Geometric B-Splines Over Triangular Domains

Christopher K. Ingram, Stephen Mann

Abstract. B-splines are a versatile curve modeling scheme. However, generalizing this technique to surface patches over the triangular domain has proven to be difficult. In this paper we develop a geometric generalization of the control point blending functions used to evaluate uniform B-Spline curves to the triangular domain.

§1. Introduction

The B-Spline manages a collection of degree n curve segments, where each segment meets its neighbor with up to C^{n-1} continuity. This continuity is provided automatically, with each segment sharing numerous control points with its neighbor.

A similar scheme is desired for triangular surface patches. Namely, the development of a control structure that automatically maintains high degrees of continuity between neighbouring degree n triangular patches. While it is known that C^k-continuity, where $k < \frac{2n-1}{3}$, is the highest level of continuity theoretically achievable with local flexibility [4], we as yet do not have any insight into what the particular control scheme is.

Most generalizations of higher order B-Splines attempt to find some method of attributing knots to each of the control points in the surface [1, 3]. The result is that the simple knot vector of the low order splines becomes a knot cloud in the higher order surfaces. However, we 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.

We will introduce a new patch scheme over an equilateral triangulated domain, G-Patches, whose construction attempts to generalize the geometry of a uniform B-Spline curve. A single degree n surface patch is associated with each equilateral triangle in the domain.

§2. Traditional Algorithms

Before introducing our new construction, it will be helpful to examine the classic constructions for Bézier curves, B-Splines and triangular Bézier patches. These algorithms will be presented in a manner that helps develop the ideas behind the new patches.

A degree n Bézier curve is defined by $n+1$ control points, $P_0 \ldots P_n$. 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

$$F(u) = \sum_{i=0}^{n} P_i \binom{n}{i} {\beta_1}^{n-i} {\beta_2}^{i}.$$

We can use blossoming [4] 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

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

By repeatedly blending neighbouring control points, we can determine the point on the curve $F(u)$. Figure 1 shows the de Casteljau algorithm being run on a set of cubic Bézier control points.

Our interest in the de Casteljau algorithm is with the ratio of the lengths of the line segments $\overline{au} : \overline{ub}$. Note, this same ratio is used for each blend in the evaluation.

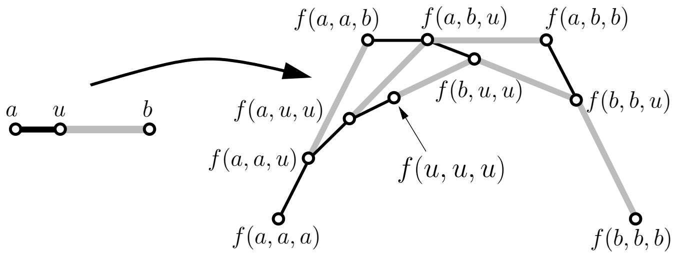

Fig. 1. Running the de Casteljau algorithm on a cubic Bézier curve. For each blend, the ratio of the lengths between $\overline{au}$ (shown in black) and $\overline{ub}$ (shown in grey) is used.

We turn our attention to the construction of a uniform B-Spline using the de Boor algorithm. We will focus on the ratios used to blend control points, and see how they relate to the ratios used in the de Casteljau algorithm. Consider the simple case where we have a degree two spline.

Here, the knot vector will be defined as $\{0, 1, 2, 3\}$. The knots are used as blossom arguments when defining the three control points $P_0 = f(0, 1)$, $P_1 = f(1, 2)$, $P_2 = f(2, 3)$. We will evaluate the image of a point u over the interval $1 \le u \le 2$ in the domain.

To generate a point on the curve, we perform convex combinations of neighbouring control points (Figure 2). 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, the first two control points are blended using the barycentric coordinates of u in terms of 0 and 2 (yielding a new point A), and the second pair are blended using the barycentric coordinates of u relative to 1 and 3 (yielding a new point B). Finally, A and B are blended with u's barycentric coordinates relative to 1 and 2.

