

# Efficient Evaluation of Triangular B-splines

*Michael Franssen,  
Remco C. Veltkamp and  
Wieger Wesselink*

UU-CS-1999-18



# Efficient Evaluation of Triangular B-splines

Michael Franssen<sup>‡</sup> Remco C. Veltkamp\* Wieger Wesselink<sup>‡</sup>

<sup>‡</sup> Technical University of Eindhoven  
Dept. Computing Science  
Den Dolech 2, 5612 AZ Eindhoven, The Netherlands  
e-mail: mfranssen@usa.net, wieger@win.tue.nl

Utrecht University, Dept. Computer Science  
Padualaan 14, 3584 CH Utrecht, The Netherlands  
email: Remco.Veltkamp@cs.uu.nl

## Abstract

Evaluation routines are essential for any application that uses triangular B-splines. This paper describes an algorithm to efficiently evaluate triangular B-splines. The novelty of the algorithm is its generality: there is no restriction on the degree of the B-spline or on the dimension of the domain. Constructing an evaluation graph allows us to reuse partial results and hence, to decrease computation time. Computation time gets reduced even more by making choices in unfolding the recurrence relation of simplex splines such that the evaluation graph becomes smaller. The complexity of the algorithm is measured by the number of leaves of the graph.

## 1 Introduction

Evaluation routines for splines are important for any application that uses splines. These applications vary from scattered data approximation to variational surface modeling or 3D-morphing applications. In the end, the resulting spline is always sampled to compute the results or for visualization.

Efficient evaluation schemes have been developed and implemented for many classes of splines, e.g. for Bezier-surfaces [Béz72] and B-patches [Sei91]. However, for triangular B-splines, which are widely used for their many desirable properties, efficient evaluation routines are restricted to the quadratic bivariate case [FS93, PS94].

In this paper we present an algorithm for efficient evaluation of triangular B-splines as introduced by Dahmen, Michelli and Seidel [DMS92]. The novelty of this algorithm is that it works for triangular B-splines of arbitrary degree and with an arbitrary number of dimensions of the domain. For simplicity, however, the main part of this paper concentrates on the bivariate case. The generalization towards arbitrary domains is discussed in section 7.

Efficiency is obtained by re-using partial results. When Grandine [Gra87] attempted this approach, he found that tabulating those partial results for reuse is more costly than simply re-computing the required value. He attributes this to the need of the entire knot-set to identify a simplex spline. Pfeifle and Seidel [PS94] use a triple of integers to identify the simplex splines encountered during evaluation of a triangular B-spline of degree 2. Unfortunately, their numbering does not scale up to higher order triangular B-splines. In this paper it is shown that the identification problem can

be avoided by constructing a directed graph (an evaluation graph) representing the simplex- and B-splines. In section 3 we describe how this graph is built and how it is used to avoid multiple evaluation of simplex splines.

To further reduce computation cost, we cut down on the number of partial results that are required. This is done by using our degrees of freedom when unfolding the recurrence relation for simplex splines.

The selection scheme for simplex splines is described in section 4 and the selection scheme for triangular B-splines is described in section 5. The complexity of the algorithm is computed in section 6 and the results are discussed in section 8.

## 2 Definitions

The notations used for simplex- and triangular B-splines differ in some papers. Therefore, we briefly review the definition of the splines we consider in this paper. This section assumes that the reader is already familiar with triangular B-splines.

### 2.1 General definitions

**Definition 2.1** [Determinant of points] Let  $V = (v_0, v_1, v_2)$  be a triple of points in  $\mathbb{R}^2$ . Then the determinant of  $V$ , denoted as  $\det(V)$  is defined as

$$\det(V) = \det \begin{pmatrix} 1 & 1 & 1 \\ v_{0x} & v_{1x} & v_{2x} \\ v_{0y} & v_{1y} & v_{2y} \end{pmatrix}. \quad (1)$$

**Definition 2.2** [Barycentric determinant] Let  $V$  be a triple of points in  $\mathbb{R}^2$  and let  $x$  be a point in  $\mathbb{R}^2$ . Then the  $i$ -th barycentric determinant of  $x$  ( $0 \leq i \leq 2$ ) is defined as

$$d_i(x | V) = \det(V[x/v_i]), \quad (2)$$

where  $V[x/v_i]$  denotes the set  $V$  in which  $v_i$  is replaced by  $x$ .

**Definition 2.3** [Barycentric coordinates] Let  $V$  be a triple of points in  $\mathbb{R}^2$  and let  $x$  be a point in  $\mathbb{R}^2$ . Then the  $i$ -th barycentric coordinate of  $x$  ( $0 \leq i \leq 2$ ) is defined as

$$\lambda_i(x | V) = \frac{d_i(x | V)}{\det(V)}. \quad (3)$$

Barycentric coordinates have the following important properties:

- $\sum_{i=0}^2 \lambda_i(x | V) = 1$  and  $\sum_{i=0}^2 \lambda_i(x | V)v_i = x$ .
- If  $x$  lies within the convex hull of  $V$ , then  $0 \leq \lambda_i(x | V)$  for  $0 \leq i \leq 2$ .

**Definition 2.4** [Half-open convex hull] Let  $V$  be a set of points in  $\mathbb{R}^2$  and let  $e_i$  denote the unit vector for dimension  $i$  for  $i = 0, 1$ . Then the half-open convex hull of  $V$  is defined as

$$[V] = \{x \in [V] \mid \exists_{\varepsilon_0, \varepsilon_1 > 0} (\forall_{0 \leq \alpha_1 \leq \alpha_0 \leq 1} (x + \alpha_0 \varepsilon_0 e_0 + \alpha_1 \varepsilon_1 e_1 \in [V]))\}, \quad (4)$$

where  $[V]$  denotes the convex hull of  $V$ .

