# Efficient and Accurate Gaussian Image Filtering Using Running Sums

Elhanan Elboher and Michael Werman

***Abstract*—This paper presents a simple and efficient method to convolve an image with a Gaussian kernel. The computation is performed in a constant number of operations per pixel using running sums along the image rows and columns. We investigate the error function used for kernel approximation and its relation to the properties of the input signal. Based on natural image statistics we propose a quadratic form kernel error function so that the output image $l_2$ error is minimized. We apply the proposed approach to approximate the Gaussian kernel by linear combination of constant functions. This results in very efficient Gaussian filtering method. Our experiments show that the proposed technique is faster than state of the art methods while preserving a similar accuracy.**

***Index Terms*—Non uniform filtering, Gaussian kernel, integral images, natural image statistics.**

## I. Introduction

IMAGE filtering is an ubiquitous image processing tool, which requires fast and efficient computation. When the kernel size increases, direct computation of the kernel response requires more operations and the process becomes slow.

Various methods have been suggested for fast convolution with specific kernels in linear time (see related work in Section II). An important kernel is the Gaussian, which is used in many applications.

In this paper we present an efficient filtering algorithm for separable non uniform kernels and apply it for very fast and accurate Gaussian filtering. Our method is based on one dimensional *running sums* (*integral images*) along single rows and columns of the image. The proposed algorithm is very simple and can be written in a few lines of code. Complexity analysis, as well as experimental results show that it is faster than state of the art methods for Gaussian convolution while preserving similar approximation accuracy.

An additional contribution of this paper is an analysis of the relation between the kernel approximation and the approximation of the final output, the filtered image. Usually, the approximation quality is measured in terms of the difference between the kernel and its approximation. However, minimizing the kernel error does not necessarily minimize the error on the resulting image. To minimize this error, natural image statistics should be considered.

In Section IV we investigate the kernel approximation error function. Based on natural image statistics we find a quadratic form kernel error measurement which minimizes the $l_2$ error on the output pixel values.

The next section reviews related methods for fast non uniform image filtering. Section III presents the filtering algorithm and discusses its computational aspects. Section IV discusses the relation between kernel approximation and natural image statistics. Section V presents our experimental results. We conclude in Section VI.

E. Elboher and M. Werman are with the Hebrew University of Jerusalem, Israel.

## II. Related Work

In the following we review the main approaches to accelerate image filtering. We describe in more detail the integral image based approach on which the current work is based.

**General and Multiscale Approaches.** Convolving an image of $N$ pixels with an arbitrary kernel of size $K$ can be computed directly in $O(NK)$ operations, or using the Fast Fourier Transform in $O(N \log K)$ operations [1].

A more efficient approach is the linear time multiscale computation using image pyramids [2], [3]. The coarser image levels are filtered with small kernels and the results are interpolated into the finer levels. This approximates the convolution of the image with a large Gaussian kernel.

**Recursive Filtering.** The recursive method is a very efficient filtering scheme for one dimensional (or separable) kernels. The infinite impulse response (IIR) of the desired kernel is expressed as a ratio between two polynomials in Z space [4]. Then the convolution with a given signal is computed by difference equations. Recursive algorithms were proposed for approximate filtering with Gaussian kernels [5], [6], [7], [8], [9], [10], anisotropic Gaussians [11] and Gabor kernels [12].

The methods of Deriche [5], [6], [7] and Young and van Vliet [8], [9] are the current state of the art for fast approximate Gaussian filtering. Both methods perform two passes in opposite directions, in order to consider the kernel response both of the forward and backward neighbors. The method of Young and van Vliet's requires less arithmetic operations per pixel. However, unlike Deriche's method the two passes cannot be parallelized.

Tan et al. [13] evaluated the performance of both methods for small standard deviations using normalized RMS error. While Deriche's impulse response is more accurate, Young and van Vliet performed slightly better on a random noise image. No natural images were examined. Section V provides further evaluation of these methods.

