Effective Five Directional Partial Derivatives-based Image Smoothing and a Parallel Structure Design

Choongsang Cho and Sangkeun Lee, Senior Member, IEEE

Abstract

Image smoothing has been used for image segmentation, image reconstruction, object classification, and three-dimensional content generation. Several smoothing approaches have been used at the pre-processing step to retain the critical edge, while removing noise and small details. However, they have limited performance, especially in removing small details and smoothing discrete regions. Therefore, to provide fast and accurate smoothing, we propose an effective scheme that uses a weighted combination of the gradient, Laplacian, and diagonal derivatives of a smoothed image. Additionally, to reduce computational complexity, we designed and implemented a parallel processing structure for the proposed scheme on a graphics processing unit (GPU). For an objective evaluation of the smoothing performance, the images were linearly quantized into several layers to generate experimental images and the quantized images were smoothed using several methods for reconstructing the smoothly changed shape and intensity of the original image. Experimental results showed that the proposed scheme has higher objective scores and better successful smoothing performance than similar schemes, while preserving and removing critical and trivial details, respectively. For computational complexity, the proposed smoothing scheme running on a GPU provided 18 and 16 times lower complexity than the proposed smoothing scheme running on a CPU and the L0-based smoothing scheme, respectively. In addition, a simple noise reduction test was conducted to show the characteristics of the proposed approach; it reported that the presented algorithm outperforms the state-of-the art algorithms by more than 5.4 dB. Therefore, we believe that the proposed scheme can be a useful tool for efficient image smoothing.

Index Terms

Image smoothing, Laplacian, parallel processing, low-complexity.

I. INTRODUCTION

The edge information of an image is important for human visual perception [1] and is commonly used in many image processing applications. For better performance in image reconstruction, segmentation, object classification, and three-dimensional content generation, an edge-preserving smoothing scheme is used at the pre-processing step to preserve critical edges so as to maintain the main structure of a given image, while removing trivial details which are small details that need to be smoothed or continuously changed [1]–[5]. Several smoothing approaches have been investigated at the pre-processing step to retain the critical edge. Bilateral filter (BLF)-based methods controlled by sparse and range filter characteristics have been used for image smoothing and noise reduction, while preserving edges [2], [6]. A guided image filter using a guidance image has an edge-preserving characteristic of a BLF; its performance is controlled by a regularization parameter and the local window size [3]. An estimation method using piecewise smoothing and anisotropic diffusion methods with an edge-stopping function have been applied to image noise reduction and segmentation [4], [7].

The BLF method, depending on the pixel intensity and its neighborhood, is heavily affected by the filter size and parameters of the intensity range, but its smoothing performance is weak for high-contrast [5]. The guided image filter scheme has a lower computational complexity and higher performance than BLF, but its performance depends strongly on the guided image [3]. The performance of the diffusion-based method is dependent on the structure of the edge-stopping function [4], [7]. Additionally, the total variation-based scheme has been used for image denoising and reconstruction, but it has ringing and staircase artifacts around big edges [8].

A sparse gradient counting scheme in an optimization framework has been proposed with a discrete number of intensity changes among neighboring pixels [1]. L0 smoothing based on sparse gradient counting has high performance with global optimization. However, the scheme has limited performance at removing trivial details and at smoothing discrete regions.

In this work, to enhance performance, we weighted the partial derivative operators in five different directions, and then merged the result into a sparse counting scheme to consider edge regions more accurately. In addition, to reduce computational complexity, we implemented the proposed scheme on a parallel processor, such as a graphics processing unit (GPU), because the smoothing method using global optimization has a high computational complexity. A parallel structure on a GPU composed of hundreds of cores [9]–[11] has been designed, and it is widely used for fast image processing. In particular, to evaluate objectively the smoothing performance, we linearly quantized images with different geometric shapes and intensity gradations into multiple layers. The quantized images were processed using the smoothing methods to generate a smoothly changed shape.
comparable to that of the original images, and then peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [12] were used to compare the results.

