# FAST ADAPTIVE ELLIPTICAL FILTERING USING BOX SPLINES

*Kunal Narayan Chaudhury†, Arrate Muñoz Barrutia◇ and Michael Unser†*

† Biomedical Imaging Group (BIG), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
◇ Center for Applied Medical Research (CIMA), University of Navarra, ES-31008 Pamplona, Spain

## ABSTRACT

We demonstrate that it is possible to filter an image with an elliptic window of varying size, elongation and orientation with a fixed computational cost per pixel. Our method involves the application of a suitable global pre-integrator followed by a pointwise-adaptive localization mesh. We present the basic theory for the 1D case using a B-spline formalism and then appropriately extend it to 2D using *radially-uniform box splines*. The size and ellipticity of these radially-uniform box splines is adaptively controlled. Moreover, they converge to Gaussians as the order increases. Finally, we present a fast and practical directional filtering algorithm that has the capability of adapting to the local image features.

***Index Terms***— B-spline, Finite-difference filter, Running-sum filter, Box spline, Directional filter.

## 1. INTRODUCTION

The most common smoothing operator is the Gaussian filter. For that reason, it is of practical interest to design efficient directional filtering strategies based on this type of filter. Fast recursive solutions for space-invariant Gaussian-like filtering have been developed [1] but the space-variant ones are subject of current research. To date, two algorithms have been proposed: one that works in 1D and uses B-splines for fast computations of the Continuous Wavelet Transform [2], and the other proposed in Computer Graphics that essentially does a rectangular smoothing using repeated integration [3].

In this paper, we propose a general technique for $N$-directional adaptive filtering using box spline formalism and discuss its implementation for the special four-directional case. Although the derivation is rather involved, the final solution is quite simple (convolution-like with an adaptive mesh) and rather easy to implement (cf. Eqn. (18)). Also, the algorithm has a constant computational cost per pixel as the support of the adaptive mesh is independent of the pointwise-adaptive scale-vector.

The paper is organized as follows. In § 2, we revisit B-splines and certain associated linear operators. In § 3, we describe an efficient B-spline based adaptive filtering technique, initially proposed in [2]. We then introduce the family of radially-uniform box splines and some related linear operators in § 4 and § 5, respectively. Finally, in § 6, we propose the adaptive directional filtering strategy followed by the description of a fast algorithm in § 7.

This work was supported in part by the Swiss National Science Foundation (under grant 200020-109415) and the Ramon y Cajal Program.

## 2. 1D LINEAR OPERATORS AND B-SPLINES

We first introduce two linear, shift-invariant operators, initially defined for real-valued functions $f(x), x \in \mathbb{R}$, and then appropriately applied to real-valued sequences $\{g[k]\}_{k\in\mathbb{Z}}$.

**Definition 2.1** *The finite-difference (FD) operator $\Delta_a^n$ of order $n \in \mathbb{Z}_+$ and scale $a \in \mathbb{R}_+$ is given by*

$$\Delta_a^n\{f\}(x) = \sum_{k=0}^{n} d_a^n[k] f(x - ak) \tag{1}$$

*where $d_a^n[k] := a^{-n}(-1)^k \binom{n}{k}$, for $0 \leqslant k \leqslant n$ and $0$, else.*

In the Fourier domain, we have: $\Delta_a^n\{f\}^\wedge(\omega) = \widehat{\Delta}_a^n(e^{ja\omega})\widehat{f}(\omega)$, with $\widehat{\Delta}_a^n(e^{ja\omega}) := \sum_k d_a^n[k]e^{-ja\omega k} = a^{-n}(1-e^{-ja\omega})^n$ interpreted as the $(2\pi/a)$-periodic frequency response of the FD filter $\{d_a^n[k]\}_{k\in\mathbb{Z}}$.

When acting on sequences $\{g[k]\}_{k\in\mathbb{Z}}$ , the FD filter acts as a discrete convolution operator. For integer $a$, we have $\Delta_a^n\{g\}[m] = \sum_{k=0}^n d_a^n[k]g[m-ak]$; for non-integer $a$, some form of interpolation is necessary as $g$ is not defined for non-integer arguments.