**Integral Image Based Methods.** Incremental methods such as *box filtering* [14] and *summed area tables*, known also as *integral images* [15], [16], cumulate the sum of pixel values along the image rows and columns. In this way the sum of a rectangular region can be computed using $O(1)$ operations independent of its size. This makes it possible to convolve an image very fast with uniform kernels.

Heckbert [17] generalized integral images for polynomial kernels of degree $d$ using $d$ repeated integrations. Derpanis et al. [18] applied this scheme for a polynomial approximation of the Gaussian kernel. Werman [19] introduced another generalization for kernels which satisfy a linear homogeneous equation (LHE). This scheme requires $d$ repeated integrations, where $d$ is the LHE order.

Hussein et al. [20] proposed Kernel Integral Images (KII) for non uniform filtering. The required kernel is expressed as a linear combination of simple functions. The convolution with each such functions is computed separately using integral images. To demonstrate their approach, the authors approximated the Gaussian kernel by a linear combination of polynomial functions based on the Euler expansion. Similar filtering schemes were suggested by Marimon [21] who used a combination of linear functions to form pyramid shaped kernels and by Elboher and Werman [22] who used a combination of cosine functions to approximate the Gaussian and Gabor kernels and the bilateral filter [23].

**Stacked Integral Images.** The most relevant method to this work is the Stacked Integral Images (SII) proposed by Bhatia et al. [24]. The authors approximate non uniform kernels by a 'stack' of box filters, i.e. constant 2D rectangles, which are all computed from a *single* integral image. The simplicity of the used function and not using multiple integral images makes the filtering very efficient.

The authors of SII demonstrated their method for Gaussian smoothing. However, they approximated only specific 2D kernels, and found for each of them a local minima of a non-convex optimization problem. Although the resulting approximations can be rescaled, they are not very accurate (see Section V). Moreover, the SII framework does not exploit the separability of the Gaussian kernel.

As shown in this paper, utilizing the separability property we find an optimal kernel approximation which can be scaled to any standard deviation. As shown in Figure 2, this approximation is richer and more accurate. Actually, separable filtering of the row and the columns by $k$ one dimensional constants is equivalent to filtering by $2k-1$ two dimensional boxes. The separability property can also be used for an efficient computation both in time and space, as described in Section III.

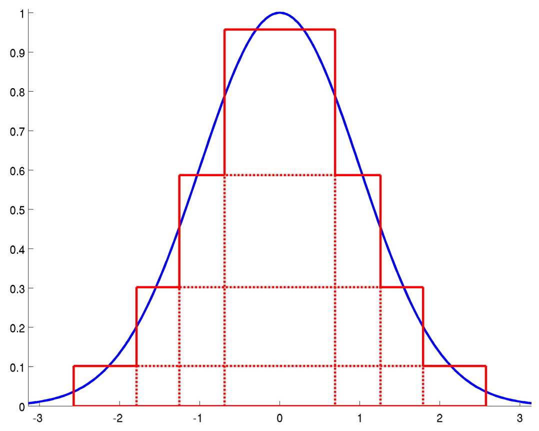
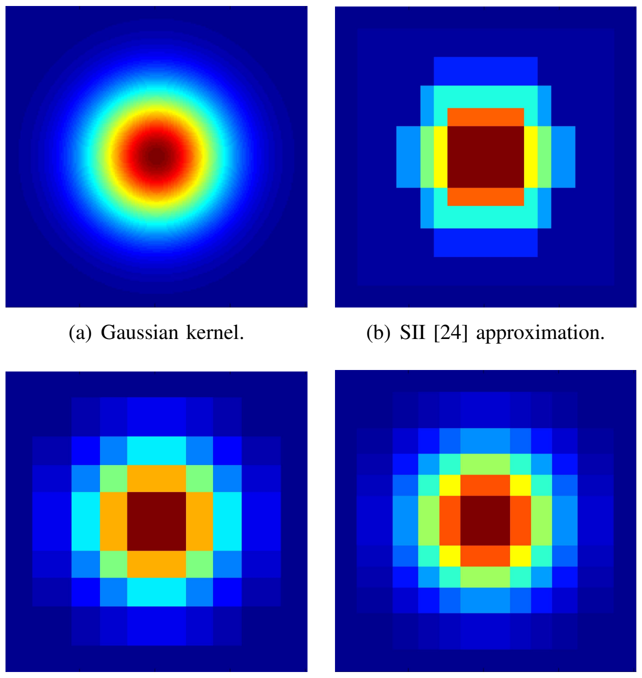


