PROCEDURAL SPLINE INTERPOLATION IN UNICUBIX

Carlo H. Séquin

Computer Science Division
Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720

ABSTRACT

UNICUBIX is an extension to the Berkeley UNIGRAFIX modeling and rendering system.[1] The original system, restricted to polyhedral object, is extended by permitting cubic curves in place of the previous linear edges between pairs of vertices, and triangular or quadrilateral patches between such curved borders. Where desired, the patches are joined together with G^1-continuity using procedures that guarantee locality of control.

A sequence of procedural steps determines first the vertex normals, then the Bézier control points for the curved edges, and finally the internal control points for all patches. These steps use an intuitive geometric approach to fitting spline curves and surfaces through the given set of vertices. A variety of rules and procedures makes it possible to provide pleasing default values for even difficult situations. There are no restrictions on the topology of the original polyhedral net defining the shape of the object.

Proceedings of the Third USENIX Computer Graphics Workshop

1. INTRODUCTION

1.1. Motivation

Berkeley UNIGRAFIX[1] started as a polyhedral modeling and rendering system[2] that could handle only objects bounded by planar faces — although of arbitrary complexity, with multiple contours and holes. However, most users of graphics systems soon tire of polygonal or polyhedral shapes and want at least rounded corners or, better yet, smooth looking continuous objects. Furthermore, CAD/CAM systems need to provide rounded surfaces since most mechanical parts are not entirely polyhedral; they contain circular holes, rounded edges, and curved fillets where two parts join.

In the creation of such objects, the user wants a simple-to-use interface that gives good results with an obvious way of specifying the round parts of the model. To stay within the framework of the Berkeley UNIGRAFIX system, we have looked at various possibilities for extending its boundary representation to curved objects, rather than taking a CSG approach with intrinsically round primitives. We have concentrated our attention on interpolating splines since they fit directly into the UNIGRAFIX paradigm where objects are described by vertices positioned in space and by edges and surface patches stretched between these points. In the modeling of technical parts, interpolating splines are also advantageous since often a few points, or even a whole profile, of the surface to be constructed are known precisely.

Local control over the surface shape is another desirable property. Often only a small part of a surface needs to be changed; one would then like the bulk of it to remain unaffected. We have tried to find interpolating splines that exhibit strong locality and integrate well with objects that are partially composed of flat surfaces. We want to avoid the overhead of representing all flat parts of an object also as spline surfaces.

These ideas are being implemented in a new modeling and rendering system called UNICUBIX. Its aim is to provide an environment for the non-mathematical user for producing geometrical objects of arbitrary shape and topology.

1.2. "Pleasing" Splines

Much work has been done in the area of interpolating splines and in the area of fitting curves and surfaces through a given set of points. The reader is referred to appropriate textbooks.[3-5]

Two pieces of work exemplify our own goals and have been particularly inspiring to the work described in this paper. Vaughan Pratt's work on conics for font definition[6] demonstrates how much can be achieved with using even very simple primitives. By a judicious choice in the selection of control elements, the individual spline patches can be made to join together in a very smooth and pleasing looking manner. Since our own goal is to provide smooth curves and surfaces in R^3 space, our lowest level primitives are cubic polynomials, rather than conics.

John Hobby's work on "Smooth, Easy to Compute Interpolating Splines"[7] demonstrates that very good results can be obtained by deviating from the strict formulation of natural splines. His curves, satisfying a mock-curvature constraint, give much more rounded and pleasing looking curves.

In the same spirit, we plan to achieve pleasing results with a minimum of

computational and conceptual overhead. We developed an approach primarily based on geometrical, rather than algebraic reasoning. The spline curve and surface patches are derived from a sequence of procedural steps that may include rules that take the special situation of a particular patch into account. This is more flexible than an approach that applies a fixed arithmetic expression to a given set of control points.

Our current implementation uses cubic Bézier curves for spline curve interpolation and for the boundaries between patches. It uses quartic triangular or bicubic quadrilateral patches that join with tangent-plane continuity to provide a smooth interpolating surface.

1.3. Paper Overview

In the first part of the paper (Sections 2 - 4) we will introduce the concept of our procedural step-by-step approach with the example of constructing planar interpolating curves through a given set of points. In Section 2 we first review the basics of spline segments. Then we present our procedural approach for the case of planar curves composed from cubic spline segments (Section 3). This procedure has two phases: first at each vertex the tangent direction of the curve is chosen; then the additional two Bézier points are chosen on these tangent lines. In Section 4 we give an overview how this approach can be extended to cases that go beyond the requirements arising in the construction of smooth shapes in UNICUBIX.

The second part of the paper (Sections 5 - 9) deals with surfaces in R^3 and starts with an overview of the approach (Section 5) followed by a review of the patches to be used (Section 6). In Section 7 we discuss the constraints that guarantee G^1-continuity.[8,9] For surfaces, our procedural approach has several phases (Section 8). First, the surface normals at all vertices are chosen. Second, the shape of the curved edge segments between the given vertices are determined. Third, the behavior of the tangent plane or the surface normal along the seams is defined. Then patches are fit into that network of curved boundaries by choosing interior control points that match the desired behavior on the seams and ensure G^1-continuity with the neighboring patches. In Section 9 we touch upon some rendering issues and review possibilities for creating Bézier rather than Gregory patches.

For more details about the UNICUBIX system and its implementation or about the underlying modeling issues the reader may request our technical reports on the subject[10,11] and should watch for forthcoming papers.

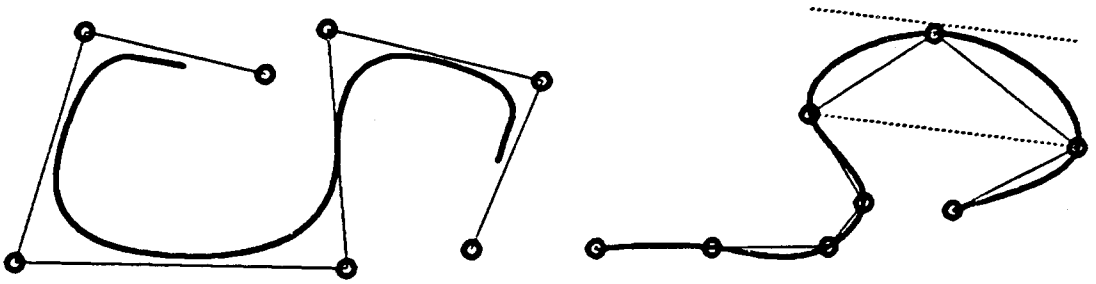
2. SHORT REVIEW OF SPLINES

This section briefly reviews a few concepts that are crucial for the understanding of our approach. More information can be found in introductory texts on geometric modeling, e.g.[4,5]

2.1. Approximating and Interpolating Splines

It is normally useful to distinguish between splines that interpolate the given vertices, and splines that only approximate these controlling points. A classical representative for an approximating curve is the B-spline. The coordinates of its control points are combined in a polynomial expression to define parametrically the coordinates of a point on

the curve as a function of the given parameter value. For the example of the cubic B-spline shown in Fig. 1a, each curve point is influenced by only four control points. This provides locality of control: if any control point is changed, at most four sections of the curve will be affected. If coinciding control points are avoided, the resulting curves are smooth. The cubic B-spline is twice differentiable and is thus said to be C^2-continuous. C^2-continuity automatically implies G^2, as well as C^1 and G^1-continuity. Another useful property is that the curve lies entirely within the convex hull of its control points.



