Three-directional Box-Splines: Characterization and Efficient Evaluation

Laurent Condat, Student Member, IEEE, and Dimitri Van De Ville, Member, IEEE

Abstract—We propose a new characterization of three-directional box-splines, which are well adapted for interpolation and approximation on hexagonal lattices. Inspired by a construction already applied with success for exponential splines [1] and hex-splines [2], we characterize a box-spline as a convolution of a generating function, that is a Green function of the spline's associated differential operator, and a discrete filter that plays the role of a localization operator. This process leads to an elegant analytical expression of three-directional box-splines. It also brings along a particularly efficient implementation.

Index Terms—Box-splines, hexagonal sampling, three-directional mesh, interpolation, approximation.

I. INTRODUCTION

The representation of a digital signal by means of a discrete/continuous model is essential for common tasks such as interpolation and resampling. For images and other two-dimensional (2-D) data, polynomial spline models based on B-splines are particularly popular, mainly due to their simplicity and excellent approximation capabilities [3].

For image data sampled on the traditional Cartesian lattice, separable B-splines can be obtained in a straightforward way using tensor products of one-dimensional (1-D) B-splines. However, in the case of sampling on a hexagonal lattice (a.k.a. three-directional mesh) separable B-splines are incapable to exploit the highly praised isotropy and twelve-fold symmetry of this sampling scheme [4], [5]. Box-splines are a multi-dimensional extension of 1-D splines [6] that have found practical applications in geometrical modelling, multiscale representation, and many other fields [7]. Among the large box-spline family, three-directional (non-separable) box-splines are particularly suitable for hexagonal lattices. They have been successfully applied in numerous problems where hexagonally sampled data are handled [8], [9].

Early algorithms to evaluate box-spline surfaces were very memory consuming and only resulted into an approximation of the surface within a given tolerance [10]–[12]. Later, more efficient methods were proposed based on the recursive properties of box-splines [13]–[16]. Here, we propose a new characterization of three-directional box-splines that provides us with a closed analytical formula, as well as an efficient implementation scheme. To this aim, we derive an explicit form of the generating function, which is the Green function of a three-directional differential operator associated with box-splines. Then, the box-spline can be expressed as the convolution of the generating function with a discrete filter, which plays the role of a localization operator. A similar construction was already applied on the Cartesian lattice to generalized polynomial splines (i.e., exponential splines and L-splines [1]) and to the design of another family of splines on the hexagonal lattice (i.e., hex-splines [2]).

II. BOX-SPLINES ON THE HEXAGONAL LATTICE

A. Mathematical preliminaries

A 2-D lattice is a set of points of the plane, characterized by two linearly independent vectors \(v_1\) and \(v_2\) grouped in a matrix \(R = [v_1, v_2]\), such that the lattice sites are the locations \(Rk\) for every \(k \in \mathbb{Z}^2\). Within this letter, we define the vectors \(e_1 = [1 0]^T\), \(e_2 = [0 1]^T\), and those shown in Fig. 1 as

\[
\begin{align*}
\mathbf{r}_1 &= \left[ \begin{array}{c} 1/2 \\ -\sqrt{3}/2 \end{array} \right], \\
\mathbf{r}_2 &= \left[ \begin{array}{c} 1/2 \\ \sqrt{3}/2 \end{array} \right], \\
\mathbf{r}_3 &= \left[ \begin{array}{c} 1 \\ 0 \end{array} \right].
\end{align*}
\]

The Cartesian lattice is then obtained for \(R = [e_1, e_2]\), and the regular hexagonal lattice, as in Fig. 1, for \(R = [r_1, r_2]\).

Bivariate functions are equivalently denoted as \(f(x_1, x_2)\), \(x_1, x_2 \in \mathbb{R}\), or \(f(\omega)\), where \(\omega = [x_1 x_2]^T\) is interpreted as a vector in \(\mathbb{R}^2\). The Fourier transform of a function \(f(x)\) in \(L_2(\mathbb{R}^2)\) is defined as \(\hat{f}(\omega) = \int_{\mathbb{R}^2} f(x) \exp(-j(\omega, x)) dx\), where \((\omega, x) = \omega^T x\) is the usual inner product of vectors.

