Addendum to “Fast $O(1)$ bilateral filtering using trigonometric range kernels”

Kunal Narayan Chaudhury[*], Daniel Sage, and Michael Unser

Abstract

In [1], we proposed a constant-time or $O(1)$ algorithm for fast Gaussian bilateral filtering [2]. This was done by approximating the Gaussian range kernel of the bilateral filter using raised cosines. Based on this trick, we formulated a parallel algorithm for the bilateral filter that involved the spatial filtering of a stack of images. The overall implementation was shown to be very fast when the σ of the Gaussian is reasonably large. However, an unfortunate drawback of this algorithm is that the size of the stack depends inversely on σ^2. This apparently makes the implementation impractical for small values of σ, even using parallel computation. In this note, we provide a simple and practical solution to this problem using truncation, where we only process the *significant* images in the stack.

1 Background

We briefly recall the idea behind the fast bilateral filtering algorithm proposed in [1]. The standard Gaussian bilateral filter is given

$$\tilde{f}(x) = \frac{1}{\eta(x)} \int_{\Omega} g_{\sigma_s}(y) g_{\sigma_r}(f(x-y) - f(x))\, f(x-y)\, dy \qquad (1)$$

where

$$\eta(x) = \int_{\Omega} g_{\sigma_s}(y) g_{\sigma_r}(f(x-y) - f(x))\, dy.$$

[*]Correspondence: Kunal N. Chaudhury (kchaudhu@princeton.edu). Kunal N. Chaudhury is at Princeton University, Princeton, USA. Michael Unser and Daniel Sage are with the Biomedical Imaging Group, École Polytechnique Fédérale de Lausanne, Switzerland.

Here $g_{\sigma_s}(\boldsymbol{x})$ is the spatial kernel and $g_{\sigma_r}(s)$ is the one-dimensional range kernel, both being Gaussian. The subscripts denote the standard deviations of the corresponding Gaussians. A direct implementation of (1) requires $O(\sigma_s^2)$ operations per pixel.

Assuming the intensity values $f(\boldsymbol{x})$ to be restricted to the interval $[-T, T]$, the idea in [1] was to approximate $g_{\sigma_r}(s)$ using raised cosine kernels. This was motivated by observation that, for all $-T \leq s \leq T$,

$$\lim_{N \longrightarrow \infty} \left[\cos\left(\frac{\gamma s}{\rho \sqrt{N}} \right) \right]^N = \exp\left(-\frac{\gamma^2 s^2}{2\rho^2} \right). \tag{2}$$

The parameters $\gamma = \pi/2T$ and $\rho = \gamma \sigma_r$ are used to control the variance of the target Gaussian on the right, and to ensure that the raised cosines on the left are non-negative and unimodal on $[-T, T]$ for every N. We refer the readers to [1] for details.

The advantage of approximating the Gaussian using the raised cosine is that it allowed one to express the otherwise non-linear transform in (1) as the superposition of Gaussian convolutions, applied on simple pointwise transforms of the image. In short, this required one to compute a series of spatial filterings

$$(F_0 * g_{\sigma_r})(\boldsymbol{x}), (F_1 * g_{\sigma_r})(\boldsymbol{x}), \cdots, (F_N * g_{\sigma_r})(\boldsymbol{x}),$$

where the image stack $F_0(\boldsymbol{x}), F_1(\boldsymbol{x}), \ldots, F_N(\boldsymbol{x})$ are obtained from pointwise transforms of $f(\boldsymbol{x})$. Each of these Gaussian filtering are then computed using a fast $O(1)$ algorithm. As a result, the overall algorithm has $O(1)$ complexity as well. The added advantage here is that one could implement the convolutions in parallel. This makes the final implementation very fast, e.g., see the timing results in [1].

2 Speed-up using truncation

The significance of the stack size N is that it allows one to arbitrarily control the accuracy of the Gaussian approximation; larger the value of N, the better is the accuracy. However, for a given σ_r, it was noted in [1] that N had to greater than a threshold value N_0 for the raised cosines to be non-negative and unimodal. Besides symmetry, these are two desirable properties of a kernel. In particular, N_0 had to be larger than the integer rounding of $4T^2/\pi^2\sigma_r^2$. This makes N_0 very large when σ_r is small. One solution to this problem is to use a tight estimate of T, which is given by the maximum variation in the local image intensities. In particular, note that T can be much smaller than the global dynamic range of the image. A good estimate of T can be obtained using a simple pre-processing.

