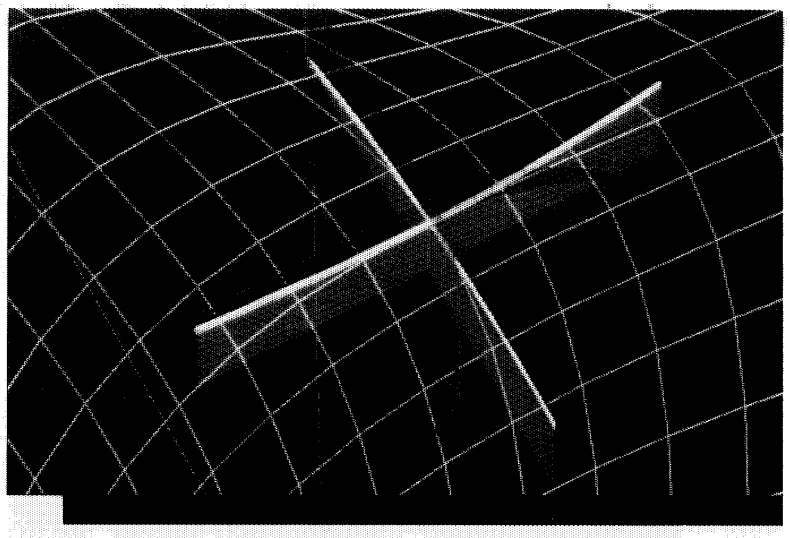

Locally Manipulating the Geometry of Curved Surfaces

Príamos N. Georgiades and Donald P. Greenberg
Cornell University

These new interactive tools let you edit a curved surface locally by altering its intrinsic geometric measures.

Several methods exist for creating and editing curved surfaces globally. We developed an approach to manipulating curved surfaces locally. You can use our methods with many previously published techniques, and, because they are based on the intrinsic differential geometry of the surface, you can apply our methods to all types of surfaces (parametric, implicit, algebraic, and so forth).

Our novel interface consists of two parts. The first uses graphic displays that illustrate specific characteristics of the surface. Users can isolate different types of curves on a surface. These curves yield unique information about the surface. Such curves include coordinate curves of parametrically defined surfaces (curves obtained by varying only one parameter), normal sections (intersections with planes normal to a point on the surface), lines of curvature, asymptotic curves, and geodesics. By moving a 3D cursor about the surface and choosing the appropriate type of curves, users can examine the local behavior of the surface in real time through these icons.

The second part of our interface uses some of the specialized icons to interactively manipulate the surface itself. With these icons, users can isolate a segment of a coordinate curve or a principal normal section, then modify its curvature and torsion or maximum and minimum normal curvature, respectively, at some point on the surface. Curvature and torsion locally define a regular, differentiable curve uniquely up to a rigid body motion. The principal curvatures actually classify all points on the surface. With the modified curve segment, a new surface function can be obtained that contains the curve and meets the neighboring patches with zero, first, or second order of continuity. By using parameterized algebraic surfaces, the surface functions can be computed by polynomial interpolation.

0272-17-16/92/0100-0054$03.00 ©1992 IEEE

While using curvature and torsion is not new in deriving curves and surfaces in computer-aided geometric design,[1] researchers have placed little emphasis on developing measures that let end users manipulate the surface directly. Researchers have proposed surface analysis methods that use intrinsic differential geometry,[2] but the resulting displays (such as color curvature maps, drawings of contour lines, lines of curvature, or geodesics) offer a global examination of the surface, and they take a lot of time to compute. Our local methods address these interactivity and real-time issues.

Locally editing curved surfaces

With local control in mind, let's briefly review the most broadly used methods of creating and editing curved surfaces. Some of the most straightforward construction methods consist of applying extruding, revolving, or sweeping to a planar curve.[3] Lofting—generating a surface by interpolating between different planar curves at points along a spine curve[4]—also falls in this category. Such surfaces are modified by editing the generating curve(s). The extent of local control depends on the curve's representation.

Constructing objects made of primitive-type surfaces (for example, cubes, spheres, and cylinders) is also fairly simple. However, since the representation of each patch of an object thus made (using constructive solid geometry, for example) is fixed and precise, you cannot exert any local control within a patch. In general, implicitly defined surfaces provide exact representation,[4] but they don't offer any interactive local control, even when you can manipulate the surface's defintion, as with skeletally defined implicit surfaces.[5]

The more flexible methods (and less straightforward) use spline-based, parametrically defined surfaces. The Bezier and B-spline formulations are the most popular of these methods. The basic interface to such surfaces consists of adding, moving, and deleting the control points that define them. Bezier surfaces offer no local control, and the degree of the resulting polynomial function increases linearly with the number of control points. B-splines allow some flexibility and local control by setting the order of the spline independently of the number of control vertices. Manipulating the knot vector of the curves in each parametric direction also adds to B-splines' flexibility.[1] Interacting with these additional variables, however, becomes increasingly complex, and with the exception of moving control points, the outcome of manipulating the variables is not very intuitive to a nontechnical user. Furthermore, both these methods hide from the user the exact extent of change that results from moving a control point.

One notable exception to this is hierarchical B-spline refinement,[6] where users operate on points on the surface corresponding to the maximal influence of each control vertex. Users can refine the patch of the surface about such points by introducing additional control points and re-representing that patch, while the rest of the surface remains unaffected.

On a more global scale, a curved-surface interface might include methods for applying regular and free-form deformations,[7,8] as well as for controlling the outline of parametrically defined surface patches by using trimming curves in the domain of the surface.[9] Most of these methods interact with the surface by imposing external constraints, that is, operating on the surface by changing features external to it. Our method, however—a method that you can use in combination with all the techniques mentioned here—changes intrinsic features.

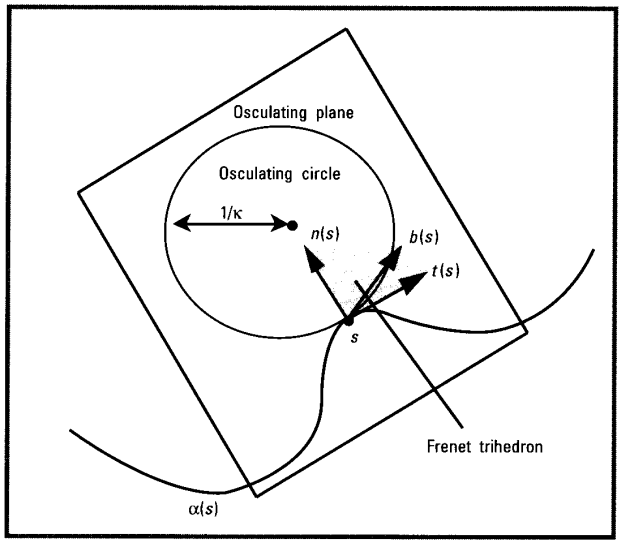

Figure 1. The osculating circle and its relation to the tangent, normal, and binormal vectors (the Frenet trihedron).