Fig. 1. Approximation of the 1D Gaussian kernel on $[-\pi,\pi]$ by $k=4$ piecewise constant functions. The dashed lines show the equivalence between 4 constant 'slices' and 7 constant 'segments' (Equation 4).



(a) Gaussian kernel.

(b) SII [24] approximation.

(c) Proposed approximation (4 constants).

(d) Proposed approximation (5 constants).

Fig. 2. Comparison of (a) exact Gaussian kernel, (b) Stacked Integral Images [24] with 5 2D boxes, and the proposed method with 4 constants (c) and 5 constants (d). Our proposed approximation is richer and more accurate since it utilizes the Gaussian separability. Instead of using 2D boxes, we use 1D segments to filter the rows and then the columns.

## III. Algorithm

### A. Piecewise Constant Kernel Decomposition

Consider the convolution $f*K$ of a function $f$ with a kernel $K$. For simplicity we first discuss the case in which $f$ and $K$ are one dimensional. In the following (Section III-D) we generalize the discussion to higher dimensions.

Suppose that the support of the kernel $K$ is $r$, i.e. that $K$ is zero outside of $[0,r]$. Assume also that we are given a partition $P=(p_0,p_1,...p_k)$, in which $0\le p_0<p_1<p_2\ ...<p_{k-1}<p_k\le r$. Thus, the kernel $K$ can be approximated by a linear combination of $k$ simple functions $K_i$, $i=1...k$:

$$K_i(t)=\begin{cases} c_i & \text{if } p_{i-1}\le t<p_i \\ 0 & \text{otherwise}\end{cases} \tag{1}$$

Using the above approximation

$$f*K\approx\sum_{i=1}^{k} f*K_i \tag{2}$$

which can be computed very efficiently, as described in Section III-C.

### B. Symmetric Kernels

Consider the case in which $K$ is a *symmetric* kernel on $[-r,r]$. The approximation can be limited to the range $[0,r]$.

| Method | Additions | Multiplications |
|---|---|---|
| Direct | $h+w-2$ | $h+w$ |
| FFT | $O(log(hw))$ | $O(log(hw))$ |
| KII [20] | 53 | 18 |
| CII [22] | $12k-8$ | $8k-8$ |
| SII [24] | $4k+1$ | $k$ |
| Deriche [7] | $8k-2$ | $8k$ |
| Young and van Vliet [8] | $4k$ | $4k+4$ |
| Proposed method | $4k$ | $2k$ |

TABLE I
ARITHMETIC OPERATIONS PER IMAGE PIXEL (SECTION III-D).

Each constant $c_i$ can be used both in the negative interval $[-p_i, -p_{i-1}]$ and in the positive one $[p_{i-1}, p_i]$, however, this requires $2k$ constant function. Actually, the same approximation can be computed using 'weighted slices':

$$S_i(t) = \begin{cases} w_i & \text{if } -p_i < t < p_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $w_i = c_i - c_{i-1}$. Now the kernel $K$ is approximated by sum of $k$ constant functions which are non zero in the overlapping intervals $[-p_i, p_i]$, with weights $w_i = c_i - c_{i-1}$. The approximation of $K$ remains the same:

$$\begin{aligned} \sum_{i=1}^{k} S_i(t) &= \sum_{i:-p_i<t<p_i} w_i = \sum_{i:-p_i<t<p_i} (c_i - c_{i-1}) \\ &= \sum_{i:p_{i-1}\leq|t|<p_i} c_i = \sum_{i=1}^{k} K_i(t) \end{aligned} \tag{4}$$

This equality is illustrated in Figure 1.

