MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Constant Time O(1) Bilateral Filtering

Fatih Porikli

TR2008-030 July 2008

Abstract

This paper presents three novel methods that enable bilateral filtering in constant time O(1) without sampling. Constant time means that the computation time of the filtering remains same even if the filter size becomes very large. Our first method takes advantage of the integral histograms to avoid the redundant operations for bilateral filters with box spatial and arbitrary range kernels. For bilateral filters constructed by polynomial range and arbitrary spatial filters, our second method provides a direct formulation by using linear filters of image powers without any approximations. Lastly, we show that Gaussian range and arbitrary spatial bilateral filters can be expressed by Taylor series as linear filter decompositions without any noticeable degradation of filter response. All these methods drastically decrease the computational time by cutting it down constant times (e.g. to 0.06 seconds per 1MB image) while achieving very high PSNR's over 45dB. In addition to the computational advantages, our methods are straightforward to implement.

CVPR 2008

Constant Time $O(1)$ Bilateral Filtering

Fatih Porikli
Mitsubishi Electric Research Laboratories, Cambridge, MA

Abstract

This paper presents three novel methods that enable bilateral filtering in constant time $O(1)$ without sampling. Constant time means that the computation time of the filtering remains same even if the filter size becomes very large. Our first method takes advantage of the integral histograms to avoid the redundant operations for bilateral filters with box spatial and arbitrary range kernels. For bilateral filters constructed by polynomial range and arbitrary spatial filters, our second method provides a direct formulation by using linear filters of image powers without any approximation. Lastly, we show that Gaussian range and arbitrary spatial bilateral filters can be expressed by Taylor series as linear filter decompositions without any noticeable degradation of filter response. All these methods drastically decrease the computation time by cutting it down constant times (e.g. to 0.06 seconds per 1MB image) while achieving very high PSNR's over 45dB. In addition to the computational advantages, our methods are straightforward to implement.

1. Introduction

Fast realization of the edge preserving bilateral filter, which is a non-linear filter imposed both in domain and range, is important for many vision tasks. There are several methods to decompose certain nonlinear filters into a sum of separable one dimensional filters or cascaded representations [1]. This can be done by using of either an eigenvalue expansion of the 2D kernel [3] or application of Singular Value Decomposition [2]. There are many approaches [4], [5] that aim to benefit from the parallel processing platforms and reconfigurable hardware.

The bilateral filter was introduced by Tomasi et al. [6] as a non-iterative means of smoothing images while retaining edge detail. It involves a weighted convolution in which the weight for each pixel depends not only on its distance from the center pixel, but also its relative intensity. Elad [7] showed that the Bayesian approach is also in the core of the bilateral filter and described the bilateral filtering as a single iteration of the diagonal normalized steepest descent algorithm.

As nicely stated in [8], the fundamental property that concerns us is the runtime per pixel, as a function of the (spatial) filter size r. This corresponds to the performance a user will experience while adjusting the filter size (or kernel radius), and is the primary differentiating characteristic between bilateral filtering algorithms.

Due to the joint spatial and range filtering, the bilateral filters are computationally very demanding. For reference, a brute-force implementation can calculate each output pixel in $O(r^2)$ time and becomes unusably slow for even moderate radii. The well known Photoshop© CS2's Surface Blur implementation, which is a bilateral filter, exhibits $O(r)$ performance characteristic similar to Huang's column-row histograms used in median filters [9].

In their excellent benchmark paper, Paris and Durand [10] analyzed accuracy in terms of bandwidth and sampling, and derive criteria for downsampling in space and intensity to accelerate the bilateral filter by extending an earlier work on high dynamic range images [11]. Their method approximates the bilateral by filtering subsampled copies of the image with discrete intensity kernels, and recombining the results using linear interpolation. In other words, this method treats the intensity image as a 3D surface, applies Gaussian smoothing to binary and intensity modulated surface, divides them to determine the filtered intensity values at the original surface location. It becomes faster as the size increases due to the greater subsampling of the surface. The exact output is dependent on the phase of the subsampling grid and the discretization leads to further loss of precision particularly on high dynamic range images. This algorithm is dissected later in [12].

One of the fastest bilateral filter implementation whose runtime converges to $O(\log r)$ was developed by Weiss [8] using a hierarchy of partial distributed histograms using a tier-based approach. Even though complexity has been lowered, simplicity has been lost due to filter size and optimal histogram count specific implementation requirements. This method is limited to rectangular spatial kernels and box filters. Another concern is the imperfect frequency response of their spatial box filter.

Here, we describe a constant time bilateral filtering method. To our knowledge, the presented $O(1)$ algorithm is the most efficient bilateral filter yet developed. This means that it will perform better than one of higher complexity as the kernel size increases. Given the trend toward higher-resolution images, which will correspondingly require higher filter kernel sizes, filtering in large kernels as fast as the small ones makes the described algorithm future-proof.

