Fast and stable evaluation of box-splines via the BB-form

Minho Kim Jörg Peters

September 7, 2008

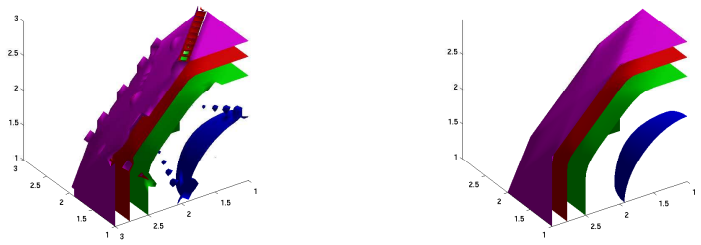
Abstract

To repeatedly evaluate linear combinations of box-splines in a fast and stable way, in particular along knot planes, the box-spline is converted to and tabulated as piecewise polynomial in BB-form (Bernstein-Bézier-form). We show that the BB-coefficients can be derived and stored as integers plus a rational scale factor and derive a hash table for efficiently accessing the polynomial pieces. This pre-processing, the resulting evaluation algorithm and use in a widely-used ray-tracing package are illustrated for splines based on two trivariate box-splines: the 7-directional box-spline on the Cartesian lattice and the 6-directional box-spline on the Face-Centered Cubic lattice.

1 Introduction

As a generalization of the univariate uniform B-spline to multivariate shift-invariant lattices, box-splines are useful in many applications. For example, they can be used to create a continuous field from data sampled at the lattice points via quasi-interpolation or cardinal interpolation. This requires exact values of the box-splines, but evaluation on lattice edges and faces requires care. Already de Boor [8] and later Kobbelt [16] observed a fundamental combinatorial challenge due to the inclusion or exclusion of certain *knot planes* (Section 3.1.5) and dealt with it in two different ways in their respective recursive box-spline evaluation algorithms. Our interest was piqued by the example of Figure 1a, where the otherwise carefully constructed algorithm of [16] fails due to subtle numerical round-off in the underlying `MATLAB`® routine.

An alternative approach to direct recursion is to evaluate after conversion to the BB-form. This approach was pioneered by Chui and Lai [3, 17] in two variables, and recently extended to a class of trivariate box-splines by Casciola et al. [1]. Since the splines we want to evaluate are linear combinations of shifts of one or more box-splines, we focus attention on these box-spline (basis or generator) functions. The point of the conversion is that the BB-form of the polynomial pieces (Section 3.2.1) has a stable evaluation algorithm also on the knot planes where the recursive algorithms encounter difficulties. In fact, along



(a) (b)

Figure 1: Isosurfaces for 10^{-1} (blue), 10^{-2} (green), 10^{-3} (red) and 10^{-10} (purple) of the 6-directional box-spline (Section 6.2), (a) evaluated by [16] and (b) the correct isosurface.

knot planes, the standard de Casteljau's algorithm for evaluating polynomials in the BB-form is of even lower complexity. The key challenges for this approach are the derivation and exact representation of the change of basis. Our first contribution, generalizing [17, 1], is Theorem 1:

(1) the BB-coefficients, expressing the polynomial pieces of a box-spline with an integer direction matrix, are *rational*.

This allows us to use a simple interpolation-based approach for deriving a change-of-basis matrix with exact integer entries, scaled by a rational number.

While both the recursive and the conversion approach benefit from localization, i.e. from determining which box-splines influence the evaluation, the conversion approach *must* efficiently determine the polynomial piece to be evaluated. This forces an understanding of the decomposition of the domain by the knot planes implied by the box-spline directions. Our second contribution is

(2) an indexing strategy, based on the box-spline directions, for finding the domain simplex of the polynomial piece for a given parameter (Section 4.1).

The one-time determination of the combinatorics of the box-splines required by the indexing strategy is at the core of the improved speed: compared to recursive evaluators that resolve the combinatorics at run time, evaluation based on the BB-form is faster by orders of magnitude (Table 3). Conversion plus indexing, both pre-computed, stored and quickly accessed, yield

(3) an algorithm for fast evaluation of splines generated by box-splines that is stable, in particular along knot planes (Algorithm 5.1).

2 Review of Existing Evaluation Techniques

Two different MATLAB® packages for evaluating box-splines, [8] and [16], are based on the recursive formula (4). These packages, which are freely available from the Internet, are immensely useful because they accommodate arbitrary direction matrices Ξ (Section 3.1.1), and are well-explained in their companion papers. As the papers point out, evaluators based on recursion face a key difficulty when evaluating a combination of shifts of the characteristic function. Unless the combinatorics of inclusion and exclusion of knot planes are correctly and *consistently* addressed, evaluation along knot planes yields incorrect results. If the evaluation is done correctly, it is called 'stable'.

De Boor [8] addresses the stability problem by perturbing evaluation points that are deemed too close to knot planes. Kobbelt [16] untangles the combinatorics explicitly to avoid round-off by deferring translation of evaluation points until the base level of the recursion where piecewise constant functions, the characteristic functions, are evaluated. This algorithm also pre-computes the normals of the knot planes in a deterministic way to avoid that a knot plane is doubly included or completely excluded by two adjacent characteristic functions. We found that [16] works well for bivariate box-splines, but the released code fails in higher dimensions as the example in Figure 1a illustrates. After analyzing the problem in more detail than we had intended, we found the flaw in the application of the MATLAB® `null` function call. Generically, the null space is determined by Singular Vector Decomposition [18]. But even minute SVD round-off errors create instability. Tellingly, we were often able to remove the instability, in the trivariate cases we tested, by adding the `'r'` parameter to the `null` function call, i.e. by enforcing close-to-rational representation via Gaussian elimination.

The algorithms of Jetter and McCool [14, 20] evaluate box-splines *approximately* by sampling in the Fourier domain followed by the inverse FFT. This way, they leverage the closed form of box-splines in the Fourier domain.

Explicit formulas for the conversion of box-splines to the polynomials in BB-form have been derived by Chui and Lai [3, 17] in two variables and by Casciola et al. [1] for a class of trivariate box-splines. The approach of [3] generates the BB-form of bivariate box-splines by comparing directional derivatives of box-splines with those of the BB-form. Applying this approach to 3- and 4-directional bivariate box-splines, [17] provides explicit Fortran codes and shows that the BB-coefficients are rational. Similarly, [1] converts the important class of trivariate box-splines spanned by four directions.

Condat and Van De Ville [5] based the evaluation of 3-directional bivariate box-splines on reduction of the box-splines to cone splines, i.e. truncated powers. Dæhlen [13] went one step further by converting the cone splines to simplex splines of one dimension lower. The approach is shown to be efficient for bivariate box-splines and, with explicit guidance along knot lines, for the 4-directional trivariate box-spline.

Cavaretta et al. [2, page 18] (see also de Boor [8, page 11]) show that, for functions satisfying refinement relations and for box-splines in particular, the

exact values on a lattice can be computed by solving an eigenvalue problem (listed as Equation (16) in Section 4.2). Values on a refined lattice can then be computed by the refinement relation (Equation (6) in Section 3.1.8). We use this fact in Section 4.2 to independently verify exactness of the BB-coefficients we compute.

Except for [14], where the goal is interpolation, all the above aim at evaluating individual box-splines. Splines in box-spline form would offhand be evaluated by evaluating shifts of the underlying box-splines individually, and then adding their contributions weighted by the coefficients.

