

Fast and Provably Accurate Bilateral Filtering

Kunal N. Chaudhury, *Senior Member, IEEE*, and Swapnil D. Dabhade

Abstract—The bilateral filter is a non-linear filter that uses a range filter along with a spatial filter to perform edge-preserving smoothing of images. A direct computation of the bilateral filter requires $O(S)$ operations per pixel, where S is the size of the support of the spatial filter. In this paper, we present a fast and provably accurate algorithm for approximating the bilateral filter when the range kernel is Gaussian. In particular, for box and Gaussian spatial filters, the proposed algorithm can cut down the complexity to $O(1)$ per pixel for any arbitrary S . The algorithm has a simple implementation involving $N + 1$ spatial filterings, where N is the approximation order. We give a detailed analysis of the filtering accuracy that can be achieved by the proposed approximation in relation to the target bilateral filter. This allows us to estimate the order N required to obtain a given accuracy. We also present comprehensive numerical results to demonstrate that the proposed algorithm is competitive with state-of-the-art methods in terms of speed and accuracy.

Index Terms—Edge-preserving smoothing, bilateral filter, kernel, approximation, fast algorithm, error analysis, bounds.

I. INTRODUCTION

Gaussian and box filters typically work well in applications where the amount of smoothing required is small. For example, they are quite effective in removing small dosages of noise from natural images. However, when the noise floor is large, and one is required to average more pixels to suppress the noise, these filters begin to over-smooth sharp image features such as edges and corners. The over-smoothing can, however, be alleviated using some form of data-driven (non-linear) diffusion, where the quantum of smoothing is controlled using the image features. A classical example in this regard is the famous PDE-based diffusion of Perona and Malik [2]. The bilateral filter was proposed by Tomasi and Maduchi [3] as a filtering-based alternative to the Perona-Malik diffusion. The bilateral filter has turned out to be a versatile tool that has found widespread applications in image processing, computer graphics, computer vision, and computational photography [4]. More recently, the bilateral filter has received renewed attention in the context of image denoising [5], [6].

In this paper, we consider a standard form of the bilateral filter where a Gaussian kernel is used for range filtering, and a box or Gaussian kernel is used for spatial filtering [3]. In this setting, the bilateral filtering of an image $\{f(i) : i \in I\}$, where I is some finite rectangular domain of \mathbb{Z}^2 , is given by

$$f_{\text{BF}}(i) = \frac{\sum_{j \in \Omega} w(j) g_{\sigma_r}(f(i) - f(j)) f(i) f(j)}{\sum_{j \in \Omega} w(j) g_{\sigma_r}(f(i) - f(j))} \quad (1)$$

where

$$g_{\sigma_r}(t) = \exp\left(-\frac{t^2}{2\sigma_r^2}\right). \quad (2)$$

The spatial filter is a Gaussian:

$$w(i) = \exp\left(-\frac{\|i\|^2}{2\sigma_s^2}\right) \quad (i \in \Omega), \quad (3)$$

or a box:

$$w(i) = 1/|\Omega| \quad (i \in \Omega). \quad (4)$$

The domain Ω of the spatial kernel is a square neighbourhood, $\Omega = [-W, W] \times [-W, W]$, where $W = 3\sigma_s$ for the Gaussian filter. We refer the interested reader to [3], [4] for a detailed exposition on the working of the filter. We note that the bilateral filter has a straightforward extension to video and volume data. Another natural extension is the cross (or joint) bilateral filter [4]. While we will limit our discussion to the standard bilateral filter, the main ideas in this paper can also be applied to the above-mentioned extensions.

A. Fast Bilateral Filtering

It is clear that a direct computation of (1) requires $O(W^2)$ operations per pixel. In fact, the computation is slow for practical settings of W . To address this issue, researchers have come up with several fast algorithms [7] - [14]. Most of these are based on some form of approximation, and provide various levels of compromise between speed and quality of approximation. One of the early algorithms for fast bilateral filtering involved the quantization of the image intensities, where the final output was obtained via the interpolation of the output of a set of linear filters [7]. It was later shown that this approximation can be used to obtain a constant-time implementation which further improves its speed [8]. In a different direction, it was observed in [9] that the bilateral filter can be conceived as a linear filter acting in three-dimensions, where the three-dimensions are obtained by augmenting the image intensity to the spatial dimensions. This observation was used to derive a fast filtering in three-dimensions, which was then sampled to obtain the final output. We refer the interested reader to [10] for a survey of fast algorithms for bilateral filtering.

The algorithms in [10], [11], [12] are particularly relevant to the present work. Here the authors proceed by approximating (2) using polynomial and trigonometric functions, and demonstrate how the bilateral filter can be decomposed into a series of spatial filterings as result. As is well-known, since spatial box and Gaussian filters can be implemented in constant-time using separability and recursion [13], the overall approximation can therefore be computed in constant-time.

B. Present Contribution

We propose a fast algorithm for computing (1) which was motivated by the line of work in [12], [14]. In particular,

A part of this work was presented at IEEE ICIP 2015 [1]. This work was supported in part by the Startup Grant awarded by the Indian Institute of Science. Address: Department of Electrical Engineering, Indian Institute of Science, Bangalore 560012, India. Correspondence: kunal@ee.iisc.ernet.in.

similar to these papers, we present a novel approximation of (2) that allows us to decompose the bilateral filter into a series of spatial convolutions. The fundamental difference between the above papers and the present approach is that, instead of approximating (2) and then translating the approximation in range space, we directly approximate the translated Gaussians appearing in (1). In particular, the computational advantages obtained using the proposed approximation are the following:

(1) For a fixed approximation order (to be defined shortly), the proposed approximation requires half the number of spatial filterings required by the approximations in [8], [10], [12].

(2) The proposed approximation does not involve the transcendental functions $\cos(\omega x)$ and $\sin(\omega x)$ which are used in [12], [14]. It only involves polynomials (and just a single Gaussian), and hence can be efficiently implemented on hardware [15]. Moreover, the rounding error is small when working with polynomials.

As will be demonstrated shortly, the proposed algorithm is generally faster and more accurate than Yang's algorithm [8], which is currently considered to be the state-of-the-art [10], [16]. In particular, we perform an error analysis whereby we compare the output obtained using the proposed algorithm with that of the exact bilateral filter. Due to the particular nature of the proposed approximation, our analysis is much more simple than that carried out for Yang's algorithm in [16]. Nevertheless, compared to Yang's algorithm, we are able to establish a smaller bound on the number of spatial filterings required to achieve a given filtering accuracy. The latter is defined in terms of the error between the outputs of the bilateral filter and the fast algorithm (this will be made precise in Section III). To best of our knowledge, with the exception of [8], this is the only fast algorithm that comes with a provable guarantee on the quality of approximation. At this point, we note that the term "accurate" is used in the paper not just to signify that the output of the fast algorithm is visibly close to that of the target bilateral filter. It also has a precise technical meaning, namely, that we can control the approximation order to make the error between the outputs of the bilateral filter and the fast algorithm arbitrarily small.

C. Organization

The rest of the paper is organized as follows. We present the proposed kernel approximation and the error analysis in Section II. In Section III, we develop a fast constant-time algorithm arising from the Gaussian-polynomial approximation. We then analyze the quality of approximation that can be achieved using our algorithm. This gives us a simple rule for tuning the approximation order for a given user-defined accuracy. We present exhaustive numerical results in Section IV, and demonstrate the superior performance of the proposed algorithm over some of the existing algorithms.