A 2-D discrete signal is denoted as \(s[k] = s[k_1, k_2]\), \(k_1, k_2 \in \mathbb{Z}\). Its representation in the continuous domain, associated with the lattice sites \(Rk\), is a weighted Dirac comb: \(s(x) = \sum_{k \in \mathbb{Z}^2} s[k] \delta(x - Rk)\). Consequently, its Fourier transform is defined accordingly as \(\hat{s}(\omega) = \sum_{k \in \mathbb{Z}^2} s[k] \exp(-j(\omega, Rk))\). For \(z = \exp(-jR^T \omega)\), we get the \(\mathbb{Z}\)-transform of \(s(z) = s[k] z^{-k}\) (\(z^{-k}\) means \(z_1^{-k_1} z_2^{-k_2}\)). Continuous and discrete convolutions are denoted by \(\ast\).

B. Definition

A 2-D box-spline model defined on a lattice \(R\) has the form

\[
f(x) = \sum_{k \in \mathbb{Z}^2} c[k] \varphi_{\Xi}(x - Rk), \quad x \in \mathbb{R}^2,
\]

where \(c[k]\) are the box-spline coefficients that are weights for the box-spline basis functions \(\varphi_{\Xi}(x)\), placed on every lattice site. They can be computed to ensure a desired property, typically that \(f\) interpolates a discrete available signal \(s\) (i.e. \(f(Rk) = s[k]\) for every \(k\)). The box-spline \(\varphi_{\Xi}(x)\) depends on a concatenated matrix of \(N\) vectors \(\Xi = [v_1 \cdots v_N]\).
(N ≥ 2), and can be defined as follows [6]: if Ξ = [v1, v2], then
\[
\varphi[v_1, v_2](x) = \begin{cases} 
1/(|\det(\Xi)|), & \text{if } \Xi^{-1}x \in [0, 1)^2, \\
0, & \text{otherwise,}
\end{cases}
\]
and inductively, \(\varphi_{\Xi \cup [v_1]}(x) = \int_0^1 \varphi_{\Xi}(x - tv) dt\).
Therefore, we have the normalization \(\int_{R^2} \varphi_{\Xi} = 1\) and the convolution property \(\varphi_{\Xi \cup v} = \varphi_{\Xi} * \varphi_v\).

On a hexagonal lattice, box-splines can be constructed using the three vectors \(r_1, r_2, -r_3\). In particular, we define the so-called Courant element [6] as \(\chi^1 = \sqrt{3} \varphi_{[r_1, r_2, -r_3]}\), where we have changed the normalization towards the density of the lattice, i.e., \(|\det R| = \sqrt{3}\). Further on, higher orders are obtained as \(\chi^n = \frac{\sqrt{3}}{\sqrt{3}} \chi^{n-1} \ast \chi^1, n > 1\). Their expression in the Fourier domain is
\[
\hat{\chi}^n(\omega) = \frac{\sqrt{3}}{2} \left( \frac{\exp(j(\omega, r_1))}{(\omega, r_1)} \prod_{i=1}^3 1 - \exp(-j(\omega, r_i)) \right)^n
= \frac{\sqrt{3}}{2} \prod_{i=1}^3 \sin(c(\omega, r_i)/2)^n.
\]

where \(\sin(c(x)) = \sin(x)/x\). The box-splines \(\chi^n(\omega)\) have several attractive properties such as an hexagonal compact support and twelve-fold symmetry, as illustrated in Figs 1 and 2. In the next section, we provide closed analytical formulas for these box-splines in the spatial domain.

