Surface Interpolation with Tensor Product Patches

Luciano Pereira dos Reis
Departamento de Exploração
Petrobras-Petróleo Brasileiro S.A.
g073@c53000.petrobras.anrj.br

Alyn Rockwood
Computer Science Dept.
Arizona State University
rockwood@asuvax.eas.asu.edu

Abstract. A method is presented for the construction of a piecewise biquartic polynomial tensor product surface interpolating arbitrarily located data, given as a 3D mesh of points or cubic curves. Macropatches composed of rectangular patches are used to fit each original face. Since only standard tensor product patches are employed, the method can be directly integrated with most existing systems and allows the use of widely available geometric processing and rendering libraries and algorithms.

1. Introduction

Parametric tensor-product surfaces have become a de facto industrial standard for modeling free-form surfaces. This type of surface is employed by the great majority of systems now in use, and supported by current graphics standards, libraries and workstations.

Despite this popularity, however, tensor-product B-spline surfaces cannot model surfaces of arbitrary topology. Methods that allow the construction of smooth (tangent plane continuous) surfaces interpolating arbitrarily located data—a problem that arises in numerous application areas such as geometric modeling, scientific visualization, medical imaging and geological modeling—generally employ triangular patches, but their use in commercial systems is troublesome, due to the burden of dealing with different patch representations [Nie93].

Here we propose such a construction scheme, which unlike other methods, makes use only of polynomial tensor product patches. We assume that the given data consists either of vertices (and possibly normals at these vertices) of a polyhedron of arbitrary topology, or of a G^1 curve network composed of cubic polynomials.

Our goal is to employ strictly polynomial patches, without singularities and of lowest possible degree. The advantages of using this kind of standard representation are:

- compatibility with existing systems, facilitating integration and data exchange;
- consequences on development cost, extensibility and maintainability, since duplicate procedures to deal with different patch types are not necessary;
- wider availability of geometric processing algorithms;
- immediate availability of advanced rendering options with efficient implementations, as this is the representation adopted by the graphics libraries and standards now in use (GL, PHIGS, etc.);

The face splitting approach adopted, depicted in Figure 1, has been used in several *n*-sided patch schemes ([Chi83, Var87, Sar87, Hah89, Zha92]). However, these schemes generally employ non-polynomial representations with singularities, or high degree patches and restricted mesh configurations.

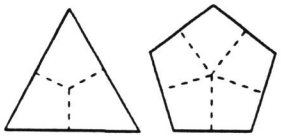

Figure 1: Face splitting in four-sided patches.

The first and basic problem to achieve G^1 continuity using this representation is that the mixed partial derivatives ("twists") of the patches around a vertex have to be compatible, as will be discussed later. It is well-known that at "even" vertices (adjacent to an even number of curves) this can only be accomplished using single non-rational polynomial patches if the boundary curves are restricted so that a compatibility condition is met [Wij84, Sar87, Wat88].

If this is not the case, however, we find that it can still be advantageous to look for the "best possible" solution (or approximate G^1 continuity), which can be adequate for many applications.

Another problem is to specify a scheme for splitting the original faces (macropatches) in four-sided subpatches, respecting the cross boundary derivative field at the original edges, maintaining the degree of the patches as low as possible, and preserving the "quality" of the surface (that is, avoiding unwanted undulations).

We use the formulation of the G^1 continuity condition in [Var93a], which allows a geometrically intuitive analysis of the problem. Using this formulation, we propose a construction scheme that leads to biquartic patches.

Improving the quality of the obtained surface is still a problem, since the construction scheme creates

"extra" degrees of freedom whose adequate setting is not trivial. This problem is common to similar methods, as discussed in [Man92].

2. Previous Work

Several methods have been developed for the generation of smooth surfaces composed of triangular and *n*-sided patches

The basic steps followed by local interpolation methods are:

1. Creation of a preliminary curve network, at least G^1, connecting the data points.
2. Generation of cross-boundary derivative information along the boundary curves. Usual approaches are to use Farin's G^1 condition [Far82]; Chiyrokura and Kimura's construction [Chi83]; or Nielson's construction (for normal vectors along the boundary) [Nie87].
3. Generation of the final patches, filling in the holes between boundary curves.

2.1. Triangular Methods

Triangular patches are general enough to model any type of object, and triangulations are very common in applications dealing with empirical data. This has led to the development of several methods for parametric interpolation using triangular interpolants. A survey and comparison of some representative methods can be found in [Man92] and also [Lou92]. The methods are classified into two groups, according to their solution to the twist compatibility problem: split-domain and convex combination methods.

2.2. N-sided Methods

Some problems lend themselves more naturally to the use of *n*-sided patches. A review and classification of several methods can be found in [Var87] (see also [Gre86]).

2.3. "Compatible" Methods

Also closely related to our problem are the methods of Sarraga and Peters. Both construct single polynomial patches per face of a curve network, and therefore cannot circumvent the compatibility problem using splitting or convex combinations.

Sarraga's method [Sar87], based on Bezier's work, also constructs surfaces composed only of rectangular Bezier patches. However, the network configuration is restricted: only originally four sided faces are handled, and only three, four (pairwise tangent) or five curves are allowed to meet at a vertex. Moreover, an asymmetric formulation of the compatibility condition is used, leading to patches of unequal and high degree (6 x 6 and 3 x 6).