(a) (b)

Figure 1: *Approximating (a) and interpolating (b) splines*

A different polynomial formulation can lead to an interpolating curve. Fig. 1b shows the example of a Catmull-Rom spline.[12,13] Again C^2-differentiability can be guaranteed with a suitable choice of weighting coefficients, but obviously the convex hull property holds no longer. While the B-splines can easily be made to have a very smooth appearance, the interpolating splines often show unnecessary wiggles and exaggerated asymmetric bulges, particularly when the control polygon has acute angles and sides of widely varying sizes.

In both cases it is often difficult for the user who places the control vertices to predict what the resulting curve is going to look like. In the case of B-splines, one does not know through what points the curve will go; and for the interpolating splines it is often difficult to guess the direction at which the curve passes through a given vertex. For the latter problem, some auxiliary construction may help. In the default formulation the tangent direction at an interpolated vertex can be found from the direction of the chord connecting the two nearest neighbors (see dotted line in Fig. 1b).

2.2. Bézier Segments

An alternative way of obtaining a smooth curve through a given set of points is to join individual curve segments together. The example shown in Fig. 2 uses three cubic Bézier segments. These are polynomial curves segments of degree 3 defined by 4 control points each. The first and last point give the start and end of the curve segment. Furthermore, the first two points together define the starting "velocity" and thus the tangent at the starting point. The last two points similarly give the ending velocity and tangent.

To join these segments together with tangent (G^1-) continuity,[8,9] requires that the last control point of one segment be shared with the first control point of the subsequent segment. Furthermore, the ending velocity vector of the first segment must be collinear

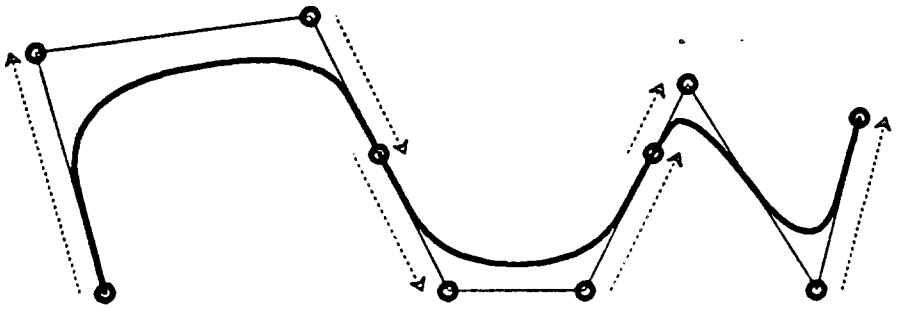


Figure 2: *Chain of Bézier-Segments*

with the starting velocity vector of the second segment. If, in addition, the velocities are equal in magnitude, the composite curve has C^1-continuity. But to give a smooth appearance, G^1-continuity is often sufficient.

Thus the task of producing a smooth curve is to find suitable control points for all segments in accordance with the above constraints.

3. RULE-BASED, GEOMETRICAL APPROACH TO SMOOTH CURVES

As an introduction to our multi-phase approach to the generation of smooth objects, we start with a procedure to obtain a smooth, pleasing looking curve through a sequence of vertices. In our demonstration case, the curve is composed from a sequence of cubic Bézier segments.

In a first phase, tangent directions are chosen at each vertex based on local information; in the case presented here, only the nearest neighbor vertices are taken into account. The second step is to place the inner Bézier control points on these tangent lines at a suitable distance from the given vertex. This determines the "velocities" of the parameterized point describing the curve on either side of the vertex and implicitly controls the bulge of the curve. G^1-continuity is ensured as long as the Bézier points on either side of the vertex lie on the same tangent line; for C^1-continuity they also would have to lie at equal distances from the vertex.

These selections are not simply made with a fixed algebraic expression involving the coordinates of the vertices, but may be based on several different computational procedures. The appropriate procedures are picked according to a set of rules that depend, for instance, on the constellation of the neighboring vertices. In particular, the end-vertices of a curve, or a cluster of collinear control points would be treated differently from other interior vertices.

3.1. Default Constructions for Tangent Direction

As shown in Fig 1b, Catmull-Rom splines derive the tangent direction of the curve at one of the interpolated vertices from the direction of the chord of the two nearest neighbor vertices. This may lead to a lopsided bulge if the two sides of the control polygon joined at this vertex are very asymmetric (Fig. 3a).

Another possible construction that avoids this problem simply defines the tangent

direction as being perpendicular to the angle bisector at this vertex (Fig. 3b). This method has the disadvantage that the tangent direction is completely insensitive to the lengths of the sides of the control polygon. Thus if the control polygon is a rectangle of any aspect ratio, the tangents will always pass through the corners at an angle of 45 degrees.

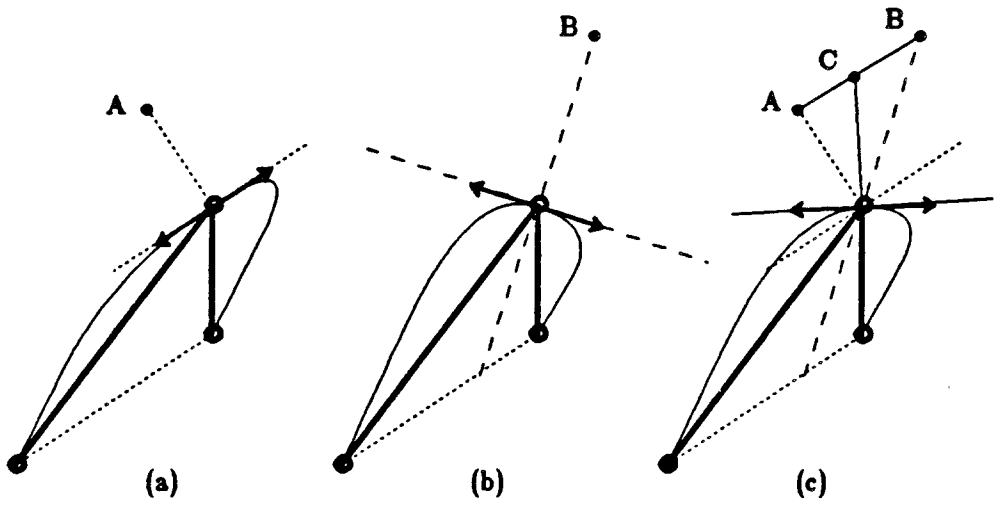


Figure 3: *Three ways of finding the tangent direction*

A good compromise can be reached by choosing the arithmetic average of the two tangent directions determined by the two methods above (Fig. 4c). The actual construction takes a normal vector of a length equal to half the chord to produce point A. It generates point B by reflecting the intersection point of the angle bisector with the chord about the central control vertex. Point C then represents the averaged "vertex normal"; the tangent direction is perpendicular to it. The weighting of the extreme solutions (a), (b) can occur in any desirable ratio. This ratio can affect the behavior of the curve locally or in a global manner, depending on whether the weighting ratio is set for each vertex individually or globally.

