COSINE INTEGRAL IMAGES FOR FAST SPATIAL AND RANGE FILTERING

Elhanan Elboher and Michael Werman

School of Computer Science
The Hebrew University of Jerusalem
Jerusalem 91904, Israel
{elhanane,werman}@cs.huji.ac.il

ABSTRACT

Non uniform filtering is important for many image processing algorithms. However, for large kernel sizes the filtering can become computationally expensive. We introduce *cosine integral images* (CII) which represent a large set of spatial and range filters, based on their frequency decomposition. The filtering requires a constant number of operations per image pixel, independent of filter size. We make use of CII to compute the Gabor filters, whose complexity is a constant $O(1)$ operations per image pixel. We also improve previous constant time approximations of spatial Gaussian smoothing and bilateral filtering.

Index Terms— Non uniform filtering, integral images, cosine transform, Gabor filter, Gaussian filter, bilateral filter.

1. INTRODUCTION

1.1. Fast Non Uniform Image Filtering

Image filtering by various kernels is an important image processing tool, which requires fast and efficient computation. The main difficulty is that most of the kernels depend on the *kernel size*, a neighborhood of the current pixel. When the kernel size increases, the computation becomes slower.

In order to overcome this difficulty, Crow [1] introduced *summed area tables*, known also as *integral images* [2]. Based on the summation (or integration) of pixel values along the image rows and columns, the sum of a rectangular region can be computed using $O(1)$ operations, independent of its size.

Crow's method makes it possible to convolve an image very fast with uniform kernels. However, many tasks require filtering by non-uniform kernels which depend on the spatial distances in the pixel's neighborhood (*spatial kernels*, e.g. Gaussian smoothing) or on the pixel values (*range kernels*, e.g. the bilateral filter [3]).

Heckbert [4] generalized integral images for spatial dth-degree polynomial kernels, by $d+1$ repeated integrations. Werman [5] extended the recursive approach for all the functions which satisfy a linear homogeneous equation.

Hussein et al. [6] proposed Kernel Integral Images (KII) for fast spatial filtering. The main idea is to decompose the filter kernel, exactly or approximately, into simple functions which are computable by integral images. Section 1.2 describes KII and related methods.

It should be remarked that Deriche's classical method [7] for Gaussian smoothing performs better than the mentioned integral image based methods.

Another issue is speeding up range kernels such as the bilateral filter (BF) [3]. For each pixel, x_0 the BF computes a weighted average its neighborhood. The weight of each neighbor x depends on pairwise value difference e.g. the pixel intensity, $exp(-\frac{\left(f(x_0)-f(x)\right)^2}{2\sigma^2})$. Previous works proposed fast BF approximation by different methods based on integral images, these methods are reviewed in [8].

1.2. Kernel Decomposition

Consider the convolution $f * K$ of an image f with an arbitrary kernel K. As shown by [6], using simple kernels K_i, $i = 0, 1, ...k$ which can be computed by integral images, the following convolution: $f * K = \sum_{i=0}^{k} w_i (f * K_i)$ can be performed using $O(k)$ operations per pixel, independent of kernel size.

Selecting appropriate functions K_i is critical for the performance of the filtering scheme. The selected functions should be able to express a large set of complicated kernels, exactly or approximately. On the other hand, the representation of each kernel should be sparse. A specific kernel should be represented by only few simple functions, otherwise it will become computationally expensive.

The authors of KII suggested using polynomial functions for the representation of complicated kernels. This selection is appropriate where the kernel can be represented by low degree polynomials. However, high degree polynomials (e.g. d=6,7,8) require not only $d+1$ integrations, but also high precision numerical representation, due to raising numbers to high powers.

Another option is to use simpler kernels K_i. Marimon [9] used linear functions for the construction of triangle-shaped

and pyramid-shaped kernels. The Stacked Integral Images (SII) proposed by Bhatia et al. [10] use as even simpler representation. A non uniform kernel (e.g. Gaussian) is approximated by a 'stack' of box filters (constants) which are all computed by accessing a single integral image. The approximation requires a non convex optimization for each kernel size and number of boxes.

1.3. Contributions

This paper introduces *cosine integral images* (CII). We suggest a different type of simple kernels: cosine functions of various frequencies u, $\cos(ux)$, which is an orthogonal basis. The desired kernel is approximated by a linear combination of cosines, which are the first k terms of its inverse Discrete Cosine Transform (DCT). The convolution with each cosine function is computed using a constant number of operations per pixel.

The frequency decomposition results in very accurate approximation of non uniform spatial and range kernels using only few terms. We demonstrate the utility of CII for such kernels by improving the approximations of Gaussian smoothing and bilateral filtering, compared with previous methods.

