Hex-splines:
A novel spline family for hexagonal lattices

Dimitri Van De Ville, Thierry Blu, Michael Unser,
Wilfried Philips, Ignace Lemahieu, Rik Van de Walle

D. Van De Ville, T. Blu, M. Unser are with the Biomedical Imaging Group (BIG), Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland. I. Lemahieu, R. Van de Walle are with ELIS, Ghent University, Ghent, Belgium. W. Philips is with TELIN, Ghent University, Ghent, Belgium.

Abstract

This paper proposes a new family of bivariate, non-separable splines, called hex-splines, especially designed for hexagonal lattices. The starting point of the construction is the indicator function of the Voronoi cell, which is used to define in a natural way the first-order hex-spline. Higher order hex-splines are obtained by successive convolutions. A mathematical analysis of this new bivariate spline family is presented. In particular, we derive a closed form for a hex-spline of arbitrary order. We also discuss important properties, such as their Fourier transform and the fact they form a Riesz basis. We also highlight the approximation order. For conventional rectangular lattices, hex-splines revert to classical separable tensor-product B-splines. Finally, some prototypical applications and experimental results demonstrate the usefulness of hex-splines for handling hexagonally sampled data.

Keywords

Bivariate splines, hexagonal lattices, approximation theory, sampling theory

I. Introduction

Digital image processing systems require a sampling strategy to represent two-dimensional data. The common approach is to take the samples on a rectangular lattice. Another possibility is to use hexagonal sampling, which provides several advantages [1, 2]. For instance, a hexagon has a twelve-fold symmetry as compared to the eight-fold of a square. Due to the improved packing density, hexagonal lattices are better suited for representing isotropically bandlimited signals. The higher degree of symmetry can also be used to design more isotropic filters to be applied on hexagonal lattices [2–4]. The six neighbors of a hexagonal cell and their connectivity is well-defined [5] and has been used for better edge detection [6, 7] and pattern recognition [8–11].

An important issue in image processing is the link between the discrete and the continuous domain. A discrete/continuous model is essential for computational tasks such as interpolation and resampling. B-spline models are especially popular and have been succesfully applied to image processing on rectangular lattices using tensor-product basis functions. This paper introduces a novel spline family, the so-called hex-splines, which provide a discrete/continuous model suitable for hexagonally sampled data. These new bivariate splines take into account the shape of the Voronoi cell and build up higher order splines by successive convolutions. For rectangular lattices, these splines revert to the classical tensor-product B-splines. The hex-splines were first introduced in our paper [12] from an application point of view (without an analytical expression). This paper presents a mathematical analysis of the hex-splines which contributes to a better understanding of their properties. Additionally, we discuss the algorithms that are needed to use these splines in practice.

The presentation is organized as follows. In Sect. II, we review the fundamental properties of lattices, cells, and one-dimensional B-splines. Next, we introduce and derive the hex-splines and discrete hex-splines. In Sect. V, we present some basic applications of the hex-splines to image processing.

There are some notational conventions we apply throughout this paper. Vectors are denoted in bold and lowercase, e.g., $\mathbf{x} = [x_1\ x_2]^{\mathrm{T}}$, while a matrix is bold and uppercase, e.g., $\mathbf{A}$. The Fourier transform

of a function $f(\mathbf{x})$ is defined as $\hat{f}(\boldsymbol{\omega}) = \int f(\mathbf{x}) \exp(-j\langle \boldsymbol{\omega}, \mathbf{x} \rangle) \mathrm{d}\mathbf{x}$. The complex conjugate is indicated by $f(\mathbf{x})^\star$. The inner product of two functions f_1 and f_2 is denoted as $\langle f_1, f_2 \rangle$ and the convolution as $f_1 * f_2$.

II. Splines for hexagonal lattices

A. Lattices

Two-dimensional periodic lattices are characterized by two (linearly independant) vectors $\mathbf{r}_1$ and $\mathbf{r}_2$. Any integer combination $\mathbf{r_k} = k_1\mathbf{r}_1 + k_2\mathbf{r}_2$ of these vectors points to a lattice site. It is often convenient to group the lattice vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ in a matrix

$$\mathbf{R} = [\mathbf{r}_1 \ \mathbf{r}_2], \tag{1}$$

such that the lattice sites are given by $\mathbf{Rk}$, where $\mathbf{k} = [k_1 \ k_2]^{\mathrm{T}}$.

A well-defined and unique tiling cell is the Voronoi cell, which contains all points that are closer to their lattice site than to any other site. The indicator function for the Voronoi cell of the origin is defined as:

$$\chi_{\mathbf{R}}(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \text{Voronoi cell}, \\ 1/m_{\mathbf{x}}, & \mathbf{x} \in \text{ edge of the Voronoi cell}, \\ 0, & \mathbf{x} \notin \text{Voronoi cell}, \end{cases} \tag{2}$$

where $m_{\mathbf{x}}$ is the number of lattice points to which $\mathbf{x}$ is adjacent. Note that $\chi_{\mathbf{R}}(\mathbf{x})$ tiles the plane by definition.

The dual or reciprocal lattice is specified by $\hat{\mathbf{R}} = [\hat{\mathbf{r}}_1 \ \hat{\mathbf{r}}_2] = \left(\mathbf{R}^{-1}\right)^{\mathrm{T}} = \mathbf{R}^{-\mathrm{T}}$. The reciprocal lattice vectors therefore satisfy the relation $\langle \mathbf{r}_k, \hat{\mathbf{r}}_l \rangle = \delta_{k-l}$, where δ_k is the kronecker-delta sequence. We mention two useful relationships where the dual lattice becomes important. Assume we have a function $f(\mathbf{x})$ and its Fourier transform $\hat{f}(\boldsymbol{\omega})$. Let $f_{\mathbf{M}}(\mathbf{x}) = f(\mathbf{Mx})$. Then it is well known that

$$\hat{f}_{\mathbf{M}}(\boldsymbol{\omega}) = \hat{f}(\mathbf{M}^{-\mathrm{T}}\boldsymbol{\omega}) / \left|\det \mathbf{M}\right|. \tag{3}$$

Likewise, the Poisson sum formula on a lattice $\mathbf{R}$ is (see App. A for the proof)

$$\sum_{\mathbf{k}} f(\mathbf{x} - \mathbf{Rk}) = \frac{1}{|\det(\mathbf{R})|} \sum_{\mathbf{k}} \hat{f}(2\pi\hat{\mathbf{R}}\mathbf{k}) \exp(j2\pi\langle \mathbf{R}^{\mathrm{T}}\mathbf{k}, \mathbf{x} \rangle). \tag{4}$$

The case of special interest in this article is the regular hexagonal lattice of the second type, which is characterized by matrices

$$\mathbf{R} = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 \\ -\frac{1}{2} & 1 \end{bmatrix}, \quad \hat{\mathbf{R}} = \begin{bmatrix} \frac{2\sqrt{3}}{3} & \frac{\sqrt{3}}{3} \\ 0 & 1 \end{bmatrix}. \tag{5}$$

Figure 1 shows the lattice vectors and the Voronoi cell of $\mathbf{R}$ and $\hat{\mathbf{R}}$. The reciprocal Voronoi cell can be considered as the "natural Nyquist region"; in other words, the effect of sampling a signal on a lattice $\mathbf{R}$ is to replicate its spectrum on the lattice sites $2\pi\hat{\mathbf{R}}\mathbf{k}$. For more details about lattices and cells, we refer to [13, 14].

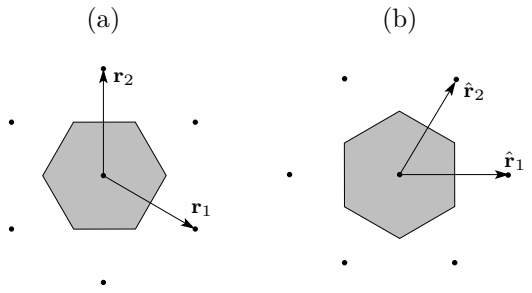

Fig. 1. (a) The regular hexagonal lattice of the second type and its Voronoi cell. (b) The dual lattice.

B. Sinc-function for hexagonal lattices

In this section, we extend the sampling theorem for hexagonal lattices. For a given lattice characterized by a matrix $\mathbf{R}$, we define the sincH-function of the corresponding reciprocal lattice $\hat{\mathbf{R}}$ as the Fourier transform of the indicator function of the Voronoi cell:

$$\hat{\chi}_{\mathbf{R}}(\boldsymbol{\omega}) = \Omega \ \mathrm{sincH}_{\hat{\mathbf{R}}}(\boldsymbol{\omega}/(2\pi)), \tag{6}$$

where we introduce the normalization by the surface area of the Voronoi cell $\Omega \triangleq |\det(\mathbf{R})|$. By definition, the function $\chi_{\mathbf{R}}(\mathbf{x})$ tiles the plane when replicated on the lattice sites of $\mathbf{R}$. So we have

$$\sum_{\mathbf{k}} \chi_{\mathbf{R}}(\mathbf{x} - \mathbf{R}\mathbf{k}) = 1, \quad \forall \mathbf{x}. \tag{7}$$

When we plug in this condition into the Poisson sum formula of Eq. (4), which holds for all $\mathbf{x}$, we obtain

$$\mathrm{sincH}_{\hat{\mathbf{R}}}(\hat{\mathbf{R}}\mathbf{k}) = \delta_{\mathbf{k}}, \tag{8}$$

where we have used $\hat{\chi}_{\mathbf{R}}(\boldsymbol{\omega})$ as defined in Eq. (6). This function takes the value 1 at the origin and 0 at the reciprocal lattice sites, as we expect from a properly defined sinc-function. By duality, it follows that the sincH-function corresponding to the lattice matrix $\mathbf{R}$ satisfies the interpolation property in the spatial domain. An explicit formula for the sincH-function is derived in Sect. IV-A. This function is shown in Fig. 2.

C. Hex-splines

We define the first-order hex-spline as $\eta_1(\mathbf{x}) = \chi_{\mathbf{R}}(\mathbf{x})$. Hex-splines of higher orders are constructed by successive convolutions:

$$\eta_{p+1}(\mathbf{x}) = \frac{\eta_1 * \eta_p(\mathbf{x})}{\Omega}, \quad p \geq 1. \tag{9}$$

Some examples of hex-splines are shown in Fig. 3. An important property, which is satisfied automatically due to Eq. (7) and the construction rule of Eq. (9), is the partition of unity:

$$\sum_{\mathbf{k}} \eta_p(\mathbf{x} - \mathbf{R}\mathbf{k}) = 1, \quad \forall \mathbf{x}, \tag{10}$$

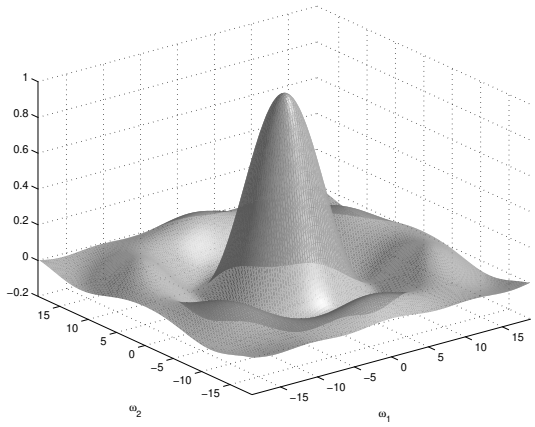

Fig. 2. The function $\mathrm{sincH}_{\hat{\mathbf{R}}}(\boldsymbol{\omega}/(2\pi))$ corresponding to the regular hexagonal lattice.

which holds for any order p.

The hex-spline basis functions are obtained by shifting η_p to each lattice site $\mathbf{Rk}$. The corresponding signal space $S(\eta_p)$ is

$$S(\eta_p) = \left\{ s(\mathbf{x}) \;\middle|\; s(\mathbf{x}) = \sum_{\mathbf{k}} c(\mathbf{k}) \eta_p(\mathbf{x} - \mathbf{Rk});\ c(\mathbf{k}) \in l_2(\mathbb{Z}^2) \right\}. \tag{11}$$

This means that each signal is characterized by its coefficients $c(\mathbf{k})$ (discrete/continuous representation), which are square-summable on the lattice $\mathbb{Z}^2$. A common way to determine the spline coefficients $c(\mathbf{k})$ is to impose the interpolating condition, which requires that $s(\mathbf{Rk}) = g(\mathbf{Rk})$ at the sampling sites. Here, $g(\mathbf{x})$ is the original signal we want to represent using the splines. For the first and second order splines, this condition is trivially satisfied by choosing $c(\mathbf{k}) = g(\mathbf{Rk})$; higher orders require an inverse filtering operation to obtain the correct values for $c(\mathbf{k})$. This operation is often called the direct spline transform [15].

