A SHAPE-AWARE STRUCTURE-PRESERVING TEXTURE SMOOTHING ALGORITHM

Bolu Liu *Xiqun Lu*
College of Computer Science and Technology, Zhejiang University, Hangzhou, 310027
xqlu@zju.edu.cn

ABSTRACT

In this paper, we introduce a shape-aware texture smoothing algorithm based on a flexible line shift scheme to separate prominent structures from textures or noises. Unlike previous texture smoothing algorithms, the line shift scheme can help to form shape-aware local area for each pixel during filtering. We integrate the line shift scheme with prior knowledge of prominent structures learned from a structured forests classifier into a framework of joint bilateral filtering. This new strategy can greatly improve the efficiency to remove textures/noise on pixels that are close to sharp corners or located in long narrow strip areas, while keep the prominent structures untouched. We demonstrate the performance of the proposed algorithm on a wide variety of images corrupted by textures and Gaussian noise.

Index Terms— line shift, texture smoothing, joint bilateral filtering, structure-preserving, learning-based edge detection

1. INTRODUCTION

Texture filtering or smoothing is to remove texture patterns from an input image without touching the prominent structures, which will produce a visual abstraction. It is useful for image editing, such as tone mapping, detail enhancement, and so on. To separate prominent structures from textures or noises depends on two factors: one is to determine an appropriate local area for local analysis, and the other one is to determine an appropriate local measure to distinguish prominent structures from textures or noises.

In contrast to Gaussian noise, texture usually affects a larger local area rather than just a single pixel [11]. In Fig.1, we show the autocorrelation functions of six Gaussian noise patches and some texture patterns, respectively. We can see that the autocorrelation function of Gaussian noise is like a delta function, while the autocorrelation functions of texture patterns are much like damped oscillation functions. These results illustrate that there are high correlation among spatial neighbors in texture patterns, and the spatial correlation mode depends on the granularity of texture pattern. Generally, an image contains prominent structures and fine scale details at multiple scales, so it is very difficult to determine an appropriate local area for local analysis, especially for those pixels located in sharp corners or in long narrow strip areas.

In this paper, we address two critical problems of texture smoothing: one is to determine an appropriately-shaped local area for each location during filtering without touching the nearby prominent structures; the other is to distinguish prominent structures from textures/noises especially when textures or noises produce comparable local contrast as those produce by the prominent structures. For the first problem, motivated by the idea of "patch shift" [1], in this paper, we propose a flexible "line shift" to find a "clean" local area for each pixel, which can be arbitrarily shaped. For the second problem, because our human vision system can effortlessly detect prominent structures from images corrupted either by textures or high-level noises. Thus, we turn to a structured learning-based edge classifier [2] to integrate the prior knowledge of prominent structures learned from human beings into texture smoothing.

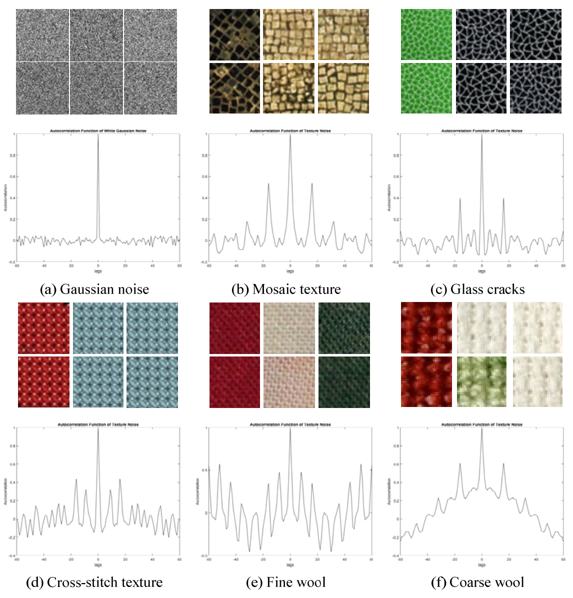

Fig.1 Autocorrelation functions of Gaussian noise and some different texture patterns, respectively.

