Fast High-Dimensional Kernel Filtering
Pravin Nair, Student Member, IEEE and Kunal N. Chaudhury, Senior Member, IEEE

Abstract—The bilateral and nonlocal means filters are instances of kernel-based filters that are popularly used in image processing. It was recently shown that fast and accurate bilateral filtering of grayscale images can be performed using a low-rank approximation of the kernel matrix. More specifically, based on the eigendecomposition of the kernel matrix, the overall filtering was approximated using spatial convolutions, for which efficient algorithms are available. Unfortunately, this technique cannot be scaled to high-dimensional data such as color and hyperspectral images. This is simply because one needs to compute/store a large matrix and perform its eigendecomposition in this case. We show how this problem can be solved using the Nyström method, which is generally used for approximating the eigendecomposition of large matrices. The resulting algorithm can also be used for nonlocal means filtering. We demonstrate the effectiveness of our proposal for bilateral and nonlocal means filtering of color and hyperspectral images. In particular, our method is shown to be competitive with state-of-the-art fast algorithms, and moreover it comes with a theoretical guarantee on the approximation error.

Index Terms—Kernel Filter, Nyström Method, Approximation, Fast Algorithm, Error Bound.

I. INTRODUCTION

The bilateral and nonlocal means filters [1], [2] are widely used for edge-preserving smoothing and denoising of images [3], [4]. These are instances of kernel filters, where the similarity (affinity) between pixels is measured using a symmetric kernel. We refer the reader to [4] for an excellent review of kernel filters. While they have proven to be useful in practice, a flip side of kernel filtering, including bilateral filtering (BLF) and nonlocal means (NLM), is their computational complexity [3]. Nevertheless, several fast algorithms have been proposed, e.g. [5]–[26], which can speed up BLF and NLM, without compromising their filtering quality. See [11], [21], [26] for a survey of these algorithms. Unfortunately, most algorithms only work with grayscale images, and cannot be extended to color, multispectral, and hyperspectral images.

Algorithms for fast BLF of color images have been proposed in [12], [21], [27]–[29]. However, to the best of our knowledge, these methods have not been extended for multispectral and hyperspectral images. Fast algorithms for generic high-dimensional BLF and NLM have been proposed in [18]–[20], [30]. A common feature of these algorithms is that they use data clustering or tessellation in high-dimensions. The state-of-the-art fast algorithms for color BLF are [19], [21], and for color NLM is [20].

More recently, it was shown in [15], [17] that fast BLF of grayscale images can be performed using the partial eigendecomposition of the kernel matrix. In fact, the interpretation of BLF (and NLM) as kernel filters goes back to [31]–[33]. While the Nyström method has widely been used in machine learning [34]–[36], it appears that [31] is the first to apply this for image filtering. Note that, unlike [15], [17], the spatial and range kernel are treated as a single kernel in [31]–[33].

The differences between our and related approaches are:
- As explained in detail in §II, it is difficult to scale [15], [17] for filtering high-dimensional (even color) images, since one needs to populate a huge kernel matrix and compute its eigendecomposition. We propose to use the Nyström method to solve this problem. As a result, we are able to perform BLF and NLM of color and hyperspectral images.
- The first difference with [31]–[33] is that we use clustering instead of uniform sampling for the Nyström approximation. A significant improvement in filtering accuracy is achieved as a result. The other difference is that if a spatial kernel has to be extended to high-dimensional case, then the Nyström approximation needs to be performed in the spatio-range space. However, we handle the spatial and range components differently—fast convolutions are used for the spatial component and Nyström approximation is used for the range component. As a result, we require lesser samples for the Nyström approximation.
- In [28], [29], clustering is used to compute “intermediate” images, which are interpolated to get the final output. On the other hand, clustering is used in our method just to obtain the “landmark points” for the Nyström approximation.
- Compared to [18]–[21], our algorithm is conceptually simple and easy to implement. Moreover, we are able to derive a bound on the filtering error incurred by the approximation. Such a guarantee is not offered by [18]–[21].

The rest of the paper is organized as follows. In §II, we introduce the notion of kernel filtering, and explain the core problem in relation to the spectral approximations in [15], [17]. We use the Nyström method in §III to overcome this problem. Numerical results are reported in §IV and we conclude in §V.

