Fast L^1 Gauss Transforms for Edge-Aware Image Filtering

[1,2]Dina Bashkirova <dina.bashkirova@riken.jp>
[1]Shin Yoshizawa <shin@riken.jp>
[2]Roustam Latypov <roustam.latypov@kpfu.ru>
[1]Hideo Yokota <hyokota@riken.jp>
[1]Image Processing Research Team,
RIKEN Center for Advanced Photonics, RIKEN,
2-1, Hirosawa, Wako, Saitama, 351-0198, Japan
[2]Institute of Computational Mathematics and Information Technologies,
Kazan Federal University,
35 Kremlyovskaya, Kazan, Russia, 420008

Abstract. Gaussian convolution and its discrete analogue, Gauss transform, have many science and engineering applications, such as mathematical statistics, thermodynamics and machine learning, and are widely applied to computer vision and image processing tasks. Due to its computational expense (quadratic and exponential complexities with respect to the number of points and dimensionality, respectively) and rapid spreading of high quality data (bit depth/dynamic range), accurate approximation has become important in practice compared with conventional fast methods, such as recursive or box kernel methods. In this paper, we propose a novel approximation method for fast Gaussian convolution of two-dimensional uniform point sets, such as 2D images. Our method employs L1 distance metric for Gaussian function and domain splitting approach to achieve fast computation (linear computational complexity) while preserving high accuracy. Our numerical experiments show the advantages over conventional methods in terms of speed and precision. We also introduce a novel and effective joint image filtering approach based on the proposed method, and demonstrate its capability on edge-aware smoothing and detail enhancement. The experiments show that filters based on the proposed L1 Gauss transform give higher quality of the result and are faster than the original filters that use box kernel for Gaussian convolution approximation.

Keywords: Gaussian smoothing, Laplace distribution, fast approximation algorithms.

DOI: 10.15514/ISPRAS-2017-29(4)-4

For citation: Bashkirova D.R., Yoshizawa S., Latypov R.H., Yokota H. Fast L1 Gauss Transforms for Edge-Aware Filtering. *Trudy ISP RAN/Proc. ISP RAS*, vol. 29, issue 4, 2017, pp. 55-72. DOI: 10.15514/ISPRAS-2017-29(4)-4

1. Introduction

Gaussian convolution is a core tool in mathematics and many related research areas, such as probability theory, physics, and signal processing. Gauss transform is a discrete analogue to the Gaussian convolution, and has been widely used for many applications including kernel density estimation [1] and image filtering [2]. Despite its reliable performance and solid theoretical foundations, Gauss transform in its exact form along with other kernel-based methods has a drawback – it is very computationally expensive (has quadratic computational complexity w.r.t. the number of points) and hard to scale to higher dimensions. Which is why there have been many attempts to overcome these problems by creating approximation algorithms, such as fast Gauss transform [3], dualtree fast Gauss transforms [4], fast KDE [5], and Gaussian kd-trees [6]. Also, box kernel averaging [7] and recursive filtering [8] have been popular in computer graphics and image processing because of their simplicity, see the surveys [9], [10] for numerical comparisons of these approximation methods.

Since high bit depth (also dynamic range) images have become popular in both digital entertainment and scientific/engineering applications, it is very important to acquire high approximation precision and to reduce artefacts caused by drastic truncation employed in many conventional methods focused on computational speed. One of the highly accurate methods is called fast L^1 Gauss transform approximation [11] based on using L^1 distance instead of conventional L^2 Euclidean metric. This L^1 metric preserves most of the properties of the L^2 Gaussian, and is separable, hence it allows to perform computations along each dimension separately, which is very beneficial in terms of computational complexity. Also, L^1 Gaussian has only one peak in Fourier domain at the coordinate origin, and therefore its convolution does not have some undesirable artefacts that box kernels and truncation methods usually have. However, this algorithm works only on one-dimensional (1D) point sets, although it can be extended to uniformly distributed points in higher dimensions by performing it separately in each dimension. In order to be able to acquire Gauss transform for non-uniformly distributed two-dimensional points and to further generalize it to higher dimensional cases, we need to extend existing method [11] to the 2D uniform case.

In this paper we propose a novel approximation method for fast Gauss two-dimensional (2D) image transform. Our method is based on extending the fast L^1 Gauss transform approximation on uniformly distributed 2D points that allows to perform Gaussian convolution quickly while preserving high accuracy. We demonstrate that efficiency of the proposed method in terms of computational complexity, numerical timing, and approximation precision.

We also successfully applied our method in the novel filtering approach based on combining the approximated L^1 Gauss transformations into the so-called guided filter [12] (joint image filtering via ridge regression). Our approach reduces computational costs while providing higher quality results compared to the conventional one. We show the application to edge-aware smoothing and image detail enhancement.

2. Fast L^1 Gauss Transform

In this section, we briefly describe the 1D domain splitting algorithm [11] employed for fast L^1 Gauss transforms.

Consider the ordered point set $\mathbb{X} = \{x_i\}_{i=1}^{N}$, $x_i \in \mathbb{R}$, $x_i \geq x_{i-1}$, $\forall i = \overline{2, N}$. Each point x_i has a corresponding value $I_i \in \mathbb{R}$, e.g. pixel intensity in case of images. The L^1 Gauss transform for each point in set $\mathbb{X}$ is given by

$$J(x_j) = \sum_{i=1}^{N} G(x_j - x_i) I_i, \quad G(x) = \exp(-\frac{|x|}{\sigma}), \quad (1)$$

where $G(x)$, $x \in$ R, is a L^1 Gaussian function (also called Laplace distribution in statistics) with its standard deviation σ. It is convenient to decompose L^1 norm by splitting its domain by using the point x_1 such that

$$|x_j - x_i| = \begin{cases} |x_j - x_1| - |x_i - x_1| & \text{if } x_1 \leq x_i \leq x_j, \\ |x_i - x_1| - |x_j - x_1| & \text{if } x_1 \leq x_j \leq x_i. \end{cases} \quad (2)$$

Thus, Gauss transform (1) using the equation (2) becomes

$$J(x_j) = I_i + G(x_j - x_1) \sum_{i=1}^{j-1} \frac{I_i}{G(x_i - x_1)} + \\ + \frac{1}{G(x_j - x_1)} \sum_{i=j+1}^{N} G(x_i - x_1) I_i. \quad (3)$$

Such representation (3) allows to reduce the amount of computational operations, since values $G(x_j - x_1)$, $\frac{1}{G(x_j - x_1)}$ and the sums $\sum_{j+1}^{N} I_i G(x_i - x_1)$ and $\sum_{i=1}^{j-1} \frac{I_i}{G(x_i - x_1)}$ can be precomputed in linear time. However, using the equation (3) may imply some numerical issues, such as overflow, if the distance between x_1 and x_l, $l \in \{i, j\}$ is relatively large. To avoid such issues, this algorithm introduced certain representative points (poles) $\{\alpha_k \in \mathbb{R}\}$ instead of using the single point x_1, where the distance between α_k and x_l is smaller than the length that causes the numerical instability. Hence the equation (3) becomes more complex form, a highly accurate truncation can be applied where $G(\alpha_k - x_j)$ is numerically equal to zero, see [11] for further technical details.