The rest of this paper is organized as following. In Section 2, we briefly review some related texture smoothing algorithms, and analyze the problems with these algorithms, especially when they are facing with pixels located in sharp corners or long narrow strip areas. In Section 3, we introduce the shape-aware texture smoothing filtering algorithm and some implementation details. In Section 4, we extensively test the proposed algorithm on a variety of images corrupted by different textures or by high-level Gaussian noise. Finally, we draw some conclusions in Section 5.

2. RELATED WORK

Due to its simplicity and effectiveness, the bilateral filtering [12] is widely applied for structure-preserving filtering. But for those images with complex textures or degenerated by high-level Gaussian noise, the large local contrast induced by local textures or high-level noise will hinder those visually homogeneous pixels to the local filtering. Consequently, textures and noise cannot be removed efficiently. Due to the same reason, those algorithms based on local contrast have the same problem for texture smoothing [13] [14] [15].

Most structure-preserving filtering approaches [1] [3] [4] [5] use regularly shaped local area, such as local patch or local circle centered at each pixel to capture texture information. To reduce the probability of blurring prominent structure, the patch shift algorithm [1] tried to find a non-centered "clean" patch for each pixel (which contains the least of nearby prominent structure), and the average value of the "clean" patch will be used as the guidance value for the joint bilateral filtering. However, one of the bottlenecks is to determine an appropriate size for the analyzing patch. In Fig.2, we give examples to illustrate the problem with the patch shift algorithm [1] when it comes to pixels that are close to a sharp corner or located in a long narrow strip area. For the sake of clarity, we enlarge the red and green wire frames in Fig.2 (a) to (b) and (c), respectively. For those pixels that are close to a salient edge but located in a large uniform texture area, it is easy for the patch shift algorithm [1] to find a "clean" patch. But for the locations shown in Fig.2 (b) and (c), it will be difficult for the patch shift algorithm [1] to find "clean" patches for those pixels. In this example, we can see that the scale of mosaic texture is larger than the scale of long narrow strip.

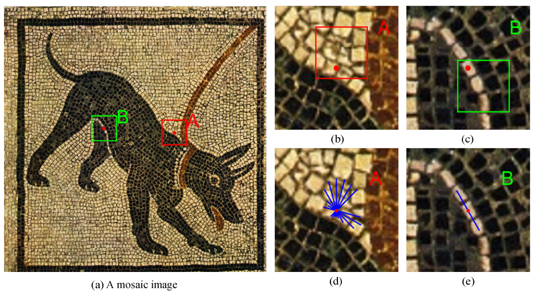

Fig.2 An example to illustrate the problem with the patch-shift [1].

3. SHAPE-AWARE TEXTURE SMOOTHING

3.1. The Idea of Line Shift

The idea of line shift is shown in Fig.3. For each pixel **p**, we can form lines with length *k* through the pixel **p** with different orientations as shown in Fig.3 (a), and these lines can shift along their own orientations radially. Along each orientation, there are totally *k* shifted line segments that will contain pixel **p**. We only show the line segments shifted along the horizontal direction in Fig.3 (b). The line segments with other orientations shift similarly along their own direction. In Fig.3 (b), we put the shifted line segments in juxtaposition for clarity. All these shifted line segments contain pixel **p** and from these shifted line segments, we will select the most "clean" line segment according to a "clean" measure (which we will explain in the next subsection). Hence, based on the line shift, these selected "clean" shifted line segments from different orientations can form a shape-aware local area for texture smoothing. Some examples of shape-aware "clean" local areas formed by the line shift are shown in Fig.1 (d) and (e).

3.2. To Select the "Clean" Line Segments

Since our human vision system can effortlessly tell prominent structures from pervasive textures or high-level noises, we turn to a structured learning-based edge classifier [2] to integrate the prior knowledge of prominent structures learned from human being into texture smoothing. In order to attach a clean measure to each line segment, we first obtain the edge confidence map of input image from the structured learning-based edge classifier [2]. This edge classifier is learned from the BSDS500 training dataset [6] that make it robustness to a wide variety of texture scales.