In [Pet91] Peters proves that a sufficient condition for the curves incident to an "even-vertex" to be compatible is that they match second-order data there. He also gives a construction leading to quartic curves respecting this constraint.

Peters' scheme assumes that the network is composed by three and four-sided faces, generating triangular and rectangular patches (one per face). The patches are quartic and biquartic in most cases, but triangular faces are quintic (or need to be split) if certain conditions occur.

2.4. Approximate Geometric Continuity

In some situations, analytical G^1 continuity might not be necessary. This may be the case, for instance, in some scientific visualization applications, when surface quality is not a primary concern; and in other modeling situations, if eventual tangent plane discontinuities are imperceptible in the rendered objects, or below machine precision in CAM applications.

Moreover, the relaxation of the continuity requirements may even be desirable. In [DeR92], the authors show that avoiding the "vertex enclosure" problem can lead to the use of fewer surface patches of lower degree.

3. Continuity Between Patches

Assuming that the curves are already specified, constructing the patches amounts to finding its interior Bezier control points. A G^1 join requires constructing appropriate cross-boundary derivatives for each boundary curve, i.e., determining the first interior row of Bezier control points. If the patches are not cubic, and more interior control points exist, they can be chosen freely, since they will not affect the cross-boundary derivatives, but only the interior shape of the patches.

First we look at the condition to join two patches with G^1 continuity. Given cubic boundary curves, there is no problem in creating appropriate cubic cross-boundary derivatives, and therefore cubic patches. Next we consider the problem of connecting more patches around a vertex. In this case cubic cross-boundaries will not be enough to satisfy the constraints.

3.1. G^1 Continuity Between Two Patches

Consider the two patches, $\mathbf{R}^{(1)}(u,v)$ and $\mathbf{R}^{(2)}(u,v)$ that share a common boundary as in Figure 2. The endpoints of the boundary, $\mathbf{r}(u)$, correspond to u=0

and u=1. We call the derivatives at u on the boundary $\mathbf{r}_u(u)$, $\mathbf{r}_v^{(1)}(u)$ and $\mathbf{r}_v^{(2)}(u)$.

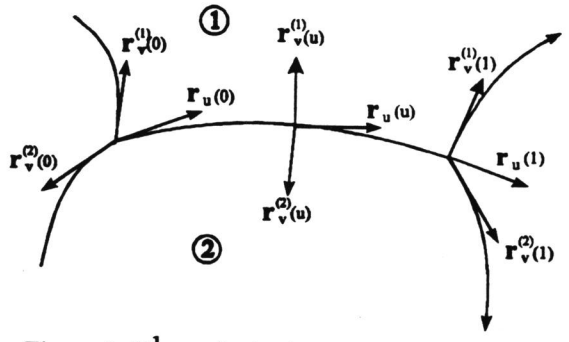

Figure 2. G^1 continuity between two patches.

Following the approach in [Var93a, Var93b], the "common direction blend" method, the cross-boundary derivatives for each patch are written as

$$\mathbf{r}_v^{(i)}(u) = \mathbf{D}(u)A^{(i)}(u) + \mathbf{r}_u(u)P^{(i)}(u), \quad i=1,2 \qquad (1)$$

A common direction vector, $\mathbf{D}(u)$, is introduced, and independent pairs of scalar "blending functions" $A^{(1)}(u)$, $P^{(1)}(u)$, and $A^{(2)}(u)$, $P^{(2)}(u)$, used at each side of the boundary curve.

Proposition: *If the cross-boundary derivatives can be written as in Equation 1, G^1 continuity is achieved.*

Proof: Surface normals are obtained by the cross products

$$\mathbf{r}_u(u) \times \mathbf{r}_v^{(i)}(u) = [\mathbf{r}_u(u) \times \mathbf{D}(u)]A^{(i)}(u), \quad i=1,2 \qquad (2)$$

which differ only by scalar factors $A^{(i)}(u)$.

□

For implementation purposes we pick $\mathbf{D}(u)$ orthogonal to $\mathbf{r}_u(u)$. If $\mathbf{r}_u(u)$ and $\mathbf{D}(u)$ were unit vectors, $A^{(i)}(u)$ and $P^{(i)}(u)$ would be the projections of $\mathbf{r}_v^{(i)}(u)$ on these directions.

The key issue for the construction of the cross-boundary derivatives is the determination of $A^{(i)}(u)$, $P^{(i)}(u)$ and $\mathbf{D}(u)$, chosen to satisfy the fixed end conditions from Equation 1 for u=0 and u=1.