III. DIFFERENTIAL CHARACTERIZATION OF BOX-SPLINES
A. B-spline refresher

In the 1-D case, a polynomial spline \(f(x)\) for uniformly sampled data can be expressed similarly to (2) as
\[
f(x) = \sum_{k \in Z} c[k] \beta^n(x - k), \quad \beta^n \text{ is the causal B-spline of degree } n \in \mathbb{N},
\]
which is defined in the spatial domain as
\[
\beta^n(x) = \Delta^{n+1} * (x)^n/n!. \tag{6}
\]
We identify \(\Delta^n\) as the \(n\)-th iterate of the finite difference filter, which is usually expressed in the \(Z\)-domain as \(\Delta^n(z) = (1 - z^{-1})^n\). Further on, we have the one-sided power function
\[1\]Note that the 1-D B-spline can also be obtained recursively as \(\beta^n = \beta^{n-1} \ast \beta^0, n > 0\), with \(\beta^0 = \mathbb{I}_{[0,1]}\) the indicator function of the unit interval.

\[
(x)^n_+ = \{x^n, \text{for } x > 0; 0, \text{otherwise}\}. \text{The filtering process acts as a localization operator on the power function; i.e., } \beta^n \text{ has a finite support. The term } (x)^n_+/n! \text{ is also called the generating function and it corresponds to the (causal) Green function of the differential operator } L^n = d^n/dx^n; \text{i.e., the function } \rho(x) \text{ such that } L^n \{\rho\}(x) = \delta(x). \text{This means that a polynomial spline of degree } n, \text{ when differentiated } n + 1 \text{ times, is a weighted Dirac comb.}

On the 2-D Cartesian lattice, we can easily use tensor-product B-splines: \(\beta^n(\omega) = \beta^n(x_1) \beta^n(x_2)\). Then, the associated differential operator is
\[
L^n = \frac{\partial^{2n}}{\partial x_1^2 \partial x_2^2} = D_{x_1}^n D_{x_2}^n \longleftrightarrow (j(\omega, e_1))^n (j(\omega, e_2))^n, \tag{7}
\]
where \(D_x f(x) = \lim_{\Delta x \to 0} (f(x + \Delta x) - f(x))/\Delta x\). In that case, the (separable) generating function is \(x^n_+/(n!)^2 = (x_1^n_+ (x_2^n_+)/(n!))^2\) and the corresponding localization operator \(\Delta^n(x) = \Delta^n(z_1) \Delta^n(z_2)\).

B. From differential operators to generating functions

Inspired by the B-spline construction using Green functions, we propose an extension for the box-splines on the hexagonal lattice. For this purpose, we introduce the three-directional differential operator \(L^n = 3 \partial_{x_1}^n D_{x_1}^n D_{x_2}^n, n \geq 1\). Its Fourier transform, in the sense of the distributions, is
\[
\hat{L^n}(\omega) = \frac{2}{\sqrt{3}} \langle j(\omega, r_1) \rangle^n (j(\omega, r_2))^n (j(\omega, r_3))^n. \tag{8}
\]

PROPOSITION: A Green function \(\rho^n(x)\) of the operator \(L^n\), \(n \geq 1\), is given by
\[
\rho^n(x) = \sum_{i=0}^{n-1} \left( n - 1 + i \right) \mu^{n-1-i,2n+1-i}(x), \tag{9}
\]
where
\[
\mu^{n_1,n_2}(x_1, x_2) = \frac{1}{n_1!n_2!} \left( \frac{2|x_2|}{\sqrt{3}} \right)^{n_1} \left( 1 - \frac{|x_2|}{\sqrt{3}} \right)^{n_2}. \tag{10}
\]

The proof is given in Appendix I. Notice that the functions \(\mu^{n_1,n_2}\) and \(\rho^n\) all have the same wedge-like support; they are causal in \(x_1\) and symmetric in \(x_2\), as illustrated in Fig. 3.

