A Geometric B-Spline Over the Triangular Domain

by

Christopher K. Ingram

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, 2003

I hereby declare that I am the sole author of this thesis. This is a true copy of my thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

For modelling curves, B-splines [3] are among the most versatile control schemes. However, scaling this technique to surface patches has proven to be a non-trivial endeavor. While a suitable scheme exists for rectangular patches in the form of tensor product B-splines, techniques involving the triangular domain are much less spectacular.

The current cutting edge in triangular B-splines is the DMS-spline [2]. While the resulting surfaces possess high degrees of continuity, the control scheme is awkward and the evaluation is computationally expensive. A more fundamental problem is the construction bears little resemblance to the construction used for the B-Spline. This deficiency leads to the central idea of the thesis; what happens if the simple blending functions found at the heart of the B-Spline construction are used over higher dimension domains?

In this thesis I develop a geometric generalization of B-Spline curves over the triangular domain. This construction mimics the control point blending that occurs with uniform B-Splines. The construction preserves the simple control scheme and evaluation of B-Splines, without the immense computational requirements of DMS-splines. The result is a new patch control scheme, the G-Patch, possessing C^0 continuity between adjacent patches.

Acknowledgements

This research has been partially funded through a scholarship from the Natural Sciences and Engineering Research Council of Canada.

I would like to thank my family for their tireless support over the last year. I would also like to thank my supervisor, Stephen Mann, for both introducing me to the field of CAGD and helping my research take shape. Additional thanks to my readers, Richard Bartels and Bruce Simpson for their valuable feedback.

Finally, a thank-you to "Sesame Street" for teaching me about triangles. I never fully appreciated their complexities at the time, but a journey must begin somewhere.

Contents

List of Figures

Chapter 1

Introduction

B-Splines are a great design tool. You pull a control point, and the nearby region of the curve bulges in that direction without affecting the smoothness of the curve. B-Splines exhibit virtually every desirable property for modelling univariate curves. This fact alone is why they are so universally used in two dimensional curve modelling. A testament to their versatility is that True Type fonts[1] are built using quadratic B-Splines to shape the outline of each character.

It is reasonable to hope that such a powerful technique should generalize to higher dimensions such as surfaces. In fact, one such method exists, and that is to form the tensor-product of B-Splines in two parametric directions. The result is a rectangular shaped B-Spline surface. These surfaces can be found in any 3-D API in the form of Non-Uniform Rational B-Splines (NURBS).

The inherent rectilinear shape of tensor-product B-Splines severely limits their use in modelling arbitrary shaped surfaces. Thus, it is felt that a triangular B-Spline surface is a more natural generalization. In 1987 Ramshaw discussed trian-

[1]Developed by Apple

gular B-Splines, and put forth what he describes as his "juiciest" challenge: find a natural way to "blossom" a triangular-patch surface that builds in the appropriate continuity conditions, similar to what is done with the B-Spline [14].

Over the last fifteen years, there has been a lot of research on triangular spline surfaces. Each has its own specialized use, but inevitably each has its own fundamental limits. What is fascinating is that among this large body of research, there is not a single scheme that can be declared the true generalization of the B-Spline. Most implementations are either computationally expensive, possess vague "shape" parameters, or both.

The current state of the art remains (arguably) DMS-splines, developed in 1992 by Dahmen, Micchelli and Seidel [2]. Their construction produces surfaces where neighbouring triangular patches in the surface meet with high degrees of continuity. This continuity does not come cheaply, however. They are difficult to model with and require unreasonable levels of computation for even low degree surfaces.

One of the primary problems with DMS-splines and other proposed solutions is they do not emulate the simple blending techniques used to determine points in a B-Spline. I hypothesize that if a true generalization exists, it should resemble the B-Spline construction. This is the idea that has motivated my research: how can the simple formulas used to blend B-Spline control points be scaled to a higher dimension, and what are the properties of the resulting surfaces?

My work has resulted in a new patch construction algorithm, the G-Patch, based solely on mimicking the geometry performed in the B-Spline evaluation. The algorithm is computationally efficient, and the resulting surfaces can be manipulated just as B-Spline curves can. The scheme is far from ideal, though, as the construction can only guarantee C^0 continuity. That is, neighbouring triangular patches

meet along their edges, but do not necessarily meet smoothly.

1.1 Overview

This thesis is organized in the following manner.

Chapter 2 discusses the mathematical foundations of relevant curve and surface constructions. This background material will also introduce the mathematical notation used throughout the rest of the thesis.

Chapter 3 introduces the theory behind the new triangular G-Patch scheme. The B-Spline construction will be examined from a geometric standpoint, and then it is generalized to higher dimension domains.

Chapters 4 through 6 establish the connection between the new G-Patch and a classic triangular patch known as the Bézier patch. Chapter 4 provides a meaningful labelling to the G-Patch control points reminiscent of those used for B-Splines. Chapter 5 goes on to show how to convert between the G-Patch and the Bézier patch. Then in Chapter 6, I demonstrate how to construct a larger surface from a collection of G-Patches.

Chapter 7 proceeds to analyze the new scheme and compare the results against the DMS-spline. In particular the continuity exhibited by G-Patch surfaces is explored.

Finally, in Chapter 8, I draw conclusions about the new construction, and I give direction for future research.

Chapter 2

Background

In this chapter I will review the basics behind various control schemes used in the construction of curves and surface patches in geometric design. This chapter also introduces my notation.

2.1 Desirable Control Scheme Attributes

Before looking at the specific details of different constructions, I first discuss some of the key attributes that are considered essential to a good control scheme. Neamtu gives an excellent discussion of necessary requirements for a multi-variate spline construction [12]. In short, the control scheme should be simple, automatically smooth, and computationally feasible over large control point networks. The following section gives further details of the desirable properties.

Piecewise Polynomial of a Fixed Degree

A large, complex curve or surface consists of a collection of smaller curves or patches of a specified degree. While representing the entire surface by a single patch is possible, it would need to be of extremely high degree to model anything complex. Constructing the entire curve or surface becomes a simple matter of generating the individual pieces and rendering them together.

Individual Piecewise Polynomials are Associated to Regions of the Domain

Very simply, the domain of the surface can be divided into small regions, and there is a single piecewise polynomial defined for each particular region.

Control Points

The shape of individual polynomials is specified by a fixed number of points which are used as coefficients in the polynomial functions. By moving the control points the user is able to change the shape of the polynomial.

Local Control

Manipulating the control points should only influence a finite region of the entire curve and surface. While neighbouring regions are likely to be effected by this manipulation, the extent of the effect should be restricted to a subset of the piecewise polynomials. More importantly, as the number of piecewise polynomials for the surface increases, the number of polynomials effected by a control point movement should not increase.

Automatic Continuity Maintenance

Given an arbitrary collection of control points, the corresponding piecewise polynomials should meet with some level of continuity automatically. The construction should not be conditional on the particular placement of the control points (such as certain points being coplanar) for this continuity to be achieved.

Interactivity

Ultimately, the constructions need to be run on real hardware, and likely will be manipulated by real people. If moving a single control point results in a significant time lag to render the altered surface, it limits the ability to incorporate the technique in real applications. Ideally, a scheme should allow for reasonably complex surfaces to be manipulated in real time by a human modeler. Generally, the greater mathematical complexity of the construction, the less interactive its implementation will be.

Simplifies to Univariate Splines

When the control scheme is used over the domain $\mathbb{R}^1$ the classic univariate splines should be generated.

Numerical Stability

The evaluation of points on the curve or surface should not suffer from any numerical stability issues. A small deviation in the original control points should not result in large movements in the points on the resulting polynomial [1]. The evaluation

should be stable for all choices of the original control points, such as the control points being near to each other or sets of control points being collinear.

Intuitive User Interface

Altering the position of a control point or knot should result in the curve or surface changing in a natural manner. For example, pulling a control point away from the surface should pull nearby regions of the surface in that direction. The placement of any knots should have a logical effect on the resulting curve or surface. By far the most subjective attribute, it is often the feature that dictates its incorporation into real applications.

Most of these desired properties result from the construction having the convex hull property. Namely, any point on the surface can be expressed as the weighted average of a neighbourhood of control points, where the weighting functions are non-negative and sum to one.

2.2 Mathematical Terminology

2.2.1 Barycentric Coordinates

There are numerous ways of representing a point in space. One possible way is to perform a linear combination of a collection of points in that space. This leads to a "coordinate-free" system, as it does not depend on any particular coordinate reference frame to specify the points. This notion of blending points is part of a bigger field of mathematics known as affine geometry. For a more formal treatment refer to any introductory text on CAGD [6, 9, 10].

Given a set W of n points, p_i, we can blend them to specify a new point, u, by weighting each point by a different scalar multiple $\beta_i \in \mathbb{R}$:

$$u = \sum_{i=1}^{n} \beta_i p_i.$$

In general, this point blending only has geometric meaning if the weights of the control points sum to one:[1]

$$\sum_{i=1}^{n} \beta_i = 1.$$

Typically the number of points in W is one more than the dimension of the space the points reside in.

If none of the points p_i can be represented as a linear combination of the remaining points, the collection of points form a *simplex*. When this occurs, the above blending is said to be a *barycentric combination* and the β_i values are known as the *barycentric coordinates* of u with respect to W.

A barycentric combination in which all the β_i values are non-negative is referred to as a *convex combination*. This type of blending ensures that the resulting point is inside the convex hull of W.

To demonstrate barycentric coordinates, a line can be described as a barycentric combination of two unique points, a and b. A point u on the line can be represented by

$$\begin{aligned} u &= \beta_1 a + \beta_2 b \\ &= \beta_1 a + (1 - \beta_1) b. \end{aligned}$$

The last line follows from the barycentric coordinates summing to one. The entire line is generated by allowing β_1 to vary from $-\infty$ to ∞. The original point a results

[1] There is another geometric meaning if the weights sum to zero; it represents a vector.

from setting $\beta_1 = 1$ and $\beta_2 = 0$ and b results from setting $\beta_1 = 0$ and $\beta_2 = 1$. Given a, b and u, the two barycentric coordinates are given by the following formulae:

$$\begin{aligned} \beta_1 &= \tfrac{b-u}{b-a} \\ \beta_2 &= \tfrac{u-a}{b-a} \end{aligned} \tag{2.1}$$

Barycentric coordinates can also be used to define a plane. Given three non-collinear points (a, b, c), any point u in the plane can be represented as a barycentric combination of the three points. The three barycentric coordinates are given by applying Cramer's rule:

$$\begin{aligned} \beta_1 &= \tfrac{\text{area}(\triangle ubc)}{\text{area}(\triangle abc)} \\ \beta_2 &= \tfrac{\text{area}(\triangle uac)}{\text{area}(\triangle abc)} \\ \beta_3 &= \tfrac{\text{area}(\triangle uab)}{\text{area}(\triangle abc)} \end{aligned} \tag{2.2}$$

As a convention, when referring to a point u in a known domain region, we will directly use its barycentric coordinates relative to the extreme points of the domain region.

2.2.2 Blossoming

Blossoms [14] or polar forms [5] are one representation for piecewise polynomials that have barycentric coordinates as its foundation. This representation is in a parametric form. Ramshaw gives the following theorem about blossoming [14]:

Theorem 1 (The Blossoming Principle) *Let F be a degree n polynomial which maps a point u in a d-dimensional domain to a point in $\mathbb{R}^k$. There exists a unique map $f : (\mathbb{R}^d)^n \to \mathbb{R}^k$ such that:*

- *f is symmetric* — $f(\ldots, u_i, \ldots, u_j, \ldots) = (\ldots, u_j, \ldots, u_i, \ldots)$
- *f is multi-affine* — $f(\ldots, \sum \beta_i u_i, \ldots) = \sum \beta_i f(\ldots, u_i, \ldots)$ *when* $\sum \beta_i = 1$
- *f is diagonal* — $f(u, \ldots, u) = F(u)$

f is said to be the multi-affine blossom of F. f evaluated at a particular set of parameters is a blossom value. The specific parameters are the blossom arguments.

As an example, consider a function $F(u) = 3u^2 + 2u + 1$. The unique blossom of F is $f(u_1, u_2) = 3u_1u_2 + u_1 + u_2 + 1$.

The multiaffine property shows the connection to barycentric coordinates, allowing us to take barycentric combinations of two blossom values to form a new blossom value.

2.2.3 Continuity

We are often interested in determining the amount of smoothness throughout a curve a surface. Blossoming provides a means of defining parametric continuity for a polynomial.

A polynomial is trivially C^∞ everywhere. A piecewise polynomial is C^∞ everywhere except at the endpoints between the individual polynomials. We consider two adjacent polynomials defined by the functions F_1 and F_2 with u being the parameter at their junction. If F_1 and F_2 are discontinuous, they are said to meet with C^{-1}-continuity at u. If the two polynomials agree at u, but do not meet smoothly, then they meet with C^0-continuity. C^k-continuity is then defined

$$F_1^{(i)}(u) = F_2^{(i)}(u) \quad \text{for } i = 0 \ldots k.$$

We can also define continuity in terms of blossoming. The two polynomials meet with C^k-continuity if for arbitrary choices of $u_1 \dots u_k$

$$f_1(u_1, \dots, u_k, \underbrace{u, \dots, u}_{n-k}) = f_2(u_1, \dots, u_k, \underbrace{u, \dots, u}_{n-k})$$

2.3 Parametric Curves

We will consider intervals of the one-dimensional domain to define a curve. For each point u in the interval, we consider its image $F(u)$. The resulting curve is generated by varying u throughout the domain interval.

2.3.1 Bézier Curves

A degree n Bézier curve is defined by $n+1$ control points, $P_0 \dots P_n$. The image of a point u with barycentric coordinates β_1 and β_2 relative to a domain interval $[a, b] : \ a, b \in \mathbb{R}$ is

$$F(u) = \sum_{i=0}^{n} P_i B_i^n(u), \tag{2.3}$$

where each $B_i^n(u)$ is one of the degree n Bernstein polynomials

$$B_i^n(u) = \binom{n}{i} {\beta_1}^{n-i} {\beta_2}^{i}. \tag{2.4}$$

We can use blossoming to give a more geometric evaluation of the curve. The original control points P_i are given by particular values of the blossom of the Bézier curve:

$$P_i = f(\underbrace{a, \dots, a}_{n-i}, \underbrace{b, \dots, b}_{i}). \tag{2.5}$$

When writing the control points, each point is often labelled with its blossom value.[2] Figure 2.1 shows the four labelled control points of a cubic Bézier curve. Notice that adjacent control points agree in $n-1$ of the blossom arguments. Using the multi-affine property of blossoming, neighbouring control points can be blended using u's barycentric coordinates. For instance $f(a,a,b)$ and $f(a,b,b)$ can be combined to yield a new point

$$f(u,a,b) = \beta_1 f(a,a,b) + \beta_2 f(a,b,b). \tag{2.6}$$

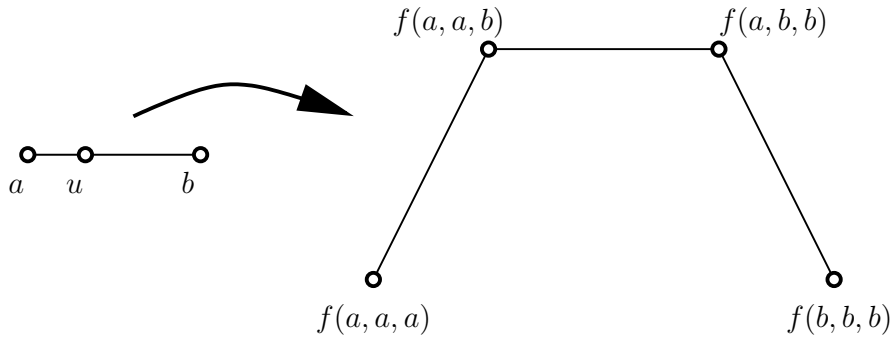

Figure 2.1: Blossom labels for the control polygon of a cubic Bézier curve defined over the domain interval $[a,b]$.

By repeatedly blending neighbouring control points, we can determine the point of the curve $F(u)$. This leads to the de Casteljau algorithm. The general form for the blending performed at each step is given by the following relation where $i+j+k = n-1$:

$$\begin{aligned} f(\underbrace{u,\ldots,u}_{i+1},\underbrace{a,\ldots,a}_{j},\underbrace{b,\ldots,b}_{k}) \;=\; & \beta_1 f(\underbrace{u,\ldots,u}_{i},\underbrace{a,\ldots,a}_{j+1},\underbrace{b,\ldots,b}_{k}) + \\ & \beta_2 f(\underbrace{u,\ldots,u}_{i},\underbrace{a,\ldots,a}_{j},\underbrace{b,\ldots,b}_{k+1}) \end{aligned} \tag{2.7}$$

[2]This label will sometimes be called the "blossom label" of the point.

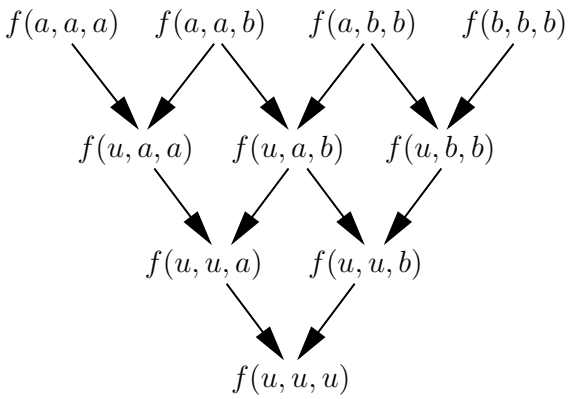

Figure 2.2: Data flow diagram for evaluating a cubic Bézier curve.

Figure 2.2 shows how the original control points for the Bézier curve are repeatedly combined to generate the final point on the curve. Figure 2.3 shows the de Casteljau algorithm being run on the original control points. Note that the ratio of the length of the line segments $\overline{au} : \overline{ub}$ is preserved for each blend.

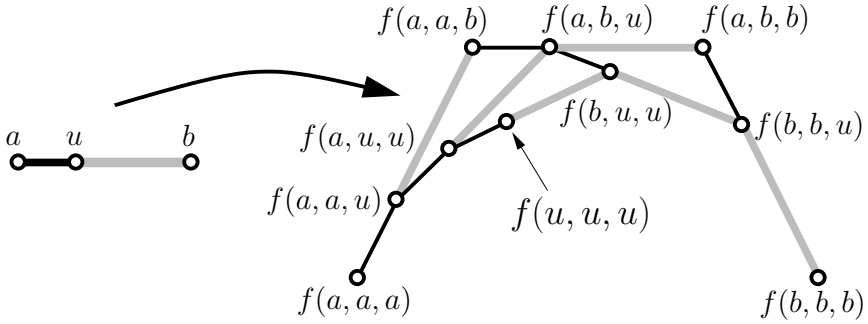

Figure 2.3: Running the de Casteljau algorithm on a cubic Bézier curve. For each blend, the ratio of the lengths between $\overline{au}$ (shown in black) and $\overline{ub}$ (shown in grey) is used.

The Bézier curve is a wonderful tool with a simple control scheme and efficient evaluation. However, modelling a sufficiently complex curve requires the Bézier

curve to have an extremely high degree. The curve also does not possess local control since moving a single control point affects the entire curve. A simple solution to the problem is to divide the domain into smaller regions, and define a Bézier curve for each region. The only problem is ensuring that adjacent Bézier curves meet smoothly.

2.3.2 B-Splines

The B-Spline is a construction that generalizes Bézier curves. A degree n B-Spline is a piecewise polynomial curve in which adjacent curve segments may meet with up to C^{n-1}-continuity. The degree of continuity is controlled by the user, and is provided automatically by the construction. Neighbouring regions of the curve also share most of their control points.

For a B-Spline the domain is partitioned into an increasing sequence of scalar values, $t_i \in \mathbb{R}$, called the *knot vector*.[3] For a degree n curve, the i'th control point P_i represents a special blossom value using knots as blossom arguments: $f(t_i, t_{i+1}, \ldots, t_{i+n-1})$. The i'th curve segment is defined over the domain interval $[t_{i+n-1}, t_{i+n}]$ and is constructed using the $n+1$ control points $P_i \ldots P_{i+n}$.

The image of a point u in this interval is defined by

$$F(u) = \sum_{i=0}^{n} P_i N_i^n(u),$$

where $N_i^n(u)$ is the i'th B-Spline basis function of degree n over the given knot vector. The basis function can be evaluated using the recurrence relation

$$\begin{aligned} N_i^n(u) &= \frac{u - t_i}{t_{i+n} - t_i} N_i^{n-1}(u) + \frac{t_{i+m+1} - u}{t_{i+m+1} - t_{i+1}} N_{i+1}^{m-1}(u) \\ N_0^0(u) &= 1 \end{aligned}$$

[3]Note that i, the index of the knot, is an integer, but the knot itself is a real number.

If the denominator is zero as a result of coincident knots, the term is defined to be zero. Note each B-Spline basis function is non-zero over only a finite interval, thus providing local control.

As with the Bézier curve, using the blossom values provide a more geometric evaluation for a point on the curve. If we consider evaluating a point on the i'th curve segment of a cubic B-Spline, the four control points that define the segment have blossom values $f(t_i, t_{i+1}, t_{i+2})$, $f(t_{i+1}, t_{i+2}, t_{i+3})$, $f(t_{i+2}, t_{i+3}, t_{i+4})$ and $f(t_{i+3}, t_{i+4}, t_{i+5})$. Neighbouring control points agree in all but one of the blossom arguments. If we take u's barycentric coordinates relative to the disagreeing arguments, we can blend a new point using the following relation:

$$\begin{aligned} f(\underbrace{u, \ldots, u}_{j}, t_{i+j}, \ldots, t_{i+n-j}) &= \beta_1 f(\underbrace{u, \ldots, u}_{j-1}, t_{i+j-1}, \ldots, t_{i+n-j}) + \\ &\quad \beta_2 f(\underbrace{u, \ldots, u}_{j-1}, t_{i+j}, \ldots, t_{i+n-j+1}) \end{aligned} \tag{2.8}$$

By repeatedly blending points in this fashion we get the de Boor algorithm [4]. Figure 2.4 shows how the original control points for the B-Spline curve are repeatedly blended to generate the final point on the curve.

One segment of the B-Spline curve is simply an alternative representation of a Bézier curve. One could easily derive the particular Bézier control points for the curve segment by inserting the appropriate knots using the de Boor algorithm. As mentioned earlier, the primary advantage of the B-Spline representation is that continuity between neighbouring segments is provided automatically.

The knot vector controls the tightness of the curves near the associated control points, while still supporting the continuity conditions. Should lower continuity

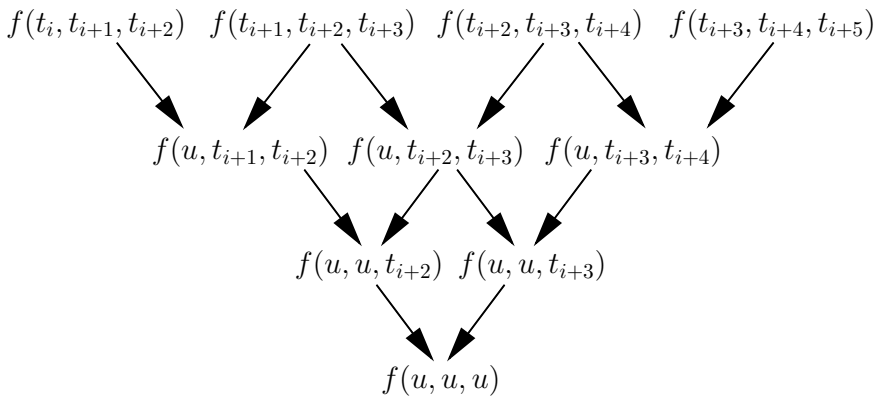

Figure 2.4: Data flow diagram for evaluating a cubic B-Spline curve.

be desired, allowing consecutive knots to have the same scalar value decreases the continuity between the two curves that meet at that place in the domain.