3.2. Twists

Now let $\mathbf{r}_u(u)$, $\mathbf{D}(u)$, $A^{(i)}(u)$ and $P^{(i)}(u)$ be given and assume the twist vectors at the endpoints are to be found. From Equation 1:

$$\mathbf{r}_{uv}^{(i)}(u) = \mathbf{D}(u)A_u^{(i)}(u) + \mathbf{D}_u(u)A^{(i)}(u) + \\ + \mathbf{r}_u(u)P_u^{(i)}(u) + \mathbf{r}_{uu}(u)P^{(i)}(u), \quad i=1,2 \qquad (3)$$

At u=0, eliminating $\mathbf{D}_u(u)$ from Equation 3 and simplifying yields

$$\frac{\mathbf{r}_{uv}^{(1)}(0)}{A^{(1)}(0)} - \frac{\mathbf{r}_{uv}^{(2)}(0)}{A^{(2)}(0)} = \mathbf{D}(0)\left[\frac{A_u^{(1)}(0)}{A^{(1)}(0)} - \frac{A_u^{(2)}(0)}{A^{(2)}(0)}\right] \\ + \mathbf{r}_u(0)\left[\frac{P_u^{(1)}(0)}{A^{(1)}(0)} - \frac{P_u^{(2)}(0)}{A^{(2)}(0)}\right] \\ + \mathbf{r}_{uu}(0)\left[\frac{P^{(1)}(0)}{A^{(1)}(0)} - \frac{P^{(2)}(0)}{A^{(2)}(0)}\right] \qquad (4)$$

from which we see, multiplying both sides by $-A^{(2)}(0)$, that

$$\mathbf{r}_{uv}^{(2)}(0) + c\mathbf{r}_{uv}^{(1)}(0) = \mathbf{p} \qquad (5)$$

for constants c and **p**.

For two patches sharing one boundary curve, with $\mathbf{r}_u(u)$, $\mathbf{D}(u)$, $A^{(i)}(u)$ and $P^{(i)}(u)$ previously defined, this equation simply states a relation between the twist vectors of the patches.

However, when more patches are arranged around a vertex, as in Figure 3, a system of equations relating the twists of each patch is formed, and $\mathbf{D}(u)$ can no longer be independently set for each boundary curve.

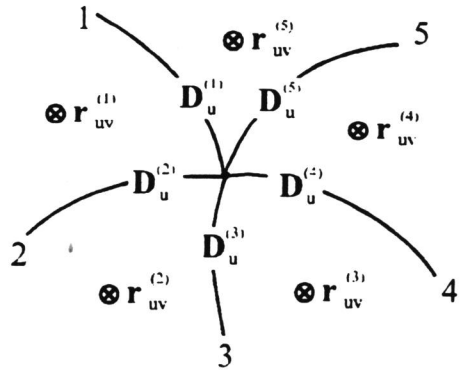

Figure 3. G^1 continuity between patches around a vertex.

3.3. G^1 Continuity Between More Patches Around a Vertex

Consider Figure 3 above. In order to satisfy G^1 continuity at a common vertex of a set of n patches – such as any original vertex of the curve network with n incoming curves, or at the middle point of a face – the twists obtained for each patch from each of its two boundary curves must be the same. In Bezier form, the "tangent ribbons" of the two curves have to be compatible. This is the "twist compatibility problem" mentioned before.

A system of equations for the n twist vectors is formed. Applying Equation 5 at each boundary curve:

$$\begin{aligned}
&\mathbf{r}_{uv}^{(2)} + c_1\mathbf{r}_{uv}^{(1)} = \mathbf{p}_1 \\
&\mathbf{r}_{uv}^{(3)} + c_2\mathbf{r}_{uv}^{(2)} = \mathbf{p}_2 \\
&\quad\vdots \\
&\mathbf{r}_{uv}^{(1)} + c_n\mathbf{r}_{uv}^{(n)} = \mathbf{p}_n
\end{aligned} \tag{6}$$

Notice that, from Equation 4, the expressions for the vectors $\mathbf{p}$ at each boundary curve do not involve $\mathbf{D}_u(0)$. However, once the twists are determined, these values can be obtained from Equation 3, and will therefore be fixed. Consequently, $\mathbf{D}(u)$ must be at least cubic to interpolate values and derivatives at the endpoints, and the cross boundary-derivatives must be at least quartic, according to Equation 1.

Solving the system for the twists as in [Var93a] results in

$$\begin{aligned}
&\mathbf{r}_{uv}^{(2)} = \mathbf{p}_1 - c_1\mathbf{r}_{uv}^{(1)} \\
&\mathbf{r}_{uv}^{(3)} = \mathbf{p}_2 - c_2\mathbf{r}_{uv}^{(2)} = \mathbf{p}_2 - c_2\mathbf{p}_1 + c_2c_1\mathbf{r}_{uv}^{(1)} \\
&\quad\vdots \\
&\mathbf{r}_{uv}^{(1)} = \mathbf{p}_n - c_n\mathbf{p}_{n-1} - c_nc_{n-1}\mathbf{p}_{n-2}\ldots(-1)^n c_n\ldots c_1\mathbf{r}_{uv}^{(1)} \\
&\quad = \mathbf{P} + C\mathbf{r}_{uv}^{(1)}
\end{aligned} \tag{7}$$

and leads to the well-known conclusion:

Theorem: *For odd n,C=-1, and a solution to twist compatibility problem (Equation 7) always exists. For even n, C=1, and a solution exists only if* $\mathbf{P}=0$. *In fact, in this case infinitely many exist.*

A geometrical proof can be found in [Rei93], and equivalent proofs by calculating the determinants of systems of equations similar to Equation 6 in matrix form can be found in [Wij84, Sar87, Wat88].