II. BACKGROUND

We begin by formulating BLF and NLM as kernel filters [4]. Suppose the input image is \( f : \Omega \rightarrow [0, R]^n \), where \( \Omega \subset \mathbb{Z}^d \) is the spatial domain, \([0, R]^n \) is the range space, and \( d \) (resp. \( n \)) is the dimension of the domain (resp. range). Let \( p : \Omega \rightarrow [0, R]^n \) be the guide image, which is used to control the filtering. For standard BLF, \( f \) and \( p \) are identical, and \( n = \rho = 1 \) and \( 3 \) for grayscale and color images. However, \( f \) and \( p \) (also \( n \) and \( \rho \)) can be different for joint BLF [3]. For NLM, \( \rho \) is generally larger than \( n \), where \( \rho \) is the number of pixels in a patch [2]. Let \( \kappa : \mathbb{R}^\rho \times \mathbb{R}^\rho \rightarrow \mathbb{R} \) be the range kernel. The filtered output \( g : \Omega \rightarrow [0, R]^n \) is given by

\[
g(x) = \frac{\sum_{y \in W_x} \omega(x-y)\kappa(p(x),p(y))f(y)}{\sum_{y \in W_x} \omega(x-y)\kappa(p(x),p(y))},
\]

where \( W_x \) is a square window around \( x \in \Omega \) consisting of \((2S+1)^d \) pixels, with \( S \) being the window radius. The spatial
kernel $\omega : \mathbb{Z}^d \to \mathbb{R}$ controls the weighting of the neighboring pixels involved in the averaging. At this point, we just assume that $\kappa$ is symmetric, i.e., $\kappa(t,s) = \kappa(s,t)$ for $t,s \in \mathbb{R}^p$. For example, $\kappa(t,s) = \exp(-\theta \|s - t\|^2)$, $\theta > 0$, for standard BLF and NLM, where $\|\cdot\|$ is the Euclidean norm.

It was shown in [15], [17] that the non-linear operations in (1) can be computed using convolutions by approximating $\kappa$. For convenience, we will describe this using our notations. Let the actual range of $p$ be

$$\mathcal{R} = \{p(x) : x \in \Omega\}. \quad (2)$$

We emphasize that $\mathcal{R}$ is a list and not a set, i.e., we allow repetition of elements in $\mathcal{R}$. In particular, let $\mathcal{R} = \{r_1, r_2, \ldots, r_m\}$ be some ordering of the elements in $\mathcal{R}$, where $m$ is the number of elements. This means that, given $\ell \in [1,m]$, $r_\ell = p(x)$ for some $x \in \Omega$. We track this correspondence using the index map $\iota : \Omega \rightarrow [1,m]$, where

$$\iota(x) = \ell \quad \text{if} \quad r_\ell = p(x). \quad (3)$$

We next define the kernel matrix $K \in \mathbb{R}^{m \times m}$ given by

$$K(i,j) = \kappa(r_i, r_j). \quad (4)$$

In terms of (4), we can write (1) as

$$g(x) = \sum_{y \in W_x} \omega(x - y)K(\iota(x), \iota(y)) f(y) \quad (5)$$

where $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$ are its eigenvalues, and $u_1, \ldots, u_m \in \mathbb{R}^m$ are the corresponding eigenvectors. Substituting (6) in (5), we can write its numerator as

$$\sum_{y \in W_x} \omega(x - y) \left\{ \sum_{k=1}^m \lambda_k u_k(\iota(x)) u_k(\iota(y)) \right\} f(y).$$

On switching the sums, this becomes

$$\sum_{k=1}^m \lambda_k u_k(\iota(x)) (\omega * h_k)(x), \quad (7)$$

where $\omega * h_k$ denotes the convolution of the image $h_k(x) = u_k(\iota(x)) f(x)$ with $\omega$. An identical argument applies for the denominator. In summary, we can compute (5) using convolutions, for which several efficient algorithms are available [37], [38]. Moreover, by considering just the largest eigenvalues, fast and accurate approximations can be obtained [15], [17].