C. From generating functions to box-splines

In the Fourier domain, the generating function \(\rho^n\) corresponds to \(\hat{\chi}^n\) without its numerator in (4). The remaining term can be identified by introducing the discrete filter
\[
\Delta(z) = (1 - z_1^{-1})(1 - z_2^{-1})(z_1z_2 - 1). \tag{11}
\]
Using the property $r_3 = r_1 + r_2$, we find that $\hat{\Delta}^n(\omega) = \Delta^n(\exp(j\omega)) \exp(j\omega_1) \exp(j\omega_2)$ is exactly the numerator of (4). We can explicitly find the filter coefficients of $\Delta^n$ by expanding the $n$-th power of the $Z$-transform of (11). By collecting the coefficient in front of the term $z_1^{-k_1}z_2^{-k_2}$, we get for every $k_1, k_2 \in \mathbb{Z}$:

$$\Delta^n[k_1, k_2] = \sum_{i=\max(k_1, k_2, 0)}^{\min(n+k_1, n+k_2, n)} \Delta^n[i] \binom{n}{i-k_1} \binom{n}{i-k_2} \binom{n}{i}.$$  \hspace{1cm} (12)

By arranging the $\Delta^n[k]$ at the lattice sites $Rk = k_1 r_1 + k_2 r_2$, we can represent the first two localization filters as:

$$\Delta = \begin{bmatrix} 1 & -2 & 1 \\ -1 & 1 \\ -2 & 2 & 2 & -2 \end{bmatrix}, \quad \Delta^2 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & -6 & 2 & 1 \\ -2 & 2 & 2 & -2 \\ 1 & -2 & 1 \end{bmatrix}. \hspace{1cm} (13)$$

Putting together (11) and (8) with the fact that $\hat{\Delta}(\omega)\hat{\rho}(\omega) = 1$, we find that $\hat{\chi}(\omega) = \hat{\Delta}(\omega)\hat{\rho}(\omega)$. Therefore, we obtain the characterization:

$$\chi^n(x) = \Delta^n \ast \rho^n(x) = \sum_{k \in \mathbb{Z}^2} \Delta^n[k] \rho^n(x-Rk). \hspace{1cm} (14)$$

The complete analytical expression of $\chi^n(x)$, $n \geq 1$, can then be written as $\chi^n(x_1, x_2) = \sum_{k_1, k_2=-n}^{n} \Delta^n[k_1,k_2] \rho^n(x-k) \rho^n(x). \hspace{1cm} (15)$

This code was used to generate the plots in Fig. 2. The computational complexity is polynomial in $n$, compared to exponential for recursive methods in the literature [13]–[16]. For example, the evaluation boxspline($1, 1, 3$) took 0.002s, while 47s were required for the same operation using the Matlab code proposed in [15] (that can evaluate any box-spline, not just the three-directional ones).

### B. Further optimization for fixed $n$

For evaluating a box-spline $\chi^n$ of fixed $n$, an attractive hybrid analytical/numerical implementation consists in determining the polynomial form $p(x)$ inside each triangle of the three-directional mesh. This polynomial, which is obtained by the sums of (15), can be precomputed, stored, and only evaluated at the end. The following code in C-language for $\chi^2$ may serve as a template: coordinates are first folded in the sector $[0, \pi/2]$, then in $[0, \pi/3]$ and finally in $[0, \pi/6]$. This is done conveniently with the coordinates $(u, v)$. The coordinates $(g = u - v/2, v)$ in the orthogonal basis $(r_1, r_2 + r_3/2)$ are the most appropriate for having short polynomials with rational coefficients in each triangle.

```c
float boxspline2(float x, float y) {
    float u = fabs(x) - fabs(y) / sqrt(3.0);
    float v = fabs(x) + fabs(y) / sqrt(3.0);
    if (fabs(u) < (v/2.0) return 0.0; /* outside the support */
    if (v > 2.0) return 0.0; /* symmetry */
    if ( (v < 1.0) return 0.5 * ((5.0/3.0 - v/8.0) * v - 3.0) * v * v / 4.0 +
        (1.0 - v / 4.0) * v * v * v / 4.0); /* triangle 1 */
    if (v > 1.0) return (v - 1.0) * (v - 1.0) / 6.0;
    return 5.0/6.0 + 1.0/6.0 + (1.0/6.0 + v * v * v / 4.0) * v * v;
    /* triangle II */
}
```