Notice that from equation 4, the condition $\mathbf{P}=0$ is automatically achieved in the case of four pairwise tangent curves.

Back to the interpolation problem, the conclusion is that in a mesh of general topology containing "odd" and "even" vertices, a "compatible" curve network would have $\mathbf{P}=0$ at every even vertex.

Therefore, if non-compatible curves are given or created, it is not possible to fit a G^1 piecewise polynomial surface to them with the configuration desired (that is, using only one patch between each original pair of curves coming into a vertex).

In this situation the strategy adopted herein is to obtain the "best possible" results, in a least squares sense: we solve the system of equations using Singular Value Decomposition [Pre88]. This gives the twists which minimize the norm of the residual of the solution of the system. The resulting surface is of course not analytically G^1, but the closest possible for the given curves. The severity of the discontinuities will depend on the curve network configuration. For the torus test case, shown in Section 5, gaps in the isophotes, indicative of G^1 discontinuity [Hag92], were minor and appeared only in a few cases, in very few spots. In the shaded surfaces discontinuities were imperceptible.

3.4. Face Subdivision

From Equation 4, linear blending functions lead to "coupling" between vertices, because the value of the derivatives of the blending functions at a vertex will depend on the function values at the opposite vertex.

This has an important consequence to the scheme presented here: when defining the cross-boundary derivatives for the subdividing curves of a face, $A(u)$ and $P(u)$ cannot be linear: the twist at the junction with the original curve is already fixed according to the cross-boundary derivatives obtained before.

One way to avoid this coupling is to make $A^{(i)}(u)$ and $P^{(i)}(u)$ quadratic, with $A_u^{(i)}(1)=P_u^{(i)}(1)=0$. From Equation 1, making $P^{(i)}(u)$ quadratic will not raise the degree of the cross-boundary derivative, since, for cubic curves, $\mathbf{r}_u(u)$ is quadratic.

However, $A^{(i)}(u)$ quadratic would make the cross-boundary derivative quintic. To avoid this, we will instead make it constant, by adopting a special constraint for the derivatives of the subdividing curves at the midpoint (see [Rei93] for details). In the case of triangular faces this can be easily achieved by positioning the middle point such that it is on the barycenter of the triangle formed by the curve-derivative vectors.

4. Construction Method

The surface construction method consists of the following steps:

- If only points and normals are given, creation of the curves.
- Solution (or approximate solution) of the twist compatibility equations at the vertices.
- Generation of compatible (or "as compatible as possible") cross-boundary derivatives along the curves, using the twists obtained.
- Subdivision of the curves and cross-boundary derivatives at u=0.5 .
- Generation of a point and normal vector at the middle of each face.
- Generation of the face-splitting curves.
- Creation of cross-boundary derivatives along the face-splitting curves.
- Generation of interior control points for the patches .

One very important feature in an efficient implementation of the scheme proposed is the use of a boundary representation (B-rep) data structure, for easy access to the adjacency information and also as an

auxiliary data-structure where the obtained information is stored with the associated topological entities.

4.1. Curve Construction

If the curves are not given, only points and normals as in the examples presented here, then they are independently constructed for each edge, using the two points and associated normals: $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{N}_1$, $\mathbf{N}_2$.

The construction method adopted is the one presented in [Nie87]: at each vertex the vector-product of the normal with the vector $(\mathbf{V}_2-\mathbf{V}_1)$ is obtained. Then the vector-product of the result with the normal is taken and normalized, giving the unit-vector in the tangent direction. The length of the tangent vector is heuristically set to be equal to the distance between $\mathbf{V}_2$ and $\mathbf{V}_1$. A cubic curve is constructed interpolating the points and tangent vectors, and stored in the data structure as an edge attribute.

One important observation: in discrete interpolation methods, this step has a fundamental influence on the quality of the final interpolant. Better schemes for setting adequate values for the tangent magnitudes would improve the quality of any surface fitting scheme. It is easy to see that settings based on heuristics and strictly local information can be arbitrarily good or bad, depending on the situation.

4.2. Cross-Boundary Derivatives for the Original Curves

The cross-boundary derivatives at both sides of each edge are obtained by direct application of Equation 1 after $\mathbf{D}(u)$ is determined. The scalar blending functions $A^{(v)}(u)$ and $P^{(v)}(u)$ are linear functions interpolating the values at the vertices $A^{(v)}(0)$, $A^{(v)}(1)$, and $P^{(v)}(0)$, $P^{(v)}(1)$, which can be calculated by decomposing $\mathbf{r}_v^{(v)}(0)$ and $\mathbf{r}_v^{(v)}(1)$, the derivatives of the winged-edge curves at each vertex, in the directions $\mathbf{D}(0)$, $\mathbf{r}_u(0)$, and $\mathbf{D}(1)$, $\mathbf{r}_u(1)$, respectively.

The derivative $\mathbf{r}_u(u)$ can be directly obtained from the edge curve.