We construct an integral histogram and use the integral histogram to find the bilateral convolution response of a rectangular box filter with uniform domain kernel, where the intensity differences can be weighted with any arbitrary range function. The integral histogram enables computation of histograms of all possible kernels in a given image. It takes advantage of the spatial positioning of data points in a Cartesian coordinate system, and propagate an aggregated function starting from an origin point and traversing through the remaining points along a scan-line. Histograms of image windows can be computed easily by using the integral histogram values at the corner points of those windows without reconstructing a separate histogram for every single one of them. For more generic Gaussian and polynomial range functions on arbitrary domain kernels, we apply Taylor series expansion of the corresponding norms. This second method can use "any" spatial kernel for bilateral filtering without increasing the complexity. We show that such bilateral filters can be expressed in terms of spatial linear filters applied on original image powers.

In the following section, we discuss the details of the adaptation of the integral histogram and Taylor series expansion. Then, we give a comparison of the computational complexity and present typical filtering results.

2. Bilateral Filtering

A filter f is a mapping defined in a d-dimensional Cartesian space $\mathcal{R}^d$. It assigns an m-dimensional response vector $y(\mathbf{p}) = [y_1, ..., y_m]$ to each point $\mathbf{p} = [x_1, ...x_d]$ using the given data I bounded within $N_1, ..., N_d$ and $0 \leq x_i < N_i$. Generally, only a small set of points within a *region of support* S is used to compute this response. The region of support, which is centered around the point $\mathbf{p}$, is also called as *kernel*. Without loss of generality, we consider the set of filters that maps to a scalar value, i.e. $m = 1$ and $y(\mathbf{p}) = y_1$. Even though we discuss single channel image filtering ($d = 2, m = 1$), the method presented here can be easily extended to higher dimensions, color images and temporal video filtering.

A 2D image filter centered at the image point $\mathbf{p}$ applies its coefficients $f(\mathbf{k})$ to the values of the underlying image points $\mathbf{k}+\mathbf{p}$ in its kernel $\mathbf{k} = [k_x, k_y] \in S$. For rectangular kernels, the coordinate of the center point can be assigned as the origin i.e. $S : -r/2 \leq k_x, k_y \leq r/2$ where r is the filter size. In case the values of the coefficients depend only on the spatial locations, the filter corresponds to a *spatial filter*. If the filter can be represented by a linear operator, e.g. as a matrix multiplication on its kernel, it is also called as a *linear filter*. For instance, a 2D Gaussian smoothing operator is a linear filter in which the coefficients' values change according to their spatial distances from the center point. Given the above notation, the response of a spatial filter can be expressed as

$$y(\mathbf{p}) = \kappa^{-1} \sum_{\mathbf{k} \in S} f(\mathbf{k}) I(\mathbf{p} + \mathbf{k}) \qquad (1)$$

where $\kappa = \sum f(\mathbf{k})$ is a scalar term to avoid bias. Note that, the above equation is same as the convolution of f and I. For simplicity, the filter is often normalized; $\sum f(\mathbf{k}) = 1$.

Bilateral filters, on the other hand, combine both *spatial* and *range* filtering. The coefficients of a *range filter* $g(\mathbf{p}, \mathbf{k})$ vary according to the intensity differences between the center and remaining points in the kernel instead of the spatial distance. The range filter is a function of the intensity difference i.e. $g(I(\mathbf{p})–I(\mathbf{p}+\mathbf{k}))$. In other words, a bilateral filter multiplies the intensity value of an image point in its kernel S by the corresponding spatial filter coefficient $f(\mathbf{k})$ and also a range filter coefficient $g(\mathbf{p}, \mathbf{k})$. Thus, the response of the bilateral filter is defined as

$$y_b(\mathbf{p}) = \kappa_b^{-1} \sum_{\mathbf{k} \in S} f(\mathbf{k}) I(\mathbf{p}+\mathbf{k}) g\left(I(\mathbf{p}) - I(\mathbf{p}+\mathbf{k})\right) \qquad (2)$$

where the normalizing term $\kappa_b = \sum f(\mathbf{k}) g(I(\mathbf{p})–I(\mathbf{p}+\mathbf{k}))$ is a scalar function of the intensity differences. As visible, the range filter may have a different value at each image point. Unlike the spatial filters, the normalizing term κ_b is not constant either. An illustration of the spatial and bilateral filters are given in Fig. 1. Due to the above range filtering property, the bilateral filter is *not* a linear filter and its response cannot be obtained by simple matrix multiplications. This is the main reason why it is computationally very demanding.