3.2. Default Selection for Velocity Magnitude

Once the tangent direction at each vertex has been determined, the next step is to select the velocities on either side of each vertex. This is equivalent to placing the inner two Bézier control points somewhere on the established tangent lines. For G^1-continuity it is not necessary that the velocities on either side of the vertex be the same. They can thus be influenced individually by the respective distances of the nearest neighbors. Again the most pleasing results over a wide range of constellations is obtained as a compromise between two extreme approaches (Fig. 4).

In Figure 4a the velocity is simply made proportional to the length of the underlying side of the control polygon. In Figure 4b both velocities on either side of the vertex are determined solely by the length of the chord between the nearest neighbors. Finally Figure 4c uses the geometric mean of the velocities determined in the above manner. In our view it gives very pleasing results in practically all cases examined.

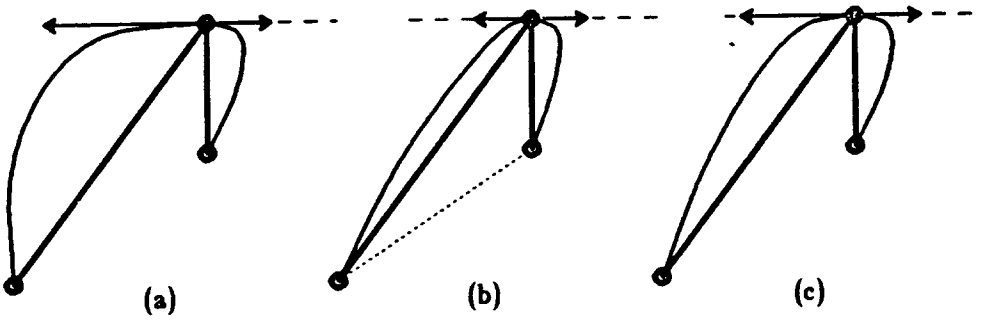


Figure 4: *Three choices for the velocity magnitude*

3.3. Shape Parameters

The procedures discussed above contain parameters that have been set to particular default values that seem to give suitable results in most cases examined. One such parameter is the weighting of the two extreme solutions for the choice of the averaged vertex normal direction (Fig. 3c). The value of this weighting factor will affect the general behavior of the tangents at the interpolated vertices. Another parameter is a multiplying factor on the calculated velocities. A bigger factor will produce more bulge of the curve segment between two vertices, while a smaller value will lead to a shorter segment that connects the two vertices more directly.

If these values are varied globally, the overall character of the whole curve can readily be affected in a uniform manner. Thus the power of shape parameters, developed for the more traditional splines,[14] can also be used in our procedural approach.

4. TOWARDS A RICHER RULE SET

Two simple procedures have been presented above to find the tangent directions and velocities at each vertex. The given rules are not sufficient to handle all constellations of vertices appearing in an unconstrained design in a satisfactory manner. To obtain "pleasing" results in all cases, the set of procedures and the set of rules to choose between them need to be enlarged.

4.1. Examples of More Sophisticated Rules

Special problems arise at the end-vertices of open curves and also when three or more control vertices are collinear, perhaps indicating that the user really wants a straight section through these points. Examples of rules that might deal with such situations are:

- If the control polygon does not have any inflections, then the interpolating spline should not have any inflections either.
- Three collinear vertices might indicate that the user really would like this piece to be straight; except when there are inflections, such as a sine wave defined by a triangular wave shape.
- When the control polygon has a very acute angle, the tangent of the curve should be almost perpendicular to the angle bisector.

- The vertex normal should never fall outside the extended lines of the control polygon at that vertex.

In the framework of the curved edges of UNICUBIX objects such problems normally do not arise. A more detailed discussion of these problems thus will be presented at some other time.[15] However, problems of a similar nature do occur in the decision steps for the creation of smooth surfaces in UNICUBIX (Section 8).

4.2. Measure of Quality, Desired Properties

The introduction of such "arbitrary" rules naturally raises the question: how should the results be evaluated?

The main goal is to obtain a "pleasing" behavior; therefore the users of the system will be the ultimate judges. We have made various attempts to obtain information from potential users. At the USENIX workshop the audience was asked to draw examples of "pleasing" interpolating splines through a given set of test vertices. One of the test shapes, interpolated with a Catmull-Rom spline (a) and with the simple compromise methods discussed above (b) is shown in Fig. 5.

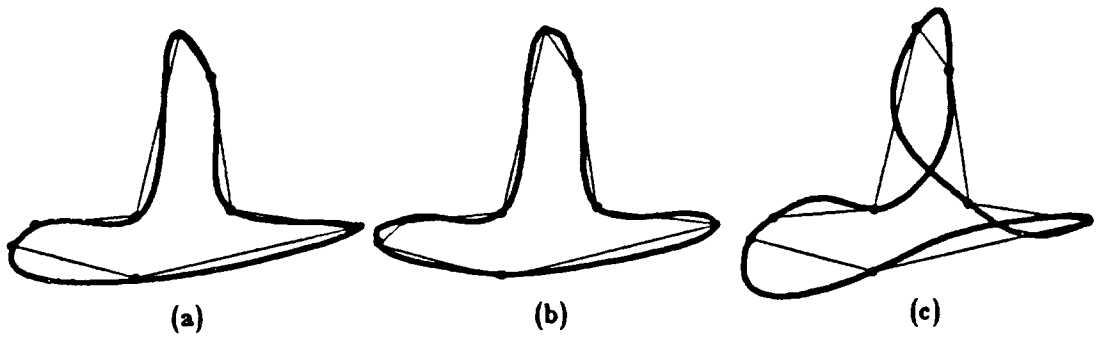


Figure 5: *Test pattern to find pleasing interpolating splines.*
A traditional spline (a), our procedural solution (b) and an artistic solution (c)

The results from the audience can typically be split in two categories: the "well-behaved" and the "artistic". The well-behaved solutions fall into a narrow range and are in good agreement with our solution displayed above (Fig. 5b). The other group displays exaggerated "loopy" curves (Fig. 5c) — often not even passing through the vertices in the sequence indicated by the control polygon — and they vary too strongly to indicate any particular trend. There is strong suspicion that these "artists" were more motivated to be original than to seriously try to find some form of minimal spline through the given vertices.

Obviously, the definition of the "best" or "most suitable" spline is application dependent. A smooth surface to be fabricated on a milling machine may be different from a surface whose primary objective is to perform well in a wind tunnel. One may have to define several application-dependent rule sets or must make the rule sets modifiable by the user. Furthermore, any practical interactive design system must allow the user to move the individual control points in areas where the default solution curve is insufficient.

5. OVERVIEW OVER SURFACE INTERPOLATION IN 3D

An equivalent procedural approach can be taken to make pleasing surfaces that interpolate a given set of points in 3D. This is an outline of the procedural steps:

1. Define the surface by placing vertices in R^3 and by using edges to indicate how the latter should be topologically linked by the surface.
2. Select the *vertex normals* - the normals of the surface at the vertices.
3. Place the Bézier points in the tangent planes at the vertices. The goal is to replace the original edges with suitable cubic curves that will lie in the surface to be constructed.
4. Define the surface normal or the tangent plane along the chosen cubic curves or - equivalently - determine the behavior of the surface seams along the borders between the various patches.
5. Fit into each mesh surrounded by cubic boundaries a Gregory patch that blends with its neighbors with G^1-continuity.
6. If desirable, subdivide the Gregory patches into Bezier patches.