Although this algorithm can be used in case of multidimensional images by applying it separately in each dimension, this separable implementation approach is not applicable to nonuniformly distributed high-dimensional point sets. Therefore, we present a novel and natural extension of the domain splitting concept on 2D cases (images) in the following sections.

3. Two-Dimensional Algorithm

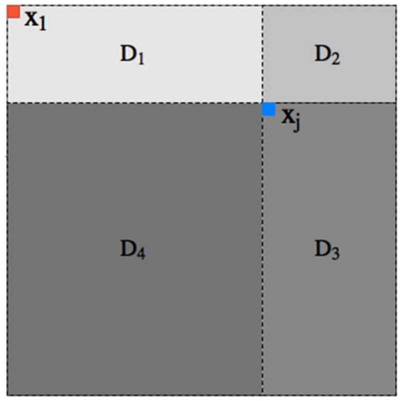
For a given 2D point set $\mathbb{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, $\mathbf{x}_i = (x_i, y_i) \in \mathbb{R}^2$, L^1 distance between two points in $\mathbb{R}^2$ is given by $|\mathbf{x}_j - \mathbf{x}_i| = |x_j - x_i| + |y_j - y_i|$, thus the Gauss transform (1) is represented by the formula:

$$J(\mathbf{x}_j) = \sum_{i=1}^{N} \exp(-\frac{|x_j - x_i| + |y_j - y_i|}{\sigma}) I_i$$

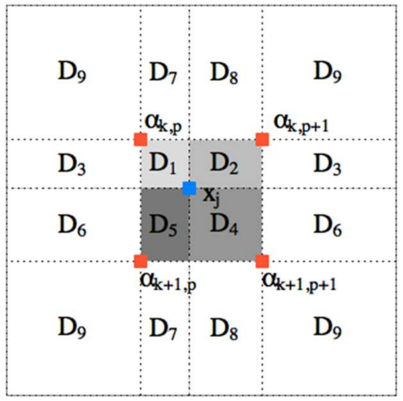
Domain splitting (2) for 2D points is given by

$$|x_j - x_i| + |y_j - y_i| = \begin{cases} |x_j - x_1| - |x_i - x_1| + |y_j - y_1| - |y_i - y_1| & \text{if } \mathbf{x}_i \in D_1 \\ |x_i - x_1| - |x_j - x_1| + |y_j - y_1| - |y_i - y_1| & \text{if } \mathbf{x}_i \in D_2 \\ |x_j - x_1| - |x_i - x_1| + |y_i - y_1| - |y_j - y_1| & \text{if } \mathbf{x}_i \in D_3 \\ |x_i - x_1| - |x_j - x_1| + |y_i - y_1| - |y_j - y_1| & \text{if } \mathbf{x}_i \in D_4, \end{cases}$$

see Fig. 1a for geometric illustration of the domains.



(a) Single pole $\mathbf{x}_l$ case



(b) Multipole $\{\alpha_k\}$ case

Fig. 1. Illustration of 2D domain splliting.

Using the above decomposition, Gauss transform is represented similar to (3):

$$J(\mathbf{x}_j) = I(\mathbf{x}_j) + F(x_j)F(y_j) \sum_{\mathbf{x}_i \in D_1(j)} \frac{1}{F(x_i)F(y_i)} I(\mathbf{x}_i) + \\ \frac{F(x_j)}{F(y_j)} \sum_{\mathbf{x}_i \in D_2(j)} \frac{F(y_i)}{F(x_i)} I(\mathbf{x}_i) + \frac{F(y_j)}{F(x_j)} \sum_{\mathbf{x}_i \in D_3(j)} \frac{F(x_i)}{F(y_i)} I(\mathbf{x}_i) + \\ \frac{1}{F(x_j)F(y_j)} \sum_{\mathbf{x}_i \in D_4(j)} F(x_i)F(y_i)I(\mathbf{x}_i), \quad (4)$$

where $F(x_j) \equiv G(x_j - x_1)$ and $F(y_j) \equiv G(y_j - y_1)$.

Precomputation and storage of values $\frac{F(y_j)}{F(x_j)}$ and $\frac{1}{F(x_j)F(y_j)}$ require $O(4N)$ operations and $O(4N)$ space, and all the subsequent sums $F(x_j)F(y_j)$, $\frac{F(x_j)}{F(y_j)}$ can be iteratively computed in $O(N)$ operations. Gauss transform for all points using the formula (4) requires $O(10N)$ as opposed to employing the separable implementation of equation

(3) for $O(6N)$ operations. Since computing the Gauss transform using the equation (4) is numerically troublesome, it is reasonable to divide the space into smaller groups and perform computations separately, as it was proposed in [11]. Let us introduce a novel 2D multipole approach for solving this problem.

Consider a set of poles $\{\alpha_k\}_{k=1}^{M}$, $\alpha_k = (a_k, b_k) \in \mathbb{R}^2$. The distance between points using poles α_k is given by

$$|\mathbf{x}_i - \mathbf{x}_j| = \begin{cases} |x_i - a_k| - |x_j - a_k| + |y_i - b_k| - |y_j - b_k| & \text{if } \mathbf{x}_i \in D_1 \\ |x_j - a_k| - |x_i - a_k| + |y_i - b_k| - |y_j - b_k| & \text{if } \mathbf{x}_i \in D_2 \\ |x_i - a_k| + |x_j - a_k| + |y_i - b_k| - |y_j - b_k| & \text{if } \mathbf{x}_i \in D_3 \\ |x_i - a_k| - |x_j - a_k| + |y_j - b_k| - |y_i - b_k| & \text{if } \mathbf{x}_i \in D_4 \\ |x_j - a_k| - |x_i - a_k| + |y_j - b_k| - |y_i - b_k| & \text{if } \mathbf{x}_i \in D_5 \\ |x_i - a_k| + |x_j - a_k| + |y_j - b_k| - |y_i - b_k| & \text{if } \mathbf{x}_i \in D_6 \\ |x_i - a_k| - |x_j - a_k| + |y_i - b_k| + |y_j - b_k| & \text{if } \mathbf{x}_i \in D_7 \\ |x_j - a_k| - |x_i - a_k| + |y_i - b_k| + |y_j - b_k| & \text{if } \mathbf{x}_i \in D_8 \\ |x_i - a_k| + |x_j - a_k| + |y_i - b_k| + |y_j - b_k| & \text{if } \mathbf{x}_i \in D_9, \end{cases}$$