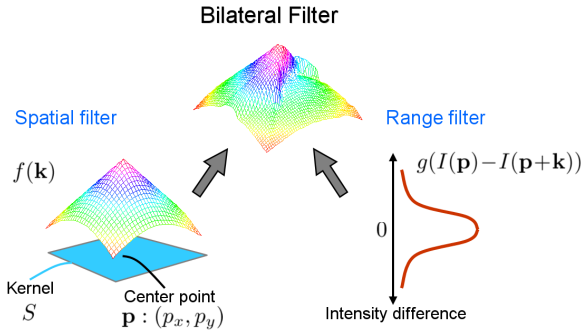

Figure 1. Bilateral filter has spatial and range components.

2.1. $O(1)$ Bilateral with Constant Spatial Filters

For now, lets assume we already have the intensity histogram h_p extracted for the current kernel S at an image point $\mathbf{p}$. In Section 2.4, we explain how to obtain such histograms of all possible spatial kernels in constant time.

We can rewrite Eq. 2 for a bilateral filter that have a constant spatial filter $f(\mathbf{k}) = c$ (box filter) and an arbitrary range filter $g(\mathbf{p}, \mathbf{k})$, which we call as ArBs bilateral, as

$$y_b(\mathbf{p}) = c\kappa_b^{-1} \sum_{\mathbf{k}\in S} I(\mathbf{p}+\mathbf{k}) g\left(I(\mathbf{p}) - I(\mathbf{p}+\mathbf{k})\right) \quad (3)$$

and $\kappa_b = \sum g(I(\mathbf{p}) - I(\mathbf{p}+\mathbf{k}))$. Fortunately, this response can be directly computed from the histogram h of the corresponding kernel as

$$y_b(\mathbf{p}) = c\kappa_b^{-1} \sum_i i h_p(i) g(I(\mathbf{p}) - i) \quad (4)$$

and

$$\kappa_b^{-1} = \sum_i h_p(i) g(I(\mathbf{p}) - i) \quad (5)$$

where the range function is accumulated over the bin values instead of the direct intensity differences. As shown, this exact formulation does not depend on the kernel size r. Even better, all of the scalar terms can be computed separately in constant time from the integral histogram [13]. In addition, any arbitrary range filter g including Gaussian and more complicated functions can be imposed.

Weiss's method [8] give the cost of the same ArBs bilateral filter in $O(\log r)$. Here, our method decreases this cost down to a constant time $O(1)$. This is the fastest bilateral filter that reported so far. Besides, his algorithm is limited to $r \leq 127$ filters. Our filter size is not limited (up to image size).

2.2. $O(1)$ Bilateral with Arbitrary Spatial Filters

The integral histogram based formulation cannot be applied to the bilateral filters that have non-constant spatial filters. Lets first consider a polynomial range filter that has the following definition

$$g(\mathbf{p}+\mathbf{k}) = [1 - (I(\mathbf{p}) - I(\mathbf{p}+\mathbf{k}))^2]^n \quad (6)$$

where n is the order of the polynomial. For $n = 1$, we can obtain the corresponding bilateral filter, which we call as PrAs, from Eq. 2 as

$$y_b(\mathbf{p}) = \kappa_b^{-1} \Big[\sum f(\mathbf{k}) I(\mathbf{p}+\mathbf{k}) - I^2(\mathbf{p}) \sum f(\mathbf{k}) I(\mathbf{p}+\mathbf{k}) + 2I(\mathbf{p}) \sum f(\mathbf{k}) I^2(\mathbf{p}+\mathbf{k}) - \sum f(\mathbf{k}) I^3(\mathbf{p}+\mathbf{k})\Big] \quad (7)$$

By denoting the power images as $I^1 = I(\mathbf{p}), I^2 = I(\mathbf{p})I(\mathbf{p})$, etc. and their corresponding linear filter responses $y_1 = \sum f(\mathbf{k}) I(\mathbf{p}+\mathbf{k})$, $y_2 = \sum f(\mathbf{k}) I^2(\mathbf{p}+\mathbf{k})$, etc., the above Eq. 7 can be rewritten as

$$y_b = \kappa_b^{-1} \left[(1-I^2) y_1 + 2I y_2 - y_3\right] \quad (8)$$

where we dropped the index $\mathbf{p}$ from the right-side of the equation for simplicity. Similarly, the normalizing term can be found as $\kappa_b = 1 - I^2 + 2I y_1 - y_2$. Note that, the spatial filter f is not constrained and any desired filter function can be chosen.

For quadratic polynomial range function ($n = 2$), PrAs bilateral filter of Eq. 2 can be written in the same manner as

$$y_b = \kappa_b^{-1} \big[(1 - 2I^2 + I^4) y_1 + 4(I - I^3) y_2 + (6I^2 - 2) y_3 - 4I y_4 + y_5\big] \quad (9)$$

where $\kappa_b^{-1} = 1 - 2I^2 + I^4 + 4(I - I^3) y_1 + [6I^2 - 2) y_2 - 4I y_3 + y_4$. Equations 8, 9 give the corresponding bilateral filters with polynomial range filters in terms of the spatial filters without any approximations.