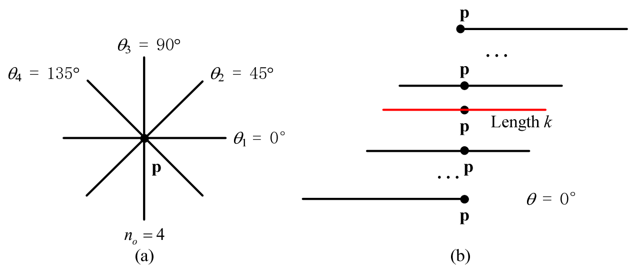

Fig.3 The idea of line shift

Then, from the *k* shifted line segments with the orientation θ, we select the line segment $L_\theta(\mathbf{p})$ containing pixel **p** that is least likely to cross nearby prominent structure:

$$CL(l_\theta(\mathbf{p})) = \mathrm{var}(ECV(l_\theta(\mathbf{p}))) \tag{1}$$

$$L_\theta(\mathbf{p}) = \arg\min_{l_\theta(\mathbf{p})}\{CL(l_\theta(\mathbf{p})) \mid l_\theta(\mathbf{p})\} \tag{2}$$

In (1), $ECV(l_\theta(\mathbf{p}))$ denotes the Edge Confidence Values of pixels on the line segment $l_\theta(\mathbf{p})$ and $CL(l_\theta(\mathbf{p}))$ represents the "clean" measure of the line segment $l_\theta(\mathbf{p})$. We use the variance of the Edge Confidence Values of pixels on a line segment as the clean measure. The smaller value $CL(l_\theta(\mathbf{p}))$ is, more "cleaner" the line segment $l_\theta(\mathbf{p})$ is. We can find the most "cleanest" line segment $L_\theta(\mathbf{p})$ from the shifted line segments of the orientation θ with the smallest *CL* value by (2).

When the "clean" line segments $L_\theta(\mathbf{p})$ from all orientations ($\theta \in (O_1 \;\cdots\; O_{n_o})$) ($n_o$ is the number of orientations we consider) are obtained, we have two strategies to choose qualified line segments to compute the guidance value for each pixel: one is based on a hard-threshold, and the other is based on a soft-threshold. In the hard-threshold strategy, we set a threshold τ. If the "clean" measure of $L_\theta(\mathbf{p})$ is larger than τ, then the line segment $L_\theta(\mathbf{p})$ will not be considered in the further computation. However, for some pixels near prominent structures, it will be highly probable that there is no qualified line segment to satisfy the condition if the threshold τ is not set appropriately. Therefore, in this paper, we turn to a soft-threshold strategy. We will not classify the selected "clean" line segments into qualified or not, but to take all the "clean" line segments from all orientations into consideration.

3.3. The Guidance Image

These selected "clean" shifted line segments from different orientations can help to form an arbitrarily shaped local "clean" area for each pixel. We combine the average intensities of these selected "clean" line segments linearly, and the weight of each "clean" line segment is hinged on its "cleanness":

$$g'_\theta(\mathbf{p}) = \frac{1}{k}\sum_{\mathbf{q}\in L_\theta(\mathbf{p})} I_l(\mathbf{q}) \tag{3}$$

$$w_\theta(\mathbf{p}) = \exp(-\sigma CL(L_\theta(\mathbf{p}))) \tag{4}$$

$$g(\mathbf{p}) = \frac{1}{W(\mathbf{p})}\sum_{\theta\in(O_1,\cdots O_{n_o})} w_\theta(\mathbf{p})g'_\theta(\mathbf{p}) \tag{5}$$