where

$$D_1 = \{\mathbf{x}_i | x_i \in D_1^x, y_i \in D_1^y\}, D_2 = \{\mathbf{x}_i | x_i \in D_2^x, y_i \in D_1^y\},$$

$$D_3 = \{\mathbf{x}_i | x_i \in D_3^x, y_i \in D_1^y\}, D_4 = \{\mathbf{x}_i | x_i \in D_1^x, y_i \in D_2^y\},$$

$$D_5 = \{\mathbf{x}_i | x_i \in D_2^x, y_i \in D_2^y\}, D_6 = \{\mathbf{x}_i | x_i \in D_3^x, y_i \in D_2^y\},$$

$$D_7 = \{\mathbf{x}_i | x_i \in D_1^x, y_i \in D_3^y\}, D_8 = \{\mathbf{x}_i | x_i \in D_2^x, y_i \in D_3^y\},$$

$$D_9 = \{\mathbf{x}_i | x_i \in D_3^x, y_i \in D_3^y\},$$

$$\begin{aligned} D_1^x &= \{x_i | a_k \le x_i \le x_j \text{ or } x_j \le x_i \le a_k\}, \\ D_2^x &= \{x_i | a_k \le x_j \le x_i \text{ or } x_i \le x_j \le a_k\}, \\ D_3^x &= \{x_i | x_i \le a_k \le x_j \text{ or } x_j \le a_k \le x_i\}, \\ D_1^y &= \{y_i | b_k \le y_i \le y_j \text{ or } y_j \le y_i \le b_k\}, \\ D_2^y &= \{y_i | b_k \le y_j \le y_i \text{ or } y_i \le y_j \le b_k\}, \\ D_3^y &= \{y_i | y_i \le b_k \le y_j \text{ or } y_j \le b_k \le y_i\}, \end{aligned}$$

see Fig. 1b for geometric illustration of the domains with their poles. The point $\mathbf{x}_j$ is assigned for one representative pole defined by

$$\alpha_k(\mathbf{x}_j) = \max_k \{\alpha_k | a_k \le x_j, b_k \le y_j\},$$

which is the closest pole to $\mathbf{x}_j$ that has absolute values of coordinate smaller than $\mathbf{x}_j$.

For each point $\mathbf{x}_j$, the multipole L^1 Gauss transform is given by the equation (5),

$$J(\mathbf{x}_j) = I_j + \mathcal{G}(x_j)\mathcal{G}(y_j) \sum_{\mathbf{x}_i \in D_1} \frac{I_i}{\mathcal{G}(x_i)\mathcal{G}(y_i)} + \frac{1}{\mathcal{G}(x_j)\mathcal{G}(y_j)} \sum_{\mathbf{x}_i \in D_5} \mathcal{G}(x_i)\mathcal{G}(y_i) I_i + \frac{\mathcal{G}(y_j)}{\mathcal{G}(x_j)} \sum_{\mathbf{x}_i \in D_2} \frac{\mathcal{G}(x_i)}{\mathcal{G}(y_i)} I_i + \frac{\mathcal{G}(x_j)}{\mathcal{G}(y_j)} \sum_{\mathbf{x}_i \in D_4} \frac{\mathcal{G}(y_i)}{\mathcal{G}(x_i)} I_i +$$
$$+ \sum_{\alpha_k \in D_9} A_k^j + \sum_{\alpha_k \in D_7} B_k^j + \sum_{\alpha_k \in D_8} C_k^j + \sum_{\alpha_k \in D_3} D_k^j + \sum_{\alpha_k \in D_6} E_k^j, \tag{5}$$

$$A_k^j = \mathcal{G}(x_j)\mathcal{G}(y_j) \sum_{\mathbf{x}_i = \lambda(k)}^{\lambda(k+1)-1} \mathcal{G}(x_i)\mathcal{G}(y_i) I_i, \quad B_k^j = \mathcal{G}(x_j)\mathcal{G}(y_j) \sum_{\mathbf{x}_i = \lambda(k)}^{\lambda(k+1)-1} \frac{\mathcal{G}(y_i)}{\mathcal{G}(x_i)} I_i, \quad C_k^j = \frac{\mathcal{G}(y_j)}{\mathcal{G}(x_j)} \sum_{\mathbf{x}_i = \lambda(k)}^{\lambda(k+1)-1} \mathcal{G}(x_i)\mathcal{G}(y_i) I_i,$$

$$D_k^j = \mathcal{G}(x_j)\mathcal{G}(y_j) \sum_{\mathbf{x}_i = \lambda(k)}^{\lambda(k+1)-1} \frac{\mathcal{G}(x_i)}{\mathcal{G}(y_i)} I_i, \quad E_k^j = \frac{\mathcal{G}(x_j)}{\mathcal{G}(y_j)} \sum_{\mathbf{x}_i = \lambda(k)}^{\lambda(k+1)-1} \mathcal{G}(x_i)\mathcal{G}(y_i) I_i.$$

where $\mathcal{G}(x_j) \equiv G(x_j - a_k)$, $\mathcal{G}(y_j) \equiv G(y_j - b_k)$,

and $\lambda(\cdot)$ is an index function defined by

$$\lambda(k) = \min_{1 \le j \le N} (\mathbf{x}_j | a_k \le x_j < a_{k+1} \text{ and } b_k \le y_j < b_{k+1}).$$

For the sake of simplicity, we assume that the numbers of poles in 2D are same *M*. Following [11], *M* and the poles $\{\alpha_k\}$ are given by

$$\{a_k\} = \{b_k\} = \frac{\{0, 1, 2, ..., (M-1)\}w}{M}, \tag{6}$$
$$w = \max(|x_1 - x_N|, |y_1 - y_N|), \; M = [\frac{w}{\varphi\sigma \log(\text{MAX})}]$$

where $[\cdot]$ is the ceiling function, MAX is the maximum value of precision (e.g., double floating point: DBL_MAX in C programming language), and ϕ is a user-specified parameter (0.5 is employed in our numerical experiments). The above pole selection scheme leads to $\max(G(a_{k+1} - a_k), G(b_{k+1} - b_k)) <$ MAX which theoretically guarantees numerical stability in our method.