3 Box-splines and Polynomials in BB-form

In Section 3.1, we briefly review the basic definitions and properties of box-splines and in Section 3.2, we review the multivariate polynomials in BB-form. In Section 3.3, we prove that the BB-coefficients of the polynomial pieces of a box-spline with integer directions are rational.

3.1 Box-splines

We use the notation and definitions made standard by [9]. In particular, a box-spline is a smooth piecewise polynomial of finite support and a *spline* in box-spline form is a linear combination of the shifts of a box-spline. If the shifts of a box-spline are linearly independent, the box-spline is a *basis* function.

3.1.1 Definition

Geometrically, the value of a box-spline with direction matrix $\Xi \in \mathbb{R}^{s\times n}$ at $x \in \mathrm{ran}\Xi \subset \mathbb{R}^s$ is the shadow-density [9, (I.3)]

$$M_\Xi(x) := \mathrm{vol}_{n-\dim \mathrm{ran}\Xi}\left(\Xi^{-1}\{x\} \cap \square\right) / |\det \Xi|,$$

i.e. the normalized volume of the intersection of a cube $\square \subset \mathbb{R}^n$, $n \geq s$, with the preimage $\Xi^{-1}\{x\}$ of x, an $(n - \dim \mathrm{ran}\Xi)$-dimensional subspace in $\mathbb{R}^n$. The cube or box gives the box-spline its name. In more detail,

- $\square := [0..1)^n$ is an n-dimensional half-open unit cube,
- Ξ is the $s \times n$ direction matrix, possibly with repeated columns, of the box-spline M_Ξ (Section 6 gives examples),
- $\mathrm{ran}\Xi$ is the subspace spanned by the column vectors $\{\xi : \xi \in \Xi\}$,
- $\Xi^{-1}\{x\}$ is the preimage of x when viewing Ξ as a linear transformation $\Xi : \mathbb{R}^n \to \mathbb{R}^s$ and
- $\mathrm{vol}_d(\cdot)$ is the d-dimensional volume of its argument.

In the following, we assume

$$\mathrm{rank}(\Xi) = s \quad \text{hence} \quad \mathrm{ran}\Xi = \mathbb{R}^s.$$

3.1.2 Degree and Continuity

A box-spline M_Ξ is a piecewise polynomial on $\mathrm{ran}\Xi$. Its degree is less than or equal to $k := k(\Xi) := \#\Xi - s$ where $\#\Xi$ denotes the number of columns of Ξ. The polynomial pieces join to form a function in $C^{m-1}(\mathrm{ran}\Xi)$ where [9, page 9]

$$m := m(\Xi) := \min\{\#\mathrm{Z} : \mathrm{Z} \in \mathcal{A}(\Xi)\} - 1$$

and [9, page 8] $\mathcal{A}(\Xi) := \{\mathrm{Z} \subseteq \Xi : \Xi\backslash\mathrm{Z} \text{ does not span } \mathbb{R}^s\}$.

3.1.3 Spline Space

A spline f spanned by a box-spline M_Ξ is an infinite linear combination of the shifts of the box-splines on the integer grid [9, page 33]:

$$f := \sum_{j \in \mathbb{Z}^s} M_\Xi(\cdot - j)a(j) \tag{1}$$

where $a : \mathbb{Z}^s \to \mathbb{R}$ is a *mesh function* that returns the coefficient corresponding to a mesh location.

3.1.4 Differentiation

With

- $D_\mathrm{Z} := \prod_{\zeta \in \mathrm{Z}} D_\zeta$ a composition of differential operators $D_\zeta := \sum_{\nu=1}^{s} \zeta(\nu) D_\nu$ and
- $\nabla_\mathrm{Z} := \prod_{\zeta \in \mathrm{Z}} \nabla_\zeta$ a composition of backward difference operators such that $\nabla_\zeta \phi := \phi - \phi(\cdot - \zeta)$,

a derivative of M_Ξ in the directions $\mathrm{Z} \subseteq \Xi$ equals the backward differences of $M_{\Xi\backslash\mathrm{Z}}$ along them [9, (I.30)]:

$$D_\mathrm{Z} M_\Xi = \nabla_\mathrm{Z} M_{\Xi\backslash\mathrm{Z}}. \tag{2}$$

3.1.5 Knot Planes

According to [9, (I.37)], a box-spline is a piecewise polynomial with pieces delineated by a shift-invariant mesh on $\mathbb{Z}^s$ generated by the collection of knot planes (hyperplanes spanned by columns of Ξ) $\mathbb{H}(\Xi)$ [9, page 16]:

$$\Gamma(\Xi) := \bigcup_{H \in \mathbb{H}(\Xi)} H + \mathbb{Z}^s. \tag{3}$$

The mesh $\Gamma(\Xi)$ decomposes $\mathbb{R}^s$ into convex polytopes.

3.1.6 The Recurrence Relation

As long as $M_{\Xi\setminus\xi}$ for $\xi \in \Xi$ is continuous at $x = \Xi t := \sum_{\xi\in\Xi} t_\xi \xi \in \mathbb{R}^s, t \in \mathbb{R}^n$, the box-spline M_Ξ can be evaluated recursively with the recurrence [9, (I.43)]:

$$(n-s)M_\Xi(x) = \sum_{\xi\in\Xi} t_\xi M_{\Xi\setminus\xi}(x) + (1-t_\xi)M_{\Xi\setminus\xi}(x-\xi). \tag{4}$$

Here t_ξ is the component of the vector t associated with the column $\xi \in \Xi$ by $x = \Xi t$.

3.1.7 Discrete Box-splines

A *discrete box-spline* b^h_Ξ associated with the direction matrix $\Xi \in \mathbb{Z}^{s\times n}$ and $h \in 1/\mathbb{N}$ can be constructed by the recurrence relation: [9, (VI.5)]

$$b^h_\Xi = h \sum_{j=0}^{1/h-1} b^h_{\Xi\setminus\xi}(\cdot - jh\xi) \tag{5}$$

with the base case of $b^h_{[]} = \delta$, the Dirac delta function.

3.1.8 The Refinement equation

A box-spline M_Ξ with $\Xi \in \mathbb{Z}^{s\times n}$ has the *refinement equation* [9, (VI.10)]

$$M_\Xi = \sum_{k\in h\mathbb{Z}^s} M_\Xi((\cdot - k)/h) m^h_\Xi(k) \tag{6}$$

where the *refinement mask* $m^h_\Xi := b^h_\Xi/h^s$ [9, (VII.7)].

3.1.9 Change of Variables

Given a square *generator matrix* [6] $\mathbf{M}$ of a non-Cartesian lattice, one can show with the help of [9, (I.23)] that a spline, generated by $M_{\mathbf{M}\Xi}$ shifted on the non-Cartesian lattice $\mathbf{M}\mathbb{Z}^s$, is equivalent to the one generated by M_Ξ shifted on the Cartesian lattice (integer grid) $\mathbb{Z}^s$:

$$\sum_{j\in\mathbf{M}\mathbb{Z}^s} |\det\mathbf{M}| M_{\mathbf{M}\Xi}(\cdot - j)\, b(j) = \sum_{k\in\mathbb{Z}^s} M_\Xi\left(\mathbf{M}^{-1}\cdot - k\right) b(\mathbf{M}k) \tag{7}$$

where $b : \mathbf{M}\mathbb{Z}^s \to \mathbb{R}$. Note that we need to scale $M_{\mathbf{M}\Xi}$ by $|\det\mathbf{M}|$ to obtain the *partition of unity* property of the spline.