### C. 1D Filtering Algorithm

In the following we describe the algorithm for the case of a *symmetric* kernel $K$ (Section III-B).

Given a 1D discrete function $f(x)$ and $k$ piecewise constant functions $S_i$, we compute the approximated convolution (Equation 2) as follows:

1) Compute the cumulative sum of $f(x)$,

$$I(x) = \sum_{x'=0}^{x} f(x') \tag{5}$$

2) Compute the convolution result for each pixel $x$,

$$\sum_{i=1}^{k} w_i(I(x+p_i) - I(x-p_i-1)) \tag{6}$$

The total cost of steps 1 and 2 is $2k$ additions and $k$ multiplications per image pixel. In the 1D case memory access is not an issue, as all the elements are usually in the cache.

### D. Higher Dimensions and More Computational Aspects

We now describe the case of convolving a 2D image $f(x,y)$ with a *separable* 2D kernel $K$. A kernel $K$ is separable if it can be expressed as a convolution of 1D filters $K_x * K_y^T$. The convolution $f * K$ can be computed by first convolving the image rows with $K_x$ and then the columns of the intermediate result by $K_y$. Hence, filtering an image with a separable 2D kernel only doubles the computational cost of the 1D case.

The space complexity is also very low. Since the convolution of $K_x$ with each row (and also $K_y$ with each column) is independent, filtering an image of size $n \times m$ requires only $O(\max(n,m))$ additional space over the input and the output images for storing the cumulative sum of a single row or column.

Similarly to the 2D case, the filtering scheme can be extended to $d$-dimensional separable kernels using $2dk$ additions and $dk$ multiplications per single pixel. The additional required space is $O(\max_i(n_i))$, where $n_i$ $(i = 1...d)$ are the sizes of the signal dimensions.

Table I counts the required operations per image pixel for 2D image filtering by a $h \times w$ separable kernel. The parameter $k$ denotes the number of terms (constants, 2D boxes etc.) depending on the specific method. Notice that experimentally our proposed method with 3 constants is more accurate than SII [24] with 5 boxes and has an accuracy similar to the recursive methods with 3 coefficients (Section V, Figure 3(b)). This means that our proposed method requires less operations than Deriche [7] and all the integral image based methods, and a comparable number of operations to Young and van Vliet [8].

The proposed method is also convenient for parallel computation. The summation step (Equation 5) can be parallelized as described in Section 4.3 of [25], while the next step (Equation 6) computes the response of each pixel independently of its neighbors. On the other hand, the recursive methods can be parallelized only partially – e.g. by filtering different rows independently. However, within each row (or column) all the computations are strongly dependent.

Notice also that Equation 6 can be computed for a small percentage of the image pixels. This can accelerate applications in which the kernel response is computed for sampled windows such as image downscaling. The recursive methods do not have this advantage, since they compute the kernel response of each pixel using the responses of its neighbors.

## IV. KERNEL APPROXIMATION

The approximation of a one dimensional kernel $K(t)$ by $k$ constant functions is determined by the partition $P$ and the constants $c_i$. Finding the best parameters is done by minimizing an approximation error function on the desired kernel. Indeed, our real purpose is to minimize the error on the *output image*, which is not necessarily equivalent. Section IV-A relates the kernel approximation error and the output image error using natural image statistics. We define a quadratic form error function for the kernel approximation, so that the output image $l_2$ error is minimized. This error function is used in Section IV-B to approximate the Gaussian kernel.

### A. Minimal Output $l_2$ Error

We denote the input signal by $x$, the output signal by $y$, and the kernel weights by $w$. The approximated output and kernel weights are denoted by $\hat{y}$ and $\hat{w}$ respectively. The upper case letter $X$ denotes the Fourier transforms of the input $x$.