If we apply the hex-splines construction to rectangular lattices, i.e., $\mathbf{r}_1 = [1\ 0]^{\mathrm{T}}$ and $\mathbf{r}_2 = [0\ 1]^{\mathrm{T}}$, we obtain the square indicator function as the first-order hex-spline. Consequently, the hex-splines revert to separable tensor-product B-splines. In particular for $\mathbf{R} = \mathbf{I}$, we have

$$\eta_p(\mathbf{x}) = \beta^{p-1}(x_1)\beta^{p-1}(x_2), \tag{12}$$

with the notation of [15] where the B-splines are indexed by their degree $n = p - 1$. Here instead, we will index the functions by their order $p = n + 1$ and stick to our subscript notation.

III. Explicit construction

In this section, we present a construction algorithm for hex-splines of any order, which allows us to determine their closed form. Although the derivation is general and can be applied to any (irregular) hexagonal Voronoi cell of a periodic lattice, our running example will be the regular hexagon introduced in Sect. II-A.

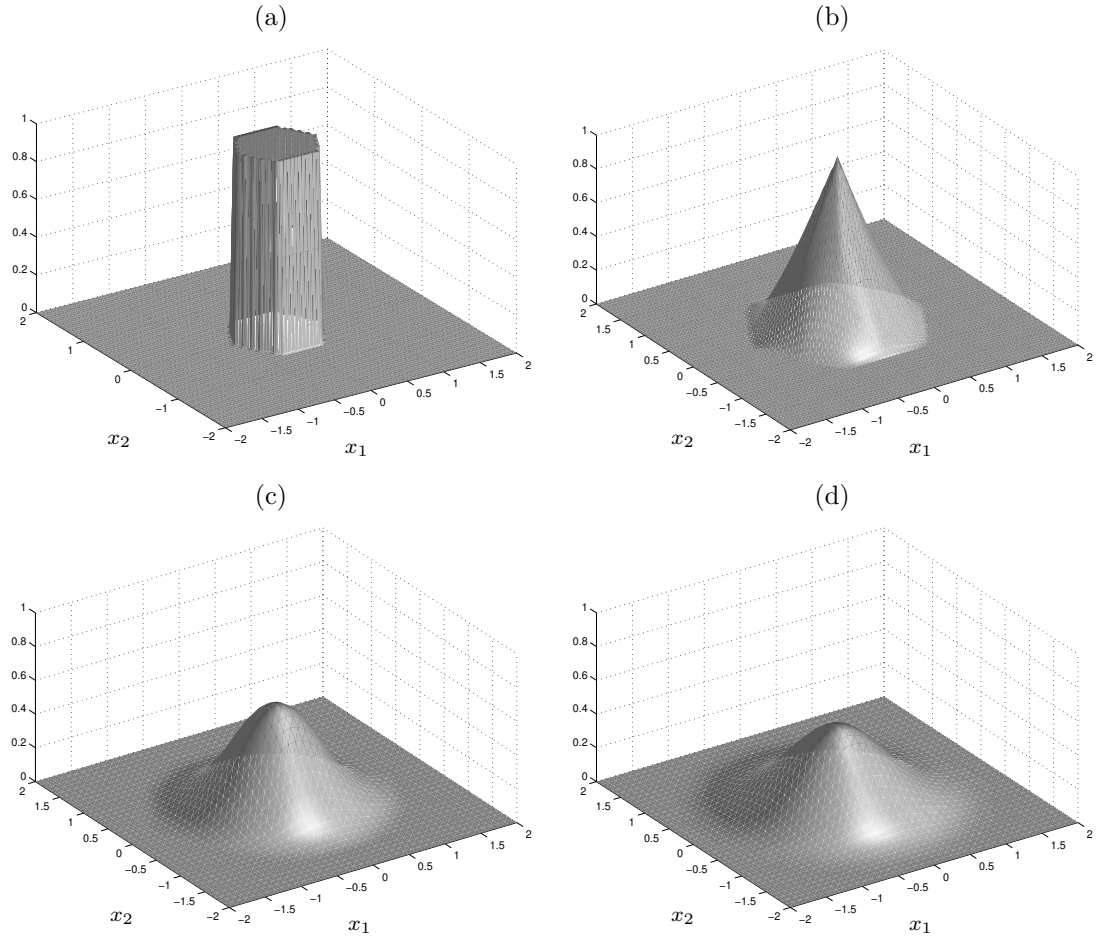

Fig. 3. Hex-splines. (a) First order ($p = 1$). (b) Second order ($p = 2$). (c) Third order ($p = 3$). (d) Fourth order ($p = 4$).

A. B-spline refresher

Our construction is inspired by the properties of one-dimensional B-splines. It is well-known that B-splines can be generated by using a suitable linear combination of shifted one-sided power functions. For instance, the first-order B-spline can be expressed as

$$\beta^0(x) = \Delta * (x)_+^0 \tag{13}$$

$$= (x + 1/2)_+^0 - (x - 1/2)_+^0, \tag{14}$$

where Δ denotes the central finite difference filter $\delta(x + 1/2) - \delta(x - 1/2)$ and where $(x)_+^0$ is the unit step function. The application of the finite difference to the step function, which is infinitely supported, produces a compactly supported function. Figure 4 graphically illustrates the generation of β^0. We now examine the convolutional properties of the components Δ and $(x)_+^0$. The n-fold convolution of the localization operator results into the n-th order finite difference

$$\Delta * \Delta^{n-1} = \Delta^n \longleftrightarrow (\exp(j\omega/2) - \exp(-j\omega/2))^n, \tag{15}$$

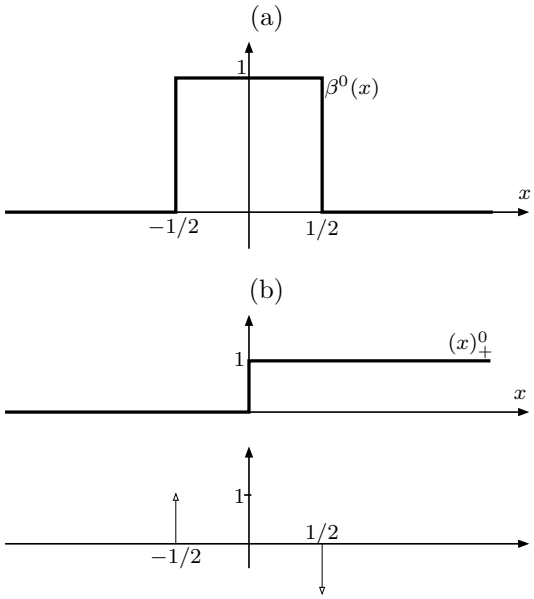

Fig. 4. (a) The one-dimensional first-order B-spline $\beta^0(x)$. (b) Construction of $\beta^0(x)$ by the step function $(x)_+^0$ and the finite difference Δ.

which is conveniently specified by its Fourier expression. Analogously, successive convolutions of the step function yield the one-sided power functions:

$$\frac{(x)_+^0}{0!} * \frac{(x)_+^{n-1}}{(n-1)!} = \frac{(x)_+^n}{n!} = \frac{\max(0,x)^n}{n!}, \tag{16}$$

which are the generating functions of higher order splines.

Relations (15) and (16) are especially useful for deriving the following explicit formula, starting from the convolutional definition of B-splines:

$$\begin{aligned} \beta^n &= \underbrace{\beta^0 * \ldots * \beta^0}_{n+1 \text{ times}} \\ &= \left(\Delta * (x)_+^0\right) * \ldots * \left(\Delta * (x)_+^0\right) \\ &= \Delta^{n+1} * \frac{(x)_+^n}{n!}. \end{aligned}$$

The corresponding two-dimensional B-splines on a rectangular lattice are obtained by simple application of the tensor-product: $\beta^n(\mathbf{x}) = \beta^n(x_1)\beta^n(x_2)$. Due to the separability, the generating function becomes $(\mathbf{x})_+^n = (x_1)_+^n(x_2)_+^n$ and the localization operator $\Delta_{x_1} * \Delta_{x_2}$. So for $n = 0$, a generating function (which fills the upper right quadrant of the plane) is placed on each corner of a square with an appropriate weight.

B. Construction of the first-order hex-spline

Now, we aim at a similar construction for hex-splines. Let us start by putting together the first-order hex-spline η_1 in a graphical way. We want the generating function to resemble $(\mathbf{x})_+^0$ but we also need

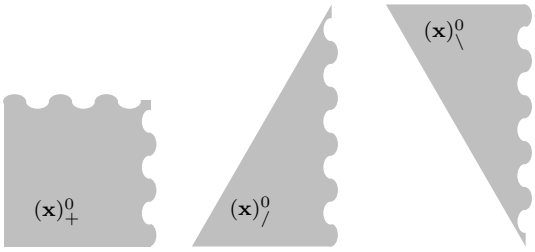

Fig. 5. Left: The generating function $(\mathbf{x})^0_+$ for the first-order two-dimensional tensor-product spline. Right: The generating functions $(\mathbf{x})^0_{/}$ and $(\mathbf{x})^0_{\backslash}$ for the first-order hex-spline. The functions are unity inside the light gray region. The curved border means the function extends infinitely in that direction.

to consider that the edges of the hexagon have three different orientations (i.e., $\theta = 0$, $\pi/3$, and $-\pi/3$). Therefore, we introduce the two generating functions $(\mathbf{x})^0_{/}$ and $(\mathbf{x})^0_{\backslash}$, shown in Fig. 5. The next step is to localize these functions in order to obtain the (compactly supported) indicator function of the hexagon, as shown in Fig. 6. First, in (a), we place both generating functions on the outer left vertex of the hexagon. Second, in (b), we create the horizontal edges by subtracting $(\mathbf{x})^0_{/}$ at the upper left vertex and $(\mathbf{x})^0_{\backslash}$ at the lower left vertex. Third, in (c), we form the right-hand edges by subtracting this time $(\mathbf{x})^0_{\backslash}$ at the upper right vertex and $(\mathbf{x})^0_{/}$ at the lower right vertex. Finally, in (d), we need to compensate for the dark region of (c) that has become -1, by putting both generating function at the outer right vertex.

Theorem 1: The first-order hex-spline is obtained by localizing the generating functions $(\mathbf{x})^0_{/}$ and $(\mathbf{x})^0_{\backslash}$, as shown by the construction above. As such, we obtain an equivalent form of Eq. (13) for the hex-splines

$$\eta_1(\mathbf{x}) = \Delta_{/} * (\mathbf{x})^0_{/} + \Delta_{\backslash} * (\mathbf{x})^0_{\backslash}, \tag{17}$$

where each generating function has been placed on four vertices through the application of localization operators $\Delta_{/}$ and $\Delta_{\backslash}$. □

C. Construction of higher order hex-splines

The higher order hex-splines are constructed by successive convolutions. As for the classical B-splines, we need formulas for the multiple convolutions between the localization operators on one side, and the generating functions on the other side. First, we introduce the vectors indicating the vertices of the hexagon as shown in Fig. 7:

$$\mathbf{e}_1 = \begin{bmatrix} -\frac{\sqrt{3}}{3} \\ 0 \end{bmatrix}, \ \mathbf{e}_2 = \begin{bmatrix} -\frac{\sqrt{3}}{6} \\ \frac{1}{2} \end{bmatrix}, \ \mathbf{e}_3 = \begin{bmatrix} -\frac{\sqrt{3}}{6} \\ -\frac{1}{2} \end{bmatrix}.$$

These allow us to derive the following properties for the localization operator.

Proposition 1: The Fourier transforms of the n-fold convolution of the operators $\Delta_{/}$ and $\Delta_{\backslash}$ are given

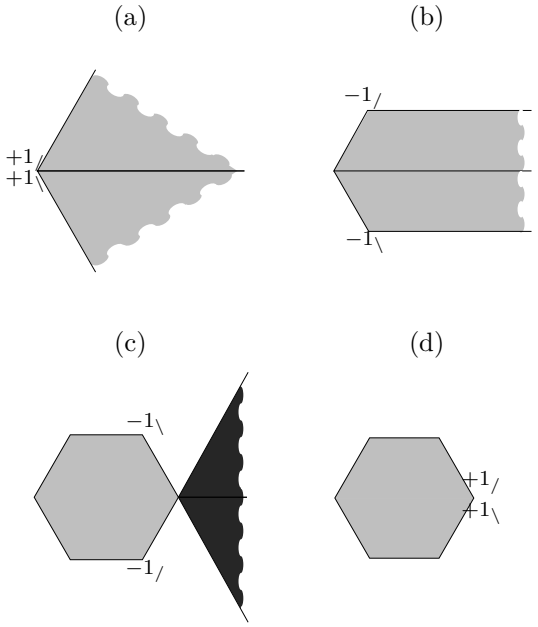

Fig. 6. The construction of the first-order hex-spline step-by-step. The light gray and dark gray regions correspond respectively to +1 and −1.