The "common direction blend" $\mathbf{D}(u)$ is a cubic vector-valued function interpolating $\mathbf{D}(0)$, $\mathbf{D}(1)$, $\mathbf{D}_u(0)$ and $\mathbf{D}_u(1)$. We picked the first two values to be unit vectors orthogonal to $\mathbf{r}_u(0)$ and $\mathbf{r}_u(1)$, respectively, on the tangent plane. $\mathbf{D}_u(0)$ and $\mathbf{D}_u(1)$ are obtained from Equation 3 after the twists for the patches around a vertex are computed.

The twists are calculated: all vertices are traversed and the system of equations in Equation 6 solved at each one. The assembly of the system requires that all values involved are collected or calculated, at the vertex itself (u=0) and at the opposite vertex (u=1), for each incident edge.

From the calculated twists we obtain $\mathbf{D}_u(0)$ for each edge, and store all values that will be used later in the topological data structure, as attributes of the edge.

Finally, all edges are traversed, and the cross-boundary derivatives are "assembled" for each one, according to Equation 1.

All polynomials are manipulated and stored in Bezier form for several reasons: intuitive geometrical meaning; numerical stability; easy operations, especially derivation, subdivision and degree elevation; and because this is the representation used for the final patches.

4.3. Face-Splitting Curve Construction

The subdivision of the faces introduces several degrees of freedom in the construction scheme. A middle-point and an associated tangent plane have to be determined for each face, and then the subdividing curves have to be constructed (Figure 1).

For each curve, its derivative at the joint with the original edge is already determined by the cross-boundary derivative field computed in Section 4.2. We also have to obey, for the triangular case, the constraint that the middle point has to be placed such that it is the barycenter of the triangle formed by the curve-derivatives emanating from it.

As for biquartic patches the curves can be cubic (from Equation 1, with $P(u)$ quadratic), there is total flexibility for positioning the face middle point. We specify the middle point as the average of the points obtained by passing cubic curves from each vertex to the opposite edge as in Nielson's method. The middle normal is obtained as an average of the three normals given by taking the cross product between each two derivatives of these cubic curves at the face middle point (u = 2/3 for the curves, coming from the vertex).

The derivative at the edge middle point is directly obtained from the cross-boundary derivative field previously constructed. At the vertex, the derivative direction is taken as the bisecting direction of the derivatives of the two incoming curves, and its magnitude as the average of their magnitudes.

At the face middle point, the derivatives are determined as follows: three default values are determined in the same fashion as described in Section 4.1 for the construction of the original curves. Then they are adjusted to satisfy the introduced constraint, which for triangles implies that they sum up to zero.

4.3.1 Comments

There is plenty of room for improvements, since many degrees of freedom for fixing the middle point and constructing the subdividing curves are available.

Like the setting of degrees of freedom during the construction of the original curves, this problem is a candidate for numerical minimization of some (global) "fairness" functional, following the ideas discussed in [Cel91, Mor92]. The drawbacks would be the high execution times and robustness concerns.

4.4. Cross-Boundary Derivatives for the Face-Splitting Curves

The cross-boundary derivatives along the subdividing curves are "assembled" using a procedure similar to the one described in Section 4.2, once the twists around the middle point are computed and $\mathbf{D}(u)$ determined.

As mentioned before, the scalar blending functions $A^{(i)}(u)$ are now made constant by imposing, by construction, a constraint to the tangent vectors at the center point, and appropriately specifying $\mathbf{D}(u)$. $P^{(i)}(u)$ will be quadratic, interpolating the values $P^{(i)}(0)$, $P^{(i)}(1)=0$ and $P_u^{(i)}(1)=0$ (with u=0 at the middle point).

As before, $\mathbf{D}(u)$ is cubic, interpolating $\mathbf{D}(0)$, $\mathbf{D}(1)$, $\mathbf{D}_u(0)$ and $\mathbf{D}_u(1)$. The direction of $\mathbf{D}(0)$ is again made orthogonal to $\mathbf{r}_u(0)$, but it is scaled such that $A^{(1)}(0) = A^{(2)}(0) = 1$. $\mathbf{D}_u(0)$ is obtained from Equation 3, somewhat simplified now, after the twists around the middle point are computed. $\mathbf{D}_u(1)$ is also obtained from Equation 3 using the twist read from the previously determined cross-boundary derivative of the original boundary curve at the junction point. Actually, after the simplifications it turns out to be equal to that twist.

4.5. Interior Control Point

After the boundaries and cross-boundary derivatives are computed, a biquartic patch still has one control point remaining to be determined, which will affect only the interior shape of the patch.

Several ways could be devised for setting this point. The objective here is to look for the less objectionable one, as "compatible" as possible with the information already generated.

For comparison, we tried two different methods. However, the difference in the final shaded surface was imperceptible in the examples used.

The first was the simplest one imaginable: averaging the surrounding control points.

As a second method, we obtained this control point by degree elevating the bicubic patch with the same boundary curves (already cubic) and with the same twists at the corners.

5. Results

The following figures show the results of the application of the method to the test data sets shown in Figure 4, an octahedron and a torus (similar to the ones in [Man92], for comparison).

Figure 5 shows the structure of the resulting tensor product patches.

Figure 6 shows the results obtained by the tensor product interpolant with isophotes superimposed. The surface quality is similar to that obtained by other methods, with noticeble ridges and undulations.