Unfortunately, computing the full kernel and its eigendecomposition becomes prohibitively expensive when $p$ is large. Just as an example, consider an 8-bit color image for which $R = 255$ and $p = 3$. Even if we assume that $m$ is just 10% of the maximum range cardinality ($= 256^3$), we will still need to populate a 3 million $\times$ 3 million matrix, and compute its eigenvalues. The situation is worse for hyperspectral images, where $p$ is of the order of tens, or even hundreds.

### III. Proposed Method

Originally, the Nyström method was used for approximating the solution of functional eigenvalue problems [39], [40]. The method has found useful applications in machine learning and computer vision for approximating the eigendecomposition of large matrices [31], [34], [35]. In the present context, the goal is to approximate (6) using a decomposition of the form

$$\hat{K} = \sum_{k=1}^{m_0} \alpha_k v_k v_k^\top, \quad (8)$$

where $\alpha_k \in \mathbb{R}$ and $v_k \in \mathbb{R}^m$. Clearly, the rank of $\hat{K}$ is at most $m_0$. Thus, for small $m_0$, $\hat{K}$ is a low-rank approximation of $K$. A large $m_0$ results in a better approximation, but at higher computational cost. In practice, a good tradeoff is required.

The original kernel $K$ is defined on $\mathcal{R}$. In the Nyström method [39], [40], we first construct a smaller kernel $A$, compute its eigendecomposition, and then “extrapolate” the eigenvectors of $A$ to approximate those of $K$. More precisely, we pick few landmarks points from $\mathcal{R}$, say, $\mathcal{R}_0 = \{\mu_1, \ldots, \mu_{m_0}\}$, and define a kernel $A \in \mathbb{R}^{m_0 \times m_0}$ on $\mathcal{R}_0$:

$$A(i,j) = \kappa(\mu_i, \mu_j) \quad (i,j \in [1,m_0]). \quad (9)$$

Clearly, $A$ is symmetric, and its size is much smaller than $K$. Thus, we can efficiently compute its eigendecomposition:

$$A = \sum_{k=1}^{m_0} \alpha_k w_k w_k^\top, \quad (10)$$

where $\alpha_k \in \mathbb{R}$ and $w_k \in \mathbb{R}^{m_0}$. We next construct $B \in \mathbb{R}^{m_0 \times m}$ on $\mathcal{R}_0 \times \mathcal{R}$ given by

$$B(i,j) = \kappa(\mu_i, r_j), \quad (11)$$

where $i \in [1,m_0]$ and $j \in [1,m]$. This captures the kernel values between the points in $\mathcal{R}$ and the landmark points. This matrix is used to extrapolate $w_k$ as follows:

$$v_k = \frac{1}{\alpha_k} B^\top w_k \quad (k \in [1,m_0]). \quad (12)$$

This completes the specification of $\alpha_k$ and $v_k$ in (8). We refer the reader to [35] for the intuition behind the approximation. The effective speedup of replacing (6) by (8) is $O(m/m_0)^3$. This is because the complexity of eigendecomposition of a $k \times k$ matrix is $O(k^3)$ [41]. In particular, the speedup is significant since $m_0 \ll m$. As will be evident shortly, we just need to compute $\langle \alpha_k \rangle$ and $\langle v_k \rangle$; we will not use $K$ explicitly.

Following [36], we select the landmark points by clustering $\mathcal{R}$. More specifically, we partition $\mathcal{R}_0$ into $m_0$ disjoint sets using $k$-means clustering, and take the centroids to be the landmarks. Note that, though $\mathcal{R}_0$ is not guaranteed to be a subset of $\mathcal{R}$, we can still apply the above approximation.

It was shown in [36] that the kernel error can be bounded by the quantization error. More specifically, let $\|K - \hat{K}\|_F$ be the kernel error ($\|\cdot\|_F$ is the Frobenius norm), and let

$$e = \sum_{i=1}^m \|r_i - \mu_{c(i)}\|^2$$

be the quantization error, where $c(i)$ is the minimizer of $\|r_i - \mu_j\|$ over $j \in [1,m_0]$. Then the following bound holds [36].
Proposition 1: Suppose there exists some $L > 0$ such that, for $w, x, y, z \in \mathbb{R}$,
\[
(\kappa(x, y) - \kappa(w, z))^2 \leq L(\|x - w\|^2 + \|y - z\|^2).
\]
Then the approximation error can be bounded as
\[
\|\hat{K} - \hat{K}\| = c_1 \sqrt{e} + c_2 e,
\]
where the positive constants $c_1$ and $c_2$ do not depend on $e$. In particular, (13) holds when $\hat{K}$ is a Gaussian.