This paper is structured as follows. In Section II, the sparse counting scheme and LO smoothing method are explained numerically. The proposed scheme and its implementation are described in Section III, followed by experimental result comparisons and analysis in Section IV. Finally, Section V summarizes the algorithm and concludes with some discussions.

II. BACKGROUND: GRADIENT SPARSENESS COUNTING AND SMOOTHING SCHEMES

Smoothing by counting the change in gradient magnitude has been applied to edge-preserving smoothing and segmentation [1], [13]. To count the non-zero gradient magnitude at a pixel \( p \), it is necessary to calculate the gradient of a smoothed image \( \nabla S_p = (\partial_x S_p, \partial_y S_p)^T \). Counting of the non-zero gradient magnitude is defined as

\[
C(S) = \# \{ p | |\partial_x S_p| + |\partial_y S_p| \neq 0 \}
\]  
(1)

and the cost function for the smoothing is defined in terms of the pixel wise difference and discontinuity measurement as

\[
F = \min_S \left\{ \sum_p (S_p - I_p)^2 + \lambda \times C(S) \right\}
\]  
(2)

where \( S \) is the smoothed image, \( \lambda \) is a weighting factor for controlling the significance of \( C(f) \), and \( S_p \) and \( I_p \) indicate the intensity values of the smoothed and input images, respectively, at a pixel point \( p \).

To solve the function, we adopt the half quadratic splitting method [14], introducing the auxiliary variables \( h_p \) and \( v_p \) as

\[
F = \min_{S,h_p,v_p} \left\{ \sum_p (S_p - I_p)^2 + \lambda C(h_p,v_p) + \beta \left( (\partial_x S_p - h_p)^2 + (\partial_y S_p - v_p)^2 \right) \right\}
\]  
(3)

where \( \beta \) is a control parameter of the similarity between auxiliary variables and their gradients. The smoothed image is obtained by applying the partial derivative with respect to \( S \) as

\[
\frac{\partial F}{\partial S} = 2(S_p - I_p) + \beta [2(\partial_x S_p - h_p) \partial_x + 2(\partial_y S_p - v_p) \partial_y]
\]  
(4)

The computational complexity of the smoothed image calculation is reduced by applying the fast Fourier transform (FFT) as

\[
S = \mathcal{F}^{-1} \left( \mathcal{F}(I) + \beta \left( \mathcal{F}(\partial_x) \mathcal{F}(h) + \mathcal{F}(\partial_y) \mathcal{F}(v) \right) \right).
\]  
(5)

where \( \mathcal{F} \) and \( \mathcal{F}^{-1} \) are the FFT and inverse FFT operators, respectively, and \( \mathcal{F}(1) \) is the FFT of the delta function. The introduced variables are calculated by individual estimation using a splitting method [1].

III. IMAGE SMOOTHING SCHEME BASED ON FIVE DIRECTIONAL PARTIAL DERIVATIVES

Smoothing around the critical edge is very important for edge-preserving smoothing. To design high performance smoothing while maintaining structural similarity to the original image and considering more detailed edge information, we propose an effective smoothing method by adopting new elements for measuring the amplitude change used for detail preservation criteria. To consider the degree of edge changes more accurately, we apply the directional partial derivative terms [15], [16]. The proposed cost function is defined by minimizing the mismatch term between the smoothed and the input images, the amplitude change counting term, and the splitting term to solve the different convex functions as

\[
F = \min_{S,x} \left\{ \sum_p (S_p - I_p)^2 + \lambda C(g_x) + \beta G(S_x,g_x) \right\}
\]  
(6)

where \( \lambda \) is a factor for controlling the significance of the counting function, and \( g_x \) is an auxiliary variable set corresponding to \( h_p,v_p,\beta h_p,\beta v_p,\alpha_x \).

