Vis Comput (2018) 34:83–92
https://doi.org/10.1007/s00371-016-1315-z

ORIGINAL ARTICLE

Constant time texture filtering

Hanli Zhao[1] · **Lei Jiang**[1] · **Xiaogang Jin**[2] · **Hui Du**[3] · **Xujie Li**[1]

Published online: 8 September 2016

Abstract We present a novel texture-filtering method which can effectively separate the main image structures from textures even with high variations. Our nonlinear image decomposition is based on a variant of the weighted-median filter which incorporates structure and texture information into the guidance image. To guarantee effective texture filtering, the guidance image not only contains prominent structure edges of the input, but also reduces the contrast in texture regions. We develop a constant time algorithm for the generation of guidance image by formulating the calculation of local extrema as a histogram volume aggregation problem. The local nature of the algorithm enables an efficient parallel GPU implementation. In addition, we demonstrate the effectiveness of our texture-filtering method in the context of detail enhancement, JPEG artifact removal, inverse halftoning, image segmentation, edge detection, and image stylization.

Keywords Edge-preserving smoothing · Texture smoothing · Bilateral filter · Median filter

1 Introduction

Textures are composed of repeating patterns that are similar in appearance and local statistics. In our world, there are many natural scenes or human-created artistic images, in which meaningful structures are blended with complex texture elements. It is easy for the human visual system to extract image structures without being disturbed by unimportant texture information. However, such an image decomposition task becomes challenging for the computer. In fact, the image decomposition is a very important operation with a variety of applications in image processing and editing.

Currently, there are two types of operations to perform image decomposition. The first type utilizes edge-preserving smoothing and effectively retains large-scale structures while removing fine-scale details. This type of algorithms [7,19,20,22] takes high-contrast regions as boundaries of large-scale structures. However, fine-scale textures usually contain high-contrast oscillations, and these algorithms will erroneously view such textures as structures. The other type separates image structures and textures based on local variations of image values. However, they may extract jaggy boundaries of textures as part of structure boundaries [2,23,24] or may fail to smooth our high-contrast texture patches [4,6]. In addition, there is no constant time texture-filtering algorithm with regard to the size of filtering patch. As a consequence, it would require much time to smooth out large-scale textures with a very large size of filtering patch.

In this paper, we present a novel method for texture-filtering based on a modification of the weighted-median filter. The weighted-median filter substitutes the value of a pixel with the weighted median in its neighborhood and avoids the weighted summation of multiple neighboring values. This simple property makes the weighted-median filter to outperform the bilateral filter in terms of the preservation of structure edges. The additional computation involved is the generation of the guidance image. We generate the guidance image by interpolating the input image and its blurred image with a modified relative total variation map [4]. As a

✉ Hanli Zhao
hanlizhao@wzu.edu.cn

1 Intelligent Information Systems Institute, Wenzhou University, Wenzhou, China

2 State Key Lab of CAD & CG, Zhejiang University, Hangzhou, China

3 New Media College, Zhejiang University of Media and Communications, Hangzhou, China

result, the guidance image not only contains prominent structure edges of the input, but also reduces the contrast in texture regions. To achieve a constant time complexity for each pixel in the whole pipeline, we formulate the calculation of local extremal values as a histogram volume aggregation problem. The whole algorithm in our approach is well suited in a parallel GPU implementation, allowing an efficient separation of structures and textures. Our texture-filtering method benefits to various applications in the context of detail enhancement, JPEG artifact removal, inverse halftoning, image segmentation, edge detection, and image stylization.

Our method improves the bilateral-texture filter proposed by Cho et al. [4] in three aspects. First, our method uses a simple interpolation algorithm to generate the guidance image without the patch shift. Second, we employ an edge-preserving-weighted-median filter for an effective structure-texture separation, while Cho et al.'s method employs the joint bilateral filter. Our filter can remove textures with extreme variations. Third, our method uses a constant time GPU-based algorithm. In summary, this paper has the following contributions:

- A new guidance image is introduced for effective texture filtering while preserving prominent edges.
- A novel structure-preserving texture filter, which effectively smoothes texture regions with extreme variations, is proposed.
- A constant time GPU-based algorithm with regard to an arbitrarily large filtering size is developed, enabling a variety of efficient texture-aware applications.

2 Related work

In recent years, many techniques have been proposed to perform effective decompositions for image structures and details. In this section, we will review some of the related works in this field.

Edge-preserving-smoothing algorithms aim at the removal of fine-scale details while preserving large-scale structures. Bilateral filter [20] is a widely used edge-preserving operator and has been applied to many applications due to its simplicity and effectiveness. The joint/cross bilateral filtering [11,16] smoothes the image by computing the range weight based on a guidance edge image. Farbman et al. [7] smoothed images using an optimization framework based on weighted least squares. Xu et al. [22] proposed L_0 gradient minimization for effective image smoothing. Subr et al. [19] performed multi-scale image decomposition by assuming details as oscillations between local minima and maxima. He et al. [9] proposed a guided image filter which has good edge-preserving smoothing properties similar to bilateral filter but does not suffer from the gradient reversal artifacts. Ma et al. [15] further utilized the guided filter to perform the edge-preserving-weighted-median filter. These algorithms depend on image intensities or gradient magnitudes to detect edges, but are not suitable for removing textures which may contain high-contrast edges.

Due to the limitations of edge-preserving-smoothing algorithms, many researchers concentrate on distinguishing textures from the main structures. Total variation regularization constraints [2,17,24] are useful for removing textures while preserving large-scale edges. Xu et al. [23] introduced a relative total variation measure and developed an efficient optimization system to further improve the separation quality of structures and textures. Zhang et al. [25] developed an acceleration algorithm for the weighted-median filter and used it as L_0-norm filter to remove textural details while preserving large-magnitude structures. However, these algorithms have difficulties in distinguishing textures near structure boundaries. Consequently, the resultant structure edges may be blended with unwanted jaggy texture boundaries or be blurred too much.

