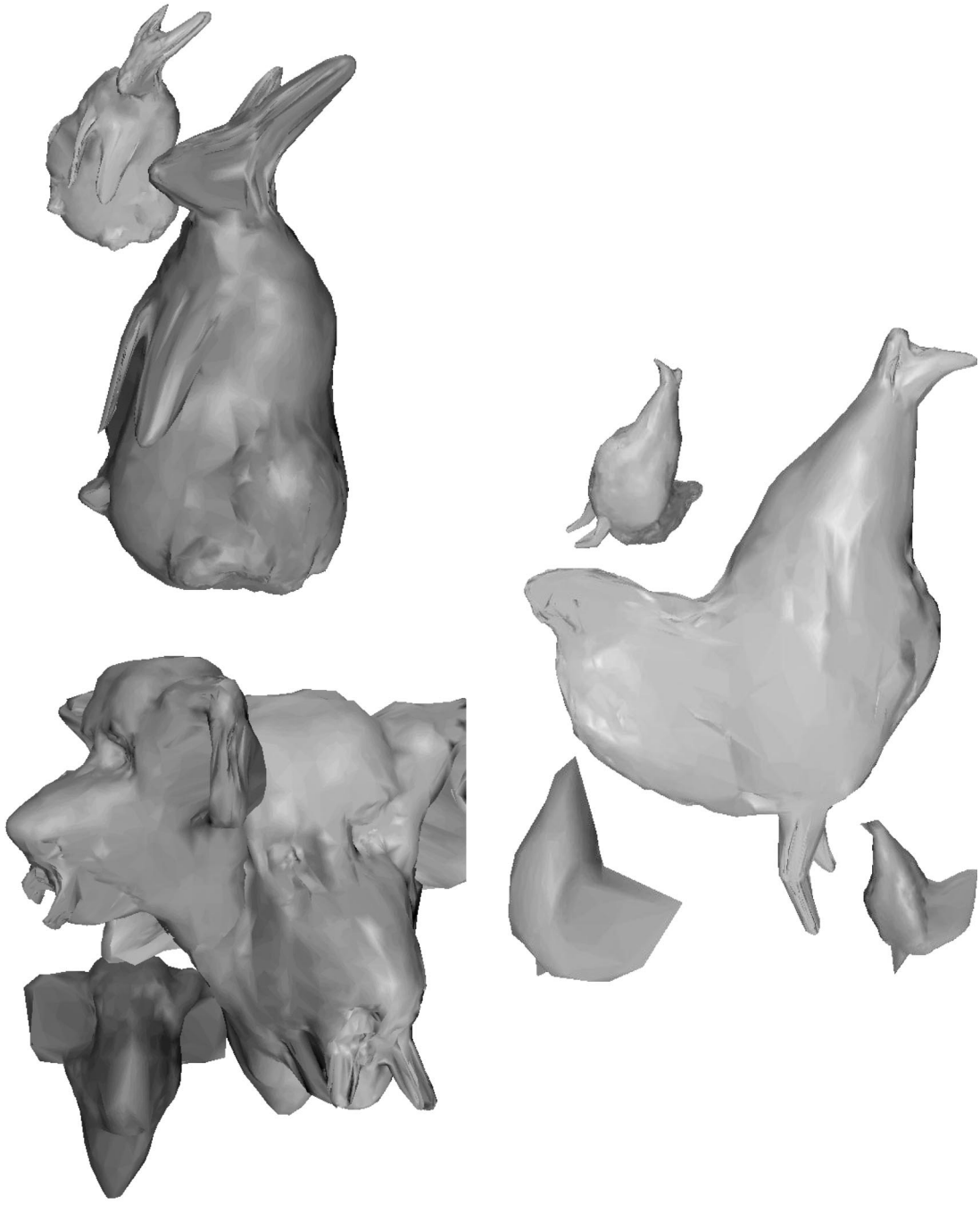


Fig. 13. Three animal figures shown in different resolutions.

finger bend angles in a comparison to determine if the hand is closed. We currently handle mouse events to translate and rotate the object using different button for translation and rotation.

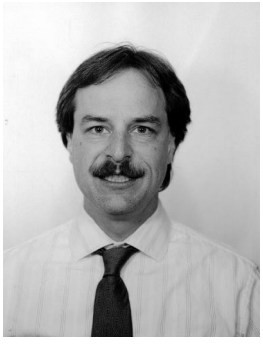
References

[1] Alciatore DG, Wohlers TT. Importing and reshaping digitized data for use in rapid prototyping: a system for sculpting polygonal mesh surfaces. Rapid Prototyping Journal 1996;2(1):13–23.
[2] Borrel P, Rappoport A. simple constrained deformations for geometric modeling and interactive design. ACM Transactions on Graphics 1994;13(2):137–55.
[3] Bryson S. Paradigms for the shaping of surfaces in a virtual environment. RNR technical report, RNR-92012, January 1992.
[4] Catmull E, Clark J. Recursively generated B-spline surfaces on arbitrary topological meshes. Computer Aided Design 1987;10(6):350–5.
[5] Doo D, Sabin M. Analysis of the behavior of recursive division surfaces near extraordinary points. Computer Aided Design 1978;10(6):356–60.
[6] Dorman J. Minicomputer part-task simulation for helicopter development. Proceedings of SAE AEROTECH paper #901814, Long Beach, California, 1990.
[7] Dyn N, Levin D, Gregory JA. A butterfly subdivision scheme for