B-Splines possess all of the desired properties listed at the start of the chapter. Along with automatically maintaining a collection of smoothly joining piecewise polynomial curves, they have a simple evaluation, provide local control and most importantly, are intuitive to use.

2.4 Parametric Surfaces

The parametric polynomial curve can be generalized to surfaces quite naturally. We will consider intervals of the two-dimensional domain to define a patch. For each point u in the interval, we consider its image $F(u)$. The resulting surface patch is generated by varying u throughout the area of the domain interval.

2.4.1 Tensor Product Surfaces

A simple patch scheme is to divide the domain into two parametric directions u and v as in Figure 2.5. A parametric curve is defined for each of the two parametric directions. By holding either the u or v parameter constant, and allowing the other parameter to vary through the domain, a curve is generated in that parametric direction. By allowing both parameters to vary, a surface is defined.

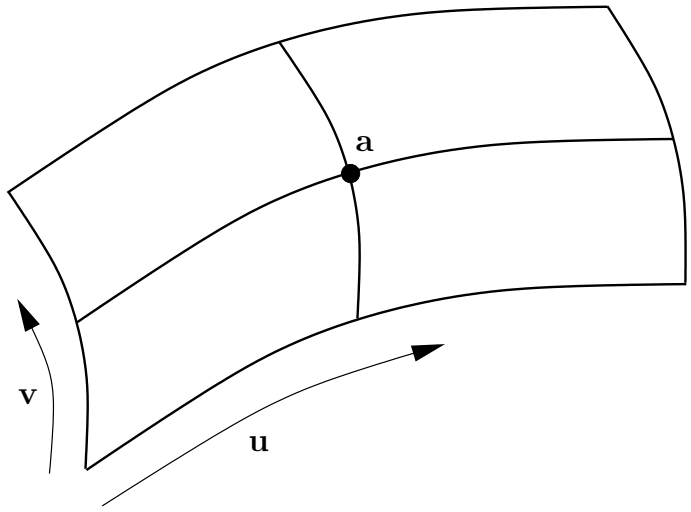

Figure 2.5: Four tensor product patches defined over two parametric directions u and v. All four patches share a common vertex a.

For instance, we can define a tensor product of a degree m and a degree n Bézier curve. The resulting surface is given by

$$F(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{i,j} B_i^n(u) B_j^m(v). \tag{2.9}$$

Optionally, B-Spline curves can be used in each of the parametric directions which yield tensor product B-Splines.

Tensor product B-Splines can be efficiently evaluated using a slight modification of the de Boor algorithm. They also inherit most of the desirable properties of univariate B-Splines such as local control and an intuitive control scheme. The only drawback is that the surfaces are inherently rectangular in shape. Neighbouring patches must also meet at a common corner vertex to join continuously as seen in Figure 2.5. If the domain cannot be naturally partitioned into quadrilaterals, tensor product B-Splines are not the appropriate modelling tools. A trimming curves approach can adapt the technique to more geometries, but there is no general technique for joining two trimmed surfaces [6].

2.4.2 Triangular Bézier Patches

A more natural way to partition the domain is into triangular regions. More surface geometries can be tiled with triangles than can be tiled with quadrilaterals. This allows the creation of arbitrarily shaped surfaces. The Bézier curve formulation can be generalized to triangular shaped surface patches.

A degree n Bézier patch is defined using a triangular layout of $\binom{n+2}{n}$ control points: $P_{i,j,k}$: $i,\ j,\ k \geq 0$ and $i+j+k=n$. We will consider an interval in the domain defined by the triangle $\triangle abc:\ a,b,c \in \mathbb{R}$. The image of a point u with barycentric coordinates β_1, β_2 and β_3 is generated by

$$F(u) = \sum_{\substack{i,\ j,\ k\geq 0 \\ i+j+k=n}} P_{i,j,k} B^n_{i,j,k}(u) \tag{2.10}$$

where each $B^n_{i,j,k}(u)$ is one of the two-dimensional degree n Bernstein polynomials given by

$$B^n_{i,j,k}(u) = \binom{n}{i,j,k} {\beta_1}^i {\beta_2}^j {\beta_3}^k \tag{2.11}$$

where $\binom{n}{i,j,k}$ is the generalized binomial coefficient defined as

$$\binom{n}{i,j,k} = \frac{n!}{i!j!k!}.$$

As with the curve case, we can use blossoming to give a geometric evaluation of the curve. The control point $P_{i,j,k}$ has a blossom value of

$$P_i = f(\underbrace{a,\ldots,a}_{i},\underbrace{b,\ldots,b}_{j},\underbrace{c,\ldots,c}_{k}). \tag{2.12}$$

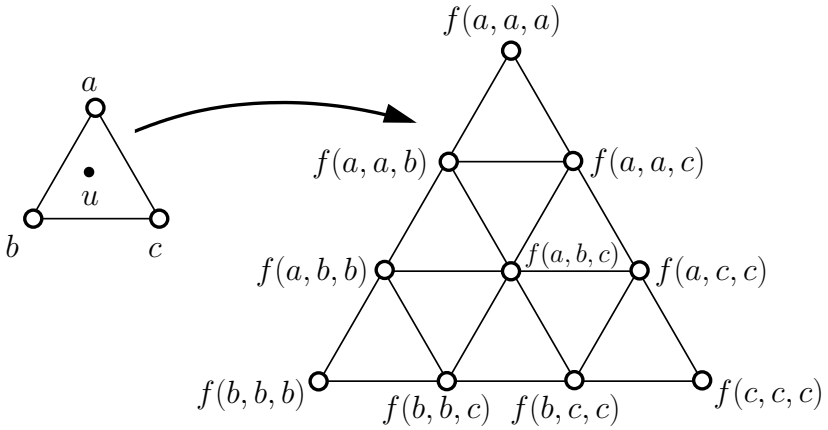

Figure 2.6: Blossom values for the control polygon of a cubic Bézier patch defined over the domain triangle $\triangle abc$.

Figure 2.6 shows the labelled control points for a cubic Bézier patch. Three neighbouring control points that form an upward pointing triangle agree in $n-1$ of their blossom arguments. Using the multi-affine property of blossoming, three neighbouring control points can be blended using u's barycentric coordinates relative to $\triangle abc$. For instance $f(a,a,b)$, $f(a,b,b)$ and $f(a,b,c)$ can be combined to yield a new point

$$f(u,a,b) = \beta_1 f(a,a,b) + \beta_2 f(a,b,b) + \beta_3 f(a,b,c). \tag{2.13}$$

By repeatedly blending triples of neighbouring control points, we can determine the point on the patch $F(u)$. This is identical to the de Casteljau algorithm used to generate points on a Bézier curve. Figure 2.7 shows the de Casteljau algorithm being run on a quadratic Bézier patch.

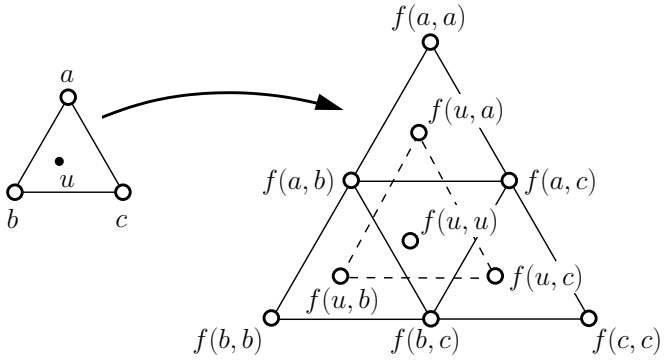

Figure 2.7: Running the de Casteljau algorithm on a quadratic Bézier patch.

The Bézier patch inherits many of the properties of the Bézier curve. Most important is that modelling sufficiently complex surfaces requires the Bézier patch to have an extremely high degree. This is easily solved by dividing the domain into smaller triangular regions, and defining a Bézier patch for each region. However, it is a non-trivial task to devise a simple control scheme in which the neighbouring patches automatically meet with some degree of continuity.

2.4.3 Constraints on Triangular Patch Continuity

The B-Spline solved the problem of automatically managing a collection of Bézier curves such that they met with C^{n-1}-continuity. It seems only natural to desire the same level of continuity for a patch construction scheme. However, Ramshaw [14] and Gallier [9] prove a bleak property for a network of Bézier triangles.

Theorem 2 (Triangular Bézier Patch Continuity Constraints) *For a surface consisting of degree $n \geq 1$ triangular Bézier patches the highest degree of continuity possible, while still providing local flexibility, is C^k-continuity, where $k < \frac{2n-1}{3}$.*

This implies that a surface of cubic Bézier patches cannot possess more than C^1 continuity. If C^2-continuity is desired, the surface must necessarily be built from quartic patches.

Despite such a strong claim, the proof gives absolutely no insight into how to construct a network of patches that possess this level of continuity. This is what led Ramshaw to propose his challenge problem discussed in the introduction; a problem that remains open to this day.

2.4.4 B-Patches

One way of looking at a region of the B-Spline is to imagine knots being "pulled out" of a Bézier curve. For a region of the domain $[a, b]$ each Bézier control point was defined by

$$f(\underbrace{a, \ldots, a}_{n-i}, \underbrace{b, \ldots, b}_{i}).$$

For the B-Spline curve representation each instance of a and b is labelled separately giving

$$f(a_{n-i-1}, \ldots, a_0, b_0, \ldots, b_{i-1}).$$

Each of the a_j and b_j values represents a different knot on the domain line such that $a = a_0 \geq a_1 \geq \ldots \geq a_{n-i-1}$ and $b = b_0 \leq b_1 \leq \ldots \leq b_{i-1}$.

The B-Patch attempts to generalize this idea by "pulling knots" out of the corner of the domain triangle defined for a Bézier patch. The collection of knots

corresponding to each corner is referred to as a *knot cloud*. For a region of the domain $\triangle abc$ Figure 2.8 shows a possible configuration of the knot clouds.

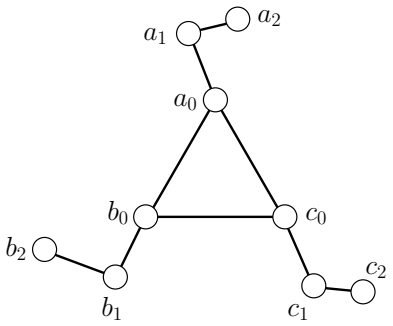

Figure 2.8: A cubic B-Patch domain region.

A Bézier patch control point with blossom label

$$f(\underbrace{a,\ldots,a}_{i},\underbrace{b,\ldots,b}_{j},\underbrace{c,\ldots,c}_{k})$$

is now relabelled as

$$f(a_0,\ldots,a_{i-1},b,\ldots,b_{j-1},c,\ldots,c_{k-1}).$$

Figure 2.9 shows the blossom values for a cubic B-Patch.

Note that three neighbouring points forming an upward pointing triangle agree in $n-1$ blossom arguments. This leads to a de Boor style algorithm to evaluate a point u in the domain. If we express u in barycentric coordinates relative to the remaining blossom arguments, we can blend the three points. Figure 2.10 shows the blending of the top three control points. The three points are weighted by u's barycentric coordinates in terms of the domain triangle $\triangle a_2 b_0 c_0$ to yield a new point $f(b_0, b_1, u)$.

Proceeding with the remainder of the upward pointing triangles, a total of six new control points are generated and are given in Figure 2.11. Once again,

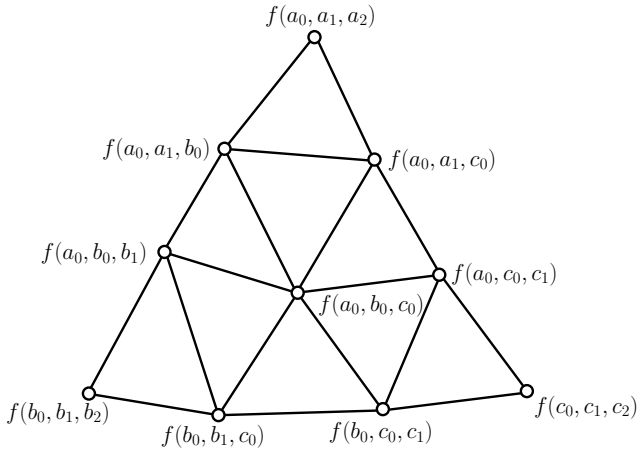

Figure 2.9: A labelled cubic B-Patch control network.

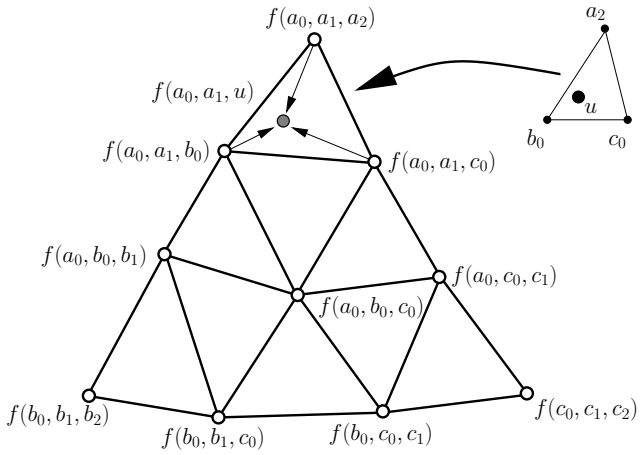

Figure 2.10: Blending three B-Patch control points.

neighbouring control points differ in only one argument, so the algorithm can be repeated. This procedure continues until the point $f(u, u, u)$ is derived.

At this point it seems that an appropriate generalization of B-Spline curves has emerged. However, the evaluation of a single B-Patch does not extend well to a network of patches. Consider just two adjacent domain regions that share a common

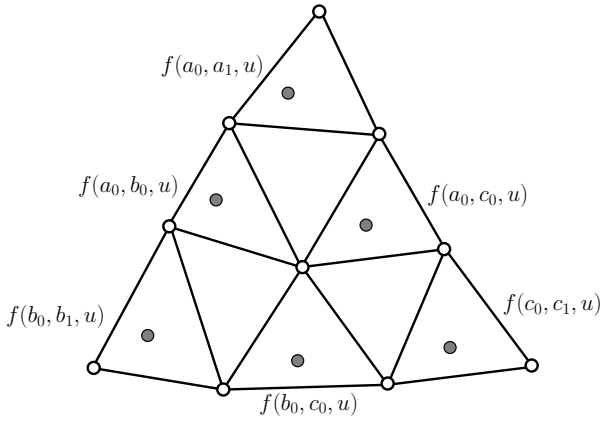

Figure 2.11: All the level one blending B-Patch points.

edge. We would like to reuse the knot clouds for the corners shared by the domain regions. We would also like the control polygons of the two corresponding patches to share control points. Figure 2.12 shows a knot and control point configuration which allows the two patches to meet with some degree of continuity. The key feature to note is that the knots along the shared domain edge are collinear.

Seidel proved that this collinearity is a necessary condition for C^0 continuity to take place [15]. This makes arbitrary surfaces impossible to construct. The collinearity requirement between all the adjacent domain triangles forces the knot clouds to collapse back to their original locations at the corner of each domain region. At this point the B-Patches degenerate to simple Bézier patches, and nothing has been gained.

2.4.5 Simplex Splines

The major problem with B-Patches is that the underlying basis functions do not automatically provide the required degrees of continuity. The simplex spline, on

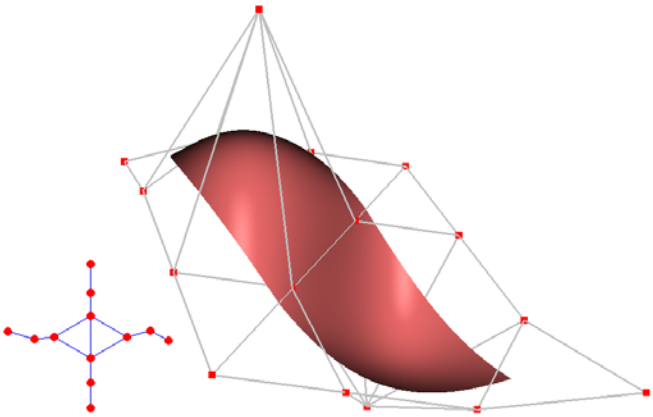

Figure 2.12: A screen capture of two cubic B-Patches meeting with C^0-continuity. The two patches share the control points along their common edge. The domain with the knot clouds is shown in the bottom left corner.

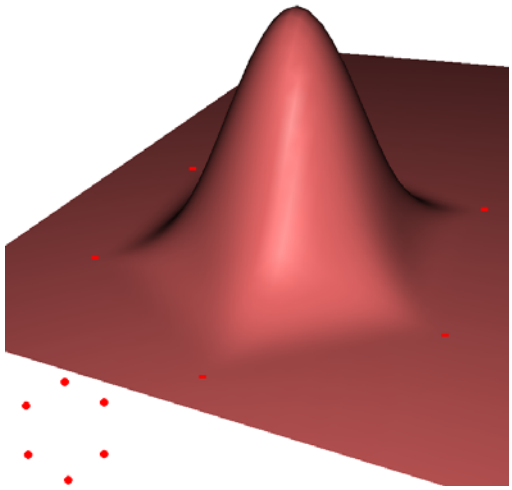

Figure 2.13: A cubic simplex spline that is C^2-continuous everywhere. The six knots in the domain defining the spline are shown in the bottom left corner.

the other hand, is a function that does.

The simplex spline is a degree n spline defined over $n+3$ points (knots) in the domain. A simplex spline exhibits C^{n-1} continuity everywhere unless three or more knots are collinear, in which case the continuity drops along that line.

We will now define the evaluation of the simplex spline over a set of knots V at a point u. The point on the spline is given by $F(u) = M(u|V)$.

If we triangulate the domain over V, any point on a line connecting two points is within the convex hull of both triangular regions sharing that edge. The half-open convex hull is a subset of the convex hull defined such that any point in the domain is assigned membership to exactly one region [8].

The simplex spline is defined recursively. Consider the base case of evaluating a point u for a simplex spline defined by a set of three knots $V = \{t_0, t_1, t_2\}$.

If u is outside the half-open convex hull defined by $V = \{t_0, t_1, t_2\}$, then

$$M(u|V) = 0$$

otherwise

$$M(u|V) = \frac{1}{\text{Area } \triangle t_0 t_1 t_2}.$$

Higher degree simplex splines of degree n are defined over the set of $n+2$ knots $V = \{t_0, \ldots, t_{n+1}\}$. An arbitrary set of three points is selected from V to form a set $W = \{t_a, t_b, t_c\}$. If β_1, β_2 and β_3 are u's barycentric coordinates with respect to W, then

$$M(u|V) = \beta_1 M(u|V \backslash \{t_a\}) + \beta_2 M(u|V \backslash \{t_b\}) + \beta_3 M(u|V \backslash \{t_c\}).$$

Figure 2.13 shows a cubic simplex defined over a hexagonal shaped knot configuration.

The simplex spline provides the necessary continuity conditions such that they can be used as basis functions. There are some drawbacks in the simplex spline evaluation that are worth mentioning. The choice of the knots to place in W during each recursive evaluation can effect the results of the computation if not chosen carefully. The evaluation is also plagued with numerical stability issues. Finally, evaluating a single point on the surface is computationally expensive.

2.4.6 DMS-Splines

The current state of the art remains DMS-splines jointly developed in 1992 by Dahmen, Micchelli and Seidel [2]. The technique attempts to merge the nice labelling of the control points found in B-Patches with the smooth basis functions found in simplex splines. The result is a patch construction scheme that yields a C^{n-1} continuous surface.

The setup is similar to B-Patches. The domain is triangulated, and with each corner of the domain a knot cloud is arranged. For a degree n triangular patch, n knots are pulled out of each corner of the domain triangles. Note that this is one more knot than was needed for a B-Patch. Figure 2.14 shows the knot clouds defined for a quadratic DMS spline. A triangular patch defined for the domain region $\triangle abc$ is specified using $\binom{n+2}{n}$ control points $P_{i,j,k}$: $i,\ j,\ k \geq 0$ and $i + j + k = n$.

To incorporate the ideas of the Simplex spline, we define a set

$$V_{i,j,k} = \{a_0, \ldots, a_i, b_0, \ldots, b_j, c_0, \ldots, c_k\}.$$

This spline is not normalized, so we define a normalization factor given by

$$d_{i,j,k} = \text{area } \triangle a_i b_j c_k.$$

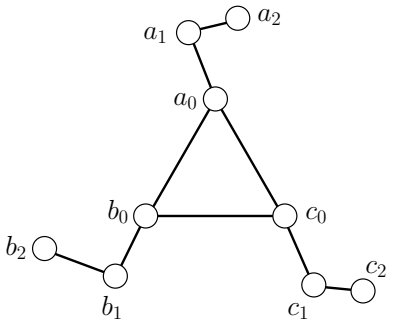

Figure 2.14: The labelled knot clouds for a quadratic DMS patch.

To evaluate a point u on the triangular patch, each control point is weighted by the Simplex spline corresponding to the set above. Figure 2.15 shows a quadratic simplex spline that is weighting one of the control points.

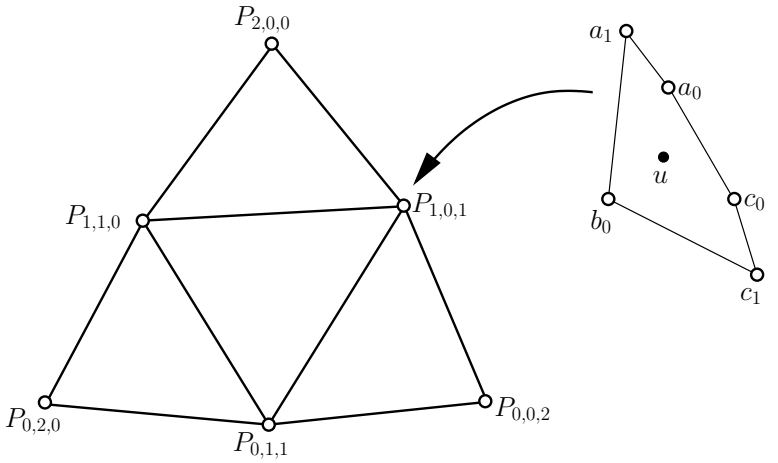

Figure 2.15: A Simplex spline weighting a quadratic DMS-spline control point.

This yields the following formula to evaluate a point on the surface:

$$F(u) = \sum_{i+j+k=n} P_{i,j,k} d_{i,j,k} M(u|V_{i,j,k}) \tag{2.14}$$

A screen capture of a single DMS patch is shown in Figure 2.16.

For a network of DMS patches, the above evaluation is not quite complete. Control points from neighbouring patches may contribute to the evaluation of a point on the surface. This is because the simplex splines weighting each control point can evaluate to non-zero values outside the domain triangle the patch is defined over. Figure 2.17 shows the result of adding a neighbouring patch to Figure 2.16. The original patch has changed shape and no longer curls back on itself along its upper left edge. Instead it is extended so that it blends smoothly into the new patch.