Karacan et al. [10] made use of region covariances in an adaptive-filtering framework to decompose an image into coarse-structural and fine-textural components. The similarity between two image patches is estimated with the region covariance matrices of simple image features, such as intensity, color, and orientation. However, the overlapping patches near edges will share similar local statistics and thus over-smooth structure edges.

Bao et al. [3] presented a weighted-average tree filter to achieve strong image smoothing based on a minimum spanning tree. Su et al. [18] introduced the iterative asymmetric sampling and the local linear model to produce the degenerative image to suppress the texture, and then used the edge correction operator and the joint bilateral filter to remove textures.

Cho et al. [4] introduced a simple but effective operator for structure-texture decomposition called bilateral-texture filter. The method uses patch shift that captures the texture statistics from the most representative texture patch clear of salient structure boundaries. In addition, they designed the measure of modified relative total variation to make patch shift work better. The result of patch shift is then incorporated into the range kernel of the joint bilateral filter. Liu et al. [14] demonstrated that the bilateral-texture filter can be used to preserve edges and eliminate redundant details in the non-sky regions for the physically plausible single-image dehazing. The bilateral filter tries to preserve structure edges but may introduce unwanted halos near edges because of the property of weighted sums. Moreover, the exact computation of bilateral filtering is not in constant time complexity and thus the running time for the bilateral-texture filter increases with the increase of the filtering size.

Recently, Du et al. [6] find that local Laplacian filters can be used for structure-preserving texture filtering by defining a remapping function, which is closely related to joint bilateral filtering. A variant of the joint bilateral filter is proposed in their approach to effectively produces smooth edges while preserving color variations. Although their method works well in most cases, it may fail to remove noisy dots with very high contrasts. Therefore, it is worthy for us to find a better filter to fulfill the task of texture filtering. In addition, we expect to develop a texture filter in constant time complexity to support very large filtering sizes.

Algorithm 1 Constant time texture filtering

Given: the input image I and the parameters k, ϵ

1. $B = \text{BoxFilter}(I)$
2. Compute the mRTV map T from I using Equ. 1
3. Compute the interpolation map α from T using Equ. 2
4. Compute the guidance image G using Equ. 3
5. Compute the histogram volume h using Equ. 4
6. **for** each volume slice $h(z)$ **in** h **do**
7. Smooth $h(z)$ using Equ. 5
8. Compute the median image I' using Equ. 6

return the texture-filtered image I'

3 Our approach

We summarize the pseudo-code of our approach in Algorithm 1. Our nonlinear image decomposition is based on a variant of the weighted-median filter that incorporates texture information into the guidance image. It involves the generation of guidance image and the weighted-median filtering. The intermediate results of our framework are shown in Fig. 1. In this section, we describe our approach in detail.

3.1 Generation of guidance image

The guidance image is the key to remove textures. Let I be an input image (Fig. 1a), p be a pixel in I, and N be the square local patch with the size $k \times k$. We define I_p as the input image pixel value of I at p, N_p as the square neighborhood patch centered at p, and refer an image with the subscript of p to the corresponding pixel value at p throughout this paper. In this subsection, we describe our algorithm to generate the guidance image G.

First, obtain a uniformly blurred image B (Fig. 1b) by applying the box filter to the input image I with the local patch N.

Then, as introduced in [4], a modified relative total variation map (mRTV) is used to measure the likelihood of containing structure edges and fine-scale textures for the local patch. The mRTV is computed as

$$T(N_p) = (I_{\max}(N_p) - I_{\min}(N_p)) \frac{\max_{r \epsilon N_p} |(\partial I)_r|}{\sum_{r \epsilon N_p} |(\partial I)_r| + \varepsilon}, \quad (1)$$

where $I_{\max}(N_p)$ and $I_{\min}(N_p)$ denote the maximum and minimum pixel intensities in N_p, and $|(\partial I)_r|$ represents the gradient magnitude of pixel r. The parameter $\varepsilon = 10^{-9}$ is added to avoid division by zero. From Eq. 1, we can see that the value of T is relatively large in a structure edge, relatively