The half-open convex hull is a generalization of the half-open domain in  $\mathbb{R}$ . Its purpose is to ensure that for any subdivision of a domain in  $\mathbb{R}^2$ , the points on the edges of the subdivision belong to exactly one sub-area.

## 2.2 Definition of simplex splines

**Definition 2.5** [Simplex splines] A simplex spline is a piecewise polynomial function defined by a finite set  $V$  of points in  $\mathbb{R}^2$ . The points in  $V$  are called *knots* and the set  $V$  itself is called the *knot-set* of the simplex spline. A simplex spline defined over a set of  $n + 3$  knots is said to have degree  $n$ . The definition of a simplex spline is given by the following recursive equation:

$$M(x | V) = \begin{cases} 0 & x \notin [V] \\ \frac{1}{|\det(V)|} & |V| = 3 \text{ and } x \in [V] \\ \sum_{i=0}^2 \lambda_i(x | W)M(x | V \setminus \{w_i\}) & |V| > 3 \end{cases} \quad (5)$$

The elements in  $W = (w_0, w_1, w_2)$  can be chosen arbitrarily from  $V$ , hence  $W \subset V$ .  $W$  is called the *split set* for  $V$ . The only restriction is that  $\det(W)$  may not be zero.

If all knots are in general position, i.e. the knot-set does not contain a collinear triple of knots, a simplex spline of degree  $n$  defined over these knots is  $C^{n-1}$  continuous. For more information about simplex splines, we refer to Traas [Tra90].

## 2.3 Definition of triangular B-splines

**Definition 2.6** [Triangular B-splines] A triangular B-splines is a piecewise polynomial function defined over an arbitrary polygonal domain in  $\mathbb{R}^2$ . For clarity, we present the construction of a triangular B-spline in a number of steps:

1. One starts by constructing a triangulation  $\mathcal{I}$  of the polygonal domain. This triangulation has to be proper, i.e. triangles may not overlap and they can only share a single edge or a single vertex.
2. Assign to every vertex  $v_i$  occurring in the triangulation  $n + 1$  knots, denoted by  $v_{i,0}, \dots, v_{i,n}$ , where  $n$  is the degree of the triangular B-spline, such that  $v_i = v_{i,0}$ . There are two important restrictions on the placement of these knots:
  - (a) For every edge  $(v_i, v_j)$  at the boundary of the polygonal domain, the entire area  $\{v_{i,0}, \dots, v_{i,n}, v_{j,0}, \dots, v_{j,n}\}$  must lie outside the polygonal domain.
  - (b) For every triangle  $I = (i_0, i_1, i_2)$  in  $\mathcal{I}$ , the determinants  $\det(i_{0,k}, i_{1,l}, i_{2,m})$  with  $k + l + m \leq n$  must have the same sign. Often these requirements are not mentioned, even though they are essential to guarantee the desired B-spline properties. More information on these restrictions can be found in [Fra95].
3. Let  $I = (i_0, i_1, i_2)$  be a triangle in  $\mathcal{I}$ . Let  $\beta$  be a triple of indices  $(\beta_0, \beta_1, \beta_2)$  such that  $|\beta| = \beta_0 + \beta_1 + \beta_2 = n$  and  $\beta_j \geq 0$ . Then the set  $V_\beta^I$ , containing  $n + 2$  knots, is defined as

$$V_\beta^I = \{v_{(i_0,0)}, \dots, v_{(i_0,\beta_0)}, v_{(i_1,0)}, \dots, v_{(i_1,\beta_1)}, v_{(i_2,0)}, \dots, v_{(i_2,\beta_2)}\}. \quad (6)$$

Each of these  $V_\beta^I$  will serve as the knot-set of a simplex spline needed to define a triangular B-spline.

4. To use the simplex splines defined over the  $V_\beta^I$ -s as a basis for a triangular B-spline, we have to 'normalize' them. That is, we have to multiply every simplex spline  $M(x | V_\beta^I)$  with the

factor  $d_\beta^I = \det(v_{i_0, \beta_0}, v_{i_1, \beta_1}, v_{i_2, \beta_2})$ . As a result, for every point  $x$  in the polygonal domain, we get

$$\left( \sum_{I \in \mathcal{I}} \sum_{|\beta|=n} d_\beta^I M(x | V_\beta^I) \right) = 1. \quad (7)$$

Hence, the normalized simplex splines form a partition of unity, making control points easy to use.

5. For every triangle  $I$  and every triple of indices  $\beta$ , we define a control point  $c_\beta^I$  in  $\mathbb{R}^3$ . The triangular B-spline surface is then defined as

$$F(x) = \sum_{I \in \mathcal{I}} \sum_{|\beta|=n} d_\beta^I M(x | V_\beta^I) c_\beta^I. \quad (8)$$

Since the normalization factors and the control points do not depend on the evaluation point  $x$ , they need not to be considered in the evaluation algorithm. The normalization factors are pre-computed once and the control coefficients are typically set (indirectly) by the user of the application.

### 3 Reusing partial results

If we naively evaluate a simplex spline of degree  $n$  recursively, the amount of constant simplex splines we encounter will be  $3^n$ . For every unfolding of equation 5 for a simplex spline of degree  $i$  we have to evaluate 3 simplex splines of degree  $i - 1$ . Since a constant simplex spline is obtained after  $n$  unfoldings, we obtain  $3^n$  simplex splines of degree zero.