The DMS construction can yield C^{n-1} continuous surfaces, but does the scheme solve Ramshaw's original problem? Unfortunately, a DMS patch does not correlate to the Bézier patch, so it is not actually solving the problem. This fact can be excused if it manages to provide the desirable control scheme attributes discussed at the start of the chapter. Regrettably, the control scheme is far from ideal. The most noticeable problem comes from the computational cost of evaluating the surface. There does not yet exist a nice coefficient based evaluation for the DMS patch. In calculating a point on the surface numerous simplex splines must be explicitly evaluated, which (as previously mentioned) is an expensive operation to perform. As well, it is not easy to determine which neighbouring control points will factor into the evaluation. Since simplex splines are explicitly calculated, the DMS spline evaluation inherits the numerical stability issues occurring in simplex splines.

Computational issues aside, the DMS control scheme does not present an elegant user interface. While manipulating the control points should not pose a problem, the placement of the knot clouds presents an enormous challenge to the user. Moving the knots has unexpected results, and it is an enormous burden trying to prevent too many knots from becoming collinear. It is also not known if there is a "good" way to place the knots automatically for the user.

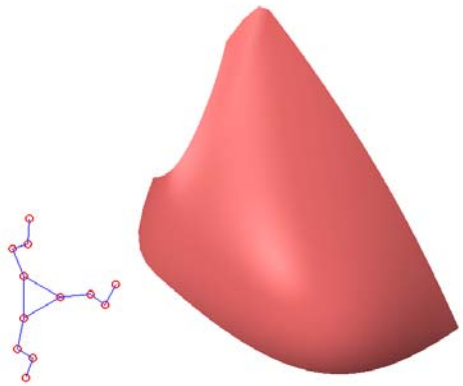

Figure 2.16: A cubic DMS patch. The knot clouds in the domain for the patch are shown in the bottom left corner.

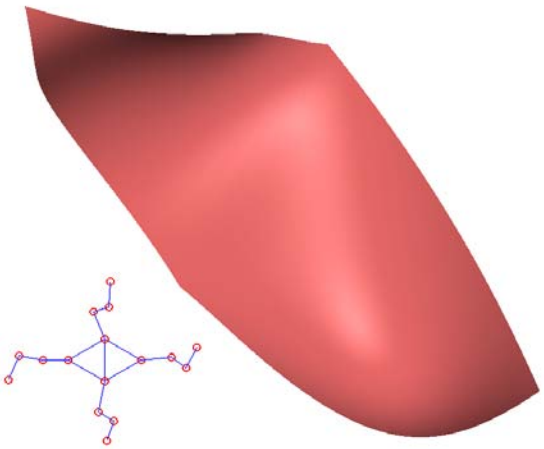

Figure 2.17: Two cubic DMS patches. The knot clouds in the domain for the patches are shown in the bottom left corner.

Chapter 3

A Novel Triangular Patch Scheme

I will introduce a new triangular patch construction scheme, G-Patches. Their construction attempts to generalize the geometry of a uniform B-Spline curve over higher dimensional domains.

Most generalizations of higher order B-Splines attempt to find some method of attributing knots to each of the control points in the surface. The result is that the simple knot vector of the low order splines becomes a knot cloud in the higher order surfaces. However, I propose that before providing the flexibility of free moving knots, it is worth studying how to create a generalization of the blending functions used in the uniform B-Spline. Thus far, no one has provided a direct higher order analogy for the uniform case. Tackling this problem on its own is worthy of investigation, and may provide the clues to developing the true generalization that is so actively being sought.

3.1 Uniform B-Spline Geometry

The general de Boor evaluation for a point u over an arbitrary knot vector requires calculating u's barycentric coordinates with respect to many different pairs of points. Assuming that all the knots are evenly spaced simplifies the computations required to generate u's image. Once we deduce how to perform the geometry over one piecewise polynomial, we can duplicate the same construction through each region of the domain. At this point the knot vector can then be discarded altogether.

Consider the region $[t_a, t_b]$ of the domain defining one segment of the piecewise polynomial. In a uniform, degree n B-Spline, if t_a and t_b are spaced k units apart, then the relevant knot vector consists of $2n$ knots $\{t_{a-n+1}, \ldots, t_a, t_b, \ldots, t_{b+n-1}\}$ all spaced k units apart. These knots are used as blossom arguments for the $n+1$ control points defining this region of the curve. We will consider u's image when u is in the range $t_a \leq u \leq t_b$.

3.1.1 Degree One B-Splines

Consider the simple case where we have a degree one spline. Here, the knot vector is $\{t_a, t_b\}$. If we look at the blossom values for the two control points, P_0 and P_1, we see that the first point is labelled $f(t_a)$, while the other is labelled $f(t_b)$. To find $F(u)$, we simply determine u's barycentric coordinates with respect to t_a and t_b and use those coordinates to blend the two control points as shown in Figure 3.1.

Using Equation 2.1 we get u's barycentric coordinates satisfying the relation

$$u = \beta_1 t_a + \beta_2 t_b, \qquad \beta_1 + \beta_2 = 1,$$

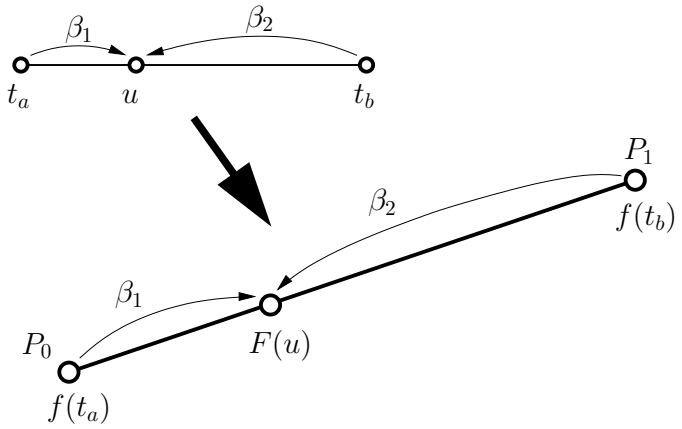

Figure 3.1: Blending the two control points in a degree one B-Spline. β_1 and β_2 are used to weight the two control points.

and the image of u is consequently

$$F(u) = \beta_1 P_0 + \beta_2 P_1.$$

3.1.2 Degree Two B-Splines

Now that we have defined the base case, we consider quadratic B-Splines. Again, u is restricted to vary from t_a to t_b. For this case, the knot vector is $\{t_{a-1}, t_a, t_b, t_{b+1}\}$ and there are three control points, P_0, P_1, P_2, with blossom values $f(t_{a-1}, t_a)$, $f(t_a, t_b)$ and $f(t_b, t_{b+1})$ respectively.

To evaluate the curve, we perform affine combinations of neighbouring control points. Notice that neighbouring control points have all but one blossom argument in common so we find the barycentric coordinates of u relative to the blossom argument unique to each point. Thus, for the first two control points, we find the barycentric coordinates of u in terms of t_{a-1} and t_b.

It is important to take a look at where the image of u will be on this line: it always lies in the half of the line nearest the middle control point. Figure 3.2 highlights these valid locations on the line in grey. Manually calculating the barycentric coordinates of u for this step is not difficult, but it would be nice if we could determine the new coordinates automatically. As it turns out, if we know the barycentric coordinates of u relative to t_a and t_b, then we can directly convert these values to the new coordinates needed for this step.

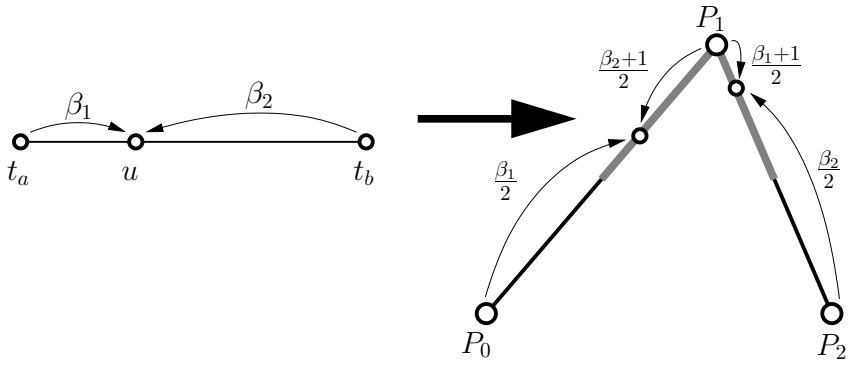

Figure 3.2: Blending the three control points in a quadratic B-Spline. The weighting of each control point is given along the arcs. The grey region of each line segment indicates possible locations of the new points after the first level blending.

Using the barycentric coordinates β_1 and β_2 for u, we map these to new barycentric coordinates, β_1' and β_2', defined relative to t_{a-1} and t_b. Using the uniformity of knot placement, we find that

$$\begin{aligned} u &= \beta_1' t_{a-1} + \beta_2' t_b \\ &= \frac{\beta_1}{2} t_{a-1} + \frac{\beta_2 + 1}{2} t_b. \end{aligned}$$

Thus we can blend the two control points to get an image on the line between them

using

$$f(u, t_a) = \frac{\beta_1}{2} P_0 + \frac{\beta_2 + 1}{2} P_1. \tag{3.1}$$

Symmetrically, we can blend the last two control points using a similar formula:

$$f(u, t_b) = \frac{\beta_1 + 1}{2} P_1 + \frac{\beta_2}{2} P_2. \tag{3.2}$$

Figure 3.2 shows the blending functions used to generate these new points.

Finally, we are left with the task of blending the two new points $f(u, t_a)$ and $f(u, t_b)$ to generate the point on the curve. Since the two points agree in all but one blossom argument, we can again apply the blossoming principle. We combine the points using u's barycentric coordinates relative to the arguments they disagree on. This calculation is identical to that used for the degree one B-Spline, therefore, we combine the points with the original barycentric coordinates of u, β_1 and β_2. Thus,

$$f(u, u) = \beta_1 f(u, t_a) + \beta_2 f(u, t_b).$$

3.1.3 Degree Three B-Splines

It is worth looking at the degree three B-Spline construction before generalizing the technique. The set of relevant knots increases by two yielding the knot vector $\{t_{a-2}, t_{a-1}, t_a, t_b, t_{b+1}, t_{b+2}\}$. There are four controls $P_0, \ldots, P_3$, with each point having three arguments in its blossom label as given in Figure 3.3.

Again, we blend neighbouring control points, and again the points differ in only one blossom argument. We need to find the barycentric coordinates of u relative to the differing blossom argument. Thus for the first two control points, we must find the barycentric coordinates of u in terms of t_{a-2} and t_b.

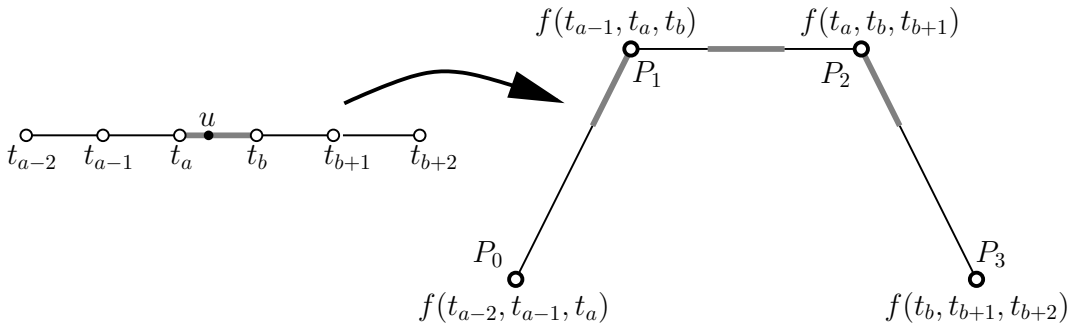

Figure 3.3: The cubic B-Spline control polygon and knot vector. The grey region of the knot vector indicates all the possible values of u. The grey region of each line segment in the control polygon indicates the locations of the new points after the first level blending.

This time, we notice that u's image will always lie in the third of the line closest to the second point. Figure 3.3 highlights in grey the valid locations of u's image on all three line segments of the control polygon. The technique of automatically converting the barycentric coordinates, β_1 and β_2 to the new coordinates β_1' and β_2' can again be performed by taking advantage of the uniformity of the knot vector. We reach a slightly different formula:

$$u = \frac{\beta_1}{3} t_{a-2} + \frac{\beta_2 + 2}{3} t_b.$$

Thus we can blend the two control points to get u's image on the line between them with

$$f(u, t_{a-1}, t_a) = \frac{\beta_1}{3} P_0 + \frac{\beta_2 + 2}{3} P_1.$$

Similarly, we can blend the last two control points by the symmetric formula:

$$f(u, t_b, t_{b+1}) = \frac{\beta_1 + 2}{3} P_2 + \frac{\beta_2}{3} P_3.$$

Finally we need to consider blending the middle two control points. If we look

at the image of u, we see it always lies in the middle third of the line. Similar to above, with a uniform knot vector the barycentric coordinates of u relative to t_{a-1} and t_{b+1} are given by

$$f(u, t_a, t_b) = \frac{\beta_1 + 1}{3} P_1 + \frac{\beta_2 + 1}{3} P_2.$$

At this point we are left with three new points with which to blend: $f(u, t_{a-1}, t_a)$, $f(u, t_a, t_b)$ and $f(u, t_b, t_{b+1})$. These blossom arguments are identical to the quadratic case, with an additional u argument. Therefore, we combine these points using the formulas from the previous section.

3.1.4 Curve Construction Algorithm

Let us examine all the blending functions used to combine the control points for a cubic B-Spline curve (Figure 3.4). The four original control points are at the top of the figure, and the point on the curve is at the bottom. This is the same as Figure 2.4, but with the various values of β_1' and β_2' used to blend points given along the edges of the diagram. β_1' always appears along edges flowing down and to the right, while β_2' always appears along edges flowing down and to the left.

At this point we see a pattern emerging in the values generated for our formulas. This pattern makes it easy to construct a dynamic program to calculate points for a degree n uniform B-Spline. We can automatically calculate the particular value of β_1' and β_2' needed to blend any two points, based on its location in the triangle. We will label the intermediate points in the blending function triangle using two indices. The first index, deg, indicates what row the point is from. All of the points in that row can be interpreted as control points for a curve of degree deg. The second index, i, indicates which of the control points in the row is being referenced

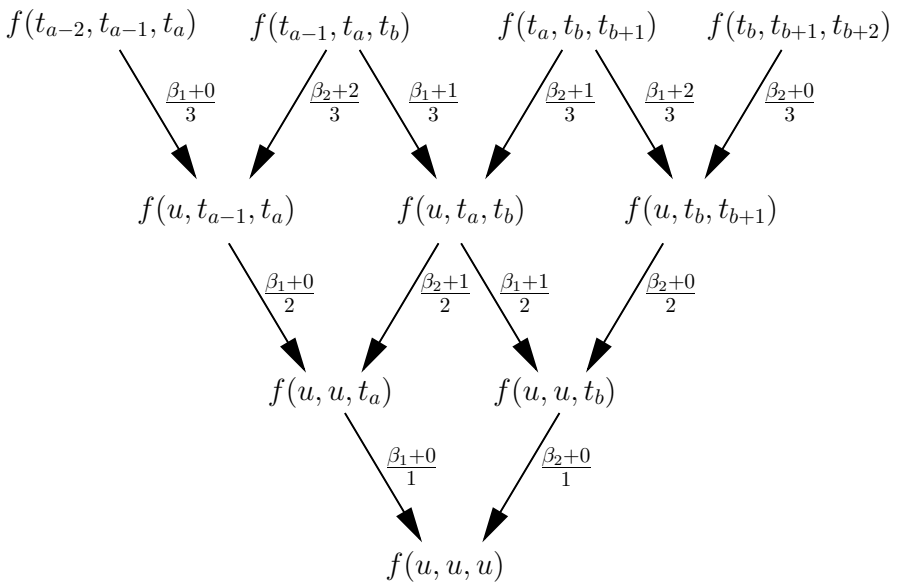

Figure 3.4: Cubic B-Spline recursive blending functions.

(Figure 3.5). The edges now show the blending function used to combine control points from level *deg*.

The ith pair of blending functions for combining points on the deg'th level is given by

$$\begin{aligned} \beta_1' &= \frac{\beta_1 + i}{deg} \\ \beta_2' &= \frac{\beta_2 + deg - i - 1}{deg} \end{aligned} \tag{3.3}$$

This represents the general form for blending the degree deg uniform B-Spline control points. At this point the knot vector has been eliminated from the calculations entirely.

Before continuing, we should verify that each pair of blending functions sum to unity as barycentric coordinates should:

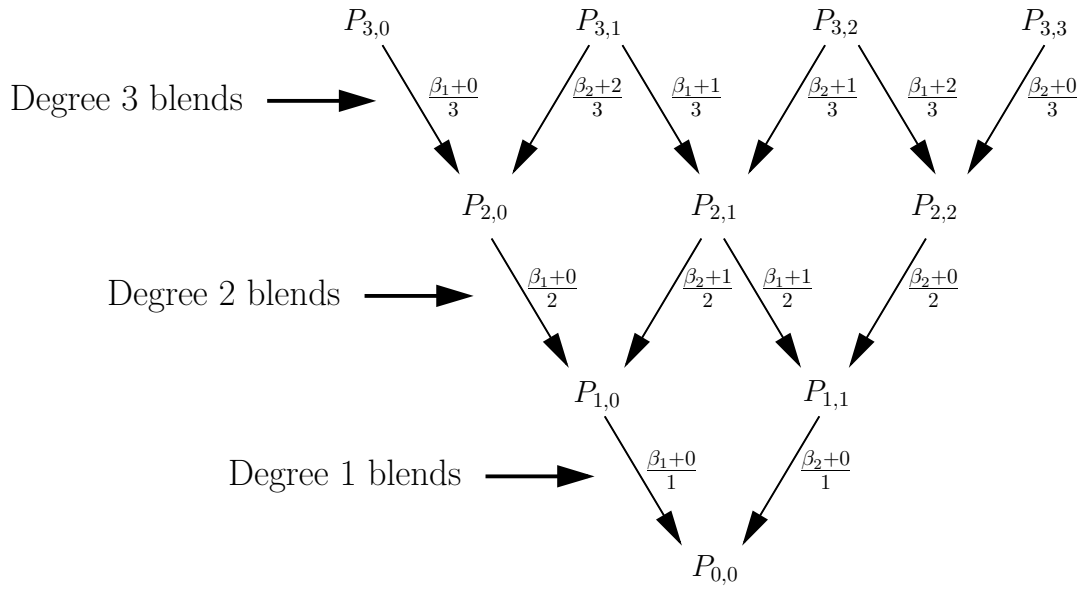

Figure 3.5: Indexing the intermediate points and blending functions.

$$\begin{aligned}
\beta'_1 + \beta'_2 &= \frac{\beta_1 + i}{deg} + \frac{\beta_2 + deg - i - 1}{deg} \\
&= \frac{\beta_1 + \beta_2 + i - i + deg - 1}{deg} \\
&= \frac{1 + deg - 1}{deg} \\
&= \frac{deg}{deg} \\
&= 1 \qquad \text{for } deg \neq 0
\end{aligned}$$

This leads to an efficient algorithm to calculate a point on the curve. It is essentially the de Boor algorithm with precomputed blending functions, and is not meant to be considered a new algorithm, simply a revisitation of the classical methodology. For this algorithm, we have u, the point in the domain, passed in as its barycentric coordinates relative to t_a and t_b.

Algorithm UNIFORMBLOSSOM(β_1, β_2, $P[\,]$):

Input: u's barycentric coordinates and, P, an array of n control points

Output: $F(u)$, the point on the curve

1: **for** deg **from** $n-1$ **to** 1 **do**

2: **for** i **from** 0 **to** $deg-1$ **do**

3: $P[i] \leftarrow \frac{\beta_1+i}{deg} \cdot P[i] + \frac{\beta_2+deg-i-1}{deg} \cdot P[i+1]$

4: **return** $P[0]$

Like the de Boor algorithm, this dynamic programming algorithm has an $O(n^2)$ running time and requires only $O(n)$ space. The difference, though, is that it does not perform explicit calculations using an underlying knot vector.

3.2 The Triangular Domain

Having simplified the construction for curves it is time to scale this up to surfaces defined over a triangular domain. At the heart of B-Splines are Bézier curves. Bézier curves are defined over a similar domain as the B-Splines, but without the concept of knots. As indicated in the previous chapter, the knots and reuse of control points are what give B-Splines their continuity properties for neighbouring curve segments. Also discussed in the previous chapter, Bézier patches are the generalization of Bézier curves to surfaces. The domain of a Bézier patch is triangular, and I would like to use the same domain in my generalization of the B-Splines.

For the moment I will focus on one piecewise polynomial patch in the domain. This piece of the domain is represented by an equilateral triangle with corners labelled t_a, t_b and t_c. We are concerned with mapping a point u inside this domain triangle to a point on the patch. We represent u with barycentric coordinates β_1,

β_2 and β_3 such that

$$u = \beta_1 t_a + \beta_2 t_b + \beta_3 t_c, \qquad \beta_1 + \beta_2 + \beta_3 = 1.$$

3.3 Uniform Triangular G-Patches

I construct the degree *deg* patches (hereafter referred to as G-Patches) out of a control net of $\binom{deg+2}{2}$ control points laid out in a triangular grid. As with Bézier patches, we will repeatedly blend three neighbouring points that form an upward pointing triangle. We label the points with a double index, P_{i_1,i_2}, where i_1 is the row, and i_2 is the particular point in that row. It is noted that i_2 is always less than or equal to i_1, and indexing starts with 0. Figure 3.6 shows the control points for a cubic patch.

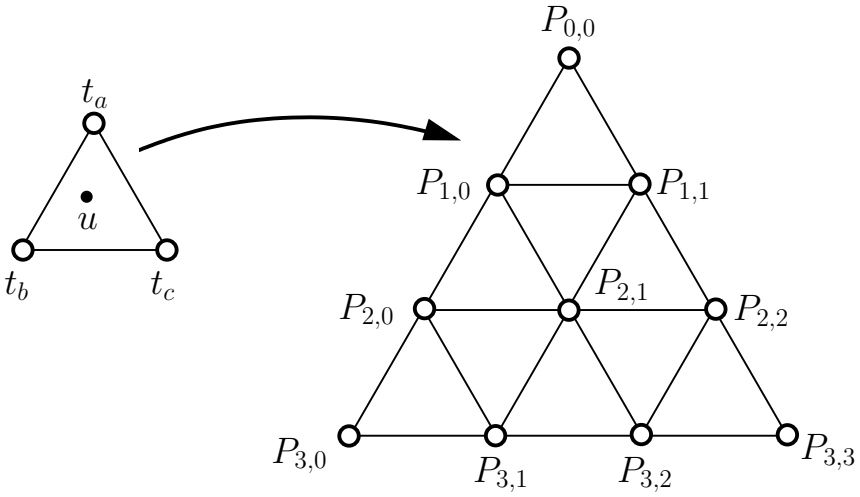

Figure 3.6: Indexing the control points for triangular patches.

For the remainder of this chapter I will refer to the control points using this indexing. At this stage we will not concern ourselves with the notion of knots; we

will just focus on developing the new blending functions. This means we do not yet have blossom values to associate with the control points. In Chapter 4 a more geometric labelling for the control points will be derived.

All that remains is to scale the derivations from Section 3.1.4 to handle the new triangular domain. As before, we will be converting β_1, β_2 and β_3 to the appropriate, new, barycentric coordinates, and using these to blend the neighbouring points.