Proposition 1 suggests that we can reduce the kernel error by making $e$ small. However, $e$ measures how well $\Theta$ is represented by the landmark points. Following this observation, $k$-means clustering was used in [36] for determining the landmarks. It was empirically shown in [36] that clustering indeed results in smaller error over uniform sampling [31], [35]. We will see that this is also true for our algorithm.

We arrive at a fast algorithm by replacing $\hat{K}$ by $\hat{K}$. It is clear from (7) that the resulting approximation is given by
\[
\hat{g}(x) = \frac{1}{\hat{\eta}(x)} \sum_{k=1}^{m_0} \alpha_k v_k(i(x)) (\omega * h_k)(x),
\]
\[
\hat{\eta}(x) = \sum_{k=1}^{m_0} \alpha_k v_k(i(x)) (\omega * d_k)(x),
\]
where $d_k : \Omega \rightarrow \mathbb{R}$ and $h_k : \Omega \rightarrow \mathbb{R}^n$ are defined as $d_k(x) = v_k(i(x))$ and $h_k(x) = d_k(x)f(x)$.

The computation of (14) and (15) involves $(n + 1)m_0$ convolutions, since for each $k \in [1, m_0]$, there are $n$ convolutions in (14) and one in (15). The main point is that we have been able to express the non-linear kernel filter using convolutions, for which efficient algorithms are available. In particular, (14) and (15) can be performed using $O(1)$ operations (w.r.t. the size of the spatial kernel), when $\omega$ is a box or Gaussian [37], [38], [42]. The overall algorithm is described in Algorithm 1 (source code in [43]), where the symbols $\oplus$, $\otimes$, and $\odot$ are used to denote pixelwise addition, multiplication, and division.

The complexity of $k$-means clustering and the eigendecomposition of $A$ are $O(|\Omega|m_0p)$ [44] and $O(m_0^3)$ [41]. On the other hand, the complexity of the convolutions in (14) and (15) is $O(|\Omega|m_0(n + p))$, where $|\Omega|$ is the number of pixels. Since the complexity of the brute-force implementation is $O(|\Omega|^{2S+1}n^{n+p})$ [3], and convolutions are the dominant operations in our algorithm, we obtain an effective speedup of $(2S + 1)^n/m_0$. This is significant as $S$ is typically large [3].

We now comment on the filtering accuracy, namely, how well is (1) approximated by (14). Intuitively, we expect the approximation to be accurate if $\hat{K} \approx K$. In fact, since the difference $\|\hat{K} - K\|_F$ is controlled by the quantization error (Proposition 1), we have the following result.

Theorem 2: Suppose $\omega$ and $\kappa$ are positive, and $\kappa$ satisfies the property in Proposition 1. Then
\[
\|\hat{g} - g\|_\infty = \max_{x \in \Omega} \|\hat{g}(x) - g(x)\| \leq C_1 \sqrt{e} + C_2 e,
\]
where $C_1, C_2 > 0$ do not depend on $e$.

The main steps of the derivation are given in the supplement. Theorem 2 is true for BLF and NLM, where $\kappa$ is a Gaussian. A practical implication of this result is that the filtering accuracy is guaranteed to increase with $m_0$ (Figure 4 in the supplement). Deriving a similar bound is difficult for [18]–[21].

IV. RESULTS

We demonstrate the effectiveness of our algorithm for BLF and NLM of high-dimensional images by comparing it with state-of-the-art algorithms. Instead of standard NLM [2], we have used PCA-NLM [45], where the denoising performance of the former is improved by applying PCA on the collection of patches. As for the dataset, we have used the color images from [46] and the hyperspectral images from [47]. Experiments were performed using Matlab on a 3.4 GHz quad-core machine with 32 GB memory. The spatial kernel $\omega$ for BLF is a Gaussian (covariance $\sigma^2 I$ and $S = 3\sigma$), while it is a box in PCA-NLM. The range kernel $\kappa$ is Gaussian (covariance $\sigma^2 I$) for both BLF and PCA-NLM. We have used the fast $O(1)$ algorithm in [37] when $\omega$ is a Gaussian, and the Matlab routine “imfilter” when $\omega$ is a box. Note that we can also use other fast Gaussian filters [38], [42] if higher accuracy is desired.