**Definition 2.2** *The running-sum (RS) operator $\Delta_b^{-1}$ of scale $b \in \mathbb{R}_+$ is given by*

$$\Delta_b^{-1}\{f\}(x) = b\sum_{k=0}^{\infty} f(x - bk), \tag{2}$$

*provided the series converges for each $x \in \mathbb{R}$.*

For integer $b$, the RS operator applied to sequences $\{g[k]\}_{k\in\mathbb{Z}}$ corresponds to the digital filter $u_b[k] := b$, for $k \in b\mathbb{N}_0$ and $0$, otherwise, with the correspondence: $\Delta_b^{-1}\{g\}[n] = \sum_{k\in\mathbb{Z}} u_b[k]g[n-k]$. Note that $y = u_b * g$ can be implemented very efficiently using the recursive equation $y[m] =$

$y[m-b]+bg[m]$, with appropriate boundary conditions [2]. The $n$-fold composition of the above RS operator will be denoted by $\Delta_b^{-n}$ with frequency response $\widehat{\Delta}_b^{-n}(e^{jb\omega}) = b^n(1-e^{-jb\omega})^{-n}$; for integer $b$, the corresponding filter $\{u_b^n[k]\}_{k\in\mathbb{Z}}$ is specified by $\sum_{k\in\mathbb{Z}} u_b^n[k]e^{-j\omega k} = b^n(1-e^{-jb\omega})^{-n}$.

Finally, we introduce the family of symmetric B-splines [4] that are closely related to the above linear operators; it is the following Fourier domain definition that makes the link apparent.

**Definition 2.3** *The symmetric B-spline $\beta_a^n(x)$ of degree $n \in \mathbb{N}_0$ and of scale $a \in \mathbb{R}_+$ is specified by the Fourier transform*

$$\widehat{\beta}_a^n(\omega) = \frac{1}{a^{n+1}}\left(\frac{e^{ja\omega/2}-e^{-ja\omega/2}}{j\omega}\right)^{n+1} \tag{3}$$

Specifically, B-splines of arbitrary scales can be expressed in terms of integer-scaled B-splines:

$$\beta_a^n(x) = \left(\Delta_a^{n+1}\circ\Delta_1^{-(n+1)}\right)\{\beta_1^n\}(x+\tau) \tag{4}$$

with $\tau := (a-1)(n+1)/2$, using the FD and RS operators [2]. It is also worth mentioning that $\beta_a^0(x) = 1/a$, for $x \in (-a/2, a/2]$ and 0 else; it is the piecewise-constant function $\mathrm{rect}(x/a)$. In the sequel, we simply denote it by $\beta_a(x)$.

## 3. SCALE-ADAPTIVE FILTERING

In this section, we revisit our 1D space-variant filter [2] based on the projections $s[m] = \langle f(x), \beta_a^{n_2}(x-m)\rangle, m \in \mathbb{Z}$, of a continuous signal model $f(x)$ of the discrete signal with scaled B-splines $\beta_a^{n_2}$. The scale $a$ controls the degree of smoothing applied around each sample.

Specifically, given the signal samples $\{f[k]\}_{k\in\mathbb{Z}}$, we consider the following B-spline model $f(x) = \sum_{k\in\mathbb{Z}} c[k]\beta^{n_1}(x-k)$ for the continuum, with the interpolation constraint $f(x)|_{x=k} = f[k], k \in \mathbb{Z}$. The expansion coefficients are then obtained by the digital filtering $c = f * (b^{n_1})^{-1}$, where $(b^{n_1})^{-1}$ is the convolution inverse of the B-spline interpolation filter $b^{n_1}[k] = \beta^{n_1}(k), k \in \mathbb{Z}$ [4]. It then turns out that the projections $\{s[m]\}_{m\in\mathbb{Z}}$ can be efficiently realized using (4):