Not all of the simplex splines of degree  $i$  with  $0 \leq i < n$  that we evaluate during the recursion are different. The knot-set  $V$  of a simplex spline of degree  $n$  contains  $n + 3$  knots. Any simplex spline of degree  $i$  we encounter by recursively unfolding equation 5, has a knot-set  $V'$  of  $i + 3$  knots that is a subset of  $V$ . Therefore there exist no more than  $\binom{n+3}{i+3}$  different simplex splines of degree  $i$ .

Evaluation will be accelerated if every simplex spline of degree  $i$  is computed only once. The problem when re-using partial results is the identification of simplex splines, because this requires comparing the entire knot-sets. In this section we will present a data structure that makes identifying simplex splines during evaluation superfluous.

We construct a directed graph, in which every node represents a simplex spline. This graph is built only once (during preprocessing) and then used for all future evaluations. Every simplex spline with degree  $i$  greater than 0 has three outgoing edges that connect it with three (different) simplex splines of degree  $i - 1$ . These three simplex splines are determined by choosing a split set  $W$  and unfolding the recurrence relation 5 for simplex splines. Note that we do not need to choose a split set for every point  $x$  in which we evaluate the simplex spline: once the graph is built, it can be used to evaluate the simplex spline in arbitrary points.

Ensuring that every node in the graph represents a unique simplex spline can be done by a simple look-up table. Simplex splines are represented by a sorted list of the indices of their knots. In our look-up table we store for every degree  $i$  with  $0 \leq i \leq n$  a sorted list of simplex splines that already exist. (Sorting can be done on alpha-lexicographical ordering of the knot-indices.) Whenever we need (a reference to) a simplex spline, we first check if the simplex spline already exists in our table. If not, we create a representation for the apparently new simplex spline and insert it in the table. If it already exists, we use the stored simplex spline.

Evaluating the simplex spline at a point  $x$  can now be done efficiently as follows:

1. We assign a number to each point  $x$  we want to evaluate.

2. Then we traverse the graph (which corresponds to unfolding the recurrence relation), starting at the node representing the simplex spline of degree  $n$ . We label every node  $V$  we visit with a pair  $(xv, rv)$ , where  $xv$  is the number corresponding to  $x$ , and  $rv$  is the value  $M(x | V)$  we obtain by evaluation.
3. If we visit a node whose value  $xv$  is equal to the number assigned to  $x$ , we know that we have visited this node for  $x$  before and hence, the label's  $rv$  is the value we need. Since every node represents a unique simplex spline, we know that every simplex spline is evaluated at most once. If we use a single evaluation graph for several simplex splines defined over one large set of knots, not all nodes will be visited during the evaluation of a single point.

## 4 Choosing split sets for simplex splines

We can use the number of nodes in the graph as a measure for the efficiency, since the data-structure from the previous section avoids multiple evaluation of simplex splines in this graph. To increase the efficiency, we have to decrease the number of nodes in our graph. Which simplex splines occur in the graph depends on our choice of split sets when unfolding equation 5. In this section, we present a selection scheme for split sets that strongly decreases the number of nodes in the graph of a single simplex spline.

Throughout this section we will use  $V = \{v_0, \dots, v_{n+2}\}$  to denote the knot-set of the simplex spline of degree  $n$  that we want to evaluate. Furthermore,  $i$  will always denote a degree between 0 and  $n - 1$  of some simplex spline in the graph. For simplicity, we assume that every choice for the split set is legal, i.e.  $V$  does not contain a triple of linearly dependent knots. In section 7, we discuss how this restriction is eliminated.

To minimize the number of simplex splines in the graph, we want to use as few different simplex splines of degree  $i$  as possible. Therefore, we want to keep the intersections of different knot-sets as large as possible. By choosing the correct split sets we will then create less different simplex splines, since more simplex splines of lower degree become shared.

To establish this similarity between splines, we split, for every degree  $i$ , the knot-set  $V$  of the original simplex spline of degree  $n$  in two disjoint sets  $Z_i$  and  $Z'_i$ . The idea is that  $Z'_i$  denotes the knots that occur in every simplex spline of degree  $i$  in our graph. Besides the element of  $Z'_i$ , the simplex splines of degree  $i$  will also have some knots from  $Z_i$ . We then keep  $Z'_i$  as large as possible. When we choose a knot  $w$  for split set  $W$ , we prefer to choose from  $Z_i$ . If we choose  $w \in Z'_i$ , we know that  $w \notin Z'_{i-1}$ , since  $w$  then causes us to create a simplex spline of degree  $i - 1$  that does not contain  $w$  and hence,  $w$  does not occur in every simplex spline of degree  $i - 1$ .

We now derive our selection scheme using the following heuristic: initially we only have the simplex spline of degree  $n$ , hence  $Z_i = \emptyset$  and  $Z'_i = V$ . The first choice for  $W$  is arbitrary and results in  $Z_{n-1} = W$  and  $Z'_{n-1} = V \setminus W$ . Note that every simplex spline of degree  $n - 1$  has exactly 2 elements of  $Z_{n-1}$ . Since a split set must contain 3 elements, we must choose one element for the split set from  $Z'_{n-1}$  for each of these simplex splines. In order to obtain the largest  $Z'_{n-2}$ , we choose the same element from  $Z'_{n-1}$  for every simplex spline of degree  $n - 1$  in the graph. This selection scheme generalizes: we get a graph in which a simplex spline of degree  $i < n$  contains two knots from  $Z_i$ . These are always chosen for the split set together with one element from  $Z'_i$  that is the same for all simplex splines of degree  $i$ .