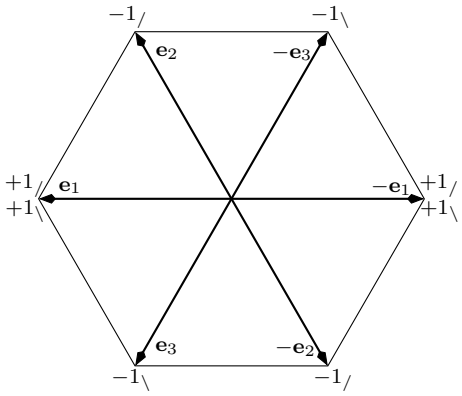

Fig. 7. The vectors $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ are introduced to indicate the vertices of the hexagon. In order to build up the first-order hex-spline, the two generating functions need to be placed at four vertices each, with the weights indicated.

by

$$\begin{aligned}\Delta_{/}^{n} &\longleftrightarrow \left(\hat{\Delta}(\langle \mathbf{e}_1+\mathbf{e}_2,\boldsymbol{\omega}\rangle)\hat{\Delta}(\langle \mathbf{e}_1-\mathbf{e}_2,\boldsymbol{\omega}\rangle)\right)^n\\ \Delta_{\backslash}^{n} &\longleftrightarrow \left(\hat{\Delta}(\langle \mathbf{e}_1+\mathbf{e}_3,\boldsymbol{\omega}\rangle)\hat{\Delta}(\langle \mathbf{e}_1-\mathbf{e}_3,\boldsymbol{\omega}\rangle)\right)^n\end{aligned}$$

where $\hat{\Delta}(\omega) = \exp(j\omega/2) - \exp(-j\omega/2)$ is the classical one-dimensional B-spline localization operator. □

For the generating functions, we first introduce a way to describe them more precisely. For instance, consider the generating function $(\mathbf{x})^0_{/}$. If the vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ are along the border of the support, we can define this function as

$$(\mathbf{x})^0_{/} = \begin{cases} 1, & \text{when } \langle \hat{\mathbf{u}}_1, \mathbf{x} \rangle > 0 \text{ and } \langle \hat{\mathbf{u}}_2, \mathbf{x} \rangle > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

The vectors $\hat{\mathbf{u}}_1$ and $\hat{\mathbf{u}}_2$ are the reciprocal ones of $\mathbf{u}_1$ and $\mathbf{u}_2$ (i.e., $\langle \mathbf{u}_k, \hat{\mathbf{u}}_l \rangle = \delta_{k,l}$), see Fig. 8. Now, a more general generating function is proposed which builds up higher polynomial degrees along the reciprocal directions:

$$(\mathbf{x})^{(n_1,n_2)}_{(\mathbf{u}_1,\mathbf{u}_2)} = \frac{(\langle \hat{\mathbf{u}}_1, \mathbf{x} \rangle)^{n_1}_+}{n_1!} \frac{(\langle \hat{\mathbf{u}}_2, \mathbf{x} \rangle)^{n_2}_+}{n_2!} \tag{19}$$

We also use the shorthand notation $(\mathbf{x})^n_{(\mathbf{u}_1,\mathbf{u}_2)}$ for $(\mathbf{x})^{(n,n)}_{(\mathbf{u}_1,\mathbf{u}_2)}$. Note that this generating function can be derived from the orthogonal case by the coordinate transformation given by the matrix $\mathbf{M} = [\hat{\mathbf{u}}_1 \ \hat{\mathbf{u}}_2]^{\mathrm{T}}$. For convenience, we have included the normalization by $n_1!n_2!$ inside the definition of the generating function. Next we choose $\mathbf{u}_1 = [1 \ 0]^{\mathrm{T}}$ and $\mathbf{u}_2$ such that $|\det(\mathbf{M})| = 1$; i.e., the vertical component of $\mathbf{u}_2$ needs to be ± 1 due to initial choice of $\mathbf{u}_1$. That way, we can describe $(\mathbf{x})^0_{/}$ and $(\mathbf{x})^0_{\backslash}$ as

$$\begin{aligned} (\mathbf{x})^0_{/} &= (\mathbf{x})^0_{(\mathbf{u}_1,\mathbf{u}_2)}, \\ (\mathbf{x})^0_{\backslash} &= (\mathbf{x})^0_{(\mathbf{u}_1,\mathbf{u}_3)}, \end{aligned}$$

where

$$\mathbf{u}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \mathbf{u}_2 = \begin{bmatrix} \frac{\sqrt{3}}{3} \\ 1 \end{bmatrix}, \ \mathbf{u}_3 = \begin{bmatrix} \frac{\sqrt{3}}{3} \\ -1 \end{bmatrix}.$$

These vectors may also be expressed in terms of $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$:

$$\mathbf{u}_1 = -\frac{\mathbf{e}_1}{\|\mathbf{e}_1\|}, \ \mathbf{u}_2 = \frac{\mathbf{e}_2 - \mathbf{e}_1}{\left\langle \mathbf{e}_2 - \mathbf{e}_1, [0 \ 1]^{\mathrm{T}} \right\rangle}, \ \mathbf{u}_3 = \frac{\mathbf{e}_3 - \mathbf{e}_1}{\left\langle \mathbf{e}_3 - \mathbf{e}_1, [0 \ -1]^{\mathrm{T}} \right\rangle}. \tag{20}$$

Next, we give the convolution rules for these generating functions (cf. proof in Appendix B).

Proposition 2: The generating functions, as defined before, satisfy the following recurrence equation:

$$(\mathbf{x})^0_{(\mathbf{u}_1,\mathbf{u}_2)} * (\mathbf{x})^{n-1}_{(\mathbf{u}_1,\mathbf{u}_2)} = (\mathbf{x})^n_{(\mathbf{u}_1,\mathbf{u}_2)}. \tag{21}$$

Crossterms, such as

$$(\mathbf{x})^{n_1}_{(\mathbf{u}_1,\mathbf{u}_2)} * (\mathbf{x})^{n_2}_{(\mathbf{u}_1,\mathbf{u}_3)}, \tag{22}$$

are denoted as $(\mathbf{x})^{(n_1+n_2+1,n_1,n_2)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)}$. Although this notation has no direct (spatial) interpretation in the sense of Eq. (19), it can be studied in the Fourier domain as we will show in Sect. IV-A. For now, we only mention how they can be computed in a recursive way:

$$(\mathbf{x})^{(n_1,n_2,n_3)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)} = \frac{1}{2\langle \mathbf{u}_1, \mathbf{u}_2 \rangle} \Big((\mathbf{x})^{(n_1+1,n_2-1,n_3)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)} + (\mathbf{x})^{(n_1+1,n_2,n_3-1)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)} \Big), \quad n_1, n_2, n_3 \geq 0. \tag{23}$$

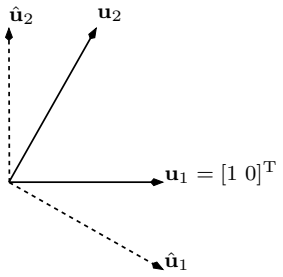

Fig. 8. The two-dimensional generating function. The vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ make up the borders of the support. The polynomial degree increases along the reciprocal vectors $\hat{\mathbf{u}}_1$ and $\hat{\mathbf{u}}_2$ for higher order generating functions.

This can be applied recursively until the second or third power of a generating function equals -1, in particular $(\mathbf{x})^{(n_1,-1,n_3)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)} = (\mathbf{x})^{(n_1,n_3)}_{(\mathbf{u}_1,\mathbf{u}_3)}$ and $(\mathbf{x})^{(n_1,n_2,-1)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)} = (\mathbf{x})^{(n_1,n_2)}_{(\mathbf{u}_1,\mathbf{u}_2)}$. □

Notice that the generating functions $(\mathbf{x})^{(n_1,n_2)}_{(\mathbf{u}_1,\mathbf{u}_2)}$ and $(\mathbf{x})^{(n_1,n_2)}_{(\mathbf{u}_1,\mathbf{u}_3)}$ are homogeneous two-dimensional polynomials of degree n_1+n_2; i.e., they satisfy the equation $f(\lambda\mathbf{x}) = \lambda^{n_1+n_2} f(\mathbf{x})$. Consequently, the crossterms are homogeneous polynomials as well. For example,

$$(\mathbf{x})^{0}_{(\mathbf{u}_1,\mathbf{u}_2)} * (\mathbf{x})^{0}_{(\mathbf{u}_1,\mathbf{u}_3)} = (\mathbf{x})^{(1,0,0)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)} = \frac{\sqrt{3}}{2}\left((\mathbf{x})^{(2,0)}_{(\mathbf{u}_1,\mathbf{u}_3)} + (\mathbf{x})^{(2,0)}_{(\mathbf{u}_1,\mathbf{u}_2)}\right) \tag{24}$$

is a homogeneous polynomial of degree 2.

Theorem 2: The higher-order hex-spline is obtained by applying the construction rule of Eq. (9) to Eq. (17):

$$\begin{aligned}\eta_p(\mathbf{x}) &= \frac{1}{\Omega^{p-1}}\sum_{k=0}^{p}\binom{p}{k}\boldsymbol{\Delta}^{k}_{/} * \boldsymbol{\Delta}^{p-k}_{\backslash} * (\mathbf{x})^{k-1}_{(\mathbf{u}_1,\mathbf{u}_2)} * (\mathbf{x})^{p-1-k}_{(\mathbf{u}_1,\mathbf{u}_3)} \\ &= \frac{1}{\Omega^{p-1}}\sum_{k=0}^{p}\binom{p}{k}\boldsymbol{\Delta}^{k}_{/} * \boldsymbol{\Delta}^{p-k}_{\backslash} * (\mathbf{x})^{(p-1,k-1,p-1-k)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)},\end{aligned}$$

for $p \geq 1$, where generating functions to the power -1 "neutralize" the convolution, i.e., $(\mathbf{x})^{-1}_{(\mathbf{u}_1,\mathbf{u}_2)} = (\mathbf{x})^{-1}_{(\mathbf{u}_1,\mathbf{u}_3)} = \delta(\mathbf{x})$. □

To facilitate the computation of these formulas for any order p, we have made available on the web an implementation for the Maple mathematical software package (see Appendix D).

IV. Hex-spline properties

A. Fourier transform

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

$$\frac{(x)^n_+}{n!} \quad \longleftrightarrow \quad \frac{1}{(j\omega)^{n+1}} + \frac{\pi j^n \delta^{(n)}(\omega)}{n!}, \tag{25}$$

where $\delta^{(n)}$ is the n-th derivative of the Dirac δ-function. There is also a "two-sided" generating function:

$$\frac{(x)^n_{\mathrm{sign}}}{n!} = \frac{x^n \mathrm{sign}(x)}{2\ n!} = \frac{(x)^n_+ + (-1)^{n+1}(-x)^n_+}{2\ n!} \quad \longleftrightarrow \quad \frac{1}{(j\omega)^{n+1}}. \tag{26}$$

Both generating functions are equivalent with respect to the localization operators (finite difference), i.e., the localization operator cancels the polynomial represented by the Dirac of the one-sided function, so leaving the two-sided version. This mechanism is well-known in 1D and it is often used to allow simpler Fourier domain manipulations.

For the two-dimensional extension by the tensor-product we obtain

$$\frac{(x_1)_{\text{sign}}^{n_1}}{n_1!}\frac{(x_2)_{\text{sign}}^{n_2}}{n_2!} \longleftrightarrow \frac{1}{(j\omega_1)^{n_1+1}(j\omega_2)^{n_2+1}}. \tag{27}$$

The sign-version of $(\mathbf{x})_{(\mathbf{u}_1,\mathbf{u}_2)}^{(n_1,n_2)}$ can be defined by considering

$$|\mathbf{x}|_{(\mathbf{u}_1,\mathbf{u}_2)}^{(n_1,n_2)} = \frac{(\langle \mathbf{x}, \hat{\mathbf{u}}_1\rangle)_{\text{sign}}^{n_1}}{n_1!}\frac{(\langle \mathbf{x}, \hat{\mathbf{u}}_2\rangle)_{\text{sign}}^{n_2}}{n_2!}, \tag{28}$$

which corresponds to a coordinate transformation of the left-hand side of Eq. (27) by the matrix

$$\mathbf{M} = [\hat{\mathbf{u}}_1\ \hat{\mathbf{u}}_2]^{\mathrm{T}} = [\mathbf{u}_1\ \mathbf{u}_2]^{-1} \tag{29}$$

Therefore, the Fourier transform of such a generating function is given by

$$|\mathbf{x}|_{(\mathbf{u}_1,\mathbf{u}_2)}^{(n_1,n_2)} \longleftrightarrow \frac{1}{(j\langle \mathbf{u}_1,\boldsymbol{\omega}\rangle)^{n_1+1}(j\langle \mathbf{u}_2,\boldsymbol{\omega}\rangle)^{n_2+1}}, \tag{30}$$

since $|\det(\mathbf{M})| = 1$ due to the proper choice of $\mathbf{u}_1$ and $\mathbf{u}_2$. We refer to appendix C for a precise explanation of how the sign-version of the generating function gets localized in two dimensions.