An additional advantage of CII is the convenient representation of high frequency terms. Opposed to high-degree polynomials, there is no numerical problem of overflow or rounding, since the sine and cosine functions are bounded in $[-1, 1]$. This makes it possible to compute high frequency kernels such as Gabor using only a few relevant cosine terms, as described in Section 3.2.

2. COSINE INTEGRAL IMAGES

2.1. DCT Based Decomposition

We first discuss the one dimensional *cosine integral images* (CII), and then generalize them to higher dimensions.

Let $K(t)$ be a 1D symmetric kernels kernel of length $2r+1$. The l_2 approximation error of $K(t)$ in $[-r, r]$ by linear combination of k simple functions K_u is

$$\int_{-r}^{r} \left(K(t) - \sum_{u=0}^{k-1} K_u(t) \right)^2 dt \tag{1}$$

Selecting K_u to be cosine functions $\cos(ut)$, the minimal l_2 error is reached by inverse Discrete Cosine Transform (DCT):

$$K(t) = \sum_{u=0}^{k-1} a_u \cos(\frac{\pi}{r-1} ut) \tag{2}$$

where a_u are the DCT coefficients of the kernel $K(t)$. (The proposed framework can be easily extended for non symmetric using more DCT coefficients).

2.2. Efficient Spatial Filtering Scheme

We describe now an efficient scheme to convolve a discrete 1D function $f(x)$ (e.g. an image row) with a cosine kernel $c_u(t) = \cos(ut)$ within a bounded segment, $t \in [-r, r]$:

$$(f * c_u)(x_0) = \sum_{x=x_0-r}^{x_0+r} f(x) \cos(u(x_0 - x)) \tag{3}$$

Based on the trigonometric identity $\cos(\alpha - \beta) = \cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta)$, we replace Equation (3) by

$$\cos(ux_0) \sum_x f(x) \cos(ux) + \sin(ux_0) \sum_x f(x) \sin(ux) \tag{4}$$

The values of $\cos(ux), \sin(ux)$ can be stored in lookup tables. Thus, the first sum in Equation (4) can be computed efficiently as follows. First we compute the cumulative sum $I_u(x) = \sum_{x'=0}^{x} f(x') \cos(ux')$.
The desired sum is $I_u(x + r) - I_u(x - r - 1)$.

To analyze the time complexity we use the cost function $C(+, \times, MA)$ from [9], where '$+$' counts additions, '$\times$' counts multiplications and 'MA' counts memory access operations ('lookups') – each of them for a single coordinate (or pixel) x. The cost of the above computation is $C(2, 1, 1)$.

The second sum in Equation (4) is computed similarly with the same cost, where $\cos(ux)$ is replaced by $\sin(ux)$. The total computational cost of Equation (4) is therefore $C(5, 4, 4)$. For $u = 0$, Equation (4) reduces to $\sum_x s(x)$ whose cost is $C(2, 0, 0)$.

To convolve a function f with linear combination of k cosine kernels $\{\cos(ut)\}_{u=0}^{k-1}$ we need $k - 1$ more additions. The multiplications of the cosine kernels with their appropriate DCT coefficient are pre-computed using the lookup tables of $\cos(ut), \sin(ut)$. The total cost is therefore $C(6k-4, 4k-4, 4k-4)$.

2.3. Higher Dimensions & Space Complexity

We now describe the 2D case in which an image $f(x, y)$ is convolved with a 2D kernel $K(t_x, t_y) = \cos(u_x t_x)\cos(u_y t_y)$. K can be expressed as a convolution of two 1D filters $K_x * K_y^T$. Thus the convolution can be computed by filtering the image rows with K_x, and then filtering the columns of the intermediate result by K_y. Hence, convolving an image with a *separable* 2D kernel only doubles computational cost of the 1D case.

The space complexity is also very low. Since the filtering of each row and column is independent, filtering an image of size $n \times m$ with k cosine kernels requires only $O(k \max(n, m))$ additional space over the input and the output images.