The techniques to join patches with G^1-continuity have been strongly influenced by the work by Farin[16,17] and by Chiyokura.[18,19] Each step in the above sequence ensures that the constraints applicable up to this point are met, so that subsequent steps do not run into inconsistencies. Again, variations in the procedures can produce objects of varying overall character.

6. BEZIER AND GREGORY PATCHES

As pointed out earlier, we use quadrilateral as well as triangular patches. This allows us to handle nets of arbitrary topology and to find a surface description that easily matches the natural symmetries of the intended shape. In this section we briefly review these patches.

6.1. Quadrilateral Patches

One of the basic surface elements used in UNICUBIX is a quadrilateral bicubic vector-valued tensor-product Bézier patch. Such a patch is defined with sixteen control points, 4 in the corner vertices, 2 along every edge, and 4 interior points (Fig. 6).

All but the 4 interior points are already given by the time we try to fit a patch into the mesh of cubic boundaries. It turns out that in the general case the four interior points do not offer enough freedom to provide G^1-continuity across all four borders. Instead we use a bicubic Gregory patch [20] which provides twice as many interior control points (Fig. 7). Every interior point is split into a pair of points where one each controls the behavior of the patch on one of the two borders coming together in the associated corner. The influence of these two points is linearly interpolated as surface points gain more distance from one border and approach the other. For instance, in the corner where the borders $u=0$ and $v=0$ come together, the weighting of the the Gregory point pair is given simply by the ratio of the parameter values of u and v respectively. Because of these split control points, the Gregory patch does not have the twist constraints that one finds in the Bézier patches. On the other hand, it is no longer a purely polynomial expression in the parameters u and v.

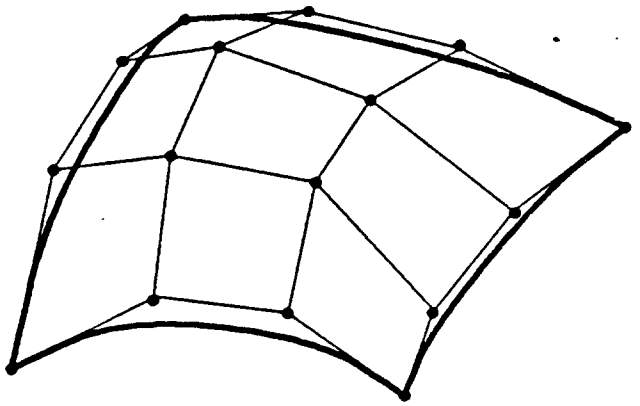


Figure 6: *Bicubic Bézier patch with its 16 control points*

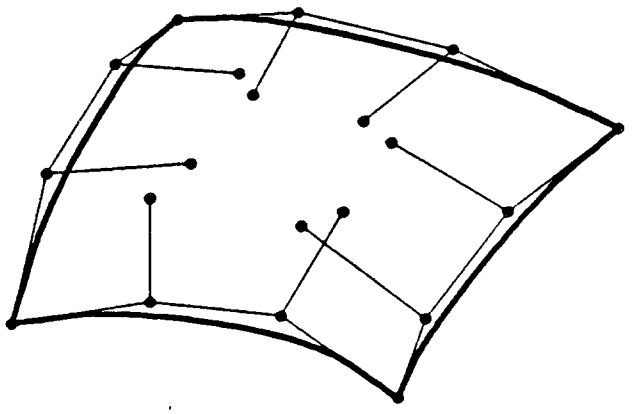


Figure 7: *Bicubic Gregory patch with its 20 control points*

6.2. Triangular Patches

Triangular patches are best described in barycentric coordinates. In this representation a point in the plane of the triangle is expressed as a linear combination of the three vertex coordinates; the three coefficients always sum to 1.

A picture of a cubic patch is shown in Fig. 8a. Such a patch has the same number of control points on the edges as the bicubic patch discussed above, but it has only a single interior control point. To provide two interior control points along each edge as in the above case, we have to use a quartic triangular patch which provides a total of three interior control points (Fig. 8b).

Furthermore, to permit us to set two interior control points independently from the constraints on each boundary, we need to split these control points as in the quadrilateral case. This leads to a triangular version of the Gregory patch (Fig. 9). Again, in the expressions that describe the surface in parametric form, the pairs of control points enter as a linearly interpolated expression that is weighted with the ratio of the two relevant parameters of the barycentric coordinate space.

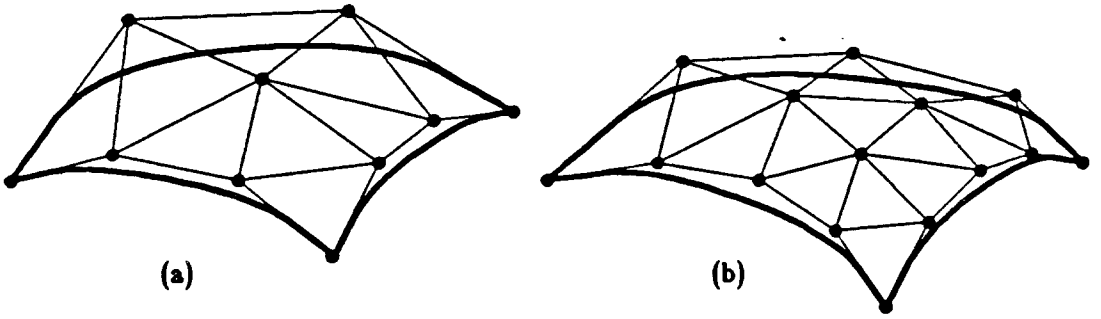


Figure 8: *Cubic (a) and quartic (b) triangular Bernstein patches.*

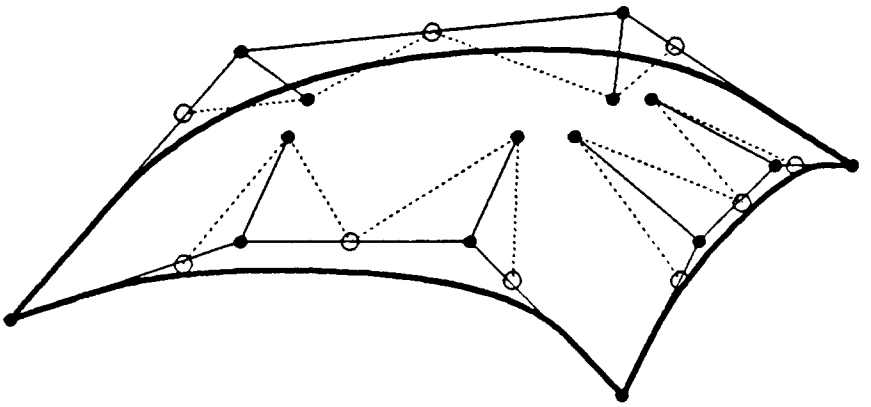


Figure 9: *Triangular quartic Gregory patch with its 18 control points*

Since the quartic Bernstein or Gregory patches have quartic borders with one more control vertex along their sides than the given cubic borders, we need to *degree elevate* our cubic borders to degree four. This is a simple process. The three new control points (○) can be found as simple linear expressions of two of the old control points (•) (Fig. 10). Intuitively they can be viewed as lying at "equidistant" intervals on the control polygon — if the sides of the control polygon are assumed to be of equal length.