### IV. IMPLEMENTATION ISSUES

#### A. The generic case

Equation (15) provides us with an efficient way to evaluate at any point $x$, any three-directional box-spline $\chi^n$. Notice that the power functions grow rapidly, as shown in Fig. 3, which could lead to problems of numerical stability. A simple remedy consists of evaluating $\chi^n$ only for $x_1 \leq 0$, which exploits the causality of $\rho^n$ in $x_1$ and the symmetry $\chi^n(x_1, x_2) = \chi^n(-x_1, x_2)$. The following Matlab code performs box-spline evaluations for a list of points $(x[m], y[m])$, indexed by $m$. The two-fold symmetry is used to fold coordinates into the sector $[5\pi/6, \pi]$, where the number of evaluations of the power functions is minimal. We use the coordinates $(u, v)$ in the basis $(r_1, r_2)$, instead of the coordinates $(x, y)$ in the canonical basis $(e_1, e_2)$. nchoosek(n, k) gives the binomial coefficient $(n, k)$.

```matlab
function val = boxspline(x, y, n)
    u = abs(x); v = abs(y);
    u = x - y / sqrt(3); v = x + y / sqrt(3);
    id = find(u > 0); v(id) = -v(id);
    id = find(u < 0); v(id) = -v(id);
    id = find(v > 2); v(id) = v(id) - 2;
    val = zeros(size(x));
    for k = n:-1:1,
        aux = abs(v - u(id));
        aux2 = (u(id) + v(id)) / 2;
        aux2(find(aux2 < 0)) = 0;
        val(id) = coeff * nchoosek(n, i-K) * nchoosek(n, l-K) * nchoosek(n, l);...
    end, end, end, end

end
```
We propose a new characterization of the three-directional box-splines, based on a Green function of the differential operator adapted to the hexagonal lattice. Together with a finite difference filter that acts as a localization operator on the generating function, this provides us with new explicit analytical formulas for the three-directional box-splines. This characterization also leads to particularly easy and efficient implementations. We provided the Matlab source code for the generic case and a further optimized C-code for the case \( n = 2 \). The latter one could be particularly interesting for high-quality visualization of data sampled on a hexagonal lattice.

Finally, we note that these box-splines can be expressed on any lattice with matrix \( R^3 \), and not only on the hexagonal one, by the simple change of basis \( \chi^\rho(RR^{-1}x) \).

**APPENDIX I**

**PROOF OF THE PROPOSITION**

We verify whether \( \rho^n \) of (9) is a Green function of \( L^n \); i.e., we need \( L^n \{ \rho^n \}(x) = \delta(x) \). First, we introduce the vectors

\[
\begin{align*}
\mathbf{r}_1^+ &= \begin{bmatrix} 1/\sqrt{3} \\ 0 \\ 0 \end{bmatrix}, & \mathbf{r}_2^+ &= \begin{bmatrix} -1/\sqrt{3} \\ 0 \\ 0 \end{bmatrix}, & \mathbf{r}_3^+ &= \begin{bmatrix} 0 \\ 0 \\ 2/\sqrt{3} \end{bmatrix},
\end{align*}
\]

which allow us to express the dual bases as \( (\mathbf{r}_1^+, \mathbf{r}_2^+, \mathbf{r}_3^+) \) and \( (-\mathbf{r}_1^+, \mathbf{r}_2^+, \mathbf{r}_3^+) \), respectively. For example, the coordinates of \( x \) in \( (r_2, r_3) \) are \((\mathbf{x}, \mathbf{r}_2^+), (\mathbf{x}, \mathbf{r}_3^+)\).