When the distance between poles is determined by the equation (6) and $G(\alpha_k - \mathbf{x}_j)$ becomes numerically zero if $|\alpha_k - \mathbf{x}_j| > \frac{w}{\varphi M}$, we can efficiently truncate Gauss transform by approximating the values:

$$\sum_{\alpha_k \in D_9} A_k^j \approx \sum_{\alpha_k \in \mu(D_9)} A_k^j, \; \sum_{\alpha_k \in D_7} B_k^j \approx \sum_{\alpha_k \in \mu(D_7)} B_k^j,$$
$$\sum_{\alpha_k \in D_8} C_k^j \approx \sum_{\alpha_k \in \mu(D_8)} C_k^j, \; \sum_{\alpha_k \in D_3} D_k^j \approx \sum_{\alpha_k \in \mu(D_3)} D_k^j,$$
$$\sum_{\alpha_k \in D_6} E_k^j \approx \sum_{\alpha_k \in \mu(D_6)} E_k^j,$$

where $\mu(D_*) = \{\mathbf{x}_i \in D_* \mid |\alpha_k(\mathbf{x}_j) - \alpha_k(\mathbf{x}_i)| \le \frac{w}{\varphi M}\}$.

In other words, instead of computing terms $A_k^j, B_k^j, C_k^j, D_k^j, E_k^j$ across all the corresponding point sets, we consider only the neighbouring points, which allows to avoid nested loop structures in our implementation and speed up the computational process.

As in the 1D algorithm [11], the terms can be iteratively computed in linear time. Assume that an image consists of $\sqrt{N} \times \sqrt{N}$ pixels and the number of poles along each dimension is *M*, total complexity of our method is $O(16N + 2\frac{\sqrt{N}}{M} + 4\frac{N}{M^2})$ which is a little bit slower than the separable implementation employed in [11] that requires $O(12N + 2\sqrt{N} + M)$ operations.

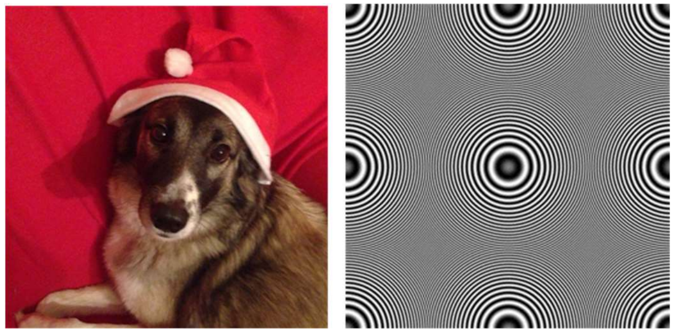

(a) Input image 1 *(b) Input image 2*

Fig. 2. Input images.

4. Numerical Experiments

We held all the experiments on Intel Core i7-6600U 2.60 GHz dual core computer with 16GB RAM and a 64-bit operating system. We compared the multipole version of our algorithm with box kernel (Box) using moving average method [7], the 1D domain splitting (YY14) with separable implementations [11], and Fast Discrete Cosine Transform (FDCT) via the FFT package [13] well-known for its efficiency.

To evaluate the performance of the methods mentioned above we used randomly generated 2D point sets with 10 different sizes from 128^2 to 5120^2 and 10 various values of $\sigma = 5,10,...,50$. The radius for the Box method was chosen equal to σ. The timing results (see Fig. 5) show that our method is slightly slower than the 1D domain splitting (YY14) despite its theoretical complexity is much larger. It is worth noticing that the implementation of our method can be further improved by using GPU-based or parallel computing techniques.

However, the accuracy evaluation results (see Table 1) show that our method achieves best approximation quality among the discussed methods. We evaluate the precision using $E_{\max}$ and PSNR measures. Consider I^e is the exact result of L^1 Gauss transform, I^a is the approximation achieved by a given algorithm, and $d_i = |I_i^e - I_i^a|$. $E_{\max}$ is calculated using formula

$$E_{\max} = \max_{1\le i\le N} d_i.$$

We also use peak signal-to-noise ratio (PSNR) [2] to measure the performance of our algorithm according to the equation

$$\text{PSNR} = -10\log\left(\sum_{i=1}^{N}\left(\frac{d_i}{\max(I_i^e, I_i^a)}\right)^2\right).$$

We performed linear image smoothing by the following normalized convolutions for each color channel:

$$\frac{\int G(\mathbf{x}-\mathbf{y})I(\mathbf{y})d\mathbf{y}}{\int G(\mathbf{x}-\mathbf{y})d\mathbf{y}} \rightarrow \frac{J(\mathbf{x}_j)}{\sum_i^N G(\mathbf{x}_j - \mathbf{x}_i)}$$

where the denominator is also obtained by our method convolving L^1 Gaussian with the image whose intensity is equal to one everywhere.

Fig. 3 illustrates the smoothing results using naive implementation (Exact), our method, Box kernel, and FDCT algorithms. The gradient magnitude $|\nabla I|$ of smoothed images on Figs. 4 and 6 show that, in contrast to FDCT and box kernel, our method does not produce some undesirable artifacts and is extremely close to the exact implementation.

Table 1. Precision and speed evaluation results (speed measured in Mpix/sec).

	Our	YY14	FDCT	Box
*E*max	**1.8×10^{-11}**	3.8×10^{-10}	0.44	3.73
PSNR	**291.05**	281.81	58.98	41.45
Speed	7.19	**9.76**	3.37	8.58