3.3.1 Degree One G-Patches

Degree one G-Patches have three control points associated with them: $P_{0,0}$, $P_{1,0}$ and $P_{1,1}$. Here we want the patch to encompass the entire triangle just as the degree one B-Spline curve covers the entire line between the two control points.

For the degree one curve, we saw the values of β_1', β_2' were simply β_1 and β_2, so it seems appropriate to use the same general formula. Thus, our new barycentric coordinates will be

$$\beta_1' = \frac{\beta_1 + 0}{1}, \qquad \beta_2' = \frac{\beta_2 + 0}{1}, \qquad \beta_3' = \frac{\beta_3 + 0}{1}. \tag{3.4}$$

The inclusion of the denominator and addition by zero will become apparent when looking at higher degree patches.

Blending the three control points with these barycentric coordinates will yield the image of u on the patch as shown in Figure 3.7.

3.3.2 Degree Two G-Patches

Degree two G-Patches have six control points $P_{0,0}, \ldots, P_{2,2}$. There are three different upward pointing triangles, resulting in three different blending functions. For the

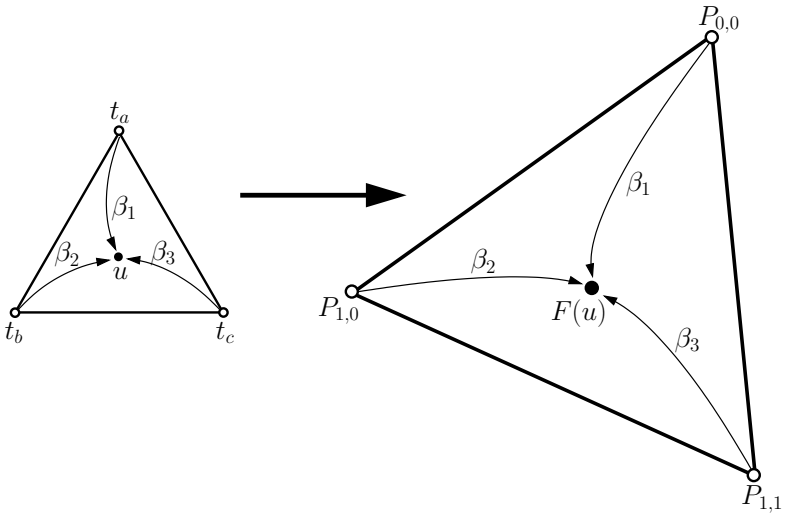

Figure 3.7: Blending the three control points in a linear G-Patch. β_1, β_2 and β_3 are used to weight the control points.

purpose of this discussion, we will refer to blending the top three control points as blend A, the bottom left three control points as blend B and the bottom right control points as blend C as shown in Figure 3.8.

For the degree two B-Splines, the blending functions mapped u onto the inner half of the points being combined. For the G-Patch, we will devise a similar blending function that maps u to regions that are located nearer the center of the control net. Thus, the barycentric coordinates used to blend A should result in the image laying in the bottom middle of the three points. Figure 3.9 highlights in grey the target regions for each blending function.

Looking back at Equation 3.1 and Equation 3.2 for the degree two B-Spline, we see that β_1' and β_2' take the form

$$\beta_1' = \frac{\beta_1 + 0}{2}, \qquad \beta_2' = \frac{\beta_2 + 1}{2}$$

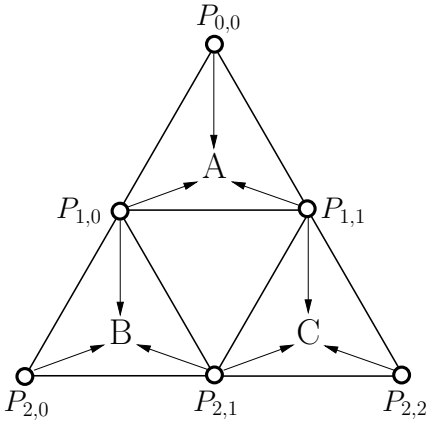

Figure 3.8: The degree two convex hull, and the corresponding control point blends.

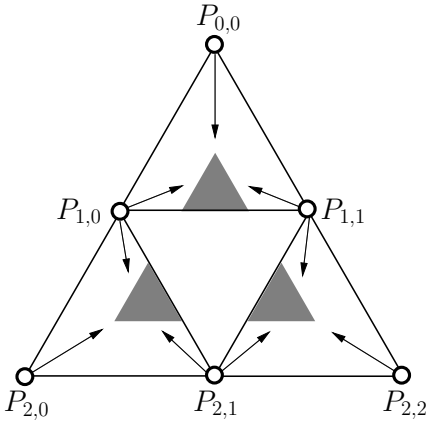

Figure 3.9: Valid regions for the first level blending of quadratic G-Patch control points. The image of u for all three blends will fall within the shaded regions.

and

$$\beta_1' = \frac{\beta_1 + 1}{2}, \qquad \beta_2' = \frac{\beta_2 + 0}{2}.$$

When blending the two leftmost points, β_1' has a "+0" term while the other barycentric coordinate has a "+1" term. When blending the rightmost pair, β_2' has

the "+0" term while the other barycentric coordinate has the "+1" term.

Using the same idea for the G-Patch, when blending the points for A, β_1' will have a "+0" term and all the other barycentric coordinates will have "+1" terms. Geometrically, these new blending functions represent barycentric combinations of points defined over a much larger domain triangle $\triangle t_a' t_b' t_c'$ shown in Figure 3.10.

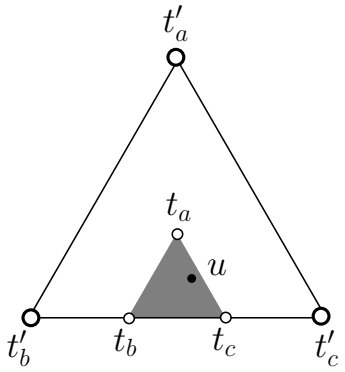

Figure 3.10: The geometric meaning of the new blending functions.

Similarly, when blending the points for B, β_2' will be the barycentric coordinate with the "+0" term, and when blending the points for C, β_3' will have the "+0" term. Finally, the denominator for each barycentric coordinate needs to be set to three so that the new barycentric coordinates sum to unity.

Summarizing, this gives three sets of blending functions for the upward pointing triangles A, B and C. The new barycentric coordinates for each are given by

$$\begin{aligned}
\textbf{A:}\quad & \beta_1' = \frac{\beta_1 + 0}{3}, \qquad \beta_2' = \frac{\beta_2 + 1}{3}, \qquad \beta_3' = \frac{\beta_3 + 1}{3} \\
\textbf{B:}\quad & \beta_1' = \frac{\beta_1 + 1}{3}, \qquad \beta_2' = \frac{\beta_2 + 0}{3}, \qquad \beta_3' = \frac{\beta_3 + 1}{3} \\
\textbf{C:}\quad & \beta_1' = \frac{\beta_1 + 1}{3}, \qquad \beta_2' = \frac{\beta_2 + 1}{3}, \qquad \beta_3' = \frac{\beta_3 + 0}{3}.
\end{aligned} \tag{3.5}$$

This yields three new points, which are in turn blended using the same evaluation as the degree one G-Patch.

3.3.3 Degree Three G-Patches

We will look at the degree three G-Patch before generalizing the new blending functions. These patches have ten control points $P_{0,0}, \ldots, P_{3,3}$ resulting in six different upward pointing triangles, each requiring its own blending function. Again, to aid in discussion, we will label the blending of neighbouring control points with the letters A through F as in Figure 3.11.

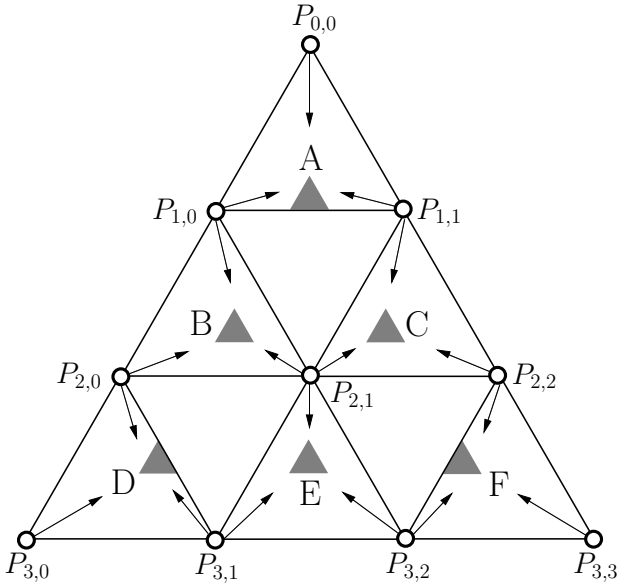

Figure 3.11: The cubic G-Patch convex hull, and the corresponding control point blends. The valid region for each blend is highlighted in grey.

For the cubic B-Splines we saw that the first blend took the form

$$\beta_1' = \frac{\beta_1 + 0}{3}, \qquad \beta_2' = \frac{\beta_2 + 2}{3}.$$

Again, β_1' has a "+0" term, but now the β_2' formula has a "+2" term.

Using this idea, the corner blends for the G-Patch should have one barycentric coordinate with a "+0" term, while the remaining two coordinates should have a "+2" term. The resulting blending functions require a denominator of 5 so that the barycentric coordinates sum to unity. This gives the following new barycentric coordinates for the corner blends:

$$\begin{aligned}
\textbf{A:}\quad & \beta_1' = \frac{\beta_1 + 0}{5}, \qquad \beta_2' = \frac{\beta_2 + 2}{5}, \qquad \beta_3' = \frac{\beta_3 + 2}{5} \\
\textbf{D:}\quad & \beta_1' = \frac{\beta_1 + 2}{5}, \qquad \beta_2' = \frac{\beta_2 + 0}{5}, \qquad \beta_3' = \frac{\beta_3 + 2}{5} \\
\textbf{F:}\quad & \beta_1' = \frac{\beta_1 + 2}{5}, \qquad \beta_2' = \frac{\beta_2 + 2}{5}, \qquad \beta_3' = \frac{\beta_3 + 0}{5}.
\end{aligned} \tag{3.6}$$

Figure 3.11 highlights in grey the valid regions for each blending function. Notice that the size of the regions are smaller than those in the quadratic case. This is due to a larger denominator in the expression.

Finally, the middle three blends need to be accounted for. Let us examine the B blend, first. If we examine the two corner blends on either side (A and D), we see that they agree on the β_3' values, so we can use this same value in B's blend. If we look at the numerators of A and D's β_1' and β_2', they look identical to those used in the degree three B-Spline blending functions. It follows that we should choose a β_1' and β_2' value for B's blend that looks like the value used in the B-Spline case, namely the numerator for β_1' and β_2' should both have a "+1" term. Applying this idea to all three middle blends, we get the following new barycentric coordinates:

$$\begin{aligned}
\textbf{B:}\quad & \beta_1' = \frac{\beta_1 + 1}{5}, \qquad \beta_2' = \frac{\beta_2 + 1}{5}, \qquad \beta_3' = \frac{\beta_3 + 2}{5} \\
\textbf{C:}\quad & \beta_1' = \frac{\beta_1 + 1}{5}, \qquad \beta_2' = \frac{\beta_2 + 2}{5}, \qquad \beta_3' = \frac{\beta_3 + 1}{5} \\
\textbf{E:}\quad & \beta_1' = \frac{\beta_1 + 2}{5}, \qquad \beta_2' = \frac{\beta_2 + 1}{5}, \qquad \beta_3' = \frac{\beta_3 + 1}{5}.
\end{aligned} \tag{3.7}$$

This gives us six new points that are recursively blended using the formulas given for the quadratic G-Patch.

3.3.4 Patch Construction Algorithm

Looking at the formulas given in Equations 3.4, 3.5, 3.6 and 3.7 a pattern once again emerges in the blending functions. By specifying the degree (deg) of the G-Patch, we can automatically calculate the particular value of β'_1, β'_2 and β'_3 needed to generate point P_{i_1,i_2} using u's original barycentric coordinates.

$$\begin{aligned} \beta'_1 &= \frac{\beta_1 + i_1}{2\,deg - 1} \\ \beta'_2 &= \frac{\beta_2 + deg + i_2 - i_1 - 1}{2\,deg - 1} \\ \beta'_3 &= \frac{\beta_3 + deg - i_2 - 1}{2\,deg - 1} \end{aligned} \tag{3.8}$$

Again, we should verify that these blending functions sum to unity as barycentric coordinates should:

$$\begin{aligned} \beta'_1 + \beta'_2 + \beta'_3 &= \frac{\beta_1 + i_1}{2\,deg - 1} + \frac{\beta_2 + deg + i_2 - i_1 - 1}{2\,deg - 1} + \frac{\beta_3 + deg - i_2 - 1}{2\,deg - 1} \\ &= \frac{\beta_1 + \beta_2 + \beta_3 + i_1 - i_1 + i_2 - i_2 + deg + deg - 1 - 1}{2\,deg - 1} \\ &= \frac{1 + 2\,deg - 2}{2\,deg - 1} \\ &= \frac{2\,deg - 1}{2\,deg - 1} \\ &= 1 \end{aligned}$$

Having a closed form for the blending functions to use at each step gives us another efficient dynamic programming algorithm to calculate a point on the surface of the G-Patch given the patch's control net. For this algorithm, we have u, the

point in the domain, passed in by its barycentric coordinates relative to the patch's domain triangle, and an $n \times n$ array of control points for the patch. We only use the lower triangular half of the array, since the control net is triangular.

Algorithm UNIFORMG-PATCH(β_1, β_2, β_3, $P[\,][\,]$):

Input: u's barycentric coordinates and, P, an $n \times n$ array of control points

Output: $F(t)$, the point on the surface

1: **for** deg **from** $n-1$ **to** 1 **do**
2: **for** i_1 **from** 0 **to** $deg-1$ **do**
3: **for** i_2 **from** 0 **to** i_1 **do**
4: $\beta'_1 \leftarrow \frac{\beta_1+i_1}{2\,deg-1}$
5: $\beta'_2 \leftarrow \frac{\beta_2+deg+i_2-i_1-1}{2\,deg-1}$
6: $\beta'_3 \leftarrow \frac{\beta_3+deg-i_2-1}{2\,deg-1}$
7: $P[i_1][i_2] \leftarrow \beta'_1 \cdot P[i_1][i_2] + \beta'_2 \cdot P[i_1+1][i_2] + \beta'_3 \cdot P[i_1+1][i_2+1]$
8: **return** $P[0][0]$

This dynamic programming algorithm has an $O(n^3)$ running time, and requires $O(n^2)$ space.

3.4 Higher Dimension Constructions

So far we looked at one and two dimensional domains. The equations given for the particular blending functions in Equations 3.3 and 3.8 are starting to exhibit a pattern. If we were to follow the same analysis for the three dimensional domain (volumes), we would derive the following formulas for the new barycentric

coordinates:

$$\begin{aligned}
\beta'_1 &= \frac{\beta_1 + i_1}{3\,deg - 2} \\
\beta'_2 &= \frac{\beta_2 + deg + i_2 - i_1 - 1}{3\,deg - 2} \\
\beta'_3 &= \frac{\beta_3 + deg + i_3 - i_2 - 1}{3\,deg - 2} \\
\beta'_4 &= \frac{\beta_4 + deg - i_3 - 1}{3\,deg - 2}
\end{aligned} \tag{3.9}$$

At this point it is possible to completely generalize the blending function for a construction of arbitrary dimension and degree.

If we have a *dim*-dimensional domain, then a point u in the domain will be represented by $dim + 1$ barycentric coordinates, $\beta_1 \ldots \beta_{dim+1}$. These barycentric coordinates will be relative to $dim + 1$ points, $t_1 \ldots t_{dim+1}$ which form an affine basis.

If we are dealing with a degree *deg* surface, then the control net will consist of $\binom{deg+dim}{2}$ control points and we will label our points with *dim* index values, $P_{i_1,\ldots,i_{dim}}$.

Using u's original barycentric coordinates, we can automatically calculate the particular values of $\beta'_1 \ldots \beta'_{dim+1}$ needed to generate the point $Pi_1, \ldots, i_{dim}$ of the given degree.

$$\begin{aligned}
\beta'_1 &= \frac{\beta_1 + i_1}{dim(deg-1)+1} \\
\beta'_{dim+1} &= \frac{\beta_{dim+1} + deg - i_{dim} - 1}{dim(deg-1)+1} \\
\beta'_j &= \frac{\beta_j + deg + i_j - i_{j-1} - 1}{dim(deg-1)+1} \qquad \text{for } 1 < j < dim + 1
\end{aligned} \tag{3.10}$$

Chapter 4

Control Point Labelling

Having derived blending equations to generalize B-Splines to higher dimensional domains, the next task is to look at providing labels for the original G-Patch control points in a manner reminiscent of the B-Spline. B-Spline curves reuse the labels of the control points for neighbouring curve segments, so a good G-Patch scheme should also allow neighbouring patches to share control points and their labels.

In this chapter the original B-Spline control point labelling provided by Ramshaw will be introduced. This notation captures the geometry of the B-Spline blending function as well as showing how neighbouring curve segments can use many of the same control points. The notation will then be extended to the G-Patch control points providing the geometric meaning behind the G-Patch blending functions. The resulting control point labelling will provide the means in Chapter 6 to extend the single G-Patch evaluation to an entire surface made up of a collection of G-Patches.

4.1 B-Spline Labelling

Typically B-Spline control points are given using a blossom value $f(a, b, c) : a, b, c \in \mathbb{R}$. For a uniform cubic B-Spline, the blossom arguments for the control point are a fixed distance, k, apart; for example $f(3, 4, 5)$ or $f(10, 20, 30)$. What is often overlooked is that the blossom value for a control point only has meaning inside a particular interval of the domain. Namely, it is only valid if one is evaluating a point u in the domain where u is between $a - k$ and $c + k$ (again, assuming the uniform knot vector). So $f(3, 4, 5)$ is valid in the evaluation of $F(2.5)$, but it has no meaning when considering $F(7.3)$.

In the original blossoming paper [14], Ramshaw explicitly associates a "validity interval" with each B-Spline blossom value to reinforce which region of the domain the point is defined. This interval is put as a subscript before the blossom arguments. Thus, the control point $f(3, 4, 5)$ would be more accurately written as $f_{\{2,6\}}(3, 4, 5)$, indicating that for $2 \leq u \leq 6$ the point represents the blossom value $f(3, 4, 5)$. Figure 4.1 shows the evaluation of two points in different regions of a uniform quadratic B-Spline. Note that the farthest right control point is not used in the evaluation of $F(2.5)$ since it is outside the validity interval defined for it.

4.1.1 The Domain

Up until this point we labelled points in the one dimensional domain with a single value u where u is allowed to vary from zero to the size of the domain (assuming the knots at the ends have full multiplicity). Choosing the leftmost point in the domain to start from is arbitrary, and one can just as easily start labelling from the other direction. This is a result of the knot vector having two parametric directions. I

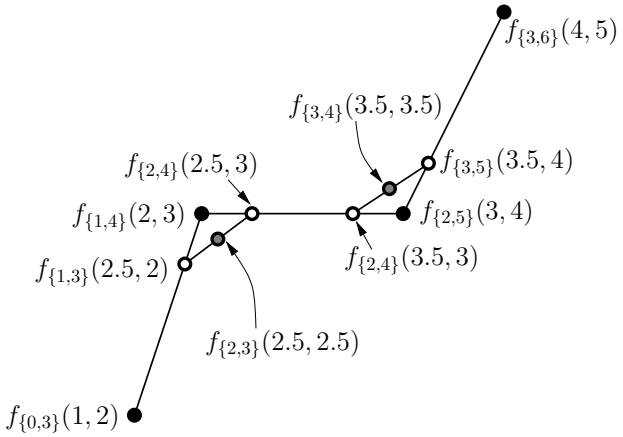

Figure 4.1: Evaluating $F(2.5)$ and $F(3.5)$ in a uniform quadratic B-Spline.

will refer to the original parametric direction as s, and the new direction as t.[1] We add a second value to each point in the domain that accounts for this alternative parametric directions (Figure 4.2). To identify which parametric direction a value refers to, the value will be written with either an "s" or "t". Note, that the "s" and "t" labels shall always be explicitly written for clarity and ease of reading.

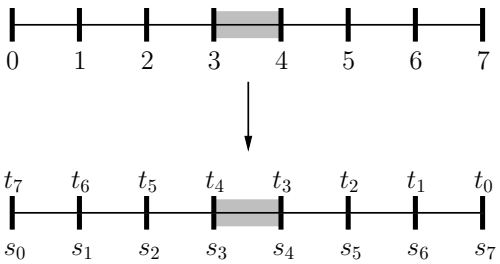

Figure 4.2: The domain doubly indexed.

Any point u in the domain is now represented by a tuple $\{s_i, t_j\}$, where $i + j$ equals the size of the domain. So while there are two parameters, there is only one

[1]This t is not related to particular knots in the knot vector.

degree of freedom. In Figure 4.2 all points in the domain must have $i + j = 7$. So the leftmost point of the shaded region would be given as $\{s_3, t_4\}$.

What is more appealing is that we can specify regions of the domain using only the endpoints of the region. Again, a tuple $\{s_i, t_j\}$ will suffice to specify the endpoints. The s_i value specifies the left endpoint, and t_j the right, so the region is defined as the segment between these two values. For instance, the shaded region in the figure is given as $\{s_3, t_3\}$. Any point $\{s_i, t_j\}$ in the shaded region must necessarily have both $s_i \geq s_3$ and $t_j \geq t_3$.

This notation has a couple of interesting properties. First, the length of an interval can be determined from the region's label by adding up the i and j values and subtracting this value from the size of the domain. This property is consistent with points being represented by a tuple whose i and j values sum to the size of the domain, since a point has no size. Second, if the i and j values sum to a value greater than the size of the domain, it indicates an empty segment. For example, $\{s_5, t_6\}$ specifies an empty interval using the domain from Figure 4.2.

The downside of doubly indexing the domain is if the domain is increased in size, all the t_j values in the domain are affected. As well, points have become a little bulkier with this notation.

4.2 B-Spline Evaluation Revisited

We can update Ramshaw's labelling of the B-Spline control points to include both parametric directions. For this example, I will consider a single region of the cubic B-Spline defined over the domain given in Figure 4.2. The vertical bars will represent the knots of the knot vector used during the evaluation. The new labelling of

the control polygon is given in Figure 4.3.

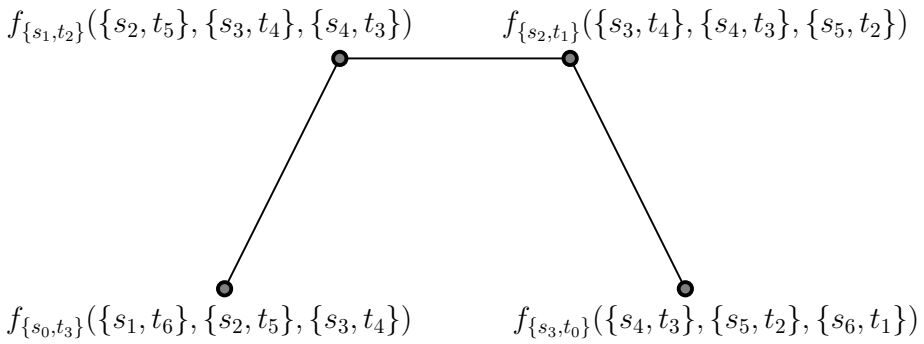

Figure 4.3: The labelled control points of a uniform cubic B-Spline.