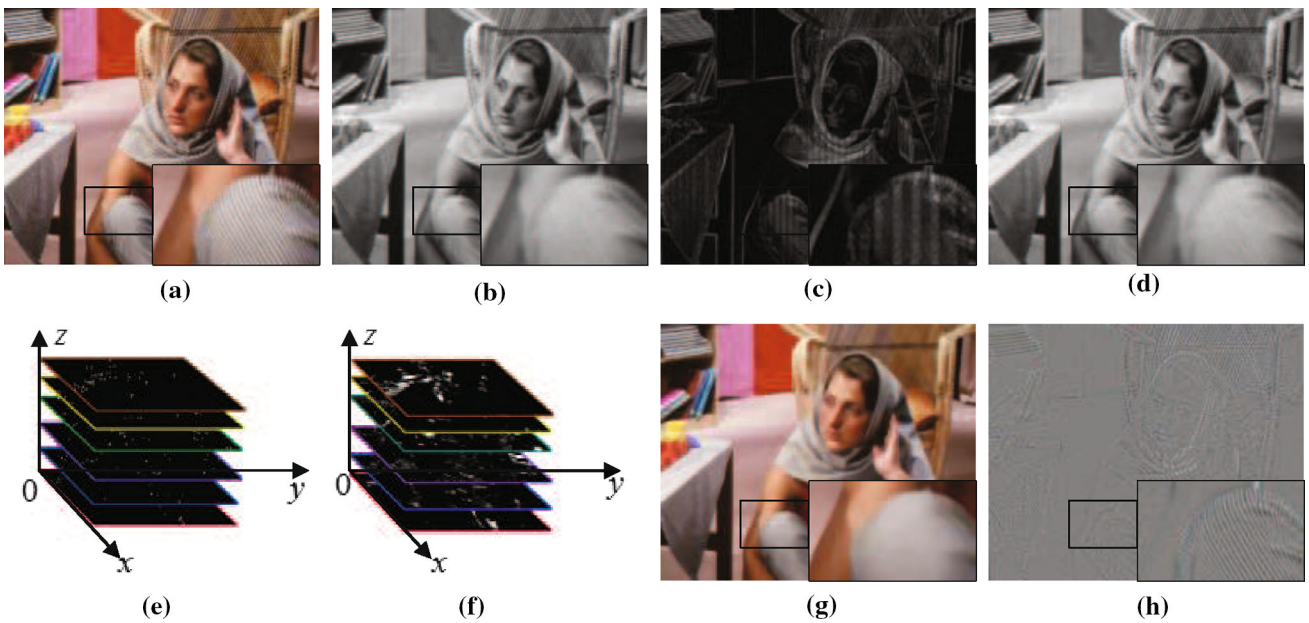


Fig. 1 Intermediate images with close-up views of our overall pipeline. **a** Input image I, **b** blurred image B, **c** mRTV, **d** guidance image G, **e** histogram volume h, **f** filtered slices, **g** result I', **h** Detail $I - I'$

small in a texture patch, and quite small in a flat region (Fig. 1c).

Finally, the guidance image is generated with a simple yet effective approach. To preserve meaningful structure edges for the texture filter, the guidance image is expected to contain structure edges as the input image. In addition, to effectively remove fine-scale textures, contrasts in textural regions of the guidance image should be intentionally lowered. With these observed properties, we generate the guidance image G with a simple linear interpolation by taking advantages of mRTV:

$$\alpha_p = \tanh(\sigma_\alpha \cdot T(N_p)), \tag{2}$$

$$G_p = \alpha_p I_p + (1 - \alpha_p) B_p, \tag{3}$$

where $\alpha_p \in [0, 1]$ is the interpolation weight, and the parameter σ_α controls the transition sharpness from edges to smooth or texture regions. In this paper, we set $\sigma_\alpha = k$. On one hand, α_p is relatively large for a pixel in a structure edge, and the value of G_p conforms to the pixel intensity of the input. On the other hand, G_p tends to use the blurred intensity for a pixel in a textural and flat region (Fig. 1d).

Note that Cho et al. [4] propose a patch shift scheme to find a nearby patch that stays clear of a prominent structure edge and generate the guidance image with the nearby blurred color. On the contrary, we use a simple interpolation without the patch shift. In the following sections, we can see that our simple interpolation results can be used to guide the filtering of textures effectively.

3.2 Weighted-median filtering

Previous approaches [4,6] apply their guidance images into joint bilateral filters for the removal of textures. Although range constraints are considered for computing weights, unwanted halos near prominent structure edges may be introduced. On the contrary, the weighted-median filter replaces the value of a pixel with the median in its neighborhood and does not introduce new pixel values. Another advantage of the median filter is that it can filter small noisy peaks with surrounding colors. Therefore, we employ the weighted-median filter rather than the bilateral filter. We adopt Ma et al.'s algorithm [15] in our framework which formulates the weighted-median filter as a histogram volume aggregation problem.

The weighted-median filter can be implemented via a three-dimensional histogram volume h which has the same width and height as the input image. For an 8-bit digital image, the possible median value only resides in a total 256 integral numbers ranging from 0 to 255. Therefore, the third range dimension of the histogram volume is 256. The values in the volume are initialized according to the following equation (Fig. 1e):

$$h(z)_p = \begin{cases} 1, & I_p = z; \\ 0, & \text{else.} \end{cases} \quad z \in [0, 255] \tag{4}$$

Then, the values of histogram bins are weighted and filtered locally with the help of the guidance image. To preserve meaningful structure edges, the values in the histogram volume are aggregated with any edge-aware-smoothing filter. In our method, we adopt the guided filter [9] which is constant time with regard to the size of filtering patch. The filter is performed on each slice $h(z)$ of h with the fixed range value z. The guided filter takes the guidance image G as the reference of structure information:

$$h'(z) = \text{GuidedFilter}(h(z), G). \tag{5}$$

The filter output is a linear transformation of the guidance image in a square window of the size ω centered at each pixel. The linear coefficients are determined by minimizing the difference between the input and the output based on an edge-preserving cost function with a regularization parameter ϵ. We set the size of ω as $2k - 1$. The regularization parameter ϵ controls the smoothness of the filtered image as the range parameter in the bilateral filter.

Finally, by accumulating the filtered histogram (Fig. 1f), the weighted-median value I'_p for each pixel p is picked. Note that the value of $h(z)_p$ is diffused to local neighboring histogram bins and many bins change their values from zero values to floating-point values after the edge-preserving-guided filtering. We show the resultant image and the corresponding detail map in Fig. 1g, h:

$$I'_p = \min r \text{ s.t. } \sum_{z=0}^{r} h'(z)_p \geq \frac{1}{2} \sum_{z=0}^{255} h'(z)_p. \tag{6}$$

3.3 Constant time GPU-based algorithm

The naïve implementation for the proposed texture filter requires the linear time complexity with regard to $k \times k$, i.e., the size of neighborhood patch N_p for each pixel p. In this subsection, we design a parallel and efficient algorithm to achieve the constant time complexity, enabling a constant time GPU-based implementation for arbitrarily large filtering sizes.

The computation of the guided filter [9] used in the weighted-median filter involves a series of box filters and local statistics which are local in nature. The box filter can be efficiently implemented using the integral image which only has constant time complexity for each pixel. In addition, the accumulation of histogram is linear regarding the intensity range. Therefore, the weighted-median filter is constant time. As for the generation of guidance image, the blurred image B is computed in constant time. However, the naïve calculation of the mRTV map (Eq. 1) requires $O(k^2)$ time

complexity. In the following, we show how mRTV can be computed in $O(1)$ time complexity.