II. GAUSSIAN-POLYNOMIAL APPROXIMATION

The present idea is to consider the translated kernel $g_{\sigma_r}(t - \tau)$ that appears in (1), where $t = f(1 - j)$ and $\tau = f(1)$. We can write

$$g_{\sigma_r}(t - \tau) = \exp\left(-\frac{\tau^2}{2\sigma_r^2}\right) \exp\left(-\frac{t^2}{2\sigma_r^2}\right) \exp\left(\frac{\tau t}{\sigma_r^2}\right). \quad (5)$$

For a fixed translation τ , this is a function of t . Notice that the first term is simply a scaling factor, while the second term is a Gaussian centered at the origin. In fact, the second term essentially contributes to the bell shape of the translated Gaussian. The third term is a monotonic exponential, which is increasing or decreasing depending on the sign of τ ; this term helps in translating the Gaussian to $t = \tau$.

We assume (without loss of generality, as will be explained at the start of Section III) that the dynamic range of the image is $[-T, T]$. That is, the arguments $t = f(1 - j)$ and $\tau = f(1)$ in (5) take values in $[-T, T]$. This means that the product τt appearing in (5) takes values in $[-T^2, T^2]$. Consider the Taylor expansion of the exponential term about the origin:

$$\exp\left(\frac{\tau t}{\sigma_r^2}\right) = \sum_{n=0}^{N-1} \frac{1}{n!} \left(\frac{\tau t}{\sigma_r^2}\right)^n + \text{higher-order terms}. \quad (6)$$

By dropping the higher-order terms, we obtain the following approximation of (5):

$$\phi_{N,\sigma_r}(t, \tau) = \exp\left(-\frac{t^2 + \tau^2}{2\sigma_r^2}\right) \left[\sum_{n=0}^{N-1} \frac{1}{n!} \left(\frac{\tau t}{\sigma_r^2}\right)^n \right]. \quad (7)$$

Being the product of a bivariate Gaussian and a polynomial, we will henceforth refer to (7) as a *Gaussian-polynomial*, where N is its approximation *order*. By construction, we have the pointwise convergence

$$\lim_{N \rightarrow \infty} \phi_{N,\sigma_r}(t, \tau) = g_{\sigma_r}(t - \tau). \quad (8)$$

We would like to note that the above idea of splitting the kernel and approximating a part of its using Taylor polynomials was employed in [17] in the context of the fast Gauss transform. To the best of our knowledge, this idea has not been exploited for fast bilateral filtering along the lines of the present work.

In Figure 1, we study the approximations corresponding to different N . The fundamental difference between the Taylor approximation in [11] and the Gaussian-polynomial approximation (8) is that instead of approximating the entire Gaussian, we approximate one of its component, namely the exponential function in (5). The intuition behind this is that the Taylor polynomial blows up as one moves away from the origin. This makes it difficult to approximate the tail part of a Gaussian using such polynomials. On the other hand, the exponential in (5) is monotonic, and hence can be closely approximated using polynomials. This point is explained with an example in Figure 2. In particular, notice in Figure 2b that the Gaussian-polynomial approximation is quite precise over the range of interest, and is comparable to the raised-cosine approximation of same order [12].

A. Quantitative Error Analysis

Before explaining how we can use Gaussian-polynomials to derive a fast bilateral filter in Section III, we study the kernel error incurred by approximating (2) using Gaussian-polynomials. We will see in Section III that a bound on the kernel error can in turn be used to bound the filtering accuracy of the fast algorithm. Note that (8) tells us that Gaussian-polynomial can be used to approximate the range kernel with

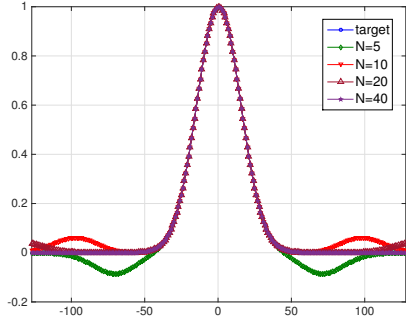


Fig. 1. Approximation of $g_{30}(t-\tau)$ using Gaussian-polynomials $\phi_{N,30}(t, \tau)$ with different N . The bivariate functions $g_{30}(t-\tau)$ and $\phi_{N,30}(t, \tau)$ have been sampled along $t = -\tau$ to generate a one-dimensional profile.

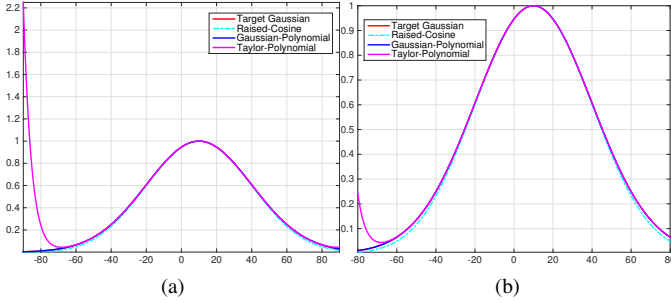


Fig. 2. Comparison of the approximations of $g_{30}(t-10)$ using raised-cosine [12], Taylor polynomial [11], and Gaussian-polynomial of order 10. We notice in (a) that the Taylor polynomial quickly goes off to $+\infty$ as one moves away from the origin. For this reason, we restricted the plot to $[-90, 90]$, although the desired approximation range is the full dynamic range $[-128, 128]$. The plots over $[-80, 80]$ are separately provided in (b) for comparing the raised-cosine and the Gaussian-polynomial approximations with the target Gaussian.

arbitrary accuracy. However, in practice, we will be required to use a Gaussian-polynomial of some fixed order N . A relevant question is the size of error incurred for a given N ? A related question is that, given some error margin $\varepsilon > 0$, how do we fix the smallest N such that the corresponding error is within ε ?

To begin with, we define the error function

$$E_{N,\sigma_r}(t, \tau) = g_{\sigma_r}(t - \tau) - \phi_{N,\sigma_r}(t, \tau) = \exp\left(-\frac{t^2 + \tau^2}{2\sigma_r^2}\right) \sum_{n=N}^{\infty} \frac{1}{n!} \left(\frac{\tau t}{\sigma_r^2}\right)^n. \quad (9)$$

The mathematical problem is one of bounding (9) for fixed N and σ_r . In this work, we consider the ℓ_∞ error given by

$$\|E_{N,\sigma_r}\|_\infty = \max \left\{ |E_{N,\sigma_r}(t, \tau)| : -T \leq t, \tau \leq T \right\}. \quad (10)$$

This is also referred to as the worst-case or uniform error. We note that one can measure the error using other means, e.g., using the ℓ_2 metric. The reason why we choose the ℓ_∞ metric is that our ultimate goal is to quantify the ℓ_∞ accuracy of the final filtering arising from the approximation, and a bound on (10) is sufficient for this purpose. Moreover, computing the ℓ_∞ error is relatively simple.

Using the inequality $(t^2 + \tau^2)/2 \geq |\tau t|$, we can bound the first term in (9) by $\exp(-|\tau t|/\sigma_r^2)$. Therefore, we have

$$\|E_{N,\sigma_r}\|_\infty \leq \max_{s \in [0, T^2]} \psi_{N,\sigma_r}(s), \quad (11)$$

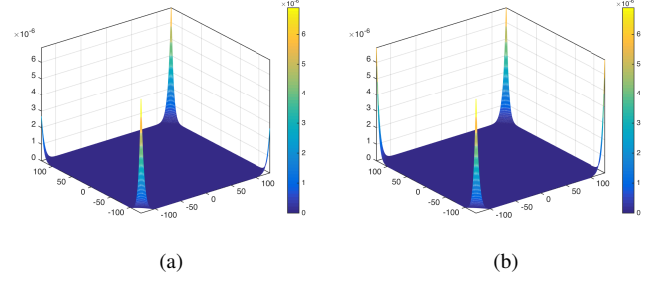


Fig. 3. Comparison of the actual error (9) and the bound in (12) for $T = 128$ and $\sigma_r = 30$. We plot the samples of the error function $E_{40,30}(t, \tau)$ over the square domain $-128 \leq t, \tau \leq 128$ in (a). We compare this with the samples of (12) over the same domain in (b), where we have set $s = |\tau t|$. Notice that the supremum of either plots are of the same order of magnitude.

where

$$\psi_{N,\sigma_r}(s) = \exp\left(-\frac{s}{\sigma_r^2}\right) \left[\sum_{n=N}^{\infty} \frac{1}{n!} \left(\frac{s}{\sigma_r^2}\right)^n \right]. \quad (12)$$

Using (11), we obtain the following result. We note that this bound is stronger than that derived for the fast Gauss transform in [17].

