Better Pasting Through Quasi-Interpolation

by

Blair Conrad

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 1999

Abstract

Hierarchical surface pasting [Bar94], developed by Barghiel, Bartels and Forsey, is an interactive surface modelling technique that is used to add local detail to a tensor product B-spline surface without increasing the overall complexity of the original surface. Surface pasting approximates displacement mapped surfaces by placing additional tensor product surfaces, called features, on the base surface. The features may be scaled, rotated, and translated arbitrarily over the base surface, and the composite surface can be evaluated using relatively little computational power. Because the feature only approximates a displacement map, a surface produced using standard surface pasting often has noticeable gaps at the edges of the pasted feature. The severity of the surface discontinuities may be made as small as desired via knot insertion, but this can result in an unacceptable degradation in the performance of the modelling software.

Quasi-interpolation [LS75] is an approximation method that approximates curves with spline curves to a high degree of accuracy. The approximation is constructed by computing coefficients that are used to weight samplings of the curve to be approximated. The Lyche-Schumaker quasi-interpolant uses coefficients that are inexpensive to compute and samplings that are relatively expensive to compute. I propose an improved surface pasting technique that uses quasi-interpolation to set the feature's boundary control vertices. For surface pasting it is necessary to have quasi-interpolants that reproduce position and derivatives at the endpoints of the curve to be approximated. In addition, as the feature surface is moved, the curve samplings must be recomputed, but the coefficients remain fixed. Therefore, I have developed a variation of the quasi-interpolant whose coefficients are expensive to compute, but whose samplings are relatively inexpensive to compute.

Surface pasting with quasi-interpolation produces features whose boundaries approximate the base surface to within the same tolerance as those produced by standard pasting, but with one third the boundary control vertices. The resulting feature has one ninth the total control vertices and can be constructed in approximately one tenth the time as with standard surface pasting.

Acknowledgements

I would like to thank my supervisor, Stephen Mann, for guiding me as I wrote this thesis. His knowledge, experience and compassion have made him a valued advisor and friend.

I am grateful to my readers, Peter Forsyth and Richard Bartels, who took the time to suggest many helpful ideas for improving this thesis. I especially appreciate the efforts of Richard Bartels, who initially donated Maple code for performing quasi-interpolation, and who has provided many valuable insights during the course of my research.

I am indebted to Kirk Haller, who was always available to explain the finer points of any subject about which I was confused, and to Tom Lyche, who originally suggested the topic of this research.

I would like to thank the many friends I have had at the University of Waterloo for making my stay here more enjoyable, and my family for their continuing support.

Financial support for the research in this thesis was provided by NSERC, CITO, and the University of Waterloo.

Finally, I would like to thank Marryat Ma, without whom I would be lost.

Trademarks

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Spline curves and surfaces are used in many areas of computer graphics and computer aided geometric design. In particular, tensor product B-spline surfaces are commonly used in modelling and computer animation because they have many attractive properties, such as a compact representation and adjustable levels of internal continuity [Far93]. A tensor product B-spline surface can be edited by subjecting it to affine transformations, by manipulating the positions of the control vertices that define the surface's shape, or by adjusting the knot vectors that determine the parameterisation and continuity of the surface.

Frequently, the user of a piece of modelling or animation software will want to add a region of local detail to a B-spline tensor product surface, but the knot structure will be too coarse to allow the fine-grained control that the user desires. Traditional methods of increasing the complexity of the surface include inserting knots using either Boehm's algorithm [Boe80] or the Oslo algorithm [CLR80]. The insertion of a knot into either of a surface's knot vectors causes an entire row or column of subpatches to be split — rather than increasing the number of subpatches locally, extra subpatches are created across the width or breadth of the surface. The presence of the additional subpatches increases the space needed to store the surface and complicates the task of making coarse-grained adjustments to the surface.

Forsey and Bartels [FB88] developed an alternative surface representation that allows the user to add local detail to a tensor product B-spline surface. Their representation, each instance of which is called a *hierarchical B-spline*, allows the user to add a detail surface, called an *overlay*, to a tensor product B-spline. A region of the base surface is selected and knot insertion is applied within that region, after which the new control vertices can be manipulated and the positions are be stored as an *offset* relative a hierarchy of local reference frames. By fixing a number of the outermost layers of control vertices with each region of local detail, $\mathcal{C}^1$ or $\mathcal{C}^2$ surface continuity can be preserved.

Hierarchical B-splines suffer from a number of drawbacks: the regions of added detail must remain parametrically aligned with the base, and it is impractical to slide the features or to maintain a library of overlays that may be added to a base surface. Wavelets [Chu92] can be used in a similar hierarchical modelling technique, but they also require the detail regions to be parametrically aligned with the base.

Displacement mapping is another technique for adding local detail to a surface while adding as few extra control vertices as possible. A displacement mapped surface consists of a base surface and a displacement feature that are combined to form a composite surface. Every point on the feature is defined by a displacement vector relative to a certain point in a reference plane. An invertible transformation is used to embed the feature's domain into the base domain, and for a given value, say t, in the base domain, the composite surface point is defined by one of two constructions. If t is not in the image of the feature's domain, the surface point is taken to be the position of the base surface evaluated at t. If t is in the image of the feature's domain, a local coordinate frame is constructed on the base surface at t and the composite surface point is taken to be the the endpoint of the displacement vector found by evaluating the feature at the pre-image of t and expressing the vector relative to this coordinate frame. Further details may be added to the feature by hierarchically composing feature surfaces.

Via displacement mapping, a library of features may be maintained, each of which can be translated or rotated on a base surface by adjusting the transformation that embeds its domain into the base domain. By adjusting the level of continuity with which a feature meets its reference plane, composite features that exhibit any desired level of continuity may be created. The greatest disadvantage to using displacement mapped surfaces is that displacement mapping is expensive. Multiple surface evaluations, up to the depth of the surface hierarchy, must used to determine each point to be rendered on the composite surface.

Barghiel [Bar94] implemented a method for constructing surfaces that approximate displacement mapped surfaces, based on a technique suggested by Forsey and Bartels [BF91]. The new method, called *surface pasting*, is a generalisation of Forsey's method that was intended to combine the flexibility of displacement mapped surfaces with the speed of evaluation enjoyed by hierarchical B-splines. The technique used to construct a composite pasted surface is similar to that used to construct a displacement mapped surface. Two tensor product B-splines surfaces, a feature and a base, are defined, as is an invertible map from the feature's domain into the base's domain. The feature's domain is embedded into the feature's range space, and each feature control vertex is represented as a displacement vector from a corresponding point in the domain. The composite surface is constructed by evaluating the position of each pasted control vertex. A coordinate frame is constructed on the base surface and the control vertex's displacement vector is represented relative this frame. This technique has all the flexibility of displacement mapping, but it is much cheaper since only the control vertices of the feature must be mapped, rather than the larger number of surface points to be rendered. This combination of flexibility and speed has drawn the attention of the modelling industry to surface pasting — recent versions of *Houdini*, a commercial animation tool produced by Side Effects Software, have included support for surface pasting.

However, surface pasting is only an approximation, and as such it does not have the same continuity properties as displacement mapping. In general, there is no guaranteed continuity between the feature and the base surfaces. Composite surfaces that are constructed so a feature surface has few control vertices or a coarse knot structure relative the base often exhibit gaps between the feature and the base. It is possible, by inserting knots into the feature surface, to cause the boundary of the feature to approximate the base to within any desired tolerance, but often many knot insertions are required to approach the desired level of approximation, and the additional control vertices in the feature dramatically increase the cost of performing the pasting operation.

In this thesis, I suggest altering the surface pasting technique to improve the approximate continuity between the feature and base surfaces. I have chosen to modify the way in which tensor product B-spline surfaces are pasted rather than to work with other types of surfaces because of the advantages inherent in working with tensor products. Tensor product surfaces have guaranteed internal continuity if the knot vectors are set properly, whereas a feature constructed from triangular Bézier patches, for example, would not have guaranteed internal continuity and some effort would have to be expended to obtain a continuous surface. Triangular B-splines can have a guaranteed level of internal continuity, but these patches are computationally expensive to evaluate and thus are considered to be unsuitable for use in an interactive surface modeller.

I propose modifying the standard surface pasting technique to use two methods to place the control vertices of the pasted feature. The method of standard pasting is used to place the interior control vertices, while one or more of the outer rings of feature control vertices are placed using a spline approximation technique.

The spline approximation technique that I use in this thesis is *quasi-interpolation*, first developed by de Boor and Fix [dBF73]. The quasi-interpolation operators were later generalised by Lyche and Schumaker [LS75], and it is their version that I use in my alternative surface pasting technique. A quasi-interpolation operator approximates a curve by calculating coefficients that are used to weight samplings of the curve to be approximated. The Lyche-Schumaker quasi-interpolation operator uses coefficients that are inexpensive to calculate and samplings that are relatively expensive to calculate.

In this thesis, I develop a specialised class of quasi-interpolants based on position and derivative samples and use these operators to place the control vertices in the outer rings of the pasted features. My quasi-interpolation operators are designed to interpolate position and derivatives at the endpoints of the approximated curve. When a feature surface is moved along the base, the curve samplings must be recomputed to place the boundaries, but the coefficients remain fixed. Therefore, I have developed an alternative representation for my quasi-interpolation operators that relies on relatively inexpensive samplings and more expensive coefficients. I discuss the error bounds on approximations created using my quasi-interpolation operators. In particular, I show that the order of convergence is the best possible.

In addition, I describe the methods by which my quasi-interpolation operators may be used to create composite pasted surfaces in which the approximate continuity between the feature and base surfaces is greatly enhanced. I focus on methods of reducing the cost of using quasi-interpolation to paste the boundary control vertices. As a result, the per vertex cost of pasting a feature using my method is comparable to, and in some cases lower than, the cost of pasting a feature using the standard surface pasting method. It possible to construct a feature surface that approximates a base to the same level of tolerance as does a feature surface constructed using standard surface pasting, but with only one third the boundary control points. The resulting feature would have only one ninth the control points of feature pasted using the standard method, and the cost of pasting the feature using the improved method would be approximately one tenth that of pasting the original feature.

Chapter 2

Background

In this chapter, I provide background material on the subjects of to hierarchical modelling and quasi-interpolation. First, I give a brief introduction to B-spline curves and tensor product B-spline surfaces, followed by a discussion of several methods, including hierarchical surface pasting, for adding local details to surfaces. Finally I conclude with an examination of curve quasi-interpolation techniques and error bounds for generic quasi-interpolants.

2.1 B-spline Curves

Much of the information that I present in this and the following section have been presented by Farin [Far93] and others. However, I chose to include these sections to provide several important details which cannot be found in Farin's book, as well as to establish notation that will be used throughout the rest of this thesis.

A parametric B-spline curve $C(u)$ is a piecewise polynomial defined by a sequence of control vertices $\{P_i\}_{i=0}^{M}$ and associated basis functions $\{B_i^m(u)\}_{i=0}^{M}$:

$$C(u) = \sum_{i=0}^{M} P_i B_i^m(u).$$

Each of the basis functions is a piecewise polynomial of degree m. The degree of the B-spline curve is determined by the degree of the basis functions. Thus, $C(u)$ is a B-spline curve of degree m.

The basis functions are defined by a sequence of nondecreasing knot values, called a *knot vector*:

$$u_0 \leq u_1 \leq u_2 \leq \cdots \leq u_{M+m-1},$$

where at most m consecutive knots may be equal, i.e., $u_i < u_{i+m}$ for all i.

The univariate B-splines of degree k are defined recursively as follows:

$$\begin{aligned} B_i^0(u) &= \begin{cases} 1 & \text{if } u_{i-1} \leq u < u_i \\ 0 & \text{otherwise} \end{cases} \\ B_i^k(u) &= \frac{u - u_i}{u_{i+k} - u_i} B_i^{k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} B_{i+1}^{k-1}(u), \text{ if } k > 0. \end{aligned}$$

Note that using the above recurrence relationship to calculate the B_i^ms requires two extra knots, u_{-1} and u_{M+m}, which were not included in the original knot vector definition. For the purpose of this calculation, these "phantom knots" may be assigned values, $u_{-1} = u_0$ and $u_{M+m} = u_{M+m-1}$. The extra knot values are only of interest when writing a program to evaluate B-splines and will not be used further in this thesis.

Basis function $B_i^m(u)$ is non-zero over $[u_i, u_{i+m})$ and attains its maximum when evaluated at $\gamma_i = \frac{u_i + u_{i+1} + \cdots + u_{i+m-1}}{m}$. The value γ_i is called the i^{th} *Greville abscissa* of C. Since $B_i^m(u)$ attains its maximum at γ_i, it is said that $C(\gamma_i)$ is *maximally influenced* by P_i. A B-spline basis function is bounded on its domain as follows: $0 \leq B_i^m(u) \leq 1$, for $u \in [u_i, u_{i+m})$.

Bounds for the derivatives of the B-spline functions are given by Lyche and Schumaker [LS75]. Fix $0 < r \leq m$ and let u be such that there exists a p, $m - 1 \leq p \leq M - 1$, with $u_p \leq u < u_{p+1}$. If $u = u_p$, suppose that the value of u_p occurs at most $m - r$ times in the knot vector. Then

$$|B_i^{m\,(r)}(u)| \leq \frac{\Upsilon_{m,r}}{\Delta_{i,p,m} \cdots \Delta_{i,p,m-r+1}}, \tag{2.1}$$

where $\Delta_{i,p,j}$ is taken to be the minimum of $u_{\nu+j}-u_\nu$ for ν such that $u_i \leq u_\nu \leq u < u_{\nu+j} \leq u_{i+m+1}$, for all $j = m-r+1, \ldots, m$, and where

$$\Upsilon_{m,r} = \frac{m!}{(m-r)!}\binom{r}{\lfloor r \rfloor}.$$

Typically, a B-spline curve is only evaluated at those values of u for which $m+1$ basis functions may be non-zero. Thus, the domain of C is taken to be $[u_{m-1}, u_M)$. In practice, it is inconvenient to deal with half-open B-spline domains, so I will extend the domain of C to $[u_{m-1}, u_M]$ by defining $C(u_M) = \lim_{u \to u_M} C(u)$.

2.1.1 B-spline Blossoms

The *blossom* of a degree n polynomial $F(x)$ is a symmetric, multi-affine map $f(x_1, x_2, \ldots, x_n)$ such that $f(\overbrace{x, x, \ldots, x}^{n \text{ terms}}) = F(x)$. The notation $\varpi[F]$ will be used to denote the blossom of F.

The blossom of a piecewise polynomial may be found by individually blossoming each of the polynomial pieces of the function. Examining the blossom of a B-spline reveals a relationship between the B-spline's knot vector and its control vertices. If $C(u)$ is a B-spline curve of degree m, with control vertices $P_0, P_1, \ldots P_M$, knot vector $u_0, u_1, \ldots, u_{M+m-1}$ and blossom c, then $P_i = c(u_i, u_{i+1}, \ldots u_{i+m-1})$.

I now present a B-spline knot insertion algorithm based on the correspondence between the B-spline blossom and the B-spline control vertices. The B-spline curve contains contain one more control vertex after the knot insertion than it did before. If the knot value u, where $u_i \leq u \leq u_{i+1}$, is inserted, with $i \geq m-1$ and $i \leq M-1$, then the adjacent control vertices

$$\left.\begin{array}{l} c(u_{i-m+2}, u_{i-m+3}, \ldots, u_i, u_{i+1}) \\ c(u_{i-m+3}, u_{i-m+4}, \ldots, u_{i+1}, u_{i+2}) \\ \qquad\qquad\vdots \\ c(u_i, u_{i+1}, \ldots, u_{i+m-1}) \end{array}\right\} m-1 \text{ control vertices}$$

corresponding to the knot vector subsequence $u_{i-m+2}, u_{i-m+3}, \ldots, u_i, u_{i+1}, \ldots, u_{i+m-1}$ are replaced with

$$\left.\begin{array}{l} c(u_{i-m+2}, u_{i-m+3}, \dots, u_i, u) \\ c(u_{i-m+3}, u_{i-m+4}, \dots, u_i, u, u_{i+1}) \\ \qquad\vdots \\ c(u_i, u, u_{i+1}, \dots, u_{i+m-2}) \\ c(u, u_{i+1}, \dots, u_{i+m-1}) \end{array}\right\} m \text{ control vertices}$$

The new control vertex sequence corresponds to the new knot vector subsequence $u_{i-m+2}, u_{i-m+3}, \dots, u_i, u, u_{i+1}, \dots, u_{i+m-1}$.

As noted before, the B-spline blossom is symmetric and affine in each argument, so the new control vertices generated above may be constructed directly from the existing knot vector and control vertices. The following construction is used:

$$\begin{aligned} c(u_{k+1}, \quad \dots, \quad & u_i, u, u_{i+1}, \dots, u_{k+m}) \\ = \quad & \frac{(u_{k+m+1} - u)c(u_k, \dots, u_{k+m}) + (u - u_k)c(u_{k+1}, \dots, u_{k+m+1})}{u_{k+m+1} - u_k} \end{aligned} \tag{2.2}$$

2.1.2 Knot Multiplicity and Continuity

A knot vector may contain repeated knots. *Knot multiplicity* is the number of times a knot value occurs in a knot vector. If a knot has multiplicity equal to the degree of the B-spline curve, it is said to have *full multiplicity.* If knot value u_i has multiplicity k, then a degree m B-spline curve is $\mathcal{C}^{m-k}$ at u_i, as are the B-splines whose support contains u_i.

The position and derivatives at the ends of a B-spline curve are determined by a small number of its control vertices. In particular, for any $j \leq m$, the j^{th} derivative at the beginning (end) of the B-spline curve is uniquely determined by the first (last) $j + 1$ control vertices, and each of these control vertices has a non-zero contribution. Furthermore, if a B-spline curve has full knot multiplicity at each end, the position at the beginning of the curve coincides with the first control vertex, and the position at the end coincides with the last control vertex.

The first derivative at the beginning (end) of a curve whose end knots have full multiplicity is a scalar multiple of the vector difference of the first (last) two control vertices. The multiple is equal to the degree of the curve divided by the difference between the first (last) two distinct knot values. Thus, if $u_0 = u_1 = \cdots = u_{m-1} < u_m$ and $u_{M-1} < u_M = u_{M+1} \cdots = u_{M+m-1}$, then

$$\frac{dC(u_0)}{du} = \frac{m}{u_m - u_{m-1}}(P_1 - P_0) \text{ and } \frac{dC(u_M)}{du} = \frac{m}{u_M - u_{M-1}}(P_M - P_{M-1}). \tag{2.3}$$

Unless explicitly stated otherwise, the end knots of all B-splines mentioned in the remainder of this thesis are assumed to have full multiplicity.

Two B-spline curves can be joined together with varying levels of continuity. Suppose that in addition to C, there is a B-spline curve C^* with degree m^*, knot vector $u_0^*, u_1^*, \ldots, u_{M^*+m^*-1}^*$, with $u_0^* = u_1^* = \cdots = u_{m^*-1}^* < u_{m^*}^*$ and $u_{M^*-1}^* < u_{M^*}^* = u_{M^*+1}^* \cdots = u_{M^*+m^*-1}^*$, and control vertices $P_0^*, \ldots, P_{M^*}^*$. To have C and C^* meet with $\mathcal{C}^0$ continuity, it is sufficient to have $P_M = P_0^*$. To ensure a $\mathcal{C}^1$ join, the two curves must meet with $\mathcal{C}^0$ continuity and must have

$$\frac{m}{u_M - u_{M-1}}(P_M - P_{M-1}) = \frac{m^*}{u_{m^*}^* - u_{m^*-1}^*}(P_1^* - P_0^*), \tag{2.4}$$

as shown in Figure 2.1.

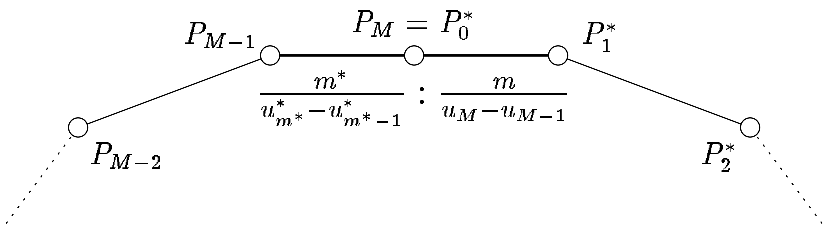

P_{M-1}
$P_M = P_0^*$
P_1^*
$\frac{m^*}{u_{m^*}^* - u_{m^*-1}^*} : \frac{m}{u_M - u_{M-1}}$
P_{M-2}
P_2^*

Figure 2.1: Joining Two Curves with $\mathcal{C}^1$ Continuity

2.1.3 Evaluation of B-spline Curve Position and Derivatives

One popular method for evaluating B-spline curves is called the *de Boor* method, which is based on the knot insertion method described in the previous section. A B-spline curve C is evaluated at a point u via the de Boor method by repeatedly inserting the knot u into the curve's knot vector until u has reached full multiplicity. If u were such that $u_{m+i-1} \leq u < u_{m+i}$, then only the control vertices $c(u_i, \ldots, u_{m+i-1}), \ldots, c(u_{m+i}, \ldots, u_{2m+i-1})$ are needed to produce $c(u, \ldots, u) = C(u)$.

This method requires $\sum_{j=1}^{m} j = m(m+1)/2$ affine combinations of control vertices to evaluate $C(u)$. Figure 2.2 shows an example of evaluating a cubic B-spline curve at u, where $u_2 \leq u < u_3$.

$$\begin{tikzcd}
 & & & c(u,u,u) & & & \\
 & & c(u,u,u_2) \arrow[ur, "\frac{u_3-u}{u_3-u_2}"] & & c(u,u,u_3) \arrow[ul, "\frac{u-u_2}{u_3-u_2}"'] & & \\
 & c(u,u_1,u_2) \arrow[ur, "\frac{u_3-u}{u_3-u_1}"] & & c(u,u_2,u_3) \arrow[ul, "\frac{u-u_1}{u_3-u_1}"'] \arrow[ur, "\frac{u_4-u}{u_4-u_2}"] & & c(u,u_3,u_4) \arrow[ul, "\frac{u-u_2}{u_4-u_2}"'] & \\
c(u_0,u_1,u_2) \arrow[ur, "\frac{u_3-u}{u_3-u_0}"] & & c(u_1,u_2,u_3) \arrow[ul, "\frac{u-u_0}{u_3-u_0}"'] \arrow[ur, "\frac{u_4-u}{u_4-u_1}"] & & c(u_2,u_3,u_4) \arrow[ul, "\frac{u-u_1}{u_4-u_1}"'] \arrow[ur, "\frac{u_5-u}{u_5-u_2}"] & & c(u_3,u_4,u_5) \arrow[ul, "\frac{u-u_2}{u_5-u_2}"']
\end{tikzcd}$$

Figure 2.2: Example of de Boor Curve Evaluation

The de Boor method of curve evaluation provides a way to evaluate curve derivatives. When the de Boor evaluation at u is stopped one level short of completion, two "control vertices" remain: $c(u_i, \overbrace{u,\ldots,u}^{m-1})$, and $c(u_{i+1}, \overbrace{u,\ldots,u}^{m-1})$, where i is such that $u_i \leq u < u_{i+1}$. The derivative of C at u can be calculated as $C'(u) = \frac{m}{u_{i+1}-u_i}(c(u_{i+1},u,\ldots,u) - c(u_i,u,\ldots,u))$. The method of derivative calculation that is based on the de Boor evaluation requires $m(m+1)/2 - 1$ affine combinations to produce the $c(u_{i+1},u,\ldots,u)$ and $c(u_i,u,\ldots,u)$, and the equivalent of one more combination to produce the derivative, for a total of $m(m+1)/2$ combinations.

To evaluate the position and derivative of a curve at a value u, first evaluate the derivative using the de Boor described above, but "cache" the values of $c(u_{i+1},u,\ldots,u)$ and $c(u_i,u,\ldots,u)$. Then the position $C(u)$ can be evaluated at a cost of one additional affine combination. Thus, the cost of evaluating the position and derivative of the curve C at u is $m(m+1)/2+1$, merely one affine combination more than evaluating either value alone.

2.2 Tensor Product Surfaces

A tensor product surface can be thought of as an extension to a B-spline curve that is evaluated over a two-dimensional domain. A tensor product surface $S(u,v)$ is defined by a rectangular grid of control vertices, $P_{i,j}$, and associated patch basis functions. The patch basis functions are formed by taking the product of two curve basis functions, $B(u)$ and $B(v)$. $B(u)$ and $B(v)$ are determined by two sets of knot vectors $u_0, \ldots, u_{M+m-1}$ and $v_0, \ldots, v_{N+n-1}$, as in §2.1. The patch domain is defined to be $D = [u_{m-1}, u_M] \times [v_{n-1}, v_N]$.

The patch control vertices are double indexed to reflect their rectangular arrangement. Thus the tensor product B-spline surface $S(u,v)$ can be expressed as a double summation:

$$S(u,v) = \sum_{i=0}^{M} \sum_{j=0}^{N} P_{i,j} B_i^m(u) B_j^n(v). \tag{2.5}$$

Here, m and n are the degrees of the curve basis functions in the u and v parametric directions, respectively. An alternative formulation of (2.5) is

$$S(u,v) = \sum_{i=0}^{M} \sum_{j=0}^{N} P_{i,j} B_{i,j}(u,v),$$

where $B_{i,j}(u,v) = B_i^m(u) B_j^n(v)$.