In this paper, we formulate the calculation of local extrema as a histogram volume aggregation problem similar to the constant time scheme in weighted-median filtering. After the histogram volume is constructed according to Eq. 4, the box filter is performed on each slice $h(z)$ of the volume h. Finally, the minimum and maximum values are picked by examining the aggregated histogram:

$$(I_{\max}(N_p), I_{\min}(N_p)) = (\max z, \min z) \text{ s.t. } h(z)_p > 0.$$

The local maximum of gradient magnitude can be computed similarly with the help of the histogram volume. We employ the Manhattan distance to compute the gradient magnitude $|(\partial I)_r| = |(\partial_x I)_r| + |\partial_y I)_r|$. The possible range of $|(\partial I)_r|$ is 511 integral numbers in $[0, 510]$. Therefore, by setting the third dimension of the histogram volume as 511, we can calculate the local gradient maximum in constant time. In addition, the local summation of gradient magnitude is computed with the constant time box filter.

There are two ways to filter a color image in our implementation. One way is to first compute the pixel intensity by converting from RGB color space to CIELab color space. Then, the gray-scale guidance image is computed with the luminance channel. Finally, the weighted-median filter is applied to each color channel using the gray-scale guidance image. In this paper, most results except the one in Fig. 2 are produced this way. An alternative way is to compute the color guidance image for the weighted-median filtering. Figure 2 shows that clearer edges are produced with the color guidance image using the same parameter values.

There are two parameters in our structure-preserving texture filter, the local patch size k, and the regularization parameter ϵ. We use $k \in \{3, 5, 7, 9, 11, 13\}$ and $\epsilon \in [0.001, 0.02]$ for all the applications in this paper. To obtain a desired texture-smoothing result, more than one iterations of our filter are necessary. We denote the iteration number as n_{iter} and usually 2∼4 iterations are enough to produce a satisfying filtering result.

The whole pipeline of our approach has constant time complexity for each pixel. The local nature of the proposed algorithm enables an efficient parallel implementation on the GPU. To verify the performance of the proposed constant time texture filter, we have implemented it using NVIDIA CUDA and tested it on a PC with a 3.20 GHz Intel Core i5-3470 CPU and an NVIDIA GeForce GTX Titan X GPU running Windows 7. As shown in Fig. 3 (top), our running time for a single application of a 1-megapixel gray-scale image takes about 1.5 s, regardless of the size of filtering kernel. We can see that our novel filter still achieves an efficient performance even on an extreme large patch ($k > 1000$) without any approximation. In summary, our texture filter has

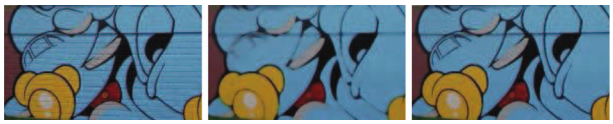


Fig. 2 Filtering difference with the same parameter values: (*left*) input, (*middle*) the result with the grayscale G, and (*right*) the result with the color G. See different *black contours* between two results

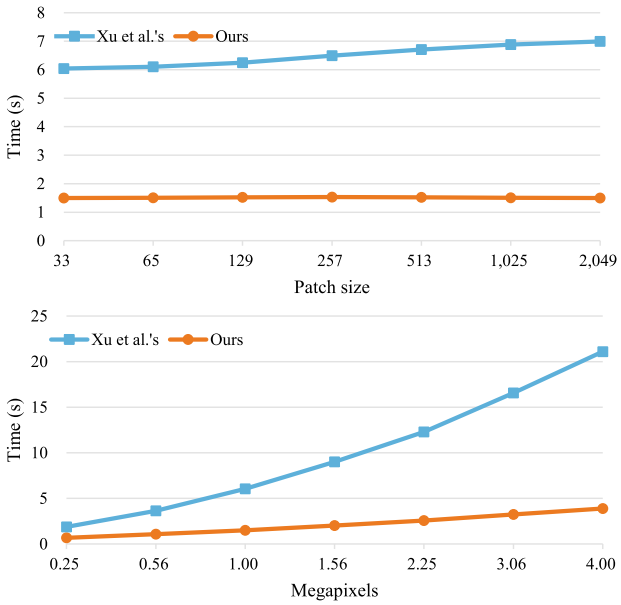


Fig. 3 Performance comparisons (*top*). The running times for a 1-megapixel gray-scale image with different patch sizes (*bottom*). The running times for different sized gray-scale images with the same patch size ($k = 33$)

constant time complexity for each pixel. Since we do not have the GPU implementation of Xu et al.'s method [23] on hand, we have only recorded the running times of their provided Matlab program in Fig. 3. The running time of Xu et al.'s method increases gradually with the increment of the patch size. We have also tested the Matlab code provided by Karacan et al. [10]. Karacan et al.'s code costs half an hour with the kernel size of 33 and requires 3.5 h when the kernel size is 65. The running time of Karacan et al.'s method increases dramatically with the increment of the filtering size. Therefore, the large time complexity of Karacan et al.'s method seriously restricts its application. We show the running times for images with different resolutions in Fig. 3 (bottom). We can see that the throughput per second of our method increases with respect to the image size. Therefore, the speedup ratio between our method and Xu et al.'s method increases gradually when the image size increases.

4 Applications and discussions

Effective structure separation from textural details enables many image-processing applications, including detail