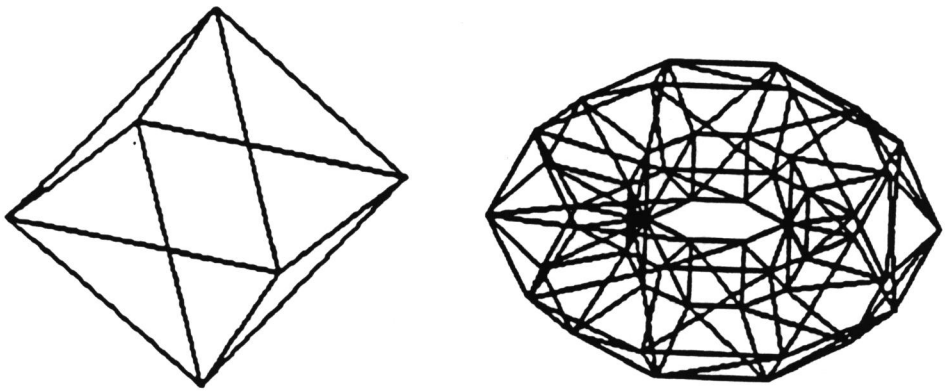

Figure 4. Test data sets.

Testing increasing and decreasing tangent values in the curves construction, the isophotes kept basically continuous, with minor gaps at very few spots appearing for high tangent values, beyond which the surface started to present cusps and loops.

As an example of the convenience of dealing exclusively with tensor-product surfaces, the surfaces presented were rendered by directly using the NURBS primitive of the graphics library (SGI's GL). As a result, the patches did not need to be tessellated, and other features offered by the library, such as trimming and texture mapping, could have been automatically used.

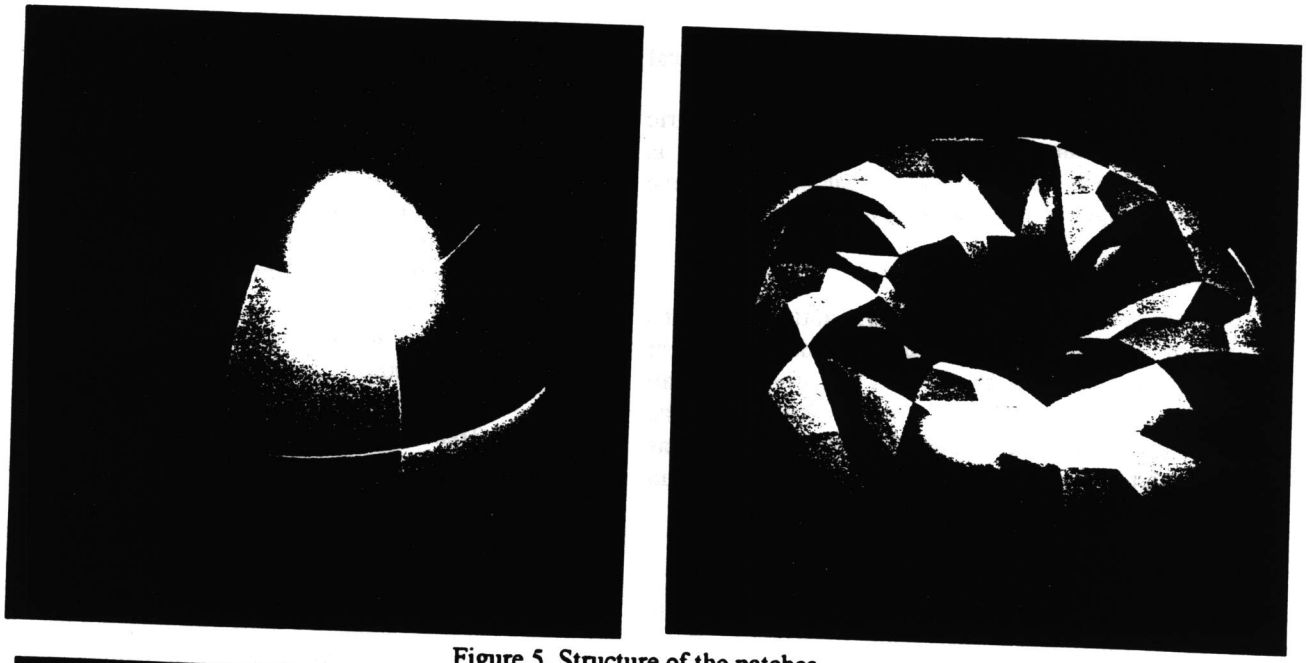

Figure 5. Structure of the patches.