Color BLF: The state-of-the-art fast algorithms for color BLF are Adaptive Manifolds (AM) [20], Permutohedral Lattice (PL) [19], and Global Color Sparseness (GCS) [21]. We have compared with them in Figure 2. The number of manifolds is set automatically in AM, whereas we have used 15 clusters in GCS and for the Nyström approximation. Following [20], [21], we used PSNR to measure the error between the brute-force and fast implementations. In Figure 2, notice that while our PSNR marginally exceeds that of GCS, it is however much better than PL and AM. Also notice the significant acceleration over the brute-force implementation obtained using our algorithm. We have also provided a table comparing the different methods on the Kodak dataset [46] in the supplement. The table shows that our method is better than GCS and PL when $\theta > 0$. As claimed in the introduction, we can see from the table that clustering provides a significant boost in filtering accuracy (10-20 dB) over uniform sampling.
Fig. 1. Gaussian denoising (noise level 25/255) of a color image using (b) PCA-NLM, its fast approximations (c) and (d), and (e) BM3D. The respective (Timing (sec), PSNR (dB), SSIM) is shown in the caption.

(a) Input (256 × 256). (b) Ours (108, 48.4). (c) [21] (107, 46.5).

(d) Brute-force, 4 min. (e) [20], (212, 38.7). (f) [19], (44.5).

Fig. 2. Visual comparison for fast BLF at $\sigma = 5$, $\theta = 50$, and $|\Omega_0| = (6\sigma^2+1)^2$. The (timing, PSNR) are mentioned, where timing is in milliseconds and PSNR is in dB. Timing is not mentioned for [19] which is implemented in C++. The breakup of timing for the proposed method is as follows: clustering (11 ms), eigendecomposition (1 ms), and convolutions (96 ms). Note that the overall timing is dominated by the convolutions.

Color NLM. AM is the state-of-the-art fast algorithm for color NLM (and PCA-NLM). In NLM, $\rho = 3(2r+1)^2$, where $r$ is the patch radius [2]. On the other hand, $\rho$ is reduced to a smaller value in PCA-NLM using PCA. Following [45], we set $\theta$ to be three times the noise level for all experiments. Denoising results are shown in Figure 1, where $S = 10$ and $r = 3$. For (b), (c), and (d), PCA was used to reduce the range dimension from $3 \times 7^2$ to 25. We used 31 clusters (resp. manifolds) for the Nyström approximation (resp. AM). Following [50], we measured the denoising performance using PSNR and SSIM (between the clean and denoised images). Note that we are superior to AM both in terms of accuracy and timing. Importantly, our PSNR is close to PCA-NLM (the method being approximated), but we are about 160× faster. In comparison with BM3D [51], our PSNR is 3 dB less. However, our timing is about half that of BM3D, since our complexity is much less than that of BM3D. Additional visual comparisons and accuracy analysis is provided in the supplement.

Hyperspectral BLF. Finally, we present a denoising result for a hyperspectral image of size $(610 \times 340) \times 103$ bands using BLF ($\sigma = 3, \theta = 100$). We have also compared with state-of-the-art methods for hyperspectral denoising [48], [49], whose parameters have been tuned accordingly. The results are shown in Figure 3. We have used $m_0 = 32$ landmarks for the Nyström approximation. As a standard practice, the PSNR and SSIM values are averaged over the spectral bands. Notice that our method can restore details better, which results in higher PSNR/SSIM values. In particular, the color is not satisfactorily restored in [48] and grains can be seen in [49]. Being a one-shot method, we are much faster than [48], [49].

V. CONCLUSION

We showed that fast bilateral and nonlocal means filtering of high-dimensional images can be performed using the Nyström approximation. The proposed algorithm can significantly accelerate the brute-force implementation of these filters, without compromising the visual quality. In particular, our algorithm is often competitive with state-of-the-art fast algorithms, and comes with provable guarantee on the filtering accuracy.
REFERENCES

[1] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” Proc. IEEE International Conference on Computer Vision, pp. 839–846, 1998.

