Faster Evaluation of Quadratic Bivariate DMS Spline Surfaces

Ron Pfeifle and Hans-Peter Seidel
Computer Graphics Group, University of Erlangen
Am Weichselgarten 9, D-91058 Erlangen, Germany
email: {pfeifle,seidel}@informatik.uni-erlangen.de

Abstract

We present a scheme for efficiently evaluating bivariate quadratic simplex splines in the context of the new B-spline scheme developed by Dahmen, Micchelli and Seidel [DMS92]. An algorithm is presented, which is based on the careful reuse of the partial results that arise when recursively evaluating simplex splines. The method is compared with previous methods. A test implementation written in "C" is found to execute 2.3 times faster than another recent implementation not employing this algorithm.

Keywords: Spline Surfaces, Evaluation Algorithms, Control Points, Recursive Evaluation, Multivariate Splines, Simplex Splines.

1 Introduction

Surface modelling has been an important activity in CAGD for many years. The most successful surface modelling schemes to date are curve techniques that have been extended in order to represent surfaces, namely the *Tensor Product B-spline* and *Tensor Product Bézier* methods [BBB87, Far93].

Unfortunately, these extended techniques are not without flaws. Since the surface patches that emerge from these schemes are essentially rectangular, it is difficult to model more complex shapes. *Triangular Bézier* patches [Far93], on the other hand, can be used to represent more general surface patches. This method can be used to define surfaces over arbitrarily shaped (polygonal) domains. However, automatic maintenance of continuity properties is not possible under this representation.

Simplex spline methods [DM82, Höl82, DMS92] overcome these difficulties by being able to represent arbitrarily shaped domains, while automatically maintaining continuity properties between different sections of a surface. The problem that these methods exhibit is that they are computationally more expensive.

This paper addresses the computational expense of evaluating B-spline surfaces built with the help of simplex splines. We present an algorithm that accelerates the evaluation of quadratic simplex spline surfaces, by keeping track of partial results and reusing them later during evaluation when possible.

The first part of the paper describes the mathematics involved, and explains our method. Section 2 reviews the definition of simplex splines and that of the DMS Spline scheme [DMS92, Sei91]. In Section 3, we highlight previous methods for the evaluation of simplex splines, explain our new method in detail, and discuss its advantages and disadvantages with respect to earlier methods.

The second part of the paper discusses implementation of the algorithm in theory and in practice. Section 4 outlines the number of operations involved in each step of the algorithm, comparing them to a more naive evaluation algorithm. In Section 5, the actual CPU usage of this algorithm is compared with that of a recent implementation that does not use our method. Finally, we present our conclusions and suggestions for further work in Section 6.

2 Review of Bivariate B-splines

We begin by recalling the definition of the bivariate simplex spline and its recurrence [Mic79], then we examine the DMS Spline scheme of [DMS92] in a bivariate quadratic setting. A good introduction to DMS Splines can be found in [Sei91].

2.1 Simplex Splines

Let $V = \{t_0, \ldots, t_{n+2}\}$, a collection of knots, and u, an arbitrary point, be taken from $\mathbb{R}^2$. Then we define the simplex spline $M(u|V)$ as follows: For $V = \{t_0, t_1, t_2\}$,

$$M(u|t_0,t_1,t_2) = \frac{\chi_{[t_0,t_1,t_2)}(u)}{2|\Delta(t_0,t_1,t_2)|}, \quad (1)$$

$$\text{where } \chi_{[t_0,t_1,t_2)} = \begin{cases} 1 & \text{if } u \in [t_0,t_1,t_2) \\ 0 & \text{otherwise} \end{cases}$$

is the characteristic function on the half-open convex hull $[t_0, t_1, t_2)$ [1].

For $V = \{t_0, \ldots, t_{n+2}\}$, $n > 0$, select three points $W = \{t_{i_0}, t_{i_1}, t_{i_2}\}$ from V, such that W is affinely independent. Then

$$M(u|V) = \sum_{j=0}^{2} \lambda_j(u|W) M(u|V \backslash \{t_{i_j}\}) \quad (2)$$

where $\lambda_j(u|W)$ are the barycentric coordinates of u with respect to the points of W. Although W is almost completely arbitrary, $M(u|V)$ is well-defined.