surface interpolation with tension control. ACM Transactions on Graphics 1990;9(2):160–9.
[8] Fournier A, Fussel D, Carpenter L. Computer rendering of stochastic models. Communications of the ACM 1982;25(6):371–84.
[9] Fowler B, Bartels R. Constraint-based curve manipulation. IEEE Computer Graphics and Applications, 1993.
[10] Hsu WM, Hughes JF, Kaufman H. Direct manipulation of free-form deformations. Computer Graphics 1992;26(2).
[11] Lawson C, Hanson R. Solving least-squares problems. New York: Prentice-Hall, 1974.
[12] Peters J. Smooth free-form surfaces over irregular meshes generalizing quadratic splines. Computer Aided Geometric Design 1993;10(3):347–62.
[13] Pratt V. Direct least-squares fitting of algebraic surfaces. Proceedings of SIGGRAPH '87 1987;21(4).
[14] Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C. 2nd ed. Melbourne, Australia: Cambridge Press, 1994.
[15] Yamashita J, Fukui T. A direct deformation method. IEEE Virtual Reality Annual International Symposium 1993:499–504.

Further reading

[16] Barr AH. Global and local deformations of solid primitives. Computer Graphics 1984;17(3):21–30.
[17] Bloomenthal J, Bajaj C, Blinn J, Cani-Cascuel M, Rockwood A, Wyvill B, Wyvill G. Introduction to implicit surfaces. San Francisco: Morgan Kaufmann, 1997.
[18] Bluth E, Stavros A, Marich K. Carotid duplex sonography: a multicenter recommendation for standardized imaging and doppler criteria. Radiographics 1988;8:487–506.
[19] Cohen M, Gortler S. Variational geometric modeling with wavelets. Symposium, Interactive 3D Graphics, 1995.
[20] Eck M, DeRose T, Duchamp T, Hughes H, Lounsbery M, Stuetzle W. Multiresolution analysis of arbitrary meshes. Proceedings of SIGGRAPH '95, 1995. p. 173–82.
[21] Farin G. Curves and surfaces for computer aided geometric design. 3rd ed. San Diego, CA: Academic Press, 1993.
[22] Finkelstein A, Salesin DH. Multiresolution curves. Proceedings of SIGGRAPH'94, 1994.
[23] Fowler B. Geometric manipulation of tensor product surfaces. Proceedings of Symposium on Interactive 3D 1992:101–8.
[24] Heckbert PS, Garland M. Multiresolution modeling for fast rendering. Graphics Interface, 1994.
[25] Lounsberry M. Multiresolution analysis for surfaces of arbitrary topological type. PhD thesis Department of Computer Science and Engineering, University of Washington, 1994.
[26] Lounsberry M, DeRose TD, Warren J. Multiresolution analysis for surfaces of arbitrary topological type. Department of Computer Science and Engineering University of Washington, technical report 93-10-05b, 1994.
[27] Parent R. A system for sculpting 3-D data. Computer Graphics 1977;11(3):138–47.
[28] Sederberg TW, Parry S. Free-form deformation of solid geometric models. Computer Graphics 1986;20(4):151–60.
[29] Streeter VL, Wylie EB. Fluid mechanics. 6th ed. Maryland: McGraw-Hill Publishers, 1975.
[30] Terzopoulos D, Qin H. Dynamic NURBS with geometric constraints for interactive sculpting. ACM Transactions on Graphics 1994;13(2):103–36.
[31] Calculus GB. Calculus and analytic geometry. 4th ed. Addison-Wesley Publishers, 1975.
[32] Wolf P. An overview of the epidemiology of stroke. Stroke 1994;9:114–6.



Jeffrey Dorman received a PhD in Computer Science and Master of Computer Science from Arizona State University and Bachelor of Science in Mechanical Engineering from Rensselaer Polytechnic Institute. His interests include system integration, mathematical modeling and computer graphics. He has been a developer for SGI, Institute for Studies in the Arts, Rocky Mountain Mathematics Consortium, Scilux Engineering, and McDonnell Douglas Helicopter Company where he was awarded the company's 1990 President's Award.

Alyn Rockwood received a PhD from the Department of Applied Math and Theoretical Physics at Cambridge University. He has had faculty positions in a German "Gymnasium", at BYU and at Arizona State University. He has additionally spent over 15 years in industrial research at Evans and Sutherland, Shape Data Ltd, SGI, a start-up company and is currently at Mitsubishi Electric Research Labs. Altogether he has spent over 25 years as a researcher in mathematics, computer graphics, and simulation. He has produced over 50 peer-reviewed articles, several patents and three books in these areas. A long time SIGGRAPH supporter, he was most recently the SIGGRAPH99 papers' chair.