To determine whether the edges need to be removed, we count the magnitude change of the edges. In the proposed method, the measurement consists of a weighted summation of the gradient, Laplacian, and diagonal derivative terms as

\[
C(g_x) = \# \left\{ p | \sum_{\alpha_x} |g_x \times \alpha_x| \neq 0 \right\}, \alpha_x \in \{ \beta_x, \epsilon_x, \eta_{hh}, \eta_{vv}, \mu_{tt} \}
\]  
(7)
where \( \omega \) is a set of weight factors for structural similarity terms, and the weight factors are less than 1. Additionally, the summation of weight factors for each horizontal and vertical direction is 1 as \( e_h + \eta_h + \mu_h = 1 \) and \( e_v + \eta_v + \mu_v = 1 \), respectively. The weight factors for gradient terms are greater than the other factors such as \( \{ e_h, e_v \} > \{ \eta_h, \eta_v, \mu_h, \mu_v \} \) since the second derivative is more sensitive to the noise than the first derivative.

The similarity between auxiliary variables \( g_s \) and edge attributes \( S_s \) is computed to solve the convex problem based on a splitting approach. Here, the weighting factors are applied to the elements of the edge attributes, and the contribution of the elements is controlled by the weighting factors as

\[
G(S_s, g_s) = \sum_{(s, g, \omega)} \omega_s (S_s - g_s)^2
\]

### A. Calculation of \( S \)

To estimate the smoothed image, we must calculate the global minimum of the objective function, excluding terms not involved in computing \( S \). Thus, the objective function can be rewritten after removing the counting term as

\[
\min_S \left\{ \sum_p (S_p - I_p)^2 + \beta G(S_s, g_s) \right\}
\]

The solution is obtained by applying the partial derivative with respect to \( S \) to this function. In particular, computational complexity can be reduced by applying FFT [11] as

\[
S = \mathcal{F}^{-1} \left( \frac{\mathcal{F}(I) + \beta \sum_{(\omega, g, \omega_s)} \omega, \mathcal{F}(\partial_\omega) \mathcal{F}(g_s)}{\mathcal{F}(1) + \beta \sum_{(\omega, \omega_s)} \omega \mathcal{F}(\partial_\omega) \mathcal{F}(g_s)} \right)
\]

Note that addition, multiplication, and division are component-wise operations.

### B. Calculation of edge attribute elements by splitting

In a similar manner, to obtain \( g_s \), we represent the cost function in Eq. 6 by excluding terms not involved in \( g_s \) as

\[
\min_{g_s} \left\{ \sum_p G(S_s, g_s) + \frac{\lambda}{\beta} \mathcal{H} \right\}
\]

Next, splitting is applied to the given function because the first term can be decomposed spatially, and the second term is converted into a binary function that returns 1, if \( \sum_{(g, \omega_s)} \omega \g_s \neq 0 \), and 0, otherwise. Additionally, the cost function is rewritten by using the binary function \( H(g_s) \) instead of the counting function \( C(g_s) \) as

\[
\sum_p \min_{g_s} \left\{ G(S_s, g_s) + \frac{\lambda}{\beta} \mathcal{H} \right\}
\]

Then, the cost function energy at each pixel \( p \) is represented as

\[
E_p = \left\{ G(S_s, g_s) + \frac{\lambda}{\beta} \mathcal{H} \right\}
\]

The optimized solution for \( g_s \) is obtained when the energy is minimum. For this, two minimum conditions are defined by \( \sum_{(g, \omega)} (\omega, S_s)^2 \leq \lambda / \beta \) and \( \sum_{(g, \omega)} (\omega, S_s)^2 > \lambda / \beta \). In the first condition, when the binary function is 1 for \( \sum_{(g, \omega_s)} \omega \g_s \neq 0 \), the energy function is represented according to the similarity term \( G(S_s, g_s) \geq 0 \) from Eq. 8 as

\[
E_p (g_s) = G(S_s, g_s) + \frac{\lambda}{\beta} \geq \frac{\lambda}{\beta} \geq \sum_{(g, \omega_s)} (\omega, S_s)^2
\]