Simplex splines possess a number of properties useful for geometric modelling:

- Piecewise Polynomial: Simplex splines are piecewise polynomials of degree n.
- Locality: For points u outside the convex hull of V, $M(u|V) \equiv 0$.
- Non-negativity: $M(u|V) \geq 0$.
- Smoothness: For vertices $t_i \in V$ in general position, $M(u|V)$ exhibits C^{n-1} continuity.

2.2 The DMS Spline Scheme

The DMS Spline scheme described in [DMS92] makes use of selected simplex splines over a triangulation in order to form smooth piecewise polynomial surfaces.

Let $T = \{\Delta(I) = [t_{i_0}, t_{i_1}, t_{i_2}] | I = (i_0, i_1, i_2) \in \mathcal{I} \subset \mathbb{Z}^3_+\}$ be an arbitrary triangulation of $\mathbb{R}^2$ or some bounded domain $D \subset \mathbb{R}^2$. Then given two domain triangles $I, J \subset \mathcal{I}$, we have that $\Delta(I) \bigcap \Delta(J)$ is either empty, or is a common vertex or edge of $\Delta(I)$ and $\Delta(J)$.

[1] A point u is in the half-open convex hull of $\{t_0, t_1, t_2\}$, if there exists $\epsilon > 0$ such that the set $\{u + s\eta + t\xi \mid s, t > 0,\ s + t < \epsilon\}$ lies entirely within the convex hull of those points, where ξ is the horizontal unit vector in $\mathbb{R}^2$ and η a vector with positive slope.

To each vertex t_i is assigned a sequence of vertices $t_{i,1}, \ldots, t_{i,n}$, called its *knot cloud*, with $t_{i,0} = t_i$. They are assigned such that if domain triangle $\Delta = \Delta(I)$ has vertices t_0, t_1, t_2, then each set $\{t_{0,i}, t_{1,j}, t_{2,k}\}$ is affinely independent for all $i, j, k = 0, \ldots, n$. Each vertex $t_{i,l}$ will generally be referred to as a *knot*.

From these knots, we build simplex splines $M(u|V^{\Delta}_{i,j,k})$ for each domain triangle Δ, and multi-index i, j, k, where $i + j + k = n$, and

$$V^{\Delta}_{i,j,k} = \{t_{0,0}, \ldots, t_{0,i}, t_{1,0}, \ldots, t_{1,j}, t_{2,0}, \ldots, t_{2,k}\} \quad (3)$$

for $i, j, k \geq 0$.

The *normalized B-splines* are then defined as $N^{\Delta}_{i,j,k}(u) = d^{\Delta}_{i,j,k} M(u|V^{\Delta}_{i,j,k})$, with $d^{\Delta}_{i,j,k} > 0$ being twice the area of $\Delta(t_{0,i}, t_{1,j}, t_{2,k})$. They form a global partition of unity.

A surface F of degree n over the triangulation T with knot net $\mathcal{K} = \{t_{i,l} | i \in \mathbb{Z}, l = 0, \ldots, n\}$ is then defined as

$$F(u) = \sum_{\Delta \in T} \sum_{i+j+k=n} c^{\Delta}_{i,j,k} N^{\Delta}_{i,j,k}(u), \quad (4)$$

where $c^{\Delta}_{i,j,k} \in \mathbb{R}^3$ individually are the *control points*, and collectively form the *control net* of the surface F.

B-spline surfaces satisfy the following properties:

- Affine Invariance: In order to transform the entire surface affinely (rotation, translation, scaling, etc...), we need only transform its control points.
- Convex Hull Property: $F(u)$ lies within the convex hull of the control points.
- Local Control: Altering the position of $c^{\Delta}_{i,j,k}$ only affects the parts of the surface defined over Δ and immediately surrounding triangles.
- Piecewise Polynomial Representation: *All* piecewise polynomials of degree n over the triangulation T can be represented this way, giving us a large set of functions with which to model surfaces.

3 The Evaluation Algorithm

A number of methods for evaluating simplex splines based on the recurrence formula (2) are found in the literature. We begin with a review of them and then present our method in detail.