The localization process is a convolution which corresponds to a product in the Fourier domain. Putting the pieces together, we get the Fourier transform of the first-order regular hex-spline

$$\begin{aligned}
\hat{\eta}_1(\boldsymbol{\omega}) &= \hat{\Delta}_{/}\widehat{(\mathbf{x})_{/}^{0}} + \hat{\Delta}_{\backslash}\widehat{(\mathbf{x})_{\backslash}^{0}} \\
&= \frac{2\sqrt{3}}{\omega_1}\left(\frac{\cos(-\omega_1/(2\sqrt{3}) + \omega_2/2) - \cos(\omega_1/\sqrt{3})}{\omega_1 + \sqrt{3}\omega_2} + \right. \\
&\quad \left. \frac{\cos(\omega_1/(2\sqrt{3}) + \omega_2/2) - \cos(\omega_1/\sqrt{3})}{\omega_1 - \sqrt{3}\omega_2}\right) \\
&= \Omega\ \mathrm{sincH}_{\hat{\mathbf{R}}}(\boldsymbol{\omega}/(2\pi)),
\end{aligned} \tag{31}$$

where we have used

$$\begin{aligned}
\hat{\Delta}_{/} &= 2\cos(\langle \mathbf{e}_1,\boldsymbol{\omega}\rangle) - 2\cos(\langle \mathbf{e}_2,\boldsymbol{\omega}\rangle) = 2\cos(\omega_1/\sqrt{3}) - 2\cos(-\omega_1/(2\sqrt{3}) + \omega_2/2), \\
\widehat{(\mathbf{x})_{/}^{0}} &= -\frac{1}{\langle \mathbf{u}_1,\boldsymbol{\omega}\rangle\langle \mathbf{u}_2,\boldsymbol{\omega}\rangle} = -\frac{1}{\omega_1(\omega_1/\sqrt{3} + \omega_2)}.
\end{aligned}$$

Equation (31) also provides us with an explicit expression for the sincH-function introduced in Sect. II-B.

Due to the simple recipe of Eq. (9), the Fourier transform for a hex-spline of order p can be derived from $\hat{\eta}_1(\boldsymbol{\omega})$ as

$$\hat{\eta}_p(\boldsymbol{\omega}) = \frac{(\hat{\eta}_1(\boldsymbol{\omega}))^p}{\Omega^{p-1}} = \Omega\ \mathrm{sincH}_{\hat{\mathbf{R}}}(\boldsymbol{\omega}/(2\pi))^p. \tag{32}$$

B. Riesz basis property

An important condition for the hex-splines to provide a sensible continuous/discrete model is to be stable (i.e., a small variation of the coefficients results into a small variation of the function) and unambiguous (i.e., each set of coefficients represents a unique function). Therefore, the basis functions should form a Riesz basis, which requires the existence of two strictly positive constants $0 < A_0$ and $A_1 < +\infty$ such that

$$A_0 \left|\left|c\right|\right|^2 \leq \left|\left|\sum_{\mathbf{k}} c(\mathbf{k})\eta_p(\mathbf{x}-\mathbf{R}\mathbf{k})\right|\right|^2 \leq A_1 \left|\left|c\right|\right|^2, \tag{33}$$

using the Euclidean norm. This expression is equivalent to

$$\Omega A_0 \leq \sum_{\mathbf{k}} \left|\hat{\eta}_p(\boldsymbol{\omega}+2\pi\hat{\mathbf{R}}\mathbf{k})\right|^2 \leq \Omega A_1, \tag{34}$$

where the central term is the Fourier transform of the sampled autocorrelation function

$$a_{\eta_p}(\mathbf{k}) = \langle \eta_p(\mathbf{x}-\mathbf{R}\mathbf{k}), \eta_p(\mathbf{x})\rangle = \Omega\ \eta_{2p}(\mathbf{R}\mathbf{k}), \tag{35}$$

and can be rewritten as $\hat{a}_{\eta_p}(\boldsymbol{\omega}) = \sum_{\mathbf{k}} a_{\eta_p}(\mathbf{k}) \exp(-j\langle\boldsymbol{\omega}, \mathbf{R}\mathbf{k}\rangle)$.

Next, we explicitly demonstrate that such constants A_0, A_1 can be found for the case of a regular hexagon. The upper bound is obtained from

$$\begin{aligned}
\hat{a}_{\eta_p}(\boldsymbol{\omega}) &= \sum_{\mathbf{k}} \eta_p * \eta_p(\mathbf{R}\mathbf{k}) \exp(-j\langle\boldsymbol{\omega}, \mathbf{R}\mathbf{k}\rangle) \\
&\leq \sum_{\mathbf{k}} |\eta_p * \eta_p(\mathbf{R}\mathbf{k})| \\
&\leq \Omega \sum_{\mathbf{k}} |\eta_{2p}(\mathbf{R}\mathbf{k})| \\
&\leq \Omega \sum_{\mathbf{k}} \eta_{2p}(\mathbf{R}\mathbf{k}) = \Omega = \frac{\sqrt{3}}{2} = \Omega A_1,
\end{aligned}$$

where we have used the positivity of the hex-splines, the partition of unity of Eq. (10) and Eq. (9).

The derivation of the lower bound is slightly more involved. First, since $\hat{a}_{\eta_p}(\boldsymbol{\omega})$ is periodic on $2\pi\hat{\mathbf{R}}$, we can concentrate our attention on the reciprocal tiling cell (i.e., the Nyquist region, characterized by

$\chi_{\mathbf{2}\pi\hat{\mathbf{R}}}(\boldsymbol{\omega}) = 1$). As such, we obtain

$$\begin{aligned}
\hat{a}_{\eta_p}(\boldsymbol{\omega}) &= \sum_{\mathbf{k}} \left|\hat{\eta}_p(\boldsymbol{\omega} + 2\pi\hat{\mathbf{R}}\mathbf{k})\right|^2 \\
&= \sum_{\mathbf{k}} \left|\hat{\eta}_1(\boldsymbol{\omega} + 2\pi\hat{\mathbf{R}}\mathbf{k})\right|^{2p} / \Omega^{2(p-1)} \\
&\geq |\hat{\eta}_1(\boldsymbol{\omega})|^{2p} / \Omega^{2(p-1)} \\
&= \Omega^2 \left|\mathrm{sincH}_{\hat{\mathbf{R}}}(\boldsymbol{\omega}/(2\pi))\right|^{2p} \\
&\geq \Omega^2 \left| \inf_{\chi_{\mathbf{2}\pi\hat{\mathbf{R}}}(\boldsymbol{\omega})=1} \mathrm{sincH}_{\hat{\mathbf{R}}}(\boldsymbol{\omega}/(2\pi)) \right|^{2p} \\
&= \frac{3}{4}\left(\frac{9+2\sqrt{3}\pi}{4\pi^2}\right)^{2p} = \Omega A_0.
\end{aligned}$$

Indeed, the sincH-function decreases monotonically from the origin to the border of the Nyquist region (see also Fig. 2). At the outer vertices it reaches its minimum, e.g., at $\boldsymbol{\omega} = [0\ 4\pi/3]^{\mathrm{T}}$, which yields A_0.

C. Relation to other spline families

To the best knowledge of the authors, the hex-splines have not been proposed before. Nevertheless, there is a connection with bivariate box-splines [16].

Two-dimensional box-splines are a family of bivariate splines. They are composed out of basic elements that can be regarded as a causal B-spline along a vector; i.e., in Fourier, a basic element along a vector $\mathbf{u}$ can be expressed as

$$\hat{\mathcal{M}}_{[\mathbf{u}]}(\boldsymbol{\omega}) = \frac{1 - \exp(j\langle\boldsymbol{\omega}, \mathbf{u}\rangle)}{j\langle\boldsymbol{\omega}, \mathbf{u}\rangle}. \tag{36}$$

In the spatial domain, such an element corresponds to

$$\mathcal{M}_{[\mathbf{u}]}(\mathbf{x}) = \delta(\langle\mathbf{u}_\perp, \mathbf{x}\rangle)\beta^0\left(\langle\mathbf{u}, \mathbf{x}\rangle - 1/2\right),$$

for normalized $\mathbf{u}$ and $\mathbf{u}_\perp$ perpendicular to $\mathbf{u}$. For example, for $\mathbf{u} = [1\ 0]^{\mathrm{T}}$ we obtain $\delta(x_2)\beta^0(x_1 - 1/2)$. A general box-spline can be obtained by performing convolutions of these basic elements (so multiplications of Eq. (36) in the Fourier domain). For instance, the box-spline $\mathcal{M}_{[\mathbf{e}_1\ \mathbf{e}_2]}$ (where $\mathbf{e}_1$ and $\mathbf{e}_2$ are linearly independant) corresponds to the indicator function of the rhomboid spanned by $\mathbf{e}_1$ and $\mathbf{e}_2$, scaled by the reciprocal of its surface area. Adding any direction by introducing a third vector, creates a "slope" (i.e., linearly increasing/decreasing) in that particular direction. It is therefore not possible to construct the first-order hex-spline in this way. Nevertheless, the hex-spline can be constructed as the sum of three box-splines using vectors along the borders of the hexagon:

$$\eta_1(\mathbf{x}) = |\det[\mathbf{v}_1\ \mathbf{v}_2]|\,\mathcal{M}_{[\mathbf{v}_1\ \mathbf{v}_2]}(\mathbf{x}) + |\det[\mathbf{v}_2\ \mathbf{v}_3]|\,\mathcal{M}_{[\mathbf{v}_2\ \mathbf{v}_3]}(\mathbf{x}) + |\det[\mathbf{v}_1\ \mathbf{v}_3]|\,\mathcal{M}_{[\mathbf{v}_1\ \mathbf{e}_3]}(\mathbf{x}), \tag{37}$$

where $\mathbf{v}_1 = -\mathbf{e}_3 - \mathbf{e}_2$, $\mathbf{v}_2 = -\mathbf{e}_3 + \mathbf{e}_1$, $\mathbf{v}_3 = -\mathbf{e}_2 + \mathbf{e}_1$. Figure 9 shows how the box-splines, each one corresponding to the indicator function of a gray rhomboid, sum up to the regular hexagon, a case where $\mathbf{v}_1 = -\mathbf{e}_1$, $\mathbf{v}_2 = \mathbf{e}_2$, and $\mathbf{v}_3 = \mathbf{e}_3$.

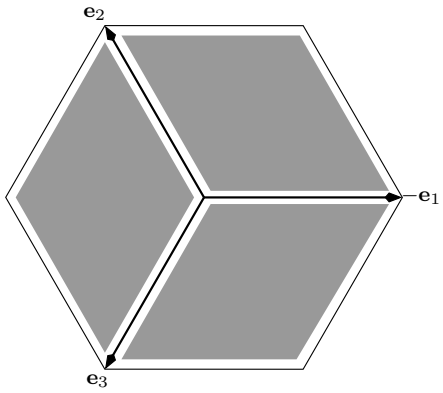

Fig. 9. Construction of the regular first-order hex-spline by the sum of three box-splines.

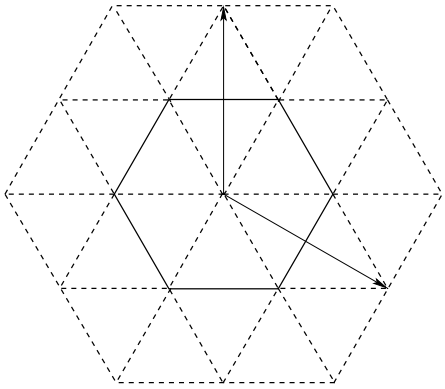

Fig. 10. Triangular mesh for the second-order hex-spline. There is one polynomial expression inside each triangle. Due to the twelve-fold symmetry, the computation can be restricted to three triangles.

For the regular case, the box-spline equivalence automatically implies that the function is a polynomial within each triangle inside the triangular mesh spanned by $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ [17]. This property tells us that we only need to determine the analytical form of the hex-splines in a limited number of regions. Figure 10 shows the triangular mesh for the second-order hex-spline, which separates the piecewise polynomial regions.

Sablonnière et al. [18] introduced other families of splines, one of them with a hexagonal support. Higher order splines were also constructed by repeated convolutions. However, our first-order hex-spline is excluded from their definition since their elementary building blocks are required to be continuous in the first place.