By this condition, the minimum energy is determined as \( E_p = (\omega, S_s)^2 \) under \( g_s = 0 \) because the mismatch term is always greater than or equal to zero. In the second condition, when the binary function is 0 for \( \sum_{(g, \omega_s)} \omega \g_s \neq 0 \), the energy function is represented as

\[
E_p (g_s) = G(S_s, g_s)
\]

In this condition, the minimum energy is obtained at \( G(S_p, g_s) = 0 \) because the mismatch term is always greater than or equal to zero. Then, it is produced under \( g_s = S_s \) since \( \omega_s \neq 0 \). Based on the condition, the minimum energy is calculated as \( E_p = \lambda / \beta \).
Therefore, the solution is summarized under the following condition as

\[
g_\ast = \begin{cases} 
0 & \text{for } \sum_{S_i \in \omega} (\omega_i \cdot S_i) \leq \lambda / \beta \\
S_i & \text{otherwise}
\end{cases}
\]

The parameter \( \beta \) is updated with \( \beta(t+1) = \kappa \times \beta(t) \) during the iterations for the optimization processing, and the initial \( \beta(0) \) is defined as \( 2 \times \lambda \) [1]. The \( \lambda \) and \( \kappa \) are empirically defined for applications because they are dependent on the characteristics of the input data and are set to less than 1 and greater than 1, respectively [1]. In the updating process, the parameter \( \beta \) is increased, and thus the decision threshold \( \lambda / \beta \) in the solution is decreased. Therefore, the given image is gradually smoothed via iterative optimization.

C. GPU-based parallel processing for the proposed scheme

Reduction of the computational cost of image smoothing methods is required for several varieties of applications. For this purpose, parallel processor-based complexity reduction can be considered [9]. In this work, an effective and fast image smoothing method on a GPU is presented by designing a parallel processing structure. One CPU is architecturally composed of only a few cores with many cache memories and instruction units [17]–[19]. A CPU core supports several software threads at a time for parallel processing, and its operation is optimized for sequential processing. In contrast, a GPU is architecturally composed of hundreds of cores that execute in parallel with other cores. Each core can execute a sequential thread, and all the cores in the same group execute the same instruction simultaneously. The kernels, which are units of a parallel program on a GPU, are executed sequentially. Especially, the threads for a kernel are identified in terms of a hierarchical indexing structure. In addition, threads are grouped into blocks, and blocks are grouped into a grid. Each thread has a unique local index in its block, and each block has a unique index in the grid [17]–[19].

To design a parallel processing scheme for image smoothing on a GPU, we have to divide an image into several blocks, as shown in Fig. 1. The blocks, which have independent physical memory space and processing units, cannot share input data and results with other blocks, and they cannot send results to the other blocks. The smoothing terms are separated into two parts: one part with unchanged terms and another part with updated terms. The derivative operators (Eq. 10) of the smoothed image and input image belong to unchanged terms, and the others are defined as updated terms. For fast processing, the
calculation of the smoothed image is performed using FFTs, and thus its complexity critically affects smoothing complexity. A two-dimensional-CUDA-based FFT (CuFFT) library [19], a fast FFT on a GPU, is used for the calculation of the smoothed image to reduce computational cost. At the pre-operation step for the unchanged terms, the FFTs of an input image and derivative operators are executed. The updating process consists of three modules. The first is a boundary expansion for the circular shift-based operation, the second is a calculation for the optimal solution in Eq. 12, and the last is a calculation for the smoothed image based on the updated auxiliary variables \( g^* \). In parallel processing, the derivative operation with circular shifting requires boundary expansion. Then, to obtain the optimal solution in Eq. 12, we execute the derivatives of the smoothed image obtained at the previous iteration, and then compute an optimal solution for the auxiliary variables \( g^* \) as in Eq. 16.

In the last module, the smoothed image is obtained from the pre-operated terms and updated auxiliary variables \( g^* \), and then the weight parameter \( \beta \) is updated by multiplying by the increment ratio as \( \beta = \beta \times \text{ratio} \). For the iteration, the updated \( \beta \) is compared with the predefined maximum value, and the smoothing is repeated until the predefined condition is met.