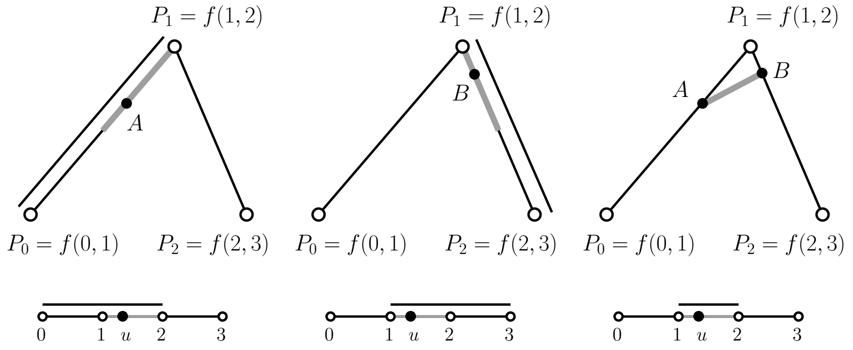

Fig. 2. Running the de Boor algorithm on a quadratic B-Spline curve. For each blend, the region where u's image may appear is shown in grey.

It is important to take a look at where the image of u will be during this process. As we vary u from 1 to 2 for the first two blends, the image of u always lies in the half of the line nearest the middle control point, P_1. Figure 2 highlights these valid locations on the line in grey. At this time it does not matter what specific knots are being used in the evaluation, the computation is always the same. The only difference between this evaluation and the de Casteljau algorithm is rather than always interpolating over the whole region between two control points, except for the final blend, only part of the region is used.

The Bézier curve formulation can be generalized to triangular shaped surface patches. A degree n Bézier patch is defined using a triangular layout of $\binom{n+2}{2}$ control points: $P_{i,j,k}$: i, j, $k \ge 0$ 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

$$F(u) = \sum_{\substack{i,\ j,\ k \geq 0 \\ i+j+k=n}} P_{i,j,k} \frac{n!}{i!j!k!} {\beta_1}^i {\beta_2}^j {\beta_3}^k.$$

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

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

Three neighbouring control points that form an upward pointing triangle agree in $n-1$ of their blossom arguments and can be blended using u's barycentric coordinates relative to $\triangle abc$. By repeatedly blending triples of neighbouring control points with the same barycentric coordinates, we can determine the point on the patch $F(u)$. Figure 3 shows the de Casteljau algorithm being run on a quadratic control point network.

Similar to the curve case, the same ratio is used for each blend during the evaluation.

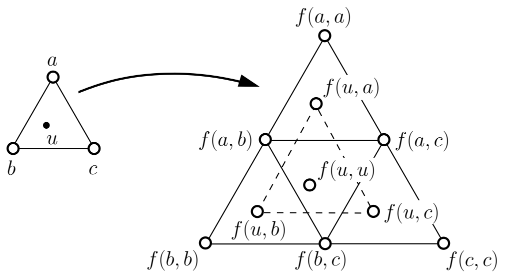

Fig. 3. Running the de Casteljau algorithm on a quadratic Bézier patch.

§3. Merging Two Ideas – The G-Patch

The triangular Bézier patch scales the idea of repeatedly blending neighbouring control points from the linear domain, to triangular domain regions. The B-Spline takes the idea of interpolating over the entire segment between two control points, and uses varying smaller sized regions of the line segment. This leads to the idea of taking the Bézier patch evaluation and not always interpolating over the entire triangular region formed between neighbors, but selecting smaller pieces of each region. The result is the G-Patch.

We will start with the quadratic G-Patch. The control network is determined over the same shaped network as the quadratic Bézier patch, however we will use a simpler labeling of the control points, $P_{i,j} : 0 \leq j \leq i \leq n$. The only issue is to determine the appropriate subregion of each upward pointing triangle to use for each blend. We present one possibility in Figure 4 that uses the inner third of each set of neighbors. Here "inner" implies towards the center of the patch. The final point on the surface would be determined by interpolating u over the full region formed by $\triangle ABC$, analogous to the final blend of the B-Spline evaluation.