As an example: for every simplex spline choose the first three knots from its knot-set. That is, initially we choose  $W = \{v_0, v_1, v_2\}$ . The element chosen from  $Z'_i$  at any level  $i$  is  $v_{n-i+2}$ . We then have for  $0 \leq i < n$  that  $Z_i = \{v_0, \dots, v_{n-i+1}\}$  and  $Z'_i = \{v_{n-i+2}, \dots, v_{n+2}\}$ .

Every simplex spline of degree  $i$  contains  $i + 3$  knots. Since the  $i + 1$  knots of  $Z'_i$  occur in every simplex spline of degree  $i$  in the graph, its knot-set must contain exactly  $(i + 3) - (i + 1) = 2$

knots from  $Z_i$ . Therefore, the number of simplex splines of degree  $i$  in the graph is limited by the number of pairs of knots that can be chosen from the  $n - i + 2$  knots of  $Z_i$ , which is  $\binom{n-i+2}{2}$ .

## 5 Choosing split sets for triangular B-Splines

The computation of a triangular B-spline requires the evaluation of an entire set of simplex splines instead of just one. Since many knots occur in several of these simplex splines, there is hope that evaluating this set of simplex splines can be done more efficiently than simply evaluating all the simplex splines separately. In this section we will exploit similarities between the knotsets of the simplex splines to get an efficient evaluation scheme.

Since the triangulation of the domain of the B-spline, is arbitrary, we will restrict our attention to the computation of the contribution of a single triangle. For the remainder of this section we will denote this triangle as  $I = (i_0, i_1, i_2)$ . The same evaluation scheme can then be used for all triangles in the domain. By using a single look-up table of simplex splines for the entire domain of the B-spline, the evaluation time is reduced even further, but we will not discuss this effect in this paper.

We start by introducing the concept of fingerprints. Fingerprints are special subsets of the knotsets of the simplex splines in the graph of the triangular B-spline. They will be used to distinguish two groups of simplex splines in the graph of this B-spline: those that contain a fingerprint and those that do not.

**Definition 5.1** [Fingerprints] For every triangle  $I = (i_0, i_1, i_2) \in \mathcal{I}$  and index  $\beta$  with  $|\beta| = n$  we define a fingerprint  $F_\beta^I$  as

$$F_\beta^I = \{v_{i_j, \beta_j} \mid 0 \leq j \leq 2 \wedge 1 \leq \beta_j\}. \quad (9)$$

Hence, the fingerprint  $F_\beta^I$  contains from every knot cloud the knot with the highest second index occurring in  $V_\beta^I$ , provided that this index is at least 1. For example:  $F_{(2,0,1)}^I = \{v_{i_0,2}, v_{i_2,1}\}$  (and not  $\{v_{i_0,1}, v_{i_0,2}, v_{i_2,1}\}$ ). In the graph of a triangular B-spline of degree  $n$ , only the  $F_\beta^I$  with  $|\beta| = n$  are called fingerprints.

The name 'fingerprint' is not chosen arbitrarily. A fingerprint  $F_\beta^I$  is the smallest set uniquely identifying the degree  $n$  simplex spline  $M(\cdot \mid V_\beta^I)$ . Hence, if a fingerprint  $F$  is a subset of the knot-set  $V$  of some simplex spline, then  $F$  uniquely defines a degree  $n$  simplex spline, which is the only one whose evaluation requires the evaluation of  $V$ .

**Lemma 5.2** Let  $x \in V_\beta^I$  such that  $x \in F_\beta^I$ . Then there exists a  $\gamma$  with  $|\gamma| = |\beta| - 1$  such that  $V_\gamma^I = V_\beta^I \setminus \{x\}$ .

PROOF: We have  $x \in F_\beta^I$ , hence  $x = v_{i_j, \beta_j}$  for certain  $j$ , where  $1 \leq \beta_j$ . Let  $\gamma$  be  $\beta - e^j$  then  $V_\gamma^I = V_{\beta - e^j}^I = V_\beta^I \setminus \{v_{i_j, \beta_j}\} = V_\beta^I \setminus \{x\}$ .  $\square$

**Lemma 5.3** Let  $V$  be the knot-set of a simplex spline in the graph of a triangular B-spline of degree  $n - 1$ . Let  $\beta$  be an index with  $|\beta| = n$ . Then  $F_\beta^I \not\subseteq V$ .

PROOF: Since  $V$  is in the graph of a triangular B-spline of degree  $n - 1$ , all paths leading to  $V$  start with a simplex spline of degree  $n - 1$ . All of these simplex splines have a knot-set  $V_\gamma^I$  for certain  $\gamma$  with  $|\gamma| = n - 1$ . Clearly, since  $|\beta| = n$  we have  $F_\beta^I \not\subseteq V_\gamma^I$  and since  $V \subseteq V_\gamma^I$  we have  $F_\beta^I \not\subseteq V$ .  $\square$