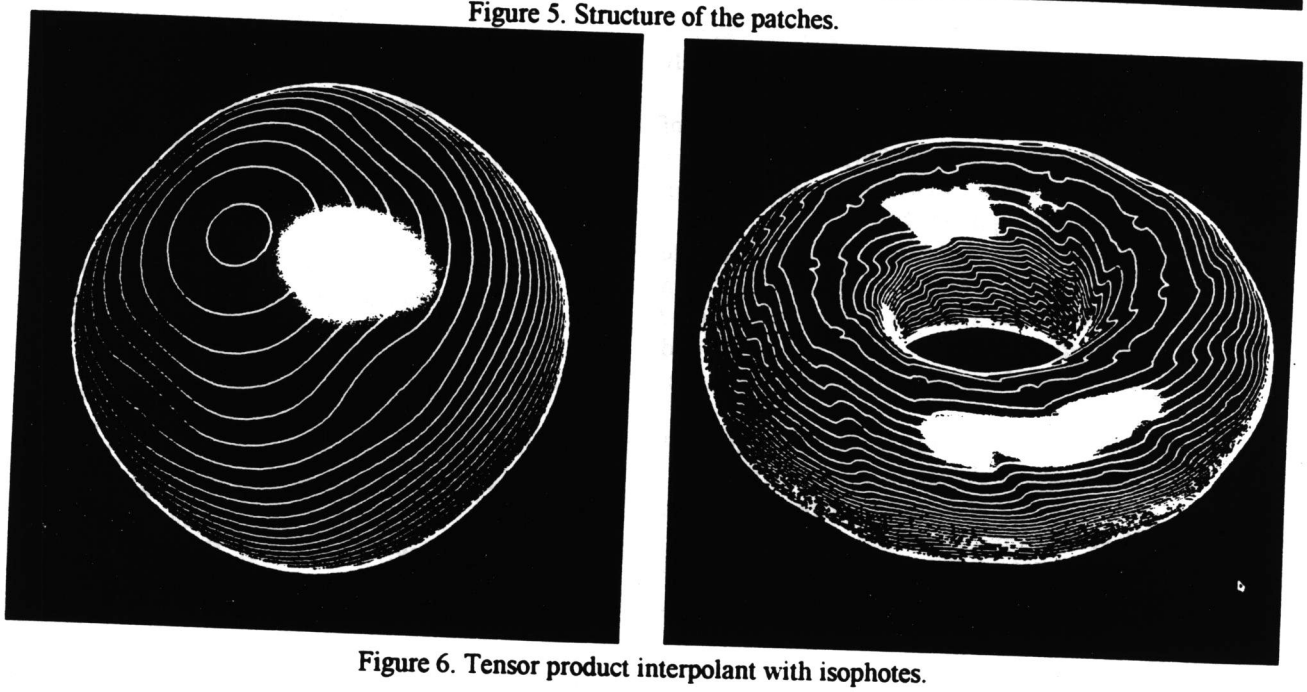

Figure 6. Tensor product interpolant with isophotes.

6. Concluding Remarks

We propose a piecewise surface construction scheme which fits biquartic polynomial tensor-product surfaces to a general network of cubic curves joining a 3D mesh of points. The resulting surface is not analytically G^1 if the network contains vertices adjacent to an even number of curves and is not "compatible", but the method leads to cross-boundary derivatives which minimize the twist discontinuities at the vertices. As noticed in the results from the implementation, this can lead to a final surface quality equivalent to that obtained by other local methods, even for test cases composed exclusively of "even" vertices.

6.1. Future Work

The quality of the surfaces obtained by local parametric interpolants is still not as good as one might desire, due to the fact that the amount of given data is not enough to match the number of degrees of freedom necessary to satisfy the problem requirements. For the method proposed, more research concerning the setting of these remaining degrees of freedom needs to be done. One

possibility to consider is the use of numerical optimization techniques.

Since in general only approximate geometric continuity is achieved, one issue to be investigated is the use of lower degree (bicubic) patches, by degree reducing the quartic cross boundary derivatives.

Acknowledgements

This work was done during the first author's Master's program at Arizona State University, under a Fulbright Scholarship sponsored by the U.S.I.A., the American Chambers of Commerce for Brazil and the Encyclopaedia Britannica. Special thanks to Dr. Tamas Varady for the generous collaboration. Support from Petrobras is also gratefully acknowledged.

References

[Cel91] G. Celniker and D. Gossard. "Deformable Curve and Surface Finite Elements for Free-Form Shape Design". *Computer Graphics (Proc. SIGGRAPH)*, Vol. 25, No. 4, July 1991, pp. 257-266.

[Chi83] H. Chiyokura and F. Kimura. "Design of Solids with Free-Form Surfaces". *Computer Graphics (Proc. SIGGRAPH)*, Vol. 17, No. 3, July 1983, pp.289-298.

[DeR92] T. DeRose and S. Mann. "An Approximate G^1 Cubic Surface Interpolant". in *Mathematical Methods in Computer Aided Geometric Design II*, T. Lyche and L.L.Schumaker, eds., Academic Press, pp. 185-196.

[Far82] G. Farin. "A Construction for Visually C^1 Continuity of Polynomial Surface Patches". *Computer Graphics and Image Processing*, 20 (1982), pp. 272-282.

[Gre86] J. Gregory. "N-sided Surface patches". in *The Mathematics of Surfaces*, J. Gregory, ed., Clarendon Press, Oxford, 1986, pp.217-232.

[Hag92] H. Hagen, S. Hahmann and T. Schreiber. "Surface Interrogation Algorithms". *IEEE Computer Graphics and Applications*, September 1992, pp.53-60.