Intrinsic geometric characteristics

To adequately describe our interface, we must first define the mathematical features used to modify the surface. (The formal expressions, derived from M.P. DoCarmo's textbook,[10] are explained in the sidebars on the following pages.) Let's begin with the definition and geometric interpretation of the most pertinent concepts.

The curvature of a curve $\alpha : I \rightarrow R^3$ at a point $s \in I \subset R$ is the length of the second derivative vector of the curve map, denoted $\kappa(s) = |\alpha''(s)|$, assuming that $\alpha(s)$ is parameterized by arc length. (Note that a straight line has zero curvature since its second derivative is zero.) The unit second derivative vector $n(s) = \alpha''(s) / |\alpha''(s)|$ is perpendicular to the unit tangent $t(s) = \alpha'(s)$ and hence called the *normal* at s. The plane spanned by the unit tangent and normal vectors is called the *osculating plane*. The circle of radius $1/\kappa$, tangent to the curve at s and lying on the osculating plane, is called the *osculating circle* (shown in Figure 1). Its curvature is constant and equal to κ. Note that there exist two such circles, one on each side of the curve. By convention, we draw the one lying on the positive side of the curve normal.

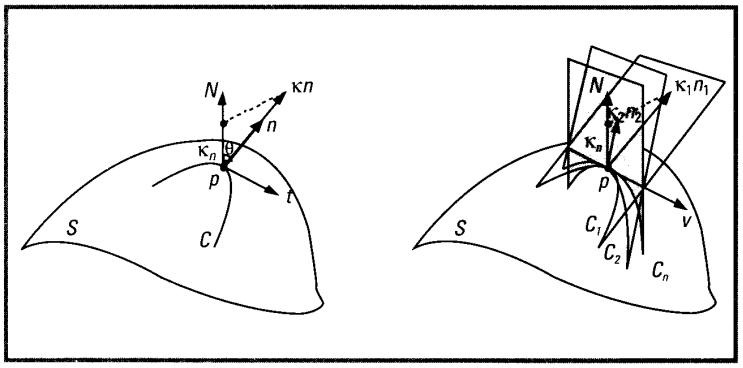

Figure 2. The normal curvature κ_n is the projection of the vector κn onto N. On the right, the two curves C_1 and C_2 have the same normal curvature at p. C_n is the normal section.

The cross-product of the tangent and normal vectors is the *binormal* at s, denoted $b(s)$. The derivative of the binormal $b'(s)$ is colinear with $n(s)$. The scalar function of s, τ, relating the two vectors (such that $b'(s) = \tau(s)n(s)$) is called the *torsion* of α at s. Observe that if α is a planar curve, the cross-product $t(s) \times n(s)$ is constant, and hence its derivative with respect to s is zero. That is, it has zero torsion. Intuitively, the torsion measures how much the curve twists away from the osculating plane.

Curvature and torsion define a regular, differentiable curve uniquely up to a rigid body motion. In other words, if two curves have the same expressions of curvature and torsion in the neighborhood of some point, then the traces of the curves will overlap in the neighborhood of those points when one of the curves is moved so that the two points and the respective curve tangents and normals are identified. This uniqueness is the result of the Fundamental Theorem of the Local Theory of Curves, and we invoke it to justify using these two measures to manipulate the definition of a curve, and hence the surface on which it lies.

The most direct way to isolate a curve on a surface is restricting one of its parameters to be constant. Such curves are *coordinate* curves. In our interface, we refer to these curves as *parametric*, to avoid misleading users to assume they are always aligned with the coordinate axes of the image space. Another class of curves isolated for use in modifying a surface are *principal sections*. Given some point on the surface, the curve of intersection of the surface with the plane defined by the surface normal and some tangent vector at that point is called a *normal section*. When this tangent vector is aligned with one of the principal directions (explained later) at that point, we call the curve a *principal normal section*, or principal section.

The *normal curvature* of a curve α lying on a surface S passing through some point $p \in S$ is defined as $k_n = \kappa \cos\theta$, where κ is the curvature of α at p, and θ is the angle between the surface unit normal N and the curve's normal vector $\mathbf{n}$. The normal curvature measures the projection of the vector $\kappa\mathbf{n}$ onto N. The normal curvature is the same for all curves passing through p that have the same tangent line (see Figure 2). Thus we speak of the normal curvature along a given direction v at p.

The maximum normal curvature k_1 and the minimum normal curvature k_2 are called the *principal curvatures* at p. The corresponding tangent vectors e_1 and e_2 are called the *principal directions* at p. These vectors are mutually orthogonal, and the product k_1k_2 is called the *Gaussian curvature* of S at p.

The Gaussian curvature is used to classify a point on a surface. It is called *elliptic* if $k_1k_2 > 0$, hyperbolic if $k_1k_2 < 0$, parabolic if $k_1k_2 = 0$ but one of k_1 or k_2 is nonzero, and planar if $k_1 = k_2 = 0$. The formulas for deriving the Gaussian curvature are included in the sidebar on this page.

All the metric concepts listed here are defined locally, in the neighborhood of a point on the surface. Furthermore, we can perform the computations needed to derive them without "leaving" the surface. Since the metric concepts are based on the partial derivatives of the surface, they are intrinsic to the

Gaussian curvature in local coordinates

The Gaussian curvature at a point p of a surface can be derived in local coordinates from the equations of Weingarten using the first and second fundamental forms of the surface. They yield a quadratic equation of the form

$$k^2 + (m_{11} + m_{22})k + m_{11}m_{22} - m_{21}m_{12} = 0$$

whose solutions are the principal curvatures at p. Since the derivation of this equation is quite lengthy, we simply provide the values of its coefficients and refer the interested reader to DoCarmo's work.[10]

$$m_{11} = \frac{fF - eG}{EG - F^2} \qquad m_{12} = \frac{gF - fG}{EG - F^2}$$

$$m_{21} = \frac{eF - fE}{EG - F^2} \qquad m_{22} = \frac{fF - gE}{EG - F^2}$$

where E, F, and G are the coefficients of the first fundamental form and e, f, and g are the coefficients of the second fundamental form. The coefficients of both forms are the result of the dot products

$$E = X_u \cdot X_u \quad F = X_u \cdot X_v \quad G = X_v \cdot X_v$$
$$e = N \cdot X_{uu} \quad f = N \cdot X_{uv} \quad g = N \cdot X_{vv}$$

where N is the surface normal at p, and X_u, X_v, X_{uv}, X_{uu}, and X_{vv} are the partial derivatives of the surface function $X(u, v)$.

surface. Hence, the study of such properties is *intrinsic geometry*.