The squared $l_2$ error of the final result is given by

$$E_2 = \sum_i (y_i - \hat{y}_i)^2 \tag{7}$$

Since $y_i = \sum_j w_j x_{i+j}$, $E_2$ can be expressed in terms of the input signal $x$ and the kernels $w, \hat{w}$:

$$\begin{aligned} E_2 &= \sum_i \Big( \sum_j w_j x_{i+j} - \sum_j \hat{w}_j x_{i+j} \Big)^2 \\ &= \sum_{i,j,k} (w_j - \hat{w}_j)(w_k - \hat{w}_k) x_{i+j} x_{i+k} \\ &= \sum_{j,k} \Big( (w_j - \hat{w}_j)(w_k - \hat{w}_k) \sum_i x_{i+j} x_{i+k} \Big) \end{aligned} \tag{8}$$

The only term which involves the input signal $x$ is the inner sum, which we denote as $A_{jk} = \sum_i x_{i+j} x_{i+k}$.

Note that $A_{jk}$ is the the autocorrelation of $x$ at location $j-k$. The error can be therefore expressed as

$$E_2 = (w - \hat{w})^{\mathrm{T}} A (w - \hat{w}) \tag{9}$$

where $A$'s entries are given by the autocorrelation of the signal $x$.

In order to make $E_2$ independent of the values of a specific $x$, we make use of a fundamental property of natural images introduced by Field [26] the Fourier spectrum of a natural image in each frequency $u$ is proportional to $\frac{1}{u}$,

$$|X_u| \propto \frac{1}{u} \tag{10}$$

Applying the Convolution Theorem, the autocorrelation of $x$ is

$$|X_u|^2 \propto \frac{1}{u^2} \tag{11}$$

Hence, $A_{jk}$ should be proportional to $\Phi$, the inverse Fourier transform of $\frac{1}{u^2}$.

Of course, the definition of $\Phi$ is problematic since $\frac{1}{u^2}$ is not defined for $u = 0$. Completing the missing value by an arbitrary number affects all $\Phi$ values. However, additional information is available which helps to complete the missing value.

Assuming that the values of $x$ are uniformly distributed in $[0, 1]$, we find that $\Phi_0$ (which equals to $A_{jj}$, the correlation of a pixel with itself) is proportional to $\int_0^1 x^2 dx = \frac{1}{3}$. In addition, assuming that distant pixels are uncorrelated, the boundary values $\Phi_r, \Phi_{-r}$ are proportional to $\mu_x^2 = \frac{1}{4}$. This means that the ratio $\frac{\Phi_0}{\Phi_r}$ should be $\frac{4}{3}$. Determining the missing zero value in the Fourier domain as 16.5 results in such a function $\Phi$.

As shown by our experiments (Section V, Figure 3(b)), approximating the Gaussian kernel using the proposed quadratic form (Equation 9) where $A_{jk} = \Phi_{j-k}$ results in a very low $l_2$ error (or high PSNR) on the output image. Modifying $\Phi$ values slightly decreases the accuracy.

### B. Gaussian Approximation

The Gaussian kernel is zero mean and its only parameter is standard deviation $\sigma$. Actually, all the Gaussian kernels are normalized and scaled versions of the standard kernel $\exp(\frac{t^2}{2})$. Therefore the approximation need to be computed only once for some $\sigma_0$. For other $\sigma$ values the pre-computed solution of $N(t)$ is simply rescaled.

We approximate the Gaussian within the range $[-\pi\sigma, \pi\sigma]$, since for greater distances from the origin kernel values are negligible. The optimization is done on the positive part $[0, \pi\sigma]$. Then the kernel symmetry is used to getcompute the weights $w_i$ as described in Section III-B.

The partition indices $p_i$ and constants $c_i$ were found by an exhaustive search using 100 samples of the Gaussian kernel with $\sigma_0 = \frac{100}{\pi}$. To get the parameters for other $\sigma$ values, we scale the indices and round them:

$$p_i^{(\sigma)} = \left\lfloor \frac{\sigma}{\sigma_0} \cdot p_i \right\rfloor \tag{12}$$

The weights are also scaled,

$$w_i^{(\sigma)} = \frac{p_i}{2p_i^{(\sigma)} + 1} \cdot w_i \tag{13}$$

The parameters for different numbers of constants are presented in Table II. Figure 1 shows the optimal approximation with 4 constant functions. Figure 2(c,d) present the 2D result of filtering image rows and columns using the proposed approximation.