Note that the validity interval for each control point can be reproduced directly from its blossom arguments. It is a tuple consisting of an s_i with index one lower than the lowest index s and t_j with index one lower than the lowest index t appearing in the blossom argument. The third control point has blossom arguments of $(\{s_3,t_4\},\{s_4,t_3\},\{s_5,t_2\})$, so the validity interval is necessarily $\{s_2,t_1\}$.

Let us evaluate at a point u in the region $\{s_3,t_3\}$ of the domain. In performing the first level of the de Boor algorithm, adjacent control points are blended using Equation 3.3. Consider blending the first two control points: $f_{\{s_0,t_3\}}(\{s_1,t_6\},\{s_2,t_5\},\{s_3,t_4\})$ and $f_{\{s_1,t_2\}}(\{s_2,t_5\},\{s_3,t_4\},\{s_4,t_3\})$. The new point is derived by inserting a knot at u_1.

To label this new point, we must determine the new validity interval, and the correct blossom arguments. The new validity interval is the intersection of the intervals for the two control points being blended. The regions $\{s_0,t_3\}$ and $\{s_1,t_2\}$ intersect over the interval $\{s_1,t_3\}$, so this is the new point's validity interval. This intersection can also be determined directly. The new interval is composed of the second interval's s value and the first interval's t value.

The blossom arguments are given by taking all the common blossom arguments between the control points, along with the newly inserted knot. The new point will then have blossom arguments of $(u_1, \{s_2, t_5\}, \{s_3, t_4\})$. Thus, the intermediate point will be labelled $f_{\{s_1,t_3\}}(u_1, \{s_2, t_5\}, \{s_3, t_4\})$. Note the validity interval of the new point is consistent with its blossom arguments; the validity tuple contains the s_i and t_j values which are one lower than the corresponding values remaining in the blossom arguments.

This new validity interval shows us where the blending formulas used in Equation 3.3 come from. When the new knot u_1 is inserted, we determine u_1's barycentric coordinates relative to the new validity interval, and use these coordinates to blend the control points.

A similar evaluation is performed for each pair of adjacent control points. The three labelled, intermediate points from the first knot insertion are given in Figure 4.4.

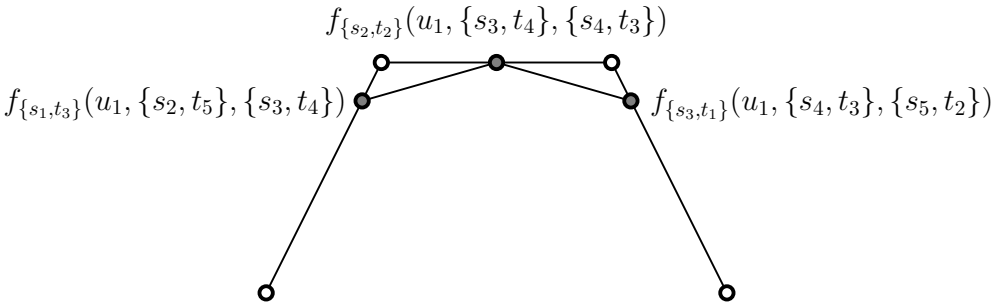

Figure 4.4: The new, labelled points after inserting knot u_1 in a cubic B-Spline.

The de Boor algorithm is repeated with these intermediate points until eventually only one point remains. Figure 4.5 shows the points resulting from inserting the knots u_2 and u_3. The final point in the evaluation is labelled as $f_{\{s_3,t_3\}}(u_1, u_2, u_3)$.

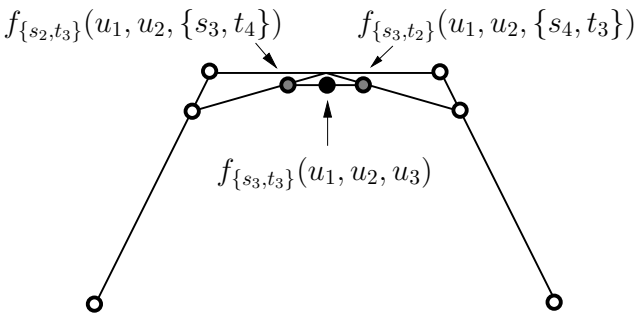

Figure 4.5: Inserting the remaining knots u_2 and u_3.

The validity interval has been reduced to the shaded region of the domain, and all that remains of the blossom arguments are the inserted knots. If $u_1 = u_2 = u_3 = u$ then $f_{\{s_3,t_3\}}(u_1, u_2, u_3)$ is a point on the B-Spline corresponding to $F(u)$.

4.3 A Geometric Labelling

Thus far, adding the extra parameters has done little except clutter the clean labelling scheme of blossoming. One major observation about the de Boor algorithm as I have outlined it here is that the blossom arguments are never directly used in the evaluation. The affine combinations performed between neighbouring control points can be determined solely from the validity intervals. From a geometric standpoint, the blossom arguments are not directly required, and from a notational standpoint could be dropped entirely. The only piece of the blossom argument of interest is which knots have been inserted.

I propose the use of the following **geometric** labelling of the B-Spline control points. Note that this is no longer a blossom label. To remind the reader that these are no longer blossom values, the labels will be given in terms of a function "g",

standing for "geometric".

A point will be labelled using its validity interval. The parameters in the function are restricted to the knots that have been inserted (given in the order they have been inserted). Since there are a variable number of arguments, a superscript in the label shows how many knots are yet to be inserted. As an example, the second B-Spline control point from Figure 4.3 is relabelled from $f_{\{s_1,t_2\}}(\{s_2,t_5\},\{s_3,t_4\},\{s_4,t_3\})$ to the geometric label $g^3_{\{s_1,t_2\}}()$.

The final point of the evaluation given in Figure 4.5 is relabelled as $g^0_{\{s_3,t_3\}}(u_1,u_2,u_3)$. For points in which there are no more knots to be inserted, it is acceptable to drop the 0 superscript. As well, if it is clear over which domain interval the evaluation was performed, the interval on the final point's label may be omitted. Combining these notational simplifications, it may be desirable to label this point simply with $g(u_1,u_2,u_3)$.

Figure 4.6 shows the evaluation for a uniform quadratic B-Spline using the geometric labelling. As before u must be in the domain region $\{s_3,t_3\}$.

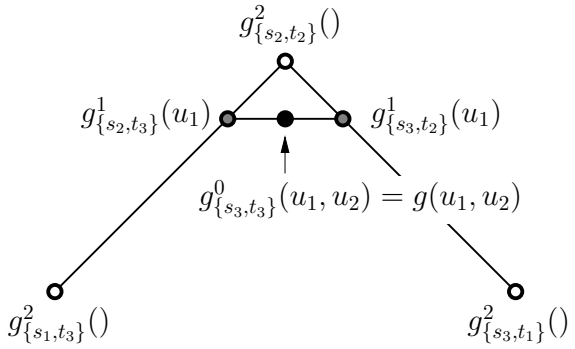

Figure 4.6: Geometric labelling of the quadratic B-Spline evaluation.

4.4 G-Patch Labelling

A similar geometric labelling can be given to the control points of the G-Patch and the intermediate points formed during the evaluation. Once the domain has been properly labelled, this new geometric labelling can be applied.

4.4.1 Domain Points

In a two dimensional domain, the knot vector becomes a knot triangle. Since we are dealing with a uniform G-Patch, this domain will be uniformly triangulated. With a knot vector, the s and t parameters represent points on the number line. With a triangular domain, there are now three parameters, r, s and t, each of these representing a line through the domain in each of the parametric directions. Figure 4.7 shows a triangular shaped domain ten units wide, along with the labelled knot lines in each parametric direction. This knot triangle will be used to define a single cubic G-Patch.

A triangular region of the domain is specified by giving a triple $\{r_i, s_j, t_k\}$ indicating the lines on the outside of the triangle. For instance, the shaded region in Figure 4.7 is contained inside the lines $\{r_3, s_3, t_3\}$. Note that the triangular regions specified in this manner always form equilateral triangles.

A point u in the domain is specified by the three intersecting lines in each parametric direction. As was the case before, valid points in the domain are subject to the constraint that the values i, j and k for the point must sum to the size of the domain (the length of any side of the domain triangle). We will be concerned with evaluating points in the shaded triangle. One such point is the top corner of this region, and it is specified by the label $\{r_4, s_3, t_3\}$.

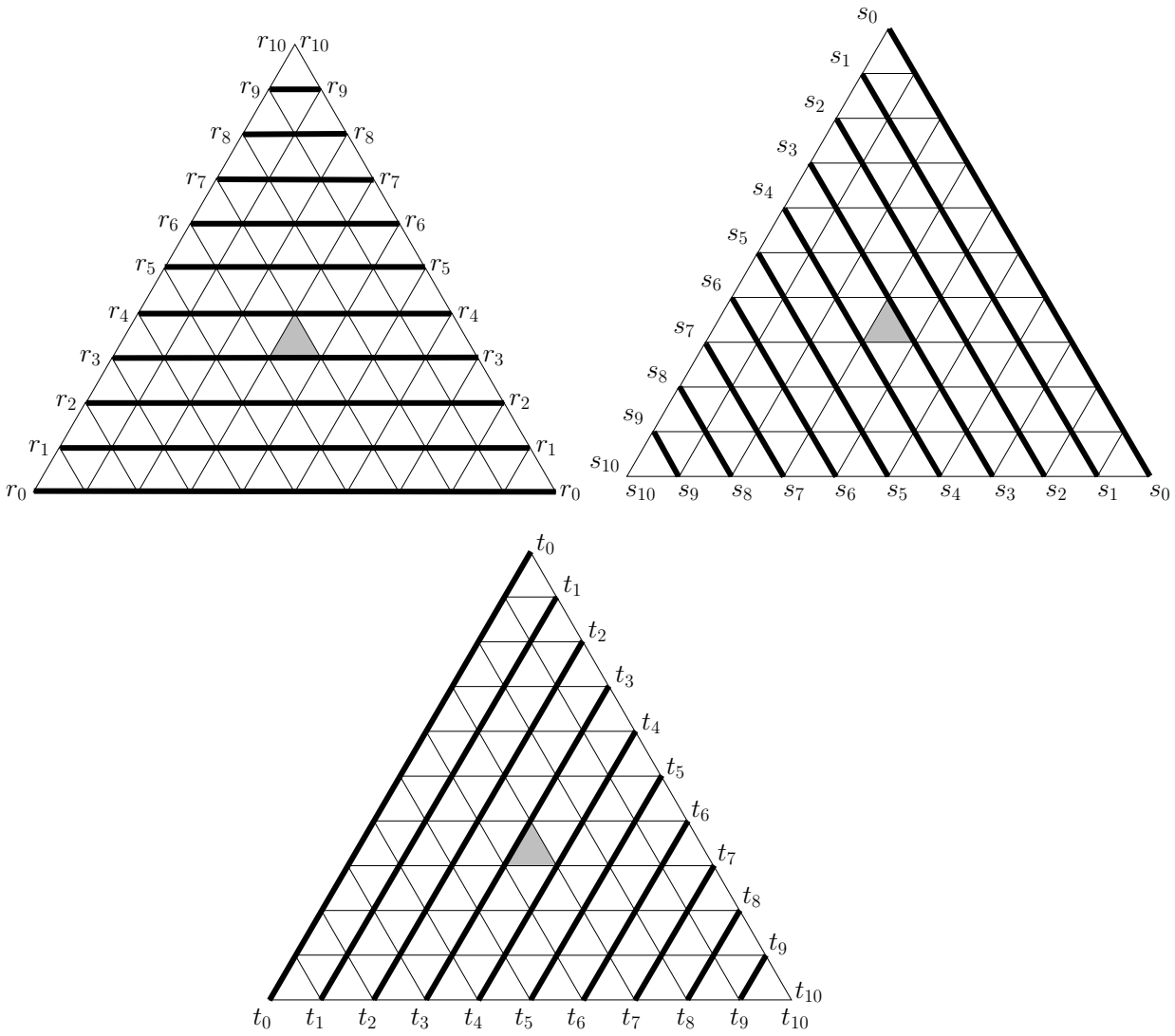

Figure 4.7: The domain triply indexed.

As before, a simple test that a point $\{r_i, s_j, t_k\}$ is inside a region $\{r_a, s_b, t_c\}$ is that i, j and k must all be at least as big as a, b and c respectively.

4.4.2 Cubic G-Patch Labels

All that remains is to determine what validity intervals are appropriate for the control points. We will concern ourselves with evaluating a patch over the grey domain region of Figure 4.7. Each validity interval represents a large equilateral triangle in this domain, and it must include this shaded region of the domain.

Looking back at Figure 3.10 and Figure 3.11 we can start to see where the blending functions were coming from. For instance $\triangle t'_a t'_b t'_c$ in Figure 3.10 corresponds to domain interval $\{r_3, s_2, t_2\}$. The three control points being blended would then have slightly larger domain intervals associated with them that overlap in this region.

Extending this idea back to the original cubic control polygon, Figure 4.8 shows the new labelling of the G-Patch control points that have the appropriate validity intervals.

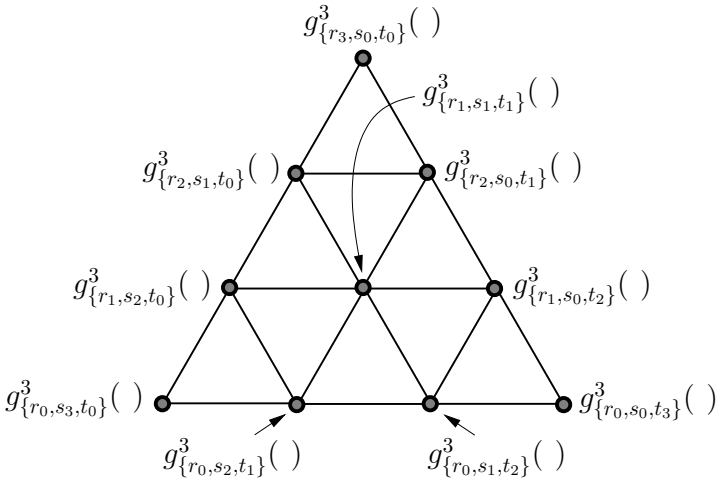

Figure 4.8: The labelled control points of a uniform cubic G-Patch.

It is worth examining the labels of the boundary control points a little closer.

Consider the bottom four control points. The r_i value of each validity interval are unchanged amongst them. If we remove the r arguments altogether the points are labelled identical to the B-Spline control points in Figure 4.3. This reinforces the connection of the G-Patch to the B-Spline.

4.4.3 Cubic G-Patch Evaluation

Let us evaluate a point u in the region $\{r_3, s_3, t_3\}$ of the domain. In performing the first level of the algorithm, three neighbouring control points are blended together. Consider blending the top three control points: $g^3_{\{r_3,s_0,t_0\}}()$, $g^3_{\{r_2,s_1,t_0\}}()$ and $g^3_{\{r_2,s_0,t_1\}}()$ by inserting the knot at u_1. This gives us the first intermediate point of the evaluation.

To label the new point, we must determine its validity interval. As before, this is determined by intersecting the three control point intervals. Figure 4.9 shows the three domain intervals. The intersection of the three regions is shown in grey, and we see the resulting validity interval is $\{r_3, s_1, t_1\}$. This intersection could also be determined directly from the original control point labels. The new interval is the first (top) point's r value, the second (bottom left) point's s value, and the third (bottom right) point's t value. Having determined the new point's domain interval, we can properly label the point with its geometric label: $g^2_{\{r_3,s_1,t_1\}}(u_1)$.

This new validity interval again reinforces the geometry of the G-Patch blending formula. When the new knot u_1 is inserted, we determine u_1's barycentric coordinates relative to this new validity interval, and use these coordinates to blend the three control points. For the three control points above, this means we are determining u_1's barycentric coordinates relative to the triangle $\{r_3, s_1, t_1\}$ as seen in Figure 4.10. Of course, rather than manually determine the new barycentric co-

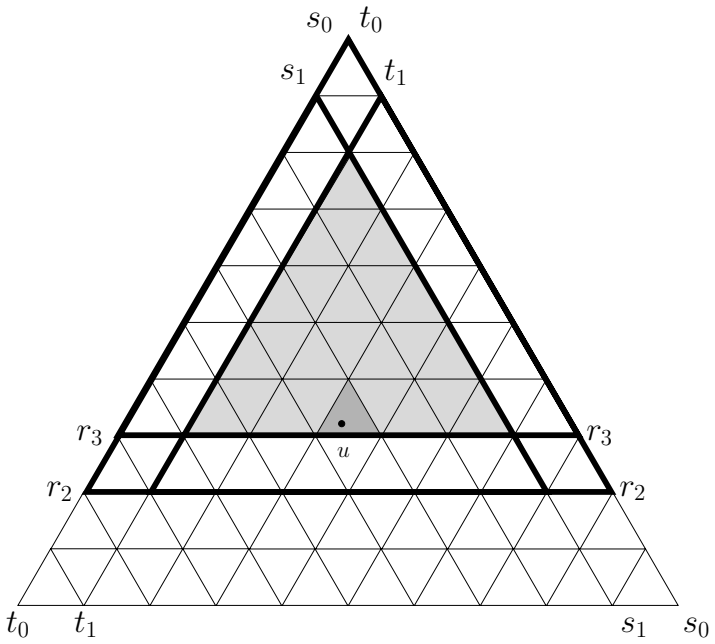

Figure 4.9: Visualizing the intersection of three domain intervals for a cubic G-Patch.

ordinates, we would simply use the precomputed values provided by Equation 3.8.

This evaluation is performed for each set of upward pointing control points. The labelled intermediate points from the first knot insertion are given in Figure 4.11.

The G-Patch evaluation is repeated on the six new control until eventually one point remains. Figure 4.12 shows the results of inserting the last two knots u_2 and u_3.

This final point in the evaluation is labelled $g^0_{\{r_3,s_3,t_3\}}(u_1, u_2, u_3)$. The validity interval has been reduced to the original shaded region of the domain, and all possible knots have been inserted. If $u_1 = u_2 = u_3 = u$ then we have a point on the G-Patch. Recall, that if the domain is already understood to be the shaded triangle, the point may be labelled more compactly by $g(u_1, u_2, u_3)$.

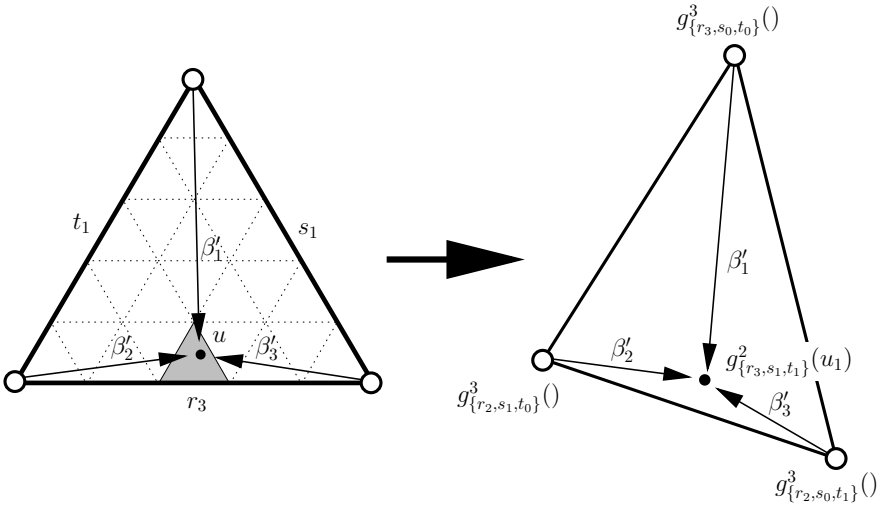

Figure 4.10: Blending three G-Patch control points using validity intervals. u can be any point inside the shaded region of the domain. The barycentric coordinates of u with respect to this larger domain interval are used to blend the three control points.

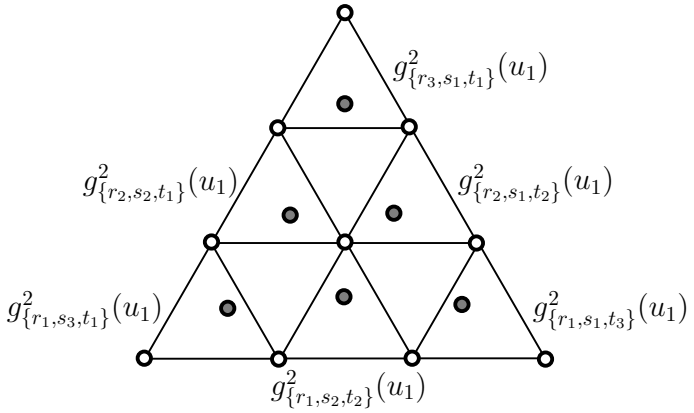

Figure 4.11: The labelled points after inserting knot u_1 in a cubic G-Patch.

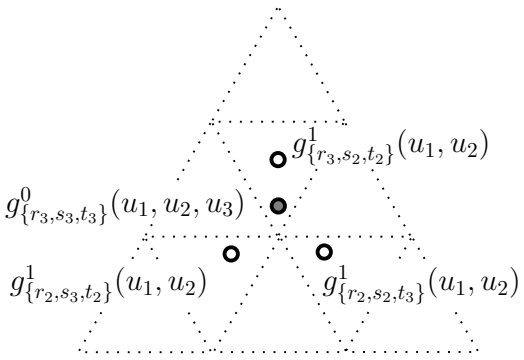

Figure 4.12: The points resulting from the final two knot insertions.

Chapter 5

The G-Patch Blending Functions

A B-Spline controls a network of Bézier curves, automatically giving the necessary continuity between neighbouring curves. For a degree n B-Spline, any region of the domain can be converted into a corresponding Bézier curve. Similarly, a degree n G-Patch can be represented by an equivalent degree n Bézier patch. In this chapter I will develop the underlying G-Patch blending functions in a manner that resembles the B-Spline basis functions, as well as show how to convert the I-Patch into Bézier form.

The labelling given in Chapter 4 is identical to the geometric B-Spline labelling with the addition of a parametric direction. Superficially, this appears to be the generalization of B-Splines over a triangular domain. However, the labels are not polar forms, and it is natural to want to generate blossom values for these points. Exploring the relationship between G-Patch and Bézier control points will allow the appropriate connection to be made.

5.1 G-Patch Asymmetry

One of the immediate consequences of the labelling in Chapter 4 is to wonder if we can recover Bézier control points for a G-Patch simply by inserting the appropriate knots. This is the technique used with B-Splines, so it seems a reasonable expectation. We will once again consider a G-Patch defined over the shaded region of Figure 4.7, namely $\{r_3, s_3, t_3\}$. To simplify the discussion, I will label the corners of this domain triangle with an a, b and c as in Figure 5.1.

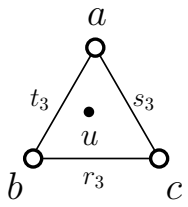

Figure 5.1: Labelling the corners of the G-Patch domain.

Consider a quadratic G-Patch defined over the domain $\triangle abc$. The corresponding Bézier patch will have six control points: $f(a,a)$, $f(b,b)$, $f(c,c)$, $f(a,b)$, $f(a,c)$ and $f(b,c)$.

Let us attempt to generate $f(a,b)$ by inserting the knot a, followed by the knot b into the G-Patch. This derivation is performed on the left of Figure 5.2. The three darkened inner points represent the insertion of the first knot a, and the black point is the result of inserting the second knot b. However, the choice to insert the a knot first is arbitrary. Polar forms are symmetric, and $f(a,b) = f(b,a)$. Let us run the derivation the other way, by inserting the b knot first followed by the a knot. The result is given on the right of Figure 5.2.