Representing geometric measures

So that users can modify the intrinsic geometric properties of the surface efficiently, the system should present changes to quantifiable characteristics in a graphically clear, concise format that can supplement numerical data. By manipulating the representation of each measure, users can effectively modify the surface itself.

We divided our typical screen layout into three viewing windows (one large and two smaller ones) with a column of menu buttons on the right-hand side. The lower left of the two smaller windows shows the parametric domain of the model being edited. The other two windows show the surface geometry itself. The menu area consists of a set of submenus with headings displayed at the top of the menu. By clicking on each heading, users make a different set of buttons fill the rest of the menu. The submenus contain the operations of the interface, grouped according to their functionality (see Figure 3). The local editing submenu contains the modification buttons and mode switches pertaining to them. The presentation submenu controls the appearance of the surface and the editing tools' icons. The label of each button explains its operation. Let's look more closely at the tools we developed as handles on the surface's geometry. We present them in the order they appear in the interface.

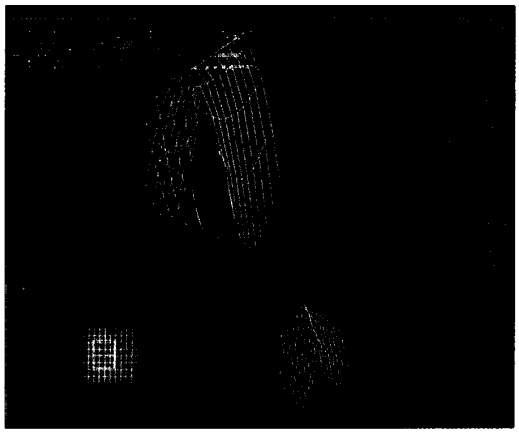

Figure 3. The surface editor interface with the local editing submenu.

3D cursor

While in surface editing mode, there is always a current reference point. When entering this mode, the system sets the reference point to the center of the first patch of the object. The user can later move it about continuously. The domain viewport acts as a potentiometer for moving the cursor, which is displayed simultaneously in the domain and image views so that the correspondence is obvious.

In the image view, the system draws one red and one blue rectangle at the current point. These rectangles intersect the surface as indicated by the cross-planes alignment switch. If the cross-planes are aligned with the parametric directions, then the system identifies the red rectangle with the plane defined by the tangent and normal vectors of the coordinate curve in the **u** direction and that passes through the current point. The blue rectangle is similarly aligned in the v direction. If users want a principal cross-plane alignment, then the system automatically aligns the red rectangle with the plane defined by the maximal curvature direction and the surface normal. It also aligns the blue rectangle with the plane defined by the minimal curvature direction and the surface normal.

To determine the extent of the local region, users can explicitly set the length of each rectangle. The boundary of the region automatically aligns with the parametric directions (see Figure 4). The rectangles are centered about the current (reference) point. The system computes their heights based on the highest and lowest point of intersection of either plane with the surface.

We use a number of display conventions for clarity. We display the patch being edited in green and all the other patches in a cool gray. By default, the polygon fill, boundary lines, and mesh lines switches are active when entering the surface definition editor. The mesh lines appear in white and the boundary lines in yellow. The local region around the current point is displayed with 25 percent transparency to expose the lines of intersection of the red and blue planes with the surface. The portion of the cross-planes above the line of intersection is given 50 percent transparency, while the portion below is opaque. The definition of "above" and "below" is established by the direction of the surface normal at the current point. For emphasis, the lines of intersection themselves are bold white. Corresponding to the cross-planes in the image view, the domain viewport displays a cursor in the shape of a cross, with its two line segments reflecting the position of the red and blue rectangles in parameter space. They are drawn in red and blue (respectively), and the extents of the local region are highlighted in bold yellow. As the user moves the current reference point around the domain, this cross-plane assembly shows the behavior of the surface at the local region.

Osculating circles

Since the curvature at a point on a curve is defined as the inverse of the radius of the osculating circle, it is only natural to visualize the curvature of the red or blue plane curve using the osculating circle. The system draws it at its exact geometric location—tangent to the curve at the current point and lying on the plane defined by the curve's tangent and normal vectors at this point (the osculating plane). The point of tangency and the center of the osculating circle are colinear with the curve's normal. The center is set above the tangency point if the curve is concave and below if it is convex. To amplify their visibility, the system draws the two circles as very thin cylinders. Each circle

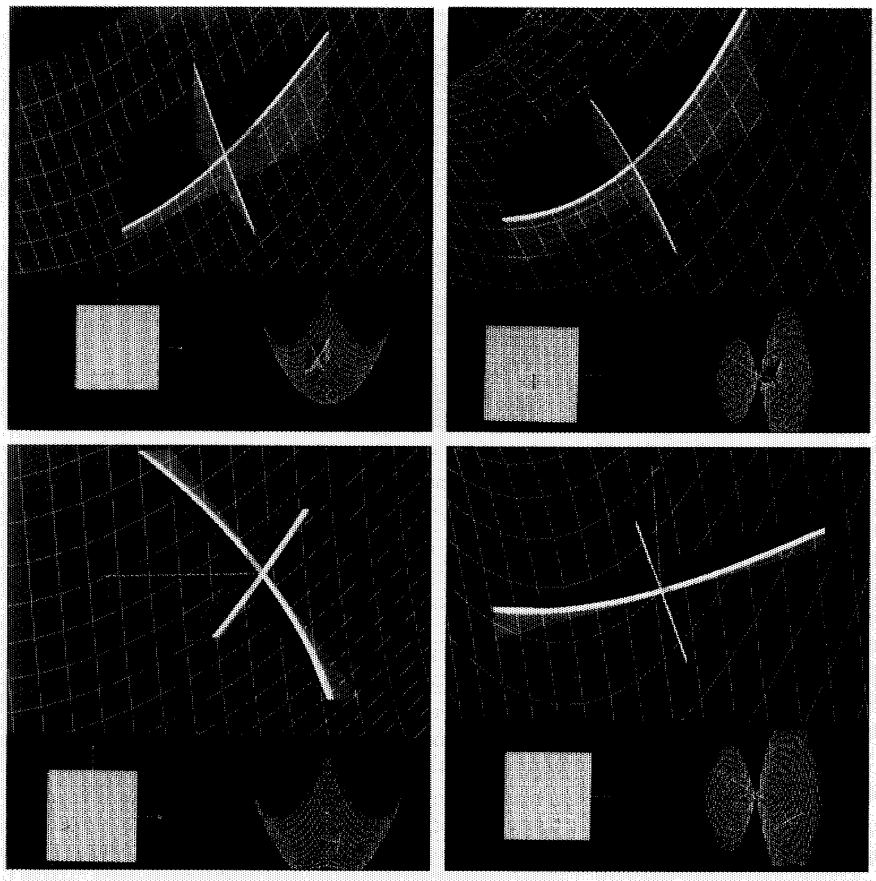

Figure 4. We use the cross-planes to define the current reference point on the surface, delimit the local editing region, and exhibit the correspondence between the parametric and geometric views. A paraboloid of revolution and a hyperbolic paraboloid surface illustrate the parametric and principal alignment of the cross-planes.

appears in the color of the cross-plane on which it lies (Figure 5). With the osculating circles switch on, users can move the current reference point around. The system continuously updates both circles to reflect the curvature of the curves of intersection of the cross-planes with the surface.