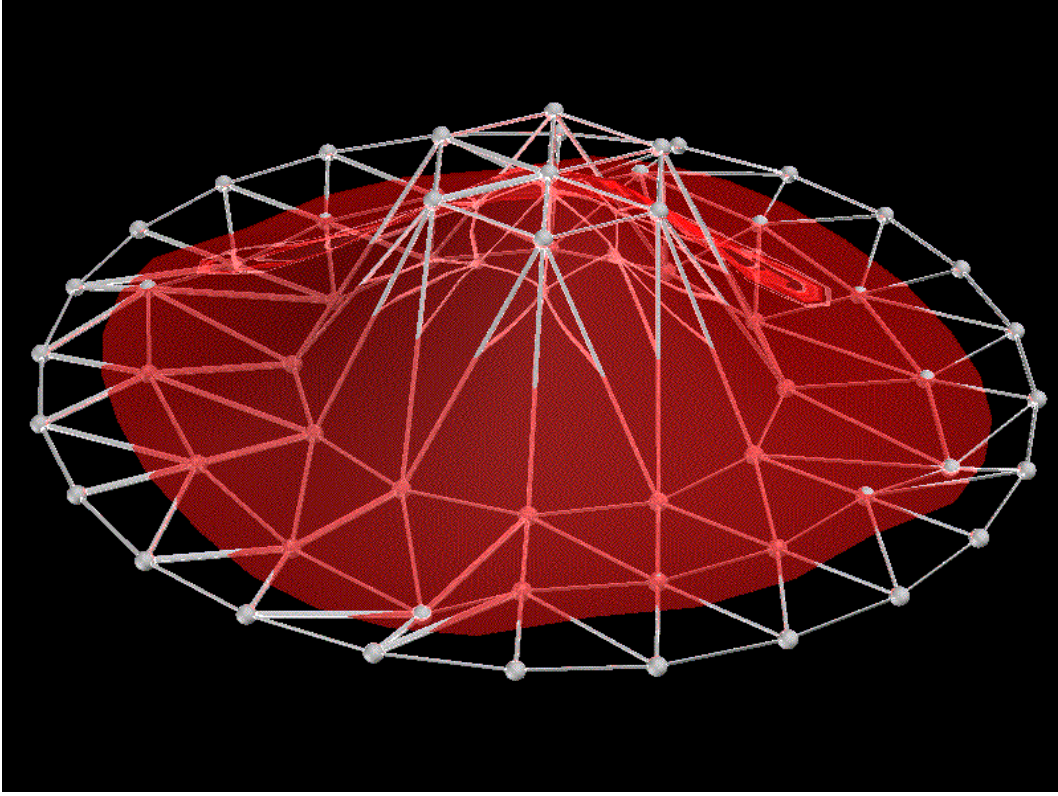


Figure 1: A B-spline of degree 4 with 6 patches.

Using the concept of fingerprints and the corresponding lemmas, we will define our selection scheme for the evaluation of triangular B-splines. The evaluation scheme will be based on lemma 5.2 and the evaluation scheme for a single simplex spline.

To compute the value of a B-spline of degree  $n$ , we have to compute  $M(x | V_\beta^I)$  for all  $\beta$  with  $|\beta| = n$ . For this computation we will construct a single evaluation graph  $G_n$ . Our idea is to use a graph  $G_{n-1}$  that efficiently computes all  $M(x | V_\gamma^I)$  with  $|\gamma| = n - 1$  and add as few new simplex splines to it as possible to obtain  $G_n$ . Initially, we use the evaluation graph for a B-spline of degree 0, which consists of a single simplex spline of degree 0. Clearly, this graph is optimal.

Using  $G_{n-1}$  as a subgraph of  $G_n$  is accomplished by the following heuristic: If a simplex spline contains a fingerprint then this fingerprint must be a subset of the split set. From this decision and lemma 5.2 it follows that evaluating all required B-splines of degree  $n$  (i.e. all B-splines with a knot-set  $V_\beta^I$  with  $|\beta| = n$ ) requires the evaluation of all B-splines with a knot-set  $V_\gamma^I$  with  $|\gamma| = n - 1$ . This is done efficiently by a graph  $G_{n-1}$  of a B-spline of degree  $n - 1$  defined over the same knot-set, which we already have. However, not every fingerprint contains 3 knots, so we still have to select a few elements for the split sets.

The remaining knots of the split sets are chosen in the same way as split sets for single simplex splines are chosen: we use the first  $3 - |F_\beta^I|$  knots from  $V \setminus F_\beta^I$ . To define 'first' we may use an arbitrary ordering on the knots, e.g. alpha-lexicographical ordering on the two indexes. Figure 1 shows a triangular B-spline of degree 4 with 6 patches.

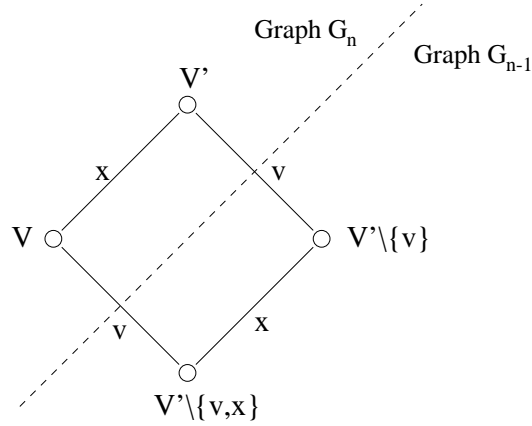


Figure 2: Sketch of the proof of lemma 6.1

## 6 Complexity

The simple selection scheme of split sets presented above yields a surprisingly efficient evaluation graph. In order to compute the number of nodes in the graph, we prove the following lemma and theorem:

**Lemma 6.1** *Let  $V$  be a simplex spline in the graph  $G_n$  of a B-spline of degree  $n$ , such that  $F_\beta^I \subseteq V$ . Let  $v$  be an element in the split set of  $V$ . Then*

- *if  $v \notin F_\beta^I$  then  $F_\beta^I \subseteq V \setminus \{v\}$ .*
- *if  $v \in F_\beta^I$  then  $V \setminus \{v\}$  exists in the graph  $G_{n-1}$  of the B-spline of degree  $n - 1$  defined over the same knot-set.*

PROOF: The first case is trivial. The second case is proved by induction on  $i$ , where  $n - i$  is the degree of the simplex spline  $V$ .