Proposition II.1.

$$\|E_{N,\sigma_r}\|_\infty \leq \sum_{n=N}^{\infty} \frac{e^{-\lambda} \lambda^n}{n!} \quad (\lambda = T^2/\sigma_r^2). \quad (13)$$

To arrive at (13), we proceed by writing (12) as

$$\psi_{N,\sigma_r}(s) = 1 - \exp\left(-\frac{s}{\sigma_r^2}\right) \sum_{n=0}^{N-1} \frac{1}{n!} \left(\frac{s}{\sigma_r^2}\right)^n.$$

After differentiation, we get

$$\psi'_{N,\sigma_r}(s) = \frac{1}{(N-1)! \sigma_r^2} \left(\frac{s}{\sigma_r^2}\right)^{N-1} \exp\left(-\frac{s}{\sigma_r^2}\right) \geq 0.$$

Thus, (12) is non-decreasing on $[0, T^2]$, whereby we conclude that the maximum in (11) is attained at $s = T^2$. This establishes Proposition II.1.

To get an idea of the tightness of the bound in (13), we compare the mesh plots of (9) and (12) in Figure 3 when $\sigma_r = 30$ and $N = 40$. While there is a gap between the error and the corresponding bound at certain values of (t, τ) , the supremum of the latter (which occurs at one of the boundaries as predicted above) is nevertheless of the same order of magnitude as the supremum of the former.

B. Relation between N and Kernel Error

Having obtained a bound on the approximation error, we consider the problem of finding the smallest N such that (10) is within some allowed error margin $\varepsilon > 0$. Note that the quantity on the right in (13) is simply the tail probability of a Poisson random variable with parameter λ . We recall that a random variable X taking values in $\{0, 1, 2, \dots\}$ is said to follow a Poisson distribution with parameter $\lambda > 0$ if

$$\text{Prob}(X = n) = \frac{e^{-\lambda} \lambda^n}{n!} \quad (n = 0, 1, 2, \dots).$$

TABLE I

COMPARISON OF THE GAUSSIAN-POLYNOMIAL ORDER OBTAINED USING (18), WHERE (1) W_0 IS COMPUTED USING THE MATLAB FUNCTION `LAMBERTW` (N'_0), (2) W_0 IS GIVEN BY (19) (N''_0), AND (3) THE SERIES EVALUATION IS REFINED USING THREE NEWTON ITERATIONS (N'''_0).

σ_r	10	15	20	25	30	35	40	45	50
N'_0	214	107	67	48	37	30	25	21	19
N''_0	270	124	74	50	37	30	25	21	19
N'''_0	214	107	67	48	37	30	25	21	19

We can thus interpret the quantity on the right in (13) as the probability $\text{Prob}(X \geq N)$. In this context, the leading question is the following: given $\varepsilon > 0$, find the smallest N such that $\text{Prob}(X \geq N) \leq \varepsilon$. The advantage of expressing the problem in this form is that it brings to our disposal various tools for bounding the tail probability. For example, assuming that $N > \lambda$, we have the Chebyshev bound [18]:

$$\text{Prob}(X \geq N) \leq \frac{\lambda}{(N - \lambda)^2}. \quad (14)$$

On the other hand, the Chernoff bound [18] when $N > \lambda$ is given by

$$\text{Prob}(X \geq N) \leq \frac{e^{-\lambda}(e\lambda)^N}{N^N}. \quad (15)$$

Numerical experiments suggest that for $\sigma_r < 70$ and for a range of values of ε (to be reported shortly), the empirically computed N is always larger than λ . Under this assumption, we have the following estimate of the smallest N using (14):

$$N_0 = \lceil \lambda + \sqrt{\lambda/\varepsilon} \rceil, \quad (16)$$

where $\lceil x \rceil$ is the smallest integer greater than or equal to x .

As is well-known, the Chernoff bound (15) is typically tighter than the Chebyshev bound. However, finding the smallest N such that

$$\frac{e^{-\lambda}(e\lambda)^N}{N^N} \leq \varepsilon \quad (17)$$

is somewhat more involved.

Proposition II.2. *Let $t \mapsto W_0(t)$ be the inverse of the map $t \mapsto t \exp(t)$ on $(0, \infty]$. Then the smallest integer greater than λ for which (17) holds is*

$$N_0 = \lceil q/W_0(qe^{-p}) \rceil, \quad (18)$$

where $p = 1 + \log(\lambda)$ and $q = -\lambda - \log \varepsilon$.

The details are provided in Appendix VI-A. While $W_0(t)$ can be computed using the Matlab script `lambertw(0, t)`, we note that $W_0(t)$ can be approximated using a series expansion [19]. In particular, the first four terms are

$$W_0(t) = t - t^2 + \frac{3}{2}t^3 - \frac{8}{3}t^4. \quad (19)$$

However, we observed that (19) provides inexact estimates when λ is large, that is, when σ_r is small. An extremely large number of terms of the series are required to get a precise estimate. To address this problem, we propose to use Newton iterations for finding the positive root of $\nu(x) = x \log x - px - q = 0$ (see Appendix VI-A for notations), where the initialization is done using (18) and (19). Namely, starting

with $x_0 = q/W_0(qe^{-p})$, we run the following iterations for $k \geq 0$:

$$x_{k+1} = x_k - \frac{\nu(x_k)}{\nu'(x_k)} = x_k - \frac{x_k \log x_k - px_k - q}{\log x_k + 1 - p}. \quad (20)$$

In practice, we noticed that about 3-4 iterations are sufficient to produce a good solution. In Table III, we illustrate the improvement obtained after performing the Newton iterations. The complete scheme for computing the order for a given accuracy ε is summarized in Algorithm 1.

Data: σ_r, ε, T .
Result: N_0 .

```

1 if  $\sigma_r \geq 70$  then
2   |  $N_0 = 10$ ;
3 else
4   |  $\lambda = (T/\sigma_r)^2$ ;
5   |  $p = 1 + \log(\lambda)$ ;
6   |  $q = -\lambda - \log \varepsilon$ ;
7   |  $t = q/(e\lambda)$ ;
8   |  $W_0 = t - t^2 + 3t^3/2 - 8t^3/4$ ;
9   |  $N_0 = q/W_0$ ;
10  | if  $\sigma_r < 30$  then
11    | for  $k = 1, 2, 3$  do
12      | |  $N_0 = N_0 - \frac{N_0 \log(N_0) - pN_0 - q}{\log N_0 + 1 - p}$ ;
13    | end
14  | end
15 end
16  $N_0 = \lceil N_0 \rceil$ ;

```

Algorithm 1: Estimation of the approximation order.