Another common type of bilateral filters, GrAs, use Gaussian range filters for additional smoothness. We apply the Taylor series expansion of the Gaussian function to approximate such bilateral filters. This method again does not have any restriction on the spatial filter. Gaussian filters are differentiable and can be expressed in terms of linear transforms. The Gaussian range filter is given by

$$\exp\left(-\alpha [I(\mathbf{k}+\mathbf{p}) - I(\mathbf{p})]^2\right). \quad (10)$$

Above equation can be rewritten as

$$\exp(-\alpha I^2(\mathbf{p})) \exp\left(-\alpha [I^2(\mathbf{p}+\mathbf{k}) - 2I(\mathbf{p}) I(\mathbf{p}+\mathbf{k})]\right) \quad (11)$$

where the first term $\exp(-\alpha I^2(\mathbf{p}))$ does not depend on the range kernel. This term also appears in the normalizing term, thus, it does not have to be computed separately.

By applying the Taylor expansion to Eq. 11, we obtain the bilateral filter expansion up to the second order derivatives as

$$y_b \approx \kappa_b^{-1} \big[y_1 + 2\alpha I y_2 + \alpha(2\alpha I^2 - 1) y_3 - 2\alpha^2 I y_4 + 0.5\alpha^2 y_5\big] \quad (12)$$

and, up to third order derivatives as

$$y_b \approx \kappa_b^{-1} \Big[y_1 + 2\alpha I y_2 + (2\alpha^2 I^2 - \alpha) y_3 - 2\alpha^2 (I - \frac{2}{3}\alpha I^3) y_4 + \alpha^2 (0.5 - 2I^2\alpha) y_5 + \alpha^3 I y_6 - (\alpha^3/6) y_7\Big] \quad (13)$$

where the normalizing terms have similar forms containing the same terms.

Therefore, a bilateral filter can be interpreted as the weighted sum of the spatial filtered responses of the powers of the original image.

2.3. Constant Time Spatial Filters

There are various ways of computing the 2D spatial linear filter responses. The box filter, also known as 'moving average' is a simple linear filter with a rectangular kernel where all kernel coefficients are equal. This filter can be easily computed in constant time $O(1)$ by using an integral image. An integral image I_Σ is the accumulated sum of original image intensities. The sum of any region $\sum I(\mathbf{p})$ can be found by only three arithmetic operations involving the values of the integral image at the corners $\mathbf{p}^{++}, \mathbf{p}^{-+}, \mathbf{p}^{+-}, \mathbf{p}^{--}$ of the region, e.g. $\sum I(\mathbf{p}) = I_\Sigma(\mathbf{p}^{++}) - I_\Sigma(\mathbf{p}^{+-}) - I_\Sigma(\mathbf{p}^{-+}) + I_\Sigma(\mathbf{p}^{--})$.

The ramp filters, e.g. triangular filter or Bartlett window can be constructed as a superposition of two box filters with the same size. The computational complexity of a ramp filter is twice the complexity of a box filter, thus they can be applied in constant time $O(1)$, and the results are visually very similar to Gaussian filter.

The response of the polynomial filters $f(\mathbf{k}) = 1 - \mathbf{k}^n$ can also be computed in constant time $O(1)$ using a set of integral images. For square distance norm, we get

$$\begin{aligned} y_1(\mathbf{p}) &= \sum_{k\in S} f(\mathbf{k}) I(\mathbf{p}+\mathbf{k}) = \sum_{\mathbf{z}\in S^z} f(\mathbf{z}-\mathbf{p}) I(\mathbf{z}) \\ &= \sum_{\mathbf{z}\in S^z} (1-(\mathbf{z}-\mathbf{p})^2) I(\mathbf{z}) \\ &= \left[1-\mathbf{p}^2\right] \sum_{\mathbf{z}\in S^z} I(\mathbf{z}) + 2\mathbf{p} \sum_{\mathbf{z}\in S^z} \mathbf{z} I(\mathbf{z}) - \sum_{\mathbf{z}\in S^z} \mathbf{z}^2 I(\mathbf{z}) \end{aligned}$$

where S^z is the new kernel around $\mathbf{z}-\mathbf{p}$. The sums $\sum I(\mathbf{z})$, $\sum \mathbf{z} I(\mathbf{z})$, $\sum \mathbf{z}^2 I(\mathbf{z})$ can be computed directly from the corresponding integral images. Since these sums require only fixed number of operations at the corner points of the rectangular regions in integral images, the total computation time is independent from the region size. The complexity is $O(1)$. This is valid for bilinear interpolating filters too.