D. Discrete hex-splines

Most generally, a spline signal model is specified by the spline coefficients as

$$s(\mathbf{x}) = \sum_{\mathbf{k}} c(\mathbf{k}) \eta_p(\mathbf{x} - \mathbf{R}\mathbf{k}), \tag{38}$$

where the coefficients $c(\mathbf{k})$ are determined by

$$c(\mathbf{k}) = \int g(\mathbf{x})\tilde{\varphi}(\mathbf{x} - \mathbf{R}\mathbf{k})\mathrm{d}\mathbf{x}. \tag{39}$$

Here, g is the original function in the continuous domain and $\tilde{\varphi}$ the so-called prefilter. A common way to select the prefilter is by imposing the interpolation condition, i.e., we require the signal model to coincide with the original function at the lattice sites:

$$s(\mathbf{R}\mathbf{l}) = g(\mathbf{R}\mathbf{l}) = \sum_{\mathbf{k}} c(\mathbf{k})\eta_p(\mathbf{R}(\mathbf{l} - \mathbf{k})). \tag{40}$$

We can derive the equivalent prefilter $\tilde{\varphi}$ in the Fourier domain as

$$\hat{\tilde{\varphi}}(\boldsymbol{\omega}) = \frac{\Omega}{\sum_{\mathbf{k}} \hat{\eta}_p(\boldsymbol{\omega} - 2\pi\hat{\mathbf{R}}\mathbf{k})^\star} = \frac{1}{\sum_{\mathbf{k}} \eta_p(\mathbf{R}\mathbf{k}) \exp(j\langle \mathbf{k}, \mathbf{R}^{\mathrm{T}}\boldsymbol{\omega}\rangle)}. \tag{41}$$

A convenient way to present this prefilter operation is by using the discrete hex-splines. Discrete hex-splines are made out of the values of the hex-splines at the lattices sites:

$$h_p(\mathbf{k}) = \eta_p(\mathbf{R}\mathbf{k}). \tag{42}$$

Table I shows the coefficients of $h_p(\mathbf{k})$ up to sixth order for the regular case. Now, Eq. (40) is rewritten as

$$g(\mathbf{R}\mathbf{l}) = \sum_{\mathbf{k}} c(\mathbf{k})h_p(\mathbf{l} - \mathbf{k}), \tag{43}$$

which can be interpreted as a discrete convolution between $h_p(\mathbf{k}) = \eta_p(\mathbf{R}\mathbf{k})$ and the coefficients $c(\mathbf{k})$; i.e., h_p corresponds to a digital filter. Clearly, $c(\mathbf{k})$ can be obtained by filtering with h_p^{-1}. It is convenient to specify these filters in the $\mathcal{Z}$-transform domain. As an example, we provide the $\mathcal{Z}$-transform of the fourth-order discrete hex-spline

$$\begin{aligned}\check{h}_4(\mathbf{z}) = \sum_{\mathbf{k}} \mathbf{z}^{-\mathbf{k}}\eta_4(\mathbf{R}\mathbf{k}) = \\ \frac{37}{81} + \frac{29}{324}\left(z_1 + z_2 + z_1^{-1} + z_2^{-1} + z_1z_2 + z_1^{-1}z_2^{-1}\right) + \\ \frac{1}{972}\left(z_1^{-1}z_2 + z_1z_2^{-1} + z_1z_2^2 + z_1^{-1}z_2^{-2} + z_1^2z_2 + z_1^{-2}z_2^{-1}\right),\end{aligned}$$

where we have the notation $\mathbf{z}^{\mathbf{k}} = z_1^{k_1}z_2^{k_2}$. The Fourier transform of h_p is given by

$$\hat{h}_p(\boldsymbol{\omega}) = \check{h}_p(\exp(j\mathbf{R}^{\mathrm{T}}\boldsymbol{\omega})). \tag{44}$$

Appendix E explains how this inverse filtering operation is performed in practice.

E. Approximation properties

Approximation theory is useful to quantify to what extent we can expect a given function $g(\mathbf{x})$ to be approximated by a signal model $s(\mathbf{x})$. If we consider the (hypothetical) function $g(\mathbf{x})$ to be known in the continuous domain, and the model $s(\mathbf{x})$ to be the spline interpolation of the sample values on the lattice

$$\mathbf{R}_T = T\mathbf{R}, \quad T \in \mathbb{R}^+, \tag{45}$$

TABLE I

Coefficients of the discrete hex-splines. The lattice sites are indicated in Fig. 11.

order p	I	II	III	IV	V
1	1	0	0	0	0
2	1	0	0	0	0
3	42/72	5/72	0	0	0
4	37/81	29/324	1/972	0	0
5	40373/108864	32567/326592	1481/326592	395/653184	0
6	182393/583200	60353/583200	3881/437400	7583/3499200	29/3499200

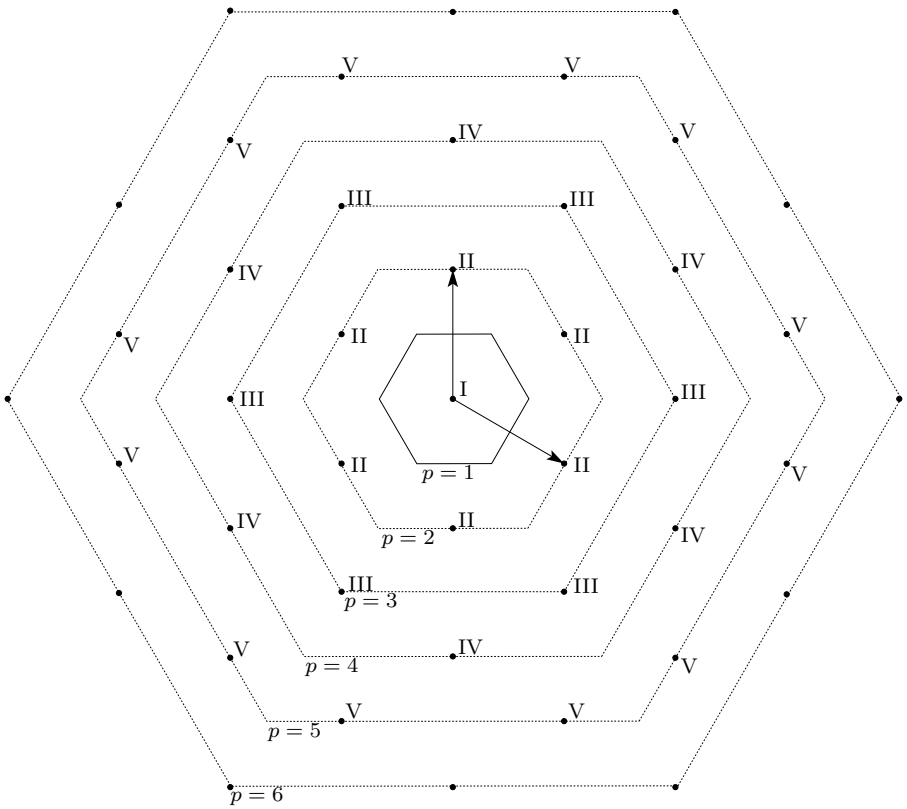

Fig. 11. As the order of the hex-splines increases, the support can be easily determined. Lattice sites are indicated and numbered. The hex-spline value at those sites are listed in Table I.

(with T a scaling factor enabling to refine the lattice), then we can examine how $s(\mathbf{x})$ approaches $g(\mathbf{x})$ as we make T smaller and smaller. The order of approximation is the power L of the sampling step T according to which the approximation error decreases:

$$||s(\mathbf{x}) - g(\mathbf{x})||^2 \propto T^{2L}. \tag{46}$$

Note that this is a purely theoretical question, since in practice $g(\mathbf{x})$ is not known.

A simple, yet powerful way to quantify the approximation error is to use the following formula in the Fourier domain: [19]

$$||s(\mathbf{x}) - g(\mathbf{x})||^2 \quad = \quad \int |\hat{g}(\boldsymbol{\omega})|^2 E(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega},$$

where $E(\boldsymbol{\omega})$ is the so-called error kernel. In our case, the extended form of the error kernel for a periodic lattice with matrix $\mathbf{R}$ is:

$$E(\boldsymbol{\omega}) \quad = \quad \left|1 - \hat{\tilde{\varphi}}(\boldsymbol{\omega})^\star \frac{\hat{\varphi}(\boldsymbol{\omega})}{\Omega}\right|^2 + \left|\hat{\tilde{\varphi}}(\boldsymbol{\omega})\right|^2 \sum_{\mathbf{n}\neq\mathbf{0}} \left|\frac{\hat{\varphi}(\boldsymbol{\omega} - 2\pi\hat{\mathbf{R}}\mathbf{n})}{\Omega}\right|^2, \tag{47}$$

where φ is the reconstruction function (i.e., the hex-spline η_p) and $\tilde{\varphi}$ the prefilter (e.g., the interpolation prefilter h_p^{-1}). By approximating $E(\boldsymbol{\omega})$ around $\mathbf{0}$ (thus, for $T \to 0$), we can determine the order approximation L; i.e., $E(\boldsymbol{\omega}) = O(||\boldsymbol{\omega}||^{2L})$ around the origin. The interpolation prefilter is given in Eq. (41). Next, we can make use of the Fourier expression of the reconstruction function

$$\frac{\hat{\varphi}(\boldsymbol{\omega})}{\Omega} \quad = \quad \frac{\hat{\eta}_p(\boldsymbol{\omega})}{\Omega} = \mathrm{sincH}_{\hat{\mathbf{R}}}(\boldsymbol{\omega}/(2\pi))^p, \tag{48}$$

which has zeros of order p at the reciprocal lattice sites; more precisely we have $\mathrm{sincH}_{\hat{\mathbf{R}}}((\boldsymbol{\omega}-2\pi\hat{\mathbf{R}}\mathbf{n})/(2\pi))^p = O(||\boldsymbol{\omega}||^p)$, for $\mathbf{n} \neq \mathbf{0}$ (see also Eq. (8)). As such, we obtain the following approximation for $E(\boldsymbol{\omega})$:

$$\left|1 - \frac{\hat{\varphi}(\boldsymbol{\omega})}{\hat{\varphi}(\boldsymbol{\omega}) + O(||\boldsymbol{\omega}||^p)}\right|^2 + \left|\hat{\tilde{\varphi}}(\boldsymbol{\omega})\right| O(||\boldsymbol{\omega}||^{2p}) = O(||\boldsymbol{\omega}||^{2p}), \quad \text{as } \boldsymbol{\omega} \to \mathbf{0},$$

with $\hat{\varphi}(\boldsymbol{\omega}) = \Omega + O(||\boldsymbol{\omega}||)$. This proves that the hex-spline indexing does indeed correspond to the order of approximation: p for η_p.

V. Applications

In this section, we apply the hex-splines to some prototypical image processing tasks.

A. Representation

Most naturally, the hex-splines provide a valuable tool to construct a continuously-defined function that interpolates sample values available on a hexagonal lattice. For demonstration purposes, we first resampled the "Lena" test image on a regular hexagonal lattice, and next used the hex-splines to represent the hexagonally sampled data. The direct spline transform (for its computation see Appendix E was used to compute the hex-spline coefficients. The hex-spline signal model, specified by these coefficients, can

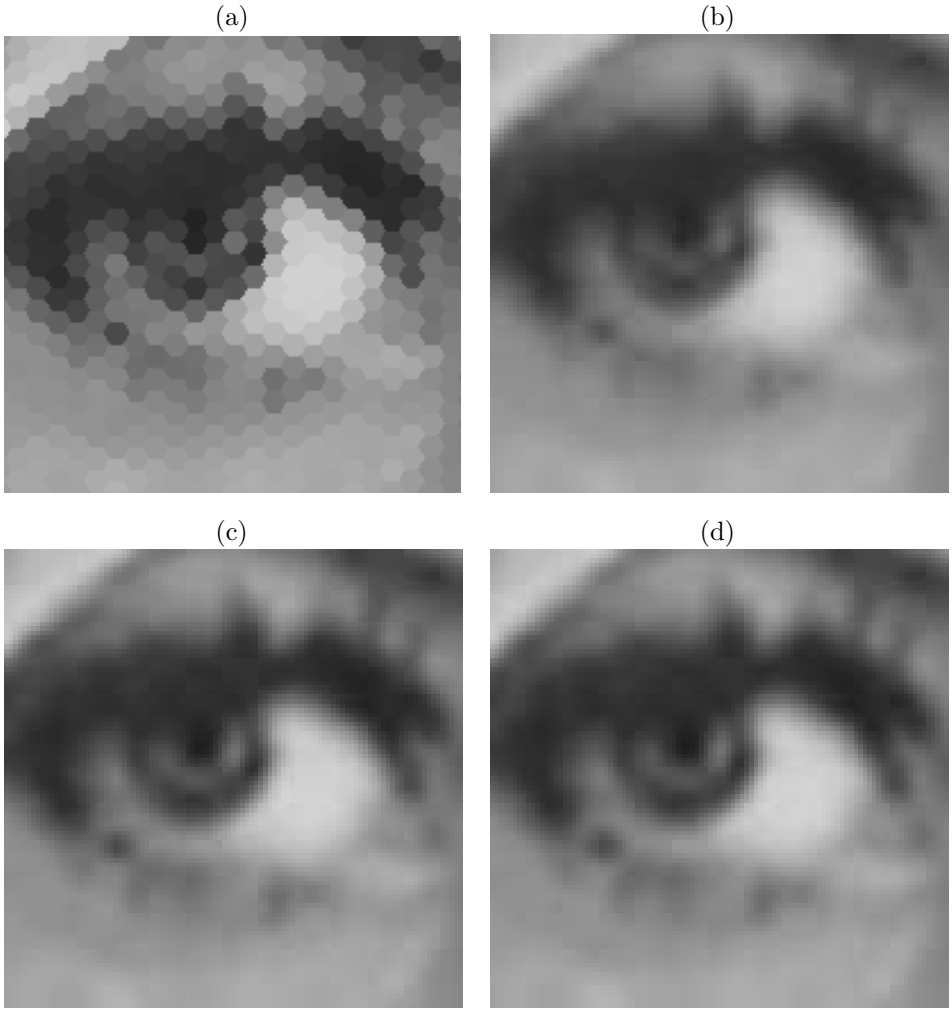

Fig. 12. The eye of "Lena", magnified using hex-splines. (a) First order. (b) Second order. (c) Third order. (d) Fourth order.

be evaluated using (11) at arbitrary locations, e.g., to zoom on the eye of "Lena". Figure 12 shows the results for first to fourth order. Clearly, as the order increases, the image appearance improves. Notice the small difference between third and fourth order.