The problem is immediately apparent: $g(a,b) \neq g(b,a)$. More troubling is that

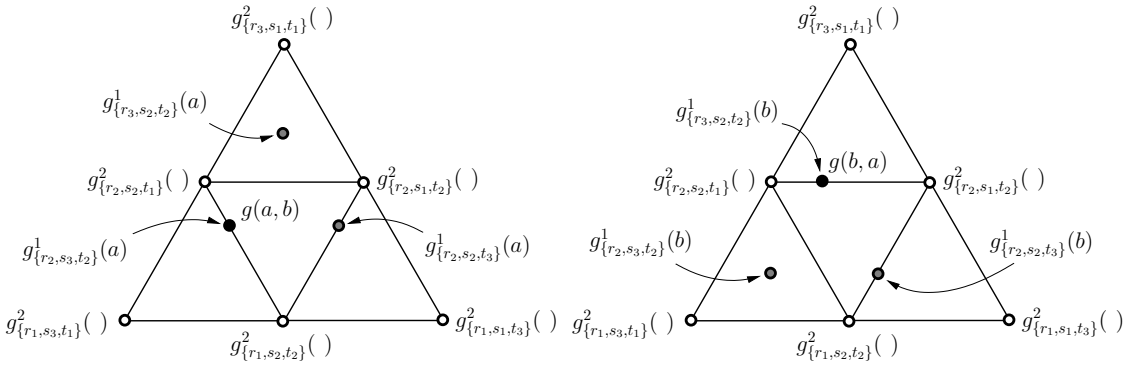

Figure 5.2: Inserting the knot a followed by b (left) and the knot b followed by a (right) into a quadratic G-Patch.

neither one of these point agrees with where $f(a,b)$ is supposed to be (somewhere in the downward pointing triangle). The order of knot insertion influences the resulting point, so the geometric labels are not symmetric (and consequently are not blossom values). For the moment I will leave this discussion. Only after knowing precisely where the Bézier control point $f(a,b)$ should occur, can this ambiguity be resolved.

5.2 G-Patch Blending Functions

When evaluating a G-Patch, the control points are recursively blended together. This has the effect of weighting each of the original control points by a different underlying function. I define $I^n_{i,j}(u)$ to be the degree n, G-Patch function weighting the control point $P_{i,j}$ for a point u in the domain. A point on the surface of the patch is represented by

$$F(u) \;=\; \sum_{i=0}^{n}\sum_{j=0}^{i} I^n_{i,j}(u)P_{i,j}. \tag{5.1}$$

This notation hides many of the details outlined in the previous chapter, namely the control points must contain the point u in their validity intervals. However, once the control polygon has been identified, it is simply a matter of substituting u's barycentric coordinates relative to the domain triangle into Equation 5.1.

An immediate question is how to calculate these blending functions? Consider the data flow diagram of the quadratic G-Patch given in Figure 5.3. The point u is represented by its barycentric coordinates (β_1, β_2, β_3) relative to the domain triangle $\triangle abc$. The original control points are on the outside and shaded white, while the intermediate points in the evaluation are in the center. The point on the surface is at the center of the diagram. The weightings used to generate the intermediate points are given along each edge. If we follow the arrows from each control point to the center, we see the contribution that each control point makes.

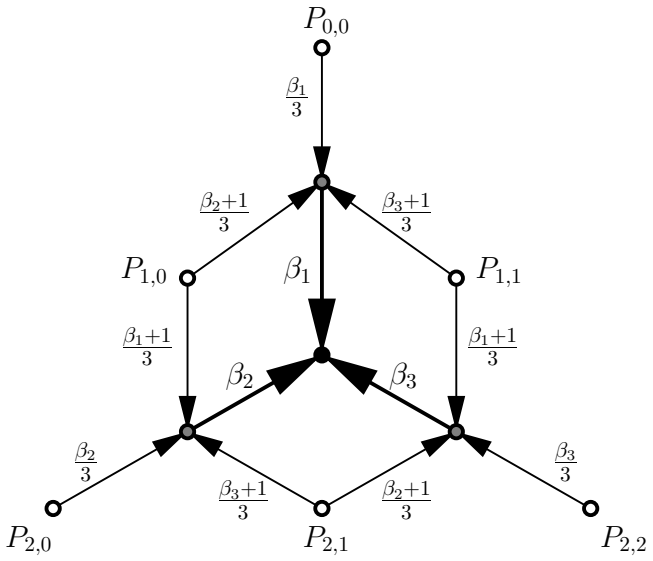

Figure 5.3: Data flow diagram for evaluating a quadratic G-Patch.

For instance, $P_{0,0}$ contributes $\frac{\beta_1}{3} \cdot \beta_1$ to the point on the surface. As an example,

if we were determining the image of $u = a$ and substitute a's barycentric coordinates ($\beta_1 = 1$, $\beta_2 = 0$, $\beta_3 = 0$) we find that $P_{0,0}$ contributes $\frac{1}{3}$ to a's image.

To insert two distinct knots, u_1 followed by u_2, u_1's barycentric coordinates would be used to follow an edge from a white point to a grey point, and u_2's barycentric coordinates would be used to follow an edge from a grey point to a black point.

Running all the arrows of the data flow diagram in reverse, putting 1 at the root and the blending function labels on the outside, we generate each of the functions weighting the control points (Figure 5.4). Summing all the paths from the root to each particular label gives the weight of that function. For example, there are two paths from the root to $I^2_{1,0}(u)$ giving a total contribution for this function of $\beta_1 \cdot \frac{\beta_2+1}{3} + \beta_2 \cdot \frac{\beta_1+1}{3}$.

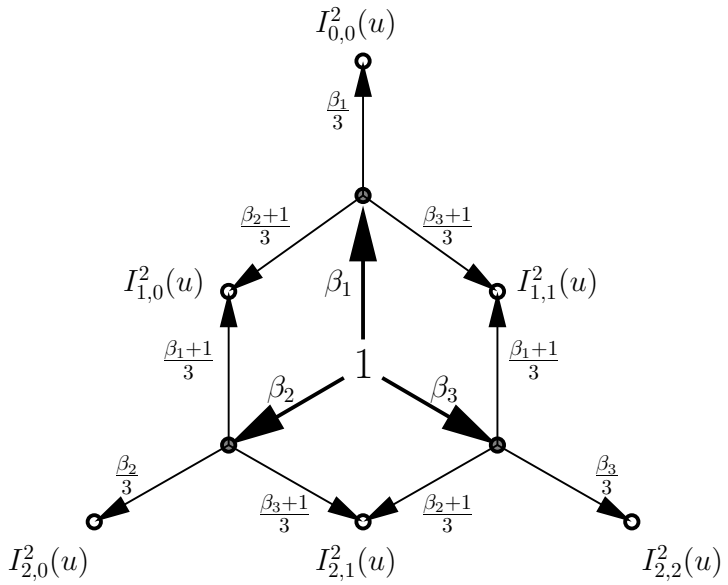

Figure 5.4: Reversing the data flow diagram for a quadratic G-Patch.

This diagram can be used directly to show how the higher degree blending

functions can be defined in terms of lower degree functions. The following is a recurrence relation for a degree m G-Patch blending function reminiscent of the Cox-de Boor-Mansfield B-Spline recurrence relation:

$$\begin{aligned} I^m_{i,j}(u) &= \frac{\beta_1 + i}{2m-1} I^{m-1}_{i,j}(u) + \frac{\beta_2 + m + j - i}{2m-1} I^{m-1}_{i-1,j}(u) + \frac{\beta_3 + m - j}{2m-1} I^{m-1}_{i-1,j-1}(u) \\ I^0_{0,0}(u) &= 1 \\ I^m_{i,j}(u) &= 0 \quad \text{if } i < 0,\ j < 0,\ i < j,\ \text{or } i > m, \end{aligned} \tag{5.2}$$

where u is expressed as the barycentric coordinates relative to the domain triangle.

From Figure 5.4, the following are the six quadratic G-Patch blending functions in terms of u's barycentric coordinates:

$$\begin{aligned} I^2_{0,0}(u) &= \frac{1}{3}\left({\beta_1}^2\right) \\ I^2_{1,0}(u) &= \frac{1}{3}\left(2\beta_1\beta_2 + \beta_1 + \beta_2\right) \\ I^2_{1,1}(u) &= \frac{1}{3}\left(2\beta_1\beta_3 + \beta_1 + \beta_3\right) \\ I^2_{2,0}(u) &= \frac{1}{3}\left({\beta_2}^2\right) \\ I^2_{2,1}(u) &= \frac{1}{3}\left(2\beta_2\beta_3 + \beta_2 + \beta_3\right) \\ I^2_{2,2}(u) &= \frac{1}{3}\left({\beta_3}^2\right) \end{aligned} \tag{5.3}$$

At this point we can visualize each basis function by evaluating all valid values of u inside the domain triangle.

A similar set of derivations was performed for a cubic G-Patch to derive ten cubic blending functions. The following three figures (5.5, 5.6, 5.7) show a rendering of three of the blending functions. Note the seven functions not rendered are similar to these, being either rotations or mirror images.

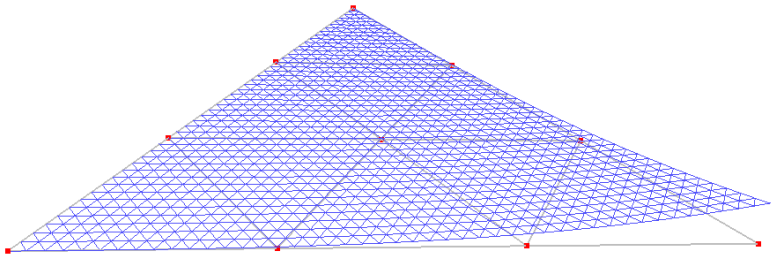

Figure 5.5: The G-Patch blending function $I^3_{3,3}(u)$.

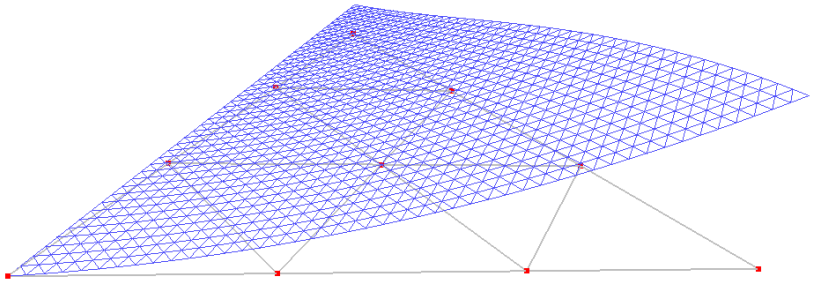

Figure 5.6: The G-Patch blending function $I^3_{2,2}(u)$.

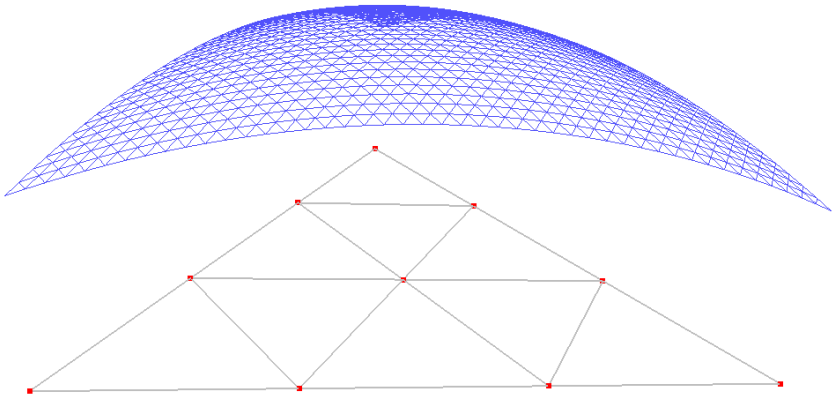

Figure 5.7: The G-Patch blending function $I^3_{2,1}(u)$.

5.3 Converting G-Patches to Bézier Form

To derive the conversion between a G-Patch and its Bézier representation, we will use the monomials as an intermediate representation. It is well known how to convert from monomial to Bézier form, so the only difficulty is converting a G-Patch to monomial form. Goldman provides an extensive discussion of converting between numerous, well-known curve and surface bases [10]. In particular, there is a discussion about representing a Bézier surface defined over the triangular domain in its monomial form.

Equation 5.3 is almost in the required monomial form. Following Goldman's approach, we set $\beta_1 = a$, $\beta_2 = b$ and $\beta_3 = 1 - a - b$ giving the quadratic G-Patch represented in the monomial basis:

$$
\begin{aligned}
I^2_{0,0}(u) &= \frac{1}{3}a^2 \\
I^2_{1,0}(u) &= \frac{2}{3}ab + \frac{1}{3}a + \frac{1}{3}b \\
I^2_{1,1}(u) &= -\frac{2}{3}a^2 - \frac{2}{3}ab + \frac{2}{3}a - \frac{1}{3}b + \frac{1}{3} \\
I^2_{2,0}(u) &= \frac{1}{3}b^2 \\
I^2_{2,1}(u) &= -\frac{2}{3}ab - \frac{2}{3}b^2 - \frac{1}{3}a + \frac{2}{3}b + \frac{1}{3} \\
I^2_{2,2}(u) &= \frac{1}{3}a^2 + \frac{2}{3}ab + \frac{1}{3}b^2 - \frac{2}{3}a - \frac{2}{3}b + \frac{1}{3}.
\end{aligned}
$$

Using these equations, we can create a matrix to convert the G-Patch control points to monomial control points, hereafter referred to as the I matrix:

$$
I = \begin{bmatrix}
1/3 & 0 & 0 & 0 & 0 & 0 \\
0 & 2/3 & 0 & 1/3 & 1/3 & 0 \\
-2/3 & -2/3 & 0 & 2/3 & -1/3 & 1/3 \\
0 & 0 & 1/3 & 0 & 0 & 0 \\
0 & -2/3 & -2/3 & -1/3 & 2/3 & 1/3 \\
1/3 & 2/3 & 1/3 & -2/3 & -2/3 & 1/3
\end{bmatrix}.
$$

To produce a row matrix, m, containing the monomial coefficients multiply

$$m = v \cdot I,$$

where v is a row vector of the G-Patch control points of the form

$$v = \begin{bmatrix} P_{0,0} & P_{1,0} & P_{1,1} & P_{2,0} & P_{2,1} & P_{2,2} \end{bmatrix}.$$

It is worth noting that m's coefficients for the monomials are provided in the order a^2, ab, b^2, a, b, 1.

The next step is to generate the matrix needed to convert monomials to Bézier representation. Farin gives a simple matrix, B, that converts Bézier to monomial form [6], so taking its inverse produces the needed matrix. To convert from Bézier form to the monomial coefficients (in the order given above) we have

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ -2 & -2 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -2 & -2 & 0 & 2 & 0 \\ 1 & 2 & 1 & -2 & -2 & 1 \end{bmatrix},$$

and its inverse is thus

$$B^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1 & 1/2 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

The conversion between G-Patch control points and Bézier control points can be performed directly by deriving a conversion matrix, C, given by

$$C = I \cdot B^{-1}.$$

Doing so with the above matrices yields the following quadratic G-Patch to quadratic Bézier patch conversion matrix:

$$C_2 = \begin{bmatrix} 1/3 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 2/3 & 1/6 & 1/3 & 1/6 & 0 \\ 1/3 & 1/6 & 2/3 & 0 & 1/6 & 1/3 \\ 0 & 0 & 0 & 1/3 & 0 & 0 \\ 0 & 1/6 & 1/6 & 1/3 & 2/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 1/3 \end{bmatrix} \tag{5.4}$$

The subscript for the conversion matrix indicates what degree patch is being converted. To produce a row vector b of Bézier control points multiply

$$b = v \cdot C_{deg}.$$

Figure 5.8 shows a quadratic G-Patch and its control polygon beside the same patch converted to Bézier control points using the C_2 matrix. The patch is the same, only the control points are different.

For those wishing to implement this conversion for higher degree patches, I have provided the conversion matrices C_3 and C_4 for cubic and quartic patches in Appendix A.

5.4 Another Conversion Technique

At first glance, the contents of the conversion matrices C_i look arbitrary. However, a closer examination shows the values have meaning in terms of knot insertion. The first observation is that all the entries are between 0 and 1, so the conversion

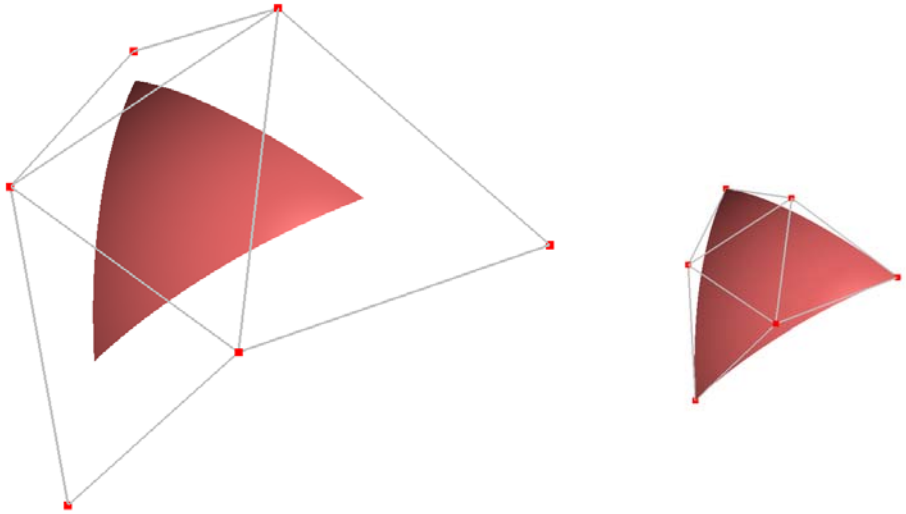

Figure 5.8: A quadratic G-Patch (left) converted to its Bézier form (right).

to Bézier form is performing convex combinations of the original G-Patch control points. This suggests the existence of a direct conversion technique by performing affine combinations of control points.

Focussing on the C_2 matrix, the first column shows the location of $f(a, a)$. It occurs in the barycenter of the top three control points. This is the same location that the G-Patch places $g(a, a)$. Looking at columns four and six show that $f(b, b)$ agrees with $g(b, b)$ and $f(c, c)$ agrees with $g(c, c)$.

The second column shows the derivation of $f(a, b)$. It only involves the middle three control points and the function evaluates to

$$f(a,b) = \frac{2}{3}P_{1,0} + \frac{1}{6}P_{1,1} + \frac{1}{6}P_{2,1}. \tag{5.5}$$

At this point it is worth looking at the formulas for the location of $g(a, b)$ and $g(b, a)$. Referring back to Figure 5.2 and performing the knot insertion calculation

on the data flow diagram in Figure 5.3 we get

$$g(a,b) = \frac{2}{3}P_{1,0} + \frac{1}{3}P_{2,1} \tag{5.6}$$

$$g(b,a) = \frac{2}{3}P_{1,0} + \frac{1}{3}P_{1,1}. \tag{5.7}$$

It appears that $f(a,b)$ falls in the midpoint between $g(a,b)$ and $g(b,a)$. A quick verification shows that indeed

$$f(a,b) = \frac{g(a,b) + g(b,a)}{2}. \tag{5.8}$$

Further exploration on cubic G-Patches and the C_3 matrix shows similar results:

$$f(a,a,b) = \frac{g(a,a,b) + g(a,b,a) + g(b,a,a)}{3}$$

$$f(a,b,c) = \frac{g(a,b,c) + g(a,c,b) + g(b,a,c) + g(b,c,a) + g(c,a,b) + g(c,b,a)}{6}$$

Thus, the location of the symmetric polar value for f is just the average of each of the asymmetric G-Patch evaluations. This provides the alternate method for converting a G-Patch into a Bézier patch. Perform all the permutations of knot insertions for the particular Bézier control point, and then average the results.

5.5 G-Patch Basis Functions

An obvious question that remains is whether or not the G-Patch blending functions, $I^n_{i,j}(u)$, form a basis. They are based on the geometry of the B-Spline, and they have a recurrence relation defining them that is similar to the B-Spline basis functions, $N^n_i(u)$, so it is tempting to make the same generalizations about the G-Patch blending functions.

Theorem 3 (G-Patch Blending Functions Form a Basis) *The set of degree n G-Patch blending functions $I^n_{i,j}(u)$ form a basis for the linear space of dimension $\binom{n+2}{2}$ for $n \leq 4$.*

Proof: The G-Patch to Bézier conversion matrix C_i is invertible for $0 \leq i \leq 4$.

□

I conjecture that this property will hold for all n, but this remains an open problem.

Chapter 6

G-Patch Networks

Recall that the primary goal for a control scheme is to regulate a collection of Bézier patches. In the previous section, I showed how a single G-Patch can be converted into a Bézier patch. The final step is to show how to create a network of G-Patches that form a larger surface.

I will now describe how to construct a surface built from a network of G-Patches in which adjacent patches share control points. The construction gives the user local control over the upward pointing patches, and attempts to smoothly fill in the downward pointing patches to complete the surface.

6.1 Upward Pointing Triangular Patches

A single G-Patch of degree n is formed using $\binom{n+2}{2}$ control points laid out in the shape of an upward pointing triangle. This patch corresponds to an upward pointing triangle in the domain with sides of unit length. Now consider the adjacent regions of the domain in each parametric direction. The corresponding patches will also be

generated by a collection of $\binom{n+2}{2}$ control points. Let us look at the control points labelling one of these patches and determine what similarities there are to the first patch.

For simplicity, the discussion will consider quadratic patches defined over the domain in Figure 6.1. There are three upward pointing triangles and one downward pointing triangle from this domain that we will consider. The domain triangles are labelled A through D for easier discussion.

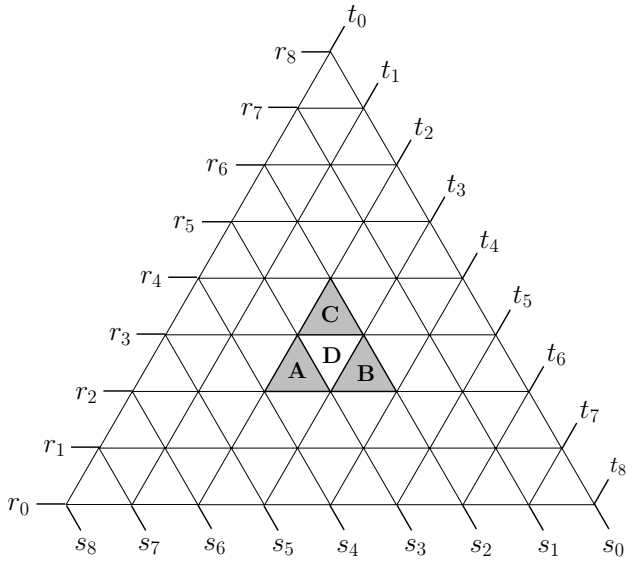

Figure 6.1: Domain for a network of four cubic G-Patches.

Consider the labels of the control points for the G-Patch corresponding to domain triangle A and compare it to that of domain triangle B (Figure 6.2). We see that the three bottom-right labels of patch A match the bottom-left labels of B. With a network of B-Spline control points, having neighbouring curves reuse the same control point when their labels agree gives the resulting network of curves high degrees of continuity. It seems appropriate to have the two neighbouring patches

reuse the same control points whenever their labels agree. Thus we can represent the two patches with nine control points as in Figure 6.3.