Several other linear spatial filters can be computed by FFT in constant time in terms of the filter size [14]. Per-point computation complexity of the underlying FFT algorithms depends only on the padded image size. For each big enough input image size, starting from certain convolution kernel size, FFT-based convolution is more advantageous than a straightforward implementation. Gaussian filter on a square kernel is separable, i.e. 2D filtering it can be decomposed into a series of 1D filtering. When the filter size is relatively small (less than few dozens), the fastest way to calculate the filtering result is direct 1D convolution. The filter symmetry can be exploited to reduce the number of multiplications by a factor of 2. When a filter size is large, direct convolution becomes expensive, and FFT-based convolution is the best choice.

To guarantee a constant time processing, we also propose to subsample the separable 1D linear spatial filters to a constant 15 taps asymmetrically (more taps towards center), which is shown to be sufficiently accurate in our experiments.

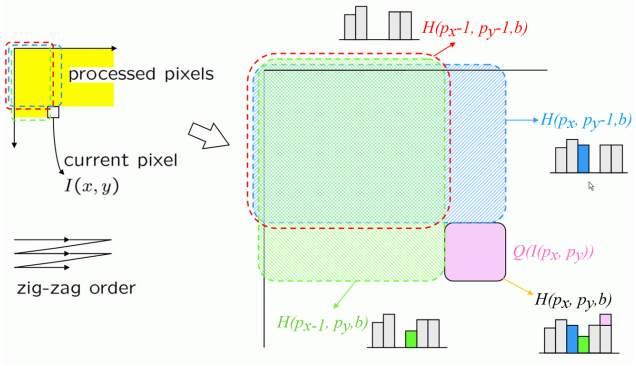

Figure 2. Propagation of integral histogram. Yellow indicates already traversed points. At each step, the current integral histogram is obtained from the integral histogram values of the three neighbors, and the bin that corresponds to current point's value is increased by one.

2.4. Integral Histograms

The integral histogram first introduced by Porikli [13]. It involves a propagation of point-wise histograms on a sequence of image points followed by an intersection of (four) histograms to determine the histogram of any (rectangular) regions.

Integral histogram $H(\mathbf{p}^m, b)$ where $b = 1..B$ at an image point at the m^{th} position along a sequence of points $\mathbf{p}^0, \mathbf{p}^1, .., \mathbf{p}^m$ is defined as

$$H(\mathbf{p}^m, b) = \bigcup_{j=0}^{m} Q(I(\mathbf{p}^j)) \tag{14}$$

where $Q(.)$ is the corresponding bin of the current point, and $\cup$ is the union operator that is defined as follows: the value of the bin b of $H(\mathbf{p}^m, b)$ is equal to the sum of the previously visited points's histogram bin values, that is the sum of all $Q(I(\mathbf{p}^j))$ while $j < m$. In other words, $H(\mathbf{p}^m, b)$ is the histogram of the region between the origin and current point; $0 \le p_x^j \le p_x^m$, $0 \le p_y^j \le p_y^m$. Note that, $H(\mathbf{p}^N, b)$ is equal to the histogram of all data points since $\mathbf{p}^N = [N_1, N_2]$ is the last point in the image. Therefore, the integral histogram can be written recursively as

$$\begin{aligned} H(p_x, p_y, b) = H(p_x-1, p_y, b) + H(p_x, p_y-1, b) \\ -H(p_x-1, p_y-1, b) + Q(I(p_x, p_y)). \end{aligned} \tag{15}$$

using the initial condition $H(0, 0, b) = 0$, which means all the bins are empty at the origin.

The scan requires updating the integral histogram for such data points that their left, upper, and upper-left neigh-

bors are already scanned in case of an image data. The integral histogram at a point is obtained by three arithmetic operations for each bin of using the integral histogram values of the three neighbors as shown in Fig. 2. The integral histogram values of the previous point is copied to the current point before the propagation.

The histogram of a region T can be computed using the propagated integral histogram values at the boundary points (p_x^+, p_y^+), (p_x^-, p_y^+), (p_x^+, p_y^-), (p_x^-, p_y^-) of the region simply as

$$h(T,b) = H(p_x^+, p_y^+, b) - H(p_x^-, p_y^+, b) - H(p_x^+, p_y^-, b) + H(p_x^-, p_y^-, b). \quad (16)$$

As opposed to the conventional histogram computation, the integral histogram method does not repeat the histogram extraction for each possible region, thus histogram extraction is not depend on the filter kernel size r.

3. Experiments

We tested our filters wits several 1 MB gray-level and color images depicting typical scenes of natural landscapes, human faces, architecture, etc. as shown in Figs. 7 and 9 top rows. To evaluate numerical accuracy, we use the peak signal-to-noise ratio (PSNR). For two intensity images $y_1, y_2 = [0;1]$, this ratio is defined as $10\log_{10}(N_1 N_2 / \sum_{\mathbf{p}} |y_1(\mathbf{p}) - y_2(\mathbf{p})|^2)$. Considering intensity values encoded on 8 bits, if two images differ from one gray level at each pixel, the resulting PSNR is 48dB. It is assumed [10] the PSNR values above 40dB often corresponds to almost invisible differences, thus, we selected it as a quality threshold.