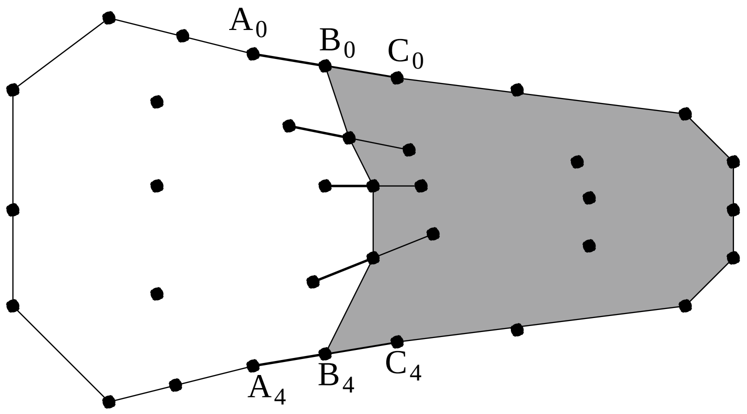


Figure 2.3: Joining Two Surfaces with $\mathcal{C}^1$ Continuity

Tensor product surface continuity arises from curve continuity. Tensor product surfaces may be joined with a desired level of continuity by constructing rows (or columns) of control vertices that would meet with that level of continuity when treated as curves. Equation (2.4) indicates that in Figure 2.3, the join between the two surfaces will be $\mathcal{C}^1$ if the boundary curves coincide and if the control vertices A_i, B_i, and C_i are collinear for each i, and

$$\frac{m}{u_M - u_{M-1}}(B_i - A_i) = \frac{m}{u^*_{m^*} - u^*_{m^*-1}}(C_i - B_i),$$

where the rightmost surface has degree m^* and knot values u^*_i in the u parametric direction.

If their underlying basis functions are formed from knot vectors whose end knots have full multiplicity, a tensor product surface's boundaries can be derived directly from the control polygon. Equation 2.5 may be rewritten as $S(u,v) = \sum_{i=0}^{M}\left(\sum_{j=0}^{N} P_{i,j}B_j^n(v)\right)B_i^m(u)$. If the end knots in the v_i knot vector have full multiplicity, then $\sum_{j=0}^{N} P_{i,j}B_j^n(v_{n-1}) = P_{i,0}$ for all i. Thus, $S(u, v_{n-1}) = \sum_{i=0}^{M}\left(\sum_{j=0}^{N} P_{i,j}B_j^n(v_{n-1})\right)B_i^m(u) = \sum_{i=0}^{M} P_{i,0}B_i^m(u)$ and the boundary of the surface corresponding to $v = v_{n-1}$ is the B-spline curve formed by the the $P_{i,0}$s and the knot vector $u_0, u_1, \ldots, u_{M+m-1}$. Similar results hold for the boundaries corresponding to $v = v_{N-n+1}$, $u = u_{m-1}$, and $u = u_{M-m+1}$.

The cross-boundary derivatives of a tensor product B-spline surface can also be computed from the surface definition. As can be seen from (2.3), the cross-boundary derivative of any point on the boundary $u = u_{m-1}$ can be found by considering the derivatives at the endpoints of the curves formed by each column of control vertices:

$$\begin{aligned}
\frac{\partial}{\partial u}S(u_{m-1}, v) &= \frac{\partial}{\partial u}\sum_{i=0}^{M}\sum_{j=0}^{N} P_{i,j}B_i^m(u_{m-1})B_j^n(v) \\
&= \sum_{j=0}^{N} B_j^n(v)\left(\sum_{i=0}^{M}\frac{\partial}{\partial u}B_i^m(u_{m-1})P_{i,j}\right) \\
&= \sum_{j=0}^{N} B_j^n(v)\frac{m(P_{1,j} - P_{0,j})}{u_m - u_{m-1}} \\
&= \frac{m}{u_m - u_{m-1}}\sum_{j=0}^{N} B_j^n(v)(P_{1,j} - P_{0,j}).
\end{aligned}$$

Evaluating a derivative at an arbitrary point on a surface is slightly more complicated. In the next section I present a method, based on de Boor curve evaluation, for performing such an evaluation.

2.3 Efficient Evaluation of Position and Derivatives

In §2.1.3 I described how the de Boor method for curve evaluation can be used to efficiently compute the position and derivative of a B-spline curve at a given domain value. In this section, I explain how the de Boor method can be extended to the efficient evaluation of position and derivatives of a tensor product B-spline surface. Similar explanations appear elsewhere and the results are well known [Car95], but I repeat them here since I use the resulting evaluation costs directly later in this thesis. Also note that the results presented in this section are summarised for easy reference in Table 3.1.

A given point, $S(u, v)$, on a tensor product surface is a convex combination of $(m+1)(n+1)$ control vertices. If u and v are such that $u_{i+m-1} \leq u < u_{i+m}$ and $v_{j+n-1} \leq v < v_{j+n}$, then the relevant control vertices are $\{P_{k,l}\}$, where $i \leq k \leq i+m$ and $j \leq l \leq j+n$. For the remainder of this section, I will make use of a notational convenience where $t^{\langle e \rangle}$ appearing in a blossom value indicates that the parameter t is repeated e times; thus $s(\overbrace{u, \ldots, u}^{m \text{ times}}; v_j, \overbrace{v, \ldots, v}^{n-1 \text{ times}})$ could be written as $s(u^{\langle m \rangle}; v_j, v^{\langle n-1 \rangle})$.

Recall that each B-spline curve control vertex P_k can also be referred to by its blossom value as $c(u_k, u_{k+1}, \ldots, u_{k+m-1})$. Similarly, each tensor product surface control vertex has a blossom value that can be obtained by concatenating one set of blossom arguments for each parametric direction:

$$P_{k,l} = s(u_k, u_{k+1}, \ldots, u_{k+m-1}; v_l, v_{l+1}, \ldots, v_{l+n-1}).$$

This construction suggests a method that may be used to evaluate a tensor product surface point: apply the de Boor method in one parametric direction, and then in the other.

To evaluate $S(u, v)$, first apply the de Boor curve evaluation method to each of the columns of contributing control vertices, as if evaluating n curves at u. Thus, for each l, the control vertices $s(u_i, \ldots, u_{i+m-1}; v_l, \ldots, v_{l+n-1}), \ldots, s(u_{i+m}, \ldots, u_{i+2m-1}; v_l, \ldots, v_{l+n-1})$ are replaced with $s(u^{\langle m \rangle}; v_l, \ldots, v_{l+n-1})$. Next de Boor curve evaluation is applied to the remaining row, as if evaluating a curve at v, to calculate $S(u, v)$. Now control vertices $s(u^{\langle m \rangle}; v_j, v_{j+1}, \ldots, v_{j+n-1})$, $\ldots$, $s(u^{\langle m \rangle}; v_{j+n}, v_{j+n+1}, \ldots, v_{j+2m-1})$ are replaced with $s(u^{\langle m \rangle}; v^{\langle n \rangle}) = S(u, v)$. Figure 2.4 demonstrates the evaluation procedure for a bicubic surface with $u_2 \leq u < u_3$ and $v_2 \leq v < v_3$.

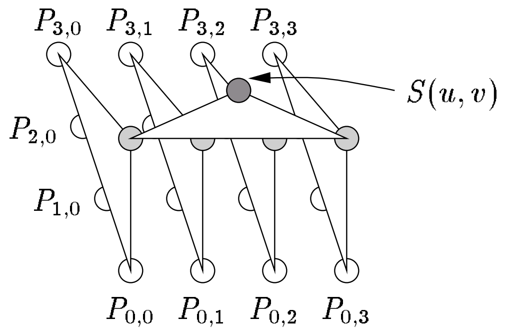


Figure 2.4: Evaluating a Surface for Position

Consider the number of affine combinations required to use this method of surface evaluation. Performing de Boor on each of $n+1$ columns of $m+1$ control vertices costs $\sum_{i=1}^{m} i = m(m+1)/2$ affine combinations, for a total of $(n+1)m(m+1)/2$. Combining the remaining row of $n+1$ control vertices costs an additional $n(n+1)/2$ affine combinations. Thus the entire evaluation requires

$$\frac{n(n+1) + (n+1)m(m+1)}{2} = \frac{mn^2 + mn + m^2 + n^2 + m + n}{2}$$

affine combinations. Note that this value is not symmetric in m and n; the number of combinations is lower if $n < m$, as compared to $m > n$. Thus, if $m > n$, the de Boor method could be applied first row-wise and then column-wise to reduce the computational cost of performing the evaluation.

Just as the de Boor curve evaluation method was adapted to evaluate curve derivatives, it can be used to find the derivatives on tensor product surfaces as well. For example, to calculate $\frac{\partial}{\partial v} S(u, v)$, reduce each column of control vertices as before, but stop one level short when applying de Boor to the remaining row of control vertices. At this point, two control vertices remain,

$s(u^{\langle m \rangle}; v_{j+n-1}, v^{\langle n-1 \rangle})$ and $s(u^{\langle m \rangle}; v_{j+n}, v^{\langle n-1 \rangle})$. The derivative is then computed as

$$\frac{\partial}{\partial v} S(u,v) = \frac{n}{v_{j+n} - v_{j+n-1}} \left(s(u^{\langle m \rangle}; v_{j+n}, v^{\langle n-1 \rangle}) - s(u^{\langle m \rangle}; v_{j+n-1}, v^{\langle n-1 \rangle}) \right).$$

The cost of computing the derivative in this manner is exactly the same as computing the position $S(u,v)$, just as in the case of curve evaluation. In addition, the position of $S(u,v)$ may be obtained with one additional affine combination:

$$S(u,v) = \frac{v_{j+n} - v}{v_{j+n} - v_{j+n-1}} s(u^{\langle m \rangle}; v_{j+n-1}, v^{\langle n-1 \rangle}) + \frac{v - v_{j+n-1}}{v_{j+n} - v_{j+n-1}} s(u^{\langle m \rangle}; v_{j+n}, v^{\langle n-1 \rangle}).$$

A similar construction may be used to efficiently obtain both $S(u,v)$ and $\frac{\partial}{\partial u} S(u,v)$.

A second-order partial derivative can be calculated (assuming the degree of the surface in the parametric direction is at least two) by stopping the second de Boor curve evaluation step *two* levels short, so three control vertices remain. Scaled pairwise differences are then taken, resulting in two "control vertices" (which are vectors). The difference of these two vectors is then taken and scaled appropriately to produce the second partial derivative. The original evaluation could then resume with the three intermediate control vertices that were produced and the first partial derivative and position could be calculated. The cost of evaluating the position, first, and second partial derivative at a point is only three affine combinations greater than evaluating the position and first partial derivative.

Attempting to evaluate the position and *both* partial derivatives of a surface at a given point is slightly more involved [MD95]. Rather than completely reducing each column of control vertices to a single point, the evaluation is stopped one step short, when two vertices remain in the column. Thus, there are two rows of control vertices which must be reduced, instead of just one. The second reduction stage is also stopped one step short, so there are four control vertices left: $s(u_{i+m-1}, u^{\langle m-1 \rangle}; v_{j+n-1}, v^{\langle n-1 \rangle})$, $s(u_{i+m}, u^{\langle m-1 \rangle}; v_{j+n-1}, v^{\langle n-1 \rangle})$, $s(u_{i+m-1}, u^{\langle m-1 \rangle}; v_{j+n}, v^{\langle n-1 \rangle})$, and $s(u_{i+m}, u^{\langle m-1 \rangle}; v_{j+n}, v^{\langle n-1 \rangle})$. Figure 2.5 shows a sample evaluation of a bicubic surface to this point, with $u_{i+m-1} \leq u < u_{i+m}$ and $v_{n-1} \leq v < v_n$.

To simplify the notation in the following discussion, let $L(s,t) = s(s, u^{\langle m-1 \rangle}; t, v^{\langle n-1 \rangle})$. Thus,

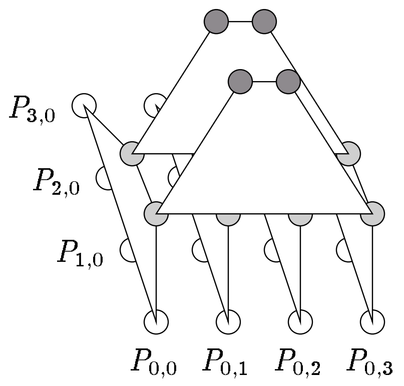


Figure 2.5: Evaluating a Surface for Position and Derivative — First Stage

the four remaining control vertices may be rewritten as $L(u_{i+m-1}, v_{j+n-1})$, $L(u_{i+m}, v_{j+n-1})$, $L(u_{i+m-1}, v_{j+n})$, and $L(u_{i+m}, v_{j+n})$.

At this point, four more affine combinations are applied to generate the control vertices $L(u_{i+m-1}, v)$, $L(u_{i+m}, v)$, $L(u, v_{j+n-1})$, and $L(u, v_{j+n})$. These newest control vertices undergo two more affine combinations that generate each of the partial derivatives of the surface at (u, v):

$$\begin{aligned}
\frac{\partial}{\partial u}S(u,v) &= \frac{m}{u_{i+m} - u_{i+m-1}}(L(u_{i+m}, v) - L(u_{i+m-1}, v)) \\
\frac{\partial}{\partial v}S(u,v) &= \frac{n}{v_{j+n} - v_{j+n-1}}(L(u, v_{j+n}) - L(u, v_{j+n-1}))
\end{aligned}$$

Finally, one last affine combination gives the position $S(u, v)$:

$$S(u,v) = \frac{u_{i+m}}{u_{i+m} - u_{i+m-1}}L(u_{i+m-1}, v) + \frac{u_{i+m}}{u_{i+m} - u_{i+m-1}}L(u_{i+m}, v).$$

An example of the second stage of the evaluation is shown in Figure 2.6.

The cost of evaluating the position and both partial derivatives at a point in this manner is considerably more expensive than evaluating only for position and one partial derivative. Partially evaluating each column of control vertices in the first stage costs $m(m+1)/2 - 1$ affine combinations, or $(n+1)(m(m+1)/2-1)$ affine combinations for all columns. Next, the two rows of control vertices must be partially evaluated, at a combined cost of $n(n+1) - 2$. In the second stage of

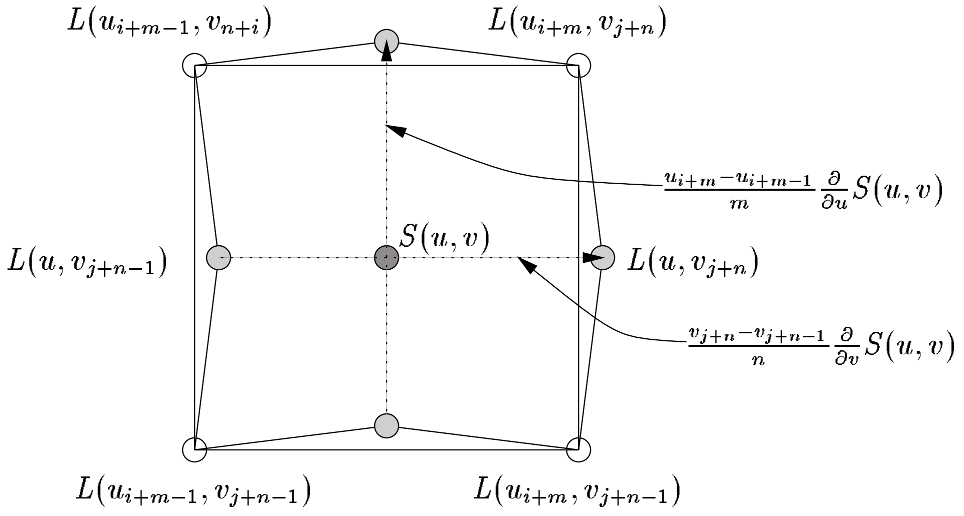

$L(u_{i+m-1}, v_{n+i})$
$L(u_{i+m}, v_{j+n})$
$\frac{u_{i+m}-u_{i+m-1}}{m}\frac{\partial}{\partial u}S(u,v)$
$S(u,v)$
$L(u, v_{j+n-1})$
$L(u, v_{j+n})$
$\frac{v_{j+n}-v_{j+n-1}}{n}\frac{\partial}{\partial v}S(u,v)$
$L(u_{i+m-1}, v_{j+n-1})$
$L(u_{i+m}, v_{j+n-1})$

Figure 2.6: Evaluating a Surface for Position and Derivative — Second Stage

evaluation, four affine combinations are used to evaluate $L(u, v_{j+n-1})$, $L(u, v_{j+n})$, $L(u_{i+m-1}, v)$, and $L(u_{i+m}, v)$. Finally two affine combinations are required to obtain the partial derivatives and one for the position, $S(u, v)$. Thus, the total number of affine combinations required to evaluate the position and both partial derivatives is

$$\begin{aligned}(n+1)(m(m+1)/2-1) \quad &+ \quad n(n+1)-2+4+2+1 \\ &= \quad \frac{nm^2+3nm+m^2+m+8}{2}.\end{aligned}$$

If the surface is bicubic, then $n = m = 3$ and the number of required combinations is 37, compared to 31 if position and only one partial derivative is calculated.

The procedure for calculating any arbitrary directional derivative at a particular point makes use of the method for calculating both partial derivatives. Suppose the directional derivative in the $\vec{t}$ direction, where $\vec{t} = \alpha\vec{u} + \beta\vec{v}$ for some α and β, were required. First the partial derivatives would be calculated, and then the directional derivative would be calculated as

$$\frac{\partial}{\partial \vec{t}}S(u,v) = \alpha\frac{\partial}{\partial u}S(u,v) + \beta\frac{\partial}{\partial v}S(u,v).$$

The cost of calculating an arbitrary directional derivative is only one more affine combination than calculating both partial derivatives, and additional directional derivatives require only one more affine combination each. Thus, calculating the position and two arbitrary directional derivatives at a point on a bicubic surface would require 39 affine combinations.

Later I will use a construction that requires the position, two directional derivatives, and a mixed-partial derivative of a surface to be calculated. Suppose $S(u,v)$, $\frac{\partial}{\partial \vec{s}}S(u,v)$, $\frac{\partial}{\partial \vec{t}}S(u,v)$, and $\frac{\partial}{\partial \vec{s}}\frac{\partial}{\partial \vec{t}}S(u,v)$, are required, where $\vec{s} = \alpha_s\vec{u} + \beta_s\vec{v}$ and $\vec{t} = \alpha_t\vec{u} + \beta_t\vec{v}$. The mixed partial derivative will be calculated as an affine combination of the partial derivatives:

$$\begin{aligned}
\frac{\partial}{\partial \vec{s}}\frac{\partial}{\partial \vec{t}}S(u,v) &= \frac{\partial}{\partial \vec{s}}\left(\alpha_t\frac{\partial}{\partial u}S(u,v) + \beta_t\frac{\partial}{\partial v}S(u,v)\right) \\
&= \alpha_s\frac{\partial}{\partial u}\left(\alpha_t\frac{\partial}{\partial u}S(u,v) + \beta_t\frac{\partial}{\partial v}S(u,v)\right) + \beta_s\frac{\partial}{\partial v}\left(\alpha_t\frac{\partial}{\partial u}S(u,v) + \beta_t\frac{\partial}{\partial v}S(u,v)\right) \\
&= \alpha_s\alpha_t\frac{\partial^2}{\partial u^2}S(u,v) + (\alpha_s\beta_t + \beta_s\alpha_t)\frac{\partial^2}{\partial u\partial v}S(u,v) + \beta_s\beta_t\frac{\partial^2}{\partial v^2}S(u,v)
\end{aligned}$$

This sum can be calculated at the cost of two affine combinations, once the two second derivatives and the mixed-partial derivative are known.

The quantities $S(u,v)$, $\frac{\partial}{\partial \vec{s}}S(u,v)$, $\frac{\partial}{\partial \vec{t}}S(u,v)$, and $\frac{\partial}{\partial \vec{s}}\frac{\partial}{\partial \vec{t}}S(u,v)$, are calculated using a procedure in which the de Boor evaluation is stopped two levels short of completion in each parametric direction, leaving a 3×3 grid of control vertices, and similar calculations are carried out as in the cases discussed earlier in this section. I will omit the details of the calculation, but 42 affine combinations are required after the 3×3 grid of control vertices is established. Thus, for a bicubic surface, the total cost would be 21 affine combinations to construct the grid plus 42, for a total of 63 affine combinations.

2.4 Adding Details to Surfaces

Many methods to add local detail to surfaces exist. In this section I briefly describe three such methods: refinement by knot insertion, hierarchical B-splines, and displacement maps. In addition to describing the techniques, I discuss the relative advantages and disadvantages of using each.

2.4.1 Adding Detail Using Knot Insertion

To add local detail to a tensor product B-spline surface, more control vertices must be added to the surface. Control vertices can be added by refining the knot vectors associated with the surface. Two popular methods for inserting knot values are Boehm's algorithm [Boe80] and the Oslo algorithm [CLR80], although it is possible to use the equation for the blossom of a curve and (2.2) to produce a knot insertion algorithm. Such an algorithm would be a generalisation of Boehm's algorithm, and would also be more efficient than the Oslo algorithm.

To add an extra knot to a surface, an entire row or column of control vertices must be added. Figure 2.7 is a schematic representation of adding three knots in each of the parametric directions to a surface that had previously had a 6×6 net of control vertices.

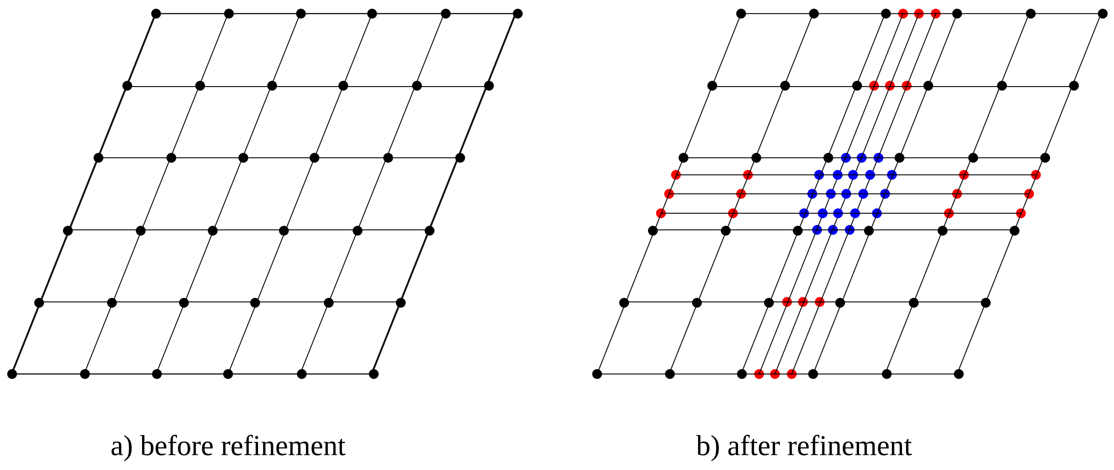


Figure 2.7: Adding Local Detail via Knot Insertion

The main advantage of adding detail to a surface by knot insertion is that it can preserve the desired continuity of the surface. Since the resulting surface is just another tensor product B-spline surface, the continuity will only be decreased by introducing repeated knots into the knot vectors.

The knot insertion method of adding detail has a number of drawbacks. The base surface must be modified, resulting in a surface with a higher complexity. In addition, many of the extra control vertices that are added may be superfluous. In the example of figure 2.7, if the surface were only modified in the centre of the patch, the 24 extra control vertices in the outer two layers are not needed.

The unneeded extra control vertices that are added by the knot insertion technique increase the storage size of the surface. Furthermore, the presence of the extra control vertices reduces the ease with which the gross shape of the surface may be adjusted. Knot insertion is not a hierarchical modelling method — it is impossible to switch easily between editing the high-level details of the surface and editing the low-level details. Once the surface has been refined, its knot structure irrevocably changes and the effect of moving a control vertex, even one somewhat removed from the area of local detail, may be altered. The increased knot density in the domain of the surface reduces the region of the surface that is changed when certain control vertices are moved.

2.4.2 Hierarchical B-splines

Hierarchical B-splines were developed by Forsey and Bartels [FB88] and are intended to add as few control vertices to a surface as possible when adding details. This scheme relies on a variant of the knot insertion method described above, but control vertices are only inserted in the region where editing is to be performed. A detail region, called an *overlay*, is created on a surface by selecting a local region of the base's domain and performing knot insertion in this region. Each control vertex in the region is then represented as an *offset* relative to a hierarchy of local reference frames. Continuity with the rest of the surface is maintained by leaving the two outermost layers of control vertices from each overlay fixed, giving $\mathcal{C}^1$ continuity. $\mathcal{C}^2$ continuity may be preserved by fixing the outer three layers of control vertices. Additional overlays may be inserted hierarchically by repeating the refinement procedure on the interior of an existing overlay.

Hierarchical B-splines allow the insertion of surface details while inserting a minimum of extra control vertices and retaining as much continuity as possible. However, hierarchical B-spline surfaces are limited in that the areas of local detail must remain parametrically aligned with the original (base) surface and have rectangular domains. In addition, it is difficult to translate the areas of local detail along the surface once they have been placed.

2.4.3 Displacement Maps

Displacement mapping is another method to add detail to a surface with as few extra control vertices as possible. Displacement mapped surfaces consist of two components, a base surface, B, and a displacement feature, D, which are combined to form a composite surface DM. B may be a simple tensor product B-spline surface or a composite surface constructed by applying a displacement map, and D is a vector-valued function defined over a compact domain D_d.