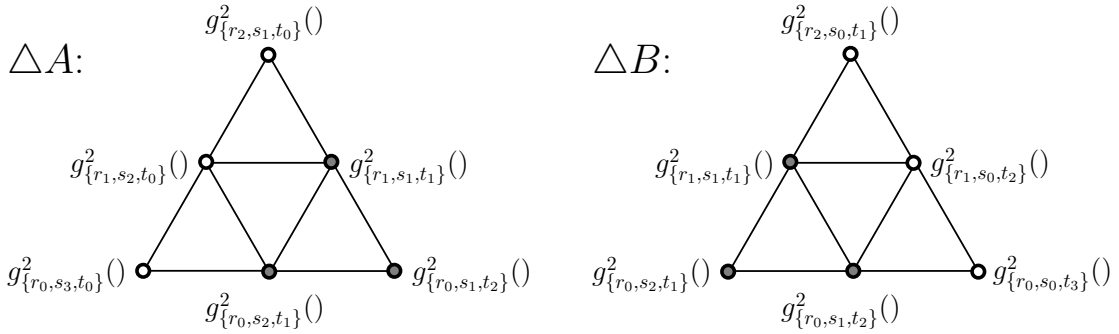

Figure 6.2: Control polygons for domain triangles A (left) and B (right). The three control points that are common to both control polygons are shown in grey.

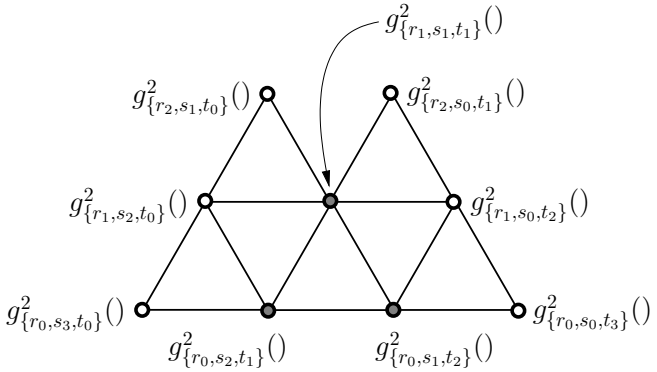

Figure 6.3: Reusing neighbouring G-Patch control points. The three grey points are used in the evaluation of patch A and patch B.

The labelling for the domain triangle C provides even more duplicated control points. It has three control point labels in common with both patch A and patch B. Again, if we set these matching labels to the same control points, we can now completely specify the three G-Patches with only ten distinct control points as

given in Figure 6.4.

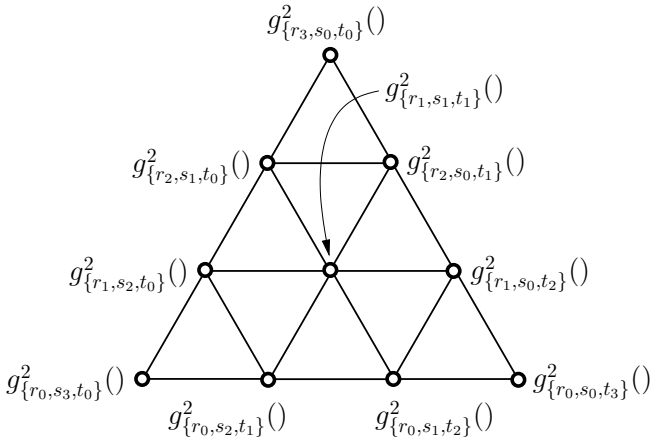

Figure 6.4: Control network for domain triangles A, B and C.

Looking at the four control point labels along the outer edges, we notice that they share a remarkable similarity to B-Spline control points. The only difference is an extra parametric direction, r. The analysis performed in Chapter 7 will unfortunately determine that patch boundaries are not, themselves, B-Spline curves.

At this point we can visualize the results of utilizing this control scheme. If the user manipulates one of the corner control points given in Figure 6.4, it will only have an effect on one of the patches. Figure 6.5 shows a control network in which all ten of the points lie uniformly in a plane except for one of the corner control points which was pulled up approximately a unit away from the plane.

Notice that the individual G-Patches agree with their neighbours at the corners of the patch. If they did not meet in this fashion, this construction would hold little value, as it would not be possible to construct even a C^0 patch network.

If the user manipulates one of the edge control points, this will effect the two G-Patches that share the control point, while leaving the other patch unaffected.

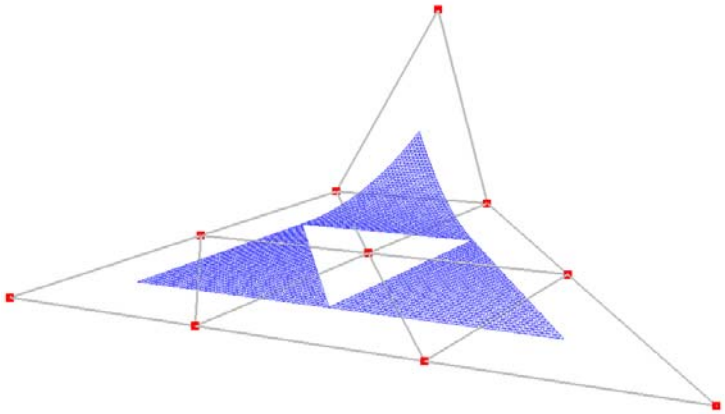

Figure 6.5: Moving a corner control point in a network of three quadratic G-Patches.

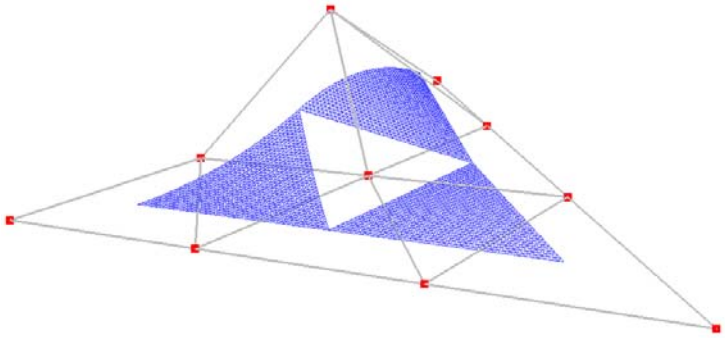

Figure 6.6: Moving an edge control point in a network of three quadratic G-Patches.

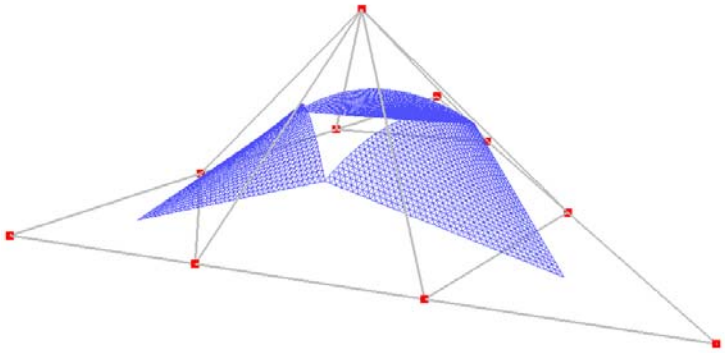

Figure 6.7: Moving the center control point in a network of three quadratic G-Patches.

Figure 6.6 shows a control network similar to Figure 6.5, but with the altered control point being located along the edge.

Finally, if the user manipulates the center control point, all three G-Patches will be pulled towards the moved control point. The result of such a movement is given in Figure 6.7.

These patch networks are scalable. Should the user wish to add more patches of the same degree in any parametric direction, it only requires adding additional rows of control points. If higher degree patch networks are desired, the same technique is used, with the only difference being that more control points will agree between neighbouring patches. For instance two adjacent cubic patches will have six control points in common. Figure 6.8 shows this scaling with a larger network of cubic G-Patches. Notice that the patches are only pulled towards the nearby control points.

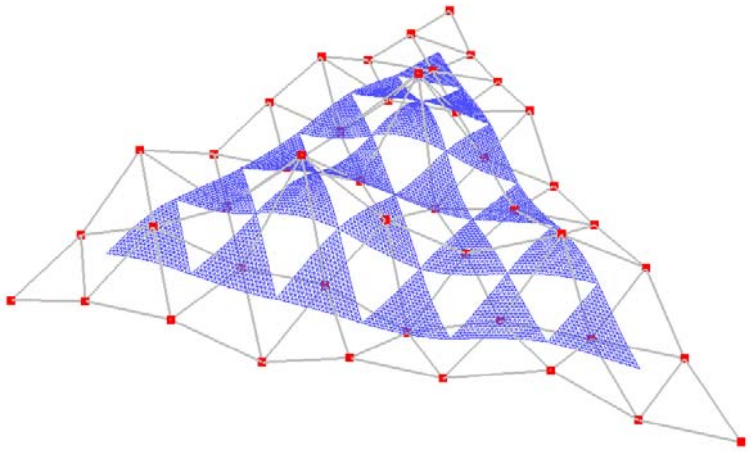

Figure 6.8: A large network of upward pointing cubic G-Patches.

6.1.1 Evaluation

The evaluation of a large network of patches is not difficult. For a degree n patch, after determining the $\binom{n+2}{2}$ control points corresponding to each region of the domain, the patch is evaluated at various values of u. Either the algorithm from Section 3.3.4 is used, or the G-Patch control points are converted to Bézier control points and a Bézier patch tessellator is used to render the surface.

6.2 Downward Pointing Triangular Patches

When dealing with one patch, we have always oriented our domain such that the base of a triangle was at the bottom. However, once we begin to stitch a large collection of triangular patches together, some pieces of the domain are oriented in the opposite direction. Namely in Figure 6.1 the domain triangle D has the

opposite orientation. I will demonstrate how to generate the downward patches for quadratic, cubic and quartic patches.

6.2.1 Quadratic G-Patches

First we need to look at how to label the domain triangle D. The three parametric lines that form the edges of the triangle are $\{r_3, s_3, t_3\}$. The original definition of domain triangle labelling from Section 4.4.1 defined valid points in the triangle as necessarily having their r, s and t parameters greater than or equal to $\{r_3, s_3, t_3\}$. However, there are no points in the domain that match this specification. Upon closer inspection, it appears that for downward pointing triangles, the definition needs to be reversed, namely that the r, s and t parameters must all be **less than** or equal to $\{r_3, s_3, t_3\}$.

The problems with downward pointing triangles go much further. What points should be used to specify the control polygon for the downward pointing patch? Ideally, we would like these points to be readily available in terms of the points of Figure 6.4. However there does not appear to be a downward pointing triangle meeting this criteria.

There is a reprieve, though, and it comes from looking at the Bézier representation of each of the three upward G-Patches. For the downward pointing patch to meet its neighbours with C^0 continuity, the control points along the edges of each neighbouring patch must agree with the new patch. This completely specifies a degree 2 Bézier patch. In Figure 6.9, a network of three G-Patches is shown, with the G-Patch control net given on the left, and the corresponding Bézier control networks on the right.

The central Bézier control points in Figure 6.9 represent the G-Patch corre-

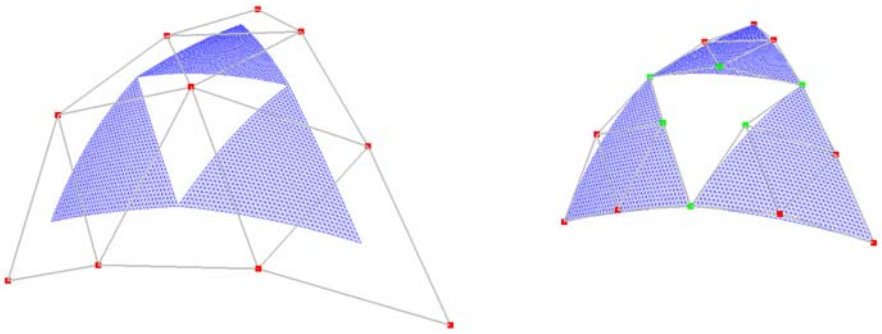

Figure 6.9: A Quadratic G-Patch network (left) converted to Bézier representation (right).

sponding to the domain triangle D. Evaluating this additional Bézier patch creates a continuous surface as seen in Figure 6.10.

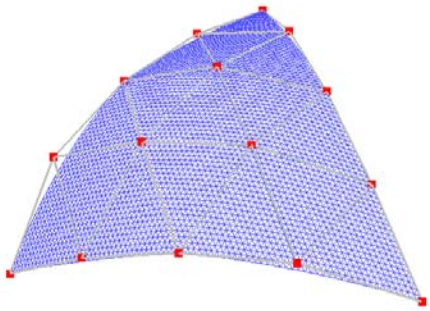

Figure 6.10: Filling the hole in a quadratic G-Patch network.

Recalling the results of Theorem 2, C^0 is as good as you can hope for with quadratic patches while still allowing local control. Thus far the G-Patch scheme

is solving Ramshaw's problem.

6.2.2 Cubic G-Patches

This same strategy can be used to handle higher degree G-Patch networks. For the cubic case, the downward pointing patch has ten Bézier control points. The outer nine points are completely defined by the edges of the neighbouring patches to ensure C^0 continuity. This leaves the center control point. At this juncture we can start to look at the C^1 continuity conditions in trying to place the new point. For this to occur, we need the first derivatives to match between the neighbouring patches. Consider the control points of the Bézier patches for domain triangles A and D as seen in Figure 6.11. We are interested in determining the grey control point of the downward patch.

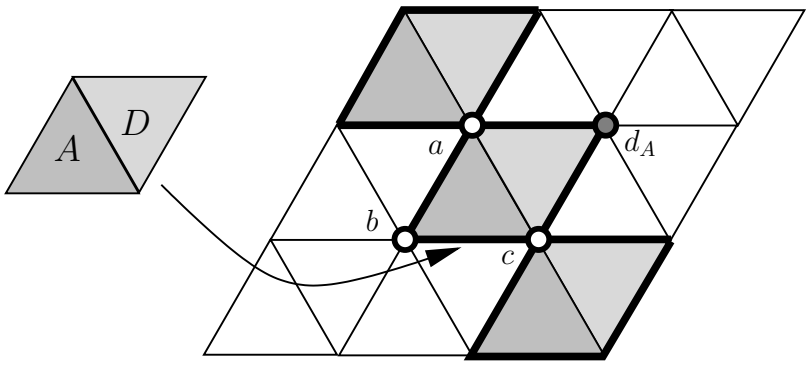

Figure 6.11: Cubic Bézier control points for neighbouring patches.

For the patches to have matching first derivatives, the three pairs of triangles that meet along the patch boundary (the greyed diamond shapes of Figure 6.11) must be coplanar. More specifically, the diamonds should have the same shape as the two domain triangles A and D. This tells us how to generate the grey

control point. By properly blending A's three white control points (a, b and c) with the appropriate barycentric coordinates, the grey point is determined (d_A). The following formula gives us the location of the new point as predicted by patch A:

$$d_A = 1a - 1b + 1c. \tag{6.1}$$

This only gives us the location that patch A predicts the Bézier control point should appear. We can just as easily use the Bézier control points of patch B and patch C (Figure 6.1) to get other locations of the d point (d_B and d_C). This gives three possible values for d. The obvious question to ask is which is the correct value to use? In an ideal world, they would all agree with each other, but that is not likely to occur. The next best thing to do is to take the average of the three predictions:

$$d = \frac{1}{3}\left(d_A + d_B + d_C\right). \tag{6.2}$$

This value of d is likely not going to agree with any of the neighbouring patches. If so, the new patch will not meet its neighbours with C^1-continuity, a fact proven more rigourously in the next chapter. The hope is that the result is close enough to not be discernable. Figure 6.12 shows the large network of cubic patches from Figure 6.8 with the downward patches now filled in using Equation 6.2. At first glance the averaging does not seem to be causing a noticeable problem, as the surface appears reasonably smooth.

6.2.3 Quartic G-Patches

Specifying the 15 quartic Bézier control points of the downward pointing G-Patches is similar to the cubic case. The 12 points on the outside of the patch are specified by the C^0 continuity conditions with its neighbours, while the three inner points are

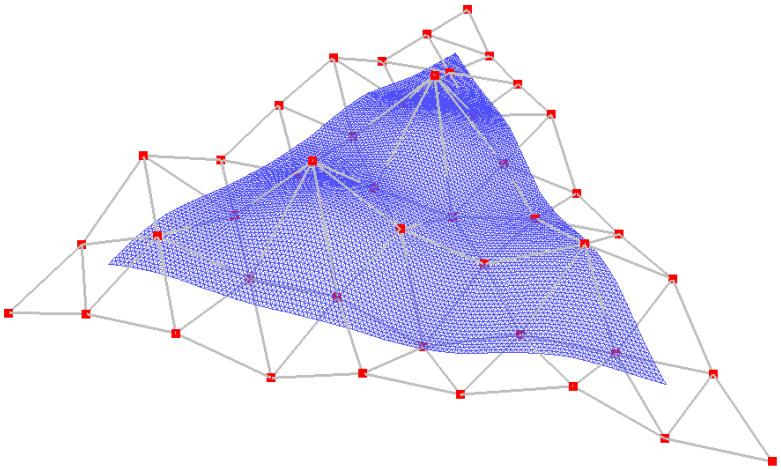

Figure 6.12: A cubic G-Patch surface.

specified using the C^1 conditions. Again, we average the location that is predicted by each of the neighbouring patches. This time, only two patches are competing for the location of each of the interior points due to the size of the patch.

This leads to an important observation. As the degree of the patches get higher, there are more degrees of freedom to place interior control points, which allows for the possibility of smoother levels of continuity when filling the holes.

A large network of quartic G-Patches with the downward patches specified in this manner is given in Figure 6.13. The surfaces are exhibiting what appear to be a higher degree of continuity, as there appears to be no noticeable discontinuities between neighbouring patches. The next chapter will formalize what level of continuity is realizable with this G-Patch network construction. As well, the surfaces will be rendered with smooth shading to allow for more subjective analysis.

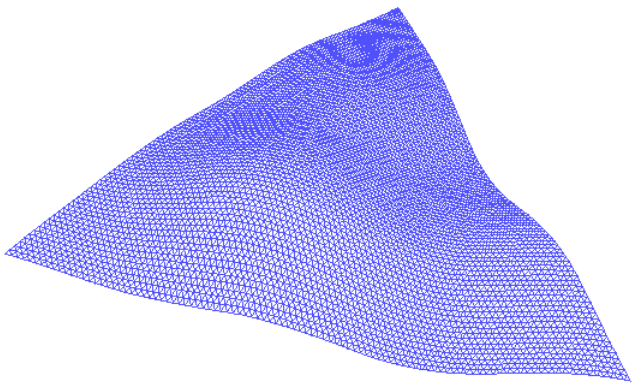

Figure 6.13: A quartic G-Patch surface.

Chapter 7

Analysis

I have provided a means to construct a locally flexible control network of G-Patches in which initial wire-frame renderings of higher degree surfaces appear to exhibit the continuity actively being sought. I will now analyze the results of my G-Patch scheme, and compare them against DMS-Splines. I will address performance issues with the implementation, followed by an examination of the overall surface continuity exhibited by neighbouring patches, and ultimately determine the limitations of the scheme.

7.1 Implementation

The G-Patch construction has been put together in a simple tool that provided the screen captures used throughout this thesis. The user has the ability to select control points and pull them in any direction. When a control point is moved, the effected upward pointing patches are recomputed. As well, any neighbouring downward pointing patches that are effected are updated.

7.1.1 User Interface

In terms of usability, manipulating G-Patches has an intuitive feel. The user tugs on one of the control points, and the nearby region of the surface bulges in that area. Regions of the surface far removed from the interaction are unaffected by the movement. The user need not be aware that the underlying surface is based on G-Patches.

Thus far, it is on par with manipulating control points in DMS-Splines. The major difference, however, is the lack of knot clouds that the user is faced with when manipulating DMS-Splines. As discussed in the background, the knot cloud is intimately related to the resulting surface, and as yet no one has found a nice way to hide this detail from the user. Thus, the G-Patch surface seems to have the advantage of a simpler control scheme.

The one interface advantage that is held by DMS-Splines is that the domain need not have a regular triangulation as the G-Patch domain requires. This allows more flexibility when trying to model with DMS-Splines.

7.1.2 Computational Requirements

Since only the local patches are effected by the movement of a G-Patch control point, the computational requirement is minimal. In a large surface, only a few of the underlying patches will ever require any recomputation. Indeed the performance bottlenecks surround the volume of data being sent to the graphics unit rather than in performing G-Patch calculations. This is not too surprising, as the evaluation does only a small amount of additional work above what a simple Bézier patch tessellator would perform.

Recall that the Bézier control points of the downward pointing patches are generated from the Bézier control points of the upward pointing patches. The additional work is merely the cost to transform the G-Patch control points into Bézier control points. The work required to change representations is proportional to the size of the conversion matrix. A degree n G-Patch will have m control points and an $m \times m$ conversion matrix where $m \in O(n^2)$. Thus, switching representations has an $O(n^4)$ complexity.

To evaluate a single point on the surface of a Bézier patch using the de Casteljau algorithm has a running time of $O(n^3)$. Typically when tessellating a Bézier patch there are at least $\Omega(n^2)$ points on the surface that are evaluated to generate a smooth looking surface. Thus, the cost to convert from an G-Patch to a Bézier patch is at least a factor of n less then the cost to actually render the patch.

Compare this to the cost of evaluating a DMS-Spline. As mentioned before, they are prohibitively expensive to evaluate as the underlying simplex basis functions need to be explicitly evaluated. Add the fact that it is extremely difficult to determine which control points from neighbouring patches contribute to a point on the surface, and this makes the DMS-spline evaluation extremely computation heavy. As such, G-Patch surfaces are much less resource intensive than DMS-splines.

Finally, the evaluation of a G-Patch is numerically stable. All the points on the surface are generated by convex combinations of the original control points. If the conversion matrix C_n is used, it contains only simple fractions which results in stable computations. This is in contrast to DMS-Splines which are plagued with stability issues.

7.2 Continuity

Given the analysis of the previous section, the G-Patch surfaces are highly desirable. The only issue that remains is to look at the resulting surfaces to see if they exhibit the desired continuity.

As mentioned in Section 2.4.3, a network of Bézier patches imposes limits on the degree of continuity that can be achieved while still having local flexibility. So having expectations such as C^2 continuity amongst a collection of cubic patches is unreal.

There will be two methods used to analyze the surfaces. First, examining the Bézier control polygon of neighbouring patches will show what inherent continuity limitations are built into the new surfaces. Finally, by turning on shading in the tool and looking at the resulting surfaces, we will be able to give a subjective analysis.

7.2.1 Bézier Continuity

Knowing how to convert the G-Patch into its Bézier representation allows us to immediately see what continuity can be achieved in the surface.

The G-Patch surface is trivially C^0 due to the procedure by which downward triangular patches are constructed. As well, adjacent upward patches agree at their common corner.

We turn our attention to determining if any of the higher degree G-Patch surfaces exhibit C^1 continuity. The answer is no. It will be sufficient to look at two neighbouring upward pointing G-Patches and the downward patch between them. The three domain triangles, A, B and C are given in Figure 7.1 with the corners of each region of the domain labelled with the points a through e.

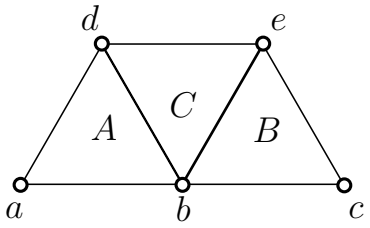

Figure 7.1: Domain labelling of three adjacent patches in an G-Patch surface.