IV. EXPERIMENTAL RESULTS

To evaluate smoothing performance, we compared a BLF-based scheme [2], a guided image filter-based scheme [3], an \( L^0 \) gradient-based approach [1], and the proposed algorithm. The parameters for the compared approaches and proposed scheme used in the experiment are summarized in Table I. Additionally, for the subjective evaluation which is usually conducted with human eye observation, two images, one of which is an image used in [1] and the other is from Kodak test images [20], as shown in Fig. 2, were smoothed. For better visual observation, the red solid line box was enlarged, as shown in Figs. 3 and 4.

![Fig. 2. Subjective evaluation of images for the smoothing scheme: (a) house and ramp and (b) barn and pond.](image-url)

The smoothed images using the BLF-based scheme and the guided image filter-based scheme for the first image are more blurred than those using the \( L^0 \)-based and the proposed schemes. The \( L^0 \) scheme provides better performance for preserving critical details than the BLF-based scheme and the guided image filter schemes. The proposed scheme, which preserves critical details, produces better performance than the \( L^0 \)-based scheme in removing trivial details, as shown in the magnified images of Fig. 3. Similarly, the BLF and the guided image filter-based approaches generated more blurred output than the \( L^0 \)-based and the proposed schemes for the second test image. Note that the proposed scheme has better performance than the \( L^0 \) scheme in removing trivial details, as shown in the enlarged parts of Fig. 4.

For an objective evaluation of the smoothing performance, four 824-by-874 images, which have different geometric shapes with intensity gradation, were used, as shown in Fig. 5. To generate a proper experimental environment, we linearly quantized each image [16] into several layers, as shown in Fig. 6. Then, we smoothed the quantized image with a staircase shape using several comparison methods designed to reconstruct the smoothly changed shape and intensity, to make it look similar to the original image. For better visual observation, a specific region was enlarged (see the red solid-line box). For an objective performance comparison, we measured the PSNR and SSIM scores by comparing the original and the smoothed images.
Fig. 3. Smoothed images. Using (a) a BLF-based scheme, (b) a guided image filter-based scheme, (c) an $L^0$-based scheme, and (d) the proposed scheme.

Fig. 4. Smoothed images. Using (a) a BLF-based scheme, (b) a guided image filter-based scheme, (c) an $L^0$-based scheme, and (d) the proposed scheme.
Fig. 5. Images used for the objective evaluation of the smoothing schemes: (a) a cone, (b) a knot, (c) springs; and (d) a teapot.

Fig. 6. Linearly quantized images for objective evaluation: (a) a cone, (b) a knot, (c) springs; and (d) a teapot.

The PSNR scores of the smoothed images are given in Table II. The BLF and the guided image-based schemes had similar PSNR scores, whereas the $L_0$-based approach had a higher score than the other two methods. In particular, the proposed method outperformed the $L_0$ approach by approximately 1.22 dB and 1.39 dB on average in the MATLAB implementation and on a GPU, respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLF</td>
<td>39.97</td>
<td>37.17</td>
<td>35.53</td>
<td>36.47</td>
<td>37.29</td>
</tr>
<tr>
<td>Guided filter</td>
<td>40.97</td>
<td>35.53</td>
<td>34.43</td>
<td>36.56</td>
<td>36.87</td>
</tr>
<tr>
<td>$L_0$ smoothing</td>
<td>42.52</td>
<td>39.48</td>
<td>36.58</td>
<td>37.54</td>
<td>39.03</td>
</tr>
<tr>
<td>Our scheme in MATLAB</td>
<td>44.58</td>
<td>40.25</td>
<td>37.78</td>
<td>38.37</td>
<td>40.25</td>
</tr>
<tr>
<td>Our scheme on a GPU</td>
<td>44.43</td>
<td>40.20</td>
<td>38.22</td>
<td>38.84</td>
<td>40.42</td>
</tr>
</tbody>
</table>

In a similar manner, SSIM scores are shown in Table III. The BLF and the guided image-based schemes provided similar scores, whereas the $L_0$-based scheme had better results than the two methods by the SSIM criterion. However, it is clearly
shown that the proposed method produces the best results both in the MATLAB and on the GPU implementations.