where $g'_\theta(\mathbf{p})$ is the average intensity of the pixels on the "clean" line segment $L_\theta(\mathbf{p})$ of the orientation θ, **q** denote the pixels on the "clean" line segment $L_\theta(\mathbf{p})$, and $I_I(\mathbf{q})$ is the intensity value at pixel $\mathbf{q}$. $w_\theta(\mathbf{p})$ in (4) is the assigned weight to the line segment $L_\theta(\mathbf{p})$ according to its "clean" measure CL ($L_\theta(\mathbf{p})$), and the parameter σ controls the smoothness. The smaller σ is, the sharper the output will be. In the implementation, we choose the smoothing parameter σ as $10k$ (where k is the length of line segments). $W(\mathbf{p}) = \Sigma_\theta w_\theta(\mathbf{p})$ in (5) is the normalization factor.

For a pixel located in a pure texture area, a uniformly blurred version from its local area is good enough for its guidance value, where a pixel that is close to prominent structure, then the shape-aware guidance value from (5) is better for its guidance value. To achieve an adaptive filtering, the final guidance image is obtained by linearly combining the guidance image obtained from Eq. (5) with a uniformly blurred version b($\mathbf{p}$) of the input intensity by a spatially adaptive weighting map $\alpha(\mathbf{p})$:

$$G(\mathbf{p}) = (1-\alpha(\mathbf{p}))g(\mathbf{p}) + \alpha(\mathbf{p})b(\mathbf{p}) \tag{6}$$

where $\alpha(\mathbf{p}) = W(\mathbf{p}) / \max_{\mathbf{p}} W(\mathbf{p})$. If a pixel $\mathbf{p}$ is located in pure texture area, then its weights $w_\theta(\mathbf{p})$ along the n_o orientations are all very large, and $\alpha(\mathbf{p})$ will be close to 1. Whereas, if a pixel $\mathbf{p}$ is near a corner or an edge, then among its weights $w_\theta(\mathbf{p})$s, some of them will be large, while others are very small, so $\alpha(\mathbf{p})$ will be correspondingly smaller, and it is highly probable to use the shape-aware average intensity (5) as the guidance value.

3.4. The Implementation Details

To mitigate the impact of small-scale texture and noise on the output of the edge classifier [2], we pre-filter input image with a small sized box filter (3×3) before estimating the edge confidence map from the input image. Then, we blur the intensity component of input image uniformly by using a box filter of size k to get the uniformly blurred version $b(\mathbf{p})$ used in (6), and k is the length of line segments used in the line shift. We set $k \in \{3, 5, 7, 9\}$ in the experiment. If the texture pattern in an input image is strong, we run the algorithm for several times recursively. But for most of images, to run the algorithm just once is enough to get desired outputs. So, the default number of iterations n_{iter} is set as 1.

To speed up the filtering, instead of forming line segments along all the n_o orientations (n_o = 6 in our implementation) at each pixel, we pre-rotate both the input image and the Edge Confidence Map of the input image according to the $n_o/2$ different angles, respectively. Then we compute the "clean" measures of the k shifted line segments of each pixel along horizontal and vertical directions in the rotated Edge Confidence Map, respectively. This will greatly improve the efficiency of our algorithm.

4. EXPERIMENTAL RESULTS

Our algorithm can form arbitrarily shaped "clean" local area for each pixel, especially for those pixels located in corners or narrow long strip areas. In this section, we first evaluate the performance of our algorithm on a wide variety of textured images. We compare our results qualitatively with those produced by some state-of-the-art structure-preserving filtering algorithms: Region Covariance (RegCov) [3], Bilateral Texture Filtering (BTF) [1], directional Relative Total Variation (dRTV) [4], Relative Total Variation (RTV) [7], Semantic Filtering (SF) [8], Rolling Guidance Filter (RGF) [9]. For fair comparison, we run the codes of these algorithms provided by the authors, and set the parameters according to the suggestions in the corresponding papers. In order to compare the seven structure-preserving filtering algorithms quantitatively, we conduct a denoising experiment in which the original images have been provided.