Note that for $\sigma_r > 70$, we use a fixed order of 10. This is because the condition $N > \lambda$ in (14) and (15) is violated in this regime. Moreover, we have noticed that a small order suffices when σ_r is large. In Figure 4, we compare the estimated order N_0 obtained using the following methods: Chebyshev (16), Chernoff (18) along with (19), and Chernoff followed by Newton iterations (20). We also compare the corresponding errors (computed using exhaustive search) given by (10). Notice that the estimates are close to that obtained using exhaustive search when $\varepsilon = 0.1$; however, when $\varepsilon = 0.001$, the Chebyshev bound is quite loose.

III. FAST BILATERAL FILTERING

We now explain how Gaussian-polynomials can be used to derive a fast algorithm for implementing (1). As a first step, we center the intensity range $\{f(i) : i \in I\}$ around the origin. This is in keeping with the Taylor expansion in (7) which is performed around the origin. A simple means of doing so is to set $t_c = T$, assuming the dynamic range to be $[0, 2T]$, and to consider the centred image $\{h(i) : i \in I\}$ given by

$$h(i) = f(i) - t_c \quad (i \in I). \quad (21)$$

The crucial observation is that the shift operation in (21) commutes with the non-linear bilateral filtering.

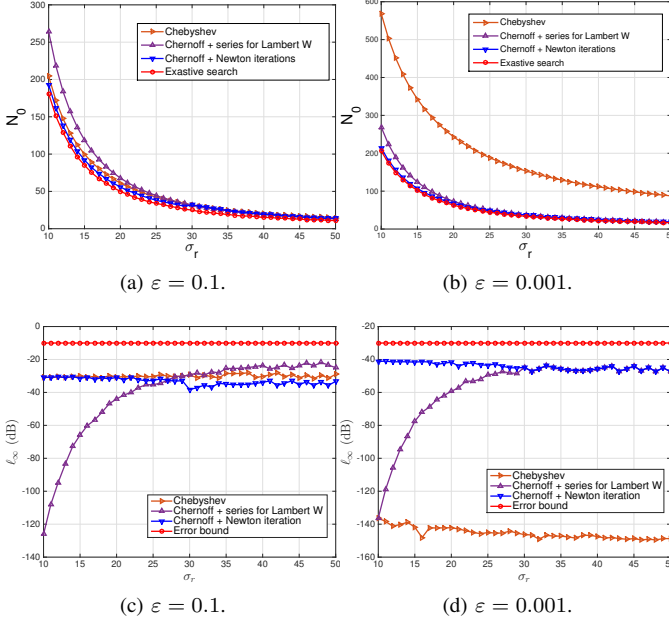


Fig. 4. For $\varepsilon = 0.1$ and 0.001 , we compare the order N_0 obtained using various methods (top row) and the corresponding error (bottom row).

Proposition III.1. For $1 \in I$,

$$f_{\text{BF}}(1) = h_{\text{BF}}(1) + t_c. \quad (22)$$

In other words, we can first centre the intensity range, apply the bilateral filter, and finally add back the centre to the output. Henceforth, we will assume that the range of the input image is $[-T, T]$. For an 8-bit grayscale image, $T = 128$.

A. Fast Algorithm

The underlying mechanism of the proposed fast algorithm is related to the fast algorithms in [12], [14]. The subtle difference is that instead of directly approximating (2), we approximate its translates in (1). In particular, we fix some order N , and approximate the range kernel in (1) using (7). This gives us the following Gaussian-Polynomial Approximation (GPA) of (1):

$$f_{\text{GPA}}(1) = \frac{\sum_{j \in \Omega} w(j) \phi_{N, \sigma_r}(f(1-j) - f(1)) f(1-j)}{\sum_{j \in \Omega} w(j) \phi_{N, \sigma_r}(f(1-j) - f(1))}. \quad (23)$$

Next, for $n = 0, \dots, N-1$, we define the images

$$G_n(1) = \left(\frac{f(1)}{\sigma_r} \right)^n, \quad F_n(1) = \exp \left(-\frac{f(1)^2}{2\sigma_r^2} \right) G_n(1), \quad (24)$$

and set

$$\bar{F}_n(1) = (F_n * w)(1) = \sum_{j \in \Omega} w(j) F_n(1-j). \quad (25)$$

We can then write (23) as (cf. Appendix VI-B)

$$f_{\text{GPA}}(1) = \frac{P(1)}{Q(1)}, \quad (26)$$

where

$$P(1) = \sigma_r \sum_{n=0}^{N-1} \frac{1}{n!} G_n(1) \bar{F}_{n+1}(1), \quad (27)$$

and

$$Q(1) = \sum_{n=0}^{N-1} \frac{1}{n!} G_n(1) \bar{F}_n(1). \quad (28)$$

Notice that we have effectively transferred the non-linearity of the bilateral filter to the intermediate images in (24), which are obtained from the input image using simple pointwise transforms. The computational advantage that we get from the above manipulation is that the spatial filtering in (25) can be computed using $O(1)$ operations per pixel when w is a box or a Gaussian [13]. The overall cost of computing (23) is therefore $O(1)$ per pixel with respect to the filter size W . This is a substantial reduction from the $O(W^2)$ complexity of the direct implementation of (1).

The complete algorithm for computing (23) is summarized in Algorithm 2, which we will continue to refer as GPA. Note that we efficiently implement steps (24) to (28) by avoiding redundant computations. In particular, we recursively compute the images in (24) and the factorials in (27) and (28). Notice that steps 6-11, 16-17, 21-22, and 26 are cheap pointwise operations. The main computation in Algorithm 2 is the spatial filtering in step 19, and the initial filtering in step 13. That is, the overall cost is dominated by the cost of computing $N+1$ spatial filtering. In this regard, we note that for the same degree N , the number of spatial filterings required in [12], [14] is $4N$, and that in [8] is $2N$. Moreover, we note that the proposed algorithm involves the evaluation of a transcendental function just once, namely in step 7. In contrast, the algorithm in [8] requires N evaluations of the Gaussian over the whole image. Thus, the present algorithm has smaller rounding errors, and is better suited for hardware implementations [15] compared to the above mentioned algorithms. Yet another key advantage with the Algorithm 2 is that we need just six images (excluding the input and output images) for the complete pipeline. As against this, the algorithm in [8] requires the computation and storage of N principal images, which are interpolated to get the final output.

B. Filtering Accuracy

It is clear that the kernel error, and hence the overall quality of approximation, is controlled by the order N . In this regard, we need a rule to fix N in Algorithm 2. As before, we will consider the worst-case error given by

$$\|f_{\text{BF}} - f_{\text{GPA}}\|_{\infty} = \max \left\{ |f_{\text{BF}}(1) - f_{\text{GPA}}(1)| : 1 \in I \right\}. \quad (29)$$

By bounding (29), we can control the pixelwise difference between the exact and the approximate bilateral filter. In particular, we have the following result which formally establishes the intuitive fact that the filtering accuracy is essentially within a certain factor of the kernel error given by (10). The details of the derivation are provided in Appendix VI-C.