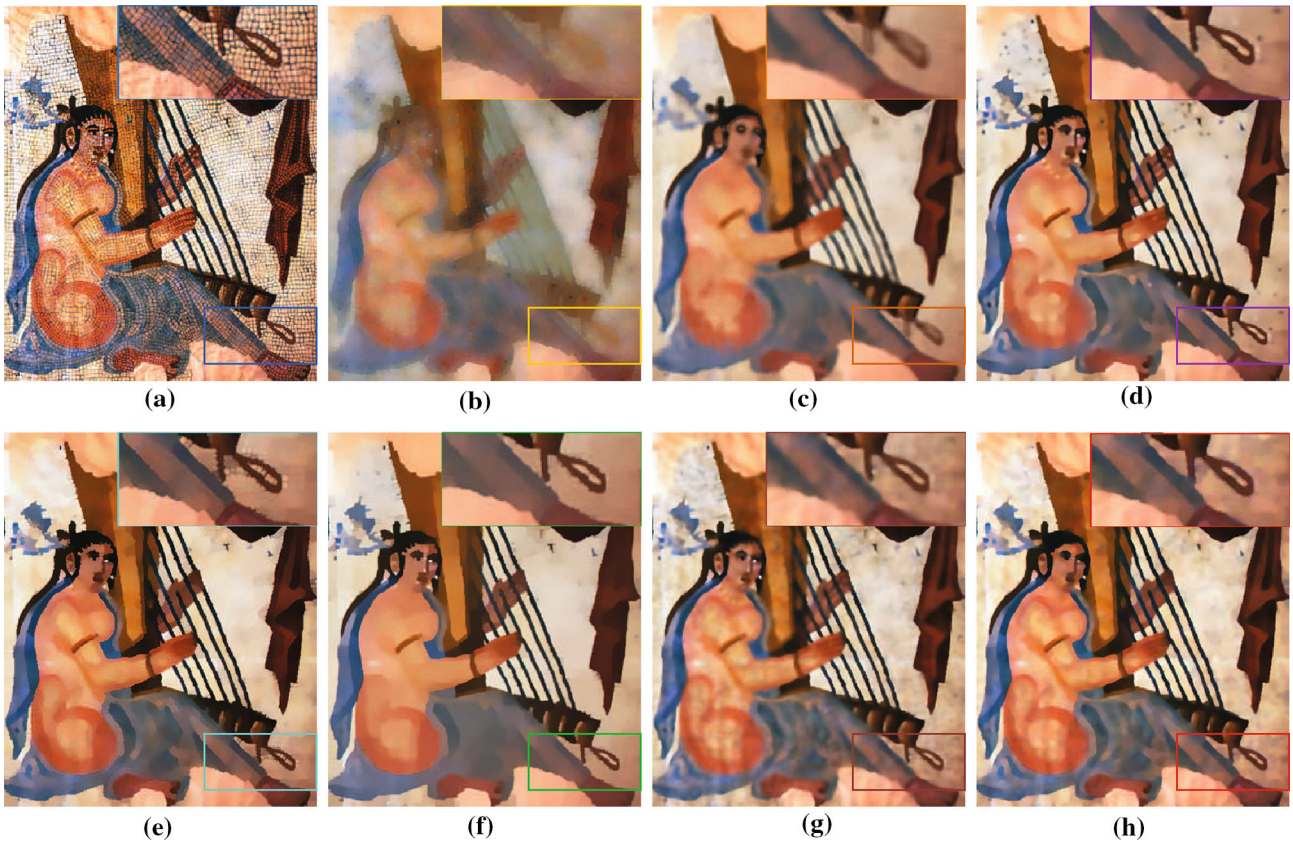


Fig. 4 Comparisons with the state-of-the-art texture-filtering methods. **a** Input, **b** Subr et al., **c** Karacan et al., **d** Du et al., **e** Zhang et al., **f** Xu et al., **g** Cho et al., **h** ours

enhancement, JPEG artifact removal, inverse halftoning, image segmentation, edge detection, and image stylization. In this section, we show various texture removal results and comparisons with the state-of-the-art algorithms. Most results are generated by the implementations provided by the authors and we have adjusted their parameters manually. The results of Cho et al.'s algorithm [4] are produced with our own implementation. Figure 4e is obtained directly from Zhang et al.'s paper [25], and Fig. 8b is downloaded from the project page of Kopf and Lischinski's paper [12].

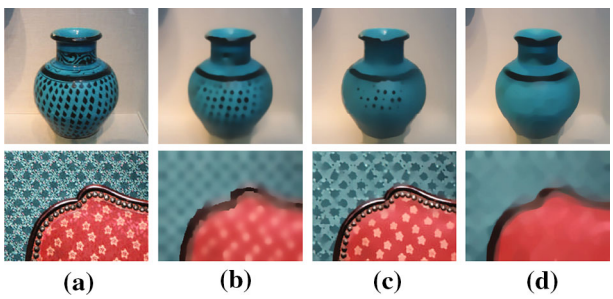


Fig. 5 Comparisons on two extreme cases (*top*). Extreme variations inside a texture region (*bottom*). Obscure borders between large-scale textures. **a** Input, **b** Cho et al., **c** Du et al., **d** ours

4.1 Texture removal

Figure 4 demonstrates the comparisons of our algorithm with the state-of-the-art texture removal algorithms. We can see that most of methods in Fig. 4 extract structures well while removing out textures. Subr et al.'s method [19] removes textures at the cost of degradation of image structures (Fig. 4b), while Karacan et al.'s method [10] may over-smooth structure edges (Fig. 4c). Du et al.'s method [6] may not work well for noisy dots with high contrasts (Fig. 4d). Both Zhang et al. [25] and Xu et al. [23] use the same relative total variation metric for removing textures and may fail to separate textures near structure edges (Fig. 4e, f). Cho et al. [4] use patch shift for effective elimination of textures near structure edges, but smooth transition between prominent structures may occur (Fig. 4g). Our method is a variant of the weighted-median-filtering method and separates structures and textures well. Moreover, clearer edges are produced with our method (Fig. 4h).

If there are extreme variations inside a texture patch (Fig. 5a, top) or obscure borders between large-scale textures (Fig. 5a, bottom), Cho et al. [4] discussed at the end of their

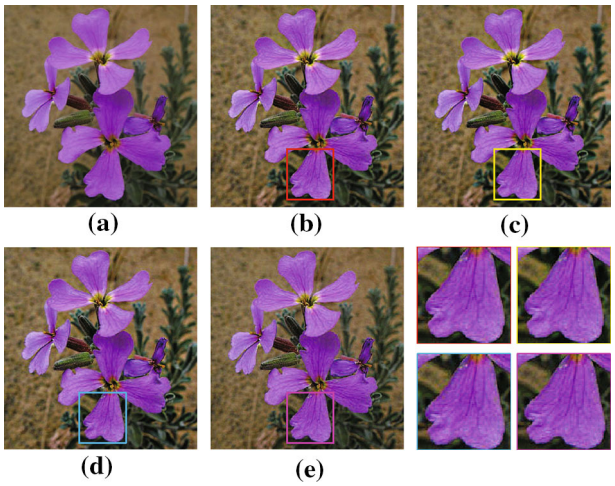


Fig. 6 Comparisons with the previous algorithms on detail enhancement. **a** Input, **b** Farbman et al., **c** Subr et al., **d** Xu et al., **e** ours

paper that their method would have trouble in smoothing out these textures (Fig. 5b). We have also experimented Du et al.'s method [6], and unfortunately, their method still fails to remove these high-contrast texture patches (Fig. 5c). With our new method, even high-contrast textures can be smoothed out in only four iterations (Fig. 5d, top). For the case in Fig. 5d bottom, large-scale textures can be well filtered, and smooth obscure borders can also be visible with our method.

