Parametrizing Singularly to Enclose Data Points by a Smooth Parametric Surface

Jörg Peters
IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598

Abstract

The problem of enclosing a data point by a smooth piecewise polynomial surface is solved with the help of a singular parametrization. The technique is illustrated by an algorithm that constructs a $C^1$ surface interpolating 3D positional and normal data.

Résumé

Une surface lisse à pièces polynomiales est construit autour d’un point par une paramétrisation singulière. La technique est illustrée par un algorithme que construit une surface $C^1$ qu’interpole des dates 3D.

Key words: Bernstein-Bézier form, $C^1$ surface, interpolation, singular parametrization.

1. Introduction

Enclosing a data point as a vertex by a complex of patches is one of the more difficult tasks when constructing smooth surfaces. Intuitively, while the first $n-1$ patches around the vertex need only join across one edge, the last has to match across two edges. When all patches are to be determined simultaneously, the problem is a circular dependence among the smoothness constraints. In general, this dependence makes it impossible to construct a smooth, regularly parametrized interpolant (with one polynomial patch per mesh facet) to a given mesh of curves [Peters '89]. Even if no mesh of curves but just discrete data are prescribed, this implies that care has to be taken when generating the patch boundaries. Three techniques are known to overcome the problem. Gregory’s rational patches [Gregory '74] break the dependence by allowing for a discontinuity in the second derivative: the first mixed derivative is not unique. A similar limitation of continuity within a mesh facet is achieved by splitting each patch into two and then joining the pair $C^1$ rather than $C^2$ (see e.g. [Farin '83]). A third technique is to force the boundary curves to match a second fundamental form at the vertex [Peters '89]. This is, for example, the case when four patches meet and the first and third, and the second and the fourth boundary curve join with high continuity (see e.g. [Bézier '77][Sarraga '86]). In particular, this explains why the vertex enclosure problem is not noticed for tensor-product constructions ([Coons '67], [Gordon '69]). On the other hand, it makes clear that the ease of such constructions is due to the special data and hence that it is difficult to extend them to the general case.

This paper offers a new, fourth alternative, namely singular (re)parametrization of the patches at the vertex. Rather than separating mixed derivatives, the idea is to make them trivially agree by setting the first derivative of each boundary curve and all mixed derivatives at the vertex to zero. By forcing the second boundary derivatives into a common plane, they take over the role of the first derivatives in defining the tangent plane. In more detail, consider a parametrization $p$ that maps the unit triangle or the unit square to $\mathbb{R}^3$. Denote the partial derivative in the direction of the $i$th unit vector by $D_i$, abbreviate

$$p_{i...j} := D_i \ldots D_j p(0,0)$$

and let $P := p(0,0)$ with $N$ the normal at $P$. Then the construction enforces

$$p_1 = p_2 = p_{12} = 0 \quad \text{and} \quad p_{11} \perp N, p_{22} \perp N. \quad (1.1)$$

Note that while setting $p_{12} = 0$ induces a flat spot on the graph of a bivariate map into $\mathbb{R}$, this is in general not the case for a parametric surface (the target of a map from $\mathbb{R}^2$ to $\mathbb{R}^3$).

Section 2 reviews the vertex enclosure problem and proves the sufficiency of constraints 1.1. As an illustration, Section 3 gives a simple algorithm for the interpolation of a mesh of points and normals in 3-space.

Graphics Interface '91
2. The vertex enclosure problem

First-order continuity or oriented tangent plane continuity between two patches \( p \) and \( q \) can be characterized by the constraint

\[
0 = f(u) := \lambda(u)D_1p(u, 0) - \mu(u)D_2p(u, 0) - \nu(u)D_2q(u, 0)
\]

where \( \lambda, \mu \) and \( \nu \) are univariate scalar-valued functions such that \( \lambda \mu > 0 \) (see e.g. [Liu '86 p.437], [Peters '88], [Degen '89 p.10], [Liu, Hoschek '89 Thm.1]). Setting \( f(0) = 0 \) for given \( p_1, p_2 \) and \( q \) pins down \( \lambda(0), \mu(0), \nu(0) \) except for a common factor. Setting \( f'(0) = 0 \) for given \( \lambda(0), \mu(0), \nu(0) \), \( p_1, p_2 \) and \( q_1 \) leads to

\[
\mu p_{12} + \nu q_{12} = \lambda p_{11} + (\lambda' p_1 - \mu' p_2 - \nu' q_2) \quad \text{at } 0.
\]

For each of \( n \) boundary curves emanating from the vertex \( P := p(0, 0) \), there is one such constraint. Denote the \( i \)-th patch by \( p_i \) and let \( N \) be the normal at \( P \). Since \( N \ast (\lambda'(0)p_1 - \mu'(0)p_2 - \nu'(0)q_2) = 0 \) and \( p_{12} = p_{21} \), there is an \( n \times n \) system of constraints

\[
\mu N \ast p_i' + \nu N \ast p_i'_{12} + \lambda N \ast p_i'_{11} = 0. \tag{2.4}
\]

In general, for given \( p_{11} \), this system is inhomogeneous and cannot always be solved since the constraint matrix is rank deficient whenever \( n \) is even (see e.g. [Watkins '88]).