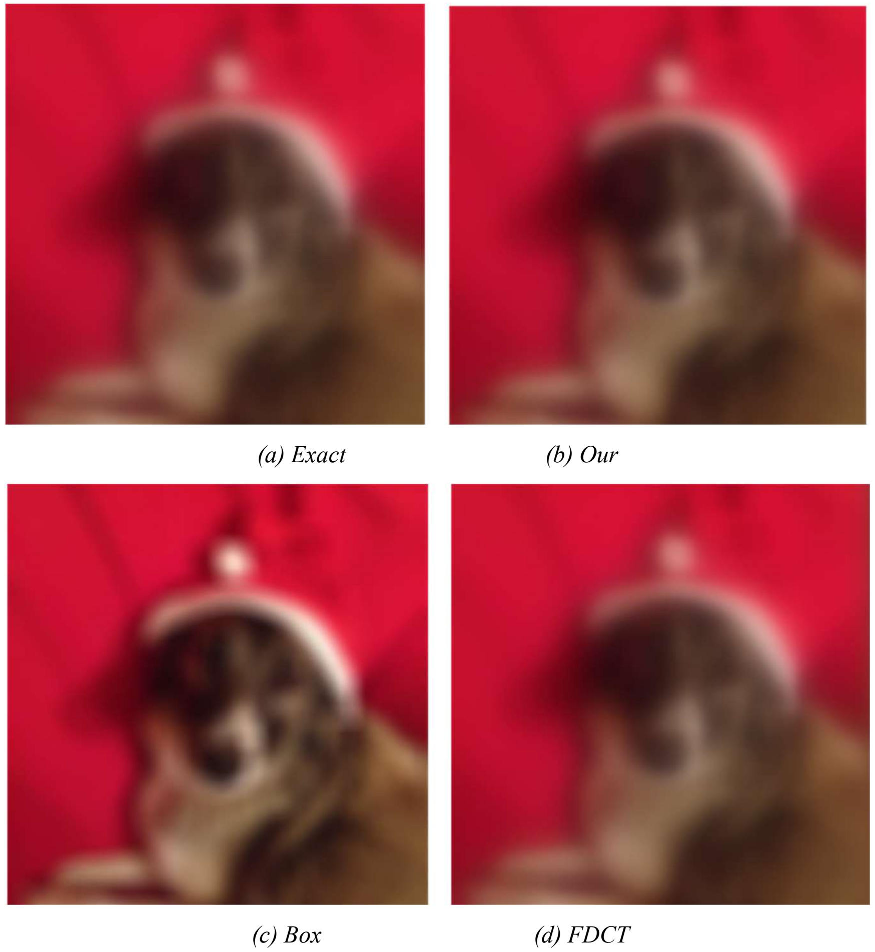


Fig. 3. Results of smoothing (σ = 20), where the input image is given by Fig.2a.

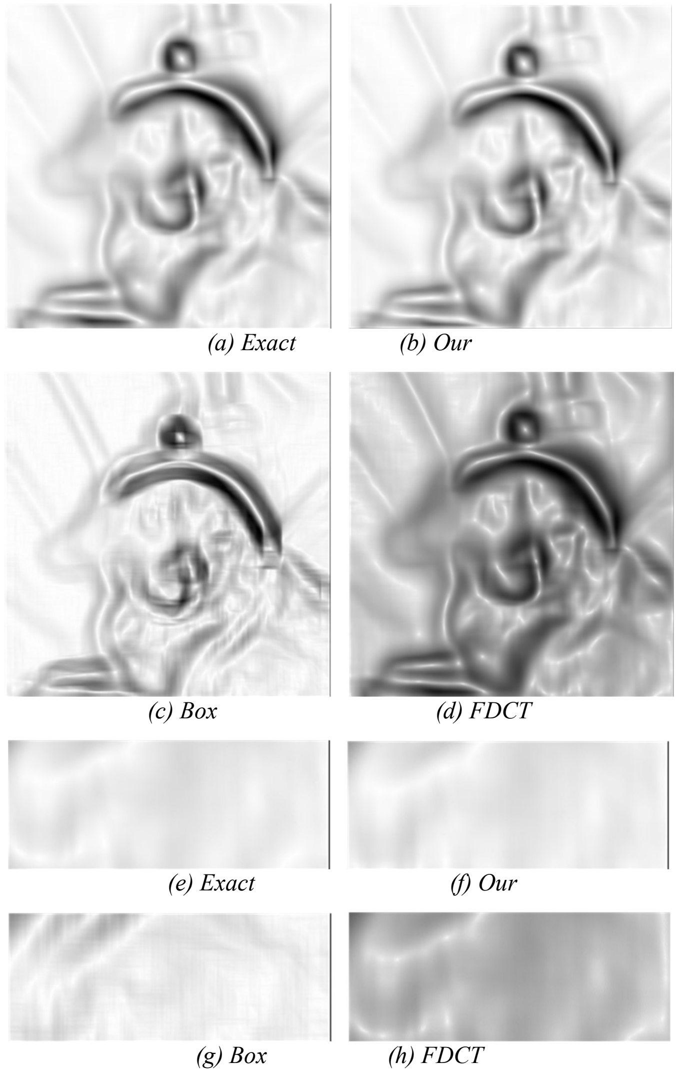


Fig. 4: Visualisation of $|\nabla I|$ for comparison of artifacts (σ = 20).

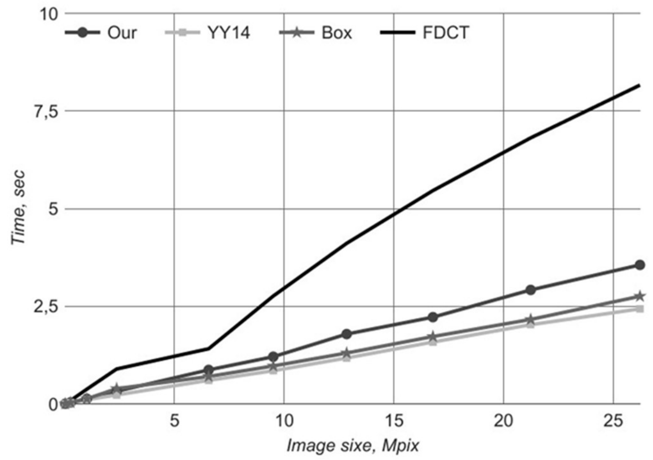
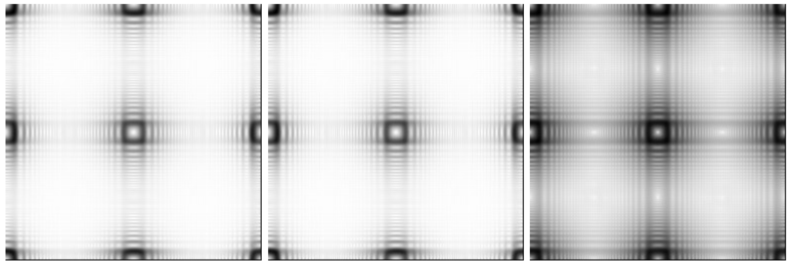
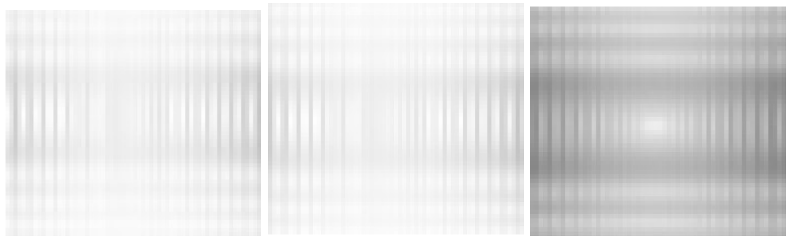


Fig. 5: Timing with respect to image size (averaged by σ).



(a) Exact *(b) Our* *(c) FDCT*



(a) Exact *(b) Our* *(c) FDCT*

Fig. 6: Visualisation of |∇I| for comparison of artifacts of FDCT (σ = 20), where the input image is given by Fig.2b.

5. Edge-Aware Filtering

The proposed algorithm for Gauss transform approach can be applied in various computer vision tasks. We present one of the possible applications of our method by introducing the novel approach for improving the so-called guided filter [12].

Guided filter is categorized into a joint image filtering technique consisting of two input images where one of them is called guidance image, and reflects guidance colors into the other input. One of the most popular joint image filters is the joint bilateral filter [14] which averages the neighbouring colors using the weights that depend on the guidance image. Guided filter is an approach for joint image filtering that allows to overcome a problem with the undesirable gradient reversal artifacts that joint bilateral filter suffers from. Besides edge-aware filtering, it has various image processing applications such as matting, flash-noflash synthesis, HDR-compression, and haze removal.