A displacement mapped surface is formed when an invertible transformation $\mathbf{T}$ is used to embed D_d into B's domain B_d. The composite surface DM is evaluated at a point (u, v) by first determining whether (u, v) lies within the image of D's domain, $\mathbf{T}(D_d)$. If not, then $DM(u, v) = B(u, v)$. Otherwise, D is evaluated at the pre-image of (u, v) under $\mathbf{T}$ to give a displacement vector $\mathbf{d} = D(\mathbf{T}^{-1}(u, v))$.

The displacement vector is added to the base surface as follows. A local coordinate frame $\mathcal{F}$ is built with origin $B(u, v)$ and frame $(\mathbf{i}, \mathbf{j}, \mathbf{k})$, where $\mathbf{i} = \frac{\partial}{\partial u'} B(u, v)$, $\mathbf{j} = \frac{\partial}{\partial v'} B(u, v)$, and $\mathbf{k} = \mathbf{i} \times \mathbf{j}$, with u' and v' being the parametric directions of D_d embedded in B_d. Then $DM(u, v)$ is taken to be the point-vector sum of $B(u, v)$ and $\mathbf{d}$ in the local coordinate frame $\mathcal{F}$.

Displacement mapping offers a number of advantages when considered as a surface modelling technique. Libraries of displacement map features may be built and later applied to arbitrary base surfaces. The features' domains may be mapped into a non-rectangular region of the base domain, and may be translated or rotated arbitrarily within the base domain. Also, displacement maps may be built so there is no loss of continuity between the "base" and "feature" portions of the composite surface. Any displacement map that produces a $\mathcal{C}^0$ surface when applied to a region of the plane will produce $\mathcal{C}^0$ composite surfaces. In practice this restriction only means that the displacement map feature must have 0 displacement along its boundary. $\mathcal{C}^1$ and higher surfaces may be constructed using similar restrictions on the displacement fields.

The drawback to using displacement maps is that they are expensive to evaluate. Every composite surface point that is evaluated will require one tensor product evaluation of the base surface, plus possibly another evaluation of the displacement map feature. If additional features are added to the composite surface in a hierarchical fashion, an extra evaluation is added for every level of the hierarchy that must be traversed to reach the base surface. In addition, since the displacement features may be oriented arbitrarily to form a composite surface and each surface point in the detail region also depends on an underlying surface, it is not possible to use more efficient tensor product grid evaluation methods for the tensor product surfaces.

Displacement mapping is a superior modelling technique in many respects. The technique is very flexible and can be used to produce composite surfaces with any desired degree of continuity. However, without restricting the points at which the displacement mapped surfaces will be evaluated, they are too expensive to use for interactive modelling — the cost of performing several surface evaluations for each one of potentially thousands of tessellation points is prohibitive.

2.5 Surface Pasting

Surface pasting is a method in which a tensor product B-spline surface, designated the *feature*, is attached to a *base* surface to add detail to a region of the base. The base surface may be either a simple tensor product surface or a compound surface formed by previous surface pasting operations. The pasting procedure does not modify the base surface, and the feature exhibits characteristics of its unpasted shape while reflecting the underlying surface.

Barghiel implemented the standard surface pasting method that is discussed in this thesis [Bar94, BBF95]. The method is based on a technique proposed by Bartels and Forsey [BF91]. In this section I explain how two concepts — Diffuse Coordinate Systems and Greville Displacement B-splines — are used in standard surface pasting, and I comment on some advantages and disadvantages of using surface pasting as a surface modelling technique.

2.5.1 The Diffuse Coordinate System

A *Diffuse Coordinate System* makes use of a *diffuse coordinate space.* In a diffuse coordinate space, each control vertex has an associated local coordinate frame. The local coordinate frame can be used to express a control vertex as an origin $O_{i,j}$ plus a displacement $\mathbf{d}_{i,j}$ in the diffuse coordinate space:

$$\begin{aligned} S(u,v) &= \sum_{i=0}^{M}\sum_{j=0}^{N} P_{i,j} B_i^m(u) B_j^n(v) \\ &= \sum_{i=0}^{M}\sum_{j=0}^{N} (O_{i,j} + \mathbf{d}_{i,j}) B_i^m(u) B_j^n(v) \end{aligned}$$

The behaviour of pasted surfaces depends on the choice of the origin and displacement vector for each control vertex. The standard surface pasting technique uses Greville Displacement to define the origin and the displacement vector [Bar94].

2.5.2 Greville Displacement B-splines

In this section, I describe the method used to obtain the offset and local coordinate frame that are used to construct each control vertex of a pasted feature. First, the domain of the surface is embedded in its range space so a local coordinate frame may be associated with each control vertex. The domain is embedded in the range space by mapping each domain point (u, v) to $(u, v, 0)$.

Each control vertex $P_{i,j}$ has an associated domain point, called a *Greville point*, $\gamma_{i,j} = (\gamma_i, \gamma_j)$, where γ_i is the i^{th} Greville abscissa in the u parametric direction, and γ_j is the j^{th} Greville abscissa in the v parametric direction. The surface point $S(\gamma_i, \gamma_j)$ is maximally influenced by $P_{i,j}$. The origin $O_{i,j}$ of $P_{i,j}$'s local coordinate frame is taken to be the *three-dimensional Greville point*, $\Gamma_{i,j}$, corresponding to $P_{i,j}$:

$$\Gamma_{i,j} = (\gamma_{i,j}, 0) = (\gamma_i, \gamma_j, 0)$$

The displacement vector $\mathbf{d}_{i,j}$ is, by definition, a vector from the origin of $P_{i,j}$'s local coordinate frame to $P_{i,j}$. Thus,

$$\mathbf{d}_{i,j} = P_{i,j} - \Gamma_{i,j},$$

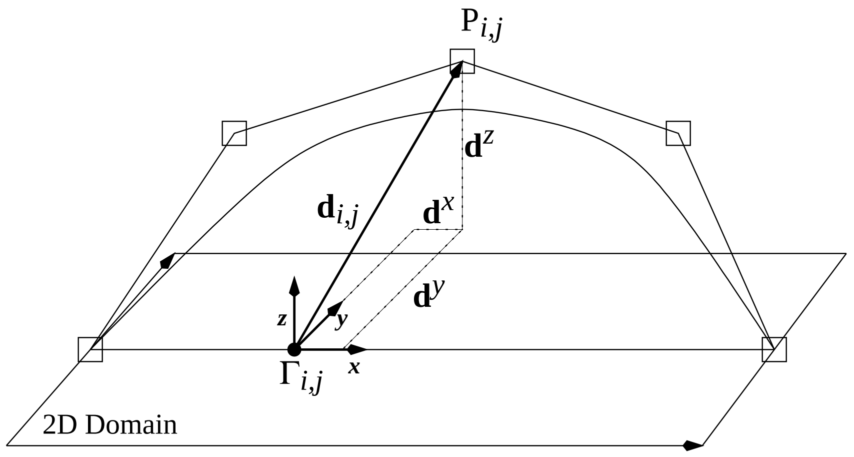


Figure 2.8: Greville Displacement B-spline

and $\mathbf{d}_{i,j}$ is called a *Greville displacement.* Using these origins and displacements, the diffuse representation of a tensor product B-spline surface is:

$$S(u,v) = \sum_{i=0}^{M}\sum_{j=0}^{N}(\Gamma_{i,j} + \mathbf{d}_{i,j})B_i^m(u)B_j^n(v).$$

Figure 2.8 illustrates the relationship between a control vertex, its associated Greville point, and its Greville displacement.

As mentioned before, the origin of $P_{i,j}$'s local coordinate frame is taken to be $\Gamma_{i,j}$. The basis for the frame is written as $(\mathbf{x}, \mathbf{y}, \mathbf{z})$, where $\mathbf{x}$ is taken to be the parametric u direction of the embedded domain, $\mathbf{y}$ is taken to be the parametric v direction, and $\mathbf{z} = (0, 0, 1)$. The Greville displacement $\mathbf{d}_{i,j}$ can be expressed relative the local frame as a linear combination of the coordinate vectors:

$$\mathbf{d}_{i,j} = d_{i,j}^x x + d_{i,j}^y y + d_{i,j}^z z$$

where $d_{i,j}^x$, $d_{i,j}^y$, and $d_{i,j}^z$ are the x, y, and z components of $\mathbf{d}_{i,j}$, respectively.

Using this construction of Greville displacements, a tensor product B-spline surface can be represented vectorially as a *Greville displacement B-spline* by removing the origins $\Gamma_{i,j}$ and expressing the control vertices solely as displacements from the corresponding Greville points:

$$S(u,v)=\sum_{i=0}^{M}\sum_{j=0}^{N}\mathbf{d}_{i,j}B_i^m(u)B_j^n(v).$$

2.5.3 Pasting Greville Displacement B-splines

At least two surfaces are involved in surface pasting: a *base surface* S_B, and a *feature surface* S_F that is pasted onto the base. Each surface is defined over its own domain. The surface pasting procedure sets the position of the pasted feature's control vertices based on the topology of the base surface and the Greville displacement B-spline representation of the feature surface.

A transformation $\mathbf{T}$ is used to map the feature domain into the base domain. $\mathbf{T}$ determines the size and placement of the feature surface relative to the base surface. Under $\mathbf{T}$, each Greville point $\gamma_{i,j}=(\gamma_i,\gamma_j)$ is mapped to $\mathbf{T}(\gamma_{i,j})=(\gamma_i',\gamma_j')$. The local coordinate frame based at $\Gamma_{i,j}$ is transformed as well. The origin of the frame is mapped to $(\gamma_i',\gamma_j',0)$, and the frame basis $(\mathbf{x},\mathbf{y},\mathbf{z})$ is mapped to $(\mathbf{x}',\mathbf{y}',\mathbf{z}')$, where $\mathbf{x}'$ and $\mathbf{y}'$ are the parametric directions of the embedded feature domain, and $\mathbf{z}'=(0,0,1)$

The pasting procedure is accomplished by creating a new local base coordinate frame $\mathcal{F}_B$ to express each of the displacement vectors $\mathbf{d}_{i,j}$. The origin of the local frame is taken to be $S_B(\gamma_i',\gamma_j')$, the point on the base surface evaluated at the transformed Greville point. The curvature of the base surface at the frame origin, $S_B(\gamma_i',\gamma_j')$, is accounted for in the construction of the frame basis, $(\mathbf{i},\mathbf{j},\mathbf{k})$, by taking the first two components to be the partial derivatives of S_B at $S_B(\gamma_i,\gamma_j)$ in the $\mathbf{x}'$ and $\mathbf{y}'$ directions. Thus, $\mathbf{i}=\frac{\partial}{\partial\mathbf{x}'}S_B(\gamma_i,\gamma_j)$, $\mathbf{j}=\frac{\partial}{\partial\mathbf{y}'}S_B(\gamma_i,\gamma_j)$, and $\mathbf{k}$ is taken to be $\mathbf{i}\times\mathbf{j}$.

The pasted control vertex $P_{i,j}'$ is constructed by embedding the feature's displacement vector $\mathbf{d}_{i,j}$ into the local base coordinate frame $\mathcal{F}_B$. Thus, $P_{i,j}'$ is calculated by performing point-vector addition in the local coordinate frame $\mathcal{F}_B$:

$$\begin{aligned}P_{i,j}' &= S_B(\gamma_i,\gamma_j)+\mathbf{d}_{i,j}\\ &= S_B(\gamma_i,\gamma_j)+d_{i,j}^x\mathbf{i}+d_{i,j}^y\mathbf{j}+d_{i,j}^z\mathbf{k}.\end{aligned}$$

2.5.4 Advantages of Surface Pasting

Surface pasting offers a number of advantages as a surface modelling technique, compared to the methods described in §2.4.

Knot insertion permanently modifies the original surface, constructing a more complex surface that can accommodate the local detail that is desired by the modeller. As a result, the surface requires more storage space and modifying the overall shape of the surface becomes more difficult. Surface pasting leaves the original surface untouched, adding a new feature surface that contains only the minimum number of extra control vertices required to add the local detail.

It is difficult to maintain a library of features that may be applied to any base when using either hierarchical B-splines or knot insertion. In addition, the rectangular areas of local detail must be parametrically aligned with the original surface and may not be translated or rotated. Using surfaced pasting, it is possible to maintain a library of feature surfaces that may be applied to any base surface and that can be rotated, translated and scaled on the base.

Finally, pasting a feature surface requires only the control vertices for the feature to be mapped, rather then every point on the composite surface as for a displacement mapped surface. Thus, in general, surface pasting is computationally less intensive than constructing a displacement mapped surface.

Surface pasting does have a number of advantages when considered as a hierarchical modelling technique, but it is not perfect. The major disadvantage of using surface pasting as a surface modelling technique is described in the following section.

2.5.5 Continuity of Pasted Surfaces

Surface pasting is an approximation technique and is not guaranteed to produce surfaces with even $\mathcal{C}^0$ continuity. One technique that has been developed to produce surfaces that look good involves setting each boundary control vertex for a feature to coincide with the Greville point corresponding to that vertex. The unpasted boundary control vertices lie in the x-y plane of the

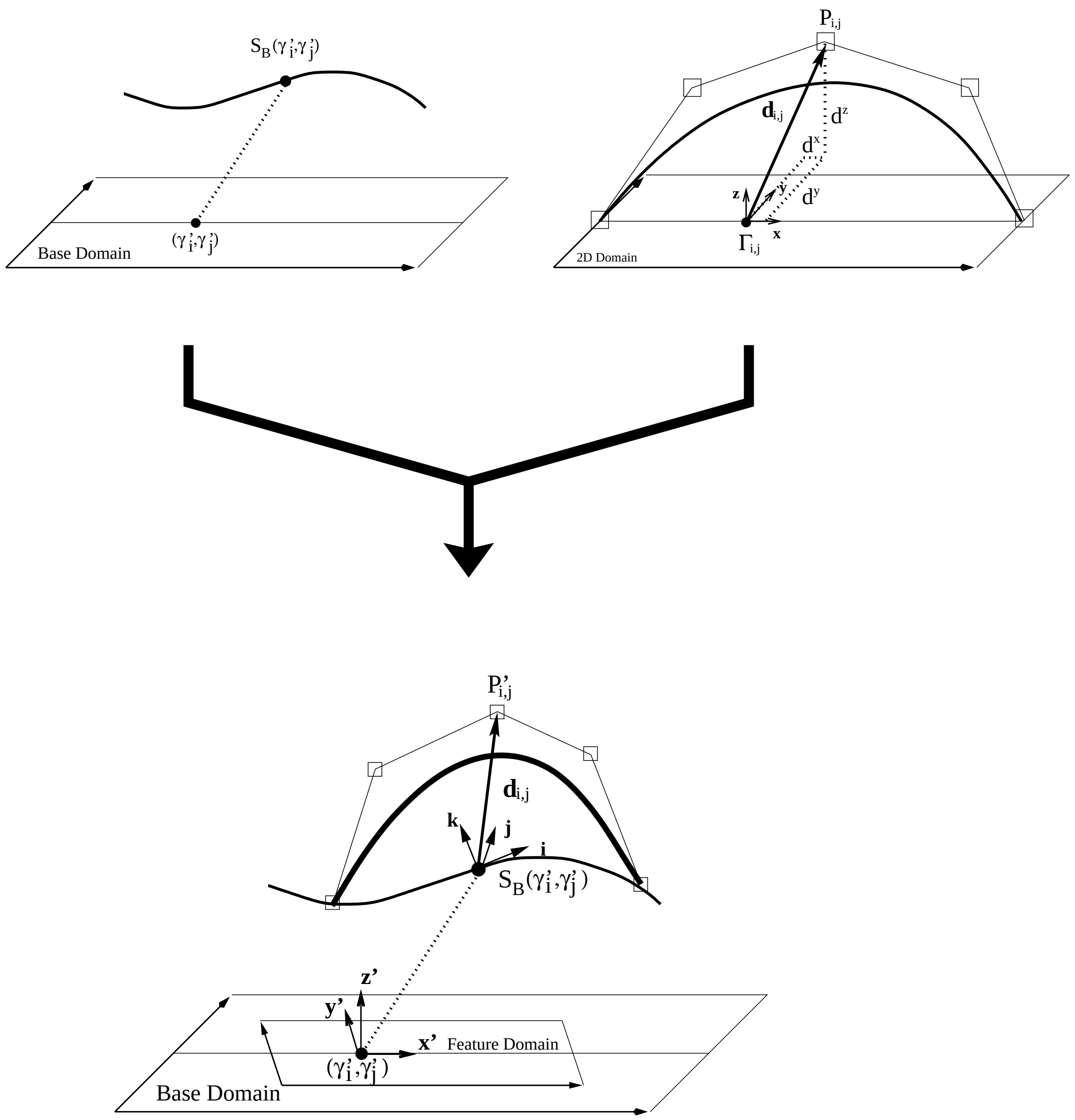


Figure 2.9: Pasting a Greville Displacement B-spline [Tsa98]

embedded feature domain. When the boundary control vertices are pasted they are positioned directly on the base surface. This construction results in a nearly $\mathcal{C}^0$ join between the base surface and the feature if the base has low curvature relative the spacing between the feature's boundary Greville points. In addition, the approximation to the base surface at the feature's boundary can be increased to any level of tolerance by refining the feature surface's knot vectors.

The procedure just described allows one to paste surfaces so they meet with nearly $\mathcal{C}^0$ continuity to the base surface. This procedure can be extended to provide an approximation to $\mathcal{C}^1$ continuity as well. The control vertices in the second layer around the boundary of the unpasted feature surface should be set to correspond to their Greville points as well. These control vertices will then be pasted so they are positioned on the base surface. The vector formed between such a control vertex and its neighbour on the boundary is the difference of two base surface points and is therefore an approximation to the partial derivative of the base surface under the boundary of the pasted feature. Thus, the feature surface will meet the base surface with an approximation to $\mathcal{C}^1$ continuity.

2.6 Divided Differences

In this section, I briefly introduce the concept of the *divided difference*, a functional that can be used to construct polynomials that interpolate certain data sets. The material in this section is intended only to discuss aspects of divided differences that are relevant to this thesis, and is derived from information presented by de Boor [dB78].

Definition 2.1 *Denote the k^{th} divided difference functional of a function f at the points $\tau_i, \ldots,$ τ_{i+k} by $[\tau_i, \ldots, \tau_{i+k}]f$. Then*

$$[\tau_i, \ldots, \tau_{i+k}]f = \begin{cases} \frac{f^{(k)}(\tau_i)}{k!}, & \text{if } \tau_i = \tau_{i+1} = \cdots \tau_{i+k} \text{ and } f \text{ is } \mathcal{C}^k \\ \frac{[\tau_{i+1}, \ldots, \tau_{i+k}]f - [\tau_i, \ldots, \tau_{i+k-1}]f}{\tau_{i+k} - \tau_i}, & \text{where } \tau_i \neq \tau_{i+k} \end{cases} \tag{2.6}$$

Note that in addition to the above properties, the divided difference functional is symmetric with respect to its arguments, and is linear in each argument.

Define a sequence of scalars $\{\tau_0, \dots, \tau_n\}$, with the value of τ_i appearing in $\{\tau_0, \dots, \tau_n\}$ at most d_i times, and let f be a function that is $\mathcal{C}^{d_i - 1}$ at τ_i. It is possible to construct a degree n polynomial p_n such that $p_n^{(j)}(\tau_i) = f^{(j)}(\tau_i)$ for each $i = 0, \dots, n$ and for all $j = 0, \dots, d_i - 1$ as follows:

$$p_n(u) = \sum_{i=0}^{n} \big([\tau_0, \dots, \tau_i] f \prod_{j=0}^{i-1} (u - \tau_j) \big). \tag{2.7}$$

The representation of p_n given above is referred to as the *Newton form* of p_n.

2.7 Quasi-Interpolation

Spline approximation techniques are methods of constructing a spline curve to approximate a given function. *Quasi-interpolation* is a spline approximation technique developed by de Boor and Fix [dBF73]. They describe a method that, given a function f defined over a region of $\mathcal{R}$ and a partition π of $\mathcal{R}$, constructs degree k spline, $F_\pi f$, that approximates f. $F_\pi f$ is called the *quasi-interpolant* of f. The quasi-interpolant is a local approximation in that its value at x depends only on the values of f in a small neighbourhood around x, it reproduces polynomials of degree k or less, and it provides a high order approximation to f, with $|F_\pi f - f|$ being $\mathcal{O}(|\pi|^{k+1})$.

2.7.1 Spline Approximation by Linear Functionals

Lyche and Schumaker have refined the quasi-interpolation operator and describe a technique for constructing a spline approximation operator, Q [LS75]. The Q operator can be applied to a real-valued function f to produce a B-spline curve, Qf, that approximates f. This section describes the construction of Qf as outlined by Lyche and Schumaker. I have taken the liberty of changing their notation to conform to that which appears earlier in this thesis and I have also expressed some concepts in terms of the blossom of a function.

The approximation Qf is generated as follows:

$$Qf = \sum_{i=0}^{M} \lambda_i f B_i^m, \tag{2.8}$$

where the $\{B_i^m\}_{i=0}^M$ are the B-splines described in §2.1 and $\{\lambda_i\}_{i=0}^M$ are linear functionals. The linear functionals are chosen so Q is applicable to a wide class of functions, Q is local, and Qf approximates smooth functions with a high order of accuracy.

The quasi-interpolant Qf approximates a smooth function f with a high order of accuracy because Q is constructed specifically to reproduce polynomials. In particular, if $\mathcal{P}_l$ is the class of polynomials of degree at most l for some $1 \leq l \leq m$, the λ_is are chosen so that

$$Qg = g \text{ for all } g \in \mathcal{P}_l. \tag{2.9}$$

Such a construction is made possible by applying the following lemma and corollary:

Lemma 2.2 *Suppose $U_\mu(x) = x^\mu$, and $u_0, \ldots, u_{M+m-1}$ are the knot values associated with the B_i^ms. If Q is as defined in (2.8), then Q satisfies (2.9) if and only if*

$$\lambda_i U_\mu = \varpi[U_\mu](u_i, \ldots, u_{i+m-1}) \tag{2.10}$$

Corollary 2.3 *Suppose that $\mathcal{F}$ is a linear space of real-valued functions on $[u_{m-1}, u_M]$ and for each $i = 0, \ldots, M$, $\{\lambda_{i,j}\}_{j=0}^l$ is a set of linear functionals over $\mathcal{F}$ such that*

$$\det(\lambda_{i,j} U_\mu)_{j,\mu=0}^l \neq 0. \tag{2.11}$$

Let $\{\alpha_{i,j}\}_{j=0}^l$ be the solution of

$$\sum_{j=0}^{l} \alpha_{i,j} \lambda_{i,j} U_\mu = \varpi[U_\mu](u_i, \ldots, u_{i+m-1}), \qquad \mu = 0, 1, \ldots, l. \tag{2.12}$$

Then for each i, $\lambda_i = \sum_{j=0}^{l} \alpha_{i,j}\lambda_{i,j}$ satisfies (2.10), and Q satisfies (2.9).

The $\alpha_{i,j}$s in Corollary 2.3 may be obtained by solving a system of linear equations. A slightly more direct method, based on extracting the $\alpha_{i,j}$s directly from a set of degree l or lower polynomials is given in the following theorem, which is due to Lyche and Schumaker.

Theorem 2.4 *For $i = 0, \ldots, M$ let $\{\lambda_{i,j}\}_{j=0}^{l}$ satisfy (2.11), and suppose $\{p_{i,j}\}_{j=0}^{l}$ are polynomials of degree at most l such that*

$$\lambda_{i,\mu} p_{i,j} = \delta_{j,\mu}, \qquad j, \mu = 0, 1, \ldots, l. \tag{2.13}$$

Then if $p_{i,j}(x) = \sum_{\nu=0}^{q_{i,j}} a_{i,j,\nu} x^{\nu}$ with $0 \le q_{i,j} \le l$, the solution of (2.12) is given by

$$\begin{aligned} \alpha_{i,j} &= \sum_{\nu=0}^{q_{i,j}} a_{i,j,\nu} \varpi[U_\nu](u_i, \ldots, u_{i+m-1}) && (2.14) \\ &= \varpi[p_{i,j}](u_i, \ldots, u_{i+m-1}) && (2.15) \end{aligned}$$

The Lyche-Schumaker quasi-interpolant approximates smooth functions with a very high order of accuracy, up to $\mathcal{O}(h^{m+1})$ for a function with a high degree of continuity. In addition, quasi-interpolants can be constructed with linear functionals that can be quickly evaluated, resulting in a fast approximation operator. The combination of high accuracy and speed make quasi-interpolation a fitting tool to improve the approximate continuity around the boundary of pasted features.

Note that quasi-interpolation operators can be constructed to reproduce spline spaces. Specifically, if l is a positive integer and π is a knot vector with no knots of multiplicity greater than l, then let $\mathcal{S}_{\pi,l}$ be the class of B-splines with knot vector π, and degree no greater than l. Then there is a choice of linear functionals and coefficients that leads to the construction of a quasi-interpolation operator with knot vector π that reproduces $\mathcal{S}_{\pi,l}$. I will not make use of such a quasi-interpolation operator in this thesis, since the linear functionals required are too costly to use in an interactive surface pasting editor.