3.1 Previous Methods

The recurrence formula provides a means for evaluating individual simplex splines at a point u in the triangulated domain. Each simplex spline can be expressed as a sum of three simplex splines of lower degree, themselves defined over fewer knots. Each of these, in turn, can also be recursively evaluated until we finally arrive at piecewise constant functions over groups of three knots.

At each stage of the recursion, we are free to choose any three knots t_i, t_j and t_k of the set of knots $t_0, \ldots, t_n$ of $M(u|t_0, \ldots, t_{n+2})$ in order to form the recursion when the triangle formed by t_i, t_j and t_k is not degenerate.

One important question is *which* three knots we should choose at each stage of the recursion. This question is compounded when we consider various *spline spaces*, where the basis functions for a spline space are constructed from simplex splines that share many knots [DM82, Höl82, DMS92].

Because the basis functions in each of these splines spaces share so many knots with each other, the recurrence formula (2) encourages the conviction that many partial results could be reused, given a clever choice of knots for recursion. This observation has the potential of speeding up a recursive evaluation scheme.

The naive approach does not consider partial result reuse: For some suitable choice of knots, all subsplines are completely re-evaluated at each stage of the recursion, regardless of whether or not these values have already been computed.

Grandine [Gra87] discusses the reuse of partial result in connection with the spline space described in [DM82, Höl82]. He characterizes which splines of lower degree contribute to more than one spline of higher degree. Unfortunately, Grandine discovered that the bookkeeping costs involved in storing and retrieving already computed results were higher than simply recomputing them. Grandine attributes this to the difficulty in "naming" partial results– essentially, a sub-spline is identified by the many knots over which it is defined.

Gmelig Meyling [GM86] also discusses the reuse of partial results in connection with the same spline space, for the quadratic bivariate case. The algorithm defined there only performs one step of the recursion, evaluating the linear simplex splines directly rather than by using the recurrence. This method takes note of which partial results can be reused, and does so when possible. Gmelig Meyling also exploits the existence of a region within a domain triangle where a special form of the recurrence may be used. This special recurrence only uses subsplines that are evaluated more than once.

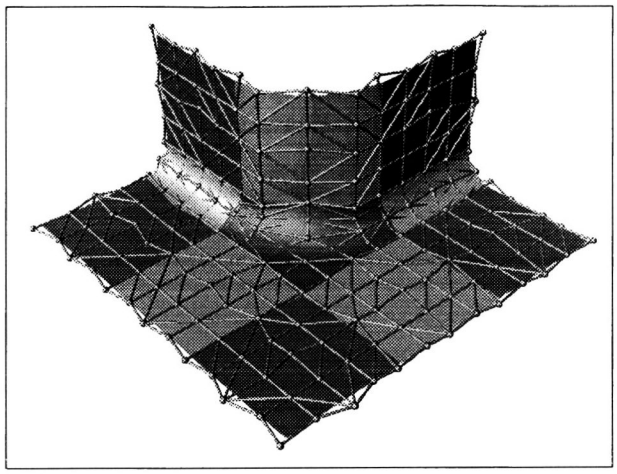

Figure 1: Filling a five-sided hole.

3.2 A New Method

Here we examine the possibility of reusing partial results for the quadratic bivariate splines defined in Section 2.2. In order to simplify the upcoming discussion, we will consider only a single domain triangle $\Delta = \Delta(r, s, t)$, with knot clouds $\{r_0, r_1, r_2\}$, $\{s_0, s_1, s_2\}$, and $\{t_0, t_1, t_2\}$.

Part of our goal will be to assign multi-indices to each simplex spline that we use. These multi-indices guide the selection of knots during recursive evaluation.

Let us first define $M_{i,j,k}(u) \equiv M(u|V_{i,j,k})$; for example, $M_{2,0,0}(u) = M(u|r_0, r_1, r_2, s_0, t_0)$. Thus we have $N_{i,j,k}(u) = d_{i,j,k} M_{i,j,k}(u)$.