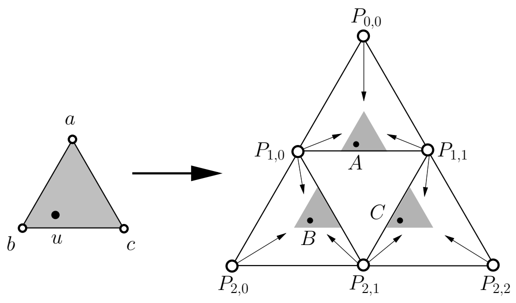

Fig. 4. The valid subregions for interpolating u's image in the quadratic G-Patch.

To perform the above computation, we need to determine the location of the corners of the subregion used to interpolate. Fortunately, these points never need to be explicitly computed. The original barycentric coordinates of u can be used to directly generate the intermediate point.

For example, consider the quadratic B-Spline computation. If u's barycentric coordinates relative to the domain interval are β_1 and β_2, then we can compute the first two blends for points A and B with

$$\begin{aligned} A &= \frac{\beta_1 + 0}{2} P_0 + \frac{\beta_2 + 1}{2} P_1 \\ B &= \frac{\beta_1 + 1}{2} P_1 + \frac{\beta_2 + 0}{2} P_2. \end{aligned}$$

Similarly, for the quadratic G-Patch case, if u's barycentric coordinates relative to the domain triangle are β_1 and β_2, then we can compute the first three blends for A, B and C with

$$\begin{aligned} A &= \frac{\beta_1 + 0}{3} P_{0,0} + \frac{\beta_2 + 1}{3} P_{1,0} + \frac{\beta_3 + 1}{3} P_{1,1} \\ B &= \frac{\beta_1 + 1}{3} P_{1,0} + \frac{\beta_2 + 0}{3} P_{2,0} + \frac{\beta_3 + 1}{3} P_{2,1} \\ C &= \frac{\beta_1 + 1}{3} P_{1,1} + \frac{\beta_2 + 1}{3} P_{2,1} + \frac{\beta_3 + 0}{3} P_{2,2}. \end{aligned}$$

The resulting quadratic curves and patches that are created are given in Figure 5. In both cases the curve and surface are contained inside the control net, and do not interpolate the exterior control points.

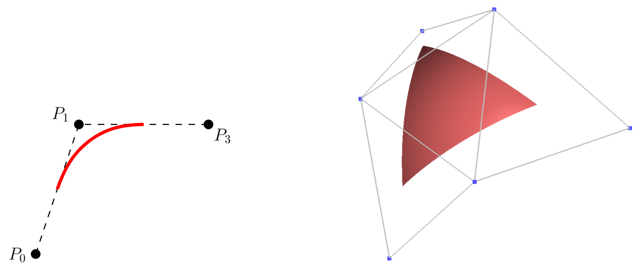

Fig. 5. One piecewise polynomial for the B-Spline (left) and the G-Patch (right).

We would like to extend the G-Patch to higher degrees. As before, it helps to first examine the B-Spline. We will start with the uniform cubic B-Spline, and determine the valid regions where u is interpolated over the line segments between neighbouring control points. Figure 6 highlights in grey the valid locations of u's image on all three lines. Notice that u's image always falls in a particular third of each line segment. The resulting three points are then blended using the quadratic blending functions. For higher degree curves, the first level blending involves the interpolation of smaller regions between control point neighbors.

This observation implies that for the cubic G-Patch, smaller regions of each upward pointing triangle should be used for the initial blends. The only issue is to determine the appropriate subregions to use. We present one possibility in Figure 6 that uses smaller regions which are, again, nearer to the center of the control network. The six points produced are then blended as was done in the quadratic case.