B. Resampling

For image data acquired directly on hexagonal lattices, the representation by hex-splines can be of direct use. Using the hex-splines, new sample values can be computed on any new lattice, e.g., to obtain an image representation on a classical cartesian lattice. Possible applications include imaging from sensors that acquire data on a hexagonal capture grid [20, 21].

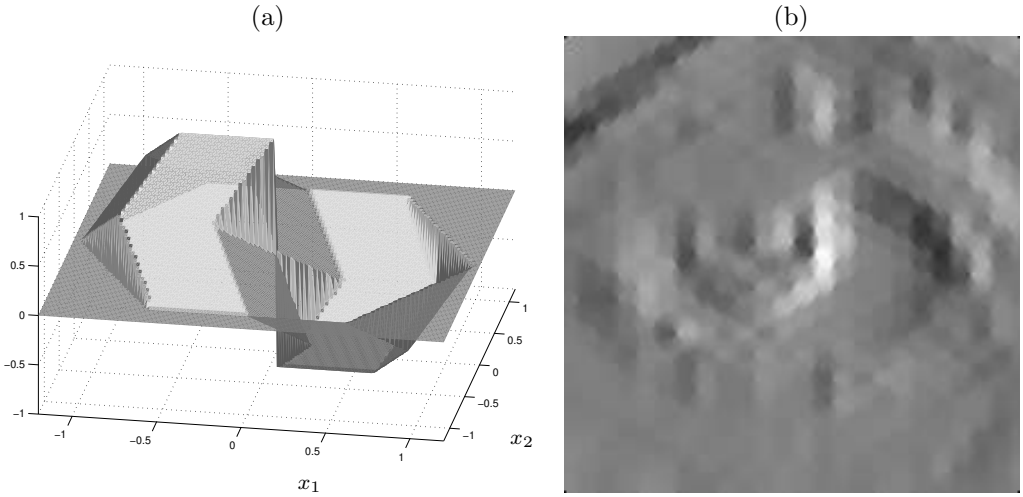

Fig. 13. (a) The first derivative in the $\mathbf{u}_1$-direction of the second-order hex-spline. (b) Applied to the eye of "Lena".

C. Hex-spline processing

Once the hex-spline coefficients are determined, several interesting operations can be applied directly in "hex-spline space" similarly as for classical B-spline processing [15]. Some examples are differentiation, filtering, and smoothing.

As an example, we describe the corresponding differentiation operation. Unlike for the classical B-splines, there is no direct formula to express a derivative in terms of lower order splines. Nevertheless, the derivative of a hex-spline can be found analytically by using the derivative of the sign-generating functions. In particular, since the derivative along the direction $\mathbf{u}_1$ corresponds to multiplying the Fourier transform by $j\langle \mathbf{u}_1, \boldsymbol{\omega} \rangle$, we obtain

$$\frac{\partial}{\partial \langle \mathbf{u}_1, \mathbf{x} \rangle} |\mathbf{x}|^{(n_1,n_2)}_{(\mathbf{u}_1,\mathbf{u}_2)} = |\mathbf{x}|^{(n_1-1,n_2)}_{(\mathbf{u}_1,\mathbf{u}_2)} . \tag{49}$$

As an example, we show the first derivative in the $\mathbf{u}_1$-direction of the second-order spline in Fig. 13 (a) and apply it to the eye of "Lena" in Fig. 13 (b).

A possible application of hex-spline processing is the modelling of a primate's vision system using hexagonal arrangements: the cones on the retina are organized in a hexagonal fashion [22, 23]. Also magnetic field computations can make use of the hex-splines since many real magnetic materials has a microscopic hexagonal structure [24, 25].

D. Least-squares resampling

Resampling from one lattice to another using a discrete/continuous model for the source lattice (such as proposed in Sect. V-B) does not take into account the properties of the target lattice. Consequently, resampling artefacts such as moiré patterns due to aliasing might arise, in particular when the target lattice is much coarser and when the original image contains many high-frequency components. Fortunately, the

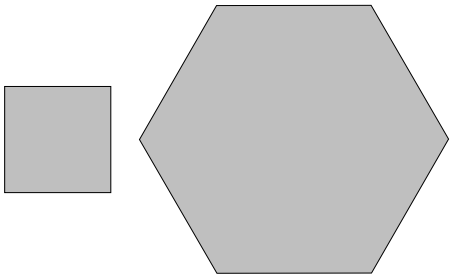

Fig. 14. To demonstrate the principle of resampling by the least-squares approach, consider the rectangular source lattice (Voronoi cell on the left) and the coarser hexagonal target lattice (Voronoi cell on the right).

use of spline models allows to elegantly incorporate information of the target lattice as proposed in [26]. We briefly describe here how this procedure can be extended to resample between two-dimensional periodic lattices in general.

Suppose that we have appropriate spline models, both for the source and the target lattice (which might be either orthogonal or hexagonal). We denote quantities related to the target lattice by an accent:

$$\begin{aligned} S(\eta_p) &= \left\{ s(\mathbf{x}) \;\middle|\; s(\mathbf{x}) = \sum_{\mathbf{k}} c(\mathbf{k}) \eta_p(\mathbf{x} - \mathbf{R}\mathbf{k}) \right\}, \\ S(\acute{\eta}_p) &= \left\{ \acute{s}(\mathbf{x}) \;\middle|\; \acute{s}(\mathbf{x}) = \sum_{\mathbf{k}} \acute{c}(\mathbf{k}) \acute{\eta}_p(\mathbf{x} - \acute{\mathbf{R}}\mathbf{k}) \right\}. \end{aligned}$$

Now, we want to determine the spline coefficients $\acute{c}(\mathbf{k})$ on the target lattice, such that the squared error between both signal representations $s(\mathbf{x})$ and $\acute{s}(\mathbf{x})$ (in the continuous domain) becomes minimal. This can be accomplished with the following algorithm, which is an adaptation of the algorithm in [26] for a hexagonal lattice.

- Determine the spline coefficients $c(\mathbf{k})$ on the source lattice by the direct spline transform, i.e., prefiltering by $(h_p)^{-1}$.
- Resample the spline coefficients $c(\mathbf{k})$ to

$$\acute{d}(\mathbf{k}) = \sum_{\mathbf{l}} c(\mathbf{l}) \xi(\acute{\mathbf{R}}\mathbf{k} - \mathbf{R}\mathbf{l}), \tag{50}$$

with $\xi(\mathbf{x}) = \eta_p * \acute{\eta}_p(\mathbf{x}) / \det(\acute{\mathbf{R}})$.

- Obtain the spline coefficients $\acute{c}(\mathbf{k})$ on the target lattice by post-filtering with $(\acute{h}_{2p})^{-1}$.

This algorithm can be used to resample between hexagonal lattices or to/from hexagonal and rectangular lattices. In general, the function $\xi(\mathbf{x})$ can be quite cumbersome to determine analytically, but in practice it can be approximated numerically. For example, consider resampling from a rectangular lattice to a much coarser hexagonal lattice. The corresponding Voronoi cells are shown in Fig. 14. Figures 15 (a) and (b) depict the functions $\xi(\mathbf{x})$ for first and second order. For the first order ($p = 1$), no pre- or post-filters are required. For the second order ($p = 2$), only the post-filter is required and is identical

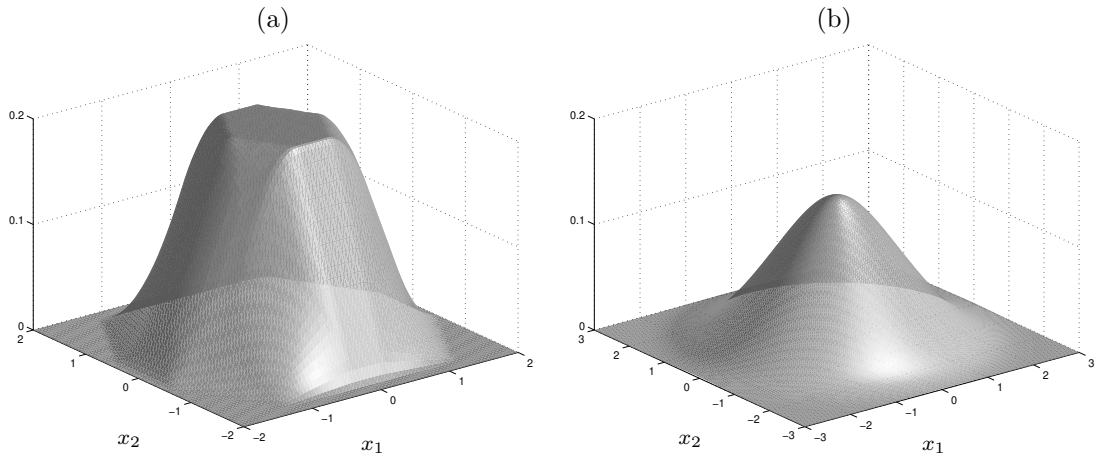

Fig. 15. Function $\xi(\mathbf{x})$ used to resample the spline coefficients from the rectangular to the hexagonal lattice by the least-squares approach. (a) First order. (b) Second order.

to the prefilter for the fourth-order hex-spline ($p = 4$). The results are shown in Fig. 16 for a part of the test image "shirt". Clearly, the classical interpolative approach of Fig. 16 (b) (using cubic B-spline interpolation on the rectangular source lattice) does not take into account the target lattice and produces moiré artifacts due to aliasing. On the other hand, resampling by the least-squares approach significantly improves the quality. The first order case of Fig. 16 (c) corresponds to so-called "surface projection", i.e., the contribution of an original sample value to a new one depends on the relative overlap between the corresponding Voronoi cells. The second order case of Fig. 16 (d) provides a sharper result. For higher order least-squares resampling, the function $\xi(\mathbf{x})$ can be approximated quite well by a Gaussian function.

The least-squares approach to resampling is powerful and can be useful to many applications. In [12], we applied the hex-splines to a gravure printing application. Another application is a moiré-suppressing prefilter for color printing, as proposed in [27]. Those papers did not use the current implementation (by Eq. (50)), which is more efficient and does not require to approximate a function with infinite support.

VI. Conclusion

In this paper, we proposed and studied a novel family of bivariate, non-separable splines for hexagonal lattices. These splines are defined in a natural way, starting from the Voronoi cell definition. We studied the mathematical foundations and presented the algorithmic tools required to make full use of the hex-splines. As a summary, we show in Table II the comparison of the most important properties and features of B-splines and hex-splines.

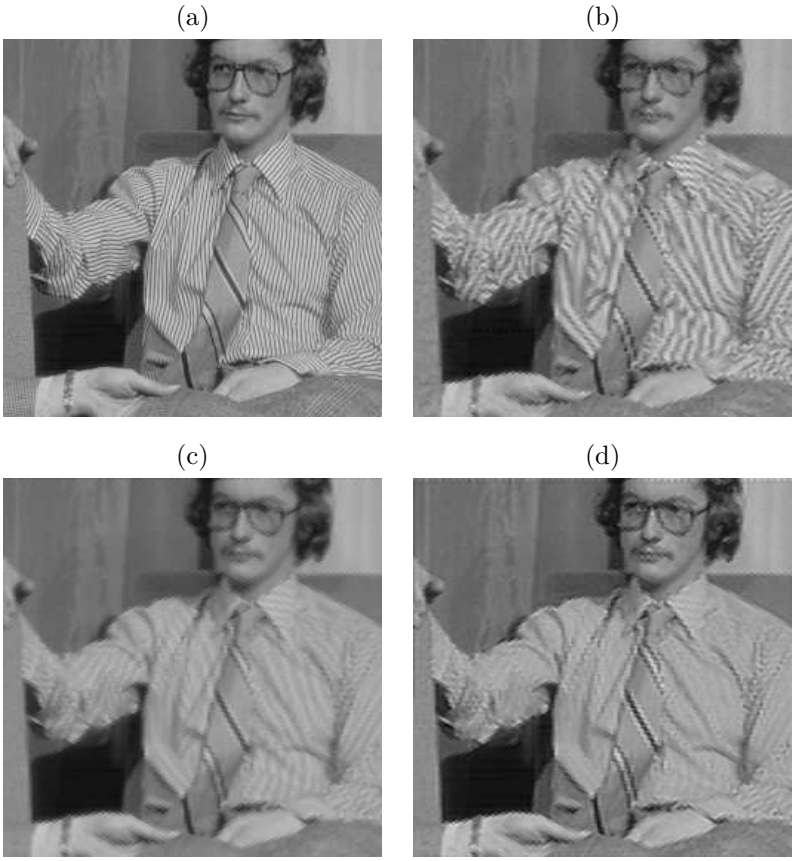

Fig. 16. (a) Part of the test image "shirt". (b) Result after resampling using the interpolative approach (cubic B-spline interpolation on the source lattice). (c) Results after resampling using the least-squares approach, first order ($p = 1$). (d) Results after resampling using the least-squares approach, second order ($p = 2$).