3.2 The BB-Form of a Multivariate Polynomial

3.2.1 Definition

Let $\{v_j \in \mathbb{R}^s : j \in \{1, ..., s+1\}\}$ be the vertices of a non-degenerate simplex σ. The map

$$\beta_\sigma : \mathbb{R}^s \to \mathbb{R}^{s+1} : x \mapsto \begin{bmatrix} v_1 & \cdots & v_{s+1} \\ 1 & \cdots & 1 \end{bmatrix}^{-1} \begin{bmatrix} x \\ 1 \end{bmatrix} \tag{8}$$

is called the *barycentric coordinate function* with respect to σ [7]. The BB-form of an s-variate polynomial of total degree $\leq d$ and coefficients $\{c_\alpha \in \mathbb{R} : |\alpha| = d, \alpha \in \mathbb{Z}_+^{s+1}\}$ defined on σ, is

$$P(u) := \sum_{|\alpha|=d} c_\alpha b_\alpha(u)$$

where

- u is in barycentric coordinates w.r.t. σ,
- $|\alpha| := \sum_{i=1}^{s+1} \alpha(i)$,
- $\binom{d}{\alpha} := d!/\prod_{i=1}^{s+1} \alpha(i)!$,
- $u^\alpha := \prod_{i=1}^{s+1} u(i)^{\alpha(i)}$ and
- $\{b_\alpha(u) := \binom{d}{\alpha} u^\alpha : |\alpha| = d\}$ are the Bernstein basis polynomials of degree d.

Denote the j-th unit vector by $\mathbf{i}_j$. Then $b_\alpha(\mathbf{i}_j) = 1$ if $\alpha = d\mathbf{i}_j$ and zero otherwise. Therefore, $P|_{v_j} = P(\mathbf{i}_j) = c_{d\mathbf{i}_j}$. In other words, the BB-form of a multivariate polynomial *interpolates* its *vertex coefficients* $\{c_\alpha : \alpha = d\mathbf{i}_j, j \in \{1, ..., s+1\}\}$.

3.2.2 Differentiation

The directional derivative of P along one of the edges of σ, $v_i - v_j$, is

$$D_{v_i - v_j} P = D_{v_i - v_j} \sum_{|\alpha|=d} c_\alpha b_\alpha = d \sum_{|\beta|=d-1} (c_{\beta+\mathbf{i}_i} - c_{\beta+\mathbf{i}_j}) b_\beta. \tag{9}$$

3.3 Box-splines with Rational BB-coefficients

In [17], Lai proved for 3- and 4-directional bivariate box-splines and in [1], Casciola et al. proved for 4-directional trivariate box-splines that the BB-coefficients of those box-splines are rational. In this section, we generalize this observation.

We start by showing that rationality is preserved by box-splines with rational direction matrices.

Lemma 1. *Let* $\Xi \in \mathbb{Q}^{s\times n}$ *and* $\operatorname{rank}(\Xi) = s$. *If* $x \in \mathbb{Q}^s$ *then* $M_\Xi(x) \in \mathbb{Q}$.

Proof. We use induction on $\#\Xi$. Let $\Xi \in \mathbb{Q}^{s\times s}$. Then $1/|\det \Xi| \in \mathbb{Q}$ and $M_\Xi(x) = \chi_{\Xi\square}/|\det \Xi| \in \mathbb{Q}$ where $\chi_{\Xi\square}$ is the characteristic function of parallelepiped $\Xi\square$, a linear transformation of the unit cube $\square$.

Now assume the claim holds for $M_{\Xi\setminus\xi}$. That is $M_{\Xi\setminus\xi}(x) \in \mathbb{Q}$ and $M_{\Xi\setminus\xi}(x - \xi) \in \mathbb{Q}$ since $x - \xi \in \mathbb{Q}^s$. Since $\operatorname{rank}(\Xi) = s$, there exists an invertible submatrix $\mathrm{Z} \in \mathbb{Q}^{s\times s}$ of Ξ. Since $\mathrm{Z}^{-1} \in \mathbb{Q}^{s\times s}$, there also exists $t \in \mathbb{Q}^n$ so that $x = \Xi t$. The recursion (4),

$$(n-s)M_\Xi(x) = \sum_{\xi\in\Xi} t_\xi M_{\Xi\setminus\xi}(x) + (1 - t_\xi) M_{\Xi\setminus\xi}(x - \xi),$$

then implies that $M_\Xi(x) \in \mathbb{Q}$. □

Now denote by $\Gamma(\Xi)$ the collection of all shifts of (s-1)-dimensional hyperplanes spanned by columns of Ξ (Equation (3)). Each $H_i \in \Gamma(\Xi)$ is defined by a plane equation $n_i^T(\cdot - j_i) = 0$. We denote by *knot vertex* the intersection x of s hyperplanes $H_1, \cdots, H_s \in \Gamma(\Xi)$ whose normals span $\mathbb{R}^s$:

$$x = N^{-1}\eta, \text{ where } N := \begin{bmatrix} n_1^T \\ \vdots \\ n_s^T \end{bmatrix} \text{ and } \eta := \begin{bmatrix} n_1^T j_1 \\ \vdots \\ n_s^T j_s \end{bmatrix}. \tag{10}$$

Lemma 2. *Let $\Xi \in \mathbb{Q}^{s\times n}$ and* $\operatorname{rank}(\Xi) = s$. *Then the polynomial pieces of M_Ξ can be represented in BB-form with vertex coefficients in $\mathbb{Q}$.*

Proof. By Section 3.1.5, the piecewise polynomials of M_Ξ can be expressed over convex polytopes delineated by knot planes with rational normals. Since $\Xi \in \mathbb{Q}^{s\times n}$, we have $n_i \in \mathbb{Q}^s$ and hence $N^{-1} \in \mathbb{Q}^{s\times s}$ in (10). Since the shifts are on the integer grid, $j_i \in \mathbb{Z}^s$, and therefore all knot vertices are in $\mathbb{Q}^s$. Since any s-dimensional convex polytope can be decomposed into s-dimensional simplices without introducing any new vertex, the claim follows from Lemma 1. □

Lemma 2 can be extended to yield the main conclusion. To accommodate shifts on the Cartesian lattice, Ξ is required to have integer entries.

Theorem 1. *Let $\Xi \in \mathbb{Z}^{s\times n}$ and* $\operatorname{rank}(\Xi) = s$. *Then the polynomial pieces of M_Ξ can be represented in BB-form with coefficients in $\mathbb{Q}$.*

Proof. We use induction on $\#\Xi$. If $\Xi \in \mathbb{Z}^{s\times s}$, then the polynomial is constant and equals the value at the vertex. By Lemma 2, this value is rational.