We now derive the Fourier expression of \( \mu^{n_1, n_2} \), which we first rewrite as

\[
\begin{align*}
\mu^{n_1, n_2}(x_1, x_2) &= (x_2)_+^0 \mu^{n_1, n_2}(x_1, x_2) + (-x_2)_+^0 \mu^{n_1, n_2}(x_1, x_2) \\
&= ((\mathbf{x}, \mathbf{r}_2^+))^n_1 ((\mathbf{x}, \mathbf{r}_2^+))^n_2 + ((\mathbf{x}, -\mathbf{r}_2^+))^n_1 ((\mathbf{x}, \mathbf{r}_2^+))^n_2.
\end{align*}
\]

From distribution theory, we know the Fourier transform of the one-sided power function

\[
(\mathbf{x})_+^n \xrightarrow{\mathcal{F}} \frac{n!}{|\omega|^{n+1}} + D(\omega),
\]

where \( D \) is essentially the \( n \)-th derivative of Dirac. This term can be omitted since it does not have any influence when applying a differential operator of order \( n \) (continuous or discrete) to \( (\mathbf{x})_+^n \), also see [2, Appendix C].

Hence, using a tensor product and a change of basis from \((\mathbf{e}_1, \mathbf{e}_2)\) to \((\mathbf{r}_2, \mathbf{r}_3)\) (with Jacobian \( \det[r_2, r_3] = \sqrt{3}/2 \)), we get

\[
(\mathbf{x})_+^n \xrightarrow{\mathcal{F}} \frac{\sqrt{3}/2}{(j(\omega, \mathbf{r}_2))^{n_1+1}(j(\omega, \mathbf{r}_3))^{n_2+1}}
\]

Similarly, the Fourier transform of \(-x_2)_+^0 \mu^{n_1, n_2} \) is obtained by replacing \( \mathbf{r}_2 \) by \( \mathbf{r}_1 \) in (18).

We now derive the functions \( \gamma^{n_1, n_2, n_3} \), for any integers \( n_1, n_2, n_3 \) as

\[
\gamma^{n_1, n_2, n_3} \xrightarrow{\mathcal{F}} \frac{\sqrt{3}/2}{(j(\omega, \mathbf{r}_1))^{n_1}(j(\omega, \mathbf{r}_2))^{n_2}(j(\omega, \mathbf{r}_3))^{n_3}}
\]

We recognize \( \rho^n = \gamma^{n,n,n} \), \( n \geq 1 \). Using the property \( r_3 = r_1 + r_2 \), we can further obtain the following recurrence relation, for \( n_1 \geq 1, n_2 \geq 1, n_3 \geq 0 \):

\[
\gamma^{n_1, n_2, n_3} = \gamma^{n_1-1, n_2, n_3+1} + \gamma^{n_1, n_2-1, n_3+1}.
\]

By recurrence on \( n_1 + n_2 \), we can also show that

\[
\gamma^{n_1, n_2, n_3} = \sum_{i=0}^{n_1-1} \left( n_2 - 1 + i \right) \gamma^{n_1-i, 0, n_2+n_3+i} + \sum_{i=0}^{n_2-1} \left( n_1 - 1 + i \right) \gamma^{0, n_2-i, n_1+n_3+i}.
\]

In the case of \( \rho^n \), we have

\[
\rho^{n} = \sum_{i=0}^{n-1} \left( n - 1 + i \right) \left( \gamma^{n-i, 0, 2n+i} + \gamma^{0, n-i, 2n+i} \right).
\]

Finally, we identify the function \( \mu^{n_1, n_2} \) as

\[
\mu^{n_1, n_2} = \gamma^{n_1+1, n_2+1} + \gamma^{n_1+1, 0, n_2+1},
\]

which results into (9).

**REFERENCES**