Proposition III.2. Suppose that the spatial filter is non-negative and normalized, i.e., $w(j) \geq 0$ for all $j \in \Omega$, and

$$\sum_{j \in \Omega} w(j) = 1. \quad (30)$$

```

1 Input:  $\{f(i) : i \in I\}$  taking values in  $[0, 2T]$ ;
2 Spatial Filter:  $\Omega$  and  $\{w(i) : i \in \Omega\}$ ;
3 Parameters:  $\sigma_r$  and  $N$ ;
4 Output:  $\{f_{\text{GPA}}(i) : i \in I\}$  given by (23);
5 for  $i \in I$  do
6    $h(i) = f(i) - T$ ;
7    $F(i) = \exp(-h(i)^2/2\sigma_r^2)$ ;
8    $G(i) = 1$ ;
9    $P(i) = 0$ ;
10   $Q(i) = 0$ ;
11   $H(i) = h(i)/\sigma_r$ ;
12 end
13  $\bar{F}(i) = (F * w)(i)$ ;
14 for  $n = 1, \dots, N$  do
15   for  $i \in I$  do
16      $Q(i) = Q(i) + G(i)\bar{F}(i)$ ;
17      $F(i) = H(i)F(i)$ ;
18   end
19    $\bar{F}(i) = (F * w)(i)$ ;
20   for  $i \in I$  do
21      $P(i) = P(i) + G(i)\bar{F}(i)$ ;
22      $G(i) = H(i)G(i)/n$ ;
23   end
24 end
25 for  $i \in I$  do
26    $f_{\text{GPA}}(i) = \sigma_r (P(i)/Q(i)) + T$ ;
27 end

```

Algorithm 2: Gaussian-Polynomial Approximation (GPA).

Then

$$\|f_{\text{BF}} - f_{\text{GPA}}\|_{\infty} \leq 2 \frac{T \|E_{N, \sigma_r}\|_{\infty}}{w(0) - \|E_{N, \sigma_r}\|_{\infty}}. \quad (31)$$

We note that the spatial filters (3) and (4) are non-negative, and that $w(i)$ appears in both the numerator and denominator of (1) and (23). Therefore, we can assume (30) without any loss of generality. In fact, (30) is automatically true for the box filter. We also recall that the range of the image is assumed to be centered; the intensity values are in the interval $[-T, T]$, where $T \approx 128$ for most grayscale images.

C. Relation between Accuracy and N_0

Note that by combining (31) with the bound in (15), we get a direct control on the filtering accuracy in terms of the approximation order. In particular, suppose that we want (29) to be within $\pm\delta$. A sufficient condition for this is that

$$\|E_{N, \sigma_r}\|_{\infty} \leq \frac{w(0)\delta}{2T + \delta}.$$

To summarize, we have the following guarantee that follows from (31).

Corollary III.3. *Suppose that N is set using Algorithm 1, where ε is given by*

$$\varepsilon = \frac{w(0)\delta}{2T + \delta}. \quad (32)$$

Then the output of Algorithm 2 is within $\pm\delta$ of the output of the bilateral filter.

IV. EXPERIMENTS AND DISCUSSION

We implemented the proposed GPA algorithm using Matlab 8.4 on an Intel 3.4 GHz Linux system with 8 GB memory. The Matlab implementation has been shared here [20]. The set of grayscale images used for the experiments are shown in Figure 5. We compared the proposed algorithm with the following fast algorithms: Yang [8], Paris [9], Weiss [22], and the Shiftable Bilateral Filter (SBF) [14]. We used the Matlab implementation of these algorithms to make the comparison fair; moreover, we used the parameter settings suggested in the respective papers. For determining the order in [8] for a given accuracy parameter δ , we have used (34).

Experiment 1 The output of the proposed GPA algorithm on a couple of images are shown in Figures 6 and 7. We also provided the output obtained using exact bilateral filtering. We performed the comparison using the box and the Gaussian kernels for the spatial filter. Notice that the speedup obtained is significant. Moreover, the filtered images are visually identical and numerically very close, in terms of the ℓ_{∞} and mean-squared errors. We have used the following definition of mean-squared error (MSE):

$$\text{MSE} = 10 \log_{10} \left\{ |I|^{-1} \sum_{i \in I} (f_{\text{BF}}(i) - f_{\text{GPA}}(i))^2 \right\},$$

where $|I|$ denotes the number of pixels in the image.

To get a better understanding of how N_0 varies with δ , we used the following approximation (see Appendix VI-D):

$$N_0 \approx 1.72 \left(\frac{T}{\sigma_r} \right)^2 + \log \left(\frac{2T}{w(0)\delta} \right). \quad (33)$$

An important point to note in (33) is the logarithmic dependence on δ . In fact, the $\log(1/\delta)$ factor can be traced back to the tail bound in (15), which, in turn, follows from the particular splitting in (7). The implication of the logarithmic dependence is that we can force δ to be quite small without blowing up N_0 .

To further highlight the importance of (33), we compared (33) with the corresponding estimate for Yang's algorithm [8]:

$$N_0 \approx \frac{1.14 \times 10^5}{\delta^{1/2} \sigma_r^2}. \quad (34)$$

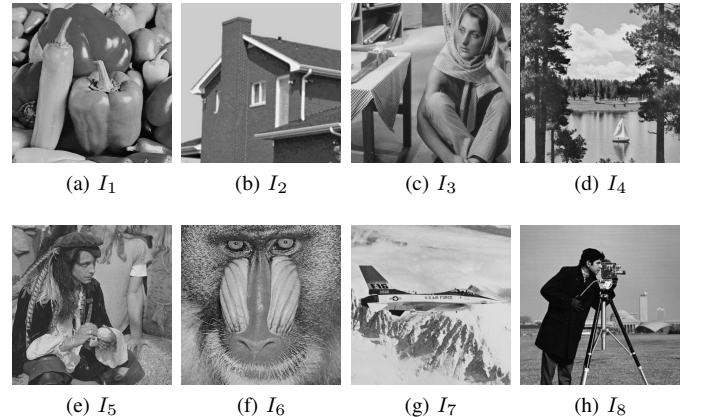


Fig. 5. List of grayscale images used for the experiments in Section IV. The images were obtained from [21]. All images are of size 512×512 .

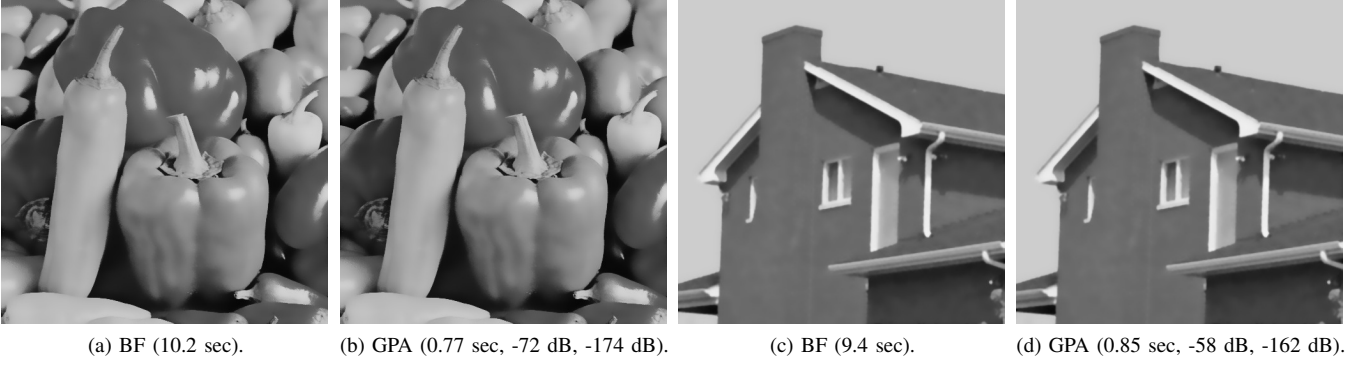


Fig. 6. Comparison of the exact bilateral filter (BF) and the proposed approximation (GPA) on images I_1 and I_2 . A Gaussian kernel ($\sigma_s = 5$) is used for the spatial filter, and $\sigma_r = 50$ for the Gaussian range kernel. The accuracy parameter was set to $\delta = 0.1$ for the GPA. In the caption of (a) and (c), we report the run time of the BF. In the caption of (b) and (d), we report the run time of the GPA, and the ℓ_∞ and mean-squared errors between BF and GPA.