[2] A. Baudes, B. Coll, and J.-M. Morel, “A non-local algorithm for image denoising,” Proc. IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 60–65, 2005.

[3] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, “Bilateral filtering: Theory and Applications,” Foundations and Trends® in Computer Graphics and Vision, vol. 4, no. 1, pp. 1–73, 2009.

[4] P. Milanfar, “A tour of modern image filtering: New insights and methods, both practical and theoretical,” IEEE Signal Processing Magazine, vol. 30, no. 1, pp. 106–128, 2013.

[5] F. Durand and J. Dorsey, “Fast bilateral filtering for the display of high-dimensional images,” ACM Transactions on Graphics, vol. 21, no. 3, pp. 257–266, 2002.

[6] S. Paris and F. Durand, “A fast approximation of the bilateral filter using a signal processing approach,” Proc. European Conference on Computer Vision, pp. 568–580, 2006.

[7] J. Chen, S. Paris, and F. Durand, “Real-time edge-aware image processing with the bilateral grid,” ACM Transactions on Graphics, vol. 26, no. 3, p. 103, 2007.

[8] F. Porikli, “Constant time O(1) bilateral filtering,” Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8, 2008.

[9] Q. Yang, K. H. Tan, and N. Ahuja, “Real-time O(1) bilateral filtering,” Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 557–564, 2009.

[10] K. N. Chaudhury, D. Sage, and M. Unser, “Fast O(1) bilateral filtering using trigonometric range kernels,” IEEE Transactions on Image Processing, vol. 20, no. 12, pp. 3376–3382, 2011.

[11] K. Sugimoto and S. I. Kamata, “Compressive bilateral filtering,” IEEE Transactions on Image Processing, vol. 24, no. 11, pp. 3357–3369, 2015.

[12] Q. Yang, N. Ahuja, and K.-H. Tan, “Constant time median and bilateral filtering,” International Journal of Computer Vision, vol. 112, no. 3, pp. 307–318, 2015.

[13] K. N. Chaudhury and S. D. Dabhade, “Fast and provably accurate bilateral filtering,” IEEE Transactions on Image Processing, vol. 25, no. 6, pp. 2519–2528, 2016.

[14] S. Ghosh and K. N. Chaudhury, “On fast bilateral filtering using Fourier kernels,” IEEE Signal Processing Letters, vol. 23, no. 5, pp. 570–573, 2016.

[15] K. Sugimoto, T. Breeckon, and S. I. Kamata, “Constant-time bilateral filter using spectral decomposition,” Proc. IEEE International Conference on Image Processing, pp. 3319–3323, 2016.

[16] E. J. Nyström, “Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben,” Acta Mathematica, vol. 54, no. 1, pp. 185–204, 1930.

[17] R. Deriche, “The numerical treatment of integral equations,” Clarendon Press, 1977.

[18] F. Porikli, “Constant time O(1) bilateral filtering,” Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8, 2008.

[19] J. Chen, S. Paris, and F. Durand, “Real-time edge-aware image processing with the bilateral grid,” ACM Transactions on Graphics, vol. 26, no. 3, p. 103, 2007.

[20] P. Nair, A. Popli, and K. N. Chaudhury, “A fast approximation of the bilateral filter using the discrete Fourier transform,” Journal of Electronic Imaging, vol. 25, no. 2, p. 023026, 2016.

[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, 2004.

[22] C. Williams and M. Seeger, “Using the Nyström method to speed up kernel machines,” Proc. Neural Information Processing Systems, pp. 682–688, 2001.

[23] R. K. Baker, “The夥 numerical treatmen of integral equations,” Clarendon Press, 1977.

[24] V. Y. Pan and Z. Q. Chen, “The complexity of the matrix eigenproblem,” Proc. ACM Symposium on Theory of Computing, pp. 507–516, 1999.

[25] E. J. Nyström, “Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben,” Acta Mathematica, vol. 54, no. 1, pp. 185–204, 1930.

[26] R. Deriche, “Recurursively implementing the gaussian kernel,” Signal Processing, vol. 44, no. 2, pp. 139–151, 1995.

[27] K. Sugimoto and S. I. Kamata, “Fast gaussian filter with second-order shift properties,” Proc. IEEE International Conference on Image Processing, pp. 514–518, 2013.