After having introduced indices that identify our simplex splines, it becomes tempting to rewrite Equation 2 in terms of index reduction, that is, in the form

$$\begin{aligned} M_{i,j,k}(u) \quad = \quad & \lambda_0(u|W_{i,j,k}) M_{i-1,j,k}(u) + \\ & \lambda_1(u|W_{i,j,k}) M_{i,j-1,k}(u) + \\ & \lambda_2(u|W_{i,j,k}) M_{i,j,k-1}(u) \end{aligned} \tag{5}$$

where $W_{i,j,k} = \{r_{i,j,k}, s_{i,j,k}, t_{i,j,k}\} \subset V_{i,j,k}$. If we assume for the moment that sets $V_{i,j,k}$ and $W_{i,j,k}$ can be appropriately defined, then it is clear which partial results can be reused. In fact, this leads to the pattern of reuse shown in Figures 2 and 3.

When i, j and k are positive, this formula does indeed hold true; every set $V_{i,j,k}$ is well-defined and

$W_{i,j,k} = \{r_i, s_j, t_k\}$. Nevertheless, negative indices appear very quickly; consider the recursive evaluation of $M_{2,0,0}$ by this definition. Thus, we are left with the problem of defining, in a consistent fashion, the sets $V_{i,j,k}$ and $W_{i,j,k}$ when one or more of i, j and k are not positive.

We will concern ourselves with the definition of appropriate sets $V_{i,j,k}$. Once these have been determined, the sets $W_{i,j,k}$ can be found using the equations:

$$\begin{aligned} \{r_{i,j,k}\} &= V_{i,j,k} \setminus V_{i-1,j,k}\,, \\ \{s_{i,j,k}\} &= V_{i,j,k} \setminus V_{i,j-1,k}\,, \\ \{t_{i,j,k}\} &= V_{i,j,k} \setminus V_{i,j,k-1}\,. \end{aligned}$$

We start by extending our notation for collections $V_{i,j,k}$ (Equation 3) to include indices with value -1. If an index is equal to -1, then no knots from the corresponding knot cloud appear in the collection $V_{i,j,k}$. For example, $V_{2,0,-1} = \{r_0, r_1, r_2, s_0\}$. Although this is an arbitrary choice, it is a natural extention of our notation and allows us to consistently use Equation 5 to recurse from quadratic to linear simplex splines.

In order to recurse from linear to piecewise constant splines, we must now define $V_{i,j,k}$ for $i+j+k = 0$ and $i, j, k >= -2$. The specific cases we need to consider in order to complete our recursion scheme are (up to a permutation of the indices) $V_{0,0,0}$, $V_{1,-1,0}$ $V_{2,-1,-1}$, $V_{1,1,-2}$ and $V_{2,-2,0}$. Of these, $V_{0,0,0}$, $V_{1,-1,0}$ $V_{2,-1,-1}$ have already been defined, leaving $V_{1,1,-2}$ and $V_{2,-2,0}$.

$\boxed{V_{1,1,-2}}$ This set must be defined in order to evaluate the linear simplex spline $M_{1,1,-1}$ (see Figure 3). $M_{1,1,-1}$ has the knot set $V_{1,1,-1} = \{r_0, r_1, s_0, s_1\}$. The sets $V_{0,1,-1}$ and $V_{1,0,-1}$ are formed by deleting from $V_{1,1,-1}$ the knots r_1 and s_1, respectively. In order that $W_{1,1,-1}$ be properly defined, $V_{1,1,-2}$ must either be the set $\{r_1, s_0, s_1\}$ or $\{r_0, r_1, s_1\}$. We choose to "steal" a knot from the cloud corresponding to the index "previous"[2] to the negative index, that is, we will drop s_0 from $V_{1,1,-1}$, giving $V_{1,1,-2} = \{r_0, r_1, s_1\}$.

$\boxed{V_{2,-2,0}}$ Similarly to the previous case, we find that $V_{2,-2,0}$ is constrained to be either the set $\{r_0, r_2, t_0\}$ or $\{r_1, r_2, t_0\}$. Both choices are consistent: We choose to define $V_{2,-2,0}$ as $\{r_1, r_2, t_0\}$.