**Proposition 3.1** *The B-spline projection $\{s[m]\}_{m\in\mathbb{Z}}$ can be computed in two steps:*

1. ***Non-Adaptive Step:*** *The running sum filter $u_1^{(n_2+1)}$ is applied to all the filtered sequence $\{c[k]\}_{k\in\mathbb{Z}}$ to get the integrated sequence $g[m] = \sum_{k\in\mathbb{Z}} u_1^{(n_2+1)}[k]c[m-k]$. Then, the continuous domain pre-integrated signal can be written as $F(x) = \sum_{k\in\mathbb{Z}} g[k]\beta^{n_1+n_2+1}(x+\tau-k)$.*
2. ***Adaptive Step:*** *At each sample position, $m \in \mathbb{Z}$, and corresponding to a specific scale $a$, the integrated sequence $g$ is then filtered using the FIR localization mask $w[k] := \Delta_a^{n_2+1}\{\beta^{n_1+n_2+1}\}(k+\tau), k \in \mathbb{Z}$ to get the local projection $s[m] = \Delta_a^{n_2+1}\{F\}(m) = \sum_{k\in\mathbb{Z}} g[k]w[m-k]$.*

## 4. RADIALLY-UNIFORM BOX SPLINES

We now extend these ideas to 2D, where the additional feature of directionality needs to be addressed. Particularly, we devise the following tensor product between a 1D smoothing operator and the identity convolution operator $\delta : \psi \mapsto \psi(0)$, both operating along orthogonal directions.

**Definition 4.1** *The generating kernel $\varphi_a$ of scale $a \in \mathbb{R}_+$ is defined as*

$$\varphi_a(\boldsymbol{x}) = \beta_a(x_1)\delta(x_2),\ \boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2 \tag{5}$$

The rotated versions $\varphi_{a,\theta}(\boldsymbol{x})$ of the generating kernel are then obtained as

$$\varphi_{a,\theta}(\boldsymbol{x}) = \boldsymbol{R}_\theta\{\varphi_a\}(\boldsymbol{x}) = \beta_a(\boldsymbol{r}_\theta^T\boldsymbol{x})\delta(\boldsymbol{r'}_\theta^T\boldsymbol{x}) \tag{6}$$

where $\boldsymbol{R}_\theta$ is the rotation operator defined as $\boldsymbol{R_\theta}\{f\}(\boldsymbol{x}) = f(R_\theta^T\boldsymbol{x})$ via the rotation matrix

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} =: \begin{pmatrix} \boldsymbol{r_\theta} & \boldsymbol{r'_\theta} \end{pmatrix} \tag{7}$$

The most elementary box spline, the $\mathrm{rect}(\boldsymbol{x})$ function, can be generated by convolving two quadrature tensor B-splines; specifically, $\mathrm{rect}(\boldsymbol{x}) = \varphi_{1,0}(\boldsymbol{x}) * \varphi_{1,\pi/2}(\boldsymbol{x})$ as per the above formalism. We generalize this idea to construct a family of box splines, which we call the *radially-uniform box splines*.

**Definition 4.2** *The radially-uniform box spline $\boldsymbol{\beta}_{\boldsymbol{a}}^N$ is defined as*

$$\boldsymbol{\beta}_{\boldsymbol{a}}^N = \varphi_{a_1,\theta_1} * \cdots * \varphi_{a_N,\theta_N} \tag{8}$$

*where $N \in \mathbb{N}, N \geqslant 2$ is the directional order, $\boldsymbol{a} = (a_1, \ldots, a_N) \in \mathbb{R}_+^N$ is the scale vector, and $\theta_k = (k-1)\pi/N, 1 \leqslant k \leqslant N$ are the rotation angles.*

Intuitively this means that we can construct 2D box splines by convolving arbitrary number of tensor B-splines, rescaled by arbitrary amounts but uniformly distributed radially. In retrospect, $\mathrm{rect}(\boldsymbol{x}) = \boldsymbol{\beta}_{(1,1)}^2$.