4.2 Detail enhancement

Detail enhancement is very important for better illustration of image details. With our edge-preserving texture-smoothing filter, the input image is decomposed into a base structure layer and a detail texture layer. Our method can produce comparable results with the state-of-the-art techniques in boosting image details. As shown in Fig. 6, we can see that flower textures are effectively enhanced with our method.

4.3 JPEG artifact removal

The JPEG compression is commonly used in the image storage. However, the frequency quantization in the JPEG compression algorithm may introduce noisy artifacts near edges. We apply our method to address this problem and compare with other methods in Fig. 7. The results show that our method can restore the interior flatness while keeping edges well. In comparison, Ma et al. [15] cannot smooth out interior noises, and Cho et al.'s method [4] tends to blur across structure edges.

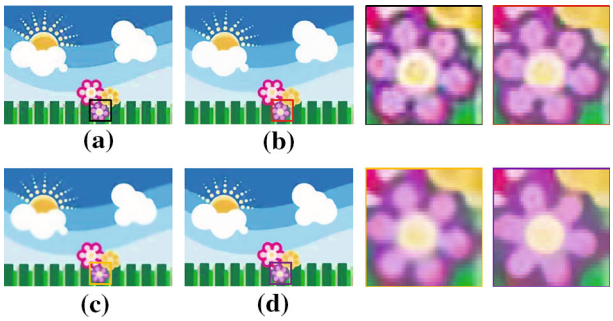


Fig. 7 Comparisons on JPEG artifact removal. **a** Input, **b** Ma et al., **c** Cho et al., **d** ours

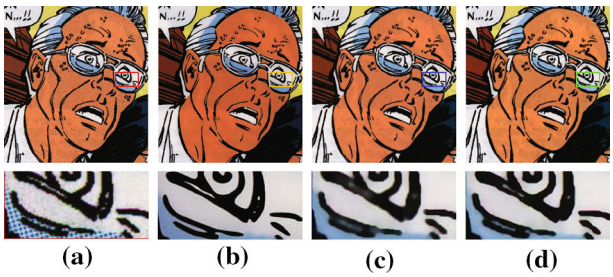


Fig. 8 Comparisons on inverse halftoning. Input image ©*Marvel Comics*. **a** Input, **b** Kopf et al., **c** Cho et al., **d** ours

4.4 Inverse halftoning

Our method can also be applied to inverse halftoning which aims to remove background dots from halftone images. Kopf et al. [12] designed a good algorithm tailored for inverse halftoning. Since our method views stipple dots as texture patches, our method can also produce satisfying results. As shown in Fig. 8, Kopf et al.'s approach gives the best result while our method and Cho et al.'s method [4] can also effectively remove the underlying stipple dots.

4.5 Image segmentation

Image segmentation is very important in image processing and multimedia applications. A popular tool is the mean-shift segmentation [5]. However, it is difficult to segment objects with textured surfaces. We use our method as a preprocessing step to remove textures and utilize the mean shift to segment objects. We show our segmentation results in Fig. 9 and compare with directly segmented results. We can see that our method segments the textured regions well, while the original method will segment many small parts.

4.6 Edge detection

Depicting visual information by edges is the simplest and oldest means to convey object shapes to the viewer. Edges usually outperform the original picture in terms of sub-

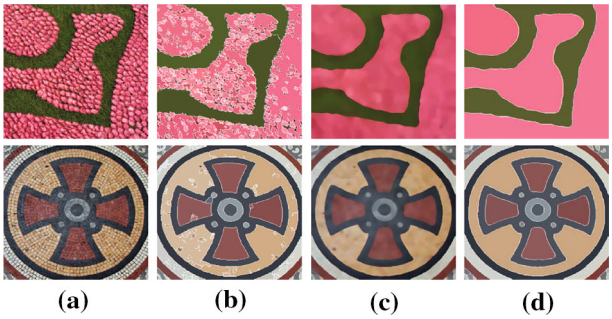


Fig. 9 Image segmentation. **a** Input images, **b** direct segmentation results, **c** our filtered images, and **d** our segmentations. Boundaries between segments are emphasized with the *white color*. The average *color* in each segmented region is displayed for a better illustration

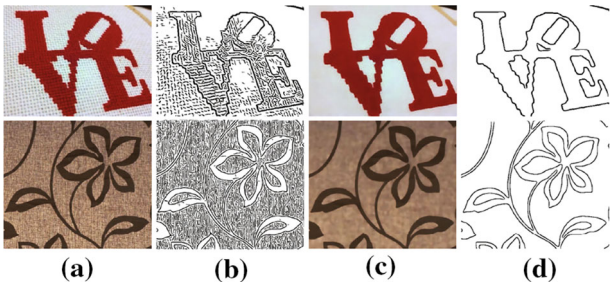


Fig. 10 Comparison on edge detection. **a** Input images, **b** direct edge detection results, **c** our filtered images, and **d** our detected edges

ject identification. However, the traditional edge detection algorithms tend to view high-contrast textures as edges. Therefore, a direct application of edge detection algorithms fails to convey object shapes for texture-rich images. We show an example of comparison on edge detection using the flow-based line drawing [13] in Fig. 10. As a result, our method can be used as an effective preprocessing step for edge detection to avoid unnecessary edges on textures.