Since $\operatorname{rank}(\Xi) = s$, for any $w \in \mathbb{Q}^s$ there exists an $y \in \mathbb{Q}^n$ so that $w = \Xi y$. By linearity of differentiation and (2),

$$D_w M_\Xi = \sum_{\xi\in\Xi} y_\xi D_\xi M_\Xi = \sum_{\xi\in\Xi} y_\xi \nabla_\xi M_{\Xi\setminus\xi}.$$

By the induction hypothesis, $M_{\Xi\setminus\xi}$ is piecewise polynomial in BB-form with coefficients in $\mathbb{Q}$. Since the knot planes are invariant under integer shifts and $\Xi \in \mathbb{Z}^{s\times n}$, $\nabla_\xi M_{\Xi\setminus\xi}$ is a difference of polynomials in BB-form with coefficients in $\mathbb{Q}$. Therefore $D_w M_\Xi$ is a polynomial in BB-form with coefficients in $\mathbb{Q}$.

Now let v_i and v_j be any two knot vertices of the domain simplex of a polynomial piece, possibly obtained by decomposition of an s-dimensional convex polytope. Then $w := v_i - v_j \in \mathbb{Q}^s$ and $D_{v_i - v_j} M_\Xi$ has rational coefficients, i.e., in the notation of (9), $(c_{\beta+\mathbf{i}_i} - c_{\beta+\mathbf{i}_j}) \in \mathbb{Q}$. The vertex coefficients are rational by Lemma 2. So rational differences propagate membership in $\mathbb{Q}$ to all the BB-coefficients c_α in (9). □

4 Preprocessing Box-Splines

We first discuss how to encode (or index) the domain simplices of the polynomial pieces for a particular direction matrix Ξ and then describe how to find the change-of-basis for conversion of box-splines to the BB-form. We note that the combinatorial work of defining the partition into domain simplices is done only once ever per box-spline generator or basis function since the result is tabulated and quickly accessed by the following indexing strategy.

4.1 Indexing Polynomial Pieces (Domains)

Unless the directions form a tensor-product, the most convenient representation of the polynomial pieces of a box-spline is the BB-form on a simplex (Section 3.2.1). The challenge is to smartly *index* each domain piece in $\mathrm{supp}M_\Xi$ to derive, store and efficiently access the BB-coefficients. Our decomposition is inspired by BSP (Binary Space Partitioning) trees (see e.g. [12, 21]). Let $\Gamma_{\square}(\Xi) \subset \Gamma(\Xi)$ be the set of knot planes of M_Ξ, each of which splits $\square$ into two s-dimensional subspaces. Then each *path* of the tree is converted into a pair of index vectors $(\mathbf{i}_\square, \mathbf{i}_\triangle) \in \mathbb{Z}^s \times \{0,1\}^q$ where $q := q(\Xi) := \#\Gamma_{\square}(\Xi)$ is the number of knot planes in $\Gamma_{\square}(\Xi)$.

Specifically (cf. Figure 2), we first circumscribe the support of the box-spline as follows. Let $\mathbf{I}_\Xi \in \mathbb{Z}^s$ be the minimal set of grid points such that

$$\mathrm{supp}M_\Xi \subset \mathbf{I}_\Xi + \square,$$

i.e. $\mathbf{I}_\Xi := \{j \in \mathbb{Z}^s : (j + \square \cap \mathrm{supp}M_\Xi) \neq \emptyset\}$. Each cube $j + \square$ for $j \in \mathbf{I}_\Xi$ is further partitioned into convex polytopes by the knot planes in $j + \Gamma_{\square}(\Xi)$. By

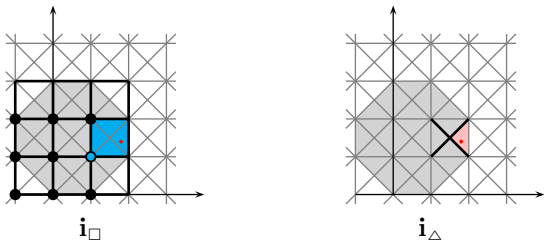


Figure 2: Indexing of the box-spline $M_{\Xi_{ZP}}$ where $\Xi_{ZP} := \left[\begin{smallmatrix} 1 & 0 & 1 & -1 \\ 0 & 1 & 1 & 1 \end{smallmatrix}\right]$ and $\mathbf{I}_{\Xi_{ZP}} = \{-1,0,1\} \times \{0,1,2\}$.

shift-invariance, each cube is partitioned alike. We now index a domain piece by

$$\mathbf{i}_\square \in \mathbf{I}_\Xi \quad \text{and}$$
$$\mathbf{i}_\triangle \in \{0,1\}^q.$$

In other words, $\mathbf{i}_\square := \lfloor \cdot \rfloor$ identifies a cube intersecting $\mathrm{supp}M_\Xi$ and $\mathbf{i}_\triangle$ identifies a polynomial piece (Figure 2). The index vector $\mathbf{i}_\triangle$ is computed by membership

test against all the knot planes in $\Gamma_{\blacksquare}(\Xi)$:

$$\mathbf{i}_{\triangle}(x) := U(N_{\Xi}(x - \lfloor x \rfloor) - \eta_{\Xi}), \quad U(t) := \begin{cases} 1, & t \geq 0 \\ 0, & t < 0, \end{cases} \tag{11}$$

where $N_{\Xi} \in \mathbb{Q}^{q \times s}$ and $\eta_{\Xi} \in \mathbb{Q}^{q}$ define the knot planes in $\Gamma_{\blacksquare}(\Xi)$.

4.2 Computing the Change of Basis

To convert M_{Ξ} to polynomial pieces in BB–form in the pre-processing step, we first need to determine a domain simplex for each piece. The domain of the polynomial piece is a convex polytope τ delineated by the knot planes. Typically, the polytope is a simplex. If not, we have several options.

- We can choose any $s+1$ vertices from τ to define the domain simplex. This, however, implies loss of the convex hull property as the BB-form will also be evaluated outside the simplex.
- We can split τ into simplices without introducing new vertices. But this makes the evaluation process more complex.
- The most practical strategy is to choose any simplex with rational vertices that encloses τ.

In the last case, it does not matter that the domain simplices of different polynomial pieces overlap, because the domain is already determined before evaluating the BB–form and we only evaluate the BB-form for points in the domain rather than on the whole enclosing simplex.

Once the domain simplex σ is determined, the change-of-basis equality for x in the domain is

$$M_{\Xi}(x) = \sum_{|\alpha|=n-s} c_{\alpha} b_{\alpha} \left(\beta_{\sigma}(x)\right). \tag{12}$$

We want to determine the $\binom{n}{s}$ BB-coefficients c_{α} of the polynomial M_{Ξ} of degree $d := n - s$ in s variables. We set up and solve the $\binom{n}{s}$ by $\binom{n}{s}$ linear system of constraints

$$\left[\binom{d}{\alpha}\left(\frac{\beta}{d}\right)^{\alpha}\right]\left[c_{\alpha}\right] = \left[M_{\Xi}\left(\frac{V_{\widetilde{\sigma}}\beta}{d}\right)\right] \tag{13}$$

arising from (12) when we choose uniformly distributed rational points $x := V_{\widetilde{\sigma}}\beta/d$ in a simplex $\widetilde{\sigma}$ $(\neq \sigma)$ with vertices $V_{\widetilde{\sigma}} := \begin{bmatrix}\widetilde{v}_1 & \cdots & \widetilde{v}_{s+1}\end{bmatrix}$, $\widetilde{v}_j \in \mathbb{Q}^s$ for $1 \leq j \leq s+1$,

$$\alpha, \beta \in \mathbb{Z}_{+}^{s+1}, |\alpha| = |\beta| = d.$$

For exactness, the vertices $V_{\widetilde{\sigma}}$ of $\widetilde{\sigma}$ should lie at rational points and for stability strictly inside the domain polytope: $\widetilde{\sigma} \subsetneq \tau \subseteq \sigma$. The existence and uniqueness of a solution c_{α} of this linear system is guaranteed by [22], a special case of Chung-Yao interpolation [4]. Note that the matrix $B := \left[\binom{d}{\alpha}\left(\frac{\beta}{d}\right)^{\alpha}\right]$ depends

only on the dimensions, s and n, of Ξ. In particular, it is independent of the simplex $\widetilde{\sigma}$ so that its inverse can be computed once to be used for all polynomial pieces of a box-spline.