2.8 Error Bounds for Quasi-Interpolants

I now discuss how well quasi-interpolants reproduce smooth functions. In particular, I will discuss methods for approximating

$$E_{r,s}(t) = \begin{cases} D^r(f - Qf)(t), & 0 \le r < s \\ D^r Qf(t), & s \le r \le m \end{cases}$$

where s is an integer with $1 \le s \le m+1$, and $t \in [u_p, u_{p+1}]$ for some $p \in \{m-1, \ldots, M-1\}$. The operator D^r takes the r^{th} derivative, i.e., $D^r f(t) = f^{(r)}(t)$. The parameter s was introduced because Qf sometimes has more derivatives than f. The following error analysis depends on f being $\mathcal{C}^{s-1}$ on some interval I around t. The exact choice of I depends on the linear functionals used in the construction of Q and will be discussed in more detail in §3.5. The following results are due to Lyche and Schumaker and apply to a wide class of quasi-interpolation operators [LS75].

Lemma 2.5 *Let Q be a quasi-interpolation operator such that $Qf = \sum_{i=0}^{M} \lambda_i f B_i^m$, where the B_i^ms are degree m B-splines over knot vector $u_0, \ldots, u_{M+m-1}$. Suppose Q is defined on a class of functions including polynomials of degree l or lower, where $l \le m$, and that Q reproduces polynomials of degree l or lower. Then if f is such that $D^r f(t)$ exists, for $0 \le r < s \le l+1$, and g is any polynomial of degree s or less,*

$$E_{r,s}(t) = \begin{cases} D^r R(t) - D^r QR(t), & 0 \le r < s \\ D^r QR(t), & s \le r \le m, \end{cases}$$

where $R(u) = f(u) - g(u)$.

Thus, $|E_{r,s}(t)|$ can be estimated by finding estimates for $|D^r R(t)|$ and $|D^r QR(t)|$. If f is $\mathcal{C}^{s-1}$ over I, then g can be chosen to be the Taylor expansion of f at t, truncated to s terms. Thus,

$$R(u) = f(u) - \sum_{i=0}^{s-1} \frac{D^i f(t)}{i!}(u-t)^i, \tag{2.16}$$

and $D^r R(t) = 0$ for $0 \leq r < s$, so the problem of estimating $|E_{r,s}(t)|$ is further reduced to finding bounds for $|D^r QR(t)|$. If $\lambda_i = \sum_{j=0}^{l} \alpha_{i,j}\lambda_{i,j}$, then

$$|D^r QR(t)| \leq \sum_{i=p-m+1}^{p+1} |\lambda_i R||D^r B_i^m(t)|,$$

with

$$|\lambda_i R| \leq \sum_{j=0}^{l} |\alpha_{i,j}||\lambda_{i,j} R|.$$

The bounds on $|D^r B_i^m(t)|$ are given in (2.1), so bounds need to be found only for the quantities $|\alpha_{i,j}|$ and $|\lambda_{i,j}R|$. The range of $|\lambda_{i,j}R|$ depends on the linear functionals, but Lyche and Schumaker have derived bounds on $|\alpha_{i,j}|$ for at least one common case.

Lemma 2.6 *Suppose Q is as in Lemma 2.5, with $\lambda_i = \sum_{j=0}^{l} \alpha_{i,j}\lambda_{i,j}$. Let $p_{i,j}(u) = c_{i,j}(u - z_{i,j,1}) \cdots (u - z_{i,j,q_{i,j}})$, where $q_{i,j} \leq l$ for each i and j. If $\alpha_{i,j} = \varpi[p_{i,j}](u_i, \ldots, u_{i+m-1})$, and $t \in [u_p, u_{p+1}]$, with R as defined in (2.16) then $|\alpha_{i,j}|$ is such that*

$$|D^r QR(t)| \leq (m+1)\Upsilon_{m,r} \max_{p-m+1 \leq i \leq p+1} \sum_{j=0}^{l} |\lambda_{i,j}R||c_{i,j}||A_{i,j}|, \tag{2.17}$$

with

$$A_{i,j} = \max \frac{|u_{\nu_0} - z_{i,j,0}| \cdots |u_{\nu_{q_{i,j}}} - z_{i,j,q_{i,j}}|}{\Delta_{i,p,m} \cdots \Delta_{i,p,m-r+1}} \tag{2.18}$$

where the maximum is taken over all choices of distinct $\nu_0, \ldots, \nu_{q_{i,j}}$, with $i \leq \nu_k \leq i+m-1$ for all k, and $\Upsilon_{m,r}$ and $\Delta_{i,p,k}$ are as defined in (2.1).

I introduce a class of quasi-interpolation operators, Q^d, in §3.3. The general error bound results presented here will be used to develop error bounds for the Q^d operators in §3.5.

Chapter 3

Quasi-Interpolated Surface Pasting

3.1 Motivation and Goals

One of the main disadvantages to using surface pasting as a modelling technique is that the boundary of a pasted feature is not guaranteed to meet the base with any degree of continuity. In practice, the feature/base boundary can be marred by obvious gaps if the base surface has high curvature or if the pasted feature has a coarse knot structure. The traditional method of reducing gaps between the base and feature is to refine the feature. The boundary control vertices of a refined feature are more closely spaced, so the feature's boundary is a better approximation of the base surface.

Although refining the feature results in a more appealing composite surface, refinement should be avoided if possible. Since the refining technique is the same as adding detail to a surface with knot insertion, it introduces the same disadvantages. Extra control vertices are inserted over the entire surface of the feature rather than just at the boundary where they are needed, and the refined feature is more complex than the original one, requiring extra storage and effort to evaluate. Furthermore, each of the feature's control vertices must be placed relative a local coordinate frame on the the base surface, so additional control vertices require extra surface evaluations to paste the feature.

In this thesis, I propose an alternative surface pasting technique that results in improved continuity between the pasted feature and underlying base without resorting to refining the feature surface. Given a feature surface to paste on a base, the new method produces a composite surface with considerably smaller discontinuities around the feature boundary. The new method does not require any changes to the knot structure of the feature and the average cost of placing each control vertex is not significantly higher than that of the original surface pasting technique.

The new surface pasting technique does not use a single method for placing each control vertex. The control vertices are logically divided into two groups — the boundary control vertices and the interior control vertices. The interior control vertices are placed as in the standard surface pasting method. The boundary control vertices consist of the outermost one or two layers of control vertices and are placed using a spline approximation technique. I have chosen Lyche and Schumaker's quasi-interpolation technique since it provides a high order of approximation and can be implemented so as to have a relatively low computational cost, as will be shown later.

3.2 General Features of Quasi-Interpolated Surface Pasting

I will present two surface pasting techniques that use quasi-interpolation to improve the approximate continuity around pasted feature boundaries. The first technique is intended to improve the approximation of the boundary position, and the second is intended to improve the approximation of both the boundary position and cross-boundary derivative.

Each feature boundary is constructed as the approximation to a curve on the base surface. Each approximation is calculated using a Lyche-Schumaker quasi-interpolation operator. Since surface pasting is used as an interactive modelling technique, the quasi-interpolation operators are designed to require relatively little computation to evaluate.

Each surface in a composite pasted surface is a tensor product B-spline surface, so it is possible to obtain the position and (between knot lines on the base) arbitrarily many derivatives of the base surface. The position operator and all the derivative operators may be used as linear functionals to construct a quasi-interpolant; however the cost of using each of these operators must be considered.

Each boundary of a feature surface is constructed as if it were a curve approximation problem. The curve to be approximated is obtained by evaluating the base surface along the curve formed by the image of the feature's domain boundary after it has been embedded in the base domain. That is, the feature boundary corresponding to parameter value $u = u_{m-1}$ is an approximation of S_B evaluated along the domain curve $\mathbf{T}(u_{m-1}, (1-t)v_{n-1} + tv_N)$, where $t \in [0, 1]$, and similarly for the other three feature boundaries.

In the following section I develop a class of quasi-interpolants that interpolate one or more derivatives at the endpoints of the approximated curve. These quasi-interpolants are used in the modified surface pasting procedure.

3.3 Quasi-Interpolants that Reproduce Derivatives

I now introduce a class of quasi-interpolation operators that use point and derivative samples to approximate curves. The quasi-interpolants are designed so the endpoints of the approximations reproduce the position and zero or more derivatives of the original curve. These operators are special cases of the Q operator that was developed by Lyche and Schumaker [LS75].

Let l, m and M be positive integers with $l \leq m < M$, and let $u_0, \ldots, u_{M+m-1}$ be a nondecreasing sequence of knots with $u_i < u_{i+m}$. Lyche and Schumaker introduce a quasi-interpolant Q, with $Qf = \sum_{i=0}^{M} \lambda_i f B_i^m$, where $\lambda_i = \sum_{j=0}^{l} \alpha_{i,j}\lambda_{i,j}$ and the $\alpha_{i,j}$s and $\lambda_{i,j}$s are as given below.

For $i = 0, 1, \ldots, M$ and $j = 0, 1, \ldots, l$, define

$$\lambda_{i,j} f = [\tau_{i,0}, \ldots, \tau_{i,j}]f \tag{3.1}$$

where for each i, $\tau_{i,0}, \ldots, \tau_{i,l}$ is a sequence of nondecreasing scalar values, with $\tau_{i,j} \in [u_{m-1}, u_M]$.

Lyche and Schumaker also defined polynomials $p_{i,j}$ with the property that for each i, $\lambda_{i,\mu} p_{i,j} =$

$\delta_{\mu,j}$ for $j, \mu = 0, 1, \ldots, l$:

$$p_{i,j}(u) = \prod_{k=0}^{j-1} (u - \tau_{i,k}). \tag{3.2}$$

Finally, a set of coefficients, $\{\alpha_{i,j}\}$, is constructed from the $p_{i,j}$s as follows:

$$\alpha_{i,j} = \varpi[p_{i,j}](u_i, \ldots, u_{i+m-1}) \tag{3.3}$$

The quasi-interpolant Q reproduces polynomials of degree l or less exactly, but is not guaranteed to reproduce the position and derivatives of the base curve at its endpoints. For a fixed integer d, with $0 \leq d \leq m$, and $d < M/2$, I define a quasi-interpolation operator Q^d that reproduces polynomials of degree m or less. In addition, if f is $\mathcal{C}^d$ at u_{m-1} and u_M, then $D^k Q^d f(u_{m-1}) = D^k f(u_{m-1})$ and $D^k Q^d f(u_M) = D^k f(u_M)$ for $k \leq d$.

For each d, the operator Q^d is an instance of Q, with $l = m$ and the values of $\tau_{i,j}$ restricted as follows:

$$\tau_{i,j} = \begin{cases} u_{m-1} & \text{if } j \leq i \leq d, \\ u_M & \text{if } i \geq M - d, \text{ and } j \leq M - i \end{cases} \tag{3.4}$$

and $\tau_{i,j} \neq \tau_{i,k}$ except where specified above. Note that $Q^d f$ can be constructed solely from position samples of f, except at u_{m-1} and u_M, where position and the first d derivatives of f are sampled via the divided difference operators.

Theorem 3.1 *If m, M, d, $u_0, \ldots, u_{M+m-1}$, $\{\lambda_i\}_{i=0}^{M}$, and $\{\tau_{i,j}\}_{i=0,j=0}^{M,m}$ are defined as in (3.1), $\{B_i^m\}_{i=0}^{M}$ are the B-spline functions of degree m over knot vector $u_0, \ldots, u_{M+m-1}$, and Q^d is an operator with*

$$Q^d f = \sum_{i=0}^{M} \lambda_i f B_i^m, \tag{3.5}$$

then Q^d reproduces polynomials of degree m or less and has following properties:

$$\begin{aligned} D^k Q^d f(u_{m-1}) &= D^k f(u_{m-1}), \text{ if } k \le d \\ D^k Q^d f(u_M) &= D^k f(u_M), \text{ if } k \le d. \end{aligned}$$

Proof: Q^d reproduces polynomials of degree m or less, as it is merely a restriction of the Q operator defined by Lyche and Schumaker [LS75].

It remains to show that $Q^d f$ shares position and d derivatives with f at its endpoints. I will use the Taylor series representation of a function to show this result. In particular, let $T^l_{\bar{u}} f$ be the Taylor series representation of f expanded around $\bar{u}$, truncated to $l+1$ terms:

$$T^l_{\bar{u}} f(u) = \sum_{j=0}^{l} \frac{(u-\bar{u})^j}{j!} D^j f(\bar{u}).$$

By construction, $D^j T^l_{\bar{u}} f(\bar{u}) = D^j f(\bar{u})$, for $j \le l$.

The B-spline curve $Q^d f$ was constructed so the i^{th} control vertex is

$$P_i = \lambda_i f = \sum_{j=0}^{m} \varpi[p_{i,j}](u_i, \ldots, u_{i+m-1}) \lambda_{i,j} f. \tag{3.6}$$

Consider the terms in (3.6) where $j > i$ and $i \le d$:

$$\begin{aligned} &\varpi[p_{i,j}](u_i, \ldots, u_{i+m-1}) \lambda_{i,j} f \\ &\quad = \lambda_{i,j} f \varpi[(\cdot - \tau_{i,0}) \cdots (\cdot - \tau_{i,j-1})](u_i, \ldots, u_{i+m-1}) \\ &\quad = \lambda_{i,j} f \varpi[(\cdot - u_{m-1})^i \overbrace{(\cdot - \tau_{i+1}) \cdots (\cdot - \tau_{i,j-1})}^{j-i-1 \text{ factors}}](\overbrace{u_{m-1}, \ldots, u_{m-1}}^{m-i \text{ parameters}}, u_m, \ldots, u_{i+m-1}) \\ &\quad = \lambda_{i,j} f \cdot 0 = 0. \end{aligned}$$

This result holds since $\varpi[p_{i,j}](u_i, \ldots, u_{m+i-1}) = 0$ whenever $j > i$ because $j \le m$ for each j, and so $j - i - 1 < m - i$. Accordingly, at least one of the $(\cdot - u_{m-1})$ factors in each term of $\varpi[p_{i,j}]$ is evaluated at u_{m-1}, so $\varpi[p_{i,j}] = 0$. Thus, for $i \leq d$,

$$\begin{aligned}
P_i = \lambda_i f &= \sum_{j=0}^{m} \varpi[p_{i,j}](u_i, \ldots, u_{i+m-1})\lambda_{i,j} f \\
&= \sum_{j=0}^{i} \varpi[p_{i,j}](u_i, \ldots, u_{i+m-1})\lambda_{i,j} f \\
&= \sum_{j=0}^{i} \varpi[(\cdot - \tau_{i,0}) \cdots (\cdot - \tau_{i,i-1})](u_i, \ldots, u_{i+m-1})[\tau_{i,0}, \ldots, \tau_{i,i}] f \\
&= \sum_{j=0}^{i} \varpi[(\cdot - u_{m-1})^j](u_i, \ldots, u_{i+m-1})[u_{m-1}, \ldots, u_{m-1}] f \\
&= \varpi\Big[\sum_{j=0}^{i} [u_{m-1}, \ldots, u_{m-1}] f (\cdot - u_{m-1})^j\Big](u_i, \ldots, u_{i+m-1}) \\
&= \varpi\Big[\sum_{j=0}^{i} \frac{D^j f(u_{m-1})}{j!}(\cdot - u_{m-1})^j\Big](u_i, \ldots, u_{i+m-1}) \\
&= \varpi\big[T^i_{u_{m-1}}\big](u_i, \ldots, u_{i+m-1})
\end{aligned}$$

And so, the first $d+1$ control vertices of $Q^d f$ correspond exactly to the first $d+1$ control vertices of $T^i_{u_{m-1}} f$ expressed as a B-spline curve with knot vector $u_0, u_1, \ldots, u_{M+m-1}$. Therefore $D^j Q^d f(u_{m-1}) = D^j f(u_{m-1})$ for $j = 0, \ldots, d$, as was discussed in §2.1.2. A similar analysis reveals that $D^j Q^d f(u_M) = D^j f(u_M)$ for $j = 0, \ldots d$. Thus, $Q^d f$ shares position and d derivatives with f at its endpoints. □

3.3.1 Another Representation of Q^d

The description of the Q^d operators given in the previous section has several advantages. The given representation clearly shows the relationship between Q^d and the Q operator and, as a result, lends itself well to error bounds analysis. There is one major drawback to the given scheme, however.

As defined, each of the Q^d operators is based on linear functionals that make use of divided differences and on rather simple polynomials, $p_{i,j}$, that are used to compute the $\alpha_{i,j}s$, the coefficients that weight the linear samples. If a given Q^d operator, that is one with fixed knot structure and $\tau_{i,j}$s, is used to construct approximations to different curves, some effort would be wasted evaluating the divided differences. A more efficient technique would use simpler linear functionals, possibly at the expense of more complicated $p_{i,j}$s. Since the approximating splines have the same knot structure and $\tau_{i,j}$s, the coefficients need only be computed once. Thus, the increased complexity in the $p_{i,j}$s, and hence in the calculation of the $\alpha_{i,j}$s, would be amortised over many approximations. The linear functionals, which must be applied for every approximation, would be less expensive to evaluate, so the overall cost of forming the quasi-interpolant would be reduced.

I designed the Q^ds to construct the boundary of pasted features. The underlying curves that are approximated using the Q^d operators are expected to change frequently as the feature surface is translated and rotated on its base. It is relatively uncommon for the knot structure of a feature surface to change frequently. If the feature's knots change infrequently, then the knot structures for the B-spline approximations created with the Q^d operators will also change infrequently.

In §3.4, I propose a method for basing the $\tau_{i,j}$s on the knot structure of the B-spline approximations. In such a scheme, an infrequently changing knot structure means that the coefficients weighting the linear functionals (the $\alpha_{i,j}$s) in Q^d also change infrequently. Thus, it would be preferable to have Q^d operators that used inexpensive linear functionals, even if the $p_{i,j}$s would be more expensive to evaluate.

In this section, I present an alternative formulation of the Q^d operators. This new representation describes a set of operators that provides identical results to the one described earlier, but makes use of simplified linear functionals. I will proceed by defining a second class of quasi-interpolants and showing that they are identical to the Q^ds defined in the previous section. Let $\underline{Q^d} f = \sum_{i=0}^{M} \underline{\lambda_i} B_i^m$, where for $i = 0, \ldots, M$,

$$\underline{\lambda_i} = \sum_{j=0}^{m} \underline{\alpha_{i,j}\lambda_{i,j}}, \tag{3.7}$$

with the $\underline{\alpha_{i,j}}$s and $\underline{\lambda_{i,j}}$s defined as follows.

For $0 \leq i \leq M$, and $0 \leq j \leq m$, define

$$\underline{\lambda_{i,j}} f = \begin{cases} D^0 f(\tau_{i,j}) & \text{if } d < i < M-d \\ D^j f(u_{m-1}) & \text{if } i \leq d \text{ and } j \leq i \\ D^j f(u_M) & \text{if } i \geq M-d \text{ and } j \leq M-i \end{cases} \tag{3.8}$$

$$\underline{p_{i,j}}(u) = \begin{cases} \prod_{k=0,k\neq j}^{m} \frac{u-\tau_{i,k}}{\tau_{i,j}-\tau_{i,k}} & \text{if } d < i < M-d \\ \frac{(u-u_{m-1})^j}{j!} & \text{if } i \leq d \text{ and } j \leq i \\ \frac{(u-u_M)^j}{j!} & \text{if } i \geq M-d \text{ and } j \leq M-i \end{cases} \tag{3.9}$$

$$\underline{\alpha_{i,j}} = \begin{cases} 0 & \text{if } i \leq d \text{ and } j > i \\ 0 & \text{if } i \geq M-d \text{ and } j > M-i \\ \varpi[\underline{p_{i,j}}](u_i, \ldots, u_{i+m-1}) & \text{otherwise} \end{cases} \tag{3.10}$$

where the $\tau_{i,j}$s are as defined in the previous section. Note that for $d < i < M-d$, the $\underline{p_{i,j}}$s are merely Lagrange polynomials.

Note that I have not defined $p_{i,j}$s for $i \leq d$ and $j > i$ or for $i \geq M$ and $j > M-i$. The polynomials have been omitted for two reasons: 1) these $p_{i,j}$s, with $\lambda_{i,\mu} p_{i,j} = \delta_{i,j}$ are quite complex, and 2) as I will show in Theorem 3.2 below, these $p_{i,j}$s are not needed in the construction of Q^d; it is sufficient to set $\alpha_{i,j} = 0$ for the appropriate values of i and j, as I have done in (3.10).

Theorem 3.2 *Suppose λ_i is as defined in (3.1), and $\underline{\lambda_i}$ is as (3.7). If f is defined for each $\tau_{i,j}$ and is d-times differentiable at u_{m-1} and u_M, then $\lambda_i f = \underline{\lambda_i} f$ for $i = 0, \ldots, M$, and thus $Q^d f = \underline{Q^d} f$.*

Proof: If $d < i < M-d$, then

$$\lambda_i f = \varpi\Big[\sum_{j=0}^{m} [\tau_{i,0}, \ldots, \tau_{i,j}] f \prod_{k=0}^{j-1} (\cdot - \tau_{i,k})\Big](u_i, \ldots, u_{i+m-1}), \text{ and} \tag{3.11}$$

$$\underline{\lambda_i} f = \varpi\Big[\sum_{j=0}^{m} D^0 f(\tau_{i,j}) \prod_{k=0,k\neq j}^{m} \frac{\cdot - \tau_{i,k}}{\tau_{i,j} - \tau_{i,k}}\Big](u_i, \ldots, u_{i+m-1}) \tag{3.12}$$

However, it is known that

$$\sum_{j=0}^{m}[\tau_{i,0},\dots,\tau_{i,j}]f\prod_{k=0}^{j-1}(\cdot-\tau_{i,k})=\sum_{j=0}^{m}D^0f(\tau_{i,j})\prod_{k=0,k\neq j}^{m}\frac{\cdot-\tau_{i,k}}{\tau_{i,j}-\tau_{i,k}} \tag{3.13}$$

and so $\lambda_i f = \underline{\lambda_i} f$ [dB78].

If $i \leq d$, then

$$\begin{aligned}
\underline{\lambda_i} f &= \sum_{j=0}^{m}\underline{\alpha_{i,j}\lambda_{i,j}f}=\sum_{j=0}^{i}\underline{\alpha_{i,j}\lambda_{i,j}f}\\
&= \sum_{j=0}^{i}\varpi[\underline{p_{i,j}}](u_i,\dots,u_{i+m-1})\underline{\lambda_{i,j}f}\\
&= \varpi\Big[\sum_{j=0}^{i}\frac{(u-u_{m-1})^j}{j!}D^jf(u_{m-1})\Big](u_i,\dots,u_{i+m-1})\\
&= \varpi\big[T^i_{u_{m-1}}f\big](u_i,\dots,u_{i+m-1})\\
&= \lambda_i f
\end{aligned}$$

A similar analysis reveals that $\underline{\lambda_i} f = \lambda_i f$ for $i \geq M-d$.

Thus, $\underline{\lambda_i} f = \lambda_i f$ for all i, and so $\underline{Q^d} f = Q^d f$. □

I have shown that my $\underline{Q^d}$ operator is merely an alternative representation for Q^d that makes use of simpler linear functionals than those proposed by Lyche and Schumaker. In particular, note that the $\underline{\lambda_{i,j}}$s consist solely of position samples along the interior of the approximation, as well as position and up to d derivatives at the endpoints.

I have implemented the Q^d operators using the alternative representation, and so I will refer largely to that representation for Q^d in the remainder of this thesis. When it is necessary to refer to the representation that uses divided differences in its linear functionals, I will mention this explicitly.

3.3.2 An Alternative Approach — Sampling Derivatives

The choice to use only position samples away from the endpoints of the approximating curve was somewhat arbitrary. Another equally valid construction for Q^d could use position and derivative samples along the curve, as demonstrated in Figure 3.1. Using derivative information could reduce the number of base samples by half, while increasing the cost of each sample by only a small amount.

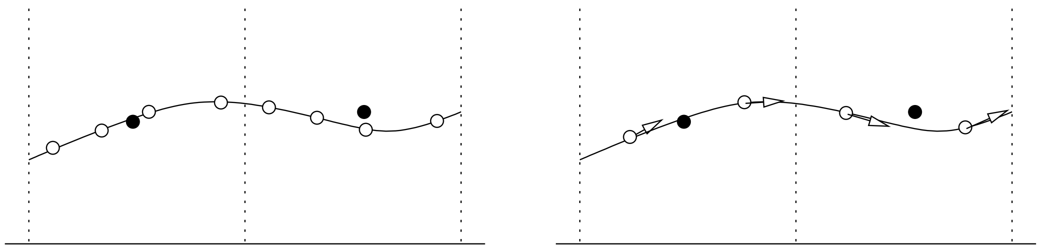


Figure 3.1: Position-Only Samples Versus Position and Derivative

If a cubic spline approximation were being constructed using the method presented in this section, four base samples would be used to place each control vertex. Sampling the base curve for both position and derivative would reduce the required number of samples to two. If the base curve to be sampled were a part of a bicubic tensor product surface, each position-only sample would cost 30 affine combinations and each sample of position and derivative would cost 38 affine combinations. Thus, sampling only for position would require 120 affine combinations per control vertex in the approximation, and sampling for position and derivative would require 76 affine combinations per control vertex.

I chose to use the position-only version of the quasi-interpolants in this thesis for a number of reasons:

1. the sampling method that I present in the next section uses sample sharing to reduce the cost of constructing the approximation without using derivatives;

2. I later develop an alternative surface pasting technique that uses position and cross-boundary derivative base samples; using derivatives along the curve would require the inclusion of a

mixed-partial derivative, which reduces the cost-effectiveness of the technique considerably;

3. the use of derivatives does not affect the order of the approximation error;

4. using derivatives as well as position significantly increases the amount of programmer effort needed to calculate coefficients used to weight the samples.