[2]There are two equivalent ways of defining "previous". Let $r \lhd s$ represent 'r is previous to s'. Then the two formulations are $r \lhd s \lhd t \lhd r$ and $r \lhd t \lhd s \lhd r$. We arbitrarily choose to use the first formulation.

i,j,k			$W_{i,j,k}$	$V_{i,j,k}$
2	0	0	r_2, s_0, t_0	r_0, r_1, r_2, s_0, t_0
0	2	0	r_0, s_2, t_0	r_0, s_0, s_1, s_2, t_0
0	0	2	r_0, s_0, t_2	r_0, s_0, t_0, t_1, t_2
0	1	1	r_0, s_1, t_1	r_0, s_0, s_1, t_0, t_1
1	0	1	r_1, s_0, t_1	r_0, r_1, s_0, t_0, t_1
1	1	0	r_1, s_1, t_0	r_0, r_1, s_0, s_1, t_0

Table 1: The sets $W_{i,j,k}$ and $V_{i,j,k}$ for the quadratic simplex splines.

i,j,k			$W_{i,j,k}$	$V_{i,j,k}$
0	2	−1	r_0, s_2, s_0	r_0, s_0, s_1, s_2
−1	2	0	s_0, s_2, t_0	s_0, s_1, s_2, t_0
1	1	−1	r_1, s_1, s_0	r_0, r_1, s_0, s_1
0	1	0	r_0, s_1, t_0	r_0, s_0, s_1, t_0
−1	1	1	t_0, s_1, t_1	s_0, s_1, t_0, t_1
2	0	−1	r_2, s_0, r_0	r_0, r_1, r_2, s_0
1	0	0	r_1, s_0, t_0	r_0, r_1, s_0, t_0
0	0	1	r_0, s_0, t_1	r_0, s_0, t_0, t_1
−1	0	2	t_0, s_0, t_2	s_0, t_0, t_1, t_2
2	−1	0	r_2, r_0, t_0	r_0, r_1, r_2, t_0
1	−1	1	r_1, r_0, t_1	r_0, r_1, t_0, t_1
0	−1	2	r_0, t_0, t_2	r_0, t_0, t_1, t_2

Table 2: The sets $W_{i,j,k}$ and $V_{i,j,k}$ for the linear simplex splines.

The knot sets for all other piecewise constant $M_{i,j,k}$'s can be defined analogously. For example, $V_{-2,2,0} = \{s_1, s_2, t_0\}$. Tables 1,2 and 3 give the contents of the various $W_{i,j,k}$ and $V_{i,j,k}$ sets.

We have succeeded in defining $M_{i,j,k}$ for $i+j+k$ from 0 to 2. In order to evaluate the basis functions $N_{i,j,k}$ at a point u, we first compute the piecewise constant simplex splines at that point (Equation 1). Those results are used to compute the linear simplex splines, which in turn are combined to evaluate the quadratic simplex splines. This tabulation of partial results and later reuse is a form of the familiar *dynamic programming* paradigm [AHU74, pages 67-69].

3.3 Discussion

It is worth discussing some of the advantages and disadvantages of this method in comparison with other schemes:

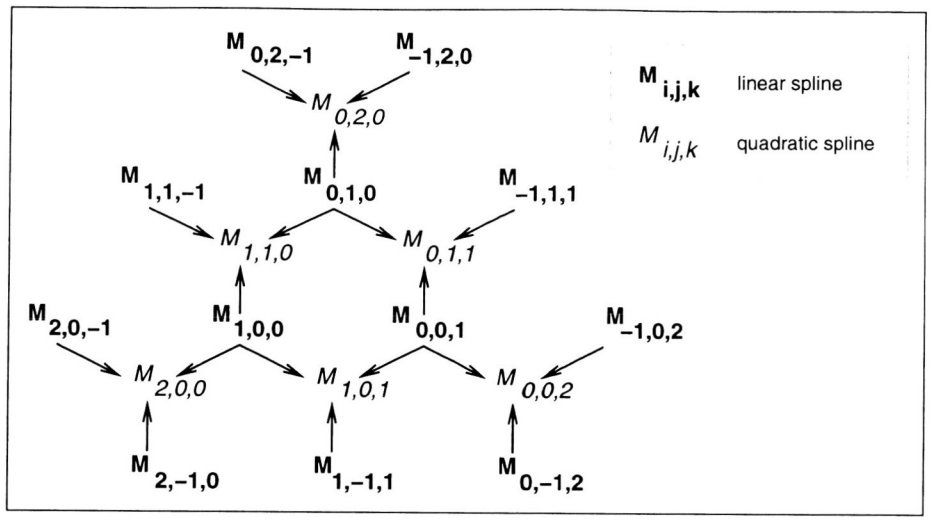

Figure 2: Constructing quadratic simplex splines from linear ones, reusing partial results whenever possible.