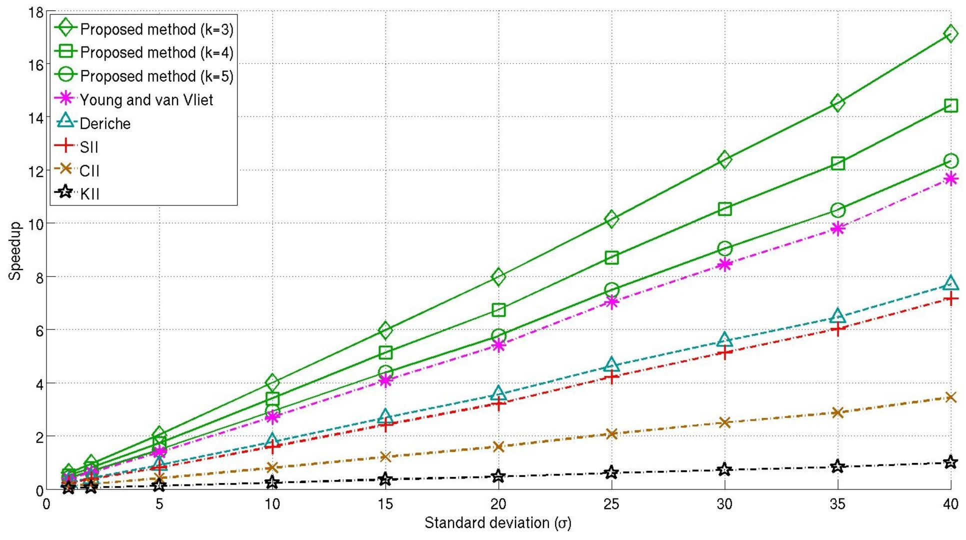
| #constants ($k$) | partition indices ($p_i$) | weights ($c_i$) |
|---|---|---|
| $k = 3$ | 23, 46, 76 | 0.9495, 0.5502, 0.1618 |
| $k = 4$ | 19, 37, 56, 82 | 0.9649, 0.6700, 0.3376, 0.0976 |
| $k = 5$ | 16, 30, 44, 61, 85 | 0.9738, 0.7596, 0.5031, 0.2534, 0.0739 |

TABLE II
ALGORITHM PARAMETERS.