**case  $i = 0$ :** Given by lemma 5.2

**case  $i > 0$ :** We need to prove that  $V \setminus \{v\}$  exists in the  $G_{n-1}$ . Let  $V'$  be an ancestor of  $V$ , say  $V' = V \cup \{x\}$  (see figure 2). Then  $V'$  contains  $F_\beta^I$  and hence, by the induction hypothesis  $V' \setminus \{v\}$  exists in  $G_{n-1}$ .  $V' \setminus \{v\}$  will contain some fingerprint  $F_\gamma^I$  of a B-spline of degree  $j$  with  $j < n$ . Clearly  $|F_\gamma^I| \leq |F_\beta^I|$  and since  $x$  is one of the first  $3 - |F_\beta^I|$  elements of  $V'$ , it will be one of the first  $3 - |F_\gamma^I|$  elements of  $V' \setminus \{v\}$  or it will be an element of  $F_\gamma^I$ . Therefore,  $x$  is an element of the split set of  $V' \setminus \{v\}$ . Now, since  $V' \setminus \{v\}$  exists in  $G_{n-1}$ ,  $V' \setminus \{v, x\} = V \setminus \{v\}$  will also exist in  $G_{n-1}$  as was to be proved.

⊠

**Theorem 6.2** *Using the split set selection scheme above, we will now prove that every simplex spline  $V$  in the graph of a degree  $n$  B-spline*

- *is either a simplex spline in the graph  $G_{n-1}$  of a B-spline of degree  $n - 1$ .*
- *or contains a fingerprint  $F_\beta^I$  for certain  $\beta$  with  $|\beta| = n$ .*

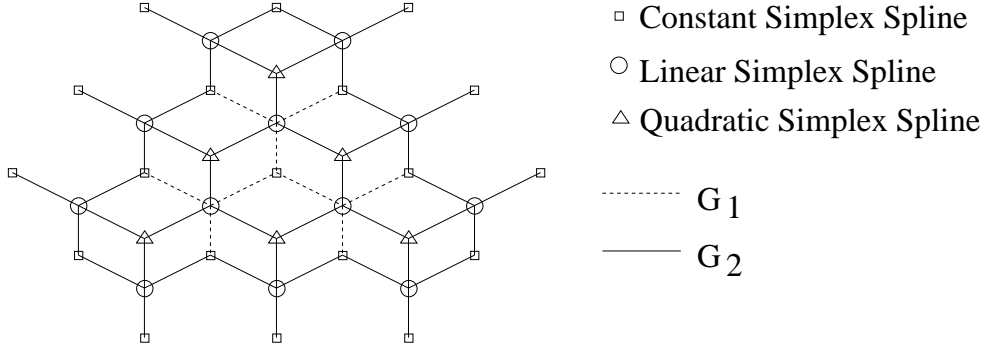


Figure 3: The graph  $G_2$  of a B-spline of degree 2. The dashed lines indicate edges from graph  $G_1$ .

PROOF: By induction on  $i$ , where  $n - i$  is the degree of the simplex spline  $V$ :

**case  $i = 0$ :**  $n - i = n$  and hence  $V$  contains a fingerprint by definition.

**case  $i > 0$ :** Let  $V'$  be an ancestor of  $V$ ; i.e.  $V = V' \setminus \{x\}$ . If  $V'$  is in  $G_{n-1}$  then  $V$  certainly is.

If  $V'$  is not  $G_{n-1}$  we get from the induction hypothesis that  $V'$  contains a fingerprint  $F_\beta^I$  for certain  $\beta$ . If  $x \in F_\beta^I$  then by lemma 6.1  $V' \setminus \{x\} = V$  exists in  $G_{n-1}$ . If  $x \notin F_\beta^I$ , then clearly  $V$  contains  $F_\beta^I$ .

☒

As an example, see figure 3: The graph  $G_1$  is a subgraph of  $G_2$ .

The complexity discussed in this section refers to the complexity of the evaluation and does not include the time required to construct the graph. Constructing the graph is done once during preprocessing using a lookup table. Consulting this lookup table cost time, but once the graph is constructed, the B-spline can be evaluated in an arbitrary number of points without ever using the lookup table again.

The complexity of the evaluation algorithm is expressed as the number  $A_n$  of constant simplex splines in the evaluation graph  $G_n$ . Let  $B_n$  denote the number of constant simplex splines in  $G_n$  that do not exist in  $G_{n-1}$ . Then we find the equation

$$A_n = A_{n-1} + B_n, \quad (10)$$

or written differently

$$A_n = A_0 + \sum_{i=1}^n B_i. \quad (11)$$

In a B-spline of degree 0 the only simplex spline is already a constant simplex spline, hence  $A_0 = 1$ . It remains to compute  $B_n$ .

From theorem 6.2 it follows that every simplex spline in  $G_n \setminus G_{n-1}$  contains a fingerprint. Since a simplex spline with fingerprint  $F_\beta^I$  can only be used to compute  $V_\beta^I$ , the graph  $G_n \setminus G_{n-1}$  contains one connected component for each  $\beta$ . All simplex splines in such a subgraph contain the same fingerprint. We will now count for each index  $\beta$  the number  $B_\beta$  of constant simplex splines in the subgraph of splines containing  $F_\beta^I$ . To compute  $B_n$ , we then use