The above solution is, however, ineffective if T is actually very large. For example, most natural images exhibit large local fluctuations in intensity. In such cases, T could be very close to the dynamic range of the entire image. We now propose a more effective solution to this problem by considering the formula that relates the stack size to the order of approximation N in (2):

$$\phi_N(s) = \left[\cos\left(\frac{\gamma s}{\rho\sqrt{N}}\right)\right]^N = \sum_{n=0}^{N} 2^{-N}\binom{N}{n}\cos\left(\frac{\gamma(2n-N)s}{\rho\sqrt{N}}\right). \quad (3)$$

Thus the stack size is $N+1$, the number of coefficients in the expansion of $\phi_N(s)$. The above expression was used in [1] for defining the images $F_0(\boldsymbol{x}), F_1(\boldsymbol{x}), \ldots, F_N(\boldsymbol{x})$ in the stack.

Let us examine the coefficients $2^{-N}\binom{N}{n}$ in (3). They are clearly positive and sum up to one. In fact, they are also unimodal and closely follow the shape of the target Gaussian. The cosine functions act as the interpolants. Note that the smallest coefficients are at the extreme and the largest coefficients are at the center. In particular, the smallest coefficient is $1/2^N$, while the largest one is $N!/2^N[(N/2)!]^2$ (assuming N to be even). For large N, this is approximately $\sqrt{2/\pi N}$ using Stirling's formula. Thus, as N gets large, the coefficients get smaller. What is more significant is that the ratio of the smallest to the largest coefficient is $1/\sqrt{\pi N 2^{2N-1}}$, and this keeps shrinking at an exponential rate with the increase in N. On the other hand, note that the interpolating cosines are always bounded within $[-1,1]$ for all N.

The above observation suggests a means for approximating $\phi_N(s)$ very closely, particularly for large N. This is namely by truncating the terms on the extremes of (3) with the smallest coefficients. That is, we use the approximation

$$\phi_{N,M}(s) = \sum_{n=M}^{N-M} 2^{-N}\binom{N}{n}\cos\left(\frac{\gamma(2n-N)s}{\rho\sqrt{N}}\right). \quad (4)$$

In particular, for a given tolerance $\varepsilon > 0$, let $M = M(\varepsilon)$ be the smallest term for which

$$\sum_{k=0}^{M+1} 2^{-N}\binom{N}{n} > \varepsilon/2. \quad (5)$$

Then, we are guaranteed that the error $|\phi_N(s) - \phi_{N,M}(s)|$ is within ε for all $-T \leq s \leq T$. Note that $\phi_{N,M}(s)$ can no longer be guaranteed to be non-negative or unimodal. However, what we can guranantee is that the negative overshoots are within $-\varepsilon$. As shown next, the quality of the approximation for a reasonable values ε is very satisfactory. The main point from the computational perspective is that the truncation M turns out to be quite large even for small ε.

3 Results

We perform some experiment to judge the quality of approximation obtained before and after the truncation. We consider the most adverse setting where $T = 255$ (maximum range for a grayscale image), and take a narrow Gaussian $g_{\sigma_r}(s)$ for which $\sigma_r = 10$. We approximate $g_{\sigma_r}(s)$ using a raised cosine. To ensure non-negativity and unimodality, the order of the cosine must at least be as large as $N_0 = 264$. This results in a very large image stack. The implementation of the bilateral filter in [1] becomes impractical for such large stacks, even with parallel computation.

To make this reasonable, we use the truncation proposed in the previous section. We set the tolerance to $\varepsilon = 0.005$. This ensures that the size of negative overshoot (oscillation) resulting from the truncation is well with -0.005. The remarkable fact is that we can use only the central 42 terms in (3) to achieve this level of accuracy. The results of the Gaussian approximation before and after the truncation are shown in Figure 1. Note that the quality of the approximation is reasonably satisfactory, even though we have dropped almost 84% of the terms in (3).