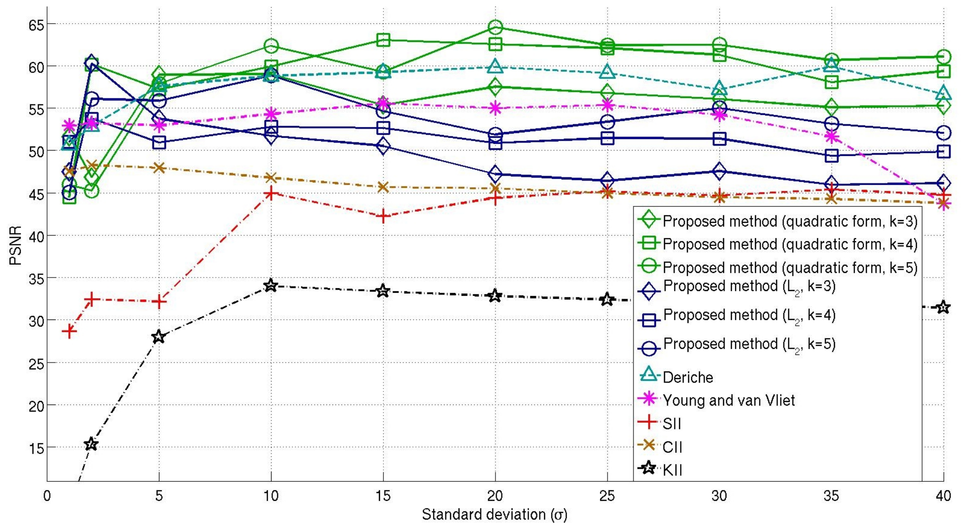
## V. Experimental Results

In order to evaluate the performance Gaussian filtering methods we made an experiment on natural images from different scenes. We used a collection of 20 high resolution images (at least 1 megapixel) taken from Flickr under creative commons. Each of the image was filtered with several standard deviation ($\sigma$) values. Speed and accuracy are measured in comparison to the exact Gaussian filtering performed by the 'cvSmooth' function of the OpenCV library [27]. The output image error is measured by the peak signal-to-noise ratio (PSNR). This score is defined as $-10log_{10}(MSE)$, $MSE$ is the mean squared error and the image values are in $[0, 1]$.

We examined our proposed method as well as the state of the art methods of Deriche [7] and Young and van Vliet [8].



(a) Speedup



(b) Accuracy

Fig. 3. Experimental results (Section V). (a) Time speedup is in comparison to exact filtering. Our proposed method with $k = 3$ is the fastest. Using $k = 5$ is similar to Young and van Vliet, while Deriche and SII are slower. (b) Accuracy of the output image: the highest PSNR (minimal $l_2$) is achieved by our proposed quadratic form approximation with $k = 4$ or $k = 5$ which are slightly better than Deriche. Using $k = 3$ is still better than all other methods including Young and van Vliet. Approximating the Gaussian kernel by $l_2$ instead of the proposed quadratic form decreases the approximation accuracy. Kernel Integral Images (KII) gives poor results for small $\sigma$ values since integral images of polynomials are numerically unstable for high ratio between $\sigma$ and the image size.

We also examined the integral image based methods of Hussein et al. (KII, [20]), Elboher and Werman (CII, [22]) and Bhatia et al. (SII, [24]), which use a similar approach to the proposed method (Section II). All the compared methods were implemented by us[1] except Young and van Vliet's filtering, for which we adopted the code of [11]. This implementation is based on the updated design of Young and van Vliet in [12] with corrected boundary conditions [28].

In order to examine the effect of using different error functions for kernel approximation, we tested our proposed method with two sets of parameters. The first set was computed by minimizing the quadratic form error function defined in Section IV-A. The second set was computed by minimizing the $l_2$ error. Since the filtering technique is identical the speed is the same, the difference is in the output accuracy.

Figure 3 presents the average results of our experiments. The highest acceleration is achieved by our proposed method with $k = 3$ constants (Figure 3(a)). Using $k = 4$ is still faster than all other methods, while $k = 5$ has equivalent speedup as Young and van Vliet's method.

The best accuracy is achieved by our quadratic form approximation with $k = 4, 5$, which is slightly better than Deriche's method (Figure 3(b)). Using $l_2$ kernel approximation decreases

[1] The CImg library contains an implemetation of Deriche's method for $k = 2$ coefficients. Due to caching considerations, our implementation is about twice as fast. For the integral image based methods there is no public implementation except CII [22] which was proposed by the authors of the current paper.

the quality of our proposed method. This demonstrates the importance of using an appropriate kernel approximation. However, these results are still similar to Young and van Vliet and better than other integral image based methods.

## VI. Conclusion

We present a very efficient and simple scheme for filtering with separable non uniform kernels. In addition we analyze the relation between kernel approximation, output error and natural image statistics. Computing an appropriate Gaussian approximation by the proposed filtering scheme is faster than the current state of art methods, while preserving similar accuracy.

## References

[1] R.C. Gonzalez and R.E. Woods, *Digital image processing*, Addison Wisley, 1992.

[2] P.J. Burt, "Fast filter transform for image processing," *Computer graphics and image processing*, vol. 16, no. 1, pp. 20–51, 1981.

[3] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *Communications, IEEE Transactions on*, vol. 31, no. 4, pp. 532–540, 1983.