Importantly, the covariance (moment) of these smoothing kernels can be arbitrarily controlled by suitably choosing the order $N$ and the pairs $\{(a_k, \theta_k)\}_{k=1}^N$. The remarkable fact is that as the order increases, these box splines become more Gaussian-like. In particular, we have the following result:

**Theorem 4.3** *Let $\{\boldsymbol{\beta}_{\boldsymbol{a}(N)}^N\}_{N\geqslant 2}$ be a sequence of box splines corresponding to the the scale-vector sequence $\{\boldsymbol{a}(N)\}_{N\geqslant 2}$ with components given by $a_k(N) = \sigma\sqrt{24/N}, 1 \leqslant k \leqslant N$. Then we have the following convergence*

$$\lim_{N\to\infty} \boldsymbol{\beta}_{\boldsymbol{a}(N)}^N(\boldsymbol{x}) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{||\boldsymbol{x}||^2}{2\sigma^2}\right) \tag{9}$$

Yet another form of convergence is achievable based on the iterated convolution of the box spline $\boldsymbol{\beta}_{\boldsymbol{a}}^N$, of a specific order $N$, with itself. Based on the Central Limit Theorem, existence of a sequence of iterated box splines that converges to a Gaussian can also be demonstrated.

## 5. 2D LINEAR OPERATORS

We now extend the definitions of the FD (1) and the RS (2) operator to 2D; the key aspect is to preserve (4) in relation to the radially-uniform box-splines.

**Definition 5.1** *The FD operator $\Delta_{a,\theta}$ of scale $a \in \mathbb{R}_+$ operating in the direction $\theta \in [0, \pi)$ is specified by*

$$\Delta_{a,\theta}\{f\}(\boldsymbol{x}) = \frac{1}{a}\Big(f(\boldsymbol{x}) - f(\boldsymbol{x} - a\boldsymbol{r}_\theta)\Big), \boldsymbol{x} \in \mathbb{R}^2 \qquad (10)$$

The $N$-directional FD operator $\Delta_{\boldsymbol{a}}^N$ corresponding to the scale-vector $\boldsymbol{a} = (a_1, \dots, a_N)$ is then defined as the composition $\Delta_{\boldsymbol{a}}^N = \Delta_{a_1,\theta_1} \circ \cdots \circ \Delta_{a_N,\theta_N}$, where $\{\Delta_{a_k,\theta_k}\}_{k=1}^N$ are FD operators of scale $a_k \in \mathbb{R}_+$ operating in the direction $\theta_k = (k-1)\pi/N$, repectively.

**Definition 5.2** *The RS operator $\Delta_{b,\theta}^{-1}$ of scale $b \in \mathbb{R}_+$ operating in the direction $\theta \in [0, \pi)$ is given by*

$$\Delta_{b,\theta}^{-1}\{f\}(\boldsymbol{x}) = b\sum_{k=0}^{\infty} f(\boldsymbol{x} - kb\boldsymbol{r}_\theta),\ \boldsymbol{x} \in \mathbb{R}^2 \qquad (11)$$

The $N$-directional RS operator $\Delta_{\boldsymbol{b}}^{-N}$ corresponding to the scale-vector $\boldsymbol{b} = (b_1, \dots, b_N)$ is then defined as $\Delta_{\boldsymbol{b}}^{-N} = \Delta_{b_1,\theta_1}^{-1} \circ \cdots \circ \Delta_{b_N,\theta_N}^{-1}$, where $\{\Delta_{b_k,\theta_k}^{-1}\}_{k=1}^N$ are RS operators of scale $b_k$ operating in the direction $\theta_k = (k-1)\pi/N$, respectively.