i,j,k			$V_{i,j,k}$	i,j,k			$V_{i,j,k}$
0	2	−2	r_0, s_1, s_2	−1	0	1	s_0, t_0, t_1
−1	2	−1	s_0, s_1, s_2	−2	0	2	s_0, t_1, t_2
−2	2	0	s_1, s_2, t_0	2	−1	−1	r_0, r_1, r_2
1	1	−2	r_0, r_1, s_1	1	−1	0	r_0, r_1, t_0
0	1	−1	r_0, s_0, s_1	0	−1	1	r_0, t_0, t_1
−1	1	0	s_0, s_1, t_0	−1	−1	2	t_0, t_1, t_2
−2	1	1	s_0, s_1, t_1	2	−2	0	r_1, r_2, t_0
2	0	−2	r_1, r_2, s_0	1	−2	1	r_1, t_0, t_1
1	0	−1	r_0, r_1, s_0	0	−2	2	r_0, t_1, t_2
0	0	0	r_0, s_0, t_0				

Table 3: The sets $V_{i,j,k}$ for the piecewise constant simplex splines

Determinism

Grandine and Gmelig Meyling spend considerable effort selecting knots at each level of the recurrence. Their choice of knots depends on the location of the parameter point u within the parameter domain. Our scheme, which fixes the choice of knots, evaluates the recurrence without the penalty of run-time knot selection. This has the disadvantage that the barycentric coefficients computed can be negative, which could lead to a loss of precision.

Grandine's difficulty with the storage and retrieval of sub-calculations is ameliorated by our method, because our partial results can always be identified by their multi-index.

This scheme also provides a way for partial results to be shared between adjacent triangles. Many lower order simplex splines evaluated by this technique use knots from only two of the three knot clouds available for consideration. These knot clouds are shared by neighbouring domain triangles.

If the vertices of neighbouring triangles are oriented in opposite directions, then the "previous" relationship described above will be preserved for their common vertices, and some partial results used in evaluating simplex splines in one triangle can be reused in neighbouring ones. Note, however, that it is generally not possible to consistently orient a triangulation globally such that neighbouring triangles have opposite orientations.

Knot Placement

Our method is designed to compute the normalized B-splines where knots are in general position. Note that the new method breaks down in cases where several knots become collinear, because some of the sets $W_{i,j,k}$ or the knot sets $V = \{t_0, t_1, t_2\}$ used in evaluating the piecewise constant simplex splines may not be affinely independent. Thus, a more complicated evaluation scheme must be employed if knot multiplicities are used.

Generalizing to Higher Degree Splines

The success of this indexing scheme seems to rely on the fact that bivariate quadratics are being calculated. The choice of having an index of -1 represent

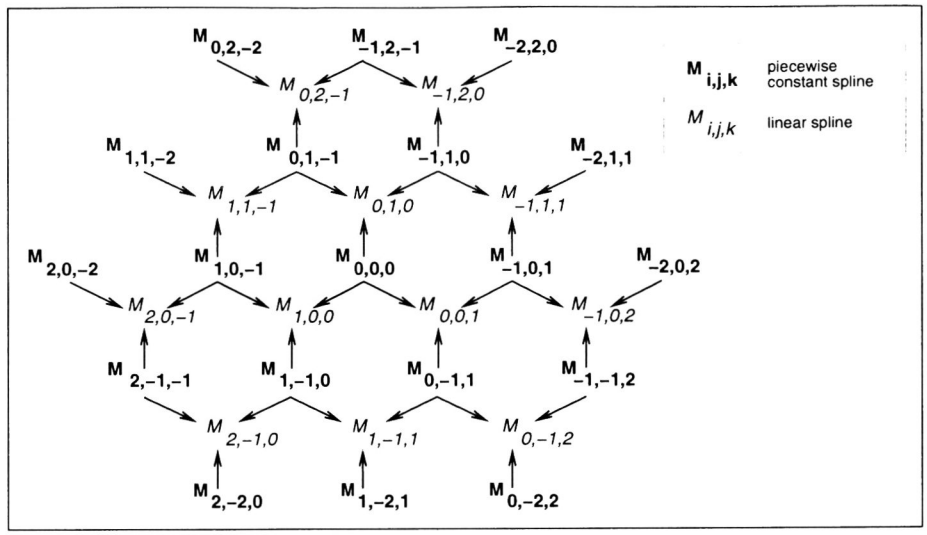

Figure 3: Constructing linear simplex splines from piecewise constant ones, reusing partial results whenever possible.