The computation can be extended for n-dimensional separable filters with cost of $C(n(6k-4), n(4k-4), n(4k-4))$. Extending CII for non separable kernels is possible, based on

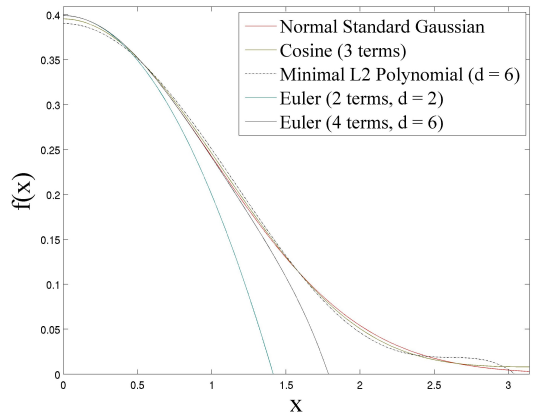

Fig. 1. Approximations of the normal standard Gaussian kernel. We compare the l_2 error in $[-\pi, \pi]$ (Equation (1)) for different approximations. The error of 3 cosine functions $\{\cos(ux)\}_{u=0}^{2}$ with the appropriate DCT coefficients is 0.0000469. This is better than the 6th degree polynomial with minimal l_2 error on $[-\pi, \pi]$, which is 0.000266. The error of the first 2 Euler terms, used in KII [6], is 0.0203. The error of 4 Euler terms (a 6th degree polynomial) is 0.0047.

their 2-dimensional cosine decomposition. However, this requires the computation of more basic cosine functions.

2.4. Range Filtering

In the above scheme the kernel argument was the spatial distance between the current pixel location, x_0 to another location x. The scheme for range filtering is almost identical. The only change is that now the arguments are the values of f in those locations $f(x_0)$, $f(x)$. When the range of f is bounded (e.g. $[0, 255]$) we store $\cos(uf(x))$, $\sin(uf(x))$ in lookup tables, indexed by gray level. Hence, the above scheme can be adapted for range filtering with the same time complexity.

3. APPLICATIONS AND RESULTS

We applied the CII filtering scheme for the spatial Gaussian and Gabor filters and for the range bilateral filter. The speedup and accuracy of CII and previous methods are compared with the direct computation of the filtering.

Note that we optimized *all* the methods. The direct computation is performed separately on rows and columns and makes use of lookup tables. This makes the direct computation faster by orders of magnitude than a naive implementation. The code of the experiments will be available on the web.

3.1. Gaussian Smoothing

The Gaussian filter is zero mean, its only parameter is standard deviation σ. Choosing a $r \times r$ window is actually an approximation of the Gaussian kernel, which is positive on $[-\infty, \infty]$. We set $r = \pi\sigma$ since for greater distances from the origin, kernel values are negligible. Since r depends on σ, all the Gaussian kernels are rescalings of the normal standard kernel $N(t) = \frac{1}{\sqrt{2\pi}} \exp(\frac{t^2}{2})$. The rescaling is $\frac{1}{\sigma} N(\frac{t}{\sigma})$.

Figure 1 shows the approximation of $N(t)$ on $t = [-\pi, \pi]$ using polynomials and cosines. The Euler expansion approximates $N(t)$ only near 0. The least square 6th-degree approximation on $[-\pi, \pi]$ has much smaller error. The most accurate approximation is achieved by 3 cosine terms.

Table 1 presents the results of convolving a large image (7.5 Mpixel) with a large Gaussian kernel using KII, SII, Deriche and CII.

Parameters (σ_x, σ_y)	Deriche	KII	SII	CII $(k = 3)$	CII $(k = 4)$
(a) $8.75, 18.3$	47.21, $\times$4.05	36.43, $\times$1.21	47.38, $\times$6.40	48.25, $\times$3.47	57.07, $\times$2.65
(b) $40, 40$	42.19, $\times$20.37	31.20, $\times$5.96	N/A	44.32, $\times$17.38	56.06, $\times$13.52
(c) $60, 60$	40.31, $\times$43.95	29.39, $\times$12.19	N/A	41.91, $\times$37.66	55.68, $\times$28.25

Table 1. Gaussian smoothing – experimental results. A 3360×2240 image (7.5 megapixel) was filtered with different (σ_x, σ_y) values. We present the PSNR score (the first number in each cell) and speedup factor in comparison to *optimized* direct convolution (see Section 3). For KII we used 2 Euler terms (=9 integral images) as in [6]. For SII we used 5 boxes. Since SII requires non convex optimization for each kernel size, we report in (a) the results for 55×115 window used in [10]. CII was examined using 3 and 4 cosine terms (=5 and 7 integral images, respectively).