We implemented the algorithm in Java as an ImageJ plugin, both with and without truncation. The source code can be found here:

http://bigwww.epfl.ch/algorithms/bilateral-filter/.

References

[1] K.N. Chaudhury, D. Sage, and M. Unser, "Fast O(1) Bilateral Filtering Using Trigonometric Range Kernels," IEEE Transactions on Image Processing, 2011.

[2] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," IEEE International Conference on Computer Vision, pp. 839-846, 1998.

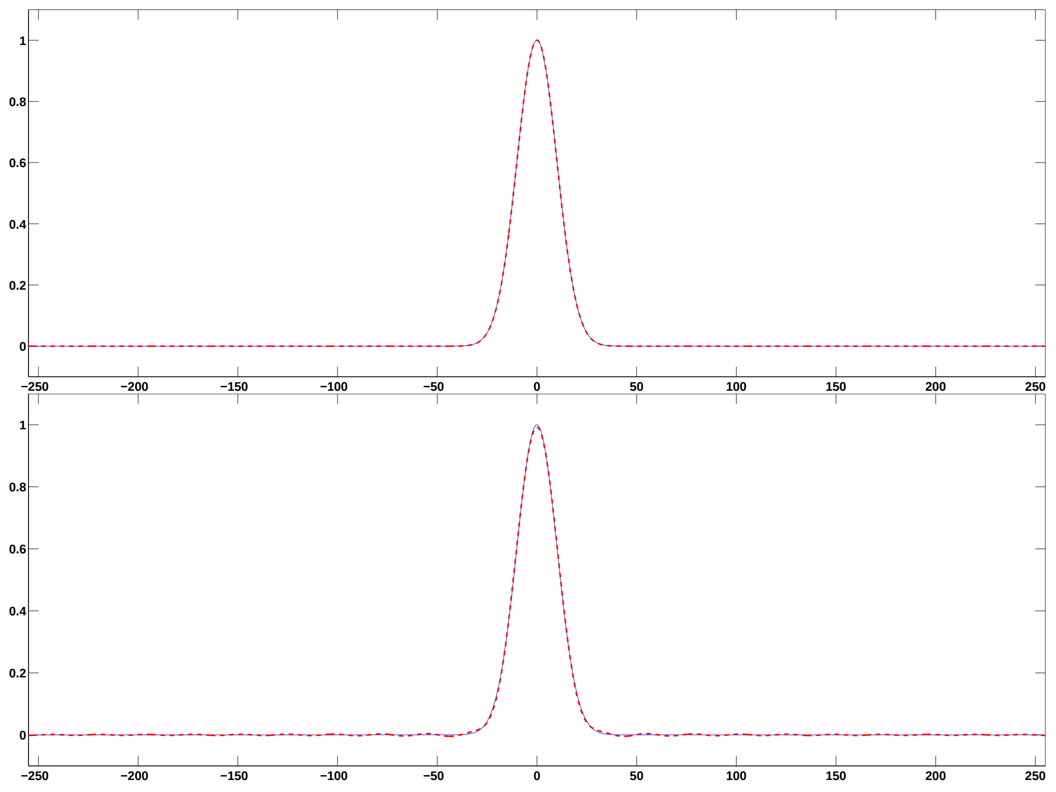

Figure 1: **Top**: Approximation of the target Gaussian $g_{\sigma_r}(s)$ using the raised cosine $\phi_N(s)$ on the interval $[-255, 255]$. We used a low value of $\sigma_r = 10$ in this case. This resulted in a very high threshold of $N_0 = 264$. The solid blue line shows the target Gaussian $g_{\sigma_r}(s)$ and the broken red line is the raised cosine $\phi_{N_0}(s)$. **Bottom**: The approximation $\phi_{N_0,M}(s)$ obtained after truncating $\phi_{N_0}(s)$. We used a tolerance of $\varepsilon = 0.005$, for which the M in (5) was found to be 111. Thus, a total of $2 \times M = 222$ terms were dropped from the series. The truncation $\phi_{N_0,M}(s)$, which has only 42 terms, is shown with a broken red line. The solid blue line is the target Gaussian. Note that the quality of approximation is reasonably good even after discarding almost 84% of the terms.