In spite of these points, using combined position and derivative samples likely would be useful in a production setting; however, I felt that the probable benefits in a research setting did not justify the extra work required to implement the technique.

Note that there are many possible methods that a quasi-interpolant could use to sample a base curve. Techniques could be devised that make use of higher-order derivatives, different distributions of position and derivative samples, integrals over sections of the curve, and so forth. A thorough discussion of the benefits of using one sampling method over another is beyond the scope of this thesis.

3.4 A Sampling Discipline

In my definition of Q^d, the values of the $\tau_{i,j}$ sample points were unspecified for a number of values of i and j. The construction of the quasi-interpolant makes the choice of $\tau_{i,j}$ unimportant when $i \leq d$ and $j > i$ or $i \geq M - d$ and $j > M - i$, but the choice of $\tau_{i,j}$ is significant for $d < i < M - d$. There are many possible methods for selecting these $\tau_{i,j}$s.

I will refer to selection of sample points as a *sampling discipline*. The quasi-interpolation operator Q^d reproduces polynomials and reproduces derivatives at the endpoints of the approximation regardless of the sampling discipline used. However, using different disciplines will result in different qualities of approximation in the interior of the curve whenever the function to be approximated is not a polynomial of degree m or lower.

The sampling discipline presented in this section was not selected to produce an optimal reproduction of the curve to be approximated. Rather, it was chosen to keep the cost of forming the quasi-interpolation low. It is assumed that the dominant cost of forming a spline approximation to a curve is the cost of sampling the original curve. One way to reduce the cost of sampling is to reduce the total number of samples used to form an approximation.

Lyche and Schumaker [LS75] do not address the issue of the cost of applying their quasi-interpolation operator. More specifically, they make no mention of reducing the number of distinct linear functionals which must be applied to the curve to be approximated. Lyche and Schumaker treat the linear functionals that are used to place each control vertex in the approximation independently. I have developed a sampling discipline that introduces dependencies between the linear functionals used to place certain of the control vertices in the approximation, with the goal of reducing the number of linear functionals that must be applied.

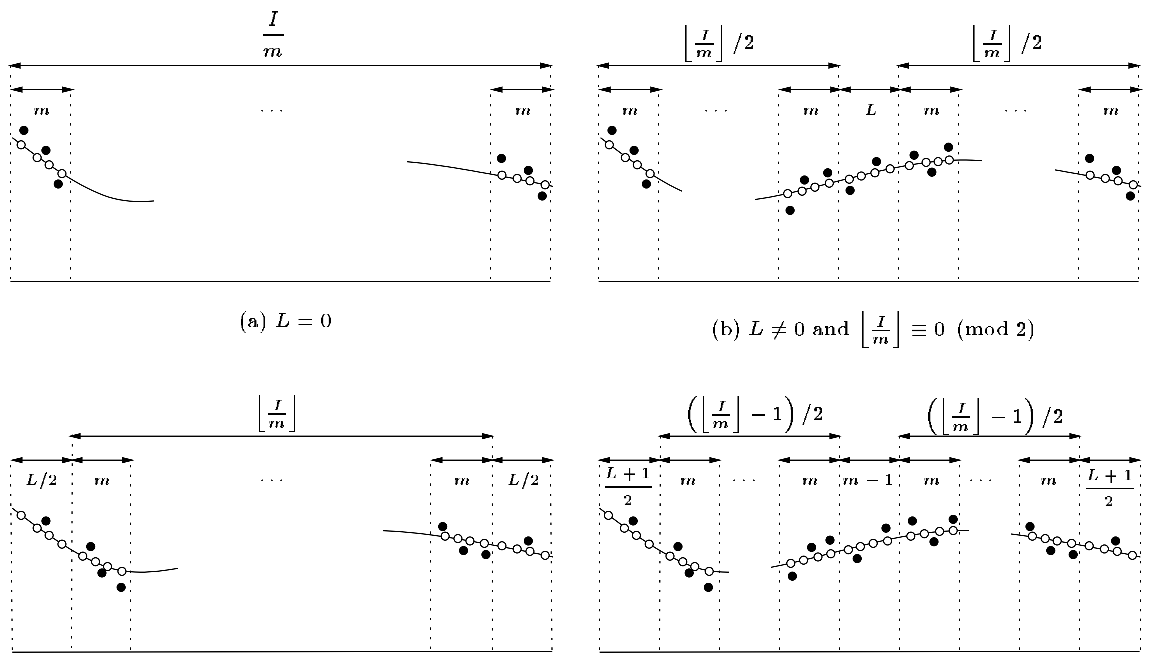
My sampling discipline reduces the number of base samples that must be made by forming the approximation's control vertices into *groups* that consist of consecutive control vertices. Within each group, every control vertex is constructed using the same set of $m + 1$ samples from the base curve; only the coefficients used to weight the individual samples differ. Of the $M + 1$ control vertices in the spline approximation, $d + 1$ from each end are explicitly set to ensure that the approximation meets the desired continuity conditions at it endpoints. The remaining $I = M - 2d - 1$ control vertices are set based on curve samples, $\tau_{i,j}$. These I control vertices are the ones that are grouped.

I have developed a grouping procedure that 1) arranges the affected control vertices into approximately I/m groups, and 2) ensures that the list of group sizes is symmetric. Pseudo-code for the grouping algorithm appears in Figure 3.2. Figure 3.3 illustrates that the grouping algorithm produces at most $\lfloor I/m \rfloor + 2$ groups of control vertices, and at most three of the groups contain fewer than m control vertices.

After the control vertices have been grouped, the values at which the base curve will be sampled must be chosen. Recall from §3.3 that $\tau_{i,j}$ is the j^{th} sample point upon which P_i's position is based, where $j = 0, \ldots, m$. Thus, for a group of control vertices $P_i, \ldots, P_{i+k}$, the required sample

```
$numFullGroups := \lfloor I/m \rfloor$
$L := I - m * numFullGroups$
if $L = 0$
    make $numFullGroups$ groups of $m$
else if $numFullGroups \equiv 0 \pmod 2$
    put leftovers in middle group; all others have $m$
else there are an odd number of full groups
    if $L \equiv 0 \pmod 2$
        first and last groups contain $L/2$ each; all others $m$
    else
        move one control vertex from middle full group to the leftovers
            the first and last groups each contain $(L+1)/2$,
            the middle group contains $m-1$, and the others contain $m$
```

Figure 3.2: Pseudocode for the Control Vertex Grouping Algorithm


$\frac{I}{m}$
m
m
$\lfloor \frac{I}{m} \rfloor /2$
$\lfloor \frac{I}{m} \rfloor /2$
m
m
L
m
m
$\lfloor \frac{I}{m} \rfloor$
$L/2$
m
m
$L/2$
$\left(\lfloor \frac{I}{m} \rfloor - 1\right)/2$
$\left(\lfloor \frac{I}{m} \rfloor - 1\right)/2$
$\frac{L+1}{2}$
m
m
$m-1$
m
m
$\frac{L+1}{2}$

(a) $L = 0$

(b) $L \neq 0$ and $\lfloor \frac{I}{m} \rfloor \equiv 0 \pmod 2$

(c) $L \neq 0$, $\lfloor \frac{I}{m} \rfloor \equiv 1 \pmod 2$, and $L \equiv 0 \pmod 2$

(d) $L \neq 0$, $\lfloor \frac{I}{m} \rfloor \equiv 1 \pmod 2$, and $L \equiv 1 \pmod 2$

Figure 3.3: Example of Control Vertex Grouping with $m = 3$. White disks are sample points, black disks are control vertices. Each column is a control vertex group labelled with the number of control vertices. The larger arrows are labelled with the number of groups.

points are

$$\begin{array}{cccc}
\tau_{i,0} & \tau_{i,1} & \cdots & \tau_{i,m} \\
\tau_{i+1,0} & \tau_{i+1,1} & \cdots & \tau_{i+1,m} \\
\vdots & \vdots & & \vdots \\
\tau_{i+k,0} & \tau_{i+k,1} & \cdots & \tau_{i+k,m}.
\end{array}$$

However, sample points are shared among all control vertices in a group, so the values in each column are equal: $\tau_{i,j} = \cdots = \tau_{i+k,j}$ for all j and the base curve need only be sampled at $\tau_{i,0}, \tau_{i,1}, \ldots, \tau_{i,m}$ for this group.

The first step in choosing sample points is to divide the domain of the approximation into intervals, one for each group of control vertices. For a group of control vertices $P_i, \ldots, P_{i+k}$, let γ_l be the Greville abscissa for control vertex P_l. I choose this group's interval to be

$$\left[\frac{\gamma_{i-1}+\gamma_i}{2}, \frac{\gamma_{i+k}+\gamma_{i+k+1}}{2}\right]. \tag{3.14}$$

All the samples (that is, the $\tau_{i,j}$s) for this group of control vertices lie in this interval.

Note that when forming Q^d, $d \geq 0$, sample points are only needed for P_l, $l = d+1, \ldots, M-d-1$. This ensures that for any control vertex group $P_i, \ldots, P_{i+k}$, we have $i \geq 1$ and $i+k \leq M-d-1$, so $i-1 \geq 0$ and $i+k+1 \leq M$. Thus, γ_{i-1} and γ_{i+k+1} exist and are contained in

$$[\gamma_0, \gamma_M] = \left[\frac{u_0 + \cdots u_{m-1}}{m}, \frac{u_M + \cdots + u_{M+m-1}}{m}\right] = [u_{m-1}, u_M],$$

since $u_0 = u_1 = \cdots = u_{m-1}$ and $u_M = u_{M+1} = \cdots = u_{M+m-1}$. This ensures that each interval lies entirely within the domain of the spline approximation.

I have chosen my sampling discipline to select the sample points by uniformly sampling the interval of the group that contains each control vertex. Thus, any for control vertex contained in the group described in the previous paragraph, the sample points will be

$$\tau_{i,j} = \frac{m-j}{m} \cdot \frac{\gamma_{i-1}+\gamma_i}{2} + \frac{j}{m} \cdot \frac{\gamma_{i+k}+\gamma_{i+k+1}}{2}, \tag{3.15}$$

for $j = 0, \ldots, m$.

This method for choosing sample points allows some sample points to be shared between groups of control vertices. For example, if there were another group of control vertices that followed the one defined above, say $P_{i+k+1}, \ldots, P_{i+k+p}$, then the sample point $(\gamma_{i+k} + \gamma_{i+k+1})/2$ would be shared between the two groups. This sharing reduces the total number of base samples needed by one less than the number of control vertex groups. Figure 3.4 is an example of how sample points are shared both within and between control vertex groups. The shaded sample points are shared between sample groups.

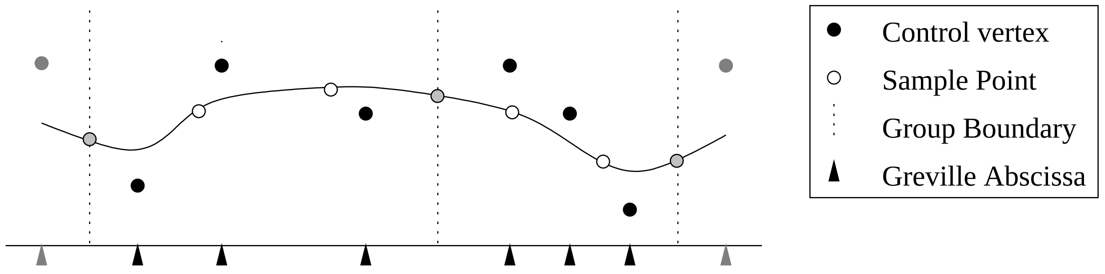


Figure 3.4: Sampling the Underlying Curve — shaded sample points are shared between groups.

3.5 Error Bounds for $Q^d f$

I now present error bounds for the Q^d quasi-interpolation operators. Specifically, I will discuss how well $Q^d f$ reproduces a smooth, real-valued function f. In this chapter I will refer to the divided difference representation of the Q^d operators as defined in Theorem 3.1 so as to easily make use of the error bound results obtained by Lyche and Schumaker [LS75]. The work in this section follows the statements made in §2.8 on error bounds for general quasi-interpolants and follows the methodology used by Lyche and Schumaker, with modifications made to accommodate the restrictions on the Q^ds.

Suppose Q^d is as in Theorem 3.1, with $\tau_{i,j}$s as defined in (3.15). Fix $u_{m-1} \leq t \leq u_M$ and p such that $u_p \leq t \leq u_{p+1}$, with $u_p < u_{p+1}$. Define $I_{i,\nu}$ to be the smallest closed interval containing $\{\tau_{i,j}\}_{j=0}^{\nu}$, and I_p to be the smallest closed interval containing $[u_p, u_{p+1}] \bigcup_{i=p-m+1}^{p+1} I_{i,m}$. The segment of the curve $Q^d f$ that corresponds to domain interval $[u_p, u_{p+1}]$ is defined by control vertices $P_{p-m+1}, \ldots, P_{p+1}$. Thus I_p contains the support of all the linear functionals used to place the control vertices that define this curve segment.

Recall that the control vertices of the approximating spline curve $Q^d f$ are arranged in groups, and that the same set of $\tau_{i,j}$s is used to place each of the control vertices in a given group. Each of the control vertex groups contains at most m control vertices, so of all the control vertices in the group containing P_{p-m+1}, the lowest possible index is $p-2m+2$. Likewise, of the control vertices in the group containing P_{p+1}, the highest possible index is $p+m$. The definition of the $\tau_{i,j}$s given in (3.15) indicates that

$$\begin{aligned} I_p &\subseteq \left([u_p, u_{p+1}] \bigcup \left[\frac{\gamma_{p-2m+1}+\gamma_{p-2m+2}}{2}, \frac{\gamma_{p+m}+\gamma_{p+m+1}}{2}\right]\right) \bigcap [u_{m-1}, u_M] \\ &= \left([u_p, u_{p+1}] \bigcup \left[\frac{\sum_{k=p-2m+1}^{p-m}(u_k+u_{k+1})}{2m}, \frac{\sum_{k=p+m}^{p+2m-1}(u_k+u_{k+1})}{2m}\right]\right) \bigcap [u_{m-1}, u_M] \\ &= \left[\frac{\sum_{k=p-2m+1}^{p-m}(u_k+u_{k+1})}{2m}, \frac{\sum_{k=p+m}^{p+2m-1}(u_k+u_{k+1})}{2m}\right] \bigcap [u_{m-1}, u_M]. \end{aligned} \tag{3.16}$$

where, for the purposes of this calculation, u_k is taken to be equal to u_{m-1} when $k < 0$, or u_M when $k > M+m-1$, and $\gamma_j = \left(\sum_{k=0}^{m-1} u_{j+k}\right)/m$.

It is the interval I_p that replaces I in the discussion of §2.8. If f is $\mathcal{C}^{s-1}(I_p)$, take R to be as defined in (2.16):

$$R(u) = f(u) - \sum_{i-0}^{s-1} \frac{D^i f(t)}{i!}(u-t)^i.$$

Then from the Taylor series for R,

$$\begin{aligned} D^j R(u) &= \frac{D^{s-1}R(\zeta)(u-t)^{s-1-j}}{(s-1-j)!} \text{ and} && (3.17)\\ D^{s-1}R(u) &= D^{s-1}f(u) - D^{s-1}f(t), && (3.18)\end{aligned}$$

for some ζ between u and t, and for $j = 0, 1, \ldots, s-1$.

In order to estimate $|\lambda_{i,j}R|$, I introduce parameters describing the spacing of the $\tau_{i,j}$s and of the u_is. Define

$$\begin{aligned}
\sigma_{i,j,\nu} &= \min_{0\le\mu\le j-\nu}(\tau_{i,\mu+\nu} - \tau_{i,\mu}), \text{ for } 1 \le \nu \le m, && (3.19)\\
\sigma_{i,\nu} &= \min_{0\le j\le l} \sigma_{i,j,\nu} && (3.20)\\
\overline{\Delta}_p &= \max_{p-2m+1\le i\le p+2m-1}(u_{i+1} - u_i) && (3.21)\\
\overline{\Delta} &= \max_{0\le i<M+m-2}(u_{i+1} - u_i) && (3.22)\\
\Delta_{p,m-r+1} &= \min_{p-m+r\le\nu\le p+1}(u_{\nu+m-r+1} - u_\nu) && (3.23)\\
\Delta_{m-r+1} &= \min_{m-1\le p\le M-1, u_p<u_{p+1}} \Delta_{p,m-r+1} && (3.24)
\end{aligned}$$

From (3.16) and (3.21) it can be seen that $|I_p| \le 3m\overline{\Delta}_p$ and

$$|t - x| \le \frac{3m+2}{2}\overline{\Delta}_p \tag{3.25}$$

for any $x \in I_p$.

The error bounds that I seek depend on the modulus of continuity of a function. If g is $\mathcal{C}^0(I)$, then the modulus of continuity of g with respect to I and Δ is defined as

$$\omega(g; \Delta; I) = \max_{x,x+h\in I, 0\le h\le\Delta} |g(x+h) - g(x)|.$$

In addition, Lemma 3.4, which bounds $|\lambda_{i,j}R|$, depends on the following lemma that is due to Lyche and Schumaker:

Lemma 3.3 *Let $0 \le \mu \le \omega - 2$. Then for any $\xi_1 \le \xi_2 \le \cdots \le \xi_\omega$ with $\eta_k = \min\limits_{1 \le \nu \le \omega - k} |\xi_{\nu+k} - \xi_\nu| > 0$, for $k = \mu+1, \ldots, \omega-1$,*

$$\big|[\xi_1, \ldots, \xi_\omega]f\big| \le \frac{\sum\limits_{\nu=0}^{\omega-\mu-1} \binom{\omega-\mu-1}{\nu} \big|[\xi_{\nu+1}, \ldots, \xi_{\nu+\mu+1}]f\big|}{\eta_{\omega-1} \cdots \eta_{\mu+1}}.$$

I now present a bound on $|\lambda_{i,j}R|$:

Lemma 3.4 *Let $0 \le d < s \le m+1$, and let $p - m \le i \le p$. Then if f is $\mathcal{C}^{s-1}(I_m)$,*

$$|\lambda_{i,j}R| \le \frac{3m+2}{2}\omega(D^{s-1}f; \overline{\Delta}_p; I_p) \begin{cases} \dfrac{|\zeta_{i,j} - t|^{s-1-j}}{j!(s-1-j)!}, & j = 0, \ldots, s-1 \\ \dfrac{2^{j-s+1}}{(s-1)!\sigma_{i,j,j} \cdots \sigma_{i,j,s}}, & j = s, \ldots, m \end{cases} \tag{3.26}$$

where $\zeta_{i,j} \in I_{i,j}$.

Proof: From (3.25), if $u \in I_p$ then

$$|D^{s-1}f(x) - D^{s-1}f(t)| \le \frac{3m+2}{2}\omega(D^{s-1}f; \overline{\Delta}_p; I_p). \tag{3.27}$$

In addition, for $j = 0, \ldots, s-1$, it can be shown that $\lambda_{i,j}R = D^j R(\zeta_{i,j})/j!$, where $\zeta_{i,j} \in I_{i,j} \subseteq I_p$. Applying equations (3.17), (3.18), and (3.27) yields the first inequality in (3.26):

$$|\lambda_{i,j}R| \le \frac{3m+2}{2}\omega(D^{s-1}f; \overline{\Delta}_p; I_p)\frac{|\zeta_{i,j} - t|^{s-1-j}}{j!(s-1-j)!}.$$

The second inequality relies on Lemma 3.3. Specifically, let $\xi_k = \tau_{i,k-1}$ and apply Lemma 3.3 with $\omega = j+1$ and $\mu = s-1$. Then

$$\begin{aligned} |\lambda_{i,j}R| &\le \frac{\sum\limits_{\nu=0}^{j-s+1} \binom{j-s+1}{\nu} \big|[\tau_{i,\nu}, \ldots, \tau_{i,\nu+s-1}]R\big|}{\sigma_{i,j,j} \cdots \sigma_{i,j,s}} \\ &\le 2^{j-s+1} \max_{0 \le \nu \le j-s} \frac{\big|[\tau_{i,\nu}, \ldots, \tau_{i,\nu+s-1}]R\big|}{\sigma_{i,j,j} \cdots \sigma_{i,j,s}}, \end{aligned} \tag{3.28}$$

since $\sum_{\nu=0}^{j-s+1}\binom{j-s+1}{\nu} = 2^{j-s+1}$.

However, $\big|[\tau_{i,\nu},\ldots,\tau_{i,\nu+s-1}]R\big| = |D^{s-1}R(\zeta_{i,j,\nu})/(s-1)!$, where $\zeta_{i,j,\nu} \in I_p$ for each $\nu = 0,\ldots,j-s+1$. Thus, combining (3.18), (3.27), and (3.28) yields the second inequality in (3.26):

$$|\lambda_{i,j}R| \le \frac{3m+2}{2}\omega(D^{s-1}f;\overline{\Delta}_p;I_p)\frac{2^{j-s+1}}{(s-1)!\sigma_{i,j,j}\cdots\sigma_{i,j,s}}. \ \Box$$

The bounds on $|\lambda_{i,j}R|$ that were calculated in Lemma 3.4 and the statements made in §2.8 provide enough information to formulate local error estimates for Q^d.

Theorem 3.5 *Let $1 \le s \le m+1$ with $m \ge 1$ and q be an integer, $1 \le q \le \infty$. Define $E^d_{r,s}$ to be the same as $E_{r,s}$ presented in §2.8, but with Q replaced by Q^d. If f is $\mathcal{C}^{s-1}$ over I_p, then for $0 \le r \le m$,*

$$||E^d_{r,s}||_{L_q[u_p,u_p+1]} \le K_p\overline{\Delta}_p^{\,s-1-r+\frac{1}{q}}\omega(D^{s-1}f;\overline{\Delta}_p;I_p) \tag{3.29}$$

where

$$K_p = \frac{(m+1)(3m+2)}{2}\Upsilon_{m,r}\frac{(\frac{5m}{2})^{s-1}}{(s-1)!}\left(\frac{\overline{\Delta}_p}{\Delta_{p,m-r+1}}\right)^r\left[2^{s-1}+\sum_{j=s}^{m}(2\rho_p)^{j-s+1}\right],$$

$$\rho_p = \max_{p-m+1\le i\le p+1}\frac{\gamma_{i+m}+\gamma_{i+m+1}-\gamma_{i-2m+1}-\gamma_{i-2m+2}}{2\sigma_{i,s}},$$

where $\Upsilon_{m,r} = \frac{m!}{(m-r)!}\binom{r}{\lfloor r/2\rfloor}$ is the constant from (2.1), and γ_k is as defined in (3.16).

Proof: I will make use of Lemmas 3.4, 2.5, and 2.6. Thus, for $u_p \leq t < u_{p+1}$,

$$\begin{aligned} |E^d_{r,s}| &\leq (m+1)\Upsilon_{m,r} \max_{p-m+1\leq i\leq p+1} \sum_{j=0}^{m} |\lambda_{i,j}R||c_{i,j}||A_{i,j}| \qquad (3.30) \\ &\leq (m+1)\Upsilon_{m,r}\frac{3m+2}{2}\omega(D^{s-1}f;\overline{\Delta}_p;I_p) \max_{p-m+1\leq i\leq p+1} \left[\sum_{j=0}^{s-1}\frac{|\zeta_{i,j}-t|^{s-1-j}}{j!(s-1-j)!}|c_{i,j}|A_{i,j} \right. \\ &\left. + \sum_{j=s}^{m}\frac{2^{j-s+1}|c_{i,j}|A_{i,j}}{(s-1)!\,\sigma_{i,j,j}\cdots\sigma_{i,j,s}}\right] \qquad (3.31) \end{aligned}$$

The $A_{i,j}$s are as in (2.18), with the following substitutions based on the $\lambda_{i,j}$s and $p_{i,j}$s used to construct the Q^d operator: $q_{i,j} = j-1$ and $z_{i,j,\nu} = \tau_{i,\nu}$ for $\nu = 0,\ldots,j-1$. Thus,

$$A_{i,j} = \max\frac{|u_{\nu_0}-\tau_{i,0}|\cdots|u_{\nu_{j-1}}-\tau_{i,j-1}|}{\Delta_{i,p,m}\cdots\Delta_{i,p,m-r+1}}$$

Furthermore, the choice of the $p_{i,j}$s dictates that $c_{i,j} = 1$ for all i and j.