While the matrix B is easily exactly inverted, the challenge is to obtain an exact right hand side $M_\Xi(V_{\widetilde{\sigma}}\beta/d)$. Reimplementing [8] (or [16]) in rational (integer-based) arithmetic does not seem practical; and rational arithmetic can be slow for high n even when we can take advantage of symmetries in the box-spline definition.

We therefore apply the following alternative approach.

(a) Choose the $s+1$ vertices of $V_{\widetilde{\sigma}}$ strictly inside the polynomial piece's domain, away from the knot planes.

(b) Compute $M_\Xi(V_{\widetilde{\sigma}}\beta/d)$ with the help of [8] (or [16]) in `MATLAB®` (due to (a), the evaluations are stable).

(c) Solve (13) in `MATLAB®` to obtain approximate coefficients $\widetilde{c}_\alpha$.

(d) Round $\widetilde{c}_\alpha$ to c_α using the `MATLAB® rat` function call to round to rational numbers [19].

Even though the rounding is in our experience no more than 10^{-10} times the coefficient size, formally one has to check correctness of the rounding. The refinement equation (6) provides the appropriate and efficient tool. We use the fact that m_Ξ^h can be computed using only integer arithmetic.

Lemma 3. *On $h\mathbb{Z}^s$, m_Ξ^h, $\Xi \in \mathbb{Z}^{s\times n}$, can be computed exactly using integer computation.*

Proof. Let

$$B_\Xi^h := \frac{1}{h^{\#\Xi}} b_\Xi^h(h\cdot) = \frac{1}{h^{\#\Xi - s}} m_\Xi^h(h\cdot)$$

hence, for $k = hi \in h\mathbb{Z}^s$, $i \in \mathbb{Z}^s$, $b_\Xi^h(k) = h^{\#\Xi} B_\Xi^h(i)$. Then the recurrence relation (5) becomes

$$B_\Xi^h = \sum_{j=0}^{1/h-1} B_{\Xi\backslash\xi}^h(\cdot - j\xi)\,. \tag{14}$$

Since $\Xi \in \mathbb{Z}^{s\times n}$, this allows B_Ξ^h to be computed by integer computation on $\mathbb{Z}^s$. Therefore b_Ξ^h and hence m_Ξ^h can be computed exactly on $h\mathbb{Z}^s$ and has the denominator $1/h^{\#\Xi - s}$. □

To verify correct rounding of the BB-coefficients of M_Ξ, we observe that the refinement equation (6) can be converted to

$$M_\Xi(\alpha) = \sum_{\beta\in\mathbb{Z}^s} M_\Xi(\alpha/h - \beta) m_\Xi^h(h\beta), \quad \alpha \in \mathbb{Z}^s. \tag{15}$$

Since $\mathrm{supp} M_\Xi$ is finite, so is $\mathrm{supp} M_\Xi \cap \mathbb{Z}^s$ and we can choose any finite superset $K \subset \mathbb{Z}^s$ containing $\mathrm{supp} M_\Xi \cap \mathbb{Z}^s$. Cavaretta et al. [2, page 18] proved that any

convergent subdivision scheme has the simple (or single) eigenvalue 1 so that the eigenvalue problem (15), augmented with the condition $\sum_{j\in \mathrm{supp} M_\Xi \cap \mathbb{Z}^s} M(j) = 1$, results in the system

$$M_\Xi(\alpha) = \sum_{\beta\in\mathbb{Z}^s} M_\Xi(\alpha/h - \beta) m^h_\Xi(h\beta), \quad \alpha \in K, \quad \sum_{\alpha\in K} M(\alpha) = 1 \tag{16}$$

and the *unique* eigenvector

$$\{M_\Xi(\alpha) : \alpha \in K\}, \tag{17}$$

whose entries are the box-spline evaluated on $\mathbb{Z}^s$.

We can now construct what we hope to be the exact eigenvector (17) by evaluating the BB-form with its rational coefficients that we obtained by rounding to a rational representation. Since each polynomial piece has finite degree, it suffices to test on a refined grid $h\mathbb{Z}^s$ so that each domain contains more evaluation points than its BB-coefficients. We emphasize that we do not solve the eigenvalue problem (16) but only verify, by comparing the values, that equality holds. Since we know all the exact denominators involved in the verification, we can pre-scale (6) and (16) so that only integer computations are required.

We also pre-compute the matrices in Equation (8) that compute barycentric coordinates u with respect to a domain simplex σ for a point $x \in \mathbb{R}^s$ in Cartesian coordinates.

5 The Spline Evaluation Algorithm

Given the indexing (hash table), the table of BB-coefficients and the pre-computed inverse matrices for computing barycentric coordinates, we can evaluate splines, i.e. linear combinations of the shifts of box-splines, efficiently and stably. The following algorithm evaluates a spline with box-spline directions Ξ and box-spline coefficients a (treated as a mesh function) at a point x. The steps are as follows. First, we find the domain index $\mathbf{i}_\triangle$ of the point x using the membership test (11). We then compute the barycentric coordinates u of x with respect to the domain simplex. Shifts of the index are used to pick up, via the hash table, all BB-coefficients that stem from box-spline shifts whose supports overlap x; and to form their linear combination with weights from a. The coefficients of the resulting polynomial are stored in P. Finally, the algorithm evaluates this polynomial with coefficient vector P. The following is the pseudocode for the

spline evaluation.

Algorithm 5.1: EVALUATESPLINE $_{\Xi}$ (a, x)

$\mathbf{i}_{\triangle} \leftarrow U(N_{\Xi}(x - \lfloor x \rfloor) - \eta_{\Xi})$

$u \leftarrow$ COMPUTEBARYCENTRIC$(\mathbf{i}_{\triangle}, x - \lfloor x \rfloor)$

$P \leftarrow \sum_{\mathbf{i}_{\square} \in \mathbf{I}_{\Xi}} a(\lfloor x \rfloor - \mathbf{i}_{\square})\ C_{\Xi}(\mathbf{i}_{\square}, \mathbf{i}_{\triangle})$

return EVALUATEBB(P, u)

Here the subscript $_{\Xi}$, rather than an argument Ξ, emphasizes that the algorithm requires the pre-processing with respect to Ξ according to Section 4,

- $a(\lfloor x \rfloor - \mathbf{i}_{\square}) \in \mathbb{R}$ is the box-spline coefficient with index $\lfloor x \rfloor - \mathbf{i}_{\square} \in \mathbb{Z}^s$,
- x is the input point (in Cartesian coordinates) to be evaluated,
- N_{Ξ} and η_{Ξ} define the knot planes in $\Gamma_{\blacksquare}(\Xi)$, (Section 4.1)
- COMPUTEBARYCENTRIC computes the barycentric coordinate using (8),
- $C_{\Xi}(\mathbf{i}_{\square}, \mathbf{i}_{\triangle})$ is a vector of all coefficients of the polynomial piece in BB-form with index $(\mathbf{i}_{\square}, \mathbf{i}_{\triangle})$, retrieved from the hash table, and
- EVALUATEBB evaluates the polynomial in BB–form with coefficients P at u.