4.1. Texture Smoothing

We test the proposed texture smoothing algorithm and the 6 algorithms on the dataset [7], which consists of over 200 texture images. Due to the space limitation, we only present three examples in Fig.5, Fig.6 and Fig.7, respectively. For the mosaic "dog" picture shown in Fig.5, among the 7 algorithms, the long narrow strip area between the left leg and the body of the dog is well preserved by our algorithm. Because the RegCov algorithm [3] use the visual features based on region covariance matrix, it will introduce some "grid-like" artifacts into the uniform area of the filtered output. Due to the reason we explained in Section 2, the BTF algorithm [1] cannot deal with long narrow strip area well. The RTV algorithm [7] blurs the fine structures of this image. The SF [8] cannot filter out textures near prominent structures, although we apply a median filter on the filtered result as recommended by the author. The filtered result most close to ours is produced by the dRTV [4], but our result can keep the fine structures better than those smoothed by the dRTV [4], such as the ears, the eye and the paws of the dog, not to mention the long narrow strip area between the leg and the body. In order to show the details clearly, we put two enlarged patches below each image, respectively.

The input image presented in Fig.6 was taken in a slant view, so the same texture pattern is with varying scales from near to far. Our method can keep the detail structure of horsehair and flower quite well. In Fig.7, we give an image corrupted by different textures. It has glass cracks on the main central part, and white dots on blue triangles decorated on the four borders. Black spots with different sizes are scattered on the flower. The glass cracks on the background and the glass cracks on the left green leaf have different granularities. From the filtered outputs, our algorithm can not only remove all these different types of textures, but also leave the white thin strips on the flower untouched. While the other algorithms either remove the white thin strips totally or blur them, and blur the decorated blue triangles, and the sharp corner of the right green leaf to some degree.

The Co-occurrence Filter (CoF) [16] is based on the bilateral filter, but instead of using a Gaussian on the range values to preserve edges it relies on a co-occurrence matrix. Pixel values that co-occur frequently in the image (i.e., inside homogeneous texture regions) will have a high weight in the co-occurrence matrix. On the other hand, pixel values that rarely co-occur (i.e., across different texture boundaries) will have a low weight in the co-occurrence matrix. The CoF filter may be appropriate for image consisted of different texture patterns, but when an image is with the same texture pattern, such as the image shown in Fig.8, it is difficult for the Cof filter to distinguish structure edges from texture edges.

4.2. Quantitative Evaluation

Since the dataset [7] didn't provide the groundtruth, we conduct a denoising experiment to illustrate the abilities of these structure-preserving filtering algorithms for denoising. We use the 200 test images in the BSDS500 dataset [6]. We add Gaussian noise with 7 high standard deviations: from 20 to 50 with an incremental step of 5 to each of these images, respectively. We apply the 7 algorithms on these 1400 noisy images, individually, and compute the PSNRs and the SSIMs [10] respectively. Fig.9 presents the mean PSNR curves and the mean SSIM curves of the 7 algorithms under different standard deviations of Gaussian noises, respectively. The ability of denoising of our algorithm is obviously much better than those of the other algorithms.

4.3. Analysis of the Parameter *k* (length of line segments)

In the BTF (patch shift) algorithm [1], the patch size *k* determines the scale under which texture or noise will be filtered. In Fig.10, we show the filtered results and the running times of our algorithm and the BTF algorithm [1] under different *k*s for an image. We can see that the running time of the BTF (the patch shift) algorithm [1] is increased exponentially when linearly increase the patch size *k*, while the running time of our algorithm is increased linearly with the line length *k*. With the increasing of the line length *k*, only the boundaries of the circles in the image are becoming coarser. This is because we use longer lines to approximate circle boundaries. However, for the BTF (patch shift) algorithm [1], with the patch size *k* increases, more and more details are blurred. Our algorithm is less sensitive to the length *k* of line segments as the BTF (patch shift) algorithm [1] does.