Equation 2.4 can, however, be trivially enforced by using a singular parametrization, i.e. a parametrization \( p \) such that \( \det Dp = 0 \). This is unconventional from the point of view of differential geometry which emphasizes and often restricts itself to regular parametrizations ([do Carmo '76 p.52],[Klingenberg '83, 3.1.1]) since then smoothness of the parametrization implies smoothness of the surface (e.g. [do Carmo '76 Prop. 3, p. 63]). Checking smoothness at singular points is more complicated as simple examples, e.g. \( t \mapsto (t^2, t^3) \) and \( t \mapsto (0, t^3) \) show: both parametrizations are smooth and singular at 0; however, the first curve is not first-order continuous at 0, while the second is.

(2.5) Lemma. A vertex can be enclosed by a \( C^1 \) surface if the boundary curves emanating from it are parametrized singularly.

Proof. Parametrize so that for \( j \in \{0, 1\} \) and \( q_i \) a tangent vector for the \( i \)-th boundary curve,

\[
D_j p_i |_0 = 0 \quad \text{and} \quad D_j^2 p_i |_0 = q_{i+j}. \tag{1.1'}
\]

Then \( f'(0) = 0 \) for any choice of \( \lambda'(0), \mu'(0) \) and \( \nu'(0) \), \( \lambda'(0) = 0 \) and \( p_{12} = 0 \) imply \( f'(0) = 0 \) and the tangent plane is defined by the second derivative since \( (\alpha D_1 + (1 - \alpha) D_2)^2 p_j |_0 = \alpha^2 q_j + (1 - \alpha)^2 q_{j+1} \). ♦

The algorithm in Section 3 constructs boundaries that satisfy (1.1) and shows that the technique, while not profound, is effective. The alternative approach, setting \( N \ast p_{11} = 0 \) to make (2.4) homogeneous, leads to flat spots if \( p_{11} \neq 0 \).

3. An algorithm for example

The following algorithm represents bivariate vector-valued patches in Bernstein-Bézier-form (see e.g. [Farin '86], [de Boor '87]) to have easy access to value and derivative information along patch boundaries. In Figure 3.2 the natural association of coefficients with the unit domain is used to label the 21 coefficients of a quintic 3-sided (total-degree) patch. Enforcing \( p_1 = p_2 = p_{12} = 0 \) at \( P \) means setting

\[
B_{ij} = B_{ik} = C_i = P_i.
\]

That is, the algorithm below constructs 3-sided quintic patches with at most 12 distinct coefficients. The reader can check that a similar 4-sided biquartic (tensor-product) patch has 12+1 distinct coefficients (the center coefficient will be free to choose). Still the patches are underconstrained: the second difference vector \( B_{ij} - B_{ik} \) needs only lie in the tangent plane. A smooth join across the edge \( ij \) is achieved by determining \( D_{ij}, \) resp. \( D_{ki} \), appropriately. For no better reason other than the author's familiarity with the approach, smoothness across the boundaries is achieved by prescribing the boundary normal as a weighted linear blend of the normals at the end points ([Peters '88]). Thus, setting

\[
n(u) := (1 - u)N^0 + \omega u N^1,
\]

where \( \omega := \frac{N^0(P^0 - P_1)}{N^1(P^1 - P_0)} \geq 0 \)

Graphics Interface '91
transforms the non-linear smoothness constraint
\[
det(D_1 \mathbf{p}(u, 0), D_2 \mathbf{p}(u, 0), D_2 \mathbf{q}(u, 0)) = 0
\] (3.1)
into the linear constraints
\[
\begin{align*}
n(u)D_1 \mathbf{p}(u, 0) &= 0, \\
n(u)D_2 \mathbf{p}(u, 0) &= 0, \\
n(u)D_2 \mathbf{q}(u, 0) &= 0,
\end{align*}
\]
for each edge and \( u \in [0..1] \). (Constraint (3.1) is equivalent to (2.2).) We note that not all data allow for a linearly varying normal. For the present purpose, however, we refuse to worry and refer to [Peters '88] for solutions. Since \((e_u), (e_v)\) and \((e_w)\) are trivially enforced at \(u = 0\) and \(u = 1\), the only work consists of pinning down the underdetermined interior coefficients, \(D_{ij}\). One option is to force the surface to be close to a cubic-biquadratic interpolant of the same data. However, this leads to 'bulgy' surfaces (cf. Figure 3.4). A better choice is to minimize the variation of the cross-boundary derivative and this is done below.

\[B_{ij}^2 = B_{ik}^2 D_{ki},
\]

(3.2) Figure: BB-coefficients for a singular quintic parametrization.