Users can modify four different measures of curvature (curvature in u, curvature in v, maximum curvature, or minimum curvature) by selecting the corresponding button. They can interactively enter one or more values for what the curvature should become at the current point. The "move-cursor" mode of the domain window is frozen so that the current point remains stationary during editing. Any one of the viewports can be used as a one-dimensional potentiometer to continuously change the curvature value. Users can also enter one value at a time using the keyboard. For each new curvature value, the system updates the local region so that the edited curve displays the new curvature (as shown in the two images of Figure 6). When the user is done editing, the display status of the interface is reset to what it was before the curvature editing button was selected.

Torsion coil

The measure of torsion of a curve at any arbitrary point is usually difficult to visualize. Torsion measures how much the curve veers away from the osculating plane at that point. More intuitively, it tells us how much twist the curve has at that point. Naturally, a planar curve has zero torsion at every point. Since the red and blue rectangles are already coplanar with the osculating plane of their respective curves, they lend themselves as a reference for representing zero torsion. Presented as if it were a rolled piece of paper, the torsion icon depicts the "veering away" of the curve from the osculating plane. The system displays the icon when the torsion coil switch is activated. Emanating from the intersection edge of the two rectangles, the icon is tangent to the rectangle whose curve's torsion it qualifies. The torsion icon has the same height as the rectangle and the same arc length. The starting radius of the coil is inversely proportional to the measure of torsion and is reduced by 20 percent every half revolution. This coil appears on the side of the plane on which the binormal lies at the current point, in the same color as the rectangle (see Figure 7). To show zero torsion, the icon overlaps the rectangle.

The interface behaves as it did for the osculating circles. Users can interactively enter new value(s) for the measure of torsion one at a time using the keyboard or continuously using the window as a potentiometer. The local region is modified accordingly. When users have finished editing, the interface reverts to its original state. Figure 8 demonstrates the interface before and after editing. The interface lets you twist the surface in the neighborhood of the current point. At its boundaries, the local region meets the adjacent patches in the specified continuity. With the torsion in u and torsion in v buttons, users can currently manipulate only parametric (coordinate) curves.

Derivation

One class of surfaces of particular importance is graphs of scalar functions in two variables. They are of interest to our effort because the parameterization of algebraic surfaces in this class is trivial. They are also important because locally any surface is the graph of a differentiable function. Given a point p of a surface S, you can choose the coordinate axes of R^3 so that the origin is at p and the vertical (y) axis is directed along the normal of S at p (and hence the xz plane is coincident with the tangent plane at p). Then we can represent a neighborhood of p in S in the form $X(u, v) = (u, h(u, v), v)$, with $(u, v) \in U \subset R^2$, and $h : U \to R$ a differentiable function.[10] The development in this section relies on this concept, and we therefore restrict it to algebraic graph surfaces. (For more details on the derivation, see Georgiades' thesis.[11])

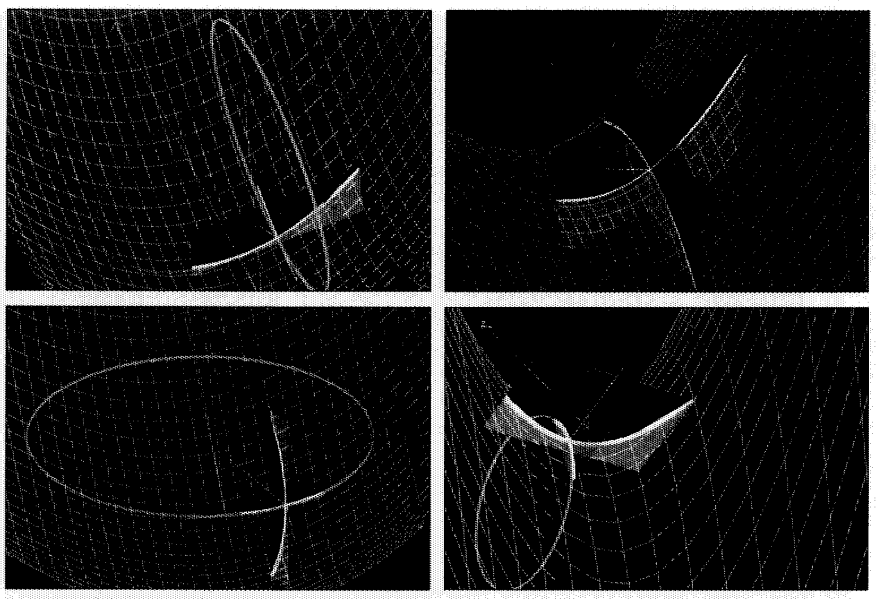

Figure 5. We use osculating circles as icons to depict the curvature of various curves of the surface that pass through the current reference point. The same two sample surfaces show the two coordinate curves and the two principal normal sections at the current reference point.

Using the curvature of parametric curves

A coordinate or parametric curve of an algebraic graph surface is the graph of a polynomial function in one variable. We can construct the polynomial that results in a particular user-input curvature measure at a certain point in an interval (with varying continuity constraints at the ends of the interval). During rendering, the eye easily perceives first derivative discontinuities, so we choose to provide continuity of up to C^2. We derive the definition for the curve segment as follows: Let r be the parametric direction of the curve (either u or v), and let C^{k_r} be the user-defined continuity in that direction. Also, let $\mu = 2k_r + 2$ and $I = [r_0, r_2]$ be the parametric interval of the curve map. Denote the point in I corresponding to the current reference point by r_1 and the input curvature by λ. If we denote the given value of the ith derivative of the curve function at r_0 and r_2 by f_0^i and f_2^i, respectively, with $i = 0 \ldots k_r$, then we compute the coefficients of the polynomial of the new curve $\beta(r) = (r, f(r))$,