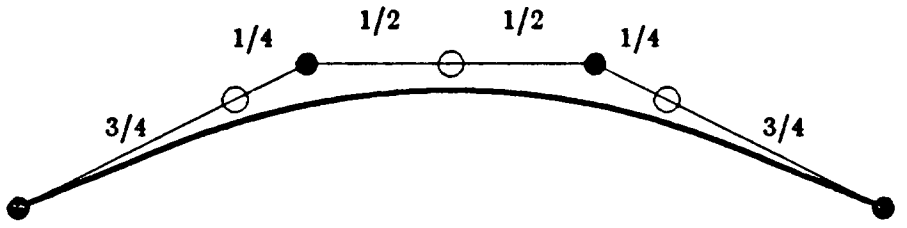


Figure 10: *Degree elevation from a cubic* (●) *to a quartic* (○) *control polygon*

With these descriptions, triangular as well as quadrilateral patches can be spliced readily to the given cubic borders (Fig. 11). As we will see in the next section, when we

determine the two interior patch control points along a boundary, we don't even have to know whether this is for a quartic triangle or for a bicubic quadrilateral patch.

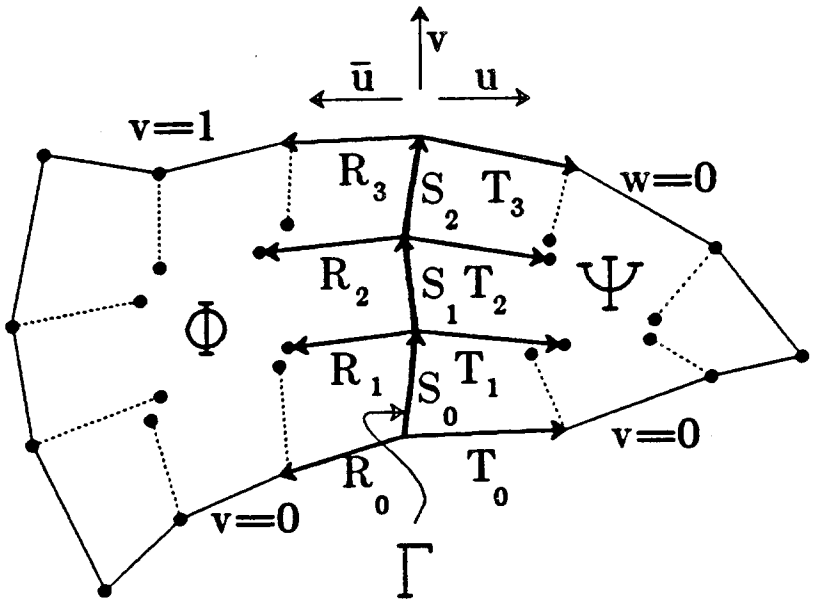


Figure 11: *Joining a quadrilateral and a triangular patch*

7. JOINING PATCHES WITH TANGENT-PLANE CONTINUITY

This section reviews several mathematical formulations for tangent-plane (G^1) continuity between adjacent patches. We explore the constraints and the number of degrees of freedom in the placement of the interior control points of the patches.

7.1. Expressing Continuity

First-degree geometric continuity (G^1-continuity) requires that both patches Φ and Ψ (Fig. 11), meeting along the boundary Γ have the same tangent plane at every point of the common boundary curve ($v \in [0,1]$). Thus for any given parametrization of the patches, the two cross-boundary derivatives $D\Phi$ and $D\Psi$ and the common derivative along the boundary $D\Gamma$ — the tangent of the boundary curve — need to lie in one plane, but the two cross-boundary derivatives need not be collinear. One elegant way to express G^1-continuity is to set the determinant of these three derivatives to zero:

$$\det (D\Phi(v), D\Psi(v), D\Gamma(v)) = 0, \quad v \in [0,1]. \tag{1}$$

This requires the selection of suitable cross-boundary derivatives. Patches Φ and Ψ can be either bicubic quadrilateral or quartic triangular patches. For the quadrilateral case the choice is obvious: one uses the partial derivative with respect to the parameter that is kept constant on the boundary curve:

$$D\Phi(v) \equiv \Phi_{\bar{u}}(\bar{u}=0,v). \tag{2}$$

For a barycentric triangle the situation is more complicated. We follow Farin[16] in

defining a *radial* derivative:

$$D\Psi(v) \equiv (1-v)\,(\Psi_s - \Psi_v) + v\,(\Psi_w - \Psi_v). \tag{3}$$

A graphical interpretation is that the cross boundary control vectors are simply given by connecting the Bézier points of the cubic border with the corresponding interior Gregory points of the quartic triangle (see also Fig 9).

In another formulation we can express the coplanarity of the three derivatives as a vanishing vector function:

$$\alpha(v)\,D\Phi(v) + \mu(v)\,D\Psi(v) + \lambda(v)\,D\Gamma(v) = 0. \tag{4}$$

Since $D\Phi(v)$ and $D\Psi(v)$ are of degree 3, while $D\Gamma(v)$ is of degree 2,

$$\mathbf{deg}\,(\alpha(v)) = \mathbf{deg}\,(\mu(v)) = \mathbf{deg}\,(\lambda(v)) - 1. \tag{5}$$

This is now a system of vector equations which decomposes into three identical scalar systems, one for each coordinate. This is simpler than the determinant equation above which mixes the components along different axes. If we further restrict the degree of the polynomial functions $\alpha(v)$ and $\mu(v)$ to degree two, the equation system becomes quite manageable.

Either of the above two formulations ties the control points on one side of the curve to the points on the other side. Patches on either side can thus not be chosen independently. In the next section we explore how many degrees of freedom we have in this system from which we want to obtain the twelve coordinate values for the two pairs of control points on either side of the border.

7.2. Degrees of Freedom

At this point, it is useful to analyze the situation on the border curve a little more closely. We would like to know how many degrees of freedom there are intrinsically before we start using them for the sake of obtaining simpler solutions. Ideally we would like to keep them all and express them in terms of natural shape parameters that could be set globally or locally and through which the user can control the shape of the surface in an intuitively obvious manner. This formulation is the subject of ongoing research.[21]

The above determinant equation (eqn 1) is a polynomial of degree 8 in v, which yields 9 equations for the coefficients of the various degrees of v. Two of these equations represent the endpoint conditions that all curves at a vertex must end in a common tangent plane. These will be automatically fulfilled if the two middle Bézier points of the boundary curves have been properly selected. Thus there remain 7 equations for the 4 vectors $\mathbf{R}_1$, $\mathbf{R}_2$, $\mathbf{T}_1$, $\mathbf{T}_2$ (12 unknown scalars). They proved to be so complicated, that we could not cast them into a usable form[10] even with the use of VAXIMA.[22] However, the system of equations shows that there are only five degrees of freedom in placing the 4 control vertices. The same result follows from the vector equation (eqn 4).