6 Examples

We illustrate the initial conversion and generation of the index function for two trivariate box-splines that are useful for reconstructing volumetric data: the 7-directional box-spline on the Cartesian lattice [24, 25, 11] and the 6-directional box-spline on the FCC lattice [10]. Both box-splines are symmetric and of low degree given their smoothness.

6.1 The 7-directional Box-Spline

6.1.1 Definition

We consider the 7-directional trivariate box-spline defined by the direction matrix

$$\Xi_7 := \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 1 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 1 & 1 & -1 & -1 & 1 \end{bmatrix}.$$

6.1.2 Degree and Continuity

The box-spline is piecewise polynomial of degree $k(\Xi_7) = 7 - 3 = 4$. It is remarkable, since $M_{\Xi_7} \in C^2(\mathbb{R}^3)$, because at most 3 directions span a hyperplane, $m(\Xi_7) = (7-3) - 1 = 3$.

6.1.3 Indexing Domain Tetrahedra

The cube $\square$ is decomposed by $q = \#\Gamma_{\square}(\Xi_7) = 6$ planes (Figure 3):

$$N_{\Xi_7} x := \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 1 \\ -1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} =: \eta_{\Xi_7}.$$

The knot planes in $\Gamma_{\square}(\Xi_7)$ split $\square$ into the 24 tetrahedra listed in Table 1 and Figure 4.

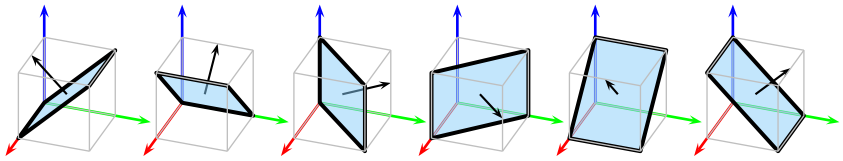


Figure 3: Six knot planes in $\Gamma_{\square}(\Xi_7)$.

Table 1: The 7-directional box-spline M_{Ξ_7}. The knot planes in $\Gamma_{\square}(\Xi_7)$ split $\square$ into 24 tetrahedra. ($v_c := (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$)

$\mathbf{i}_{\triangle}$	vertices				$\mathbf{i}_{\triangle}$	vertices			
0 0 0 1 1 1	v_c	1 $\frac{1}{2}$ $\frac{1}{2}$	1 1 0	1 1 1	0 1 1 0 0 1	v_c	0 $\frac{1}{2}$ $\frac{1}{2}$	0 1 1	0 1 0
1 0 0 1 1 0	v_c	1 $\frac{1}{2}$ $\frac{1}{2}$	1 0 1	1 0 0	1 1 1 0 0 0	v_c	0 $\frac{1}{2}$ $\frac{1}{2}$	0 0 0	0 0 1
1 0 0 1 1 1	v_c	1 $\frac{1}{2}$ $\frac{1}{2}$	1 1 1	1 0 1	1 1 1 0 0 1	v_c	0 $\frac{1}{2}$ $\frac{1}{2}$	0 0 1	0 1 1
0 0 0 1 1 0	v_c	1 $\frac{1}{2}$ $\frac{1}{2}$	1 0 0	1 1 0	0 1 1 0 0 0	v_c	0 $\frac{1}{2}$ $\frac{1}{2}$	0 1 0	0 0 0
0 0 1 1 1 1	v_c	$\frac{1}{2}$ 1 $\frac{1}{2}$	1 1 1	1 1 0	1 0 0 0 1 0	v_c	$\frac{1}{2}$ 0 $\frac{1}{2}$	1 0 0	1 0 1
0 1 1 1 0 1	v_c	$\frac{1}{2}$ 1 $\frac{1}{2}$	0 1 0	0 1 1	1 1 0 0 0 0	v_c	$\frac{1}{2}$ 0 $\frac{1}{2}$	0 0 1	0 0 0
0 1 1 1 1 1	v_c	$\frac{1}{2}$ 1 $\frac{1}{2}$	0 1 1	1 1 1	1 1 0 0 1 0	v_c	$\frac{1}{2}$ 0 $\frac{1}{2}$	1 0 1	0 0 1
0 0 1 1 0 1	v_c	$\frac{1}{2}$ 1 $\frac{1}{2}$	1 1 0	0 1 0	1 0 0 0 0 0	v_c	$\frac{1}{2}$ 0 $\frac{1}{2}$	0 0 0	1 0 0
1 1 0 1 1 1	v_c	$\frac{1}{2}$ $\frac{1}{2}$ 1	1 0 1	1 1 1	0 0 0 1 0 0	v_c	$\frac{1}{2}$ $\frac{1}{2}$ 0	1 1 0	1 0 0
1 1 1 0 1 1	v_c	$\frac{1}{2}$ $\frac{1}{2}$ 1	0 1 1	0 0 1	0 0 1 0 0 0	v_c	$\frac{1}{2}$ $\frac{1}{2}$ 0	0 0 0	0 1 0
1 1 1 1 1 1	v_c	$\frac{1}{2}$ $\frac{1}{2}$ 1	1 1 1	0 1 1	0 0 1 1 0 0	v_c	$\frac{1}{2}$ $\frac{1}{2}$ 0	0 1 0	1 1 0
1 1 0 0 1 1	v_c	$\frac{1}{2}$ $\frac{1}{2}$ 1	0 0 1	1 0 1	0 0 0 0 0 0	v_c	$\frac{1}{2}$ $\frac{1}{2}$ 0	1 0 0	0 0 0

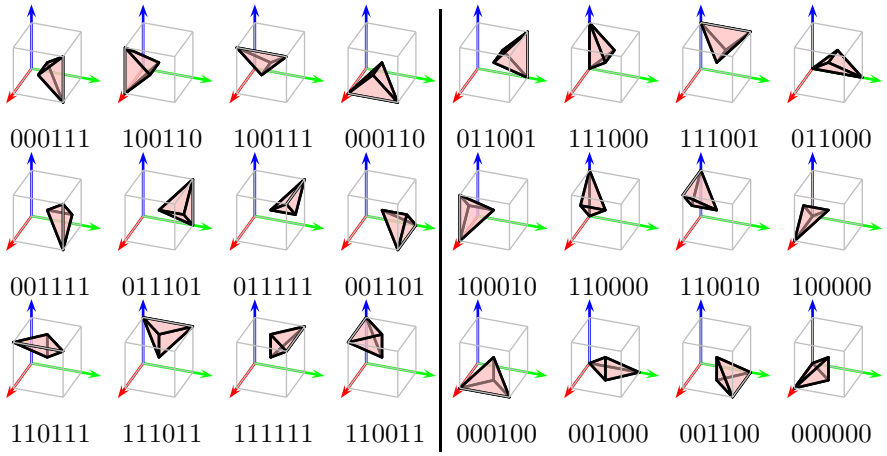


Figure 4: 24 tetrahedra generated by 6 knot planes in $\Gamma_{\blacksquare}(\Xi_7)$.