[4] J.G. Proakis and D.G. Manolakis, *Digital signal processing: principles, algorithms, and applications*, Engelwoods Cliffs, NJ: Prentice Hall, 1996.

[5] R. Deriche, "Using canny's criteria to derive a recursively implemented optimal edge detector," *International journal of computer vision*, vol. 1, no. 2, pp. 167–187, 1987.

[6] R. Deriche, "Fast algorithms for low-level vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 78–87, 1990.

[7] R. Deriche, "Recursively implementating the gaussian and its derivatives," *Research Report 1893,INRIA, France*, 1993.

[8] I.T. Young and L.J. van Vliet, "Recursive implementation of the gaussian filter," *Signal processing*, vol. 44, no. 2, pp. 139–151, 1995.

[9] L.J. Van Vliet, I.T. Young, and P.W. Verbeek, "Recursive gaussian derivative filters," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*. IEEE, 1998, vol. 1, pp. 509–514.

[10] J.S. Jin and Y. Gao, "Recursive implementation of log filtering," *real-time imaging*, vol. 3, no. 1, pp. 59–65, 1997.

[11] J.M. Geusebroek, A.W.M. Smeulders, and J. van de Weijer, "Fast anisotropic gauss filtering," *Image Processing, IEEE Transactions on*, vol. 12, no. 8, pp. 938–943, 2003.

[12] I.T. Young, L.J. van Vliet, and M. van Ginkel, "Recursive gabor filtering," *Signal Processing, IEEE Transactions on*, vol. 50, no. 11, pp. 2798–2805, 2002.

[13] S. Tan, J.L. Dale, and A. Johnston, "Performance of three recursive algorithms for fast space-variant gaussian filtering," *Real-Time Imaging*, vol. 9, no. 3, pp. 215–228, 2003.

[14] MJ McDonnell, "Box-filtering techniques," *Computer Graphics and Image Processing*, vol. 17, no. 1, pp. 65–70, 1981.

[15] F.C. Crow, "Summed-area tables for texture mapping," in *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. ACM, 1984, pp. 207–212.

[16] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Citeseer, 2001, vol. 1.

[17] P.S. Heckbert, "Filtering by repeated integration," *ACM SIGGRAPH Computer Graphics*, vol. 20, no. 4, pp. 315–321, 1986.

[18] K.G. Derpanis, E.T.H. Leung, and M. Sizintsev, "Fast scale-space feature representations by generalized integral images," in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*. IEEE, 2007, vol. 4, pp. IV–521.

[19] M. Werman, "Fast convolution," *J. WSCG*, vol. 11, no. 1, pp. 528–529, 2003.

[20] M. Hussein, F. Porikli, and L. Davis, "Kernel integral images: A framework for fast non-uniform filtering," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[21] D. Marimon, "Fast non-uniform filtering with Symmetric Weighted Integral Images," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE, 2010, pp. 3305–3308.

[22] E. Elboher and M. Werman, "Cosine integral images for fast spatial and range filtering," in *IEEE International Conference on Image Processing, 2011*. IEEE, 2011, to apper.

[23] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on*. IEEE, 2002, pp. 839–846.

[24] A. Bhatia, W.E. Snyder, and G. Bilbro, "Stacked Integral Image," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1530–1535.

[25] G. Hendeby, J.D. Hol, R. Karlsson, and F. Gustafsson, "A graphics processing unit implementation of the particle filter," *Measurement*, vol. 1, no. 1, pp. x0, 2007.

[26] D.J. Field et al., "Relations between the statistics of natural images and the response properties of cortical cells," *J. Opt. Soc. Am. A*, vol. 4, no. 12, pp. 2379–2394, 1987.

[27] OpenCV v2.1 documentation, "cvsmooth," http://opencv.willowgarage.com/documentation/c/image_filtering.html.

[28] B. Triggs and M. Sdika, "Boundary conditions for young-van vliet recursive filtering," *Signal Processing, IEEE Transactions on*, vol. 54, no. 6, pp. 2365–2367, 2006.