Importantly, based on (11), the application of $\Delta_{\boldsymbol{b}}^{-N}$ on a discrete sequence $\{g[\boldsymbol{k}]\}_{\boldsymbol{k}\in\mathbb{Z}^2}$ is then (non-uniquely) given by

$$\Delta_{\boldsymbol{b}}^{-N}\{g\}[\boldsymbol{m}] = b_1 \sum_{k_1=0}^{\infty} \cdots b_N \sum_{k_N=0}^{\infty} g_{\text{int}}\Big(\boldsymbol{m} - \sum_{j=1}^{N} k_j b_j \boldsymbol{r}_{\theta_j}\Big) \qquad (12)$$

where $g_{\text{int}}(\boldsymbol{x})$ is some form of interpolation of the discrete sequence $g[\boldsymbol{k}]$. The good news is that for a specific choice of $N$ and $\boldsymbol{b}$, the operator $\Delta_{\boldsymbol{b}}^{-N}$ admits a convolution kernel (filter) and (12) then has a unique form; this motivates the following definition and subsequent simplification.

Let $0 \leqslant \theta < \pi$ and $b \in \mathbb{R}_+$ be such that $b\cos\theta, b\sin\theta \in \mathbb{Z}$. Then the RS filter $\{u_{b,\theta}[\boldsymbol{k}]\}_{\boldsymbol{k}\in\mathbb{Z}^2}$ of scale $b$ and acting in the direction $\theta$ is defined as

$$u_{b,\theta}[\boldsymbol{k}] = \begin{cases} b, & \text{for } \boldsymbol{k} \in (b\cos\theta, b\sin\theta)\mathbb{N}_0 \\ 0, & \text{else} \end{cases} \qquad (13)$$

The $N$-directional RS filter $u_{\boldsymbol{b}}^N[\boldsymbol{k}]$ corresponding to the integration scale-vector $\boldsymbol{b} = (b_1, \dots, b_N)$ is then defined as the 2D convolution $u_{\boldsymbol{b}}^N[\boldsymbol{k}] = (u_{b_1,\theta_1} * \cdots * u_{b_N,\theta_N})[\boldsymbol{k}]$ of the RS filters $\{u_{b_k,\theta_k}\}_{k=1}^N$ of scale $b_k$ and acting in the direction $\theta_k$; provided all the constituent filters are well-defined. Then (12) is uniquely given by $\Delta_{\boldsymbol{b}}^{-N}\{g\}[\boldsymbol{m}] = \sum_{\boldsymbol{k}\in\mathbb{Z}^2} u_{\boldsymbol{b}}^N[\boldsymbol{k}] g[\boldsymbol{m} - \boldsymbol{k}]$.

Importantly, as in the 1D case, it turns out that $\boldsymbol{\beta}_{\boldsymbol{a}}^N$ can be factorized using the above linear operators as follows

$$\boldsymbol{\beta}_{\boldsymbol{a}}^N(\boldsymbol{x}) = \big(\Delta_{\boldsymbol{a}}^N \circ \Delta_{\boldsymbol{b}}^{-N}\big)\{\boldsymbol{\beta}_{\boldsymbol{b}}^N\}(\boldsymbol{x} + \boldsymbol{\tau}) \qquad (14)$$

where $\boldsymbol{\tau} := 0.5\Big(\sum_{k=1}^N (a_k - b_k)\cos\theta_k, \sum_{k=1}^N (a_k - b_k)\sin\theta_k\Big)$ is the shift-vector. The role of the scale-vector $\boldsymbol{b}$ in (14) is to tie the RS operator $\Delta_{\boldsymbol{b}}^{-N}$ to the underlying lattice $\mathbb{Z}^2$, and will be used to advantage in the sequel.

## 6. SPACE-VARIANT ADAPTIVE FILTERING

Our goal is to adaptively filter the signal $f(\boldsymbol{x})$, obtained by interpolating the 2D signal samples $\{f[\boldsymbol{k}]\}_{\boldsymbol{k}\in\mathbb{Z}^2}$, with appropriately elongated and orientated box splines $\boldsymbol{\beta}_{\boldsymbol{a}}^N(\boldsymbol{x})$ by suitably adjusting the scale-vector $\boldsymbol{a}$ at the different spatial locations. In other words, given some pre-assigned scale-vector map $\boldsymbol{a} : \mathbb{Z}^2 \to \mathbb{R}_+^N$, we want to compute the projections $s[\boldsymbol{m}] := \langle f, \boldsymbol{\beta}_{\boldsymbol{a}(\boldsymbol{m})}^N(\cdot - \boldsymbol{m})\rangle$, at spatial locations $\boldsymbol{m} \in \mathbb{Z}^2$.