6.2 The 6-directional Box-Spline on the FCC Lattice

6.2.1 Definition

A 6-directional trivariate box-spline can be defined by the following direction matrix [10]:

$$\widetilde{\Xi}_6 := \mathbf{M}_{\text{fcc}}\Xi_6 := \begin{bmatrix} 0 & 0 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & -1 \end{bmatrix}$$

where

$$\mathbf{M}_{\text{fcc}} := \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \Xi_6 := \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & -1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

6.2.2 Spline Space

The 6-directional box-spline is associated with the FCC (Face-Centered Cubic) lattice. By (7), splines on the FCC lattice are generated by the shifts of the (scaled) box-spline $|\det \mathbf{M}_{\text{fcc}}| M_{\mathbf{M}_{\text{fcc}}\Xi_6}$ since $\mathbf{M}_{\text{fcc}}$ is the generator matrix for the FCC lattice [6].

6.2.3 Degree and Continuity

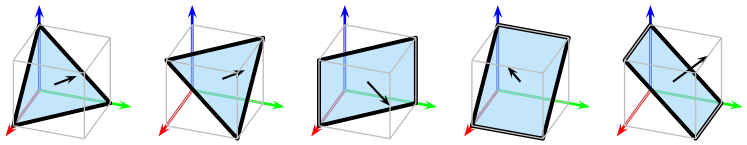
The box-spline $M_{\widetilde{\Xi}_6}$ is piecewise polynomial of total degree $\leq k(\widetilde{\Xi}_6) = 6-3 = 3$. At most 3 directions in $\widetilde{\Xi}_6$ span a hyperplane. Therefore $m(\widetilde{\Xi}_6) = (6-3)-1 = 2$ and $M_{\widetilde{\Xi}_6} \in C^1(\mathbb{R}^3)$.

6.2.4 Indexing Domain Tetrahedra

We have $q = \#\Gamma_{\square}(\Xi_6) = 5$ planes defined by (Figure 5)

$$N_{\Xi_6} := \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}^T \quad \text{and} \quad \eta_{\Xi_6} := \begin{bmatrix} 1 & 2 & 1 & 1 & 1 \end{bmatrix}^T.$$

$\square$ is split into 10 tetrahedra as specified in Table 2 and Figure 6.



$(1,1,1)$ $(1,1,1)$ $(1,1,0)$ $(1,0,1)$ $(0,1,1)$

Figure 5: 5 knot planes in $\Gamma_{\square}(\Xi_6)$.

Table 2: The 6-directional box-spline M_{Ξ_6}. The knot planes in $\Gamma_{\square}(\Xi_6)$ split $\square$ into 10 tetrahedra.

$\mathbf{i}_{\triangle}$		vertices			$\mathbf{i}_{\triangle}$		vertices		
0 0 0 0 0	0 0 0	1 0 0	0 1 0	0 0 1	1 1 1 1 1	1 1 1	1 0 1	0 1 1	1 1 0
1 0 0 1 0	v_c	1 0 1	0 0 1	1 0 0	1 0 0 1 1	v_c	0 1 1	0 0 1	1 0 1
1 0 0 0 1	v_c	0 0 1	0 1 1	0 1 0	1 0 0 0 0	v_c	0 0 1	0 1 0	1 0 0
1 0 1 1 0	v_c	1 0 1	1 0 0	1 1 0	1 0 1 1 1	v_c	0 1 1	1 0 1	1 1 0
1 0 1 0 0	v_c	0 1 0	1 1 0	1 0 0	1 0 1 0 1	v_c	0 1 1	1 1 0	0 1 0

7 Comparison and an Application

Table 3 illustrates the relative efficiency of [8], [16] and Algorithm 5.1. The table entries are the result of densely evaluating in one octant of the 6-directional trivariate box-spline, respectively of the 7-directional trivariate box-spline. No linear combination of box-splines is evaluated. All three MATLAB® implementations are designed to handle vector input and avoid for-loops. The measurements used MATLAB® on a Linux system with Intel® Core™2 CPU 6400 @2.13GHz (2MB cache) and 2GB memory. The comparison shows Algorithm 5.1 to be faster by orders of magnitude. We explain the difference in speed as the result of pre-resolution of box-spline combinatorics, as encoded in the indexing and the tabulation of BB-coefficients prior to running Algorithm 5.1. The other two packages are more general and resolve the combinatorics at run time.

To compute high-quality ray-tracing of level sets of a 3D field reconstructed by the shifts of a trivariate box-spline, we implemented a MATLAB® script that exports the BB-form of a spline formed by a box-spline [15]. Specifically, we

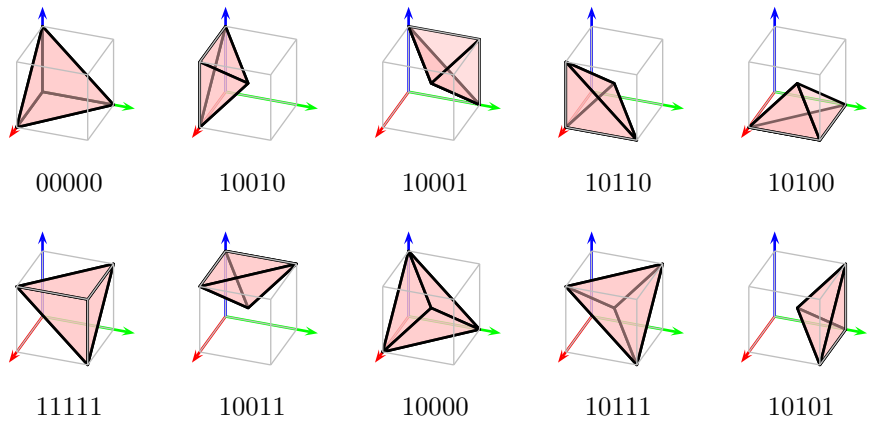


Figure 6: 10 tetrahedra generated by the 5 knot planes in $\Gamma_{\square}(\Xi_6)$.

output POV-Ray[23] script format. POV-Ray is a popular and freely available ray-tracing engine. The setup requires only adding one internal function to POV-Ray that evaluates a polynomial in BB-form, e.g. using de Casteljau's algorithm. Figures 7 and 8 show examples.

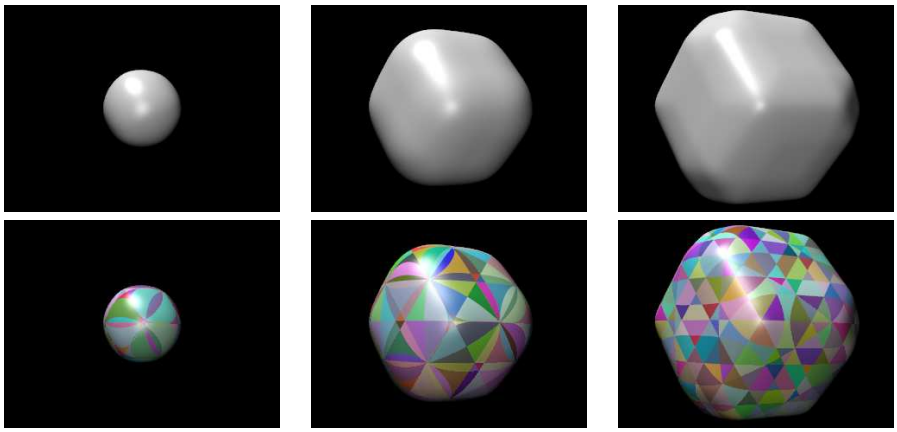


Figure 7: Ray-traced images of several level sets of the 7-directional box-spline. In the bottom images, a random color is assigned to each polynomial piece. The images are rendered by POV-Ray [23].