Appendices

A. Poisson sum formula

To derive the Poisson sum formula for a hexagonal lattice, we start with the classical formula for an orthogonal lattice with $\mathbf{k} \in \mathbb{Z}^2$:

$$\sum_{\mathbf{k}} f(\mathbf{x} - \mathbf{k}) = \sum_{\mathbf{k}} \hat{f}(2\pi\mathbf{k}) \exp(j2\pi\langle \mathbf{k}, \mathbf{x}\rangle). \tag{51}$$

Introducing the function $f(\mathbf{x}) = g(\mathbf{R}\mathbf{x})$, we obtain

$$\sum_{\mathbf{k}} g(\mathbf{R}\mathbf{x} - \mathbf{R}\mathbf{k}) = \frac{1}{|\det(\mathbf{R})|} \sum_{\mathbf{k}} \hat{g}(2\pi\hat{\mathbf{R}}\mathbf{k}) \exp(j2\pi\langle \mathbf{k}, \mathbf{x}\rangle). \tag{52}$$

Finally, we substitute $\mathbf{R}\mathbf{x} = \mathbf{y}$, which yields the Poisson sum formula for an hexagonal lattice:

$$\sum_{\mathbf{k}} g(\mathbf{y} - \mathbf{R}\mathbf{k}) = \frac{1}{|\det(\mathbf{R})|} \sum_{\mathbf{k}} \hat{g}(2\pi\hat{\mathbf{R}}\mathbf{k}) \exp(j2\pi\langle \mathbf{R}^{\mathrm{T}}\mathbf{k}, \mathbf{y}\rangle). \tag{53}$$

TABLE II

COMPARISON BETWEEN CLASSICAL TENSOR-PRODUCT B-SPLINES AND HEX-SPLINES.

	B-splines $\beta^n(\mathbf{x}) = \beta^n(x_1)\beta^n(x_2)$	Hex-splines $\eta_p(\mathbf{x})$
generating functions	$(x_1)_+^n (x_2)_+^n$	$(\mathbf{x})_{/}^n$, $(\mathbf{x})_{\backslash}^n$, and crossterms
localization operator	$\Delta(x_1)\Delta(x_2)$	$\Delta_{/}(\mathbf{x})$, $\Delta_{\backslash}(\mathbf{x})$, and crossterms
convolution property	$\beta^n(\mathbf{x}) = \beta^{n-1} * \beta^0(\mathbf{x})$	$\eta_p(\mathbf{x}) = \frac{\eta_{p-1} * \eta_1(\mathbf{x})}{\Omega}$
Fourier formula	$(\mathrm{sinc}(\omega_1/(2\pi))\mathrm{sinc}(\omega_2/(2\pi)))^{n+1}$	$\mathrm{sincH}_{\hat{\mathbf{R}}}(\boldsymbol{\omega}/(2\pi))^p$
order of approximation	$n+1$	p
polynomial degree	$2n$	$2(p-1)$
symmetry	eight-fold	twelve-fold
positivity	yes	yes
partition of unity	yes	yes
compactly supported	yes, square of area $(n+1)^2$	yes, hexagon of area $p^2\Omega$
separability	yes	no
induction formula	yes	only for generating functions

B. RECURRENCE FORMULA FOR GENERATING FUNCTIONS

The recurrence equation of (21) can be demonstrated by using the Fourier transform of their sign-version (see Eq. (30)).

The recursive relationship (23) for the case of crossterms, is also proved using the sign-version. Consider the first-order generating functions using respectively the vectors $\mathbf{u}_1 = [1\ 0]^{\mathrm{T}}$ and $\mathbf{u}_2 = [u\ 1]^{\mathrm{T}}$, and $\mathbf{u}_1 = [1\ 0]^{\mathrm{T}}$ and $\mathbf{u}_3 = [u\ -1]^{\mathrm{T}}$. The Fourier transforms correspond to

$$\frac{1}{(j\omega_1)(j(u\omega_1+\omega_2))},$$
$$\frac{1}{(j\omega_1)(j(u\omega_1-\omega_2))},$$

with

$$\mathbf{M}^{-1} = \begin{bmatrix} 1 & u \\ 0 & 1 \end{bmatrix}. \tag{54}$$

The crossterm $|\mathbf{x}|^{(1,0,0)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)}$ of both generating functions, corresponds to the following Fourier expression,

which can be separated using partial fractions and $\zeta = \omega_2/u\omega_1$:

$$\begin{aligned}
&\frac{1}{(j\omega_1)^2(j(u\omega_1+\omega_2))(j(u\omega_1-\omega_2))} \\
&= \frac{1}{(j\omega_1)^2(ju\omega_1)^2(1+\zeta)(1-\zeta)} \\
&= \frac{1}{2(j\omega_1)^2(ju\omega_1)^2}\left(\frac{1}{1+\zeta}+\frac{1}{1-\zeta}\right) \\
&= \frac{1}{2\langle \mathbf{u}_1,\mathbf{u}_2\rangle (j\omega_1)^3}\left(\frac{1}{(j(u\omega_1+\omega_2))}+\frac{1}{(j(u\omega_1-\omega_2))}\right),
\end{aligned}$$

which gives us

$$|\mathbf{x}|^{(1,0,0)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)} = \frac{1}{2\langle \mathbf{u}_1,\mathbf{u}_2\rangle}\left(|\mathbf{x}|^{(2,0)}_{(\mathbf{u}_1,\mathbf{u}_2)} + |\mathbf{x}|^{(2,0)}_{(\mathbf{u}_1,\mathbf{u}_3)}\right).$$

Applying this equation to a general crossterm results in Eq. (23).

This recursive equation can also be used to compute some derivative generating functions. For example, when computing the derivative in the direction of $\mathbf{u}_2$, the Fourier transform of one of the generating functions can get a $(j\langle \mathbf{u}_2,\boldsymbol{\omega}\rangle)$ in the nominator. This can be undone by applying

$$(\mathbf{x})^{(n_1+1,n_2-1,n_3)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)} = 2\langle \mathbf{u}_1,\mathbf{u}_2\rangle (\mathbf{x})^{(n_1,n_2,n_3)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)} - (\mathbf{x})^{(n_1+1,n_2,n_3-1)}_{(\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)}.$$

C. Localization of two-sided generating functions

We briefly show how the sign-version of $|\mathbf{x}|^0_{(\mathbf{u}_1,\mathbf{u}_2)} = |\mathbf{x}|^0_{/}$ gets localized by $\Delta_{/}$. We start with the following lemma.

Lemma 1: Given a one-sided generating function $(\mathbf{x})^0_{(\mathbf{u}_1,\mathbf{u}_2)}$ and the corresponding $\Delta_{/}$, we have:

$$\Delta_{/} * (\mathbf{x})^0_{(-\mathbf{u}_1,\mathbf{u}_2)} = -\Delta_{/} * (-\mathbf{x})^0_{(\mathbf{u}_1,\mathbf{u}_2)}, \tag{55}$$

$$\Delta_{/} * (\mathbf{x})^0_{(\mathbf{u}_1,-\mathbf{u}_2)} = -\Delta_{/} * (\mathbf{x})^0_{(\mathbf{u}_1,\mathbf{u}_2)}. \tag{56}$$

Proof: The Fourier transforms of the one-sided generating functions $(\mathbf{x})^0_{(-\mathbf{u}_1,\mathbf{u}_2)}$ and $-(-\mathbf{x})^0_{(\mathbf{u}_1,\mathbf{u}_2)}$ are respectively

$$\begin{aligned}
(\mathbf{x})^0_{(-\mathbf{u}_1,\mathbf{u}_2)} &\longleftrightarrow \left(\frac{-1}{j\langle \boldsymbol{\omega},\mathbf{u}_1\rangle}+\pi\delta(\langle \boldsymbol{\omega},\mathbf{u}_1\rangle)\right)\left(\frac{1}{j\langle \boldsymbol{\omega},\mathbf{u}_2\rangle}+\pi\delta(\langle \boldsymbol{\omega},\mathbf{u}_2\rangle)\right) \\
&= \frac{1}{\langle \boldsymbol{\omega},\mathbf{u}_1\rangle\langle \boldsymbol{\omega},\mathbf{u}_2\rangle} - \frac{\pi\delta(\langle \boldsymbol{\omega},\mathbf{u}_2\rangle)}{j\langle \boldsymbol{\omega},\mathbf{u}_1\rangle} + \frac{\pi\delta(\langle \boldsymbol{\omega},\mathbf{u}_1\rangle)}{j\langle \boldsymbol{\omega},\mathbf{u}_2\rangle} + \pi^2\delta(\langle \boldsymbol{\omega},\mathbf{u}_1\rangle)\delta(\langle \boldsymbol{\omega},\mathbf{u}_2\rangle), \\
-(-\mathbf{x})^0_{(\mathbf{u}_1,\mathbf{u}_2)} &\longleftrightarrow \frac{1}{\langle \boldsymbol{\omega},\mathbf{u}_1\rangle\langle \boldsymbol{\omega},\mathbf{u}_2\rangle} + \frac{\pi\delta(\langle \boldsymbol{\omega},\mathbf{u}_2\rangle)}{j\langle \boldsymbol{\omega},\mathbf{u}_1\rangle} + \frac{\pi\delta(\langle \boldsymbol{\omega},\mathbf{u}_1\rangle)}{j\langle \boldsymbol{\omega},\mathbf{u}_2\rangle} - \pi^2\delta(\langle \boldsymbol{\omega},\mathbf{u}_1\rangle)\delta(\langle \boldsymbol{\omega},\mathbf{u}_2\rangle).
\end{aligned}$$

Using this result, the remaining terms of the difference between the right and left hand side of Eq. (55) can be "neutralized" using the localization of $\delta(\langle \boldsymbol{\omega},\mathbf{u}_2\rangle)$, which equals zero:

$$\begin{aligned}
\hat{\Delta}_{/}(\boldsymbol{\omega})\delta(\langle \boldsymbol{\omega},\mathbf{u}_2\rangle) &= \hat{\Delta}(\langle \boldsymbol{\omega},\mathbf{e}_1+\mathbf{e}_2\rangle)\hat{\Delta}(\langle \boldsymbol{\omega},\mathbf{e}_1-\mathbf{e}_2\rangle)\delta(\langle \boldsymbol{\omega},\mathbf{u}_2\rangle) \\
&= \hat{\Delta}(\langle \boldsymbol{\omega},\mathbf{e}_1+\mathbf{e}_2\rangle)\underbrace{\hat{\Delta}(\langle \boldsymbol{\omega},\lambda\mathbf{u}_2\rangle)\delta(\langle \boldsymbol{\omega},\mathbf{u}_2\rangle)}_{0} \\
&= 0,
\end{aligned}$$

where $\lambda = -\langle \mathbf{e}_2 - \mathbf{e}_1, [0\ 1]^{\mathrm{T}} \rangle$, according to the definition of $\mathbf{u}_2$. $\square$

Next, we write the two-sided generating function $|\mathbf{x}|^0_{/}$ in terms of one-sided generating functions using Eq. (26) and applying Eqs. (55) and (56):

$$\begin{aligned} \Delta_{/} * |\mathbf{x}|^0_{/} &= \frac{\Delta_{/}}{4} * \left((\mathbf{x})^0_{(\mathbf{u}_1,\mathbf{u}_2)} - (\mathbf{x})^0_{(\mathbf{u}_1,-\mathbf{u}_2)} + (\mathbf{x})^0_{(-\mathbf{u}_1,-\mathbf{u}_2)} - (\mathbf{x})^0_{(-\mathbf{u}_1,\mathbf{u}_2)} \right) \\ &= \frac{\Delta_{/}}{2} * \left((\mathbf{x})^0_{(\mathbf{u}_1,\mathbf{u}_2)} + (-\mathbf{x})^0_{(\mathbf{u}_1,\mathbf{u}_2)} \right) \\ &= \frac{\Delta_{/}}{2} * \left((\mathbf{x})^0_{/} + (-\mathbf{x})^0_{/} \right). \end{aligned} \tag{57}$$

This corresponds to a symmetrized version of the "intermediate" part of the hex-spline (this can easily checked by applying the construction algorithm of Sect. III-B to $(-\mathbf{x})^0_{/}$). Analogously we can obtain

$$\Delta_{\backslash} * |\mathbf{x}|^0_{\backslash} = \frac{\Delta_{\backslash}}{2} * \left((\mathbf{x})^0_{\backslash} + (-\mathbf{x})^0_{\backslash} \right). \tag{58}$$

Adding up both parts, i.e., Eqs. (57) and (58), results into the first-order hex-spline.