the absence of any knots from a particular knot cloud leads to a consistent indexing scheme for quadratic surfaces. It is unclear, however, that this choice forms an appropriate starting point for surfaces of greater degree. Without such a starting point, one must develop a much larger set of arbitrary rules in order to define, for example, the sets $V_{3,-2,-1}$ and $V_{1,-3,2}$, which arise in the case of a cubic surface.

Other Enhancements

Gmelig Meyling makes use of the existance of a region within each domain triangle where a faster evaluation algorithm can be employed to evaluate the basis functions of the [DM82, Höl82] scheme. The DMS Spline scheme exhibits a similar region, where a variant of the "de Boor" algorithm can be used to evaluate the surface.

4 Computation Costs

We wish to describe the number of operations involved in computing a linear combination $\sum_{i+j+k=2} c_{i,j,k} N_{i,j,k}(u)$ using the technique outlined above, over a single domain triangle. We defer to Section 5 discussion of the CPU usage of an actual implementation.

Here is an accounting of the operations required by this method:

- Computing the area of each piecewise constant knot set requires the evaluation of a determinant. This must be done for each of the 19 piecewise constant splines, but need only be done when the knot locations are established, and not each time a point is evaluated.

- The characteristic function $\chi_{[\,]}$ must be calculated for each of the 19 piecewise constant splines, which is essentially the computation of a set of barycentric coordinates[3].

- For each of the 12 linear and 6 quadratic splines, the algorithm calculates a set of barycentric coordinates for u.

- For each of the 6 normalized B-splines, a normalization factor is calculated (one determinant). These normalizing factors are fixed for a given set of knots, and can be pre-calculated.

Therefore, once pre-calculation is done, $(6 + 12 + 19 =)$ 37 sets of barycentric coordinates are calculated for each evaluated point u. If a normal vector at $F(u)$ is also desired, this can be found with an additional two barycentric coordinates calculations, since the derivative evaluation formula for simplex

[3] For a point u lying directly on the boundary of the triangle Δ, more work needs to be done to determine if u is within the half-open convex hull.

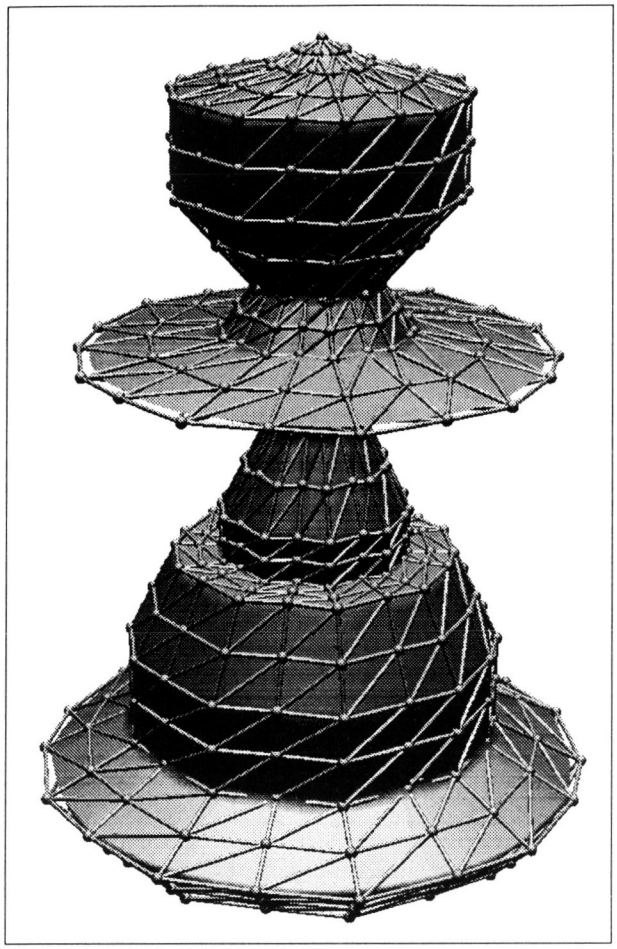

Figure 4: A chess piece composed of over 250 patches with control net.

splines makes use of the same partial results as the recursion scheme [FS93].