Once again, we do not wish to explicitly compute the endpoints of each subregion to interpolate, and again we can use the original barycentric coordinates to compute the first level of new control points. For the cubic B-Spline we use the following three blending functions

$$\begin{aligned} A &= \frac{\beta_1+0}{3}P_0 + \frac{\beta_2+2}{3}P_1 \\ B &= \frac{\beta_1+1}{3}P_1 + \frac{\beta_2+1}{3}P_2 \\ C &= \frac{\beta_1+2}{3}P_1 + \frac{\beta_2+0}{3}P_2. \end{aligned}$$

For the cubic G-Patch, the blending functions for the leftmost three in-

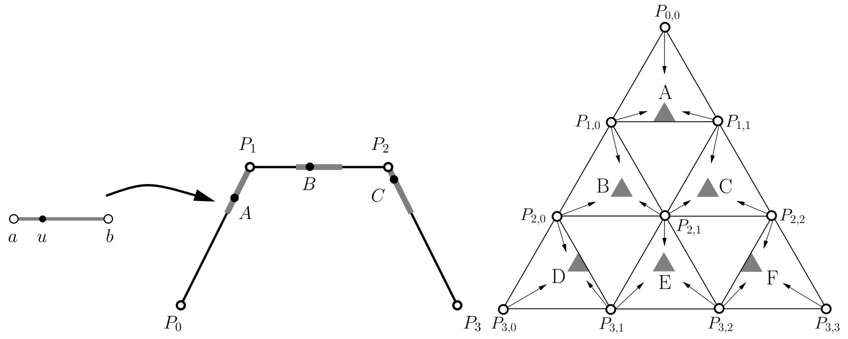

Fig. 6. The valid subregions for interpolating u's image in the cubic B-Spline (left) and the cubic G-Patch (right).

termediate points are given by

$$\begin{aligned} A &= \frac{\beta_1+0}{5}P_{0,0} + \frac{\beta_2+2}{5}P_{1,0} + \frac{\beta_3+2}{5}P_{1,1} \\ B &= \frac{\beta_1+1}{5}P_{1,0} + \frac{\beta_2+1}{5}P_{2,0} + \frac{\beta_3+2}{5}P_{2,1} \\ D &= \frac{\beta_1+2}{5}P_{2,0} + \frac{\beta_2+0}{5}P_{3,0} + \frac{\beta_3+2}{5}P_{3,1} \end{aligned}$$

with the other three blending functions being similarly specified [2].

Looking at the formulas for the blending functions of G-Patches of varying degrees, a pattern emerges. Given a degree n triangular network of control points and the barycentric coordinates of u, we can easily compute $F(u)$ using a de Boor style dynamic programming algorithm [2].

§4. Bézier Representation and Basis Functions

For a B-Spline, each piecewise polynomial is a degree n curve and any piece of the domain can be converted into a corresponding Bézier curve. For G-Patches, we can construct a change of basis matrix, C_n, to convert to Bézier form [2]. A key feature of the matrix is that each Bézier control point is a convex combination of the original G-Patch control points. An example of a G-Patch with its Bézier representation is given in Figure 7.

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. We 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

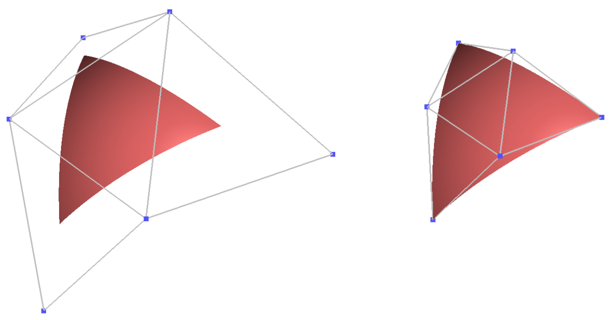

Fig. 7. The same quadratic patch with G-Patch control points (left) and Bézier control points (right).

domain. A point on the surface of the patch is represented by

$$F(u) \quad = \quad \sum_{i=0}^{n} \sum_{j=0}^{i} I_{i,j}^{n}(u) P_{i,j},$$

where u is expressed as the barycentric coordinates relative to the domain triangle. This notation hides many of the details outlined in the previous chapter, but resembles the form often used to define B-Splines.

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.

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

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

Theorem 1. *The set of degree n G-Patch blending functions $I_{i,j}^{n}(u)$ 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_n is invertible for $0 \le n \le 4$ [2]. $\square$

We conjecture that this property will hold for all n.