First, as in the 1D case, we consider the continuous representation $f(\boldsymbol{x}) = \sum c[\boldsymbol{k}]\ \phi(\boldsymbol{x} - \boldsymbol{k})$, where $\phi(\boldsymbol{x})$ is a 2D interpolating function. The expansion coefficients are given by $c = f * (b^n)^{-1}$, where $(b^n)^{-1}$ is the convolution inverse of the interpolation filter $b^n[\boldsymbol{k}] := \phi(\boldsymbol{k}), \boldsymbol{k} \in \mathbb{Z}^2$. Using (14), the desired projections can then be efficiently realized in two stages:

(1) **Non-Adaptive Step**: The entire image $f$ is filtered once using the fixed-scale RS filter to give the pre-integrated image:

$$g_{\boldsymbol{b}}[\boldsymbol{m}] = \begin{cases} \Delta_{\boldsymbol{b}}^{-N}\{c\}[\boldsymbol{m}], & \text{with interpolation} \\ (u_{\boldsymbol{b}}^N * c)[\boldsymbol{m}], & \text{without interpolation} \end{cases} \qquad (15)$$

This also gives us the continuous domain pre-integrated image $F(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^2} g_{\boldsymbol{b}}[\boldsymbol{k}](\phi * \boldsymbol{\beta}_{\boldsymbol{b}}^N)(\boldsymbol{x} + \boldsymbol{\tau} - \boldsymbol{k})$;

(2) **Adaptive Step**: The pre-integrated image $g_{\boldsymbol{b}}$ is then locally filtered using the pointwise localization operator $\Delta_{\boldsymbol{a}(\boldsymbol{m})}^N$ to give the adaptively filtered output

$$s[\boldsymbol{m}] = \Delta_{\boldsymbol{a}(\boldsymbol{m})}^N\{F\}(\boldsymbol{k}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^2} w_{\boldsymbol{a}(\boldsymbol{m})}[\boldsymbol{k}] g_{\boldsymbol{b}}[\boldsymbol{m} - \boldsymbol{k}] \qquad (16)$$

at each location $\boldsymbol{m} \in \mathbb{Z}^2$ corresponding to the scale-vector $\boldsymbol{a}(\boldsymbol{m})$, where $w_{\boldsymbol{a}(\boldsymbol{m})}[\boldsymbol{k}] := \Delta_{\boldsymbol{a}(\boldsymbol{m})}^N\{\phi * \boldsymbol{\beta}_{\boldsymbol{b}}^N\}(\boldsymbol{k} + \boldsymbol{\tau})$, $\boldsymbol{k} \in \mathbb{Z}^2$, is the localization mask, and $\boldsymbol{\tau} = 0.5(\sum_{k=1}^N (a_k(\boldsymbol{m}) - b_k)\cos\theta_k, \sum_{k=1}^N (a_k(\boldsymbol{m}) - b_k)\sin\theta_k)$ is the pointwise shift-vector.

## 7. FAST ELLIPTICAL FILTERING ALGORITHM

Finally, we focus on the family of four-directional box spline $\boldsymbol{\beta}_{\boldsymbol{a}}^4(\boldsymbol{x}) = (\varphi_{a_1,0} * \varphi_{a_2,\pi/4} * \varphi_{a_3,\pi/2} * \varphi_{a_4,3\pi/4})(\boldsymbol{x})$, corresponding to the directional order $N = 4$ and scale-vector

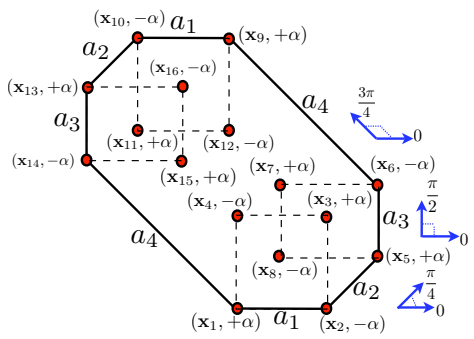


**Fig. 1**. **Affine Mesh Geometry: The pair $(\cdot,\cdot)$ denotes the position of a mesh vertex and the corresponding weight.**