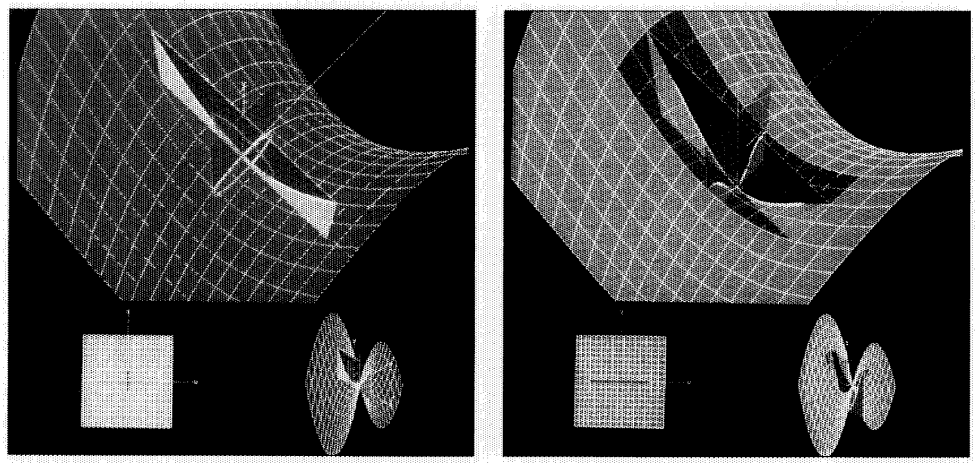

Figure 6. The interface before and after modifying the curvature of the parametric curve aligned with the red

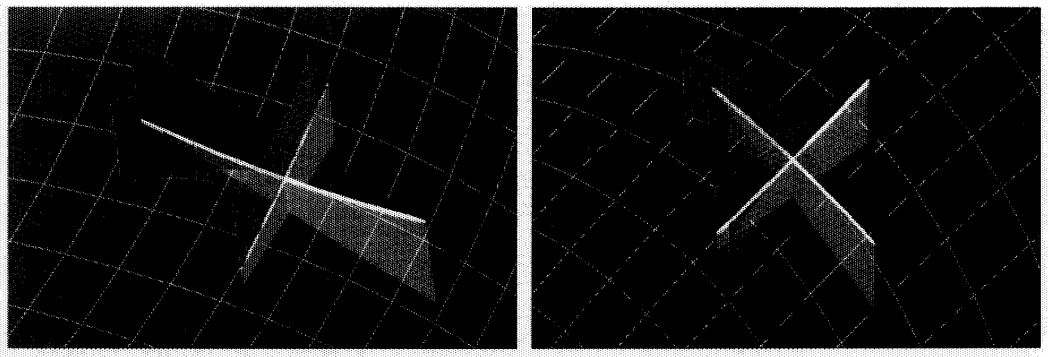

Figure 7. The torsion coil illustrates the torsion of the coordinate curves at the current point. In these nonlinear parameterizations of the paraboloid and the hyperboloid, the intersections of the cross-planes with the surfaces are still plane curves, but they don't coincide with the coordinate curves.

$$f(r) = b_0 + b_1 r + \cdots + b_\mu r^\mu$$

by solving the following system of equations:

$$\frac{f''(r_1)}{(1 + (f'(r_1))^2)^{3/2}} = \lambda \qquad \textbf{(1)}$$

$$\begin{aligned}
b_0 + r_0 b_1 + \cdots + r_0^\mu b_\mu &= f_0^0 \\
b_0 + r_2 b_1 + \cdots + r_2^\mu b_\mu &= f_2^0 \\
b_1 + 2r_0 b_2 + \cdots + \mu r_0^{\mu-1} b_\mu &= f_0^1 \\
b_1 + 2r_2 b_2 + \cdots + \mu r_2^{\mu-1} b_\mu &= f_2^1 \\
2b_2 + \cdots + \mu(\mu-1) r_0^{\mu-2} b_\mu &= f_0^2 \\
2b_2 + \cdots + \mu(\mu-1) r_2^{\mu-2} b_\mu &= f_2^2
\end{aligned} \qquad \textbf{(2)}$$

where we use only the first three equations for C^0, the first five for C^1, and all seven for C^2 continuity. (We get Equation 1 from the expression of curvature given in the sidebar on the next page.)

Surface from curve

Because the boundaries of the graph (defining the local region) are aligned with the parametric directions u and v, which are in turn mapped into the x and z directions in image space, we can derive the surface graph function by interpolation. The parametric domain of $\beta(r)$ is already identified with either u or v, so we denote the other parameter by s the order of continuity in that direction by k_s, and set $r_a = r_0$ and $r_b = r_2$. We generate a number of interpolating curves in the s direction so that they pass through $\beta(r)$ and meet the adjacent patches with C^{k_s} continuity. We use their coefficients to interpolate the surface function. The resulting surface polynomial contains $\beta(r)$ and is of degree $\mu = 2k_r + 2$ when restricted in the r indeterminate. (The derivation of these interpolating curves is discussed in the sidebar on the next page.) We denote these curves by $p_i(s)$, $i = 0 \ldots \mu$, and let ν be their highest degree ($2k_s + 2$ or the degree of either boundary curve at r_a and r_b, whichever is greater).

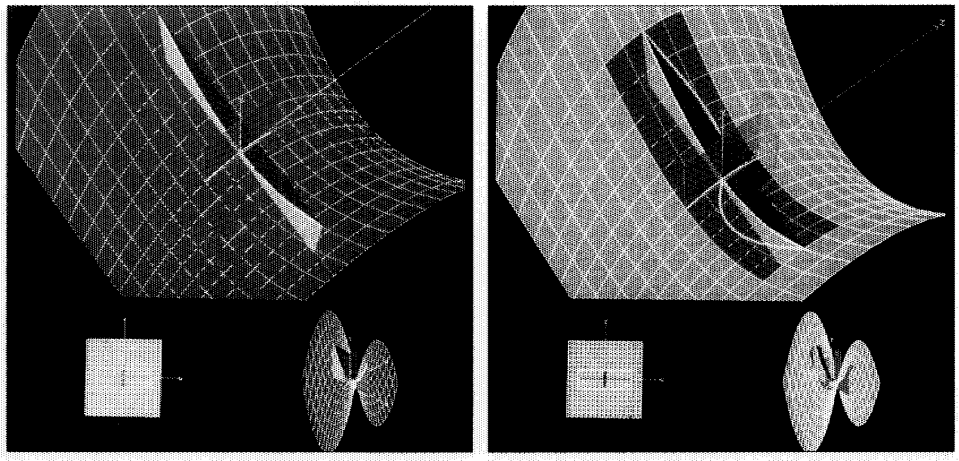

Figure 8. The interface before and after modifying the torsion of the parametric curve aligned with the red cross-plane. We use the twisted line in the interpolation of the new surface function.