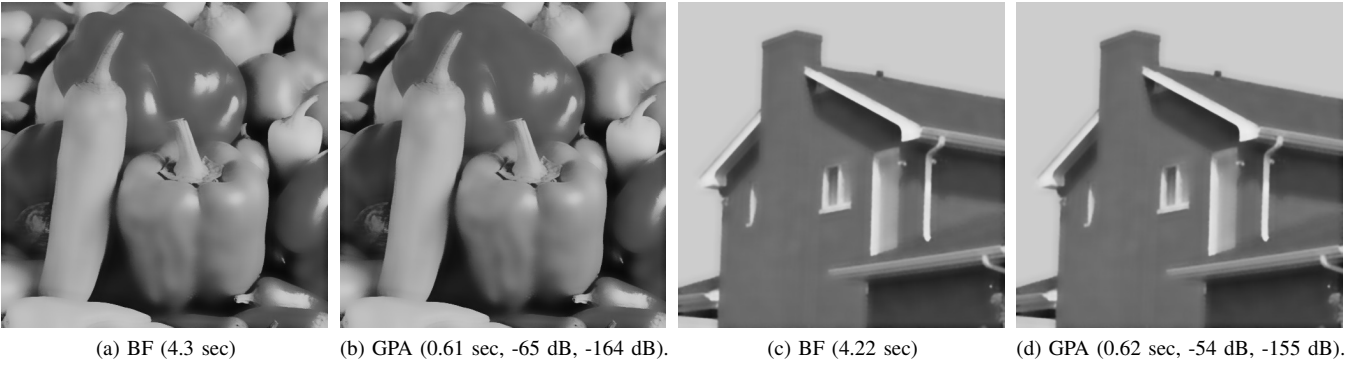


Fig. 7. The setup here is identical to that in Figure 6, with the difference that a box kernel ($W = 10$) is used for the spatial filter instead of the Gaussian.

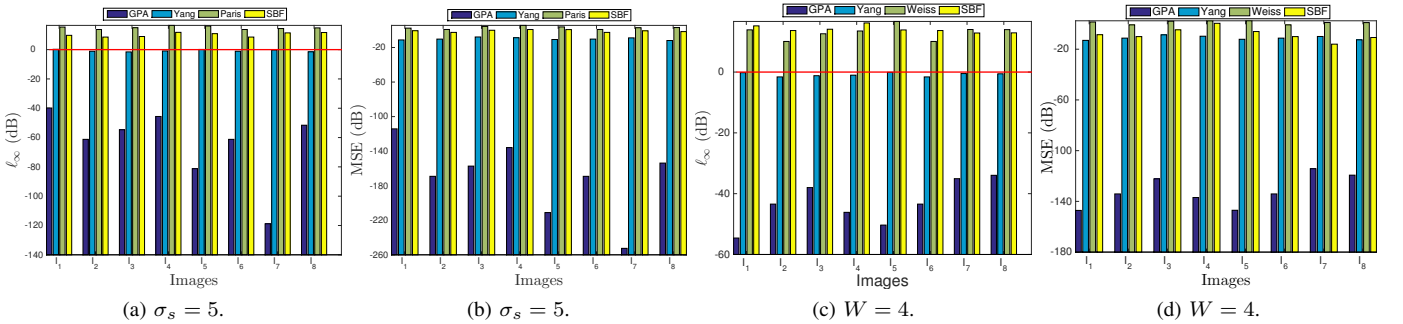


Fig. 10. Comparison of the filtering accuracy (ℓ_∞ error and MSE) of various fast algorithms on the images in Figure 5. The red horizontal lines in 10a and 10c represent the accuracy parameter δ used for GPA and Yang's algorithm. The tolerance for SBF was set to be 0.01. In 10a - 10b, we show the results for a Gaussian spatial kernel, and in 10c - 10d we show the results for a box kernel. We used $\sigma_r = 30$ for the Gaussian range kernel in all the experiments.

The above estimate was recently derived in [16]. In particular, notice that the dependence on σ_r is similar to that in (33). However, the dependence on δ is much more strong in (34) compared to (33), since $\log(1/\delta) \ll \delta^{-1/2}$ when $\delta < 1$. Moreover, the leading constant in (34) is much larger than the constant in the first term in (33). As an example, when $\delta = 3$ and $\sigma_r = 50$, we have $N_0 \approx 27$ for Yang's algorithm (this is the estimate reported in [16]). On the other hand, the corresponding estimate for our algorithm is $N_0 \approx 19$ (assuming that $T = 128$ and that we use a box filter of size 3×3). The difference becomes dramatic for smaller values of δ . For example, when $\sigma_r = 50$ and $\delta = 0.01$, the

estimate from (33) is 24, while that from (34) is 456. Further comparisons are provided in Table II. Notice that the order for Yang's approximation explodes when $\delta < 1$ (sub-pixel accuracy). It is also seen from the table that (33) provides a close approximation of (18) for the setting under consideration.

Experiment 2 A graphic comparison of the algorithms for various settings of the spatial and range kernels is presented in Figures 8 and 9. As before, we performed the comparison for both the box and Gaussian spatial filters. It is evident from these results that the proposed method is competitive with existing methods in terms of the speed-accuracy tradeoff.

Experiment 3 We next compared the proposed algorithm

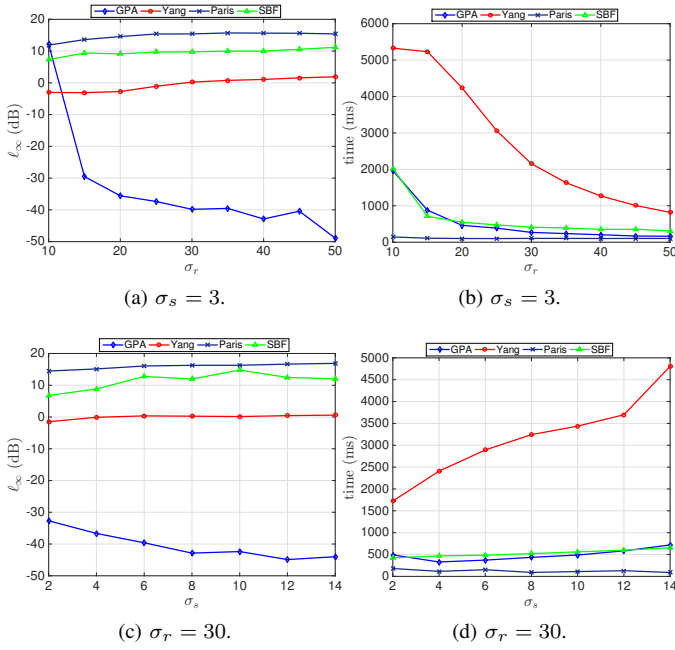


Fig. 8. Comparison of the filtering accuracy and the run time of four different algorithms as a function of the parameters σ_s (Gaussian spatial filter) and σ_r . We used image I_1 in Figure 5 for the comparison. We used $\delta = 1$ for GPA and Yang's algorithm [8]. A tolerance of 0.01 was used for the SBF [14].