<table>
<thead>
<tr>
<th>Method</th>
<th>SSIM 1</th>
<th>SSIM 2</th>
<th>SSIM 3</th>
<th>SSIM 4</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLF</td>
<td>0.996</td>
<td>0.957</td>
<td>0.990</td>
<td>0.996</td>
<td>0.985</td>
</tr>
<tr>
<td>Guided filter</td>
<td>0.997</td>
<td>0.954</td>
<td>0.976</td>
<td>0.997</td>
<td>0.981</td>
</tr>
<tr>
<td>$L_0$</td>
<td>0.996</td>
<td>0.995</td>
<td>0.990</td>
<td>0.997</td>
<td>0.995</td>
</tr>
<tr>
<td>Our scheme in MATLAB</td>
<td>0.997</td>
<td>0.997</td>
<td>0.991</td>
<td>0.997</td>
<td>0.996</td>
</tr>
<tr>
<td>Our scheme on a GPU</td>
<td>0.997</td>
<td>0.997</td>
<td>0.992</td>
<td>0.998</td>
<td>0.996</td>
</tr>
</tbody>
</table>

To evaluate the computational complexity for sequential and parallel processing, we implemented the proposed scheme for sequential processing in MATLAB running on a machine with an Intel Core i7-3960 3.3GHz CPU equipped with 28 GB of RAM and implemented the parallel processing in CUDA running on an NVIDIA GeForce TITAN 837-kHz with 2688 cores. The previous four images were used, and the average computational time was measured, as shown in Table IV. For the $L_0$-based and the proposed schemes, iterative operation for the smoothing step was repeated 15 times. The proposed approach requires more computational time than the $L_0$-based scheme for sequential processing in the MATLAB implementation. However, when the proposed scheme was implemented on a GPU, the computational time was reduced by about 16 and 18 times compared to the proposed and $L_0$-based schemes, respectively, implemented on a CPU.

For visual observation, the smoothed results from each scheme are shown in Fig. 7, and the partially enlarged regions including the original, quantized, and resultant images are illustrated in Fig. 8. The quantized images in Fig. 8-(b) have a staircase shape, and they were smoothed by each algorithm. The smoothed images using the previous existing methods have discrete regions, as shown in Fig. 8(c–e). This indicates that the given region needs more smoothing. However, the proposed scheme with sequential and parallel processing generated images that were more continuous than and much closer to the original image in Fig. 8(a).

In addition, the smoothing technique can be a useful tool for noise reduction in the field of signal reconstruction. To effectively show the characteristics of the proposed approach and its robustness to noise, we added normally distributed pseudo-random noise [21] to a 200 $\times$ 200 stripe-shaped image with 20% noise density, as shown in Fig. 9(a). Like in the previous experiments, we performed noise reduction using each smoothing method, and its result is provided in Fig. 9(b–e). The BLF and the guided image filter-based approaches eliminated the noise while preserving the edges, but their outputs still contained much noise. In contrast, the result of the $L_0$-based scheme was much better than those of the two filtering-based approaches even though it did not remove the noise, which looked like salt and pepper [22]. However, the proposed scheme provided much better performance than the other compared approaches, as shown in Figs 9(b–e). For graphical and better observable comparison, intensity values along a single row of each result (see the white-dotted line) were plotted, as shown in Fig. 10. The $L_0$-based scheme gave better noise reduction performance than the two filtering-based approaches, but the noisy values at the spike points were not eliminated because high gradient magnitude was considered as an edge in the $L_0$-based approach. However, the proposed scheme suppressed or removed the noise in the same cases because it considers five directional partial derivatives to reflect the status of neighboring pixels. For numerical and objective comparison, the PSNR score was measured, as shown in Table V. It is clearly shown that the proposed approach gave the highest score. In particular, the proposed method outperformed the $L_0$-based approach by approximately 5.6dB.