Then each p_i is

$$p_i(s) = a_{i_0} + a_{i_1}s + \cdots + a_{i_\nu}s^\nu$$

where $p_0(s)$ and $p_\mu(s)$ are the boundary curves at r_a and r_b, respectively. We write the sought function $h(r, s)$: $[r_a, r_b] \times [s_a, s_b] \rightarrow R$ as

$$h(r, s) = q_0(r) + s(q_1(r)) + s^2(q_2(r)) + \cdots + s^\nu(q_\nu(r))$$

such that each $q_j(r_i)$, $0 \le i \le \mu$, is

$$q_j(r_i) = d_{j_0} + d_{j_1}r_i + d_{j_2}r_i^2 + \cdots + d_{j_\mu}r_i^\mu = a_{i_j}$$

Hence, we produce $h(r, s)$ by computing each q_j, $0 \le j \le \nu$, with the following Vandermonde matrix system. This matrix is guaranteed to have nontrivial solutions for distinct $r_0 \ldots r_\mu$.[12]

$$\begin{bmatrix} 1 & r_0 & r_0^2 & \cdots & r_0^\mu \\ 1 & r_0 & r_0^2 & \cdots & r_0^\mu \\ 1 & r_1 & r_1^2 & \cdots & r_1^\mu \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ 1 & r_\mu & r_\mu^2 & \cdots & r_\mu^\mu \end{bmatrix} \begin{bmatrix} d_{j_0} \\ d_{j_0} \\ d_{j_1} \\ \cdot \\ \cdot \\ \cdot \\ d_{j_\mu} \end{bmatrix} = \begin{bmatrix} a_{0_j} \\ a_{0_j} \\ a_{1_j} \\ \cdot \\ \cdot \\ \cdot \\ a_{\mu_j} \end{bmatrix} \quad (3)$$

Using the torsion of parametric curves

Even though the coordinate curves of a surface given as the graph of a function are planar, we let the user twist such curves locally around the current reference point, thus introducing torsion. We compute a new nonplanar curve, having the input measure of torsion and the same curvature as the planar one. At its ends, this new curve meets the adjacent patches with C^0, C^1, or C^2 continuity. Then we use the curve to generate additional interpolating curves from which we obtain the surface function.

To have nonzero torsion, the curve must vary simultaneously in more than one coordinate. We express a u coordinate curve as $\alpha(r) = (r, f(r), g(r))$ and a v coordinate curve as $\alpha(r) = (g(r), f(r), r)$. In either case, we seek two polynomial functions $f(r)$ and $g(r)$ defined over the interval $I = [r_0, r_2]$, such that the curve α has at a point $r_1 \in I$ a given torsion of σ and a given curvature of λ. The curve α also satisfies the set continuity constraints at the ends of the interval r_0 and r_2 (see Figure 9). Again we set up and solve a system of simultaneous equations, the number of which depends on the order of continuity we choose. Using the same notation, the constraints resulting from the end conditions are $f(r_0) = f_0^0$, $f(r_2) = f_2^0$, $f'(r_0) = f_0^1$, $f'(r_2) = f_2^1$, $f''(r_0) = f_0^2$, $f''(r_2) = f_2^2$, and $g(r_0) = g(r_2) = s$, $g'(r_0) = g'(r_2) = 0$, $g''(r_0) = g''(r_2) = 0$. In all, there are $2(2k_r + 3)$ pieces of information, so the polynomials $f(r)$ and $g(r)$ need to be each of degree $2k_r + 2$. Let their coefficients be

Curvature and torsion of curves

The curvature and torsion of a regular curve $\alpha : I \rightarrow R^3$, not necessarily parameterized by arc length, at a point $r \in I \subseteq R$ can be expressed in general as

$$\kappa(r) = \frac{|\alpha'(r) \times \alpha''(r)|}{|\alpha'(r)|^3}$$

$$\tau(r) = -\frac{(\alpha'(r) \times \alpha''(r)) \times \alpha'''(r)}{|\alpha'(r) \times \alpha''(r)|^2}$$

The curvature of a u and v coordinate curve of a surface given as the graph of a function $X(u, v) = (u, h(u, v), v))$ is, respectively,

$$\kappa(r) = \frac{h_{uu}(r, c)}{(1 + (h_u(r, c))^2)^{3/2}} \quad \text{and} \quad \kappa(r) = \frac{h_{vv}(c, r)}{(1 + (h_v(c, r))^2)^{3/2}}$$

where

$$h_u = \frac{\partial h}{\partial u}, \; h_v = \frac{\partial h}{\partial v}$$

and

$$h_{uu} = \frac{\partial^2 h}{\partial u \partial u}, \; h_{vv} = \frac{\partial^2 h}{\partial v \partial v}, \text{ and } c \text{ is a constant.}$$

Deriving interpolating polynomials

To derive the interpolating curves needed to compute the surface function, intersect each curve with the curve computed from the input curvature and torsion while maintaining C^{k_s} continuity at its boundaries. The resulting system of equations is quite similar to Equations 2, where we substitute the variable r by s and μ by $\nu = 2k_s + 2$. Write each polynomial as

$$p_i(s) = a_{i_0} + a_{i_1}s + \cdots + a_{i_\nu}s^\nu$$

The point of intersection of $p_i(s)$ with the given coordinate curve (denoted $\alpha(r)$ and $\beta(r)$, in the article) provides the last equation needed to complete the system. The matrix form in the case of $k_s = 2$ is

$$\begin{bmatrix} 1 & s_0 & s_0^2 & s_0^3 & s_0^4 & s_0^5 & s_0^6 \\ 1 & s_1 & s_1^2 & s_1^3 & s_1^4 & s_1^5 & s_1^6 \\ 1 & s_2 & s_2^2 & s_2^3 & s_2^4 & s_2^5 & s_2^6 \\ 0 & 1 & 2s_0 & 3s_0^2 & 4s_0^3 & 5s_0^4 & 6s_0^5 \\ 0 & 1 & 2s_2 & 3s_2^2 & 4s_2^3 & 5s_2^4 & 6s_2^5 \\ 0 & 0 & 2 & 6s_0 & 12s_0^2 & 20s_0^3 & 30s_0^4 \\ 0 & 0 & 2 & 6s_2 & 12s_2^2 & 20s_2^3 & 30s_2^4 \end{bmatrix} \begin{bmatrix} a_{i_0} \\ a_{i_1} \\ a_{i_2} \\ a_{i_3} \\ a_{i_4} \\ a_{i_5} \\ a_{i_6} \end{bmatrix} = \begin{bmatrix} f_0^0 \\ f_1^0 \\ f_2^0 \\ f_0^1 \\ f_2^1 \\ f_0^2 \\ f_2^2 \end{bmatrix}$$

Its determinant is $\det = 4(s_0 - s_1)^3(s_0 - s_2)^9(s_2 - s_1)^3$, hence the matrix is invertible for $s_0 < s_1 < s_2$.