In contrast to this are the results for the naive approach. The naive approach calculates a set of barycentric coordinates for each of the 6 quadratic splines, a set of barycentric coordinates for each of the 18 linear sub-splines defined by recursion, and the $\chi_{[]}$ function for each of the 54 subsequent piecewise constant sub-splines, for a total of 78.

5 Numerical Results

Now we compare the actual CPU usage of an implementation in "C" of this algorithm, as compared with the evaluation algorithm implemented in [Fon92, FS93], over a single domain triangle.

The computer system used for measurement was the IRIX 4.0.5H operating system, running on a *Silicon Graphics Indigo XS24-4000.*

Algorithm	30056 points	one point
Fong	10.41 s	346 μs
New	4.47 s	149 μs

Table 4: CPU usage for both algorithms.

104 points from an equally spaced triangular grid (lying completely within the domain triangle) were evaluated by both algorithms, repeated so that a total of 30056 point evaluations were performed, and CPU usage was measured. This was repeated 10 times, and the average taken. The results given in Table 4 show that the new method is roughly 2.3 times faster than the previous implementation.

6 Conclusions and Further Work

We have presented an algorithm for efficiently evaluating quadratic bivariate B-splines, whose knots are in general position, by careful reuse of partial results during calculation. It compares favourably with previous techiques in terms of number of operations performed per evaluation. Moreover, when compared in implementation against another technique, this method ran roughly 2.3 times faster.

There are a number of areas discussed where further investigation is warranted. One is the investigation into whether there exists some extension of this technique that will work in the evaluation of higher degree splines. Another is the search for a means of altering this technique so that it will work with the more general knot configurations permitted under the DMS Spline scheme. A final direction would be to establish exactly which partial results for B-splines defined over one triangular domain could be reused by overlapping B-splines defined over neighbouring triangles.

Acknowledgements

We would like to thank our anonymous referees for their helpful comments and criticisms.

References

[AHU74] A.V. Aho, J.E. Hopcroft, and J.D. Ullman.

The Design and Analysis of Computer Algorithms. Addison-Wesley, 1974.

[BBB87] R.H. Bartels, J.C. Beatty, and B.A. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann Publishers, 1987.

[DM82] W. Dahmen and C. A. Micchelli. On the linear independence of multivariate B-splines, I. triangulations of simploids. *SIAM J. Numer. Anal.*, 19(5):993–1012, October 1982.

[DMS92] W. Dahmen, C. A. Micchelli, and H.-P. Seidel. Blossoming begets B-spline bases built better by B-patches. *Mathematics of Computation*, 1(1):97–115, July 1992.

[Far93] G.E. Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, 3. edition, 1993.

[Fon92] P. Fong. Shape control for multivariate B-spline surfaces over arbitrary triangulations. Master's thesis, University of Waterloo, 1992.

[FS93] P. Fong and H.-P. Seidel. An implementation of triangular B-spline surfaces over arbitrary triangulations. *Computer Aided Geometric Design*, 10:267–275, 1993.

[GM86] R. H. J. Gmelig Meyling. *Polynomial Spline Approximation in Two Variables.* PhD thesis, University of Amsterdam, 1986.

[Gra87] T. A. Grandine. The computational cost of simplex spline functions. *SIAM J. Numer. Anal.*, 24(4):887–890, August 1987.

[Höl82] K. Höllig. Multivariate splines. *SIAM J. Numer. Anal.*, 19(5):1013–1031, October 1982.

[Mic79] C. A. Micchelli. On a numerically efficient method for computing multivariate B-splines. In W. Schempp and K. Zeller, editors, *Multivariate Approximation Theory.* Birkhäuser, Basel, 1979.

[Sei91] H.-P. Seidel. Polar forms and triangular B-Spline surfaces. In *Blossoming: The New Polar-Form Approach to Spline Curves and Surfaces SIGGRAPH '91 Course Notes # 26.* ACM SIGGRAPH, 1991.