We compared our implementations against PhotoShop CS2, which features an implementation of Huangs $O(r)$ algorithm [9], against Pixfoliate, a PhotoShop plugin distributed by Weiss implementing his $O(\log r)$ algorithm [8], and against full kernel (FFT convolution) C++ implementations provided by Paris and Durand for their sampling based approach [10]. Timing was conducted on a PowerMac G5 1.6 GHz for Weiss's method, and on a P4 3.2 GHz for the others.

Paris and Durand's method provides low computation times especially for larger kernels thanks to the downsampling, where small sampling ratios correspond to limited approximations and high ratios to more aggressive downsamplings. We kept the sampling rate proportional to the Gaussian bandwidth ($s_s/\sigma_s \approx s_r/\sigma_r$) in our tests as recommended in [10]. This method also has a truncated, faster version that uses spatial convolution.

The computation times are given in Fig. 3 in log-log scale. As visible, our methods have clearly $O(1)$ time complexity and they are significantly faster than the other approaches.

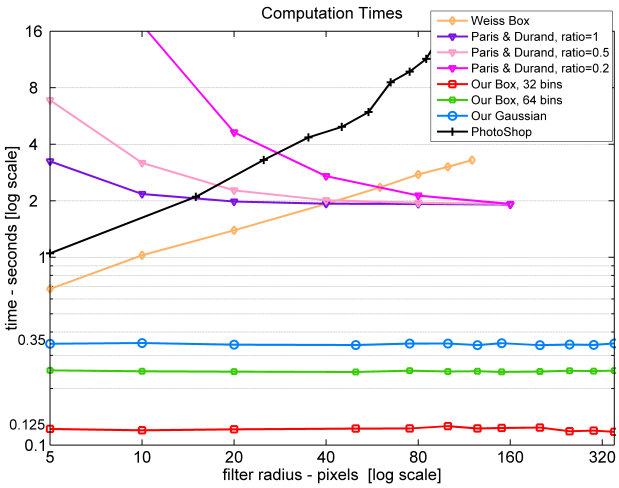

Figure 3. Our $O(1)$ methods have faster processing times.

As described in the previous section, there are three $O(1)$ versions: I) ArBs; bilateral filters with arbitrary range (including Gaussian, etc.) and the box spatial filters (Eqns. 4, 5, 16), II) PrAs; bilateral filters with polynomial range and arbitrary spatial filters (Eqs. 8, 9), III) GrAs; bilateral filters with Gaussian range and arbitrary spatial filters (Eqns. 12, 13).

For ArBs, we construct the integral histograms for the whole image. This takes ~30 millisecond for 32 bins and ~60 millisecond for 64 bins integral histograms on average per 1 MB single channel image with MSVC++ compiler. To compute the response for any given spatial filter size, we apply the intersection rule (Eq. 16) and find the weighted sum of the histogram with the range kernel. This takes approximately 0.06 seconds per 1 MB image for 16 bins, 0.125 seconds for 32 bins and 0.25 seconds for 64 bins. We analyzed the accuracy by comparing our results with the exact filter. The corresponding PSNR results are given in Fig. 4. As visible, even the low resolution (e.g. $B = 16$) integral histograms provide remarkably high PSNR values (with PSNR's over 45 dB). Results above this threshold are visually very similar to the exact filter responses. We give the ArBs filter results in the middle rows of Figs. 7, 9. The range filters are set as Gaussian functions with $\sigma_r^2 = 0.15$ and $\sigma_r^2 = 0.025$ with spatial kernel sizes $r = 31$ and $r = 21$, respectively.

Note that, the integral histogram does not restrict the spatial filter size r when the bilateral filter is applied to each image point. In other words, it is possible to *change the filter size adaptively* at each point while sweeping through the image if it is desired.

For PrAs bilateral filters, the proposed linear filter based results are almost identical with the exact versions. These filters use subsampled 1D linear filters at 15 taps. The com-

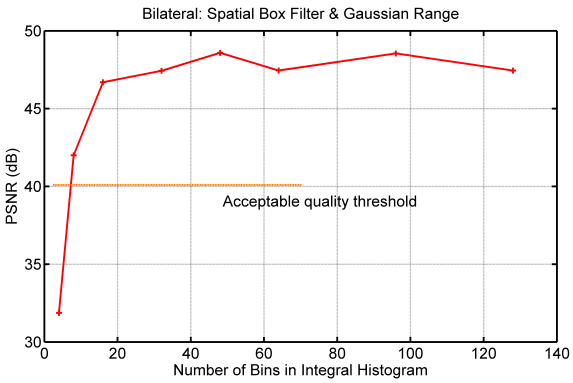

Figure 4. PSNR accuracy of the presented $O(1)$ bilateral filter (with box spatial) given in Eq. 4 in comparison to the exact filter. As visible, even the low bins, the integral histogram based method provides remarkably good results above the threshold.