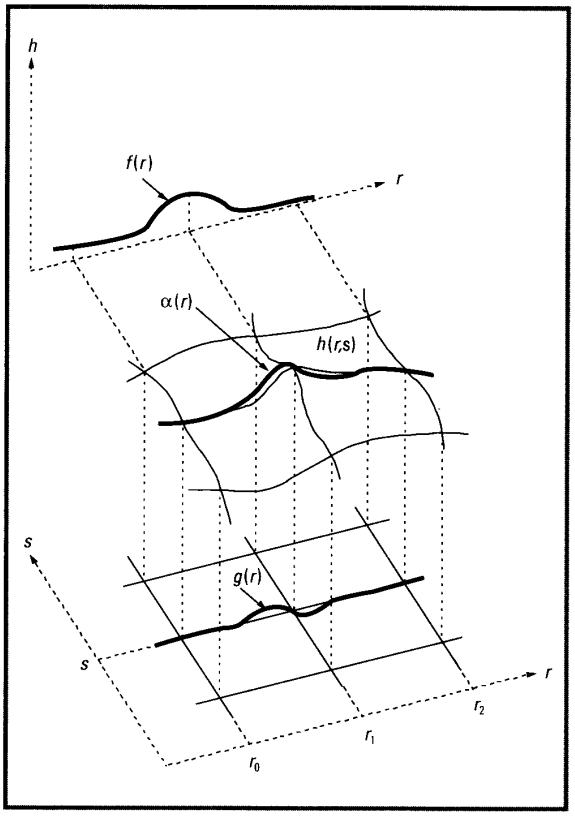

Figure 9. The curve $a(r) = (r, f(r), g(r))$ has torsion $\sigma \neq 0$ at $r = r_1$ and zero torsion at $r = r_a$ and $r = r_b$.

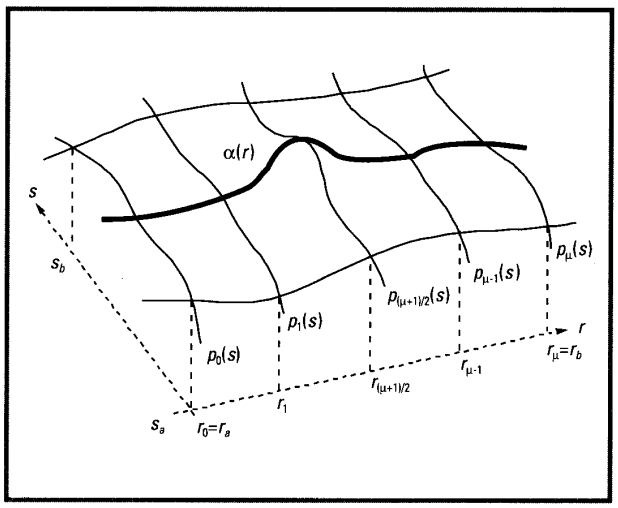

Figure 10. The plane curves $p_i(s)$ cross over $a(r)$ and meet the boundaries with the continuity set in the s direction.

$$f(r) = b_0 + b_1 r + b_2 r^2 + \cdots + b_\mu r^\mu$$
$$g(r) = c_0 + c_1 r + c_2 r^2 + \cdots + c_\mu r^\mu$$

with $\mu = 2k_r + 2$. We can summarize these conditions by two sets of equations. The set representing the constraints on $f(r)$ is identical to Equations 2. We can use the same equations for $g(r)$, by substituting c for b, s for f_h^0, and 0 for f_h^1 and f_h^2, $h = 0, 2$. The system is completed by the curvature and torsion equations, derived from the expressions in the "Curvature and torsion of curves" sidebar.

$$\frac{(f''(r_1)g'(r_1) - f'(r_1)g''(r_1))^2 + (g''(r_1))^2 + (f''(r_1))^2}{((f'(r_1))^2 + (g'(r_1))^2 + 1)^3} = \lambda^2 \quad \textbf{(4)}$$

$$\frac{f''(r_1)g'''(r_1) - f'''(r_1)g''(r_1)}{(f''(r_1)g'(r_1) - f'(r_1)g''(r_1))^2 + (g''(r_1))^2 + (f''(r_1))^2} = \sigma \quad \textbf{(5)}$$

Once we compute the curve $\alpha(r)$, $r_0 \le r \le r_2$, we fit it on a surface patch whose boundaries are aligned with the coordinate directions, as defined by the extents of the cross-planes. We obtain interpolating curves as in the "Using the curvature of parametric curves" section. These curves should pass through $\alpha(r)$ (see Figure 10). Then use their coefficients as the entries of the Vandermonde matrix (Equation 3), which computes the final patch function (see Figure 11). Observe that $\alpha(r)$ is no longer a coordinate curve of the new surface patch. Instead, we use it to compute a patch that approximately contains α, and its parametric directions are aligned with the coordinate axes. How good the approximation is—and hence how high the degree of the function in the r indeterminate—depends on the number of interpolating curves computed in the s direction.

Using the Gaussian curvature

As users modify the Gaussian curvature at a point on the surface, the program derives a new surface definition by reducing the problem to modifying the curvature of a coordinate curve. The Euler formula given in the sidebar on the next page relates the normal curvature to the principal curvatures at some point on the surface. Replacing the normal curvature with its definition with respect to the curvature of a coordinate curve yields

$$\kappa \cos\theta = k_1 \cos^2\varphi + k_2 \sin^2\varphi$$
$$\kappa = \frac{k_1 \cos^2\varphi + k_2(1 - \cos^2\varphi)}{\cos\theta} \quad \textbf{(6)}$$

where κ is the curve's curvature, θ is the angle between the surface normal N and the curve normal n, and all other quantities are as defined earlier. Note that $\cos\theta = N \cdot n$ is never zero

on graph surfaces, as can be shown by direct substitution. By using the values of the maximum and minimum normal curvatures (which we've already computed or the user has set), we can get a value for the desired curvature for the coordinate curve. Hence, as described under "Using the curvature of parametric curves," we can use it to calculate a new patch graph function.