4.7 Image stylization

Salient structural edges are emphasized while small-scale detail regions are artificially flattened in image stylization.Traditional image stylization algorithms [21,26] use edge-preserving smoothing filters to remove local-contrast regions. Here we demonstrate our method on removing unimportant textural details in image stylization and show a comparison in Fig. 11. Both images use dark edges to emphasize high-contrast regions. In addition to unnecessary fine-scale details, meaningful structures of clouds and grasses are also smoothed out because of low contrasts with Xu et al.'s method [22]. Moreover, all high-contrast details of grasses are emphasized. In comparison, only fine-scale texture details are removed with our method.

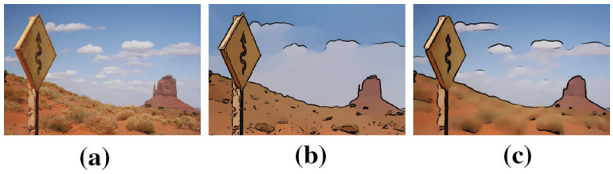


Fig. 11 Comparison on image stylization. **a** Input, **b** Xu et al., **c** ours

Table 1 Statistics of numbers of each rank among three methods

Rank	Cho et al.	Du et al.	Ours
Best	56	110	149
Average	139	96	80
Worst	120	109	86

Table 2 Comparisons of evaluation quantities on contour detection

Method	ODS	OIS	AP
gPb-owt-ucm	0.719	0.743	0.750
Ours + gPb-owt-ucm	0.720	0.751	0.772

5 Evaluation

5.1 User study

A user study was conducted to assess the qualities of our texture-filtered images. 21 volunteers were invited to participate in the user study. We provided the volunteers with 15 groups of images. Each group contains four images (an input image, the result of Cho et al. [4], the result of Du et al. [6], and the result of ours). To give a fair comparison, all the test images were obtained directly from Cho et al.'s website and Du et al., respectively, and all the volunteers did not know which image was filtered with which method. Each volunteer was asked to give an overall consideration to each group and rank the filtered images.

We recorded the numbers of each rank (best, average, or worst) among the three methods in Table 1. Our method gets 47.3 % of the total numbers which the volunteers chose as the best, whereas it gets only 27.3 % of the total numbers in which the volunteers chose as the worst. The statistics show that the volunteers prefer our method to the comparison methods.

5.2 Quantitative benchmarks

Quantitative benchmarks were performed to verify that our algorithm can be used as an effective preprocessing step for edge detection and image segmentation. We employed the Berkeley Segmentation Datasets and Benchmarks (BSD500) and the open source gPb-owt-ucm algorithm [1]. The gPb-owt-ucm code was run with default parameters given by the authors. First, we selected 62 texture-rich images among 200

Table 3 Comparisons of evaluation quantities on image segmentation

Method	ODS	OIS	AP
gPb-owt-ucm	0.726	0.759	0.726
Ours + gPb-owt-ucm	0.727	0.761	0.734

test images in BSD500. Then, we filtered the selected images using our texture filter. Finally, we collected the 62 texture-filtered images and the remaining 138 original images as our new test images for benchmarks.

It is pointed that the boundary quality is evaluated with three quantities in BSD500: the best F-measure on the dataset for a fixed scale (ODS), the aggregate F-measure on the dataset for the best scale in each image (OIS), and the average precision on the full recall range (AP). For each quantity, the higher score represents the better accuracy. Table 2 gives comparisons of evaluation quantities on edge detection for the original test images and our new test images while Table 3 shows the comparisons on image segmentation. From the tables we can see that our method can improve the quality of edge detection for texture-rich images.

6 Conclusions

Structure-preserving texture filtering is a valuable technique for a variety of applications in computer graphics and image processing. Our method is a simple yet effective improvement for the original weighted-median filter. The use of the weighted-median filter benefits to the extraction of structures from textures. The use of histogram volumes enables the proposed algorithm an efficient parallel GPU implementation with constant time complexity. Various experimental results and comparisons have been conducted to demonstrate the effectiveness of the presented method.

An image usually contains multiple scales of textures, larger patch sizes can filter larger scale textures but may smooth out meaningful structures. This limitation motivates us to develop an adaptive scheme for the removal of multi-scale textures in the future. Although our texture-filtering algorithm can be easily parallelized using CUDA, it still cannot achieve real-time performance on modern GPUs. We would like to investigate multi-GPU programming or image downsampling schemes [8] to further improve the performance of the proposed method with tolerable visual effects. In addition, we also plan to apply our method to other applications related to texture filtering.

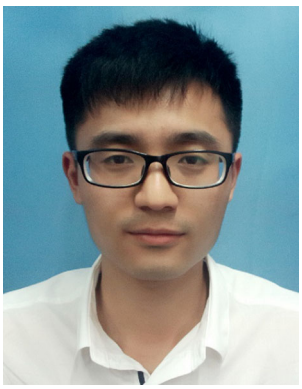
Acknowledgements We would like to thank our anonymous reviewers for their constructive suggestions and comments which definitely improve the paper. This paper was supported by the Zhejiang Provincial Natural Science Foundation of China (Grant Nos. LY15F020019 and LQ14F020006), the Science and Technology Planning Project of Wenzhou, China (Grant No. G20150019), and the Open Project of the State Key Lab of CAD&CG, Zhejiang University (Grant Nos. A1610 and A1510). X. Jin was supported by the National Natural Science Foundation of China (Grant No. 61472351). H. Du was partially supported by the Scientific Research Fund of Zhejiang Provincial Education Department, China (Grant No. Y201534269) and the Higher Education Class Teaching Reform Project of Zhejiang Province (Grant No. kg2015283). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GeForce GTX Titan X GPU used for this research.