This means that we cannot select one patch at will and then fit the other one to it with G^1-continuity. The exact nature of the restriction is not currently understood. However certain construction methods will lead to a solvable system of equations. In particular, certain implicit or explicit assumptions about the behavior of the seam along the shared border will lead to equation systems that are not hard to solve.

7.3. Specifying the Behavior of the Seam

We can use some of the available degrees of freedom to nail down the behavior of the surface at the border and then solve for the position of the interior control points in accordance with the chosen behavior.

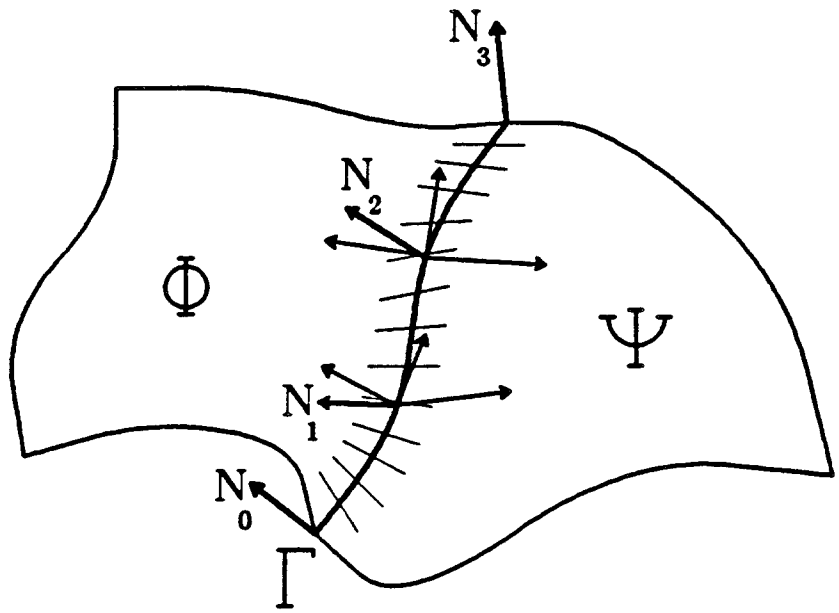


Figure 12: *Normals and tangents on the seam between two patches*

In one approach, the behavior of the *surface normal* $\mathbf{N}(v)$ is specified along the border, and the patches on either side then have to be constructed so that their tangent planes are perpendicular to the given normal direction at every point. This constraint is best expressed as a vanishing dot-product between the normal and the cross-boundary derivative at any point along the border Γ:

$$(\mathbf{N}(v), D\Phi(v)) = 0, \quad v \in [0,1]. \tag{6}$$

Of course, if we prescribe the normal along the border, it must be done in such a manner that at every point it is perpendicular to the curve's tangent:

$$(\mathbf{N}(v), D\Gamma(v)) = 0, \quad v \in [0,1]. \tag{7}$$

This looks like the most promising approach to extract some explicit shape parameters. We are currently studying the various possibilities of defining a default behavior for the normal vector.

In another approach, a generic *cross-boundary tangent* $\mathbf{T}(v)$ is defined, that together with the border tangent $D\Gamma(v)$ defines the tangent plane at every point. The two patches then have to match this tangent behavior, or — equivalently — their cross-boundary derivatives must be linear combinations of the border tangent and the defined cross-boundary tangent:

$$\alpha(v)\, D\Phi(v) = \lambda(v)\, D\Gamma(v) + \mathbf{T}(v), \tag{8}$$

$$\mu(v)\, D\Psi(v) = \rho(v)\, D\Gamma(v) + \mathbf{T}(v). \tag{9}$$

In order to obtain a generic tangent behavior, Chiyokura starts by setting the cross-boundary tangents at the two endpoints to $\mathbf{T}_0 - \mathbf{R}_0$ and $\mathbf{T}_3 - \mathbf{R}_3$, respectively (see Fig. 11). Although the cross-boundary tangent can be cubic in general, Chiyokura linearly interpolates it between the two values at the ends of the boundary curve. With this approach the degree of the polynomial equations from the vanishing vector function (eqn 4) is reduced drastically, and the computation for the two interior control points of a patch becomes quite simple. This procedure is performed separately for the two patches on either side of the border curve.[18,19]

Our current implementation of UNICUBIX follows this last approach. Other approaches are being studied and their trade-offs are evaluated. We are also investigating how to extract the most useful geometric shape parameters from this approach.

8. CONSTRUCTION RULES IN UNICUBIX

UNICUBIX has a mode in which it automatically converts a polyhedral object into a smooth surface through the given vertices. The process follows the step-by-step procedures outlined in Section 5.

8.1. Choice of Vertex Normals

The first task is to choose suitable vertex normals (Fig. 13). Obviously these normals should reflect any inherent symmetries of the polyhedral shape. Furthermore, they should be tessellation independent. Since our program can only handle triangular and quadrilateral patches, polygonal faces with more sides must be subdivided first. This is done by an automatic tessellation program that returns triangles or convex quadrilaterals. Since this tessellation is performed with a sweep plane algorithm, the result, and, in particular, the number of cut-lines coming together at one vertex, is dependent on the orientation of the object in space. However, we would like the shape of the final smooth surface to be orientation independent.

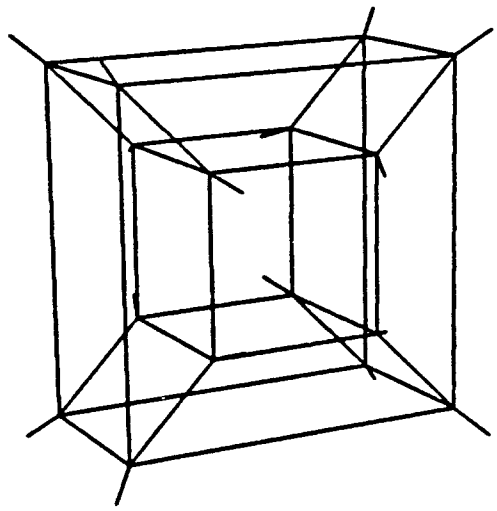


Figure 13: *Polyhedral frame of square torus with vertex normals*

A procedure that achieves this goal is to average all face normals of the polygons that share a vertex, weighing the individual contributions with the angle of the polygon at that vertex. Any extra cut in a face will simply produce two smaller contributions summing to the same result.

8.2. Defining the Cubic Borders

The next step is to choose two inner Bézier points for each curve segment replacing the original straight edges. These points must lie in the tangent plane of the topologically nearest vertex; within that plane each vertex has two degree of freedom.

In our first implementation we have taken a very simple approach. Each edge leaving a particular vertex is projected onto its tangent plane. The corresponding Bézier point is then chosen at a distance from the vertex equal to one third the length of the original edge. This gives good results for convex bodies but is not quite satisfactory for the inner parts of a torus or similar concave regions. More elaborate rules to deal with this case are under development.

Once the Bézier points have been determined, the cubic border curves can be drawn; they are displayed in Figure 14 together with the vertex normals.