Consider a point set $\mathbb{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i = (x_i, y_i) \in \mathbb{R}^2$, a guidance image $g = g(\mathbf{x}) \in \mathbb{R}$, an input image $I(\mathbf{x}_i) \in \mathbb{R}$, a desired output image $H(\mathbf{x}_i) \in \mathbb{R}$, and an image region $\Omega(\mathbf{x})$ centered at $\mathbf{x}$. The guided filter is defined as the following linear transformation:

$$H(\mathbf{y}) = ag(\mathbf{y}) + b, \mathbf{y} \in \Omega(\mathbf{x}),$$

where $a, b \in \mathbb{R}$ are the coefficients constant in $\Omega(\mathbf{x})$ that depend on the input image I. Such representation is very useful for image processing tasks, since it preserves the gradient extrema $\nabla H = a\nabla g$, and hence the edges of the guidance image. The coefficients a and b are obtained using the linear ridge regression model [15]:

$$K(a,b) = \sum_{\mathbf{y}\in\Omega(\mathbf{x})} W(\mathbf{x}-\mathbf{y})((ag(\mathbf{y}) + b - I(\mathbf{y}))^2 - \epsilon a^2),$$

where $W(\mathbf{x}-\mathbf{y})$ is the weight that determines the importance of the point $\mathbf{y}$ in $\Omega(\mathbf{x})$ and ϵ is the regularization parameter. One can obtain values a and b by minimizing $K(a,b)$: $\frac{\partial}{\partial a}K(a,b) = 0$ and $\frac{\partial}{\partial b}K(a,b) = 0$. This leads to the following representation:

$$a = \frac{f(Ig) - f(I)f(g)}{f(g^2) - f(g)^2 + \epsilon}, \quad b = f(I) - af(g). \qquad (7)$$

Here $f(*)$ is an averaging function. Since a point $\mathbf{y}$ is included in many overlapping regions $\Omega(\mathbf{x})$ and values a and b for $\mathbf{y}$ are different for each region, the final coefficients are found by averaging over all possible values of $\mathbf{y}$:

$$H(\mathbf{x}) = f(a)g(\mathbf{x}) + f(b). \qquad (8)$$

Guided filtering of color images involves inversion of 3×3 coefficient matrix in order to solve the equation (7) (see [12] for further details). If we set $I \equiv g$, then the guided filter preserves salient edges while smoothing the flat regions (edge-aware filtering). In the simplest case of $I \equiv g$ and I being is a grayscale image, computing guided filter involves performing 4 smoothing operations (e.g. $f(I), f(I^2), f(a), f(b)$), and it takes 33 smoothing operations for a color image if $I \neq g$. Which is why the choice of the smoothing operator $f(*)$ is crucial, since it determines the overall speed and quality of filtering. Authors of the guided filter [12] suggested employing classic L^2 Gauss transform or box kernel method but prefer the latter due to its simplicity and speed despite the fact that box kernel produces undesired artifacts discussed above.

We introduce the new approach for computing guided filtering where our L^1 Gauss transform algorithm is employed for $f(*)$ instead of the box kernel method. As it was shown before, our algorithm gives a much higher quality of smoothing, and this allows us to eliminate smoothing of $f(a)$ and $f(b)$ in the equation (8):

$$H(\mathbf{x}) = ag(\mathbf{x}) + b \qquad (9)$$

Thus, using our algorithm involves 2 operations of $f(*)$ compared to 4 operations in the original method if $I \equiv g$ (grayscale case), and 21 operations compared to 33 operations if $I \neq g$ and both of them are color images.

We examined edge-aware filtering on color images, where the number of $f(*)$ is equal to 21 for the box kernel method and 10 for our approach (9 operations for smoothing of the coefficients and one operation for normalization). As seen on the Figs. 7 and 9, our approach with the reduced amount of smoothing operations $f(*)$ gives quality of edge-aware filtering higher than [12] with the box kernel method, and is faster (0.24 and 0.28 sec for Figs. 9a and 9d respectively).

We examine the differences of equations (8) and (9) in terms of filtering quality on Figs. 9 and 10, which show us that the box kernel method causes artifacts similar to linear filtering case.

We also applied our approach for the detail enhancement filter defined by:

$$D(\mathbf{x}) = I(\mathbf{x}) + \tau(I(\mathbf{x}) - H(\mathbf{x})),$$

where τ is the enhancement parameter. The experiments show that applying our approach for detail enhancement filtering gives high quality results (see Fig. 8).

Fig.10: Edge-aware filtering results (σ=8, ε=0.04). a: input image, b-d: visualization of gradients $|\nabla H|$ of edge-aware filtering via our approach, eq. (9) and box kernel using eqs. (9) and (8) respectively.

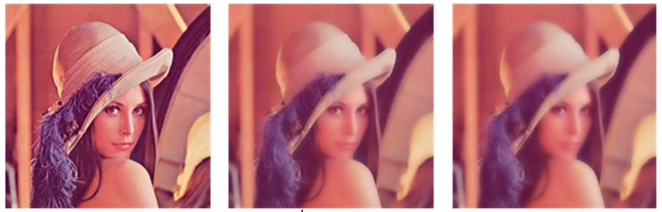
6. Conclusion

In this paper[1] we presented a novel and fast approximation method for L^1 Gauss 2D image transforms. Series of numerical experiments have shown that our method is generally more accurate than the conventional methods and faster than the widely used FFT. We also demonstrated capability of the proposed method in image smoothing application where the conventional box kernel averaging and FFT both suffer from undesirable artifacts. Despite our method is slightly slower than the separable implementations of 1D algorithm [11], this approach can be efficiently used for non-uniformly distributed points.

We have also proposed a novel approach for improving the guided filtering [12] via our L^1 Gauss transform and showed its advantages in terms of quality and speed over [12].

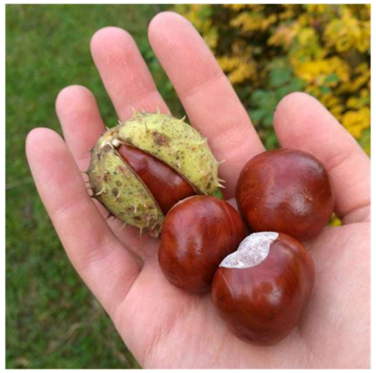
[1] It is an extension of our previous work [16]. The main difference from [16] is the novel approach to joint image filtering and its numerical experiments.