In addition, note that the ρ_ps were chosen so that $|x_{\nu_\mu} - \tau_{i,\mu}| \leq \rho_p\sigma_{i,s}$. Also, $|x_{\nu_\mu} - \tau_{i,\mu}| \leq \frac{5m}{2}\overline{\Delta}_p$ and recall that $\zeta_{i,j} \in I_p$, so $|\zeta_{i,j} - t| \leq \frac{3m+2}{2}\overline{\Delta}_p \leq \frac{5m}{2}\overline{\Delta}_p$. Thus, inequality (3.31) becomes

$$\begin{aligned} |E^d_{r,s}| &\leq \frac{(3m+2)(m+1)}{2}\Upsilon_{m,r}\omega(D^{s-1}f;\overline{\Delta}_p;I_p) \\ &\cdot \max_{p-m+1\leq i\leq p+1}\left[\sum_{j=0}^{s-1}\frac{(\frac{5m}{2}\overline{\Delta}_p)^{s-1-j}}{j!(s-1-j)!}\frac{(\frac{5m}{2}\overline{\Delta}_p)^{j}}{\Delta^r_{p,m-r+1}} + \sum_{j=s}^{m}\frac{2^{j-s+1}(\rho_p\sigma_{i,s})^j}{(s-1)!(\Delta_{p,m-r+1})^r\sigma_{i,s}^{j-s+1}}\right] \\ &\leq \frac{(3m+2)(m+1)}{2}\Upsilon_{m,r}\omega(D^{s-1}f;\overline{\Delta}_p;I_p) \\ &\cdot\frac{1}{(s-1)!}\max_{p-m+1\leq i\leq p+1}\left[\frac{(\frac{5m}{2}\overline{\Delta}_p)^{s-1}}{\Delta^r_{p,m-r+1}}\sum_{j=0}^{s-1}\frac{(s-1)!}{j!(s-1-j)!} + \sum_{j=s}^{m}\frac{2^{j-s+1}\rho_p^j\sigma_{i,s}^{s-1}}{(\Delta_{p,m-r+1})^r}\right] \\ &\leq \frac{(3m+2)(m+1)}{2}\Upsilon_{m,r}\omega(D^{s-1}f;\overline{\Delta}_p;I_p)\frac{1}{(s-1)!(\Delta_{p,m-r+1})^r} \\ &\cdot\max_{p-m+1\leq i\leq p+1}\left[2^{s-1}(\frac{5m}{2}\overline{\Delta}_p)^{s-1} + \sum_{j=s}^{m}2^{j-s+1}\rho_p^j(\frac{5m}{2}\overline{\Delta}_p)^{s-1}\right] \end{aligned}$$

$$
\begin{aligned}
&\leq \frac{(3m+2)(m+1)}{2}\Upsilon_{m,r}\frac{(\frac{5m}{2}\overline{\Delta}_p)^{s-1}}{(s-1)!(\Delta_{p,m-r+1})^r} \\
&\quad \cdot[2^{s-1}+\sum_{j=s}^{m}2^{j-s+1}\rho_p^j]\omega(D^{s-1}f;\overline{\Delta}_p;I_p) \\
&\leq \frac{(3m+2)(m+1)}{2}\Upsilon_{m,r}\frac{(\frac{5m}{2})^{s-1}}{(s-1)!}\left(\frac{\overline{\Delta}_p}{\Delta_{p,m-r+1}}\right)^r[2^{s-1}+\sum_{j=s}^{m}2^{j-s+1}\rho_p^j] \\
&\quad \cdot(\overline{\Delta}_p)^{s-r-1}\omega(D^{s-1}f;\overline{\Delta}_p;I_p).
\end{aligned}
$$

Therefore,

$$|E_{r,s}^d| \leq K_p\overline{\Delta}_p^{s-1-r}\omega(D^{s-1}f;\overline{\Delta}_p;I_p). \tag{3.32}$$

Now (3.32) implies (3.29) for $q=\infty$, and integrating (3.32) over $[u_p, u_{p+1}]$ proves (3.29) for $1 \leq q < \infty$. □

The local error estimate calculated in Theorem 3.5 leads directly to the following theorem that bounds the global error estimate for $Q^d f$:

Theorem 3.6 *Assume the conditions of Theorem 3.5 hold. Furthermore, suppose f is C^{s-1} on $[u_{m-1}, u_M]$. For $0 \leq r \leq m$,*

$$||E_{r,s}^d||_{L_q[u_{m-1},u_M]} \leq K\overline{\Delta}^{s-r-1+\frac{1}{q}}\omega(D^{s-1}f;\overline{\Delta};[u_{m-1},u_M]). \tag{3.33}$$

Where

$$K = \frac{(m+1)(3m+2)}{2}\Upsilon_{m,r}\frac{(\frac{5m}{2})^{s-1}}{(s-1)!}\left(\frac{\overline{\Delta}}{\Delta_{m-r+1}}\right)^r\left[2^{s-1}+\sum_{j=s}^{m}(2\rho)^{j-s+1}\right],$$

and

$$\rho = \max_{2m-1\leq i\leq M-m-1}\frac{\gamma_{i+m}+\gamma_{i+m+1}-\gamma_{i-2m+1}-\gamma_{i-2m+2}}{2\sigma_{i,s}}.$$

Proof: Note that $\overline{\Delta}_p \leq \overline{\Delta}$ for all p, and $\Delta_{m-r+1} \leq \Delta_{p,m-r+1}$. Furthermore $\rho_p \leq \rho$, so $K_p \leq K$.

Therefore, the statement (3.33) follows from (3.29). □

3.6 Pasting Features with Approximate Continuity

In this section, I describe two methods for using the Q^d operators as part of modified surface pasting techniques. The modified pasting techniques set one or two of the outermost rings of feature control vertices using a quasi-interpolation operator. Setting the outermost layers of control vertices in this manner improves the approximate $\mathcal{C}^0$ or $\mathcal{C}^1$ continuity of the composite surface, relative a standard pasted surface.

Note that as the surface pasting technique is modified to produce surfaces with increased levels of approximate continuity, more rows of boundary control vertices will be set by the quasi-interpolation operators. Thus, care must be taken to ensure that feature surfaces that themselves are detailed do not have their details obscured when the outermost control vertices are set.

The first of the two techniques that I present in this section is intended to produce an approximate $\mathcal{C}^0$ composite surface by pasting features so their boundary position approximates the underlying base curve. The second technique pastes features whose boundary position and cross-boundary derivatives are intended to approximate the underlying curves on the base surface, thus improving the approximate $\mathcal{C}^1$ continuity of the composite surface. Extending the techniques presented in this section to form composite surfaces with better approximate $\mathcal{C}^2$ or higher continuity should be straightforward.

3.6.1 Q^0-pasting

In this section I describe how to construct composite pasted surfaces that have improved approximate $\mathcal{C}^0$ continuity around the feature boundary. The Q^0 operator, which reproduces position at the endpoints of the approximation, is used to implement the improved pasting procedure, which I call Q^0-pasting.

Each boundary of the feature surface is placed as if it were the solution of a separate curve approximation problem. Thus, four curve approximations are constructed, and each of these approximations is used to set the control vertices that contribute to one of the feature's boundary curves. Typically surface pasting is performed in a higher dimensional space, such as $\mathcal{R}^3$. Therefore, when constructing feature boundaries, the curves to be approximated will typically be higher-dimensional parametric curves, rather than real-valued functions. The procedure in this case is to form an approximation to each component of the curve to approximate in turn and to compose the approximations. Thus, a curve

$$C(u) = \begin{bmatrix} C_1(u) \\ C_2(u) \\ \vdots \\ C_n(u) \end{bmatrix} \text{ would be approximated by } QC(u) = \begin{bmatrix} QC_1(u) \\ QC_2(u) \\ \vdots \\ QC_n(u) \end{bmatrix}.$$

The Q^0 operator was designed so the control vertices at the ends of each boundary approximation are set consistently with the adjacent boundary. Appendix A.1 provides a verification that the corner control vertices of the feature surface are set consistently by each of the boundary approximations. This verification also shows that the corner control vertices are set so corners of the feature are coincident with the corresponding points on the base surface.

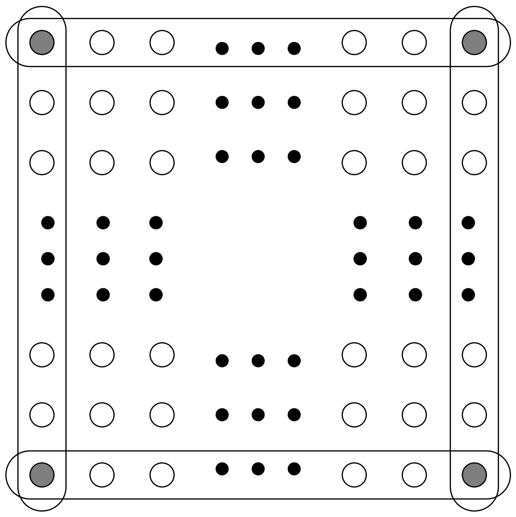


Figure 3.5: Corner Control Vertices are Part of Two Boundaries

3.6.2 Q^1-pasting

One advantage of Q^0-pasted surfaces is that the gaps between the base and the feature surface are greatly reduced. Unfortunately, the improvement in $\mathcal{C}^0$ continuity is not enough to make attractive surfaces in all cases: surfaces constructed as described in the preceding section often exhibit a pronounced discontinuity in the cross-boundary derivative. This discontinuity is often worse than that encountered in surfaces constructed via traditional surface pasting. I have developed a second improved surface pasting method that reduces the cross-boundary discontinuity with little or no reduction in approximate $\mathcal{C}^0$ continuity. This second method is called Q^1-pasting because it makes use of the Q^1 operator, and it is based in part on Q^0-pasting.

A feature surface is Q^1-pasted on a base by setting two layers of control vertices for each feature boundary. First, the outermost layer of control vertices is set, using the Q^1 operator to approximate the base surface. Then the adjacent layer is set, using a similar technique, except a derivative field derived from the base is approximated, rather than the base's position.

I will describe the boundary construction procedure for the control vertices that determine the position and cross-boundary derivative of the boundary $u = u_{m-1}$; the procedure for setting the control vertices for each of the other three boundaries is analogous. Control vertices $P_{0,0}, \ldots, P_{0,N}$ determine the position of the boundary in question, and these together with $P_{1,0}, \ldots, P_{1,N}$ determine the cross-boundary derivative.

First, the position of the boundary is set as follows: the base curve $S_{u_{m-1}}(v) = S_B(\mathbf{T}(u_{m-1}, v))$ is approximated using the Q^1 operator, and $P_{0,0}, \ldots, P_{0,N}$ are set using this approximation. Next, the derivative field running across $S_{u_{m-1}}(v)$ on the base must be sampled. More precisely, the field to be approximated is

$$S'_{u_{m-1}}(v) = \frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v)),$$

where u^* is the image of the feature's parametric u direction at (u_{m-1}, v), under $\mathbf{T}$.

The approximation $Q^1 S'_{u_{m-1}}(v)$ is used to produce a number of "control vertices", $V_0, \ldots, V_N$. However, $S'_{u_{m-1}}(v)$ is a vector-valued curve, and so is $Q^1 S'_{u_{m-1}}(v)$. Thus, each V_j is a vector, and the desired cross-boundary curve for S_F is

$$\frac{\partial}{\partial u} S_F(u_{m-1}, v) = \sum_{j=0}^{N} B_j^n(v) V_j \tag{3.34}$$

The V_is indicate how the $P_{1,i}$'s should be set to cause S_F's cross-boundary derivative to approximate $S'_{u_{m-1}}(v)$. The derivative across the boundary $u = u_{m-1}$ is defined by:

$$\frac{\partial}{\partial u} S_F(u_{m-1}, v) = \frac{m B_0^{m-1}(u_{m-1})}{u_m - u_{m-1}} \sum_{j=0}^{N} B_j^n(v)(P_{1,j} - P_{0,j}), \tag{3.35}$$

where the knot vector that defines B_0^{m-1} is $\{u_1, \ldots, u_{M+m-2}\}$. Thus u_{m-1} has full multiplicity with respect to B_0^{m-1}, so $B_0^{m-1}(u_{m-1}) = 1$, and (3.35) can be simplified and combined with (3.34) to indicate the desired settings for the $P_{1,j}$s:

$$\frac{m}{u_m - u_{m-1}} \sum_{j=0}^{N} B_j^n(v)(P_{1,j} - P_{0,j}) = \sum_{j=0}^{N} B_j^n(v) V_j. \tag{3.36}$$

Solving for the $P_{1,j}$s gives

$$P_{1,j} = \frac{u_m - u_{m-1}}{m} V_j + P_{0,j}.$$

It is this setting for the $P_{1,j}$s that I use in Q^1-pasting.

The boundary construction described above sets the feature control vertices

$$P_{0,0}, \ldots, P_{0,N}, P_{1,0}, \ldots, P_{1,N}.$$

Constructing each of the other three boundaries sets the following feature control vertices:

$$\begin{aligned}
&P_{M,0}, \ldots, P_{M,N}, P_{M-1,0}, \ldots, P_{M-1,N}, \\
&P_{0,0}, \ldots, P_{M,0}, P_{0,1}, \ldots, P_{M,1}, \text{ and} \\
&P_{0,N}, \ldots, P_{M,N}, P_{0,N-1}, \ldots, P_{M,N-1},
\end{aligned}$$

as illustrated in Figure 3.6. This figure reveals a potential problem: each corner of the feature surface contains a 2×2 group of control vertices that are set when each of two adjacent boundaries are constructed. The Q^1 operator was designed so these control vertex groups would be set consistently when the two boundaries were constructed. Appendix A.2 contains a verification of this property.

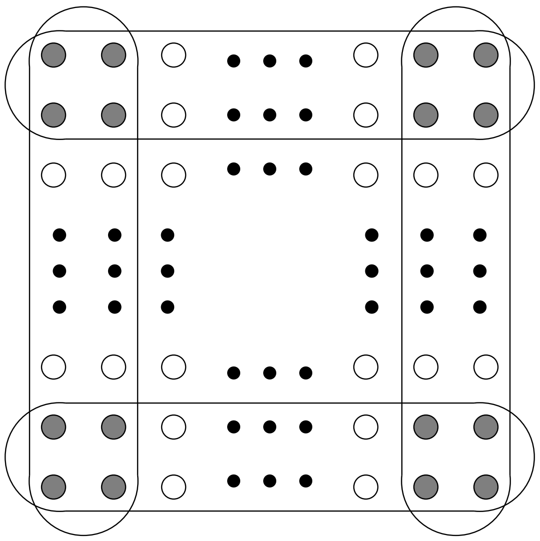


Figure 3.6: Groups of Four Corner Control Vertices are Part of Two Boundaries

3.7 Computational Analysis

One of the primary considerations in the construction of the improved surface pasting techniques developed in this chapter has been the computational complexity of the algorithm. In particular, my goal was to construct a method that was not significantly more expensive than traditional surface pasting. In this section I compare the cost of pasting a feature surface onto a base with traditional pasting, Q^0-pasting, and Q^1-pasting.

In this analysis, I concentrate on the number of affine combinations of control vertices that must be performed to place a control vertex. For the purposes of this discussion, the cost of finding the difference of a pair of control vertices or taking the cross product of a pair of vectors is considered to be one affine combination. The cost, in affine combinations, of mapping a displacement vector into a new coordinate frame is taken to be the dimension of the space minus one; I have been assuming the surfaces exist in a three-dimensional space, so this cost will be two affine combinations.

I also assume that all surfaces being pasted are bicubic tensor product surfaces. This assumption is reasonable because, in practice, many of the surfaces used in modelling and computer animation are bicubic. In §2.3 I calculated the cost of evaluating bicubic surfaces to obtain the values of various surface properties, including position, directional derivatives and others. Table 3.1 summaries some of these costs for easy reference.

Quantities Calculated	Number of Affine Combinations
Position only	30
Position and 1 Derivative	38
Position and 2 Derivatives	39
Position, 2 Derivatives and a Mixed Derivative	63

Table 3.1: Costs of Calculating Bicubic Surface Properties

In all three forms of surface pasting that I have described, a certain number of interior control vertices are set by finding the image of the control vertex's Greville point in the base domain and evaluating the base surface to produce a local coordinate frame. The displacement vector associated with the control vertex is expressed in the local coordinate frame to give the position of the control vertex.

The coordinate frame constructed on the base surface is formed by two directional derivatives and a normal to the base surface. The normal is calculated by taking the cross product of the two directional derivatives. Thus, the position and two directional derivatives at a point on the base surface must be evaluated, at a cost of 39 affine combinations. Forming the third coordinate frame vector from the first two costs one additional affine combination and 2 combinations are used to map the displacement vector into the coordinate frame. Thus, 42 affine combinations are required to place each feature control vertex in this manner.

In the standard surface pasting method, all control vertices are placed as described above, whereas in Q^0- and Q^1-pasting, only selected interior control vertices are set this way. In the remainder of this section, I concentrate on the costs of placing those control vertices whose method of placement is different in either Q^0- or Q^1-pasting than it is in traditional surface pasting.

In Q^0-pasting, the outermost ring of control vertices is set differently than in the standard method. Each of the four corner control vertices is placed so that it is coincident with a particular point on the base surface. Thus, it is sufficient to evaluate the base surface once, for position only, to place each corner, at a cost of 30 affine combinations.

Each of the non-corner boundary control vertices are set to be a convex combination of a group of four base surface points. However, each group of four sample points is potentially used to place more than one control vertex, and one or two sample points may be shared between groups, as shown in Figure 3.4. Consider the number of samples required to place the non-corner control vertices of the feature boundary corresponding to $v = v_{n-1}$.

There are $M - 1$ non-corner control vertices along the boundary, which are divided into at most $\lfloor \frac{M-1}{3} \rfloor + 2$ sample groups. Each sample group consists of four sample points, with the last sample point being shared with the next control group. Thus, at most $3\left(\lfloor \frac{M-1}{3} \rfloor + 2\right) + 1$ distinct sample points must be calculated. There are two cases: $M - 1 \equiv 0 \pmod 3$ or not. If $M - 1 \equiv 0 \pmod 3$, there are exactly $\frac{M-1}{3}$ control vertex groups, as shown in Figure 3.3. In this case, $3\frac{M-1}{3} + 1 = M$ distinct sample points must be calculated. If $M - 1 \not\equiv 0 \pmod 3$, then $\lfloor \frac{M-1}{3} \rfloor < \frac{M-1}{3}$, so $3\left(\lfloor \frac{M-1}{3} \rfloor\right) < M - 1$ and $3\left(\lfloor \frac{M-1}{3} \rfloor + 2\right) + 1 \le M + 5$. Thus, whatever the value of $M - 1$, at most $M + 5$ distinct sample vertices need be computed, at a cost of 30 affine combinations each. Similarly, placing the feature boundary corresponding to $v = v_N$ requires at most $M + 5$ distinct sample points, and the other two boundaries require at most $N + 5$ sample points.

Each non-corner boundary control vertex is a combination of four sample points, which takes the equivalent of 3 affine combinations to evaluate. Thus, the total cost of placing the non-corner boundary control vertices of a feature with Q^0-pasting is at most

$$\begin{aligned} 30(2(M+5+N+5)) &+ 3 \cdot 2(M-1+N-1) = 60(M+N+10) + 6(M+N-2) \\ &= 66(M+N) + 588 \end{aligned}$$

affine combinations. Placing the corner vertices takes an additional 120 affine combinations, for a total of $66(M+N)+708$.

There are $2(M+N)$ boundary control vertices in a surface with $(M+1)\times(N+1)$ control vertices, so the cost of placing all the boundary control vertices for a feature using the standard surface pasting method is $42\cdot 2(M+N) = 84(M+N)$ affine combinations. The cost difference between pasting the boundary control vertices using Q^0-pasting and the standard method is $708+(66-84)(M+N) = 708-18(M+N)$ affine combinations.

It is not uncommon for pasted features to have at least nine control vertices in each parametric direction, which would mean that $M=N=8$. The cost to place the control vertices for such a patch is 3402 affine combinations for the standard method, and 3822 for Q^0-pasting, a fairly reasonable increase considering the relative qualities of the feature boundaries.

In Q^1-pasting, the two outermost rings of control vertices are set differently than in the standard method. At each corner of the feature, a group of four control vertices is placed so the feature matches the base surface in position, two directional derivatives (corresponding to the feature's partial derivatives), and the mixed-partial derivative. Thus, for each feature corner, the base must be evaluated to find the position, two directional derivatives, and a mixed-partial derivative, at a cost of 63 affine combinations.

After the position and derivative vectors are extracted from the base surface, the four control vertices in the corner group must be set. Consider the corner corresponding to parametric values $u=u_{m-1}$ and $v=v_{n-1}$: the control vertices to set are $P_{0,0}, P_{0,1}, P_{1,0}, and P_{1,1}$. I discussed the exact procedure for placing these four control vertices in §3.6.2; the discussion below is only intended to motivate the costs associated with the procedure. $P_{0,0}$ takes the value $S_B(u_{m-1}, v_{n-1})$, while $P_{0,1}$ and $P_{1,0}$ are each set as $P_{0,0}$ plus a scaled directional derivative, which costs one affine combination each to compute. Finally, $P_{1,1}$ can be set by taking $P_{0,1}$ and adding a scaled directional derivative corresponding to the u parametric direction, plus the scaled mixed-partial derivative, costing an additional two affine combinations. Thus, $63+1+1+2=67$ affine combinations are required to set each group of four corner control vertices.

Along each boundary, excluding the four control vertices in each corner, there are two rows of control vertices that must be set. The two rows of control vertices can be regarded as a sequence of control vertex pairs, each of which depends on the position and cross-boundary derivative of 4 base samples. However, each group of four samples is potentially used to place more than one pair of control vertices, and the last sample is shared with the next group, just as in the case of Q^0-pasting. Consider the number of samples required to place the two outermost rows of control vertices of the feature boundary corresponding to $v = v_{n-1}$, excluding the groups of four control vertices at each corner.

There are $M-3$ pairs of non-corner control vertices along the boundary, which are divided into at most $\lfloor \frac{M-3}{3} \rfloor + 2$ sample groups. Each sample group consists of four sample points, with the last sample point being shared with the next control group. Thus, at most $3\left(\lfloor \frac{M-3}{3} \rfloor + 2\right) + 1 \leq M+3$ distinct sample points must be calculated, at a cost of 30 affine combinations each. Similarly, placing the two rows of control vertices that contribute to the feature boundary corresponding to $v = v_N$ requires at most $M+3$ distinct sample points, and the other two boundaries require at most $N+3$ sample points.

Each control vertex that is in the outermost ring, but is not part of a four-vertex corner group, is a combination of four samples, which takes the equivalent of three affine combinations to evaluate. Furthermore, every vertex in the second outermost ring is set to be the position of its neighbour on the outermost ring plus a scalar factor times a cross-boundary derivative approximation. The cross-boundary derivative approximation is, in turn, based on four cross-boundary derivatives, and requires an additional three affine combinations to calculate. Once the derivative approximation is known, scaling it and adding it to the outer control vertex costs only one affine combination. Only the position and one derivative is required for each sample, so the cost of evaluating the base at a sample point is 38 affine combinations. Thus, the total cost of placing the two outermost rings control vertices of a feature, exclusive of the corner groups, with Q^1-pasting is at most

$$\begin{aligned} 38 \cdot 2(M+3+N+3) &+ (2\cdot 3+1)2(M-3+N-3) = 76(M+N+6) + 14(M+N-6) \\ &= 90(M+N) + 372 \end{aligned}$$

affine combinations.

Placing the four groups of corner vertices takes an additional 268 affine combinations, for a total of $90(M+N)+640$. There are $4(M+N-2)$ control vertices in the two outermost rings for a surface with $(M+1)\times(N+1)$ control vertices. These require $42\cdot 4(M+N-2) = 168(M+N)-336$ affine combinations to place using the standard method. Thus the standard method uses an additional $78(M+N)-976$ affine combinations to paste the outer two rows of control vertices. Even feature surfaces with very few control vertices, for example surfaces with 7×8 control vertices, have $M+N\geq 13$, so Q^1-pasting requires fewer affine combinations to place the feature's control vertices.

3.8 Error Bounds for Q^d-Pasting

The error bounds derived in section §3.5 apply when a Q^d operator is used to approximate a general curve that has certain continuity restrictions. In this section, I examine how those error bounds may be used to derive bounds on the errors involved when a feature surface is pasted onto a base using one of the Q^d-pasting methods.

Let S_F be a tensor product surface of degree m in the u parametric direction and degree n in the v parametric direction, with knot vectors $u_0,\ldots,u_{M+m-1}$ and $v_0,\ldots,v_{N+n-1}$. Suppose S_F is pasted on another tensor product surface S_B, with $\mathbf{T}$ being the transformation that maps S_F's domain into S_B's. Without loss of generality, consider how the feature boundary edge $S_F(u,v_{n-1})$ is pasted; the analysis is similar for the other edges.

Denote the curve that underlies $S_F(u,v_{n-1})$ on S_B by $S_{v_{n-1}}(u)$; then $S_{v_{n-1}}(u) = S_B(\mathbf{T}(u,v_{n-1}))$. Place the control vertices of $S_F(u,v_{n-1})$ by applying Q^d to $S_{v_{n-1}}$. If $E^d_{r,s}(t)$ is as defined in Theorem 3.5 with f replaced by $S_{v_{n-1}}$, then from Theorem 3.6,

$$||E^d_{r,s}||_{L_q[u_{m-1},u_M]} \leq K\overline{\Delta}^{s-r-1+\frac{1}{q}}\omega(D^{s-1}S_{v_{n-1}};\overline{\Delta};[u_{m-1},u_M]),$$

if $S_{v_{n-1}}$ is $\mathcal{C}^{s-1}$ over $u_{m-1},\ldots,u_M$ and $s\leq m+1$.

However, $S_{v_{n-1}}$ is a curve on a B-spline surface, so slightly more is known about the error bound. The following lemma will be used to form another expression for the bounds on $E^d_{r,s}$.

Lemma 3.7 *If f is a piecewise polynomial that is at least $\mathcal{C}^0$ on $[a,b]$, then $\omega(f;\Delta;[a,b])$ is either 0 or is $\mathcal{O}(\sqrt{\nu}\Delta)$, where ν is the dimension of the range space of f.*

Proof: Suppose $f(u) = [f_1(u)f_2(u)\cdots f_p(u)]^T$, where f_i is a $\mathcal{C}^0$ polynomial over $[a,b]$ for $i = 1,2,\ldots,p$. There exists some i, with $1 \leq i \leq p$ such that $\omega(f_j;\Delta;[a,b]) \leq \omega(f_i;\Delta;[a,b])$ for $j = 1,\ldots,p$, so $\omega(f;\Delta;[a,b]) \leq \sqrt{\nu}\,\omega(f_i;\Delta;[a,b])$.