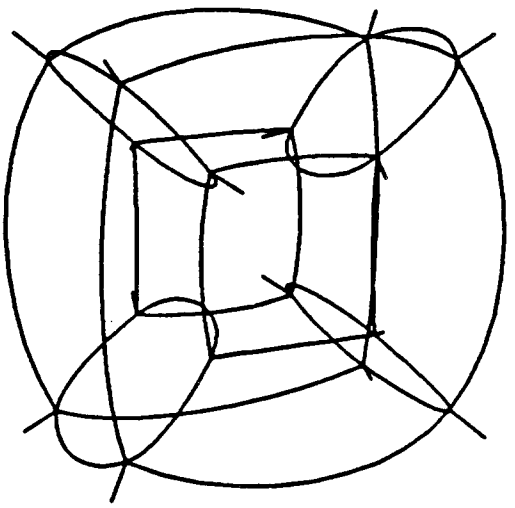


Figure 14: *Cubic boundary curves generated from the frame of Figure 13*

8.3. Specifying the Seams

In our first implementation we have followed Chiyokura's approach,[19] specifying in an implicit manner the behavior of the surface patches along their seams (see Section 7.3). Cross-boundary derivatives are derived from the Bézier points associated with the end-vertices of the cubic border curves and linearly interpolated along these curves. Alternatives involving the surface normals are under investigation.

Of course, not all borders need to be smoothly interpolated by the adjacent patches. Objects in UNICUBIX can display a mixture of rounded and of sharp edges. For the case of sharp edges, the procedures above need to be modified suitably. In particular, the determination of the vertex normals changes. At any vertex there can now be more than

one distinct normal vector, associated with different faces coming together with different tangent planes.

The procedure to determine the control points for the cubic edges must then pick the proper normal carefully. But afterwards, the determination of the interior patch control points proceeds along the same scheme unless the patch happens to be a completely flat face. In this case, it will be treated as a many-sided planar polygon.

8.4. Filling in the Patches

The specification of the seams implicitly defines the tangent behavior of the patches at all their boundaries. By using Chiyokura's formulas,[19] we can generate the two interior Gregory points along every border curve (Fig. 15). This then defines the patch fully.

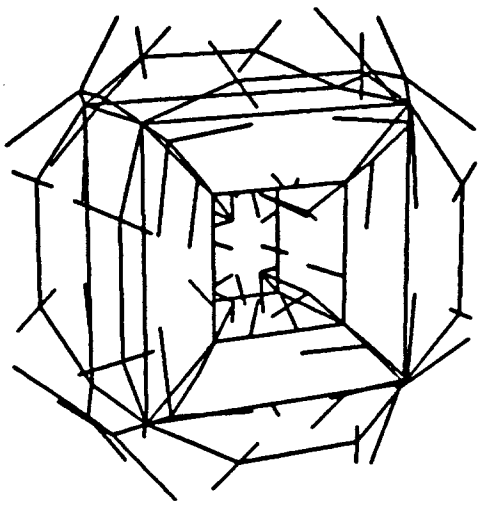


Figure 15: *Cubic border frame with all control points*

We can now evaluate the patch at a regular grid of points in the rectilinear or barycentric triangular parameter space, and connect these points on the surface with a mesh of straight edges and triangular or quadrilateral faces. The result is a net-representation of the surface patches as shown in Figure 16.

9. RENDERING ISSUES

9.1. Results

The net representation in the previous section is a regular UNIGRAFIX description [1] of a polyhedral object — provided that the facets are planar. Triangles, of course, are no problem, but with quadrilaterals we cannot be sure. Often the inherent symmetry of the object assures that the quadrilaterals are also planar; these are the preferred cases for quadrilateral patches. In other cases one may use a net fine enough so that the deviation from planarity causes no problem in the renderer (Fig 17). One of our renderers, ugdisp,[23] has a special option to accommodate somewhat warped polygons. This same renderer also has the capability to do smooth Gouraud shading of polyhedral objects.

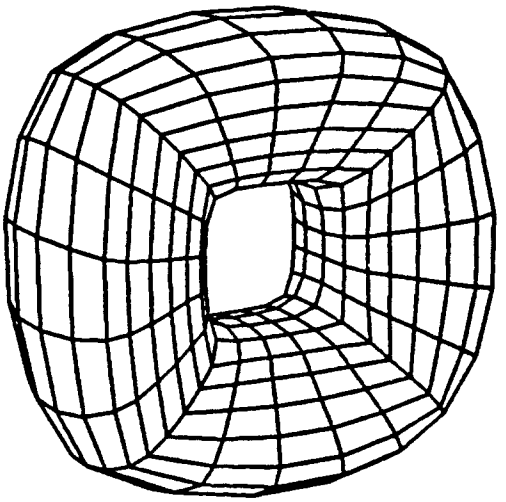


Figure 16: *Net-representation generated from Fig. 15*

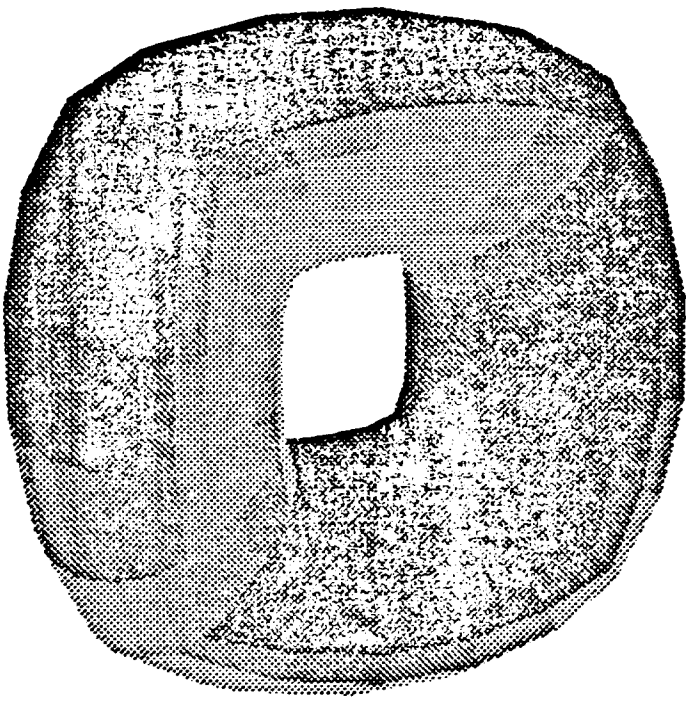


Figure 17: *Shaded rendering of object shown in Figure 16.*

9.2. Conversion to Bézier Patches

Bézier patches are more attractive than Gregory patches for a variety of reasons. Their polynomial form leads to computational efficiency in rendering. Furthermore, there is a wealth of published and implemented algorithms for subdivision, rendering, and mutual intersection of such surfaces, which are not yet available for the case of Gregory patches. We have thus tried to fill the meshes between our cubic boundaries with Bézier

patches. However, this cannot be achieved with single Bézier patches of any degree because of the twist constraints at the corners. The boundary curves would have to be subjected to extra constraints to make possible the use of Bézier patches in each mesh.

By using Gregory patches we have obtained complete locality of control. The cubic boundaries can be specified with few constraints, and the patches can then be fitted into the meshes individually. Ideally we would like to retain this property. To do so with Bézier patches, we must partition each mesh and introduce at least one new boundary at each corner of the mesh. The exact shape of these internal boundaries must be left under the control of the system so that the twist constraints of the Bézier patches can be fulfilled automatically.