$\boldsymbol{a} = (a_1, \ldots, a_4) \in \mathbb{R}^4_+$ in (8). It is interesting to note that $\boldsymbol{\beta}^4_{\boldsymbol{a}}(\boldsymbol{x})$, corresponding to $\boldsymbol{a} = (1, \sqrt{2}, 1, \sqrt{2})$, is also known as the ZP (Zwart-Powell) element [5] in box spline literature. Below, we outline the corresponding implementation aspects:
**(1) Non-Adaptive Step**: First, we simplify the pre-filtering by selecting the kernel $\phi(\boldsymbol{x}) = \delta(\boldsymbol{x})$, with the result that $c[\boldsymbol{k}] = f[\boldsymbol{k}]$. Further, the fact that (15) can be implemented (without interpolation) using the RS filter $u^4_{\boldsymbol{b}}[\boldsymbol{k}]$, with $\boldsymbol{b} = (1, \sqrt{2}, 1, \sqrt{2})$, further simplifies the computation. In particular, the pre-integrated image can be expressed as

$$g_{\boldsymbol{b}}[\boldsymbol{k}] = (u_{1,0} * u_{\sqrt{2},\pi/4} * u_{1,\pi/2} * u_{\sqrt{2},3\pi/4} * f)[\boldsymbol{k}] \quad (17)$$

Now, due to this tensor structure, (17) can be then efficiently implemented in a recursive fashion in four steps:
$(S1)$ $F_0 = u_{1,0} * f$, computed as
$F_0[k_1, k_2] = f[k_1, k_2] + F_0[k_1 - 1, k_2]$,
$(S2)$ $F_{\pi/4} = u_{\sqrt{2},\pi/4} * F_0$, computed as
$F_{\pi/4}[k_1, k_2] = \sqrt{2}F_0[k_1, k_2] + F_{\pi/4}[k_1 - 1, k_2 - 1]$,
$(S3)$ $F_{\pi/2} = u_{1,\pi/2} * F_{\pi/4}$, computed as
$F_{\pi/2}[k_1, k_2] = F_{\pi/4}[k_1, k_2] + F_{\pi/2}[k_1, k_2 - 1]$,
$(S4)$ $g_{\boldsymbol{b}} = u_{\sqrt{2},3\pi/4} * F_{\pi/2}$, computed as
$g_{\boldsymbol{b}}[k_1, k_2] = \sqrt{2}F_{\pi/2}[k_1, k_2] + g_{\boldsymbol{b}}[k_1 + 1, k_2 - 1]$;
**(2) Adaptive Step**: Finally, simplifying (16), we get

$$\boxed{s[\boldsymbol{m}] = \sum_{i=1}^{16} \mathfrak{h}[i] g_{\mathcal{I}}(\boldsymbol{m} + \boldsymbol{\tau} - \boldsymbol{x}_i)} \quad (18)$$

where $g_{\mathcal{I}}(\boldsymbol{x}) = \sum_{\boldsymbol{k}} g_{\boldsymbol{b}}[\boldsymbol{k}] \boldsymbol{\beta}^4_{\boldsymbol{b}}(\boldsymbol{x} - \boldsymbol{k})$, is the ZP interpolation [5] of the discrete sequence $g_{\boldsymbol{b}}[\boldsymbol{k}]$, and $\mathfrak{h}[i] = (-1)^{i+1}\alpha, 1 \leqslant i \leqslant 16$, are the non-zero weights of the affine FD mesh, with $\alpha := (a_1 a_2 a_3 a_4)^{-1}$.