If f is constant over $[a,b]$, then each of the f_js are also constant over $[a,b]$, so $\omega(f_i;\Delta;[a,b]) = 0$. Clearly, if f is constant over $[a,b]$ then $\omega(f;\Delta;[a,b]) = 0$.

Otherwise, f_i is a non-constant piecewise polynomial over $[a,b]$. Construct a partition $a = t_0 < t_1 < \cdots < t_\mu = b$ such that f_i is a polynomial over $[t_k, t_{k+1}]$ for $k = 0,\ldots,\mu-1$. Let κ be such that $\omega(f_i;\Delta;[t_k,t_{k+1}]) \leq \omega(f_i;\Delta;[t_\kappa,t_{\kappa+1}])$ for all k. Then $\omega(f_i;\Delta;[a,b]) \leq \mu\omega(f_i;\Delta;[t_\kappa,t_{\kappa+1}])$. However, $\omega(f_i;\Delta;[t_\kappa,t_{\kappa+1}]) = |f_i(x+h) - f_i(x)|$ for some $h \leq \Delta$ with x and $x+h \in [t_\kappa,t_{\kappa+1}]$. The function f_i is a non-constant polynomial over $[t_\kappa,t_{\kappa+1}]$, so $|f_i(x+h) - f_i(x)|$ is at least linear in h. Thus, f_i is $\mathcal{O}(\Delta)$, and so f is $\mathcal{O}(\sqrt{\nu}\Delta)$. □

Theorem 3.8 *Suppose Q^d is as defined in Theorem 3.1 and $E^d_{r,s}$ is the error term defined in Theorem 3.5, with $s \leq m+1$ and $r \leq m$. If f is a $\mathcal{C}^{s-1}$ polynomial over $[u_{m-1}, u_M]$, then*

$$|E^d_{r,s}| \text{ is } \mathcal{O}(\sqrt{\nu}\;\overline{\Delta}^{s-r}), \tag{3.37}$$

where ν is the dimension of the range space of f.

Proof: From Theorem 3.6,

$$|E^d_{r,s}| \leq K\overline{\Delta}^{s-r-1}\omega(D^{s-1}f;\overline{\Delta};[u_{m-1},u_M]).$$

Lemma 3.7 reveals that $\omega(D^{s-1}f;\overline{\Delta};[u_{m-1},u_M])$ is $\mathcal{O}(\sqrt{\nu}\;\overline{\Delta})$ since f is $\mathcal{C}^{s-1}$ on $[u_{m-1},u_M]$, so (3.37) follows immediately. □.

Since $S_{v_{n-1}}$ is a curve on a tensor product B-spline surface, it is a piecewise polynomial over $[u_{m-1}, u_M]$. Recall that I assumed $S_{v_{n-1}}$ to be $\mathcal{C}^{s-1}$ on $[u_{m-1}, u_M]$. Thus, from Theorem 3.8, $|S_F(u, v_{n-1}) - S_B(\mathbf{T}(u, v_{n-1}))|$ is $\mathcal{O}(\sqrt{\nu}\ \overline{\Delta}^s)$. Note that in most surface pasting applications, ν is taken to be 3. Furthermore, error bounds for derivatives along the boundary as well as for cross-boundary derivatives can be calculated. Each of the derivative curves would be $\mathcal{C}^{s-2}$ over $[u_{m-1}, u_M]$, and so $|\frac{\partial}{\partial u} S_F(u, v_{n-1}) - \frac{\partial}{\partial u} S_B(\mathbf{T}(u, v_{n-1}))|$ is $\mathcal{O}(\sqrt{\nu}\ \overline{\Delta}^{s-1})$. The error $|\frac{\partial}{\partial v} S_F(u, v_{n-1}) - \frac{\partial}{\partial v} S_B(\mathbf{T}(u, v_{n-1}))|$ is also $\mathcal{O}(\sqrt{\nu}\ \overline{\Delta}^{s-1})$, if the feature surface is Q^1-pasted.

Frequently a composite pasted surface will be constructed from pasted bicubic tensor product B-splines. If a bicubic tensor product feature is pasted onto a bicubic single base surface that has no repeated interior knots, then the curve underlying each boundary edge is a $\mathcal{C}^2$ sextic B-spline curve. Furthermore, the derivative on the base along the curve that underlines the boundary edge is a $\mathcal{C}^1$ quintic B-spline curve, as is the cross boundary derivative on the base surface. If this common situation holds, then the maximum error for the position of the feature boundary would be $\mathcal{O}(\sqrt{\nu}\ \overline{\Delta}^3)$, and the error for the derivative along the boundary would be $\mathcal{O}(\sqrt{\nu}\ \overline{\Delta}^2)$, as would the error for the cross-boundary derivative if the feature were Q^1-pasted. Note that each repeated knot line that intersects the image of the feature surface's domain will reduce the continuity of the base curve by one less than the multiplicity of the knot.

I have not calculated direct bounds on the differences between tangent planes to the feature and base surfaces. Nevertheless, the bounds on both the cross-boundary derivative and the derivative along the boundary are the best possible and suggest a high rate of convergence for the tangent planes. The empirical results given in Chapter 5 support this conjecture.

No error estimates are available for the case where a feature surface straddles the boundary between two separate components of a composite base surface. As mentioned in §2.5.5, there is no continuity across the boundary between a pasted feature and its base surface. Thus, if the feature edge $S_F(u, v_{n-1})$ is pasted over such a boundary, the underlying curve is not even $\mathcal{C}^0$, so none of the error bounds that have been calculated for the Q^d quasi-interpolation operators apply.

In practice, discontinuity between a Q^0- or Q^1-pasted feature and its base is small, so a feature edge pasted across such a discontinuity approximates its underlying curve to an acceptable tolerance, as long as the discontinuity is located away from the endpoints of the feature boundary.

A feature boundary could be made to poorly approximate the underlying curve if the feature is Q^1-pasted in such a way that a $\mathcal{C}^1$-discontinuity lies near the endpoint of one of its boundaries. In this situation, the second (or second last) control vertex of the boundary would be set using the derivative value from one side of the $\mathcal{C}^1$-discontinuity, while the bulk of the boundary lies over the other side of the discontinuity. If the $\mathcal{C}^1$-discontinuity were large enough, the placement of this control vertex could cause a noticeable gap between the feature boundary and the underlying curve.

An effect that is similar to that described in the last paragraph might be noticed under other circumstances, such as when the endpoint of a Q^1-pasted feature boundary lies on an area of the base that has high curvature, and the knot spacing of the feature boundary is relatively coarse. In general, however, the knot spacing of feature surfaces is assumed to be denser than that of the base surface, so such situations should arise infrequently.

Chapter 4

Implementation — quasiPaste

I have implemented a surface pasting modeller, called *quasiPaste*, that can use quasi-interpolation to place the boundaries of pasted features. The modeller is an extension of Chan's surface pasting modeller, *pasteInterface*, which allows the user to manipulate composite pasted surfaces in world space [Cha96, CMB97]. The modellers *pasteInterface* and *quasiPaste* are implemented using *Open Inventor*.

With *pasteInterface*, an arbitrary tensor product surface may be pasted onto an existing composite surface. The user can then modify the resulting surface by translating or rotating features on their bases, translating individual feature corner points along the base, unpasting individual features, or by using several other operations.

I extended *pasteInterface* to enable the use of Q^0-pasting and Q^1-pasting in addition to standard surface pasting. In *quasiPaste*, the user is able to adjust the pasting style of any surface in the hierarchy. Additional functions that I have provided in *quasiPaste* include the ability to print continuity information or to refine a selected feature. In this chapter I discuss the use of these features in more detail.

I have added three entries to the *Pasting* menu in *quasiPaste* to allow the user to control the pasting style of each surface in the pasting hierarchy: *Non-quasi Pasting*, *Q0-Pasting*, and *Q1-Pasting*. Selecting any of these entries causes any subsequently pasted surfaces to be pasted using the named technique. In addition, if a pasted surface is selected at the time the menu entry is selected, that surface will be repasted using the selected method.

The second feature that exists in *quasiPaste*, but not in *pasteInterface* is surface refinement, which is activated by selecting a surface from the pasted hierarchy and choosing *Refine Surface* from the *Pasting* menu. The selected surface is removed from the pasting hierarchy and replaced with a refined version of the surface that is pasted using the same method that was used to paste the original surface.

The refined surface is constructed from the original surface by inserting knots at the midpoint of each non-degenerate domain interval in each of the parametric directions. The result is a surface that, when unpasted, duplicates the appearance of the original surface. However, the refined surface has four times the number of non-degenerate surface patches. For example, a bicubic tensor product surface with knot vectors

$$\{u_0, u_0, u_0, u_1, u_2, u_2, u_3, u_4, u_4, u_4\}, \text{ and}$$
$$\{v_0, v_0, v_0, v_1, v_2, v_3, v_3, v_3, v_4, v_5, v_5, v_5\}$$

$u_i < u_j$ and $v_i < v_j$ for all $i < j$, would be replaced with knot vectors

$$\left\{u_0, u_0, u_0, \tfrac{u_0+u_1}{2}, u_1, \tfrac{u_1+u_2}{2}, u_2, u_2, \tfrac{u_2+u_3}{2}, u_3, \tfrac{u_3+u_4}{2}, u_4, u_4, u_4\right\}, \text{ and}$$
$$\left\{v_0, v_0, v_0, \tfrac{v_0+v_1}{2}, v_1, \tfrac{v_1+v_2}{2}, v_2, \tfrac{v_2+v_3}{2}, v_3, v_3, v_3, \tfrac{v_3+v_4}{2}, v_4, v_5, v_5, v_5\right\}.$$

Note that the original surface would contain 80 control vertices and the refined surface would contain 168.

The final additional feature that I have included in *quasiPaste* allows the user to access information based on the approximate continuity between a selected surface and the composite base on which it is pasted. When the *Print Continuity Info* option is selected from the *Pasting* menu, the currently-selected surface is sampled at various points around its boundary and these values are compared to those found at corresponding points on the base surface. Statistics based on the comparisons are printed to standard output, as described below.

The feature surface is sampled for position and normal at each of its corners, and these values are compared to the position and normal on the underlying point on the base surface. If either the position differs between feature and base, the difference is output. The amount by which the normals differ is measured by taking the dot product of the normals and subtracting the result from 1. Thus, a normal difference near 0 indicates very good agreement, and a value near 1 indicates nearly perpendicular normals. If the normal difference is non-zero, the difference is output. The corner positions of the feature should coincide with the underlying base surface under the three pasting techniques discussed in this thesis. The normals at the feature corners and the underlying surface should agree when the feature is Q^1-pasted, but not necessarily when either Q^0-pasting or standard surface pasting is used.

Next, each of the four feature boundaries is sampled for position and normal at a number of uniformly-spaced points. The underlying points on the base surface are likewise sampled for position and normal. A parameter, which is changeable only at compile time, determines the number of samples that are taken. The number of samples taken per edge of the feature boundary is equal to the parameter value times the number of feature domain intervals in either of the two parametric domain directions, whichever is greater. At least 100 samples are taken, even if there are very few domain intervals.

A summary of the differences found between the feature and the base is output, including the number of samples taken and the minimum, maximum, average and standard deviation of both the length of the difference position at each point and the normal difference, as defined above. Sample output from the *Print Continuity Information* function appears in Figure 4.1. Note that the position differences reported are not normalised in any way. When sampled approximate continuity values are reported in Chapter 5, the size of the composite surface is given as a frame of reference.

```
Corner 1 normal difference:  1.8275e-03
Corner 2 normal difference:  4.5950e-04
Corner 3 normal difference:  4.8986e-03
Corner 4 normal difference:  4.1460e-04
Continuity sampled at 400 points
  Position length differences:
    min:     7.602457e-05
    max:     4.340427e-02
    avg:     1.479329e-02
    std dev: 1.238206e-02
  Normal differences:
    min:     1.560127e-06
    max:     2.903743e-02
    avg:     5.480401e-03
    std dev: 7.662064e-03
```

Figure 4.1: Format of Approximate Continuity Output

Chapter 5

Empirical Results

In this section I discuss the results obtained when surfaces are pasted using the standard surface pasting technique, Q^0-pasting, Q^1-pasting, and standard surface pasting with knot insertion. I present both numerical and visual data comparing the quality of the boundary between a feature and its base surface. Unless explicitly stated otherwise, all surfaces discussed in this section are bicubic and features are not parametrically aligned with their base surfaces. Continuity sample information provided was generated by sampling each feature edge at 10 different positions for each domain interval in the u or v parametric direction, whichever was greater, with a minimum of 100 samples taken no matter the number of domain intervals.

Figure 5.1 shows the result of pasting a bicubic surface with $[6 \times 6]$ subpatches, on a bicubic base, using standard surface pasting method and using Q^0-pasting. Note that the gap between the base and feature surfaces in Figure 5.1(b) is greatly reduced, relative the gap in Figure 5.1(a). Approximate continuity statistics gathered from both composite surfaces appear in Figure 5.2.

Figures 5.3(a) and 5.3(b) show the composite surfaces presented above, but from a slightly different angle. The gap between the feature and the base is still visible in the former image, and is all but eliminated from the latter, but another interesting artifact is revealed. The cross-boundary derivative fields of the feature and base match more closely in the composite surface that was constructed using standard pasting than in the Q^0-pasted surface. The statistics reported in

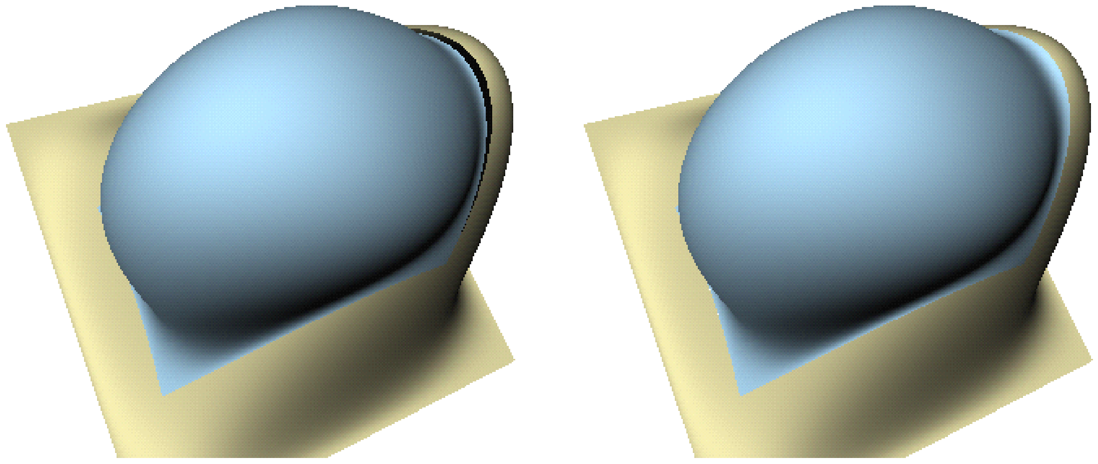

(a) Standard Pasting (b) Q^0-Pasting

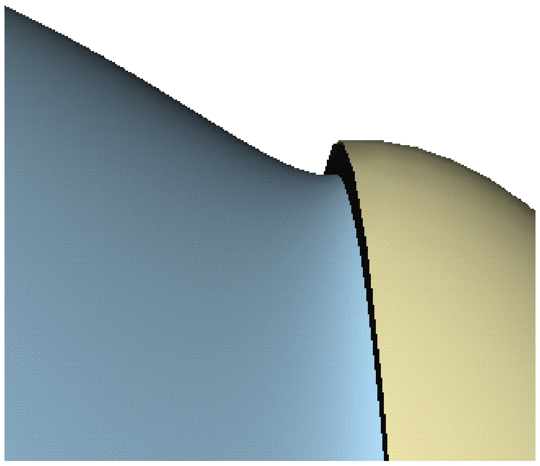
Figure 5.1: Standard and Q^0-Pasting Example

```
Corner 1 normal difference:  4.224112e-04
Corner 2 normal difference:  1.347006e-04
Corner 3 normal difference:  1.118903e-03   Corner 2 normal difference:  1.500367e-05
Corner 4 normal difference:  1.162927e-04   Corner 4 normal difference:  1.339990e-06
Continuity sampled at 400 points            Continuity sampled at 400 points
  Position length differences:                Position length differences:
    min:     3.736685e-05                       min:     1.145530e-09
    max:     1.572044e-02                       max:     2.570768e-03
    avg:     5.392649e-03                       avg:     2.581981e-04
    std dev: 4.397894e-03                       std dev: 5.765663e-04
  Normal differences:                         Dot product of normals:
    min:     6.496074e-06                       min:     6.777174e-09
    max:     7.198891e-03                       max:     2.341422e-01
    avg:     1.108036e-03                       avg:     2.608988e-02
    std dev: 1.718773e-03                       std dev: 5.590261e-02
```

(a) Standard Pasting (b) Q^0-Pasting

Figure 5.2: Sampled Continuity Information — opposite base corners are separated by 1.414 units

Figures 5.2(a) and 5.2(b) regarding the dot product of the normals reflect this lack of continuity.


(a) Standard Pasting

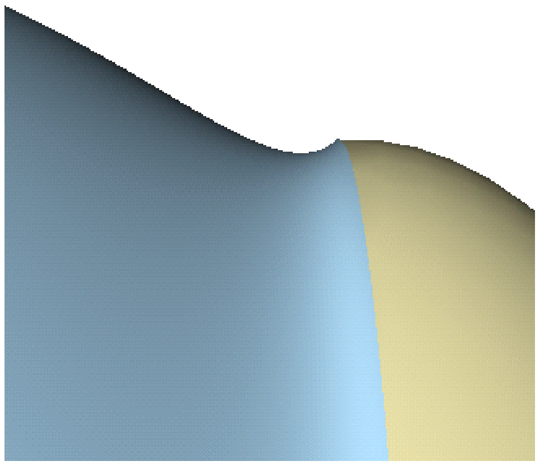
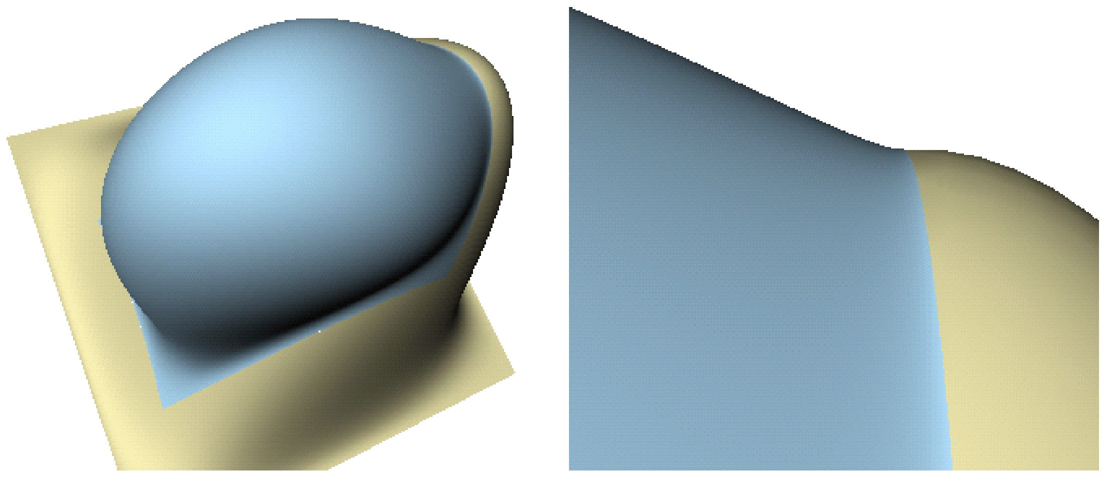

(b) Q^0-Pasting

Figure 5.3: Cross-Boundary Agreement Example

As mentioned earlier, the cross-boundary derivative field is determined by the difference between control vertices in the outermost row of the surface and their neighbours in the next outermost row. In traditional surface pasting, all control vertices in the two outermost rows of the feature surface are placed on the base surface. Thus, the vector between adjacent control vertices is a difference of positions, which is a reasonable approximation of a scalar multiple of the cross-boundary derivative. Therefore, the cross-boundary derivative field of the feature that was pasted using the standard method is often a close approximation to the corresponding base derivative field.

In Q^0-pasting the second row of control vertices is placed just as in the standard method, but the outermost layer is placed differently. In general, Q^0-pasting places these control vertices off the base surface. Thus, the vectors formed by control vertices in the outermost rows of the feature and their neighbours in the next row have no relationship to the base's cross-boundary derivative field. As a result, features pasted using Q^0-pasting are not expected to have a high degree of correspondence between the cross-boundary derivative fields of the base and feature.

Figures 5.4(a) and 5.4(b) reveal the results of repasting the feature using Q^1-pasting. The gap between the base and the feature is significantly reduced, as in the Q^0-pasted example, and the cross-boundary derivative fields between the feature and base match much more closely than when the feature is Q^0-pasted. The continuity statistics generated from this composite surface reveal the high degree of approximate $\mathcal{C}^0$ and $\mathcal{C}^1$ continuity, as can be seen in Figure 5.5.



(a) Q^1-Pasting (b) Q^1-Pasting Cross-Boundary Example

Figure 5.4: Q^1-Pasting Example

Table 5.1 contains a summary of sampled continuity statistics. In addition to the information that has been provided in this chapter, three extra rows have been included. The last three rows contain information about the approximate continuity that results when a feature surface is refined according to the refinement operation described in Chapter 5 and then pasted using standard surface pasting. I have provided data for a feature that has been refined once and then pasted, and also for features that were refined twice or three times before being pasted.

The table reveals that the Q^1-pasted feature has superior approximate $\mathcal{C}^0$ and $\mathcal{C}^1$ continuity, compared to that of a feature that was pasted using the standard method after zero or one refinement operations have been applied. The feature that had the refinement operation applied twice before being pasted using standard surface pasting displays superior $\mathcal{C}^0$ approximate continuity

```
Continuity sampled at 400 points
  Position length differences:
    min:     7.843729e-11
    max:     1.864818e-03
    avg:     1.413118e-04
    std dev: 3.853936e-04
  Dot product of normals:
    min:     3.050893e-13
    max:     4.842428e-05
    avg:     3.109586e-06
    std dev: 8.310277e-06
```

Figure 5.5: Sampled Continuity Information, Q^1-Pasting — opposite base corners are separated by 1.414 units

and $\mathcal{C}^1$ continuity that is nearly comparable to that of the Q^1-pasted feature. Note, however that the doubly refined feature has 729 control vertices compared to 81 for the Q^1-pasted feature, and requires 30618 affine combinations to perform the pasting, compared to only 3030 for the Q^1-pasted feature. Thus, for this example, 10 times the computations would have to be performed using standard pasting in order to obtain a composite surface that has quality comparable to the Q^1-pasted surface.

Pasting Mode	Mean Position Difference	Maximum Position Difference	Maximum Normal Difference	# Control Vertices	# Affine Combinations
Standard	5.392e-03	1.572e-02	7.198e-03	81	3402
Q^0	2.581e-04	2.570e-03	2.341e-01	81	3822
Q^1	1.413e-04	1.864e-03	4.842e-05	81	3030
Standard, 1 Refinement	1.543e-03	4.550e-03	1.670e-03	225	9450
Standard, 2 Refinements	4.089e-04	1.149e-03	3.847e-04	729	30618
Standard, 3 Refinements	1.049e-04	2.891e-04	9.641e-05	2061	109242

Table 5.1: Feature Continuity and Complexity Statistics

Empirical data were also gathered that measured the reduction in the largest $\mathcal{C}^0$ discontinuity along a feature boundary. Specifically, a bicubic feature surface was pasted onto a $\mathcal{C}^2$ bicubic base surface in such a way that the feature's parametric domain directions were not aligned with the base's domain, using both standard and Q^0 surface. The maximum position differences between the feature boundary and the base surface were sampled for each pasting surface. This procedure was repeated several times, and at each step the feature surface was refined one more level than for the previous step. These data appear in Table 5.2.

	Q^0-Pasting		Standard Pasting	
Refinement Level	Max Position Error	Ratio to Previous	Max Position Error	Ratio to Previous
0	2.566e-03	N/A	1.571e-02	N/A
1	3.875e-04	6.62	4.550e-03	3.45
2	8.974e-05	4.31	1.149e-03	3.95
3	1.206e-05	7.43	2.886e-04	3.98
4	9.171e-07	13.15	7.231e-05	3.99
5	2.298e-07	3.99	1.809e-05	3.99
6	4.312e-08	5.32	4.526e-06	3.99
7	1.712e-10	25.17	1.131e-06	4.00

Table 5.2: $\mathcal{C}^0$ Error Convergence — Bicubic Feature on a Bicubic Base

Note that in Table 5.2, the maximum $\mathcal{C}^0$ boundary discontinuity for the Q^0-pasted surface is consistently smaller than the maximum $\mathcal{C}^0$ boundary discontinuity for the surface created using standard surface pasting. In addition, the overall rate of convergence of the error is faster for the Q^0-pasted surface. For the samples given, the geometric average of the amount by which the $\mathcal{C}^0$ discontinuity is reduced is approximately a factor of 10.0 per refinement, while the average amount by which the discontinuity in the standard pasted surface is reduced is 3.90.