§5. G-Patch Surfaces

Recall that the primary goal for a control scheme is to maintain a collection of triangular patches. The final step is to show how to form a surface from a network of G-Patches where adjacent patches share control points.

5.1. Upward Pointing Triangular Patches

For this discussion, we will consider quadratic patches defined over the equilateral triangulated domain in Figure 8. There are three upward pointing triangles and one downward pointing triangle labeled A through D. A single G-Patch will be defined for each region.

Each upward triangle will have a set of six G-Patch control points associated with it. To maximize control point reuse, we will have A's bottom right control points be reused as B's bottom left control points. The corresponding control points for A and B are outlined in Figure 8. By further reusing control points, the G-Patch for domain triangle C only requires one additional control point. Furthermore, the central grey control point is shared by all three patches, implying that moving it will have an effect on all three patches.

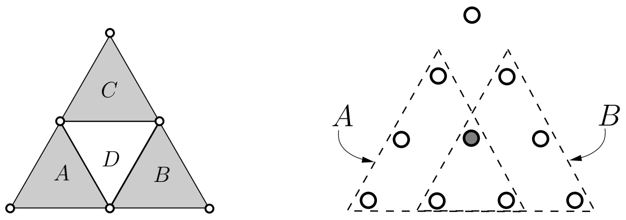

Fig. 8. The domain partitioned into four regions (left) and the corresponding network of quadratic G-Patch control points for domain regions A, B and C (right).

Scaling this to larger surfaces is easily accomplished. Should more patches of the same degree in any parametric direction be desired, it only requires adding additional rows of control points. If higher degree patch networks are needed, the same technique is used, differing only in more control points being shared between neighbouring patches.

Due to the choices made for the underlying blending functions, the resulting G-Patches will necessarily meet at their corners. Figure 9 shows a wire-frame rendering of three upward pointing G-Patches.

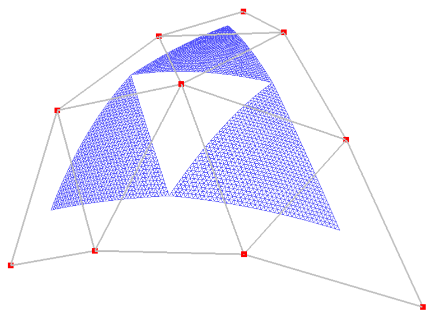

Fig. 9. Three upward pointing G-Patches and the corresponding network of control points.

One question is how does the G-Patch construction relate to the B-Patch construction [5]? The B-Patch has blossom labels that allow for a de Casteljau style algorithm, while there is no satisfactory labeling of the G-Patch control points. However, B-Patches suffer from an inability to have neighboring patches share control points and corresponding knot clouds while still maintaining even C^0-continuity. The G-Patch is built on the foundation of sharing control points, and achieves C^0-continuity automatically.

5.2. Downward Pointing Triangular Patches

While it is easy to find control points for the upward pointing patches, it is not clear how to specify the downward patches. Of immediate concern, is that in Figure 8 there does not exist a set of six control points which form a downward pointing G-Patch . Even with larger networks of control points, the downward pointing triangles are not lined up to correctly fill the holes left by the upward pointing patches.

With the Bézier representation of each upward pointing patch, however, we can construct a downward pointing Bézier patch using standard techniques. In Figure 10, the original network of three G-Patches is shown with its Bézier control points specifying a new downward pointing triangle. Similar hole filling can occur with higher degree patches. A quartic G-Patch surface with 25 individual patches is given in Figure 11.

§6. Continuity

The only issue that remains is to look at the resulting surfaces to see if they exhibit the desired continuity. The G-Patch surface is trivially C^0 due to the procedure by which downward triangular patches are constructed. Knowing how to convert the G-Patch into its Bézier representation allows us to see what additional levels of continuity can be achieved.