[Hah89] J. Hahn. "Filling Polygonal Holes with Rectangular Patches", in W. Strasser and H.P. Seidel, ed., *Geometric Modeling: Algorithms and New Trends*, Springer - Verlag, 1989, pp.81-91.

[Lou92] M. Lounsbery, S. Mann, and T. De Rose. "Parametric Surface Interpolation". *IEE Computer Graphics and Applications*, September 1992, pp.45-52.

[Man92] S. Mann et al., "A Survey of Parametric Scattered Data Fitting Using Triangular Interpolants", in *Curve and Surface Design*, H. Hagen, ed., SIAM, Philadelphia, 1992, pp.145-172.

[Mor92] H.P. Moreton and C.H. Sequin, "Functional Optimization for Fair Surface Design", *Computer Graphics (Proc. SIGGRAPH)*, Vol 26, July 1992.

[Nie87] G. M. Nielson, "A Transfinite, Visually Continuous, Triangular Interpolant", in *Geometric Modeling: Algorithms and New Trends*, G. Farin, ed., Society for Industrial and Applied Mathematics, Philadelphia, 1987, pp.235-246.

[Nie93] G. M. Nielson, "CAGD's Top Ten: What to Watch", *IEEE Computer Graphics and Applications*, January 1993, pp. 35-37.

[Pet91] J. Peters, "Smooth Interpolation of a Mesh of Curves", *Constructive Approximation*, 7 (1991), pp. 221-246.

[Pre88] W. H. Press et al, *Numerical Recipes in C*, Cambridge University Press, 1988.

[Rei93] L.P. Reis, "Surface Interpolation with Polynomial Tensor Product Patches", Master's Thesis, Arizona State University, August 1993.

[Sar87] R. Sarraga, "G^1 Interpolation of Generally Unrestricted Cubic Bezier Curves", *Computer Aided Geometric Design*, 4 (1987), pp.23-40.

[Var87] T. Varady, "Survey and New Results in n-sided Patch Generation", in *The Mathematics of Surfaces II*, R. Martin, ed., Clarendon Press, Oxford, 1987, pp.203-235.

[Var93a] T. Varady and A. Rockwood, "Vertex Blending: the Control Frame Approach", Preprint, June 1993.

[Var93b] T. Varady, Manuscript and Personal Communication, May 1993.

[Wat88] M. A. Watkins, "Problems in Geometric Continuity", *Computer Aided Design*, 20 (8), pp. 499-502.

[Wij84] J. J. van Wijk, "Bicubic Patches for Approximating Non-Rectangular Control-Point Meshes", *Computer Aided Geometric Design*, 3 (1), pp. 1-13.

[Zha92] Y. Zhao and A. Rockwood, "A Convolution Approach to N-Sided Patches and Vertex Blending", Preprint, 1992.

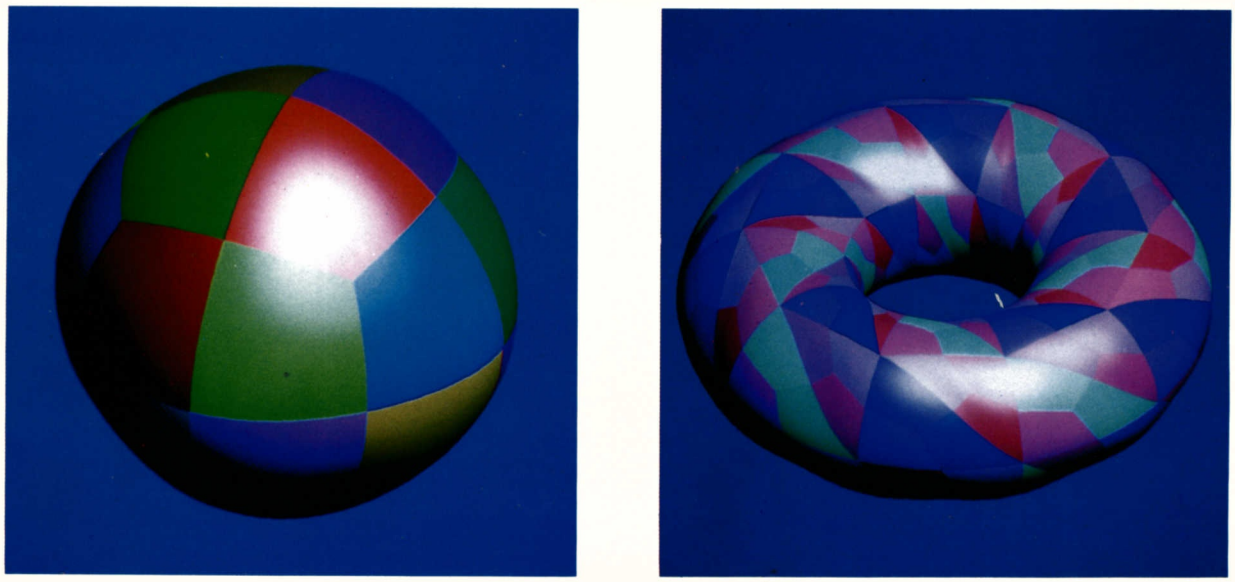

Figure 5

Figuras a cores do artigo ***Surface interpolation with tensor product patches***..