The table reveals a curious result, however. After two refinements, the amount by which the $\mathcal{C}^0$ discontinuity in the standard pasted surface is reduced is consistently near 4. This suggests that the order of the error is $\mathcal{O}(h^2)$, where h is the maximum knot difference in one of the feature's knot vectors. However, the reduction in the position difference for the Q^0-pasted surface at each step shows no signs of converging, although the theoretical results obtained in §3.8 indicate that the order of the error is $\mathcal{O}(h^3)$.

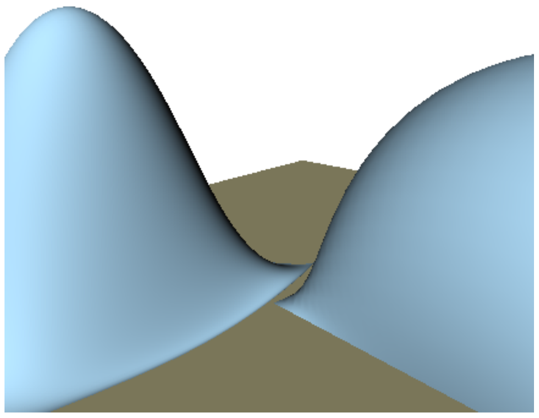
Given the theoretical error bounds, the size of the maximum position difference between the boundary of the Q^0-pasted feature and its base should be reduced by a factor of 8 at each refinement. I have no theory to account for the uneven reduction in the error for the Q^0-pasted surface.

I performed another experiment similar to the one that produced the data in Table 5.3. This second experiment examined the reduction of the position error along a bicubic feature surface pasted on a biquartic base. The results of the test appear in Table 5.3. The data in this table reveal that the maximum position difference along the boundary of the standard pasted feature is reduced by a factor of 4 at each refinement, just as before. The error for the boundary for the Q^0-pasted surface, however, behaves better than in the previous test. The ratio of successive error values approaches 16, the result indicated by the theoretical error bound of $\mathcal{O}(h^4)$.

	Q^0-Pasting		Standard Pasting	
Refinement Level	Max Position Error	Ratio to Previous	Max Position Error	Ratio to Previous
0	1.423e-03	N/A	2.939e-02	N/A
1	1.501e-04	9.48	8.565e-03	3.43
2	1.122e-05	13.37	2.163e-03	3.95
3	7.846e-05	14.30	2.886e-04	3.98
4	1.050e-07	7.47	1.360e-04	3.99
5	6.684e-09	15.71	3.402e-05	3.99
6	4.183e-10	15.97	8.509e-06	3.99
7	2.616e-11	15.98	2.127e-06	4.00

Table 5.3: $\mathcal{C}^0$ Error Convergence — Bicubic Feature on a Biquartic Base

Finally, I will demonstrate a situation where Q^1-pasting gives a noticeably lower-quality surface than does Q^0-pasting. This situation is an example of the conditions outlined at the end of §3.8, where I conjecture on instances in which Q^d-pasting might not be desirable. In Figure 5.6, the leftmost feature has been pasted across the boundary of the base and the rightmost feature, which contains a large $\mathcal{C}^1$ discontinuity. In the Q^1-pasted case, the leftmost feature's cross-boundary derivatives are forced to agree with the underlying surface. As a result, the second-last control vertex on the feature boundary can not be set so as to have the boundary most closely approximate the underlying curve. In the Q^0-pasted example, only the first and last control vertices are set to interpolate the position of the underlying curve, so the remaining vertices are placed to produce a better approximation of the base curve.


(a) Q^1-Pasting

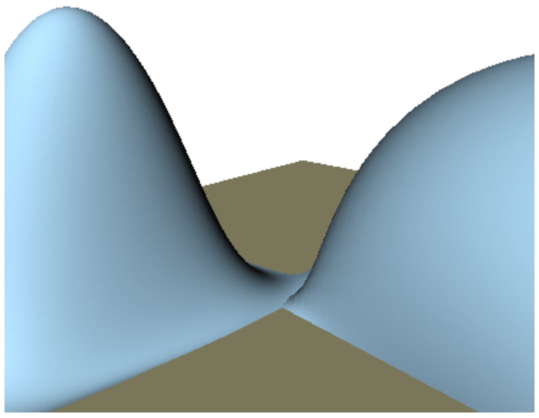

(b) Q^0-Pasting

Figure 5.6: Q^0-Pasting Behaves Better than Q^1-Pasting

Chapter 6

Conclusion

6.1 Summary

A number of research projects have concentrated on the uses of surface pasting, including projects that studied the feasibility of using surface pasting with both domain space [Bar94] and world space [Cha96] user interfaces, as well as one that studied the behaviour of animated pasted surfaces [Tsa98]. Surface pasting has also been incorporated into *Houdini*, a commercially available system for developing computer animations. Surface pasting has been demonstrated to be a flexible tool for interactively constructing composite surfaces that contain areas of local detail.

While surface pasting has a number of strengths, including its flexible modelling paradigm and the fact that it can be used at interactive speeds on currently available hardware, it has been criticised for the lack of continuity between a pasted feature and the base. If a feature surface has a coarse knot structure or is pasted on an area of the base that has high curvature, there can be a noticeable gap between the feature and the base. I undertook the line of research presented in this thesis with the intention of developing an alternative pasting technique that reduces the gaps around the feature surface without unduly increasing the cost of pasting the feature.

In this thesis, I developed a special class of quasi-interpolation operators based on those proposed by Lyche and Schumaker [LS75]. These Q^d operators can be used to approximate smooth curves with a high degree of accuracy. The approximation is constructed so that, at the endpoints of the approximation, the position and first d derivatives of the underlying curve are reproduced. In addition, the Q^d operator requires only point evaluation of the underlying curve, except at the approximation endpoints, where the curve must be evaluated at its first d derivatives.

When a feature surface is moved on the base surface, the sampling of the underlying curve must be recalculated, but the coefficients that weight the samples remain fixed. I proposed an alternative set of linear functionals that may be used to construct the Q^d operators. My set of linear functionals have a lower evaluation cost than those proposed by Lyche and Schumaker, at the expense of increasing the computation required to calculate the coefficients that weight the linear functionals.

Lyche and Schumaker make no mention of the cost of applying their quasi-interpolations operator. I have developed a method to limit the cost of applying the Q^d operators so the total cost of pasting a feature using my method is not significantly higher than for standard surface pasting. I reduce the cost of applying the Q^d operators by grouping the control vertices in the approximation $Q^d f$. Each group of control vertices could contain up to as many control vertices as the degree of the approximating curve. Each control vertex in the group would then be placed using the same set of linear functionals; only the coefficients used to weight each linear functional would differ between vertices in a given group. In addition, by carefully choosing the linear functionals used in the Q^d operator, I share some linear functionals between adjacent control vertex groups. Use of the grouping mechanism that I propose allows the construction of curve approximations using only slightly more than one linear functional evaluation per control vertex in the approximation.

The approximation $Q^d f$ reproduces f to a high degree of accuracy. The order of convergence of $|D^r Q^d - D^r f|$ is $\mathcal{O}(h^{q+1-r})$ where q is the minimum of the degree of the approximating curve and the continuity of f over the region to be approximated. A curious result of this convergence bound is that the error involved when a cubic curve is used to approximate a $\mathcal{C}^3$ piecewise quartic curve converges more quickly than when a cubic curve is used to approximate a $\mathcal{C}^2$ piecewise cubic curve. Note that the use of control vertex groups to share linear functionals increases the coefficient of the approximation's rate of convergence. Excellent curve approximations were obtained in empirical tests, but if the higher coefficient becomes a concern, the maximum number of control vertices in a control vertex group could be reduced, with a corresponding increase in the cost of constructing the approximation.

Error bounds were not calculated for the case in which an approximation to a less than $\mathcal{C}^0$ curve is generated. In such a situation, the error $|Q^d f - f|$ can never be reduced to less than half of the size of the largest jump discontinuity, so the order of convergence is at best $\mathcal{O}(1)$ on any interval that is less than $\mathcal{C}^0$. Despite the lack of mathematical bounds, empirical testing showed that the approximating curve behaved well, with reasonable errors in the discontinuous region.

I suggested methods by which the Q^d operators may be used to place one or more boundary control vertex rings for pasted features. Features constructed using either Q^0-pasting or Q^1-pasting had significantly reduced $\mathcal{C}^0$ discontinuities between the feature boundary and the base surface, relative surfaces obtained from the standard pasting technique. The feature cross-boundary derivatives of the Q^0-pasted surfaces tended to match the corresponding derivative on the base surface poorly. Usually this correspondence was worse than in the surfaces constructed using standard surface pasting. The Q^1-pasted surfaces exhibited much better cross-boundary derivative correspondence than surfaces produced using either Q^0-pasting or the standard method.

The use of efficient tensor product B-spline evaluation techniques reduces the cost of forming pasted surfaces using either of the methods that I propose. The cost of pasting a small bicubic feature using Q^0-pasting is approximately one third higher than pasting the same feature using the standard method. As the size of the pasted feature grows, the difference in cost between Q^0-pasting and the standard method diminishes. When the sum of the number of rows and columns of control vertices in a feature reaches 20, the cost of pasting the feature is approximately equal, and with larger features, Q^0-pasting becomes cheaper. Pasting a feature using Q^1-pasting is cheaper than using standard surface pasting for features as small as 7×8 control vertices, and Q^1-pasting is comparatively less expensive with increasingly larger feature surfaces. In addition, if the feature surface is of higher than bicubic degree, the per vertex cost savings of Q^0- and

Q^1-pasting is improved.

The cost comparisons mentioned in the previous paragraph assume that the pasting methods operate on equal-sized feature surfaces. The greatest increase in efficiency in using Q^0- or Q^1-pasting is not realized by having a lower average cost of placing each feature control vertex. The greatest benefit to using the new pasting techniques is that the boundaries of the feature surfaces meet the base with a higher order of approximate continuity than they do when the features are placed using the standard method. Thus, it is possible to produce a feature surface using Q^1-pasting that has an error tolerance comparable to that of a feature produced using the standard method, but with considerably fewer boundary control vertices. Thus the total number of control vertices, and hence the cost of pasting, would be greatly reduced relative the standard pasting method.

The use of a Q^d-pasting technique should reduce the amount of user effort required to model surfaces, relative the effort required to model surfaces using the standard pasting technique. Empirical testing has shown that when identical features are pasted on a base surface, the quality of the composite surface is higher if the feature is pasted using Q^d-pasting, relative the quality of a surface constructed using standard surface pasting. The improved surface quality means that there should be less need to refine a pasted feature to obtain a more attractive composite surface. The reduction in paste-examine-refine cycles should translate into less effort expended by the user of the modelling software.

Empirical testing has revealed cases in which the cost of pasting a feature with Q^1-pasting is an order of magnitude less than the cost of refining the feature and using standard surface pasting to obtain a composite surface with comparable quality. The knot spacing in the feature and base surfaces, the curvature of the base surface, and other factors combine to affect the quality of surfaces produced using any surface pasting technique. Thus, the improvement in surface quality realized when using a form of Q^d-pasting over standard pasting will vary.

One last advantage to using either of the new surface pasting techniques is that they should improve the quality of texture mapped composite surfaces. The high order of approximate continuity between the base and feature surfaces considers the parameterisation of the surfaces. Thus, the texture coordinates should not suffer from any "drift" that would be apparent if the approximate continuity were merely geometric in nature.

6.2 Future Work

There are a large number of variations on the Q^d operators that might be explored in the future. In particular, further research should be done to experiment with the placement of the $\tau_{i,j}$s. Perhaps certain non-uniform placements, such as a Tchebychev arrangement, might yield better results than the uniform placement that I suggest.

In addition, the decision not to use derivative samples when setting the interior control points of the curve approximation could be examined. Although the coefficients needed to weight the linear functionals would be more complicated, there might be an improvement in the coefficient in the error term.

Likewise, the decision to group control vertices and share curve samples should be reviewed. An alternative method might involve constructing a set of slightly more linear functionals than there are control vertices, and having a sliding window that dictates which samples are used to place each control vertex. Otherwise, if the linear functionals are not shared between control vertices in the approximation, it is possible to choose linear functionals that cause quasi-interpolation operators to reproduce splines. The error coefficient would certainly be reduced in this case, but research would be necessary to determine whether the reduction justifies the added cost of using unshared linear functionals.

One technique that could be used to improve the agreement between the feature boundary and the underlying base curves would be to detect the knot lines on the base and adjust the knot values for the feature surface. This method can be applied to the standard surface pasting method and would, with the additional constraint that features be parametrically aligned with their base and have equal or higher degree, allow the feature boundaries to exactly reproduce the underlying base curve. If this technique is applied to a Q^d-pasted surface, it would be possible to arrange the knots so the feature's boundary control vertices are placed to interpolate each of the polynomial sections of the base curve. Thus, the order of approximation would be increased to the maximum based on the degree of the feature boundary. Likewise, if the feature surface were parametrically aligned with the base, the knot vectors matched properly, and the feature surface had equal or higher degrees in each direction, exact reproduction of the base curves would occur.

One further optimisation applies only to Q^1 or higher pasting. If the cross-boundary derivative field on the base is being sampled, only one more affine combination is required to sample the derivative along the boundary as well. This additional derivative could be used to place the outermost ring of control vertices, effectively doubling the base samples that are available for this purpose. One possible scenario would have each sample of the boundary derivative paired with the corresponding position sample. Thus, the size of the interval spanning the support of the linear functionals used to set a control vertex would be nearly halved, which would reduce the coefficient in the error bound significantly.

Finally, it would be interesting to extend quasi-interpolated surface pasting to apply to the interior control points of pasted features as well. The Q^d-pasting techniques that I have developed only attempts to cause the boundary of the pasted feature to approximate the base surface. By placing each interior control vertex using quasi-interpolation techniques, it would be possible to have the entire pasted feature more closely approximate the ideal displacement mapped feature. A naive implementation of whole surface quasi-interpolated pasting would use $(degree_u + 1) \times (degree_v + 1)$ linear functionals per pasted control vertex, possibly reduced by a sharing technique similar to the one I describe in this thesis. However, it likely would be desirable to retain the current Q^d-pasting methods for the outer rings of control vertices.

The theoretical justification for quasi-interpolation is that the B-spline curves do form a basis for polynomial curves of the same degree. Therefore, a major drawback to whole surface Q^d-pasting is that tensor product B-splines surfaces do not form a basis of polynomial surfaces. I anticipate, however, that whole surface Q^d-pasting would produce results that would closely approximate analogous displacement mapped surfaces.

Appendix A

Verifying Corner Consistency

A.1 Corner Consistency in Q^0-pasting

In this section I provide a verification that the corner control vertices of a Q^0-pasted feature surface are set consistently. The material in this section directly references the contents of §3.6.1.

I will show that control vertex $P_{0,0}$ is set consistently; the proof that the other three corners are set properly is similar.

Control vertex $P_{0,0}$ contributes to the boundary curve corresponding to parameter value $u = u_{m-1}$, and to the boundary $v = v_{n-1}$. These boundaries are constructed by approximating the curves $S_{u_{m-1}}(v) = S_B(\mathbf{T}(u_{m-1}, v))$ and $S_{v_{n-1}}(u) = S_B(\mathbf{T}(u, v_{n-1}))$, respectively. Thus, $S_F(u, v_{n-1}) = Q^0 S_{v_{n-1}}(u)$, and $S_F(u_{m-1}, v) = Q^0 S_{u_{m-1}}(v)$.

Since S_F is a tensor product surface whose end knots have full multiplicity, $S_F(u_{m-1}, v_{n-1}) = P_{0,0}$, so it suffices to show that the two boundary curves agree on a single value of $S_F(u_{m-1}, v_{n-1})$. Consider the feature boundary curves evaluated at the beginning of their domains: by the definition of Q^0,

$$\begin{aligned} Q^0 S_{v_{n-1}}(u_{m-1}) &= S_{v_{n-1}}(u_{m-1}) = S_B(\mathbf{T}(u_{m-1}, v_{n-1})), \text{ and} \\ Q^0 S_{u_{m-1}}(v_{n-1}) &= S_{u_{m-1}}(v_{n-1}) = S_B(\mathbf{T}(u_{m-1}, v_{n-1})). \end{aligned}$$

Thus, using the Q^0 operator to set the boundary control vertices of a feature results in a consistent setting for the corner control vertices of the feature.

A.2 Corner Consistency in Q^1-pasting

In this section I provide a verification that each group of four corner control vertices of a Q^1-pasted feature surface is set consistently. The material in this section directly references the contents of §3.6.2.

I will show how the control vertices $P_{0,0}, P_{0,1}, P_{1,0}$, and $P_{1,1}$ are set consistently when the adjacent boundaries are constructed; the proof that the other three groups are set consistently is similar.

I will first describe how the four control vertices are set when the boundary $u = u_{m-1}$ is constructed, and then compare this to how they are set when the boundary $v = v_{n-1}$ is constructed, to show that the vertex positions are the same. I constructed Q^1 so the position and first derivative at the endpoints of an approximation, $Q^1 f$, to a base curve f match those of the base curve. Thus, when boundary $u = u_{m-1}$ is constructed, $P_{0,0}$ and $P_{0,1}$ are set so $S_F(u_{m-1}, v_{n-1}) = S_{u_{m-1}}(v_{n-1}) = S_B(\mathbf{T}(u_{m-1}, v_{n-1}))$, and

$$\frac{\partial}{\partial v} S_F(u_{m-1}, v_{n-1}) = \frac{\partial}{\partial v} S_{u_{m-1}}(v_{n-1}) = \frac{\partial}{\partial v^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1}))$$

where v^* is the image, under $\mathbf{T}$, of S_F's parametric v direction at (u, v). Since S_F is a tensor product surface whose end knots have full multiplicity, the restrictions on $P_{0,0}$ and $P_{0,1}$ mean that

$$P_{0,0} = S_B(\mathbf{T}(u_{m-1}, v_{n-1})) \quad \text{and} \tag{A.1}$$

$$P_{0,1} = P_{0,0} + \frac{v_n - v_{n-1}}{n} \frac{\partial}{\partial v^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})). \tag{A.2}$$

Q^1 is used to place $P_{1,0}$ and $P_{1,1}$ as well, using values obtained from

$$\frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})).$$

Specifically, $P_{1,0}$ is set so

$$\frac{\partial}{\partial u} S_F(u_{m-1}, v_{n-1}) = \frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})).$$

This requires that

$$P_{1,0} = P_{0,0} + \frac{u_m - u_{m-1}}{m} \frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})). \tag{A.3}$$

Furthermore, $P_{1,1}$'s position must be such that

$$\frac{\partial}{\partial v} \frac{\partial}{\partial u} S_F(u_{m-1}, v_{n-1}) = \frac{\partial}{\partial v^*} \frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})),$$

which translates into the following restriction:

$$\begin{aligned}
P_{1,1} = P_{0,0} \quad &+ \quad \frac{u_m - u_{m-1}}{m} \frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})) \\
&+ \quad \frac{v_n - v_{n-1}}{n} \frac{\partial}{\partial v^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})) \\
&+ \quad \frac{v_n - u_{v-1}}{n} \frac{u_m - u_{m-1}}{m} \frac{\partial}{\partial v^*} \frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})), \text{ since} \qquad (A.4) \\
P_{1,1} = P_{0,0} \quad &+ \quad \frac{u_m - u_{m-1}}{m} \frac{\partial}{\partial u} S_F(u_{m-1}, v_{n-1}) \\
&+ \quad \frac{v_n - v_{n-1}}{n} \frac{\partial}{\partial v} S_F(u_{m-1}, v_{n-1}) \\
&+ \quad \frac{v_n - v_{n-1}}{n} \frac{u_m - u_{m-1}}{m} \frac{\partial}{\partial v} \frac{\partial}{\partial u} S_F(u_{m-1}, v_{n-1}).
\end{aligned}$$

I now consider how $P_{0,0}, P_{0,1}, P_{1,0}$ and $P_{1,1}$ are set when the feature boundary $v = v_{n-1}$ is constructed. Define $S_{v_{n-1}}$ and $S'_{v_{n-1}}$ similarly to $S_{u_{m-1}}$ and $S'_{u_{m-1}}$:

$$\begin{aligned}
S_{v_{n-1}}(u) &= S_B(\mathbf{T}(u, v_{n-1})) \\
S'_{v_{n-1}}(u) &= \frac{\partial}{\partial v^*} S_B(\mathbf{T}(u, v_{n-1}))
\end{aligned}$$

$P_{0,0}$ and $P_{1,0}$ are set so $S_F(u_{m-1}, v_{n-1}) = S_{v_{n-1}}(u_{m-1}) = S_B(\mathbf{T}(u_{m-1}, v_{n-1}))$, and

$$\frac{\partial}{\partial u} S_F(u_{m-1}, v_{n-1}) = \frac{\partial}{\partial u} S_{v_{n-1}}(u_{m-1}) = \frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1}))$$

Since S_F is a tensor product surface whose end knots have full multiplicity, the restrictions on $P_{0,0}$ and $P_{1,0}$ mean that

$$\begin{aligned} P_{0,0} &= S_B(\mathbf{T}(u_{m-1}, v_{n-1})) \quad \text{and} && \text{(A.5)} \\ P_{1,0} &= P_{0,0} + \frac{u_m - u_{m-1}}{m} \frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})) && \text{(A.6)} \end{aligned}$$

Q^1 is used to place $P_{0,1}$ and $P_{1,1}$ as well, using values obtained from $S'_{v_{n-1}}(u)$. Specifically, $P_{0,1}$ is set so

$$\frac{\partial}{\partial v} S_F(u_{m-1}, v_{n-1}) = S'_{v_{n-1}}(u_{m-1}) = \frac{\partial}{\partial v^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})).$$

This requires that

$$P_{0,1} = P_{0,0} + \frac{v_n - v_{n-1}}{n} \frac{\partial}{\partial v^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})). \qquad \text{(A.7)}$$

Furthermore, $P_{1,1}$'s position must be such that

$$\frac{\partial}{\partial v} \frac{\partial}{\partial u} S_F(u_{m-1}, v_{n-1}) = \frac{\partial}{\partial u^*} S'_{v_{n-1}}(u_{m-1}) = \frac{\partial}{\partial v^*} \frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})),$$

which translates into the restriction that

$$\begin{aligned} P_{1,1} = P_{0,0} &+ \frac{v_n - v_{n-1}}{n} \frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})) \\ &+ \frac{u_m - u_{m-1}}{m} \frac{\partial}{\partial u^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})) \\ &+ \frac{u_m - u_{m-1}}{m} \frac{v_n - v_{n-1}}{n} \frac{\partial}{\partial u^*} \frac{\partial}{\partial v^*} S_B(\mathbf{T}(u_{m-1}, v_{n-1})). \qquad \text{(A.8)} \end{aligned}$$

Equations (A.1) and (A.5), (A.2) and (A.7), (A.3) and (A.6), and (A.4) and (A.8) show that the four control vertices $P_{0,0}$, $P_{0,1}$, $P_{1,0}$ and $P_{1,1}$ are set consistently. Similar arguments show that the other three corner control vertex groups are set consistently.

Bibliography

[Bar94] C. Barghiel. Feature oriented composition of B-spline surfaces. Master's thesis, Univeristy of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1994. (Available as Computer Science Department Technical Report CS-94-13).

[BBF95] C. Barghiel, R. Bartels, and D. Forsey. Pasting spline surfaces. In L. Schumaker M Daehlen, T. Lyche, editor, *Mathematical Methods for Curves and Surfaces*, pages 31–40. Vanderbilt University Press, 1995.

[BF91] R. Bartels and D. Forsey. Spline overlay surfaces. Technical Report CS-92-08, Univeristy of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1991.

[Boe80] W. Boehm. Inserting new knots into a B-spline curve. *Computer-Aided Design*, 12:199–201, 1980.

[Car95] Jeromy Carrière. Evaluating tensor product and triangular Bézier surfaces. Technical Report CS-95-22, Univeristy of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1995.

[Cha96] L. K. Y. Chan. World space user interface for surface pasting. Master's thesis, Univeristy of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1996. Available on WWW as ftp://cs-archive.uwaterloo.ca/cs-archive/CS-96-32/.

[Chu92] Charles Chui. *An Introduction to Wavelets.* Academic Press, 1992.

[CLR80] E. Cohen, T. Lyche, and R. Riesenfeld. Discrete B-splines and subdivision techniques in computer aided geometric design and computer graphics. *Computer Graphics and aImage Processing*, 14(2):87–111, 1980.

[CMB97] L. K. Y. Chan, S. Mann, and R. Bartels. World space surface pasting. In *Proceedings of Graphics Interface*, pages 146–154, Kelowna, May 1997.

[dB78] Carl de Boor. *A Practical Guide to Splines.* Applied Mathematical Sciences. Springer-Verlag, New York, 1978.

[dBF73] C. de Boor and G. J. Fix. Spline approximation by quasiinterpolants. *Journal of Approximation Theory*, 8:19–45, 1973.

[Far93] G. E. Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, Inc., 1993.

[FB88] D. Forsey and R. Bartels. Hierarchical B-spline refinement. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):205–212, 1988.

[LS75] T. Lyche and L. Schumaker. Local spline approximation methods. *Journal of Approximation Theory*, 15:294–325, 1975.

[MD95] Stephen Mann and Tony DeRose. Computing values and derivatives of Bézier and B-spline tensor products. *Computer Aided Geometric Design*, 12(1):107–110, February 1995.

[Tsa98] C. L. F. Tsang. Animated surface pasting. Master's thesis, Univeristy of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1998. Available on WWW from ftp://cs-archive.uwaterloo.ca/cs-archive/CS-98-19/.