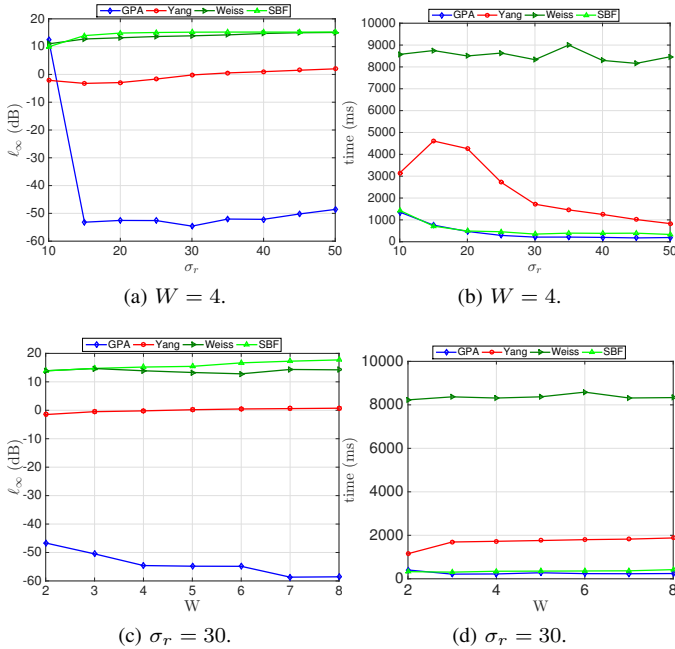


Fig. 9. Comparison of the filtering accuracy and the run time as a function of W and σ_r . The settings are identical to that in Figure 8; the difference here is that we have used a box spatial kernel instead of a Gaussian kernel.

with existing fast algorithms on the images shown in Figure 5. A summary of the comparisons (in terms of maximum pixelwise error and MSE) is provided in Figure 10.

Experiment 4 Finally, we performed a detailed comparison of the proposed algorithm with Yang's algorithm, which is widely considered to be the state-of-the-art algorithm. In the first comparison, we fixed an image and the parameters of

the bilateral filter. The order N was then varied and the corresponding error and run times were noted. The results are presented in Table III. Notice that the run time of GPA is consistently smaller than that of Yang's algorithm for both the box and Gaussian kernels. Indeed, as remarked earlier, for a fixed order N , Yang's algorithm [8] requires $2N$ spatial filterings, while GPA requires only $N + 1$ spatial filterings. Thus, the runtime of GPA is about half of that of Yang's algorithm. Moreover, beyond a certain N , GPA provides much better filtering accuracy. We performed a similar experiment by varying δ , the results of which are reported in Table IV. Notice that the run time of Yang's algorithm becomes prohibitively large when δ is small.

V. CONCLUSION

We presented a novel fast algorithm for approximating the bilateral filter. The algorithm was shown to be both fast and accurate in practice using extensive experiments. The space and time complexity of the proposed algorithm is smaller than the state-of-the-art algorithm of Yang [8], and, moreover, was shown to provide much better accuracy. We also performed an error analysis of the approximation scheme, and presented a rule for setting the approximation order that can guarantee the filtering accuracy to be within a desired margin. Before concluding, we note that the proposed algorithm can be used to perform cross bilateral filtering, and can also be extended for the filtering of video and volume data.

VI. APPENDIX

A. Derivation of (18)

Taking the logarithm of (17), we can restate the problem as one of finding the smallest integer $x > \lambda$ such that

$$\nu(x) = x \log x - px - q \geq 0 \quad (35)$$

where $p = 1 + \log(\lambda)$ and $q = -\lambda - \log \varepsilon$.

Notice that $\nu'(\lambda) = 0$ and $\nu''(x) = 1/x > 0$. Hence, $\nu(x)$ is strictly convex over $(0, \infty)$ with a minimum at $x = \lambda$. Since $\nu(\lambda) = \log \varepsilon < 0$ when $\varepsilon < 1$, we conclude that there exists some $\theta > \lambda$ for which $\nu(\theta) = 0$. The smallest integer solution of (17) is precisely $\lceil \theta \rceil$. To find θ , we solve the equations $\nu(\theta) = 0$ and $\theta > \lambda$. Note that we can write $\nu(\theta) = 0$ as

$$\frac{q}{\theta} \exp\left(\frac{q}{\theta}\right) = qe^{-p}, \quad (36)$$

which is of the form $y \exp(y) = qe^{-p}$, where $y = q/\theta$. The inverse of the mapping $y \mapsto y \exp(y)$ is a well-studied function called the Lambert W-function [19]. In particular, the

TABLE II
COMPARISON OF THE ORDER N_0 REQUIRED TO ACHIEVE A DESIRED ACCURACY δ WHEN $\sigma_s = 5$ AND $\sigma_r = 30$.

δ	10^{-3}	10^{-2}	0.05	0.1	1	3
$[N_0]$ using (18)	49	46	45	44	41	40
$[N_0]$ using (33)	49	47	45	44	42	41
$[N_0]$ using (34)	4006	1267	566	401	127	73

TABLE III

COMPARISON OF THE PROPOSED GPA ALGORITHM WITH YANG'S ALGORITHM [8] FOR DIFFERENT ORDER N . THE ℓ_∞ ERROR AND THE MSE ARE IN DECIBELS, WHILE THE TIME IS IN MILLISECONDS. THE COMPARISON IS DONE ON IMAGE I_5 USING BOTH BOX AND GAUSSIAN SPATIAL FILTERS; THE TYPE OF SPATIAL FILTER IS MENTIONED WITHIN BRACKETS. THE RESPECTIVE PARAMETERS FOR THE BOX AND GAUSSIAN FILTER ARE $W = 4$ AND $\sigma_s = 5$, AND $\sigma_r = 30$ FOR THE GAUSSIAN RANGE KERNEL. NOTICE THAT THE ACCURACY OF GPA SATURATES ABOVE $N = 60$.

N	GPA (Gaussian)			Yang (Gaussian)			GPA (Box)			Yang (Box)		
	ℓ_∞	MSE	time	ℓ_∞	MSE	time	ℓ_∞	MSE	time	ℓ_∞	MSE	time
10	14.87	8.48	85	10.55	11.08	217	15.81	6.61	85	10.30	9.24	252
20	2.97	-20.07	146	7.90	4.88	365	1.55	-22.24	152	7.63	3.08	413
30	-18.23	-67.35	210	6.44	1.33	519	-18.63	-69.46	173	6.09	-0.46	455
40	-50.81	-137.34	275	5.15	-1.19	695	-50.80	-139.08	197	4.79	-2.98	546
50	-93.31	-225.95	346	4.29	-3.12	857	-92.72	-226.90	300	3.87	-4.91	805
60	-119.03	-254.19	407	3.46	-4.72	995	-120.69	-258.14	295	3.06	-6.51	767
65	-119.03	-254.19	439	3.11	-5.41	1067	-120.69	-258.14	323	2.70	-7.20	840
70	-119.03	-254.19	477	2.79	-6.06	1175	-120.69	-258.14	456	2.38	-7.85	1216

TABLE IV

COMPARISON OF THE GPA ALGORITHM WITH YANG'S ALGORITHM [8] AT DIFFERENT δ . SEE TABLE III FOR THE PARAMETER SETTINGS.

δ	GPA (Gaussian)			Yang (Gaussian)			GPA (Box)			Yang (Box)		
	N_0	ℓ_∞	time	N_0	ℓ_∞	time	N_0	ℓ_∞	time	N_0	ℓ_∞	time
0.05	45	-71.15	445	567	-6.29	11143	44	-66.51	429	567	-6.71	8905
0.1	44	-66.89	383	401	-4.78	8407	43	-62.45	207	401	-5.20	5381
0.5	42	-58.65	376	180	-1.29	3776	41	-54.59	282	180	-1.71	3092
1	41	-54.68	370	132	-0.22	2635	41	-54.59	288	132	-0.20	1981
2	41	-54.68	377	90	1.71	1692	40	-50.80	192	90	1.30	1137
3	40	-50.81	295	74	2.55	1305	39	-47.10	262	74	2.14	1246

inverse (which is generally multivalued) in this case is given and by

$$\frac{q}{\theta} = W_0(qe^{-p}),$$

where $W_0(t)$ is one of the two branches of the Lambert W-function [19]. This gives us estimate (18).