Algorithm

**Input** A mesh of data points and their normals such that \( \omega > 0 \). Each facet has 3 or 4 edges.

**Output** A quintic-biquartic \( C^1 \) surface that interpolates the mesh.

**Tools** * is the vector product, \( \times \) the cross product. 
\quad \text{nbrs}(k) \rightarrow \text{returns the number of neighbors of point k.}
\quad \text{nbr}(k, i) \rightarrow \text{returns the i\textsuperscript{th} neighbor of point k.}
\quad \text{sds}(k, i) \rightarrow \text{returns the number of edges of the i\textsuperscript{th} facet attached to point k.}
\quad \text{sds}(f) \rightarrow \text{returns the number of edges of the facet f.}
\quad \text{tanproj}(v, n) \rightarrow \text{returns} \ (v - v \times n) n, \text{the vector component of} \ v \text{perpendicular to} \ n.

\[\text{for } i = 1:\text{points} \quad [\text{construct the boundaries } B_{ij}^2] \]
\[\text{for } m = 1:\text{nbrs}(i) \quad l = \text{nbr}(i, m); \quad [\text{e.g.} \ l = j \text{ or } l = k] \]
\[\text{[singularity at } P_i] \]
\[B_{ij}^1 = P_i; \; C_{ik} \leftarrow P_i; \]
\[\text{[construct the } B_{ik}^2] \]
\[\Delta = P_i - P_j; \sigma = N_i \times \Delta; \]
\[\text{if quintic then} \]
\[t \leftarrow \tanproj(\Delta, N_i); \]
\[\alpha \leftarrow 3\sigma/(5t \times N_i); \]
\[\text{if } \alpha \leq 0 \text{ [check for inflection]} \]
\[\text{error} [\text{no linear } n(u) \text{ possible}]; \]
\[\text{else [bicubic patch]} \]
\[m \leftarrow N_i \times N_i; \]
\[t \leftarrow m \times N_i; \]
\[\alpha \leftarrow \sigma/(m \times m); \]
\[\text{endif} \]
\[B_{ij}^2 \leftarrow P_i + \alpha t; \]
\[\text{for } f = 1:\text{faces} \quad [\text{construct the } D_{ij}] \]
\[\text{for } i = 1: \text{sds}(f) \quad [j = i+1 \text{ etc. below}] \]
\[d_i = (B_{ij}^2 - i, i+1 - P_i + B_{ij+1, i+2}^2 - B_{ij+1, i+1}/2; \]
\[F_i = (B_{ij}^2 + B_{ij+1, i+1})/2; \]
\[D_{ij}^* = d_i + F_i; \quad \text{[target value for } D_{ij, i+1}] \]
\[\text{solve} \]
\[\begin{bmatrix}
1 & N_i N_{i+1} \\
N_i N_{i+1} & 1
\end{bmatrix}
\begin{bmatrix}
l_1 \\
l_2
\end{bmatrix} =
\begin{bmatrix}
1 \\
N_i N_{i+1}
\end{bmatrix}
\] (3.3)
\[\begin{bmatrix}
(D_{ij}^* - B_{ij+1, i}^2) \times N_i \\
(D_{ij}^* - B_{ij+1, i+1}^2) \times N_{i+1}
\end{bmatrix}; \]
\[D_{ij, i+1} \leftarrow D_{ij}^* - l_1 N_i - l_2 N_{i+1}; \]
\[\text{if sds}(f) = 4 \]
\[\text{bicubic center coefficient} \leftarrow \text{average} \]
\[\text{of the surrounding coefficients} \]

The interpolants in Figures 3.4-6 fit to the same 5 vertices of an upside-down pyramid with square base. However, the first aims at minimizing the distance to a cubic-biquadratic interpolant, the second is the output of the above algorithm and the third is obtained by adjusting the tightness across one boundary by setting
\[D_{ij}^* = d_i + F_i + \gamma N, \]
and increasing \( \gamma \) from 0 to 5. \( N \) is the average of the normals at the vertices of the patch. The figures show two of the four 3-sided and the 4-sided patch. The checker pattern follows isoparametric lines and thus points to the singular vertices. Figure 3.7 shows the author's favorite object of genus 2 constructed with 50 patches. The effect of adjusting 'tightness' (altering the least

Graphics Interface '91
squares target value) is illustrated in Figures 3.8 and 3.9.

Picking and deforming a patch is done in real time—the recomputation involves only one patch. Shading requires some care: since the parametric derivative is zero, the normal at a singular vertex cannot be computed from the parametrization. However, the normals are part of the input data and hence need not be recomputed. Adaptive subdivision does the rest.