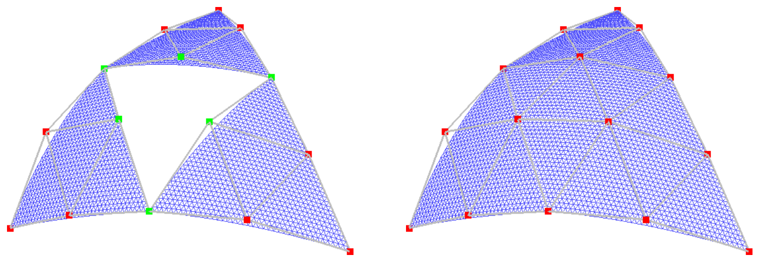

Fig. 10. Three upward pointing quadratic G-Patches shown with their Bézier control points (left). The six control points around the hole specify an additional downward pointing Bézier patch (right).

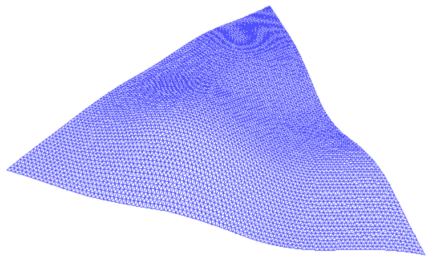

Fig. 11. A network of 25 quartic G-Patches specified by 45 control points.

It will be sufficient to look at two neighbouring upward pointing G-Patches, A and B, from Figure 8. Suppose all the G-Patch control points are co-planar except for the grey control point which is pulled out of the plane. Figure 12 shows a rotated view of this scenario, with the G-Patch control points shown in white. Consider the corresponding Bézier control points (black and grey) for the two patches. In particular the three points along the bottom edge where they meet (grey). A necessary C^1-continuity condition is for these three Bézier control points to be co-linear. However, the scenario described assures that the three points form a “v”-shape (heavy black line), precluding any chance of C^1-continuity. Moreover, regardless of the degree of the G-Patch network, there is always a scenario where moving a single control point disrupts the C^1-continuity at this location.

§7. Conclusions

We have presented a geometric generalization of the B-Spline curve scheme to triangular domains. It has a simple, stable evaluation, but it fails to

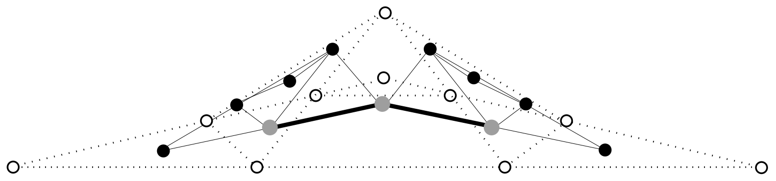

Fig. 12. Two neighbouring G-Patches, and their corresponding Bézier control points viewed along the horizon. The dotted lines represent the G-Patch control network, while the solid lines represent the two Bézier patch control network.

guarantee high levels of continuity. Thus, G-Patches are not the generalized triangular B-Spline being sought. However, we feel they represent a step towards determining a true triangular B-Spline scheme. Our choice of the smaller triangular regions to interpolate over during the G-Patch evaluation is just one possible scheme. A different choice, one that automatically takes into account the downward pointing patches could lead to higher degrees of continuity in the network.

§8. References

1. Dahmen, W. A., C. A. Micchelli, and H.-P. Seidel, Blossoming begets B-spline bases built better by B-patches, Math. Comp. **59** (1992), 97–115.
2. Ingram, C. K., *A Geometric B-Spline Over the Triangular Domain*, Master's Thesis, University of Waterloo, 2003.
3. Neamtu, M., What is the natural generalization of univariate splines to higher dimensions?, in *Mathematical Methods in CAGD: Oslo 2000*, T. Lyche and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 2001, 355–392.
4. Ramshaw, L., *Blossoming: A connect-the-dots approach to splines*, Digital Equipment Corporation, June, 1987.
5. Seidel, H.-P., Symmetric recursive algorithms for surfaces: B-patches and the de Boor algorithm for polynomials over triangles, Constr. Approx. **7** (1991), 257–279.

Christopher K. Ingram, Stephen Mann
University of Waterloo
Waterloo, Ontario, Canada
`c2ingram@uwaterloo.ca`, `smann@uwaterloo.ca`
`http://www.cgl.uwaterloo.ca/{∼c2ingram, ∼smann}`