3.2. Gabor Kernels

The 1D Gabor kernel around the pixel x_0 is defined by

$$\exp\left(-\frac{(x_0 - x)^2}{2\sigma^2}\right) \cos\left(\frac{2\pi}{\lambda}(x_0 - x) + \psi\right) \qquad (5)$$

In two dimensions, the coordinate system is rotated in angle θ. Computing the Gabor kernel is then equivalent to 1D Gabor filtering of the new x axis, and a Gaussian smoothing of the new y axis. Therefore we separate the 2D kernel by rotating the input image by θ, and rotate the result back by $-\theta$ afterwards. As in Section 2, we split the cosine term and rewrite Equation (5)

$$\begin{aligned} &\exp\left(-\tfrac{(x_0-x)^2}{2\sigma^2}\right) \cos\left(\tfrac{2\pi}{\lambda}x_0 + \psi\right) \cos\left(\tfrac{2\pi}{\lambda}x\right) + \\ &+ \exp\left(-\tfrac{(x_0-x)^2}{2\sigma^2}\right) \sin\left(\tfrac{2\pi}{\lambda}x_0 + \psi\right) \sin\left(\tfrac{2\pi}{\lambda}x\right) \end{aligned} \qquad (6)$$

Since the exponential terms are Gaussians, they can be decomposed as in Section 3.1. We store lookup tables for the

Gabor parameters $(\sigma_x, \sigma_y, \lambda, \psi, \theta)$	CII parameters (k, #in.ims.)	PSNR (dB)	Time speedup
(45, 45, 90, -10, 60)	$k = 4$, #14	52.02	$\times$ 7.08
(25, 25, 40, 0, -45)	$k = 3$, #10	52.03	$\times$ 4.18
(35, 15, 50, 25, 0)	$k = 3$, #10	48.67	$\times$ 3.07
(45, 25, 70, -12, 130)	$k = 3$, #10	48.01	$\times$ 5.62
(75, 35, 140, 0, -65)	$k = 4$, #14	53.47	$\times$ 6.12
(33, 64, 10, 10, 17)	$k = 3$, #10	43.77	$\times$ 16.13

Table 2. Gabor filtering - experimental results. A 2880×3840 image (7.5 Mpixel) was filtered by various Gabor kernels using CII. The results are compared to direct convolution (optimized).

Input parameters	CII		Yang et al. [8]	
	Time (sec.)	PSNR (dB)	Time (sec.)	PSNR (dB)
512×512, r=15, σ=0.08, k=12	0.18	58.64	0.32	53.61
640×480, r=14, σ=0.2, k=5	0.1	56.26	0.11	52.36
2119×3148, r=64, σ=0.08, k=12	5.85	51.70	7.26	51.61
2119×3148, r=64, σ=0.2, k=5	2.30	55.10	3.08	45.09

Table 3. Bilateral filtering experiment (Section 3.3).

following expressions: $\cos(ut)\cos\left(\frac{2\pi}{\lambda}t\right)$, $\cos(ut)\sin\left(\frac{2\pi}{\lambda}t\right)$, $\sin(ut)\cos\left(\frac{2\pi}{\lambda}t\right)$ and $\sin(ut)\sin\left(\frac{2\pi}{\lambda}t\right)$. Then, Equation 6 can be efficiently computed using the CII filtering scheme from Section 2.2.

Table 2 compares the filtering of an image by various Gabor kernels using CII against the direct computation. While preserving high accuracy, CII speeds up the computation much more than previously reported factors, e.g. ~ 1.61 using the Reshuffling method [11] for PSNR = 50dB.

3.3. Bilateral Filtering

The CII range filtering (Section 2.4) was applied to approximate bilateral filtering (BF) [3].

We examined our method in comparison to Yang et al. [8], which is the current state of the art fast BF approximation, using their published code (the updated version without spatial quantization). To filter $n \times m$ images we set spatial box kernel with radius $r = 0.03 \min(n, m)$ and range Gaussian kernels with different σ values. The parameter k denotes the number of sampled values (for Yang et al.) or the number of cosine kernels (for CII). Table 3 presents the experimental results [1].

4. SUMMARY

We presented *cosine integral images* (CII), an efficient spatial and range filtering scheme. The filtering requires $O(1)$ operations per image pixel independently of the kernel size. We demonstrated that CII makes it possible to perform an efficient and accurate image filtering by the Gaussian and Gabor spatial kernels and by the bilateral filter.

[1] In our current implementation, filtering columns is slower than rows.

5. REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] P.S. Heckbert, "Filtering by repeated integration," in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. ACM, 1986, pp. 315–321.

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

[6] 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.

[7] R. Deriche, "Recursively implementating the Gaussian and its derivatives," 1993.

[8] Q. Yang, K.H. Tan, and N. Ahuja, "Real-time o (1) bilateral filtering," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 557–564.

[9] 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.

[10] 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.

[11] F. Porikli, "Reshuffling: a fast algorithm for filtering with arbitrary kernels," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 2008, vol. 6811, p. 18.