References

1. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5), 898–916 (2011)
2. Aujol, J.F., Gilboa, G., Chan, T., Osher, S.: Structure-texture image decomposition-modeling, algorithms, and parameter selection. Int. J. Comput. Vis. **67**(1), 111–136 (2006)
3. Bao, L., Song, Y., Yang, Q., Yuan, H., Wang, G.: Tree filtering: efficient structure-preserving smoothing with a minimum spanning tree. IEEE Trans. Image Process. **23**(2), 555–569 (2014)
4. Cho, H., Lee, H., Kang, H., Lee, S.: Bilateral texture filtering. ACM Trans. Graph. 33(128), 128:1–128:8 (2014)
5. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)
6. Du, H., Jin, X., Willis, P.: Two-level joint local laplacian texture filtering. Vis.Comput. 1–12 (2015). doi:10.1007/s00371-015-1138-3
7. Farbman, Z., Fattal, R., Lischinski, D., Szeliski, R.: Edge-preserving decompositions for multi-scale tone and detail manipulation. ACM Trans. Graph. **27**(3), 67:1–67:10 (2008)
8. He, K., Sun, J.: Fast guided filter (2015). arXiv:5050.0996
9. He, K., Sun, J., Tang, X.: Guided image filtering. IEEE Trans. Pattern Anal. Mach. Intell. **35**(6), 1397–1409 (2013)
10. Karacan, L., Erdem, E., Erdem, A.: Structure-preserving image smoothing via region covariances. ACM Trans. Graph. **32**(6), 176:1–176:11 (2013)
11. Kopf, J., Cohen, M.F., Lischinski, D., Uyttendaele, M.: Joint Bilateral Upsampling. ACM Trans. Graph. **26**(3), 96:1–96:5 (2007)
12. Kopf, J., Lischinski, D.: Digital reconstruction of halftoned color comics. ACM Trans. Graph. **31**(6), 140:1–140:10 (2012)
13. Kyprianidis, J.E., Döllner, J.: Image abstraction by structure adaptive filtering. In: Proceedings of EG UK Theory and Practice of Computer Graphics, pp. 51–58 (2008)
14. Liu, C., Zhao, J., Shen, Y., Zhou, Y., Wang, X., Ouyang, Y.: Texture filtering based physically plausible image dehazing. Vis. Comput. **32**(6), 911–920 (2016)
15. Ma, Z., He, K., Wei, Y., Sun, J., Wu, E.: Constant time weighted median filtering for stereo matching and beyond. In: Proceedings of the 2013 IEEE International Conference on Computer Vision. ICCV '13, pp. 49–56. IEEE Computer Society, Washington, DC, USA (2013)
16. Petschnigg, G., Szeliski, R., Agrawala, M., Cohen, M., Hoppe, H., Toyama, K.: Digital photography with flash and no-flash image pairs. ACM Trans. Graphi. **23**(3), 664–672 (2004)
17. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Phys. D Nonlinear Phenom. **60**(1–4), 259–268 (1992)
18. Su, Z., Luo, X., Deng, Z., Liang, Y., Ji, Z.: Edge-preserving texture suppression filter based on joint filtering schemes. IEEE Trans. Multimedia **15**(3), 535–548 (2013)
19. Subr, K., Soler, C., Durand, F.: Edge-preserving multiscale image decomposition based on local extrema. ACM Trans. Graph. **28**(5), 147:1–147:9 (2009)

20. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. Proceedings of the Sixth International Conference on Computer Vision. ICCV '98, pp. 839–846. IEEE Computer Society, Washington, DC, USA (1998)
21. Winnemöller, H., Olsen, S.C., Gooch, B.: Real-time video abstraction. ACM Trans. Graph. **25**(3), 1221–1226 (2006)
22. Xu, L., Lu, C., Xu, Y., Jia, J.: Image smoothing via l0 gradient minimization. ACM Trans. Graph. **30**(6), 174:1–174:12 (2011)
23. Xu, L., Yan, Q., Xia, Y., Jia, J.: Structure extraction from texture via relative total variation. ACM Trans. Graph. **31**(6), 139:1–139:10 (2012)
24. Yin, W., Goldfarb, D., Osher, S.: Image cartoon-texture decomposition and feature selection using the total variation regularized L1 functional. In: Proceedings of the Third International Conference on Variational. Geometric, and Level Set Methods in Computer Vision, VLSM '05, pp. 73–84. Springer, Berlin (2005)
25. Zhang, Q., Xu, L., Jia, J.: 100+ times faster weighted median filter (wmf). In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. CVPR '14, pp. 2830–2837. IEEE Computer Society, Washington, DC, USA (2014)
26. Zhao, H., Jin, X., Shen, J., Mao, X., Feng, J.: Real-time feature-aware video abstraction. Vis. Comput. **24**(7), 727–734 (2008)

Hanli Zhao is an associate professor at Wenzhou University, Wenzhou. He received his B.E. degree in software engineering from Sichuan University in 2004 and Ph.D. degree in computer science from Zhejiang University in 2009. His current research interests include interactive image editing, computer animation, geometric processing, and GPU parallel computing.



Lei Jiang is a postgraduate student at Wenzhou University, Wenzhou. He received his B.E. degree in computer science from Suzhou University in 2013. His research interests include interactive image editing and GPU parallel computing.

Xiaogang Jin is a professor of the State Key Lab of CAD&CG, Zhejiang University. He received his B.Sc degree in computer science in 1989, and M.Sc and Ph.D. degrees in applied mathematics in 1992 and 1995, all from Zhejiang University. His current research interests include virtual try-on, insect swarm simulation, traffic simulation, implicit surface modeling and applications, creative modeling, sketch-based modeling, and image processing. He received an ACM Recognition of Service Award in 2015. He is a member of the IEEE and ACM.

Hui Du is currently a faculty member of the Zhejiang University of Media and Communications. He received his MSc degree from the Zhejiang University of Technology in 2004 and his Ph.D. degree from the State Key Lab of CAD&CG, Zhejiang University in 2013. His research interests include image processing and computer vision.

Xujie Li is a faculty member at Wenzhou University, Wenzhou. He received his B.E. degree in communication engineering in 2005 and M.E. degree in communications and information systems in 2008, both from Ningbo University. His research interests include computer graphics, image processing, and GPU parallel computing.