D. Implementation: the analytical form

To obtain the analytical expression of the hex-splines it is safer to rely on a mathematical software package for symbolic manipulations. We have made an implementation available for Maple at `http://bigwww.epfl.ch/demo/hexsplines/`. This software is also able to obtain the analytical form of irregular hex-splines, using the same construction technique as explained in this article. In particular, we assume the following lattice matrix $\mathbf{R}$ is given, corresponding to the general class of semi-regular hexagonal lattices of the second type [14]:

$$\mathbf{R} = \begin{bmatrix} b & 0 \\ -\frac{a}{2} & a \end{bmatrix}, \tag{59}$$

with $2b > a > 0$. The values a and b can be passed to the program. Hexagonal lattices of the first type can easily be obtained by exchanging x_1 and x_2.

The package makes use of two auxiliary functions:

- `hexlocalization`: Computes the localization operator given the vectors $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$. The arguments `k1` and `k2` indicate the power to which both localization operators are raised.
- `hexbuilding`: Computes a generating function given the vectors $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$. The argument `pow` is a vector containing the powers (n_1, n_2, n_3) of a general term. In the implementation, the powers correspond to the powers of the Fourier terms. The coordinates `xc1` and `xc2` are used for the Heaviside terms which delineate the support of the generating functions. This allows us to compute the analytical form inside a triangular patch of the hex-splines.

These auxiliary functions are called by the main routine `diffhexspline` that composes a general hex-spline with lattice parameters a and b, a given order, and eventually derivatives in one of the three

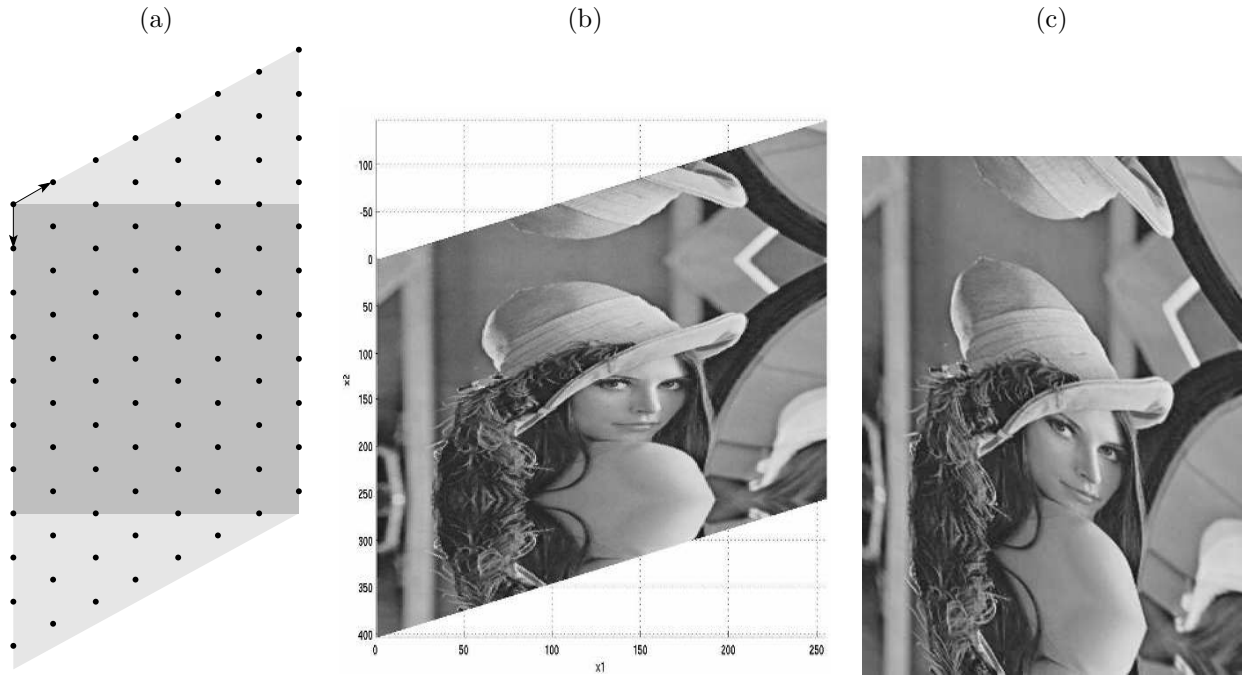

Fig. 17. (a) A square image, sampled on a hexagonal lattice, is embedded by a rectangular support. The light gray regions are redundant. (b) An example for the test image "Lena", mirror boundary conditions are applied. (c) The test image "Lena" in the integer coordinate system (k_1, k_2).

main directions. The other functions initialize some parameters for common cases: `hexspline` omits the derivative, `diffrhexspline` is for a regular hex-spline, and `rhexspline` computes the regular hex-spline without derivatives.

E. Implementation: the direct spline transform

The direct spline transform computes the spline coefficients for a hex-spline representation given the sample values. As shown in Sect. IV-D, this corresponds to the inverse filter operation of the corresponding discrete hex-spline. For one-dimensional B-splines (and higher-dimensional versions by the tensor-product extension), a fast recursive algorithm exists [28]. Unfortunately, the discrete hex-splines cannot be separated in a series of causal and anti-causal recursive filters. In this paper, we propose an equivalent implementation of the inverse filter in Fourier domain by $1/\hat{h}_p(\boldsymbol{\omega})$.

There are two approaches to compute the Fourier coefficients of samples taken on a hexagonal grid. First, one can make use of a true hexagonal discrete Fourier transform [29,30]. Second, the hexagonal data is embedded in a rectangular support spanned by the lattice vectors, i.e., the index $[k_1 k_2]^{\mathrm{T}}$ $(0 \leq k_1 < N_1$, $0 \leq k_2 < N_2)$ relates to $k_1\mathbf{r}_1 + k_2\mathbf{r}_2$. Here, we have preferred the second approach which can make use of widely available FFT-algorithms on a rectangular lattice. First, the data is embedded as shown in Fig. 17. Regions outside the support of the original image is extended by mirror boundary conditions. The result, i.e., the image of Fig. 17 (c), is indexed as $g_{\mathbf{R}}(\mathbf{k}) = g(\mathbf{R}\mathbf{k})$. Next, the Fourier coefficients is computed by

a regular FFT-algorithm. Taking into account Eqs. (3) and (44), filtering needs to be done by

$$\frac{|\det(\mathbf{R})|}{\hat{h}_p(\mathbf{R}^{-\mathrm{T}}\mathbf{n})}, \tag{60}$$

where $\mathbf{n} = [n_1/N_1 \; n_2/N_2]^{\mathrm{T}}$, $0 \leq n_1 < N_1$, $0 \leq n_2 < N_2$. Finally, the inverse FFT provides the spline coefficients.

Acknowledgments

The authors would like to express their gratitude to the anonymous reviewers for their valuable remarks that have contributed to the final quality of this paper.

This work was partially supported by the FWO (Fund for Scientific Research - Flanders, Belgium), through a mandate of Research Assistant (Dimitri Van De Ville).

References

[1] D.P. Petersen and D. Middleton, "Sampling and reconstruction of wave-number-limited functions in N-dimensional Euclidean spaces," *Information and control*, vol. 5, pp. 279–323, 1962.

[2] R.M. Mersereau, "The processing of hexagonally sampled two-dimensional signals," *Proceedings of the IEEE*, vol. 67, no. 6, pp. 930–949, June 1979.

[3] R.M. Mersereau, "Two-Dimensional Nonrecursive Filter Design," *Topics in Applied Physics: Two-Dimensional Digital Signal Processing I*, vol. 42, 1981.

[4] Wenzhe Li and Alfred Fettweis, "Interpolation filters for 2-D hexagonally sampled signals," *International Journal of Circuit Theory and Applications*, vol. 25, pp. 259–277, 1997.

[5] M. J. E. Golay, "Hexagonal parallel pattern transformations," *IEEE Transactions on Computers*, vol. C-18, no. 8, pp. 733–740, Aug. 1969.

[6] Richard C. Staunton, "The design of hexagonal sampling structures for image digitisation and their use with local operators," *Image and Vision Computing*, vol. 7, no. 3, pp. 162–166, 1989.

[7] Lee Middleton and Jayanthi Sivaswamy, "Edge detection in a hexagonal-image processing framework," *Image and Vision Computing*, vol. 19, no. 14, pp. 1071–1081, Dec. 2001.

[8] A. Laine and S. Schuler, "Mammographic feature enhancement by multiscale analysis," *IEEE Transactions on Medical Imaging*, vol. 13, no. 4, pp. 725–740, Dec. 1994.

[9] A. P. Fitz and R. J. Green, "Fingerprint classification using a hexagonal fast fourier transform," *Pattern Recognition*, vol. 29, no. 10, pp. 1587–1597, 1996.

[10] Senthil Periaswamy, "Detection of microcalcifications in mammograms using hexagonal wavelets," M.S. thesis, University of South Carolina, 1996.

[11] R. C. Staunton, "One-pass parallel hexagonal thinning algorithm," *IEE Proc. Vision, Image and Signal Processing*, vol. 148, no. 1, pp. 45–53, 2001.

[12] Dimitri Van De Ville, Wilfried Philips, and Ignace Lemahieu, "Least-squares spline resampling to a hexagonal lattice," *Signal Processing: Image Communication*, vol. 17, no. 5, pp. 393–408, May 2002.

[13] Eric Dubois, "The sampling and reconstruction of time-varying imagery with application in video systems," *Proceedings of the IEEE*, vol. 73, no. 4, pp. 502–522, Apr. 1985.

[14] Robert A. Ulichney, *Digital Halftoning*, MIT Press, 1987.

[15] Michael Unser, Akram Aldroubi, and Murray Eden, "B-spline signal processing," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 821–848, Feb. 1993.

[16] C. de Boor, K. Höllig, and S. Riemenschneider, *Box splines*, vol. 98 of *Applied Mathematical Sciences*, Springer-Verlag, 1993.

[17] C. de Boor and K. Höllig, "B-splines from parallelepipeds," *J. Analyse Math.*, , no. 42, pp. 99–115, 1982.

[18] Paul Sablonnière and Driss Sbibih, "B-splines with hexagonal support on a uniform three-direction mesh of plane," *C. R. Acad. Sci. Paris*, vol. 319, no. I, pp. 277–282, 1994.

[19] Thierry Blu and Michael Unser, "Quantitative Fourier analysis of approximation techniques: Part I—interpolators and projectors," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2783–2795, Oct. 1999.

[20] Marc Tremblay, Stephane Dallaire, and Denis Poussart, "Low level segmentation using CMOS smart hexagonal image sensor," in *Proceedings of the Conference on Computer Architectures for Machine Perception*. 1995, pp. 21–28, IEEE.

[21] Stefan Jung, Roland Thewes, Thomas Scheiter, Karl F. Goser, and Werner Weber, "Low-power and high-performance CMOS fingerprint sensing and encoding architecture," *IEEE Journal of Solid State Circuits*, vol. 34, no. 7, pp. 978–984, July 1999.

[22] D.H. Mugler and M. D. Ross, "Vestibular receptor cells and signal detection: Bioaccelerometers and the hexagonal sampling of two-dimensional signal," *Mathematical and Computer Modelling*, vol. 13, no. 2, pp. 85–92, 1990.

[23] Brian A. Wandell, *Foundations of Vision*, Sinauer Associates, 1995.

[24] M. Mansuripur and R. Giles, "Demagnetizing fields computation for dynamic simulation of the magnetization reversal process," *IEEE Transactions on Magnetics*, vol. 24, no. 1, pp. 23–27, Jan. 1988.

[25] R. Giles, P. R. Kotiuga, and M. Mansuripur, "Parallel micromagnetic simulations using fourier methods on a regular hexagonal lattice," *IEEE Transactions on Magnetics*, vol. 27, no. 9, pp. 3815–3818, Sept. 1991.

[26] M. Unser, A. Aldroubi, and M. Eden, "Enlargement or reduction of digital images with minimum loss of information," *IEEE Transactions on Image Processing*, vol. 4, no. 3, pp. 247–258, Mar. 1995.

[27] Dimitri Van De Ville, Wilfried Philips, Ignace Lemahieu, and Rik Van de Walle, "Suppression of sampling moire in color printing by spline-based least-squares prefiltering," *Pattern Recognition Letters*, vol. 24, no. 11, pp. 1787–1794, July 2002.

[28] M. Unser, "Fast B-spline transforms for continuous image representation and interpolation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 277–285, Mar. 1991.

[29] James C. Ehrhardt, "Hexagonal fast fourier transform with rectangular output," *IEEE Transactions on Signal Processing*, vol. 41, no. 3, pp. 1469–1472, Mar. 1993.

[30] A. M. Grigoryan, "Efficient algorithms for computing the 2-D hexagonal fourier transforms," *IEEE Transactions on Signal Processing*, vol. 50, no. 6, pp. 1438–1448, June 2002.