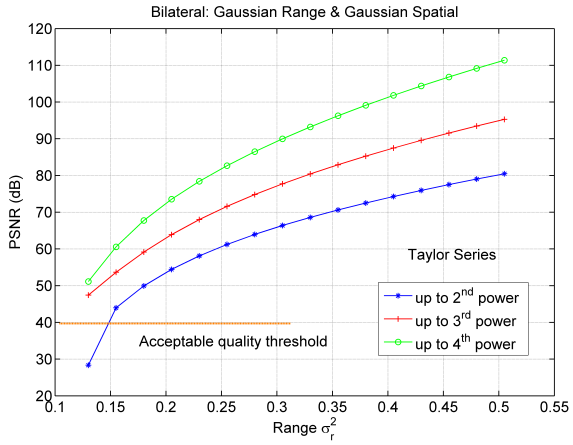

Figure 5. Accuracy of $O(1)$ bilateral filter (with Gaussian spatial and Gaussian range) in comparison to the exact filter. Colors indicate different powers of derivatives (Eqns. 12, 13,etc) in Taylor series. For smoother range functions (larger σ_r^2), the proposed filter becomes almost identical with the exact filter.

putation time varies between $0.2 \sim 0.3$ seconds depending on the spatial filter shape and independent of the filter size.

For GrAs bilateral filters, we applied Taylor series expansion with linear subsampled filter. To evaluate the approximation accuracy, we compared the filter responses against the exact versions. Figure 5 shows PSNR graphs of bilateral filters with Gaussian spatial ($r = 15, \sigma_s^2 = 1$) and Gaussian range ($0.1 < \sigma_r^2 \leq 0.5$) filters at different powers the derivatives in Taylor series (Eqns. 12, 13, and fourth order). We observed for smoother range functions, i.e. $\sigma_r^2 > 0.12$), the proposed filter becomes almost identical with the exact filter with PSNR's well above 50dB. However, for smaller variance values, the approximations are not valid. The algorithm is still $O(1)$, albeit with a higher constant by requiring $\sim$0.35 second per 1 MB image. Sample GrAs filter results are given in the bottom row of the Fig. 9. The range filters are set as Gaussian functions $\sigma_r^2 = 0.13$ with a Gaussian spatial kernel $\sigma_s^2 = 1, r = 21$.

Figure 6. Filters have almost identical responses as the exact ones.

Figures 6 and 8 show close up views, and effect of σ_r^2 respectively. Even though we decreased the computation time to a fraction of the other methods and kept the complexity in constant time with respect to the filter size, the results are very similar to exact filter responses.

The computation times given above is for single channel gray-level images. To process color images, we applied the same filter to each channel separately and combined the independent channel responses into a single color vector. We used a 128-bins histogram for ArBs for each channel. For the baseline exact bilateral filter, which are utilized in PSNR comparisons, we constructed the responses in the RGB color space by computing the L_2 vector distance in the range filter and scaling the color coefficients proportionally. The computation times of the color images are tripled. We observed the PSNR's are still well above 40dB.

On Distributed Histograms

Weiss' main idea is to operate on multiple columns at once in the filtering time, as they are all using overlapping kernels. Given a multi-column operating framework, it stores positive values for the pixels not in column i's histogram and negative values for the pixels in column i's histogram that should not appear in column $i+1$'s histogram. This results in histograms adjacent to the base only keeping track of one positive column and one negative column, and the histograms r distance away keeping track of r positive columns and r negative columns. The runtime of this is $O(r)$ because the outer-most histograms are not gaining a lot by sharing very few columns with the base and there is overhead in maintaining a group of partial histograms,

Figure 8. Effect of different range kernel variances for ArBs box filter: original, $\sigma_r^2 = 0.02$, $\sigma_r^2 = 0.2$ at $r = 10$.

especially when r is large.

When it is extended to include multiple tiers of base histograms, it corresponds to a large shared base histogram on the first tier, a few medium sized partial histograms spaced 7 points apart on the second tier, and many tiny partial histograms at single pixel increments on the third tier. In a completely theoretical analysis of filtering of huge radii, keeping hundreds of thousands of partial histograms on 7 tiers, the paper claims the runtime to converge to $O(\log r)$. It also requires keeping one large complete dictionary in the center column of the sweep to determine if the given neighbor points is outside r columns from the center point. However, maintaining these dictionaries is a lot slower than the histograms. Much more time is spent calculating the bilateral filter given the neighborhood as well as managing the dictionaries that the window movement cost is not nearly as significant [15].