Our method is applicable only to uniformly distributed structures, such as images. Hence our future work includes extending the proposed method to higher-dimensional nonuniform cases which can be done for example by using treelike structures. We also would like to investigate possible applications of the proposed method to various machine learning and image processing tasks, such as regression, segmentation, and registration.

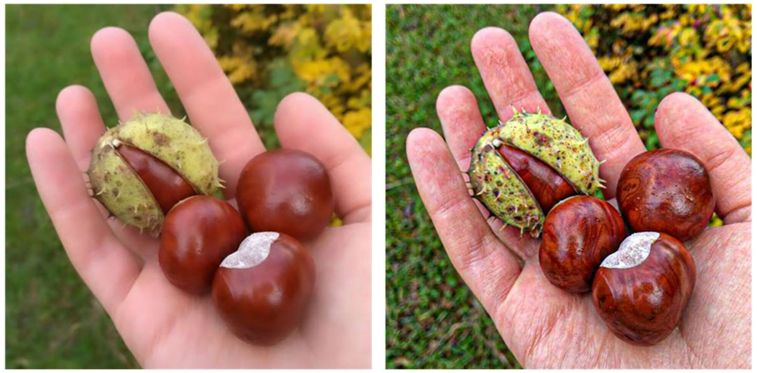


(a) Input *(b) L^1 GT (#f: 10)* *(c) Box (#f: 21)*

Fig. 7: Edge-aware filtering results (σ=8, ε=0.0016).

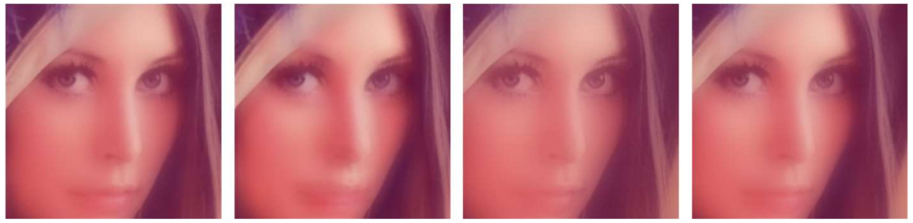


(a) Input

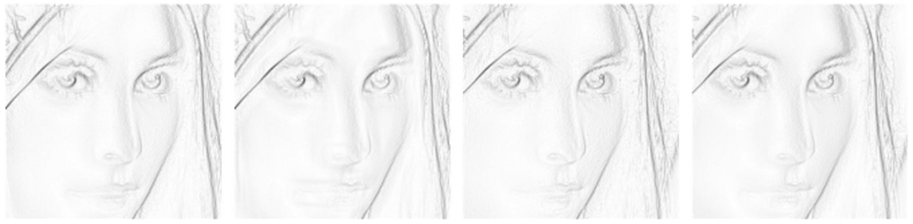


(b) Edge-aware filtering *(c) Detail enhancement*

Fig. 8: Our results of edge-aware filtering and detail enhancement (σ=8, ε=0.04, τ=3).

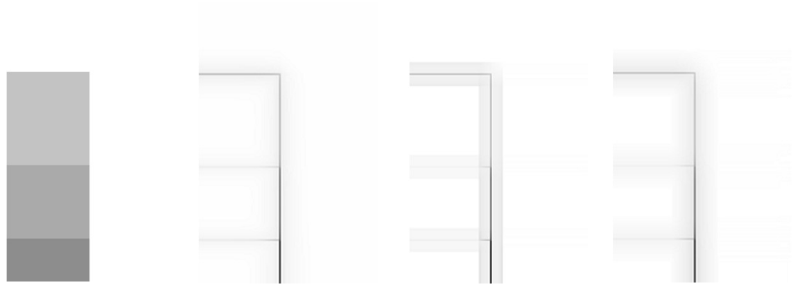


(a) L^1 GT (# f: 10) *(b) Box (# f: 9)* *(c) L^1 GT (#f: 22)* *(d) Box (#f: 21)*



(e) |∇H| L^1 GT of (a) *(f) |∇H| Box of (b)* *(g) |∇H| L^1 GT of (c)* *(h) |∇H| Box of (d)*

Fig. 9: Edge-aware filtering results (σ=8, ε=0.0016). a: L^1 Gauss transform with eq. (9), b: using box kernel with eq. (9), c: L^1 Gauss transform with eq. (8), d: box kernel with eq. (8). e-h: visualization of |∇H| of the corresponding images.



(a) Input *(b) |∇H| Our (#f: 10)* *(c) |∇H| Box (#f: 9)* *(d) |∇H| Box (#f: 21)*

Acknowledgements

We would like to thank the anonymous reviewers of this paper for their valuable comments. This work was supported in part by Grants-in-Aid for Scientific Research of Japan (15H05954 and 16K15019).

References

[1]. A. Elgammal, R. Duraiswami, and L. Davis, “Efficient kernel density estimation using the fast Gauss transform with applications to color modeling and tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 25, no. 11, pp. 1499–1504, 2003.

[2]. S. Paris and F. Durand, “A fast approximation of the bilateral filter using a signal processing approach,” in *Proc. of European Conference on Computer Vision (ECCV)*. Springer, 2006, pp. 568–580.

[3]. L. Greengard and J. Strain, “The fast Gauss transform,” *SIAM Journal on Scientific and Statistical Computing*, vol. 12, no. 1, pp. 79–94, 1991.

[4]. D. Lee, A. Gray, and A. Moore, “Dual-tree fast Gauss transforms,” *Advances in Neural Information Processing Systems (NIPS)*, vol. 18, pp. 747–754, 2006.

[5]. C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis, “Improved fast Gauss transform and efficient kernel density estimation.” in *Proc. of International Conference on Computer Vision (ICCV)*, vol. 1, 2003, pp. 464–471.

[6]. A. Adams, N. Gelfand, J. Dolson, and M. Levoy, “Gaussian kd-trees for fast high-dimensional filtering,” in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, ACM, 2009.

[7]. E. Dougherty, *Digital Image Processing Methods*. CRC Press, 1994.

[8]. R. Deriche, “Fast algorithms for low-level vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 12, no. 1, pp. 78–87, 1990.

[9]. D. Lang, M. Klaas, and N. de Freitas, “Empirical testing of fast kernel density estimation algorithms,” University of British Columbia, Technical Report UBC TR-2005-03, 2005.

[10]. P. Getreuer, “A survey of Gaussian convolution algorithms,” *Image Process. On Line*, vol. 3, pp. 276–300, 2013.

[11]. S. Yoshizawa and H. Yokota, “Fast L^1 Gaussian convolution via domain splitting,” in *Proc. of IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 2908–2912.

[12]. He, Kaiming, Jian Sun, and Xiaoou Tang. "Guided image filtering", *IEEE Transactions on Pattern Analysis and Machine Intelligence* (TPAMI), vol. 35, no. 6, pp. 1397-1409, 2013.