$$B_n = \sum_{|\beta|=n} B_\beta. \quad (12)$$

$B_\beta$  is computed similar to the number of constant simplex splines in the graph of a single simplex spline: write the elements of  $V_\beta^I$  as the elements of  $F_\beta^I$ , followed by the elements of  $V_\beta^I \setminus F_\beta^I$  in the order used by the selection algorithm. Effectively, the algorithm chooses the first 3 elements from  $V_\beta^I$  and does the same for all descendents in the graph that contain  $F_\beta^I$ . Like in section 4, we use  $Z_i$  and  $Z'_i$  to split  $V_\beta^I$  in two parts. As a result,  $Z_i$  contains the first  $n - i + 2$  elements of  $V_\beta^I$ , written in the order above, and  $Z'_i$  contains the remaining elements. Again, all simplex splines of degree  $i$  will contain 2 elements of  $Z_i$ . Since we are only interested in simplex splines from  $G_n \setminus G_{n-1}$ , these elements include  $F_\beta^I$ , hence we can only choose  $2 - |F_\beta^I|$  elements from the last  $n - i + 2 - |F_\beta^I|$  elements of  $Z_i$ . Hence, in  $G_n \setminus G_{n-1}$  there are  $\binom{n-i+2-|F_\beta^I|}{2-|F_\beta^I|}$  simplex splines of degree  $i$  that contain  $F_\beta^I$ ; i.e.  $B_\beta = \binom{n+2-|F_\beta^I|}{2-|F_\beta^I|}$ . To compute  $B_n$  we now sum over all  $\beta$ , distinguishing between  $|F_\beta^I|$ :

**case  $|F_\beta^I| = 3$ :**  $B_\beta = \binom{n-1}{-1} = 0$

**case  $|F_\beta^I| = 2$ :**  $B_\beta = \binom{n}{0} = 1$ . One out of three indices is 0; the other two vary between 1 and  $n - 1$ , summing up to  $n$ . Hence, there are  $3(n - 1)$  of these cases.

**case  $|F_\beta^I| = 1$ :**  $B_\beta = \binom{n+1}{1} = n + 1$ . There are 3 of these cases:  $(n, 0, 0)$ ;  $(0, n, 0)$ ; and  $(0, 0, n)$ .

Hence,  $B_n = 3(n - 1) + 3(n + 1) = 6n$ .

We can now finally compute the number  $A_n$  of constant simplex splines in the evaluation graph  $G_n$  of a B-spline of degree  $n$ :

$$A_n = 1 + \sum_{i=1}^n 6i = 1 + 3n + 3n^2. \quad (13)$$

Hence, to compute the contribution of one triangle  $I \in \mathcal{I}$  to the value of the B-spline of degree  $n$  in a point  $x$ , we only have to evaluate  $1 + 3n + 3n^2$  constant simplex splines.

## 7 Generalizations

Although our algorithm can deal with B-splines of arbitrary degree, there are still some restrictions on its use. In this section we discuss how these restrictions are eliminated.

### 7.1 Arbitrary knot-sets

One annoying restriction is that all knot-sets  $V$  must be in general position, i.e. every triple of knots in  $V$  is linearly independent. Sometimes one deliberately introduces a few linearly dependent knot-sets to model splines with continuity less than  $C^{n-1}$ .

We can allow arbitrary knot-sets if we consider a special case in our selection scheme for split sets: if a selected split set  $W$  is linearly dependent ( $\det(W) = 0$ ), we arbitrarily choose a different split set. If no suitable split set can be found then obviously  $V$  is linearly dependent. But then  $[V] = \emptyset$  and hence  $M(x | V) = 0$  by definition of equation (5).

## 7.2 Domains in $\mathbb{R}^s$

Another important restriction is that the algorithm is still limited to domains in  $\mathbb{R}^2$ . Lifting this restriction to higher dimensional spaces is straightforward, although it requires more work than the previous generalization.

The generalizations of determinants, barycentric determinants, barycentric coordinates and the half-open convex hull are straightforward. Therefore, we only explicitly give those generalizations that affect our algorithm.

**Definition 7.1** [ $s$ -variate simplex splines] For domains in  $\mathbb{R}^s$ , simplex splines are defined over sets of at least  $s + 1$  points in  $\mathbb{R}^s$ . The recursive equation then becomes:

$$M(x | V) = \begin{cases} 0 & x \notin [V] \\ \frac{1}{|\det(V)|} & |V| = s + 1 \text{ and } x \in [V] \\ \sum_{i=0}^s \lambda_i(x | W) M(x | V \setminus \{w_i\}) & |V| > s + 1 \end{cases} \quad (14)$$

But now  $W$  is a tuple of  $s + 1$  knots and hence, the selection algorithm must be extended.

When evaluating a single simplex spline it is sufficient to choose the first  $s + 1$  elements of each knot-set, like we selected the first 3 knots in the case where  $s = 2$  (see section 4). The number of constant simplex splines in the graph then becomes  $\binom{n+s}{s}$ .

**Definition 7.2** [ $s$ -variate simplicial B-splines] To define simplicial B-splines we need a subspace  $\mathcal{I}$  of  $\mathbb{R}^s$  that is properly divided into simploids (subspaces bounded by  $s + 1$  vertices). "Properly" means that the simploids do not intersect and that if they share an edge, a hyperplane, etc, they share the *entire* edge, hyperplane, etc.

To form a basis for the B-spline we assign  $n + 1$  knots  $v_{i,0}, \dots, v_{i,n}$  to every vertex  $v_i$  in the domain, such that  $v_i = v_{i,0}$ . Furthermore, let  $I \in \mathcal{I}$  be a simpliod in the domain and let  $\beta = (\beta_0, \dots, \beta_s)$  be a tuple with  $|\beta| = \sum_{i=0}^s \beta_i = n$ , then  $V_\beta^I$  is defined as

$$V_\beta^I = \{v_{i_0,0}, \dots, v_{i_0,\beta_0}, \dots, v_{i_s,0}, \dots, v_{i_s,\beta_s}\} = \{v_{i_j,k} \mid 0 \leq j \leq s \wedge 0 \leq k \leq \beta_j\}. \quad (15)$$

After generalizing the normalization factors and defining the control coefficients, the formula for a B-spline remains exactly the same:

$$F(x) = \sum_{I \in \mathcal{I}} \sum_{|\beta|=n} d_\beta^I M(x | V_\beta^I) c_\beta^I. \quad (16)$$

Hence we have to compute the simplex splines  $V_\beta^I$  for all  $\beta$  with  $|\beta| = n$  and  $I \in \mathcal{I}$ .