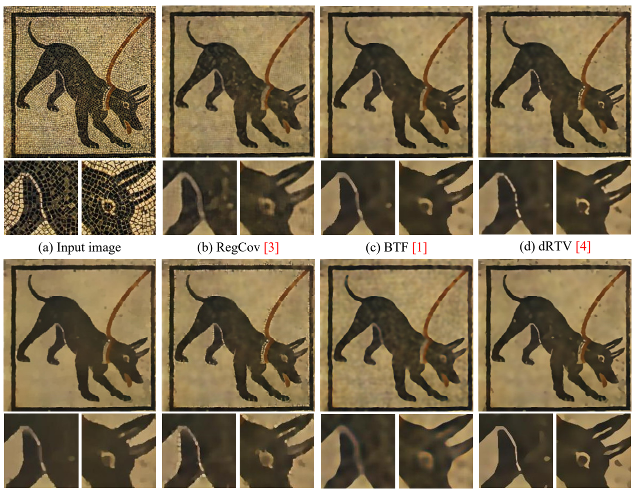

Fig.5 Texture smoothing results for a mosaic image.

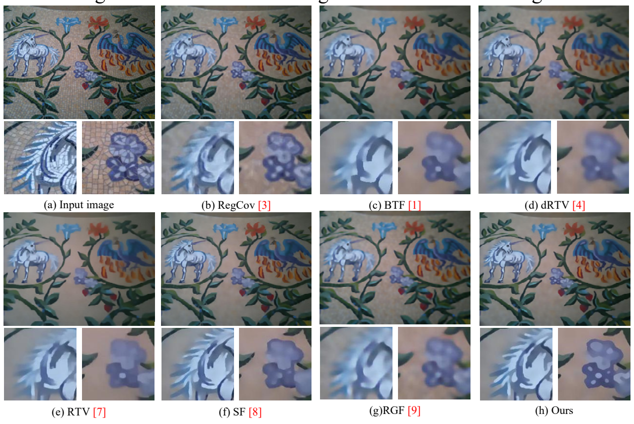

Fig.6 Texture smoothing results for a slant-viewed mosaic image

5. CONCLUSIONS

In this paper, we propose a flexible line shift scheme that can help constructing shape-aware "clean" local area for each pixel. Meanwhile, we use the priori knowledge of prominent structures learned by structured forests to improve the ability to tell the prominent structures from textures/noises. Then, we integrate the two new schemes into a framework of the joint bilateral filtering. Compared with some state-of-the-art structure-preserving filtering algorithms, the proposed method can preserve prominent structures well after filtering, especially for those pixels around sharp corners or located in long narrow strips.

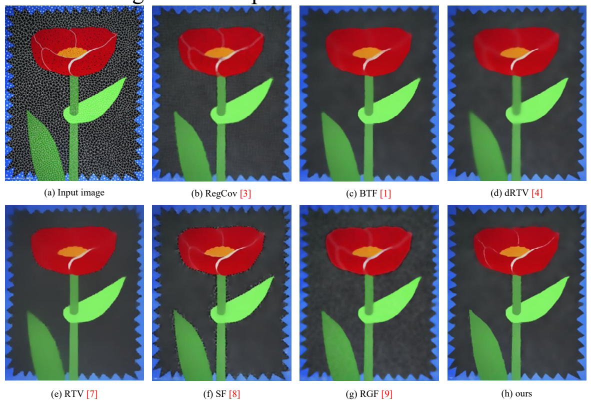

Fig.7 Texture smoothing results for an image with different texture patterns

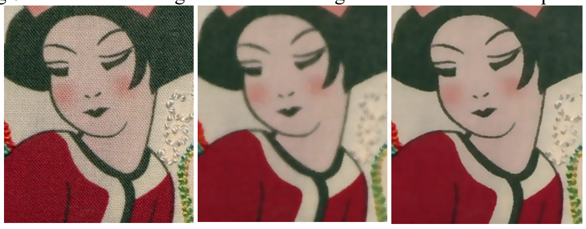

Fig.8 Comparison with the CoF filter [16] when an image is with homogeneous texture pattern (from left to right: the input image, the results of the CoF and our result).