[13]. T. Ooura, *General Purpose FFT (Fast Fourier/Cosine/Sine Transform) Package*. www.kurims.kyoto-u.ac.jp/~ooura/fft.html, 2006.

[14]. Kopf, Johannes, et al. "Joint bilateral upsampling." *ACM Transactions on Graphics (TOG)*, vol. 26. no. 3. ACM, 2007.

[15]. Tikhonov, Andrey. "Solution of incorrectly formulated problems and the regularization method." *Soviet Meth. Dokl.* 1965, Vol. 163, No. 3.

[16]. D. Bashkirova, S. Yoshizawa, R. Latypov and H. Yokota. “Fast L1 Gauss 2D Image Transforms”, *in Proc. of Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE)*, Institute for System Programming, RAS, pp. 145-149, 2017. Available at http://syrcose.ispras.ru/2017/SYRCoSE2017_Proceedings.pdf, accessed June 10, 2017.

Быстрое L1-преобразование Гаусса для сглаживания изображений с сохранением границ

[1,2]Дина Башкирова <dina.bashkirova@riken.jp>
[1]Шин Йошидзава <shin@riken.jp>
[2]Рустам Латыпов <roustam.latypov@kpfu.ru>
[1]Хидео Йокота <hyokota@riken.jp>
[1]Image Processing Research Team, RIKEN Center for Advanced Photonics, RIKEN
2-1, Hirosawa, Wako, Saitama, 351-0198, Japan
[2]Институт вычислительной математики и информационных технологий,
Казанский (Приволжский) Федеральный Университет,
420008 Россия, г. Казань, Кремлевская 35

Аннотация. Преобразование Гаусса, также как и его дискретный аналог, является важнейшим инструментом во множестве математических дисциплин и находит свое применение во многих научных и инженерных областях, таких как математическая статистика и теория вероятностей, физика, математическое моделирование, машинное обучение и обработка изображений и прочие. Ввиду высокой вычислительной сложности преобразования Гаусса (квадратичная сложность относительно количества точек и экспоненциальная — относительно размерности точек), необходимы эффективные и быстрые методы его аппроксимации, обладающие большей точностью по сравнению с существующими ныне методами, такими как Быстрое Преобразование Фурье или оконное преобразование. В данной статье предложен новый метод аппроксимации преобразования Гаусса для равномерно распределенный множеств точек (например, двумерных изображений), основанный на использовании $\boldsymbol{L^2}$ метрики и метода разделения доменов. Такой подход позволяет значительно сократить количество вычислительных операций путем выполнения предварительных вычислений, и снизить вычислительную сложность метода до линейной. Результаты ряда численных экспериментов показали, что разработанный алгоритм позволяет получить более высокую точность аппроксимации без потери скорости вычисления в сравнении со стандартными методами. Также в качестве примера применения предлагаемого алгоритма была разработана новая схема смежной фильтрации изображения. Было показано, что новый фильтр на основе быстрого $\boldsymbol{L^1}$ преобразования Гаусса позволяет получить результат более высокого качества при сопоставимой скорости вычисления и при этом избежать появления нежелательных артефактов в результате обработки, таких как эффект ореола.

Ключевые слова: фильтр Гаусса, распреледение Лапласа, быстрый метод аппроксимации

DOI: 10.15514/ISPRAS-2017-29(4)-4

Для цитирования: Башкирова Д.Р., Йошидзава Ш., Латыпов Р.Х., Йокота Х. Быстрое L1-преобразование Гаусса для сглаживания изображений с сохранением границ. *Труды ИСП РАН*, том 29, вып. 4, 2017 г., стр. 55-72 (на английском). DOI: 10.15514/ISPRAS-2017-29(4)-4

Список литературы

[1]. A. Elgammal, R. Duraiswami, and L. Davis, “Efficient kernel density estimation using the fast Gauss transform with applications to color modeling and tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 25, no. 11, pp. 1499–1504, 2003.
[2]. S. Paris and F. Durand, “A fast approximation of the bilateral filter using a signal processing approach,” in *Proc. of European Conference on Computer Vision (ECCV)*. Springer, 2006, pp. 568–580.
[3]. L. Greengard and J. Strain, “The fast Gauss transform,” *SIAM Journal on Scientific and Statistical Computing*, vol. 12, no. 1, pp. 79–94, 1991.
[4]. D. Lee, A. Gray, and A. Moore, “Dual-tree fast Gauss transforms,” *Advances in Neural Information Processing Systems (NIPS)*, vol. 18, pp. 747–754, 2006.
[5]. C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis, “Improved fast Gauss transform and efficient kernel density estimation.” in *Proc. of International Conference on Computer Vision (ICCV)*, vol. 1, 2003, pp. 464–471.
[6]. A. Adams, N. Gelfand, J. Dolson, and M. Levoy, “Gaussian kd-trees for fast high-dimensional filtering,” in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, ACM, 2009.
[7]. E. Dougherty, *Digital Image Processing Methods*. CRC Press, 1994.
[8]. R. Deriche, “Fast algorithms for low-level vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 12, no. 1, pp. 78–87, 1990.
[9]. D. Lang, M. Klaas, and N. de Freitas, “Empirical testing of fast kernel density estimation algorithms,” University of British Columbia, Technical Report UBC TR-2005-03, 2005.
[10]. P. Getreuer, “A survey of Gaussian convolution algorithms,” *Image Process. On Line*, vol. 3, pp. 276–300, 2013.
[11]. S. Yoshizawa and H. Yokota, “Fast L^1 Gaussian convolution via domain splitting,” in *Proc. of IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 2908–2912.
[12]. He, Kaiming, Jian Sun, and Xiaoou Tang. "Guided image filtering." *IEEE Transactions on Pattern Analysis and Machine Intelligence* (TPAMI), vol. 35, no. 6, pp. 1397-1409, 2013.
[13]. T. Ooura, *General Purpose FFT (Fast Fourier/Cosine/Sine Transform) Package*. www.kurims.kyoto-u.ac.jp/~ooura/fft.html, 2006.
[14]. Kopf, Johannes, et al. "Joint bilateral upsampling." *ACM Transactions on Graphics (TOG)*, vol. 26. no. 3. ACM, 2007.
[15]. Тихонов, А. Н. "О некорректных задачах линейной алгебры и устойчивом методе их решения." *ДАН СССР, 1965,* 163.3.
[16]. D. Bashkirova, S. Yoshizawa, R. Latypov and H. Yokota. “Fast L1 Gauss 2D Image Transforms”, *in Proc. of Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE)*, Institute for System Programming, RAS, 2017, pp. 145-149. Доступно по ссылке http://syrcose.ispras.ru/2017/SYRCoSE2017_Proceedings.pdf, дата обращения 10.06.2017.