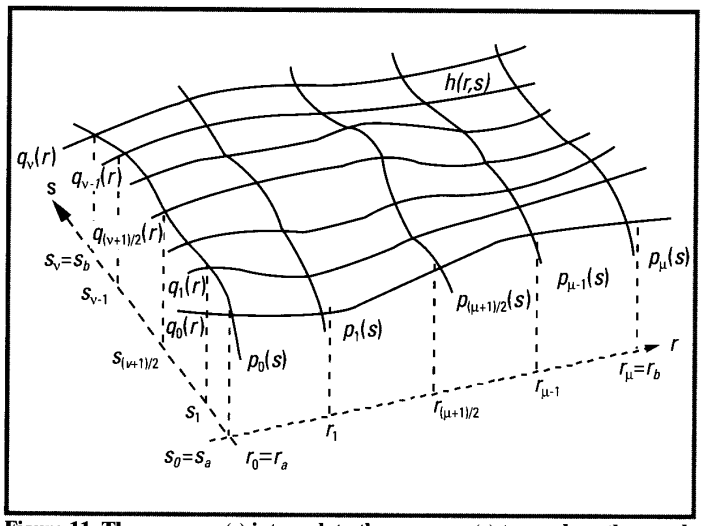

Figure 11. The curves $q_j(r)$ interpolate the curves $p_i(s)$ to produce the graph function $h(r, s)$.

Conclusions

The interactive tools we described in this article provide a new, powerful means for locally manipulating curved surfaces. By depicting key intrinsic measures of the geometry of a surface, these tools give users an intuitive, tangible handle on the surface. We present them as a complementary capability to the well-established techniques of modeling by means external to the surface.

Specifically, a 3D cursor consisting of a pair of intersecting planes lets users traverse the surface and delimit a local region of interest. The osculating-circle and torsion-coil icons depict the curvature and torsion of the curves of intersection of these planes with the surface. The osculating circle shows either curve's curvature at the point of intersection with the other curve, drawn at its exact geometric location. Literally twisting the cross-plane away from the osculating plane illustrates the torsion at that point.

Since our approach is local, it differs from previous methods that treat the entire surface. As a system of examining a surface, the 3D cursor, the osculating circles, and the torsion coil let users freely traverse the surface and toggle these icons on and off with the surface normal. Thus, while the entire surface is rendered in some uniform format, this interface reveals surface characteristics that are normally difficult to see. The advantage is real-time display, but the resulting image doesn't provide a macroscopic view of the surface's intrinsic measures, like color curvature maps and contour lines.

With the handles that lie right on the surface, users can manipulate any arbitrary point on the surface and, with the lengths of the cross-planes, users can define a local region of interest of arbitrary size around it (within the constraints of regularity and locality). Spline-based methods only let users manipulate control points or points of maximal influence, and the local regions of interest, when editable, must correspond to some partition of the parametric space based on the control vertices. In our interface, once users define the local region, they can employ the display tools we developed to modify the geometric measures the tools represent. The new values of these measures yield new curves from which the system interpolates a new surface patch. Since we restricted our work to local neighborhoods, the logical next step is to extend the implementation to edit boundary curves. ❑

Acknowledgments

This work was done while Príamos N. Georgiades was a graduate student at Cornell University. The research we describe here was funded in part by a grant from the National Science Foundation entitled "Interactive Input and Display Techniques" (#DCR8203929). We implemented our research on equipment generously donated by Hewlett-Packard. We thank James E. West of the mathematics department of Cornell University for his help with this work. We also thank the other Cornell faculty, staff, and students in the computer graphics program who worked on the 3D modeler in whose context this program was developed: Roy Hall, Mimi Bussan, Len Wanger, Mark Reichert, Paul Wanuga, Kathy Kershaw, Ricardo Pomeranz, and Peter Pruyn.

Normal curvature in terms of principal curvatures

The principal directions e_1 and e_2 at a point p of a surface S form an orthonormal basis of the tangent plane at p. Any vector v lying on the tangent plane can be expressed as

$$v = e_1 \cos\varphi + e_2 \sin\varphi$$

where φ is the angle between v and e_1. The normal curvature at p in the direction of v is related to the principal curvatures by the expression

$$k_n = k_1 \cos^2\varphi + k_2 \sin^2\varphi$$

known as the *Euler formula*.[10]

References

1. G.E. Farin, *Curves and Surfaces for Computer Aided Design,* Academic Press, Boston, 1988.
2. J.M. Beck, R.T. Farouki, and J.K. Hinds, "Surface Analysis Methods," *IEEE CG&A,* Vol. 6, No. 12, Dec. 1986, pp. 18-35.
3. J.D. Foley et al. *Computer Graphics: Principles and Practice,* Addision-Wesley, Reading, Mass., 1990.
4. W. Tiller, "Rational B-Splines for Curve and Surface Representation," *IEEE CG&A,* Vol. 3, No. 6, Sept. 1983, pp. 61-69.
5. J. Bloomenthal and B. Wyvill, "Interactive Techniques for Implicit Modeling," Proc. 1990 Symposium on Interactive 3D Graphics, in *Computer Graphics,* Vol. 24, No. 2, March 1990, pp. 109-116.
6. D.R. Forsey and R.H. Bartels, "Hierarchical B-Spline Refinement," *Computer Graphics* (Proc. Siggraph), Vol. 22, No. 4, Aug. 1988, pp. 205-212.
7. A.H. Barr, "Global and Local Deformations of Solid Primitives," *Computer Graphics* (Proc. Siggraph), Vol. 18, No. 3, July 1984, pp. 21-30.
8. T.W. Sederberg and S.R. Parry, "Free-Form Deformation of Solid Geometric Models," *Computer Graphics* (Proc. Siggraph), Vol. 20, No. 4, Aug. 1986, pp. 151-160.
9. M.S. Casale, "Free-Form Solid Modeling with Trimmed Surface Patches," *IEEE CG&A,* Vol. 7, No. 1, Jan. 1987, pp. 33-43.
10. M.P. DoCarmo, *Differential Geometry of Curves and Surfaces,* Prentice-Hall, Rio de Janeiro, 1976.
11. P.N. Georgiades, *Interactive Methods For Manipulating the Intrinsic Geometry of Curved Surfaces,* master's thesis, Cornell University, Ithaca, N.Y., Jan. 1991.
12. W.H. Press et al., *Numerical Recipes in C,* Cambridge University Press, Cambridge, England, 1988.

Príamos N. Georgiades is a software engineer at CrystalGraphics in Santa Clara, California. His primary area of interest in computer graphics is 3D design and modeling at the level of interface design and its geometrical applications (using curved surfaces in particular).

Georgiades received his BS in architecture and MS in computer graphics at Cornell University. Between degrees, he practiced as an intern architect for one year at the Hillier Group in Princeton, New Jersey.

Donald P. Greenberg is director of the computer graphics program at Cornell University and the founder of Cornell's Computer Aided Design Instructional Facility. His research interests include radiosity and hidden-surface algorithms, geometric modeling, color science, and realistic image generation.

Greenberg received his PhD from Cornell in 1968. He is a member of ACM Siggraph and IEEE.

Georgiades can be reached at CrystalGraphics, 3350 Scott Blvd., #29, Santa Clara, CA 95054.