Such high-quality ray-tracing is hard to obtain by subdivision, unless the ray-tracing algorithm is carefully designed for recursion. Use of [8] or [16] is precluded by their lack of speed.

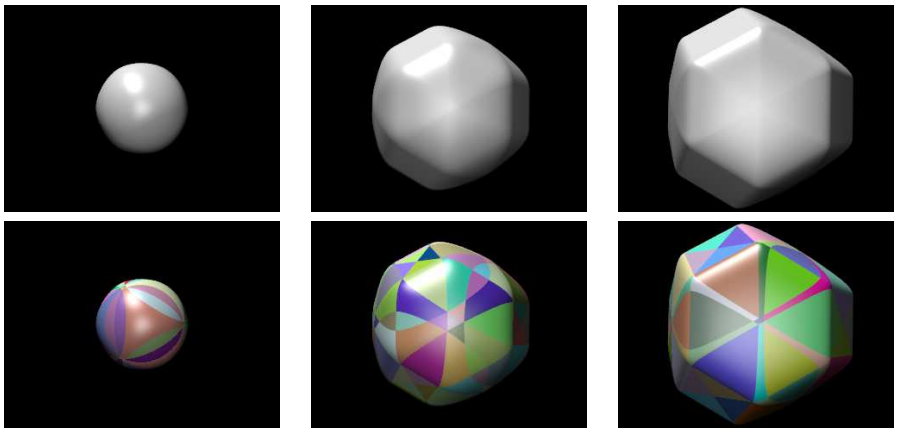


Figure 8: Ray-traced images of several level sets of the un-scaled 6-directional box-spline $M_{\tilde{\Xi}_6}$. In the bottom images, a random color is assigned to each polynomial piece. The images are rendered by POV-Ray [23].

References

[1] Giulio Casciola, Elena Franchini, and Lucia Romani, *The mixed directional difference-summation algorithm for generating the Bézier net of a trivariate four-direction box-spline*, Numerical Algorithms **43** (2006), no. 1, 1017–1398.

[2] Alfred S. Cavaretta, Charles A. Micchelli, and Wolfgang Dahmen, *Stationary subdivision*, American Mathematical Society, Boston, MA, USA, 1991.

[3] Charles K. Chui and Ming-Jun Lai, *Algorithms for generating B-nets and graphically displaying spline surfaces on three- and four-directional meshes*, Computer Aided Geometric Design **8** (1991), no. 6, 479–493.

[4] K. C. Chung and T. H. Yao, *On lattices admitting unique lagrange interpolations*, SIAM Journal on Numerical Analysis **14** (1977), no. 4, 735–743.

[5] Laurent Condat and Dimitri Van De Ville, *Three-directional box-splines: Characterization and efficient evaluation*, Signal Processing Letters, IEEE **13** (2006), no. 7, 417–420.

[6] John Horton Conway and Neil J. A. Sloane, *Sphere packings, lattices and groups*, third ed., Springer-Verlag New York, Inc., New York, NY, USA, 1998.

[7] Carl de Boor, *B-form basics*, Geometric modeling, SIAM, Philadelphia, PA, 1987, pp. 131–148.

[8] Carl de Boor, *On the evaluation of box splines*, Numerical Algorithms **5** (1993), no. 1–4, 5–23.

Table 3: Comparison of evaluation time of three packages for N^3 points distributed over: $[0.5..3]^3$ for the 7-directional box-spline and $[1..3]^3$ for the 6-directional box-spline.

algorithm	spline	time (× multiple of Alg. 5.1)		
		21^3	31^3	41^3
[8]	7-dir	20.273 (×144)	75.297 (×154)	187.716 (×153)
	6-dir	1.867 (×34)	7.088 (×39)	18.147 (×41)
[16]	7-dir	52.728 (×375)	207.841 (×424)	550.423 (×450)
	6-dir	3.645 (×66)	14.035 (×78)	37.232 (×84)
Alg. 5.1	7-dir	0.141	0.490	1.223
	6-dir	0.055	0.181	0.445

[9] Carl de Boor, Klaus Höllig, and Sherman Riemenschneider, *Box splines*, Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[10] Alireza Entezari, *Optimal sampling lattices and trivariate box splines*, Ph.D. thesis, Simon Fraser University, Vancouver, Canada, July 2007.

[11] Alireza Entezari and Torsten Möller, *Extensions of the Zwart-Powell box spline for volumetric data reconstruction on the Cartesian lattice*, IEEE Transactions on Visualization and Computer Graphics **12** (2006), no. 5, 1337–1344.

[12] Henry Fuchs, Zvi M. Kedem, and Bruce F. Naylor, *On visible surface generation by a priori tree structures*, SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques (New York, NY, USA), ACM Press, 1980, pp. 124–133.

[13] Morten Dæhlen, *On the evaluation of box splines*, Mathematical methods in computer aided geometric design (San Diego, CA, USA), Academic Press Professional, Inc., 1989, pp. 167–179.

[14] Kurt Jetter and Joachim Stöckler, *Algorithms for cardinal interpolation using box splines and radial basis functions*, Numerische Mathematik **60** (1991), no. 1, 97–114.

[15] Minho Kim, *`tribox`: Matlab packages for evaluation of 6- and 7-directional trivariate box-splines*, `http://www.cise.ufl.edu/research/SurfLab/tribox`.

[16] Leif Kobbelt, *Stable evaluation of box-splines*, Numerical Algorithms **14** (1997), no. 4, 377–382.

[17] Ming-Jun Lai, *Fortran subroutines for B-nets of box splines on three- and four-directional meshes*, Numerical Algorithms **2** (1992), no. 1, 33–38.

[18] MathWorks, *Matlab 7 Function Reference: Volume 2 (F-O)*, 2006.

[19] ———, *Matlab 7 Function Reference: Volume 3 (P-Z)*, 2006.

[20] Michael D. McCool, *Accelerated evaluation of box splines via a parallel inverse FFT*, Computer Graphics Forum **15** (1996), no. 1, 35–45.

[21] Bruce F. Naylor, *Binary space partitioning trees*, Handbook of Data Structures and Applications, Chapman & Hall/CRC, 2005 (Mehta and Sahni, eds.), 2005.

[22] R. A. Nicolaides, *On a class of finite elements generated by lagrange interpolation*, SIAM Journal on Numerical Analysis **9** (1972), no. 3, 435–445.

[23] Persistence of Vision Pty. Ltd., *POV-Ray: The Persistence of Vision Raytracer*, `http://www.povray.org`.

[24] Jörg Peters, *C^2 surfaces built from zero sets of the 7-direction box spline*, IMA Conference on the Mathematics of Surfaces (Glen Mullineux, ed.), Clarendon Press, 1994, pp. 463–474.

[25] Jörg Peters and Michael Wittman, *Box-spline based CSG blends*, Proceedings of the fourth ACM symposium on Solid modeling and applications, SIGGRAPH, ACM Press, 1997, pp. 195–205.