<table>
<thead>
<tr>
<th>Method</th>
<th>Noised image</th>
<th>BLF</th>
<th>Guided filter</th>
<th>$L_0$</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR(dB)</td>
<td>20.5</td>
<td>20.8</td>
<td>26.8</td>
<td>39.4</td>
<td>43.0</td>
</tr>
</tbody>
</table>

V. CONCLUSION

In this paper, a five directional partial derivatives-based image smoothing scheme implemented on a parallel processor was presented, which showed efficient and effective performance. In particular, partial derivatives such as gradient, Laplacian, and diagonal derivatives of the smoothed image were weighted and combined to consider edge attributes in detail. Additionally, to reduce computational cost, we separated the smoothing cost function into pre-operation and updated terms for the parallel processing of the proposed scheme. Then, the pre-operation terms were processed before the iterative operation, and the updated terms were computed using the iterative scheme that was implemented in CUDA on a GPU. The experimental results showed
that the proposed scheme could produce effective smoothing while maintaining almost identical smoothing results to the original image when a linearly quantized image was reconstructed. Additionally, the proposed smoothing scheme running on a GPU provided about 18 and 16 times lower complexity on average than the proposed smoothing scheme running on a CPU and the $L^0$-based scheme, respectively. Therefore, we believe that the proposed scheme can be a useful tool for fast image smoothing. Note that related software, including source code for the reproducible images, is available at http://mmc.cau.ac.kr/publications-2/international-journal/.

ACKNOWLEDGMENT

This work was partially supported by the IT R&D program of MSIP/IITP [GK130100, R0126-15-1016, R-20150520-001343] and by the NRF funded by the Ministry of Education, Science and Technology(NRF-2014R1A2A1A11049986).

REFERENCES

Fig. 8. Original and smoothed images by different schemes. Using (a) original data, (b) quantized data, (c) a BLF-based scheme, (d) guided image filter-based scheme, (e) an $L^0$-based smoothing scheme, (f) the proposed scheme in MATLAB, and (g) the proposed scheme on a GPU.

Fig. 9. Comparison of the noise reduction by the smoothing approaches. Using (a) a noised image, (b) a BLF-based scheme, (c) a guided image filter-based scheme, (d) an $L^0$-based smoothing scheme, and (e) the proposed scheme.


Fig. 10. Graphical comparison along a horizontal line of each result. Using (a) a noised image, (b) a BLF-based scheme, (c) a guided image filter-based scheme, (d) an $L^0$-based smoothing scheme, and (e) the proposed scheme.
ChoongSang Cho received his B.S. degree in electronic engineering from Suwon University, Gyeonggi, Korea, and his M.S. degree from the Department of Information and Communications at Gwangju Institute of Science and Technology (GIST), Gwangju, Korea. Since 2008, he has been a researcher at the Multimedia IP Research Center, Korea Electronics Technology Institute (KETI), Gyeonggi, Korea. Currently, he is also pursuing a PhD degree at the Graduate School of Advanced Imaging Science, Multimedia & Film of Chung-Ang University, Seoul, Korea. His research interests include digital hologram, three-dimensional imaging, and image segmentation.

Sangkeun Lee (SM’12) received his BS and MS degrees in electronic engineering from Chung-Ang University, Seoul, Korea, in 1996 and 1999, respectively, and a PhD degree in electrical and computer engineering from Georgia Institute of Technology, Atlanta, GA, in 2003. He is an associate professor in the Graduate School of Advanced Imaging Science, Multimedia & Film at Chung-Ang University, Seoul, Korea. From 2003 to 2008, he was a staff research engineer with the Digital Media Solutions Lab, Samsung Information Systems America, Irvine, CA, USA, where he was involved in the development of video processing and enhancement algorithms (DNIe) for Samsung’s HDTV. His current research and development interests include digital video and image processing, vision computing, especially video analysis/synthesis, denoising, compression for UHDTV and multimedia applications, as well as CMOS image sensors. He is also a senior member of the IEEE.