The corresponding (relative) positions $\{\boldsymbol{x}_i\}_{i=1}^{16} \subset \mathbb{R}^2$ of the mesh vertices are shown in Table 1, with the convention $a'_j = a_j/\sqrt{2}$, for $j = 2, 4$. Figure 1 gives the spatial ordering of the mesh vertices. The shift-vector in (18) is specified as $\boldsymbol{\tau} = (\tau_1, \tau_2)$, where $\tau_1 = (\sqrt{2}a_1 + a_2 - a_4 - \sqrt{2})/2\sqrt{2}$ and $\tau_2 = (a_2 + \sqrt{2}a_3 + a_4 - 3\sqrt{2})/2\sqrt{2}$. Note that $\boldsymbol{\tau}, \{\mathfrak{h}[i]\}$ and $\{\boldsymbol{x}_i\}$ are in fact defined pointwise in (18) using the scale-vector map $\boldsymbol{a} : \mathbb{Z}^2 \to \mathbb{R}^4_+$ (cf. (16)); we dropped the pointwise index $\boldsymbol{m}$ just to simplify the equation.

**Table 1. Mesh Vertices**

| | |
|---|---|
| $\boldsymbol{x}_1 : (0, 0)$ | $\boldsymbol{x}_9 : (a_1 + a'_2 - a'_4, a_3 + a'_2 + a'_4)$ |
| $\boldsymbol{x}_2 : (a_1, 0)$ | $\boldsymbol{x}_{10} : (a'_2 - a'_4, a_3 + a'_2 + a'_4)$ |
| $\boldsymbol{x}_3 : (a_1, a_3)$ | $\boldsymbol{x}_{11} : (a'_2 - a'_4, a'_2 + a'_4)$ |
| $\boldsymbol{x}_4 : (0, a_3)$ | $\boldsymbol{x}_{12} : (a_1 + a'_2 - a'_4, a'_2 + a'_4)$ |
| $\boldsymbol{x}_5 : (a_1 + a'_2, a'_2)$ | $\boldsymbol{x}_{13} : (-a'_4, a_3 + a'_4)$ |
| $\boldsymbol{x}_6 : (a_1 + a'_2, a_3 + a'_2)$ | $\boldsymbol{x}_{14} : (-a'_4, a'_4)$ |
| $\boldsymbol{x}_7 : (a'_2, a_3 + a'_2)$ | $\boldsymbol{x}_{15} : (a_1 - a'_4, a'_4)$ |
| $\boldsymbol{x}_8 : (a'_2, a'_2)$ | $\boldsymbol{x}_{16} : (a_1 - a'_4, a_3 + a'_4)$ |

Note that the algorithm has a fixed computational cost per output pixel as the size of the support of the FD mask in (18) is independent of the scale-vector. Specifically, the number of non-null weights is $4 \times 4 = 16$, i.e., $4$ clusters of $4$ points each, as shown in Figure 1.

## 8. CONCLUSION

We presented a novel elliptical filtering algorithm with fixed computational cost per output pixel. Our main goal was to formalize the adaptive elliptical filtering strategy using box splines and propose a fast and practical implementation algorithm. Computation of the scale-vector map presents a separate challenge in itself and will be discussed elsewhere.

## 9. REFERENCES


[1] J.M. Geusebroek and A.W. M. Smeulders, "Fast anisotropic Gaussian filtering," *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 938–943, August 2003.

[2] A. Muñoz Barrutia, R. Ertlé, and M. Unser, "Continuous Wavelet Transform with arbitrary scales and $\mathcal{O}(N)$ complexity," *Signal Processing*, vol. 82, no. 5, pp. 749–757, May 2002.

[3] P. S. Heckbert, "Filtering by repeated integration," *International Conf. on Computer Graphics and Interactive Techniques*, vol. 20, no. 4, pp. 315–321, 1986.

[4] M. Unser, "Splines: A perfect fit for signal and image processing," *IEEE Signal Processing Magazine*, vol. 16, no. 6, pp. 22–38, 1999.

[5] P. B. Zwart, "Multivariate splines with nondegenerate partitions," *SIAM Journal on Numerical Analysis*, vol. 10, no. 4, pp. 665–673, 1973.