4. Conclusion

We described three methods that enable constant time bilateral filtering for gray-level and color images. In addition to being independent of the filter size, our computation times are among the fastest reported so far. Our contributions are threefold:

- We introduced an integral histogram based constant time bilateral filtering method for ArBs bilateral filters with arbitrary range (including Gaussian, etc) and box spatial filter. This method takes advantage of the overlapping kernels to avoid redundant operations. It is accurate (PSNR $>$ 45dB) and extremely fast. It also enables setting spatial filter size *adaptively* at each point.
- We derived formulations for constant time PrAs bilateral filters that have polynomial range and arbitrary spatial filters by linear filters. These filters give identical responses as their exact versions. As above, these filters are very fast; processing time is under 0.3 sec. for a 1 MB gray-level image independent of the filter size r.
- We showed constant time GrAs bilateral filters that have Gaussian range and arbitrary spatial filters (down to $\sigma_r = 0.13$) can be expressed by Taylor series, which transform non-linear bilateral filtering into linear filtering of image powers and adaptive setting of linear filter taps. This expansion is almost identical to the exact filter (PSNR $>$ 50dB).

Considering the trend toward higher-resolution images, which will correspondingly require larger filter kernel size, filtering in large kernels as fast as the small ones makes the described algorithms necessary and future-proof.

References

[1] W. Wells, "Efficient synthesis of Gaussian filters by cascaded uniform filters", *IEEE Trans. Pattern Anal. Machine Intell.*, vol.8, 234-239, 1986. 1

[2] W. S. Lu, H. P. Wang, A. Antoniou, "Design of 2D FIR digital filters by using the singular value decomposition", *IEEE Trans. Circuits Syst*, vol.37, 35-36, 1990. 1

[3] J. M. Geusebroek, A. Smeulders, J. Weijer, "Fast anisotropic Gauss filtering", *IEEE Transaction on Image Processing*, vol.12:8, 2003. 1

[4] J. Torresen, J. W. Bakke, L. Sekanina, "Efficient image filtering and information reduction in reconfigurable logic", *In Proc. 22nd Norchip Conference*, 2004. 1

[5] C. Sigg, M. Hadwiger, "Fast third-order texture filtering", *In Matt Pharr, editor, GPU Gems 2: Programming Techniques for High-Performance Graphics*, 313-329, 2005. 1

[6] C. Tomasi, R. Manduchi, "Bilateral filtering for gray and color images", *In Proc. International Conference on Computer Vision*, 839846. 1998. 1

[7] M. Elad, "On the bilateral filter and ways to improve it", *IEEE Transactions on Image Processing*, vol.11, 1141-1151, 2002. 1

[8] B. Weiss, "Fast median and bilateral filtering", *In Proc. SIGGRAPH*, 2006. 1, 3, 5

[9] T.S. Huang "Two-Dimensional Signal Processing II: Transforms and Median Filters". Berlin: Springer-Verlag, 209-211, 1981. 1, 5

[10] S. Paris, F. Durand, "A fast approximation of the bilateral filter using a signal processing approach", *In Proc. European Conference on Computer Vision*, 2006. 1, 5

[11] F. Durand, J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images", *ACM Transactions on Graphics*, vol.21, 2002. 1

[12] G. Guarnieri, S. Marsi, G. Ramponi, "Fast bilateral filter for edge-preserving smoothing", *Electronics Letters*, vol.42, 396-397, 2006 1

[13] F. Porikli, "Integral histogram: a fast way to extract histograms in Cartesian spaces", *In Proc. Computer Vision and Pattern Recognition*, vol.1, 829-836, 2005. 3, 4

[14] I. Young, J. Gerbrands and L. van Vliet, "Fundamentals of Image Processing", *Delft University of Technology*, 1995. 4

[15] C. Robson, "An implementation of fast median and bilateral filtering", it http://pages.cs.wisc.edu/ robson/vision/, 2007. 7

Figure 7. **Top** Original images. **Bottom** $O(1)$ ArBs bilateral with box spatial and Gaussian range ($\sigma_r^2 = 0.15$, $f : 31 \times 31$). PSNR is 301.88dB (i.e. almost identical to the exact filter result). Images are 1 MB.

Figure 9. **Top** Original color images. **Middle** $O(1)$ ArBs bilateral with box spatial and Gaussian range ($\sigma_r^2 = 0.05$, $f : 21 \times 21$). PSNR is 50.61dB. **Bottom** $O(1)$ GrAs bilateral with Gaussian spatial ($\sigma_s^2 = 1$, $f : 10$) and Gaussian range ($\sigma_r^2 = 0.13$). PSNR is 55.12dB. Images are 1 MB. Each channel is processed independently.