(3.4) Figure: Degree-minimizing interpolant to 5 points bounding an (inverted) pyramid. The square base gives rise to the 4-sided patch on top.

4. Conclusion

A new technique, singular polynomial parametrization at the vertices, is shown to overcome the problem of enclosing a vertex by a complex of patches. This means that one needs not resort to rational patches or change the topology by splitting patches in order to build smooth, interpolating surfaces of arbitrary genus. However, singular polynomial patches are of slightly higher degree and the calculation of normals close to the vertex requires care.

Acknowledgement: This research was supported by NSF DMS-8701275.

(3.5) Figure: Interpolant to 5 points minimizing the cross-boundary derivative (default: $\gamma = 0$).

(3.6) Figure: Interpolant to 5 points with $\gamma = 5$.

5. Appendix: Derivation of the algorithm

We first consider a quintic triangular patch and use the abbreviation

$$ p \sim [b^0, \ldots, b^d] \quad \text{for} \quad p: t \mapsto \sum_{j=0}^{d} t^j (1-t)^{d-j} b^j, $$

Graphics Interface '91
so that
\[ [b^0, \ldots, b^d][c^0, \ldots, c^d] = [b^0 c^0, \ldots, \sum_{k+l=j} b^k c^l, \ldots, b^d c^d], \]
where \( \langle b, c \rangle \) is the scalar product of \( b \) and \( c \). Since
\[
np_1 \sim [n^0, n^1][u^0, 4u^1, 6u^2, 4u^3, u^4] \]
\[
:= [N_i, \omega N_j][B^1_{ij} - P_i, 4(B^2_{ij} - B^1_{ij}), 6(B^3_{ij} - B^2_{ij}), 4(B^4_{ij} - B^3_{ij}), P_j - B^1_{ij}],
\]
is a polynomial of degree five and the setup is symmetric, it suffices to show that the first three coefficients of \( np_1 \),
\[
n^0 u^0, \quad n^1 u^0 + 4n^0 u^1, \quad 4n^1 u^1 + 6n^0 u^2,
\]
are zero. By choosing \( B^1_{ij} = P_i \) and forcing \( B^2_{ij} \) to lie in the tangent plane, e.g. \( B^2_{ij} = P_i + \alpha t \) with \( t \) the projection of \( P_j - P_i \) into the tangent plane at \( P_i \), the first two coefficients vanish and the boundary curve is normal to \( N_i \). It remains to show that
\[
6u^2 n^0 + 4u^1 n^1 = 0 \quad \text{and} \quad 4u^2 n^0 + 6u^2 n^1 = 0.
\]
Since \( u^2 = (P_j - P_i) - u^1 - u^3 =: \Delta - u^1 - u^0 \), we obtain
\[
6u^2 n^0 \Delta + 4u^1 n^1 = 6\Delta n^0 - 6u^3 n^0 + 4u^1 n^1
\]
\[
6u^2 n^1 \Delta + 4u^3 n^0 = 6\Delta n^1 - 6u^1 n^1 + 4u^3 n^0
\]
or, equivalently,
\[
5u^1 n^1 = 3\Delta (2n^0 + 3n^1)
\]
\[
5u^3 n^0 = 3\Delta (3n^0 + 2n^1).
\]
Hence, the third coefficient vanishes if
\[
\alpha = \frac{3\Delta (2n^0 + 3n^1)}{5\Delta} = \frac{6\Delta N_i/\omega + 9\Delta N_j}{5t N_j}
\]
\[
= \frac{-6\Delta N_j + 9\Delta N_j}{5t N_j} = \frac{3\sigma}{5t N_j}.
\]

Graphics Interface '91
The analysis of the biquartic case is simpler. Again, choosing $B_{ij} = P_i$ sets to zero the first and the last entry in

$$np_i \sim [n^0, n^1][u^0, 3u^1, 3u^2, u^3] =: [N_i, \omega N_j]$$

$$[B_{ij}^0 - P_i, 3(B_{ij} - B_{ij}^1), 3(B_{ij}^1 - B_{ij}), P_j - B_{ij}^3].$$

Forcing the middle coefficient of the boundary, $B_{ij}$, to lie on the intersection of the tangent planes at $P_i$ and $P_j$ sets to zero the second and fourth entry:

$$n^0(B_{ij} - B_{ij}^0) = 0 = n^1(B_{ij}^1 - B_{ij}).$$

The remaining term is zero by choice of $\omega$:

$$n^0(B_{ij}^1 - B_{ij}) + n^1(B_{ij} - B_{ij}^1)$$

$$= (n^0 + n^1)(B_{ij}^1 - B_{ij})$$

$$= (N_i + \omega N_j) \Delta = 0.$$