**Definition 7.3** [ $s$ -variate fingerprints] To compute  $s$ -variate B-splines we generalize the definition of fingerprints to

$$F_\beta^I = \{v_{i_j,\beta_j} \mid 0 \leq j \leq s \wedge 1 \leq \beta_j\}. \quad (17)$$

Computing the number of constant simplex splines in the graph is done in the same way as for  $s = 2$ . The number of constant simplex splines in the graph of an  $s$ -variate B-spline of degree  $n$  is

$$A_n^s = 1 + \sum_{i=1}^n \sum_{k=1}^{s+1} \binom{i-1}{k-1} \binom{s+1}{k} \binom{i+s-k}{s-k}, \quad (18)$$

which is a generalization of equation 13.

Note that the algorithm described in the previous sections is exactly the special case of the general algorithm for  $s = 2$ .

Pfeifle and Seidel [PS95] also introduced a class of spherical triangular B-splines. Since this class of splines uses the same recursive pattern, it is also straightforward to use our algorithm for these splines.

## 8 Discussion

In this paper we introduced selection schemes for the efficient evaluation of simplex- and triangular B-splines. In contrast with previous approaches these evaluation schemes are able to deal with splines of arbitrary degree, any number of dimensions in the domain, arbitrary (non-general) knot-placements, and different variants of the B-splines scheme. We derived that the complexity of algorithm for the bivariate case is  $\mathcal{O}(n^2)$  where  $n$  is the degree of the spline.

To a large extent the efficiency of the algorithm is the result of our look-up table. Grandine's conclusion, that searching for previously computed results is more expensive than simply re-computing the required value, does not hold, since the graph we construct does not depend on the point in which we evaluate the spline. Hence, we only have to look-up simplex splines during the construction of the evaluation graph and can then evaluate the spline in any point without ever searching the table again.

The selection schemes to obtain efficient evaluation graphs are surprisingly simple. For simplex splines we only need to fix the order of the knots and repeatedly select the first three knots in the knot-set. For triangular B-splines we preferably select elements from the fingerprint and complete the split set with the first knots remaining from the knot-set using a fixed order.

The properties proved in lemma 6.1 and theorem 6.2 are not straightforward. If the ordering of the knots would not be fixed, but for instance, would depend on the simplex spline under consideration, the algorithm would not work as it turned out during our experiments. Also, if we use a slightly different definition for fingerprints, e.g.  $F_\beta^I = \{v_{i_j, \beta_j} \mid 0 \leq j \leq 2\}$ , the required properties do no longer hold.

Since simplex splines are uniquely defined by their knot-sets, enumerating simplex splines is a non-trivial matter. In contrast to [PS94], we avoid explicitly enumerating every simplex spline. The enumeration scheme used by Pfeifle and Seidel could never have been sufficient, since the number of possible names in their enumeration is less than the number of simplex splines that occur during the evaluation of splines of higher degree. Because of this our algorithm scales up to arbitrary degrees and arbitrary domains, and their algorithm does not.

For B-splines of degree 2 our algorithm yields the same efficient graph as Pfeifle and Seidel (see figure 3). In [PS94] 78 pairs of barycentric coordinates are computed for each triangle. This corresponds to 156 barycentric determinants, provided that the 3rd coordinate is computed as 1 minus the other two coordinates. However, by using our look-up table for barycentric determinants we avoid multiple evaluation. Therefore the number of determinants actually computed by our algorithm is the number of pairs of knots that can be chosen from the  $3n + 3$  knots of a B-spline, i.e.  $\binom{3n+3}{2}$ . For  $n = 2$  this yields only 36 determinants.

## References

- [Béz72] P. Bézier. *Numerical Control, Mathematics and Applications*. Series in Computing. Wiley, 1972.

- [DMS92] W. Dahmen, C.A. Micchelli, and H.-P. Seidel. Blossoming begets B-splines built better by B-patches. *Mathematics of Computation*, 59(199):97–115, July 1992.
- [Fra95] Michael Franssen. Evaluation of DMS-splines. Master’s thesis, Eindhoven University of Technology, 1995.
- [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.
- [Gra87] T.A. Grandine. The computational cost of simplex spline evaluation. *SIAM Journal on Numerical Analysis*, 24(4):887–890, August 1987.
- [PS94] Ron Pfeifle and Hans-Peter Seidel. Faster evaluation of quadratic bivariate DMS spline surfaces. *Graphics Interface*, pages 182–189, 1994.
- [PS95] R. Pfeifle and H.-P. Seidel. Spherical triangular B-splines with application to data fitting. In F. Post and M. Göbel, editors, *Computer Graphics Forum*, volume 14, pages C89–C96, Maastricht, the Netherlands, August 28–September 1 1995. Blackwell Publishers.
- [Sei91] H.-P. Seidel. Symmetric recursive algorithms for surfaces: B-patches and the de boor algorithm. *Constructive Approximation*, 7:257–279, 1991.
- [Tra90] C.R. Traas. *Computation of Curves and Surfaces*, chapter Practice of bivariate quadratic simplicial splines, pages 383–422. Kluwer Academic Publishers, Dordrecht, 1990.