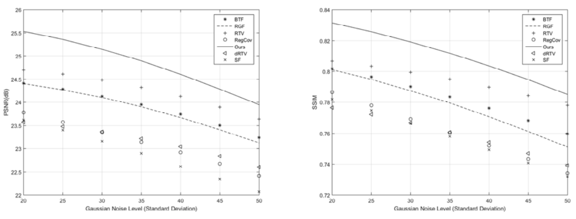

Fig.9 The denoising results of the 7 algorithms under different levels of Gaussian noise.

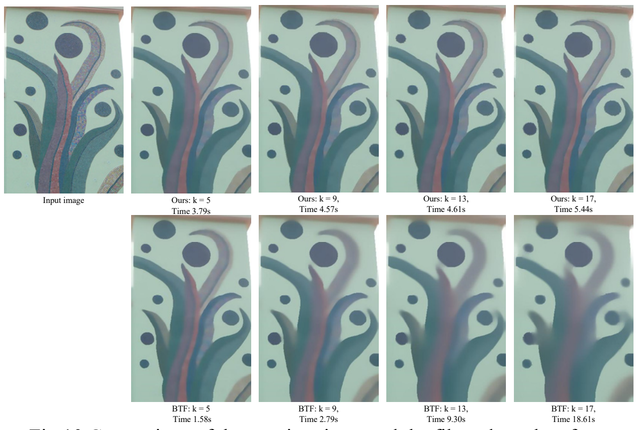

Fig.10 Comparison of the running times and the filtered results of our algorithm with those of the BTF [1] (in seconds) under different *k*s.

6. REFERENCES

[1] H. Cho, H. Lee, H. Kang, S. Lee, " Bilateral texture filtering," ACM Transactions on Graphics, 33(4):1-8,2014.

[2] P. Dollar and C.L. Zitnick, "Fast edge detection using structured forests," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.37, no.8, pp.1558-1570, 2015.

[3] L. Karacan, E. Erdem and A. Erdem, "Structure-preserving image smoothing via region covariances," ACM Transactions on Graphics, 32(6):1-11, 2013.

[4] J. Jeon, H. Lee, H. Kang, S. Lee, "Scale-aware structure-preserving texture filtering," Computer Graphics Forum, 35(7):77-86, 2016.

[5] P. Xu, and W. Wang, "Improved bilateral texture filtering with edge-aware measurement," IEEE Trans. On Image Processing, vol.27, no.7, pp.3621-3630, 2018.

[6] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," IEEE Trans. On Pattern Analysis and Machine Intelligence, vol.33, no.5, pp.898-916, 2011.

[7] L. Xu, Q. Yan, Y. Xia and J. Jia, "Structure extraction from texture via relative total variation," ACM Transactions on Graphics, 31(6), 139:1-10, 2012.

[8] Q. Yang, "Semantic filtering," in proc. of IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[9] Q. Zhang, X.Y. Shen, L. Xu, J. Jia, "Rolling guidance filter," in Proc. of European Conference on Computer Vision (ECCV), 2014.

[10] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," IEEE Trans. On Image Processing, vol.13 no.4, pp.600-612, Apr. 2004.

[11] X. Lu and H. Sakaino, "A spatial adaptive filter for smoothing of non-Gaussian texture noise," in Proc. Of ICASSP, 2009.

[12] C. Tomasi, R. Manduchi, "Bilateral filtering for gray and color images," in Proc. of ICCV, 1998.

[13] Z. Farbman, R. Fattal, D. Lischinski and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," ACM Transactions on Graphics, 27(3):67, 2008.

[14] L. Xu, C. Lu, Y. Xu and J. Jia, "Image smoothing via L0 gradient minimization," ACM Transactions on Graphics, 30(6)174:1-12, 2011.

[15] M. Kass and J. Solomon, "Smoothed local histogram filters," ACM Trans. Graph, 29(4), 2010.

[16] R. Jevnisek and A. Avidan, "Co-occurrence filter" in Proc. of CVPR, 2017.