We are concerned with possible configurations the underlying Bézier control points take for these three patches when different G-Patch control points are moved. Figure 7.2 shows the first two layers of Bézier control points around point b, the shared corner of the three patches. Note that this diagram applies to G-Patches of degree greater than or equal to two. A necessary (although not sufficient) condition for the three patches to meet with C^1 continuity is that the grey control points must all be coplanar. Taking this one step further the three control points $f(a, b, \ldots, b)$, $f(b, \ldots, b)$ and $f(b, \ldots, b, c)$ must be collinear. If there is some arrangement of points in a G-Patch control network that cause these three Bézier control points to not be collinear, we will have shown the surface is not C^1.

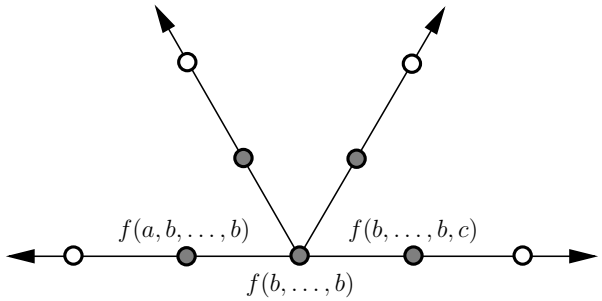

Figure 7.2: Bézier control points for three adjacent patches in an G-Patch surface.

Quadratic G-Patches

We will start with quadratic surfaces. The corresponding control network will have ten control points. The bottom left six control points influence A's patch, while the bottom right six control points influence B's patch. To simplify the discussion, the control points will be laid in a uniform triangular grid in a plane set at $z = 0$. One of the control points will be moved perpendicular to the plane such that its z value is 1. Figure 7.3 shows a top down view of the control points. The grey control point is the one that has been moved out of the plane. Notice that it is the topmost control point that is common to both A and B's patch.

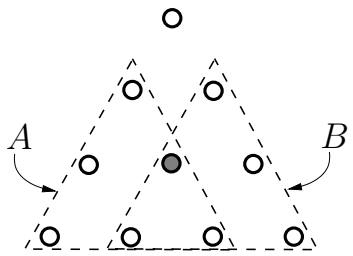

Figure 7.3: Top down projection of quadratic G-Patch surface control points. All the points are coplanar except the grey point which is above the plane.

Now we need to verify that the Bézier control points ($f(a, b)$, $f(b, b)$ and $f(b, c)$) are collinear. For the time, we will restrict our attention to the z coordinate of the three points. We generate the z value of the Bézier control points by multiplying a vector of the z values of the individual G-Patch control points by the conversion matrix (Section 5.3).

Using the C_2 conversion matrix given in Equation 5.4, we generate the z value of A's Bézier points by multiplying

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \cdot C_2 = \begin{bmatrix} \frac{1}{3} & \frac{1}{6} & \frac{2}{3} & 0 & \frac{1}{6} & \frac{1}{3} \end{bmatrix}.$$

Note, the result of the multiplication is simply the third row of the conversion matrix. $f(a,b)$ is the fifth element of the resulting vector, while $f(b,b)$ is the sixth element.

We generate the z value of B's Bézier points by multiplying

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot C_2 = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{1}{6} & \frac{1}{3} & \frac{1}{6} & 0 \end{bmatrix}.$$

Note, this is now the second row of the conversion matrix. $f(b,b)$ is the fourth element of the resulting vector, while $f(b,c)$ is the fifth element.

Summarizing, $f(a,b) = \frac{1}{6}$, $f(b,b) = \frac{2}{6}$ and $f(b,c) = \frac{1}{6}$. However, there is no way the three points can be collinear, as their z values form an inverted "v" shape. The complete set of Bézier control points (in three space) for the A and B patches are shown in Figure 7.4. Clearly the control points are not collinear, and thus the surface is not C^1.

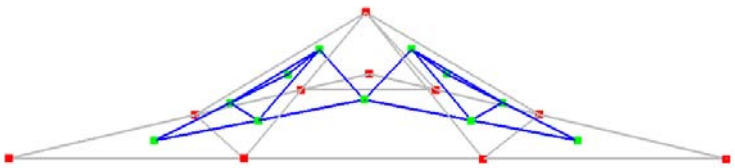

Figure 7.4: Quadratic G-Patch Bézier points not meeting C^1.

Cubic G-Patches

The corresponding G-Patch control network will have fifteen control points. The bottom left ten control points influence A's patch, while the bottom right ten control points influence B's patch (Figure 7.5). Again, the topmost control point that influences both patches is moved out of the plane.

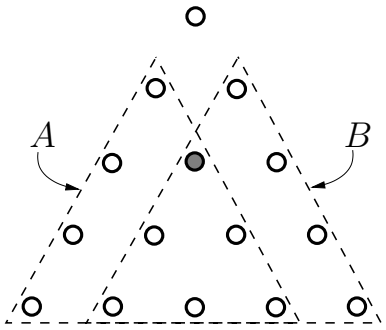

Figure 7.5: Top down projection of cubic G-Patch surface control points. All the points are coplanar except the grey point which is above the plane.

Now we need to determine if the Bézier control points $f(a,b,b)$, $f(b,b,b)$ and $f(b,b,c)$ are collinear. Again, we will consider the z coordinate of the three points.

If we use the cubic conversion matrix C_3 given in Appendix A, then the third row of this conversion matrix will provide the z values of the Bézier control points:

$$\left[\begin{array}{cccccccccc} \frac{4}{15} & \frac{1}{9} & \frac{4}{15} & \frac{1}{45} & \frac{7}{90} & \frac{2}{15} & 0 & \frac{1}{45} & \frac{2}{45} & \frac{1}{15} \end{array} \right].$$

$f(a,b,b)$ is the ninth element of this vector, while $f(b,b,b)$ is the tenth element. Similarly, the z value of B's Bézier points are the second row of C_3:

$$\left[\begin{array}{cccccccccc} \frac{4}{15} & \frac{4}{15} & \frac{1}{9} & \frac{2}{15} & \frac{7}{90} & \frac{1}{45} & \frac{1}{15} & \frac{2}{45} & \frac{1}{45} & 0 \end{array} \right].$$

$f(b,b,b)$ is the seventh element of the resulting vector, while $f(b,b,c)$ is the eighth element.

Summarizing, $f(a,b,b) = \frac{2}{45}$, $f(b,b,b) = \frac{3}{45}$ and $f(b,b,c) = \frac{2}{45}$. As with quadratic patches, the the three points are not collinear, as their z values form an inverted "v" shape, and thus cubic G-Patch surfaces are not C^1.

Higher degree G-Patches

If we follow this example to higher degree patches, the same conclusion is always reached. Moving the topmost point common to both patches causes the three necessary Bézier control points to not be collinear. Thus, G-Patch surfaces in general are not guaranteed to be C^1.

If there is one redeeming factor, it is that as the degree is raised, one must move a G-Patch control point much further to bring the necessary Bézier control points noticeably out of alignment. Consider the quartic G-Patch surface in Figure 7.6. Again, the control point is moved only a unit out of the plane, however the Bézier control points appear to be almost linear.

7.2.2 Shaded G-Patch Surfaces

Knowing that the G-Patch is only guaranteed to be C^0 reduces the expectations we hope to achieve with smoothly shaded surfaces. However, the wire frame surfaces rendered in the previous chapters started to look smooth at higher degrees.

Figures 7.7, 7.8 and 7.9 show the same G-Patch control network rendered with increasing degree G-Patches. As we progress to higher degree patches, there is a dramatic increase in the amount of smoothness between patches. However, even

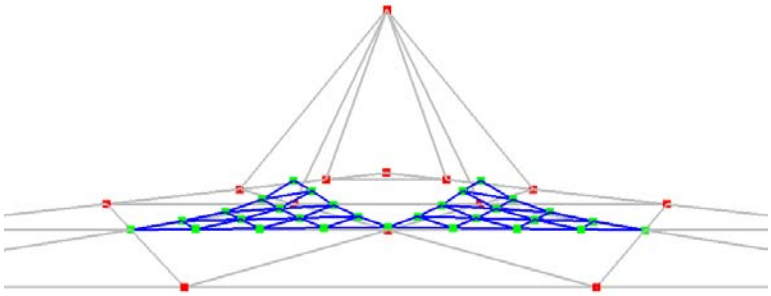

Figure 7.6: Quartic G-Patch Bézier points not meeting C^1.

in the quartic case (Figure 7.9), most of the edges between the underlying patches are still discernable to the eye. If a user is not moving control points too wildly, though, quartic surfaces (and higher) may yield satisfactory results.

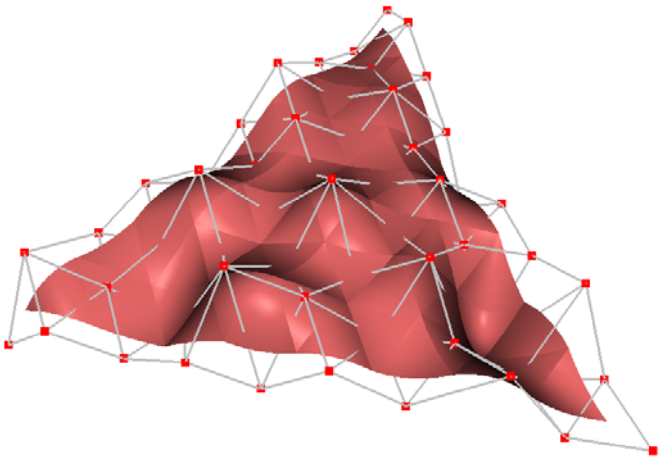

Figure 7.7: A shaded quadratic G-Patch surface and its control net.

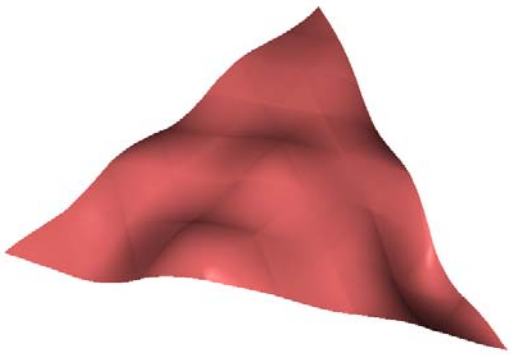

Figure 7.8: A shaded cubic G-Patch surface over the same control net.

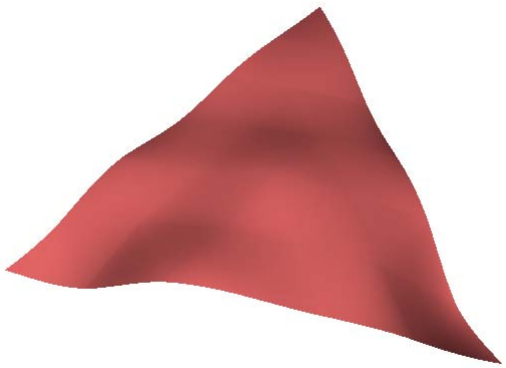

Figure 7.9: A shaded quartic G-Patch surface over the same control net.

Chapter 8

Conclusions

Is the G-Patch network going to replace tensor product B-Splines as the surface modelling tool of choice? Definitely not. However, this was not the goal of the research. Its purpose was to take a purely geometric view of how points are blended with the uniform B-Spline and scale the functions up to the elusive triangular domain.

I summarize the results of the new patch scheme and close with some direction for future work.

8.1 Summary

I have provided a piecewise polynomial surface control scheme that allows the user to control the shape of a network of upward pointing Bézier patches. Each downward pointing hole of the resulting network is then filled in by constructing an additional Bézier patch whose control points are set in positions that allow the patch to meet its three neighbours as smoothly as possible.

This G-Patch network possesses many of the desirable properties discussed in Section 2.1. First and foremost is that the blending formulas reduce to the classic univariate B-Spline construction when the domain is $\mathbb{R}^1$. This leads to G-Patch networks inheriting most of the desirable properties possessed by B-Splines. For instance the G-Patch construction has local control. As well, the evaluation of a point on the surface is fast and numerically stable, owing to a coefficient-based evaluation. Finally, the manipulation of the surface is extremely intuitive, since there are no knot clouds for the user to place. To deform the surface, the user only needs to alter the position of a nearby control point.

The downside is that, in its current form, the G-Patch surface cannot guarantee anything more than C^0 continuity between adjacent Bézier patches, regardless of the degree of the G-Patch network. A small consolation is higher degree G-Patch surfaces require greater disparity in the positions of the control points for this lack of smoothness to be detectable. This single deficiency, though, all but removes the G-Patch from being a viable modelling tool.

8.2 Future Work

It is my hope that the introduction of this new surface construction will guide new research into triangular B-Splines. Since this a new patch scheme, a number of avenues merit further analysis.

The G-Patch control scheme provides a simple way to control a network of Bézier patches, but does so without concern for continuity. It is hoped that a more sophisticated evaluation can be developed that uses the G-Patch control scheme at its root, but builds in higher degrees of continuity. One idea is to take the existing G-Patch control points and perform a level of subdivision hidden from the user.

Using these new control points as degrees of freedom they could be placed such that the resulting surface is smoother, possessing C^1-continuity and higher.

The technique used to specify the patches for the downward pointing triangles is inelegant. A simple blossom-style evaluation to derive control points for the downward triangles would solve this problem. Also, turning the control network upside down results in an entirely different evaluation. Perhaps averaging the two views of the control points would make for a smoother surface, and resolve the asymmetry in the construction.

Another area of future work involves the G-Patch basis functions. Throughout the thesis, the functions used to combine the control points have been referred to merely as "blending functions". Although I did show that they were in fact basis functions up to quartic patches, what is needed is a formal proof that the degree n G-Patch blending functions form a basis. The G-Patch to Bézier patch conversion matrices exhibit remarkable structure. Providing a simple means of determining the conversion matrix C_i, or determining a closed form representation of the blending functions may facilitate such a proof.

The relationship between the G-Patch evaluation and the Bézier patch should also be investigated further. It cannot be an accident that the symmetric polar value for points on the Bézier patches are simply the average of each of the asymmetric G-Patch evaluations.

Finally, the G-Patch construction is based on the notion of a uniform spline. For B-Spline curves, it is a simple matter to move knots in the domain to change the flexibility in regions of the spline. Incorporating the ability to move "knot lines" through the domain to control the new surface warrants further research.

8.3 A Closing Thought

Is the G-Patch the true generalization of the B-Spline? Optimistically I believe it is not; if it were, it would mean there is no higher dimension construction that automatically builds in continuity, and Ramshaw's challenge mentioned in the introduction has no solution. Consequently, the only way to get smooth triangle-patch surfaces would be to necessarily throw expensive computations at the problem as found with DMS-splines.

What the G-Patch hopefully represents is a step towards determining the true triangular B-Spline.

Appendix A

Conversion Matrices

Chapter 5 described two procedures used to generate a conversion matrix, C_n, which translates a set of degree n G-Patch control points to Bézier control points. Rarely would one want to perform this calculation in a live application. It is much simpler to precompute the required matrix once and use it as needed.

The following pages give the cubic and quartic conversion matrices C_3 and C_4.

A.1 Cubic G-Patch Conversion Matrix

$$C_3 = \begin{bmatrix}
\frac{1}{15} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{4}{15} & \frac{4}{15} & \frac{1}{9} & \frac{2}{15} & \frac{7}{90} & \frac{1}{45} & \frac{1}{15} & \frac{2}{45} & \frac{1}{45} & 0 \\
\frac{4}{15} & \frac{1}{9} & \frac{4}{15} & \frac{1}{45} & \frac{7}{90} & \frac{2}{15} & 0 & \frac{1}{45} & \frac{2}{45} & \frac{1}{15} \\
\frac{1}{15} & \frac{2}{15} & \frac{2}{45} & \frac{4}{15} & \frac{7}{90} & \frac{1}{45} & \frac{4}{15} & \frac{1}{9} & \frac{1}{45} & 0 \\
\frac{4}{15} & \frac{2}{5} & \frac{2}{5} & \frac{2}{5} & \frac{8}{15} & \frac{2}{5} & \frac{4}{15} & \frac{2}{5} & \frac{2}{5} & \frac{4}{15} \\
\frac{1}{15} & \frac{2}{45} & \frac{2}{15} & \frac{1}{45} & \frac{7}{90} & \frac{4}{15} & 0 & \frac{1}{45} & \frac{1}{9} & \frac{4}{15} \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{15} & 0 & 0 & 0 \\
0 & \frac{1}{45} & \frac{1}{45} & \frac{1}{9} & \frac{7}{90} & \frac{2}{45} & \frac{4}{15} & \frac{4}{15} & \frac{2}{15} & \frac{1}{15} \\
0 & \frac{1}{45} & \frac{1}{45} & \frac{2}{45} & \frac{7}{90} & \frac{1}{9} & \frac{1}{15} & \frac{2}{15} & \frac{4}{15} & \frac{4}{15} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{15}
\end{bmatrix}$$

A.2 Quartic G-Patch Conversion Matrix

$$C_4 = \begin{bmatrix}
\frac{1}{105} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{11}{105} & \frac{8}{105} & \frac{17}{420} & \frac{4}{105} & \frac{31}{1260} & \frac{1}{90} & \frac{2}{105} & \frac{17}{1260} & \frac{1}{126} & \frac{1}{420} & \frac{1}{105} & \frac{1}{140} & \frac{1}{210} & \frac{1}{420} & 0 \\
\frac{11}{105} & \frac{17}{420} & \frac{8}{105} & \frac{1}{90} & \frac{31}{1260} & \frac{4}{105} & \frac{1}{420} & \frac{1}{126} & \frac{17}{1260} & \frac{2}{105} & 0 & \frac{1}{420} & \frac{1}{210} & \frac{1}{140} & \frac{1}{105} \\
\frac{11}{105} & \frac{2}{15} & \frac{13}{210} & \frac{16}{105} & \frac{1}{14} & \frac{19}{630} & \frac{2}{15} & \frac{1}{14} & \frac{19}{630} & \frac{1}{105} & \frac{11}{105} & \frac{13}{210} & \frac{19}{630} & \frac{1}{105} & 0 \\
\frac{12}{35} & \frac{47}{140} & \frac{47}{140} & \frac{167}{630} & \frac{32}{105} & \frac{167}{630} & \frac{73}{420} & \frac{271}{1260} & \frac{271}{1260} & \frac{73}{420} & \frac{11}{105} & \frac{29}{210} & \frac{47}{315} & \frac{29}{210} & \frac{11}{105} \\
\frac{11}{105} & \frac{13}{210} & \frac{2}{15} & \frac{19}{630} & \frac{1}{14} & \frac{16}{105} & \frac{1}{105} & \frac{19}{630} & \frac{1}{14} & \frac{2}{15} & 0 & \frac{1}{105} & \frac{19}{630} & \frac{13}{210} & \frac{11}{105} \\
\frac{1}{105} & \frac{2}{105} & \frac{1}{140} & \frac{4}{105} & \frac{17}{1260} & \frac{1}{210} & \frac{8}{105} & \frac{31}{1260} & \frac{1}{126} & \frac{1}{420} & \frac{11}{105} & \frac{17}{420} & \frac{1}{90} & \frac{1}{420} & 0 \\
\frac{11}{105} & \frac{73}{420} & \frac{29}{210} & \frac{167}{630} & \frac{271}{1260} & \frac{47}{315} & \frac{47}{140} & \frac{32}{105} & \frac{271}{1260} & \frac{29}{210} & \frac{12}{35} & \frac{47}{140} & \frac{167}{630} & \frac{73}{420} & \frac{11}{105} \\
\frac{11}{105} & \frac{29}{210} & \frac{73}{420} & \frac{47}{315} & \frac{271}{1260} & \frac{167}{630} & \frac{29}{210} & \frac{271}{1260} & \frac{32}{105} & \frac{47}{140} & \frac{11}{105} & \frac{73}{420} & \frac{167}{630} & \frac{47}{140} & \frac{12}{35} \\
\frac{1}{105} & \frac{1}{140} & \frac{2}{105} & \frac{1}{210} & \frac{17}{1260} & \frac{4}{105} & \frac{1}{420} & \frac{1}{126} & \frac{31}{1260} & \frac{8}{105} & 0 & \frac{1}{420} & \frac{1}{90} & \frac{17}{420} & \frac{11}{105} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{105} & 0 & 0 & 0 & 0 \\
0 & \frac{1}{420} & \frac{1}{420} & \frac{1}{90} & \frac{1}{126} & \frac{1}{210} & \frac{17}{420} & \frac{31}{1260} & \frac{17}{1260} & \frac{1}{140} & \frac{11}{105} & \frac{8}{105} & \frac{4}{105} & \frac{2}{105} & \frac{1}{105} \\
0 & \frac{1}{105} & \frac{1}{105} & \frac{19}{630} & \frac{19}{630} & \frac{19}{630} & \frac{13}{210} & \frac{1}{14} & \frac{1}{14} & \frac{13}{210} & \frac{11}{105} & \frac{2}{15} & \frac{16}{105} & \frac{2}{15} & \frac{11}{105} \\
0 & \frac{1}{420} & \frac{1}{420} & \frac{1}{210} & \frac{1}{126} & \frac{1}{90} & \frac{1}{140} & \frac{17}{1260} & \frac{31}{1260} & \frac{17}{420} & \frac{1}{105} & \frac{2}{105} & \frac{4}{105} & \frac{8}{105} & \frac{11}{105} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{105}
\end{bmatrix}$$

Bibliography

[1] R. Burden and J. Faires. *Numerical Analysis*. Brooks/Cole Publishing Company, 1997.

[2] C.A. Dahmen, W.A. Micchelli and H.-P. Seidel. Blossoming begets B-spline bases built better by B-patches. *Mathematics of Computation*, 59(199):97–115, July 1992.

[3] C. de Boor. On calculating with B-splines. *Journal of Approximation Theory*, 6:50–62, 1972.

[4] C. de Boor. *A Practical Guide to Splines*. Springer, 1978.

[5] P. de Casteljau. *Formes à Pôles*. Hermes, Paris, 1985.

[6] G Farin. *Curves and Surfaces for CAGD*. Morgan Kaufmann, fifth edition, 2002.

[7] M.G.J. Franssen. Evaluation of DMS-splines. Master's thesis, Eindhoven University of Technology, 1995.

[8] R.C. Franssen, M. Veltkamp and W. Wesselink. Efficient evaluation of triangular b-spline surfaces. *Computer Aided Geometric Design*, 17:863–877, 2000.

[9] J Gallier. *Curves and surfaces in geometric modelling*. Morgan Kaufmann, 2000.

[10] R Goldman. *Pyramid Algorithms*. Morgan Kaufmann, 2003.

[11] G. Greiner and H.-P. Seidel. Modeling with triangular B-splines. *IEEE Computer Graphics and Applications*, 14:56–60, 1994.

[12] M. Neamtu. What is the natural generalization of univariate splines to higher dimensions? *Mathematical Methods for Curves and Surfaces: Oslo 2000*, pages 355–392, 2001.

[13] R.F. Pfeifle. *Approximation and Interpolation using Quadratic Triangular B-Splines*. PhD thesis, University Erlangen-Nurnberg, 1995.

[14] L. Ramshaw. Blossoming: A connect-the-dots approach to splines. Technical report, Digital Equipment Corporation, June, 1987.

[15] H.-P. Seidel. Symmetric recursive algorithms for surfaces: B-patches and the de Boor algorithm for polynomials over triangles. *Constructive Approximation*, 7:257–279, 1991.