Using this general approach, we have found a way to fill triangular meshes with three Bézier triangles rather than with one Gregory patch (Fig. 18a). Similarly we can replace a quadrilateral Gregory patch by either five quadrilateral Bézier patches (Fig. 18b), or by a mixture of triangles and quadrilaterals.[11]

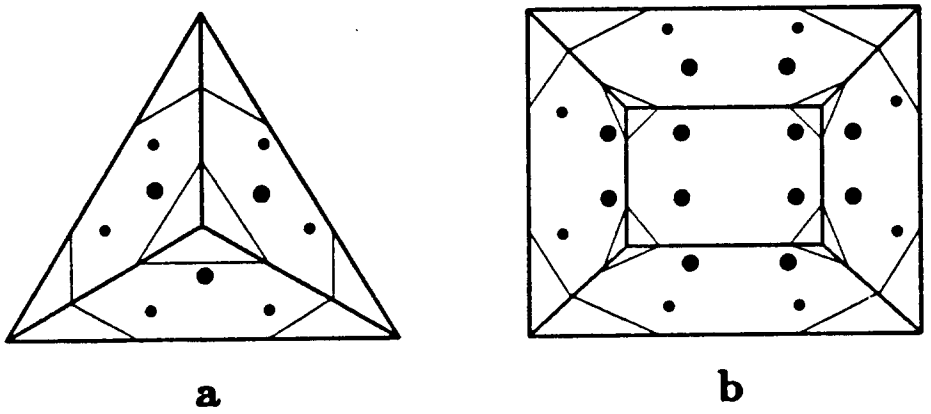


Figure 18: *Replacing Gregory patches with multiple Bézier patches*

In all cases, along every outer boundary of the original mesh two interior patch control points are determined with the same techniques as used to find the two Gregory points (Section 8.4); these are the small dots in Figure 18. The remaining interior control points (large dots) and all the control points on the internal boundaries are then determined so that one obtains a system of three to five Bézier patches that join with G^1-continuity and meet the specification of the external seams.[11]

The replacement of a quadrilateral with four Bézier triangles so far has resisted our attempts to find a solution.

10. CONCLUSION

We have implemented a prototype of a practical modeling system for the non-mathematical user. UNICUBIX has a very terse description of the objects since it needs to store only the information for the shape of the cubic boundary curves. All information for the shapes of the patches is implicitly given by the shape of the boundary curves and by some global (or possibly local) shape parameters.

This implicit information is extracted from the given UNICUBIX description in a sequence of procedural steps. In turn, the vertex normals are constructed, then the inner two Bézier points for the cubic borders are determined if they are not explicitly given, from that the interior patch control points can be computed, and finally a net representing the surface patch can be constructed.

This same procedural approach could be extended to curves and surface patches of higher order, making it possible, for instance, to obtain interpolating splines with curvature continuity.

11. ACKNOWLEDGEMENTS

This development has been made possible by the dedicated effort of Lucia Longhi and Leon Shirman, who worked out the mathematical foundation for the described approach and who created the UNICUBIX programs. Thanks also go to Brian Barsky and Mark Segal who read this manuscript on very short notice. This effort is supported by Tektronix, Inc. and by the Semiconductor Research Corporation.

REFERENCES

1. C.H. Séquin, "Berkeley UNIGRAFIX, A Modular Rendering and Modeling System," *Proc. of the 2nd USENIX Computer Graphics Workshop*, Monterey CA, pp. 38-53 (Dec. 1985).
2. C.H. Séquin and P.S. Strauss, "UNIGRAFIX," *Proc. 20th Design Automation Conf.*, Miami Beach, FL, pp. 374-381 (June 1983).
3. I. D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacture,* Ellis Horwood Ltd (1979).
4. M. E. Mortenson, *Geometric Modeling,* Wiley, New York (1985).
5. R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An Introduction to the Use of Splines in Computer Graphics,* Morgan-Kaufmann Publishers, Inc., Los Altos, California. To appear.
6. V. Pratt, "Techniques for Conic Splines," *SIGGRAPH'85 Conf. Proceedings*, pp. 151-159 (July 1985).
7. J. Hobby, "Smooth Easy to Compute Interpolating Splines," *Discrete and Computational Geometry* **1**, pp. 123-140 (1986).
8. B. A. Barsky and T. D. DeRose, "Geometric Continuity of Parametric Curves," Tech. Report, U.C. Berkeley (Oct. 1984).
9. A. Fournier and B. A. Barsky, "Geometric Continuity with Interpolating Bézier Curves (Extended Summary)," pp. 337-341 in *Proceedings of Graphics Interface '85*, Montreal (27-31 May 1985). Revised version published in *Computer-Generated Images – The State of the Art*, edited by Nadia Magnenat-Thalmann and Daniel Thalmann, Springer-Verlag, 1985, pp. 153-158.
10. L. Longhi, "Interpolating Patches Between Cubic Boundaries," Master's Report, U.C. Berkeley (Dec. 1985).
11. L. Shirman, "Symmetric Interpolation of Triangular and Quadrilateral Patches

between Cubic Boundaries," Tech. Report, U.C. Berkeley (Dec. 1986).

12. E. E. Catmull and R. J. Rom, "A Class of Local Interpolating Splines," pp. 317-326 in *Computer Aided Geometric Design*, ed. R. E. Barnhill and R. F. Riesenfeld, Academic Press, New York (1974).
13. T. D. DeRose and B. A. Barsky, "Geometric Continuity and Shape Parameters for Catmull-Rom Splines (Extended Abstract)," pp. 57-64 in *Proceedings of Graphics Interface '84*, Ottawa (June 1984).
14. B. A. Barsky and J. C. Beatty, "Local Control of Bias and Tension in Beta-splines," *ACM Transactions on Graphics* **2**(2), pp. 109-134 (April, 1983). Also published in *SIGGRAPH '83 Conference Proceedings* (Vol. 17, No. 3), ACM, Detroit, 25-29 July, 1983, pp. 193-218.
15. C. H. Séquin, "Rule Based Spline Interpolation," work in progress.
16. G. Farin, "A Construction for Visual C1 Continuity of Polynomial Surface Patches," *Computer Graphics and Image Processing* **20**, pp. 272-282 (1982).
17. G. Farin, "Triangular Bézier-Bernstein Patches," *Computer Aided Geometric Design* **3**(2), pp. 83-127 (1986).
18. H. Chiyokura and F. Kimura, "Design of Solids with Free-form Surfaces," *Computer Graphics (Siggraph'83 Conf. Proc.)* **17**(3), pp. 289-298 (1983).
19. H. Chiyokura, "Localized Surface Interpolation Method for Irregular Meshes," *Proc. Computer Graphics Conf.*, Tokyo (1986).
20. J. A. Gregory, "Smooth Interpolation Without Twist Constraints," pp. 71-87 in *Computer Aided Geometric Design*, ed. R. E. Barnhill and R. F. Riesenfeld, Academic Press, New York (1974).
21. C. H. Séquin, "Shape parameters for G1 continuous patches," work in progress.
22. R. J. Fateman, "Addendum to the MACSYMA Reference Manual for the VAX," Tech. Report, CS Div., U.C. Berkeley (1982).
23. N. Gal, "Hidden Feature Removal and Display of Intersecting Objects in UNIGRAFIX," Master's Report, U.C. Berkeley (Jan. 1986).