B. Derivation of (26)

In terms of (24), we can write $\phi_{N,\sigma_r}(f(1-j)-f(1))f(1-j)$ as

$$\sigma_r \exp\left(-\frac{f(1)^2}{2\sigma_r^2}\right) \sum_{n=0}^{N-1} \frac{1}{n!} G_n(1) F_{n+1}(1-j). \quad (37)$$

On substituting (37) in the numerator of (23), and exchanging the summations, we get

$$\sum_{j \in \Omega} w(j) \phi_{N,\sigma_r}(f(1-j)-f(1))f(1-j) = \exp\left(-\frac{f(1)^2}{2\sigma_r^2}\right) P(1),$$

which gives us (27) where we have used (25). Similarly, on substituting (37) in the denominator of (23), and exchanging the summations, we get

$$\sum_{j \in \Omega} w(j) \phi_{N,\sigma_r}(f(1-j)-f(1)) = \exp\left(-\frac{f(1)^2}{2\sigma_r^2}\right) Q(1),$$

where $Q(1)$ is given by (28). Cancelling the common exponential term from the numerator and denominator, we get (26).

C. Derivation of (31)

To establish (31), we write (1) as $f_{\text{BF}}(1) = P_1(1)/Q_1(1)$, where

$$P_1(1) = \sum_{j \in \Omega} w(j) g_{\sigma_r}(f(1-j)-f(1)) f(1-j),$$

$$Q_1(1) = \sum_{j \in \Omega} w(j) g_{\sigma_r}(f(1-j)-f(1)).$$

Similarly, we write (23) as $f_{\text{GPA}}(1) = P_2(1)/Q_2(1)$, where

$$P_2(1) = \sum_{j \in \Omega} w(j) \phi_{N,\sigma_r}(f(1-j)-f(1)) f(1-j),$$

and

$$Q_2(1) = \sum_{j \in \Omega} w(j) \phi_{N,\sigma_r}(f(1-j)-f(1)).$$

We can then write $f_{\text{BF}}(1) - f_{\text{GPA}}(1)$ as

$$\begin{aligned} &= \frac{P_1(1)(Q_2(1) - Q_1(1)) + Q_1(1)(P_1(1) - P_2(1))}{Q_1(1)Q_2(1)} \\ &= \frac{1}{Q_2(1)} \left[f_{\text{BF}}(1)(Q_2(1) - Q_1(1)) + P_1(1) - P_2(1) \right]. \end{aligned} \quad (38)$$

We uniformly upper-bound (resp. lower-bound) the numerator (resp. denominator) in (38). In particular, note that

$$\|f_{\text{BF}}\|_\infty \leq T. \quad (39)$$

This follows from the fact that $f_{\text{BF}}(1)$ in (1) can be expressed as a convex combination of $\{f(1-j) : j \in \Omega\}$. On the other hand, $Q_2(1) - Q_1(1)$ is

$$\sum_{j \in \Omega} w(j) [g_{\sigma_r}(f(1-j)-f(1)) - \phi_{N,\sigma_r}(f(1-j)-f(1))].$$

Therefore, using (30), we get

$$\|Q_1 - Q_2\|_\infty \leq \|E_{N,\sigma_r}\|_\infty. \quad (40)$$

Similarly,

$$\|P_1 - P_2\|_\infty \leq \|E_{N,\sigma_r}\|_\infty T. \quad (41)$$

To uniformly lower-bound $Q_2(1)$, we note that for $1 \in I$,

$$Q_1(1) = w(0)g_{\sigma_r}(0) + \sum_{j \in \Omega \setminus \{0\}} w(j)g_{\sigma_r}(f(1-j) - f(1)) \geq w(0),$$

where we have used the non-negativity of the range and spatial kernels. Using the inverse triangle inequality along with (40), we have for $1 \in I$,

$$|Q_2(1)| \geq Q_1(1) - |Q_2(1) - Q_1(1)| \geq w(0) - \|E_{N,\sigma_r}\|_\infty. \quad (42)$$

Combining (38) - (42), we arrive at (31).

D. Derivation of (33)

Note that typically $\delta \ll T$. For example, T is in hundreds for a grayscale image, whereas, $\delta \sim 1$. Therefore, it follows from (32) that $\varepsilon \approx w(0)\delta/(2T)$. On the other hand, from (18) and (19), we have

$$N_0 \approx \frac{q}{t - t^2} = \frac{e\lambda}{1 - (q/e\lambda)},$$

where $t = q/e\lambda$ and $q = -\lambda + \log(1/\varepsilon)$. Since $|q| < e\lambda$,

$$\frac{1}{1 - (q/e\lambda)} \approx 1 + (q/e\lambda).$$

Therefore, $N_0 \approx e\lambda + q = (e - 1)\lambda + \log(1/\varepsilon)$.

VII. ACKNOWLEDGEMENTS

The authors thank Dr. Alessandro Foi and the anonymous reviewers for their useful comments and suggestions.

REFERENCES

- [1] K. N. Chaudhury, "Fast and accurate bilateral filtering using Gauss-polynomial decomposition," *Proc. IEEE International Conference on Image Processing*, pp. 2005 - 2009, 2015.
- [2] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629-639, 1990.
- [3] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *Proc. IEEE International Conference on Computer Vision*, pp. 839-846, 1998.
- [4] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, *Bilateral Filtering: Theory and Applications*, Now Publishers Inc., 2009.
- [5] C. Knaus and M. Zwicker, "Progressive image denoising," *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 3114-3125, 2014.
- [6] K. N. Chaudhury and K. Rithwik, "Image denoising using optimally weighted bilateral filters: A SURE and fast approach," *Proc. IEEE International Conference on Image Processing*, pp. 108-112, 2015.
- [7] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 257-266, 2002.
- [8] 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.
- [9] 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.
- [10] K. Sugimoto and S. I. Kamata, "Compressive bilateral filtering," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3357-3369, 2015.
- [11] F. Porikli, "Constant time $O(1)$ bilateral filtering," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.
- [12] 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.
- [13] R. Deriche, "Recursively implementing the Gaussian and its derivatives," *Research Report*, INRIA-00074778, 1993.
- [14] K. N. Chaudhury, "Acceleration of the shiftable algorithm for bilateral filtering and nonlocal means," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1291-1300, 2013.
- [15] J. M. Muller, *Elementary Functions: Algorithms and Implementation*, Birkhauser Boston, 2006.
- [16] S. An, F. Boussaid, M. Bennamoun, and F. Sohel, "Quantitative error analysis of bilateral filtering," *IEEE Signal Processing Letters*, vol. 22, no. 2, pp. 202-206, 2015.
- [17] C. Yang, R. Duraiswami, and N. A. Gumerov, "Improved fast gauss transform," *Technical Report CS-TR-4495*, UMIACS, Univ. of Maryland, College Park, 2003.
- [18] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, 2005.
- [19] R. M. Corless, G. H. Gonnet, D. E. Hare, D. J. Jeffrey, and D. E. Knuth, "On the Lambert W function," *Advances in Computational Mathematics*, vol. 5, no. 1, pp. 329-359, 1996.
- [20] K. N. Chaudhury and S. Dabhade, *Fast and Accurate Bilateral Filtering* (www.mathworks.com/matlabcentral/fileexchange/56158), MATLAB Central File Exchange, retrieved March 25, 2016.
- [21] http://www.imageprocessingplace.com/root_files_V3/image_databases.htm.
- [22] B. Weiss, "Fast median and bilateral filtering," *Proc. ACM Siggraph*, vol. 25, pp. 519-526, 2006.