Pacific Graphics 2016
E. Grinspun, B. Bickel, and Y. Dobashi
(Guest Editors)

Volume 35 (2016), Number 7

Scale-aware Structure-Preserving Texture Filtering

Junho Jeon[1], Hyunjoon Lee[1], Henry Kang[2], and Seungyong Lee[1]

[1]Dept. of Computer Science and Engineering, POSTECH, Pohang, Korea
[2]Dept. of Mathematics and Computer Science, University of Missouri - St. Louis, USA

Figure 1: *Scale-aware texture filtering. Compared with previous methods, ours effectively filters out multiple-scale textures from the input image, while preserving structure edges and small-scale salient features, such as corners, without oversharpening and overblurring artifacts.*

Abstract

This paper presents a novel method to enhance the performance of structure-preserving image and texture filtering. With conventional edge-aware filters, it is often challenging to handle images of high complexity where features of multiple scales coexist. In particular, it is not always easy to find the right balance between removing unimportant details and protecting important features when they come in multiple sizes, shapes, and contrasts. Unlike previous approaches, we address this issue from the perspective of adaptive kernel scales. Relying on patch-based statistics, our method identifies texture from structure and also finds an optimal per-pixel smoothing scale. We show that the proposed mechanism helps achieve enhanced image/texture filtering performance in terms of protecting the prominent geometric structures in the image, such as edges and corners, and keeping them sharp even after significant smoothing of the original signal.

Categories and Subject Descriptors (according to ACM CCS): I.4.3 [Image Processing and Computer Vision]: Enhancement—Smoothing

1. Introduction

Noise or texture filtering often involves smoothing of signal. Linear translation-invariant filters like Gaussian, for instance, aggressively smooth out data noise or small-scale oscillations by averaging the signal values around each data point at the expense of losing important structural information such as edges. On the other hand, edge-preserving filters [TM98,FFLS08,HST10,PHK11,XLXJ11], which have now become mainstream in visual processing, are designed to protect edges while removing noise. They take into account the structure of the signal and give different smoothing weights across signal discontinuities in order to avoid blurring them out.

One of the deficiencies of the classical edge-preserving filters is that they often rely on gradient-based measure to detect edges, and thus may fail to remove small-scale, high-contrast oscillating details (e.g., texture) that make little contribution to the image semantics. Recently, more sophisticated approaches have emerged that are capable of identifying and filtering such textures while still preserving important structures [KEE13,CLKL14,ZSXJ14]. Although these techniques generally deliver successful structure-texture separation, it is still a challenge on certain images to keep structure edges sharp while aggressively taking out high-contrast and large-scale texture patterns. Moreover, existing methods have trouble in

preserving the corners which are small-scale but semantically important visual features in the image.

In this paper, we present a novel filtering method for structure-texture separation based on adaptive scales of filter kernels. The central idea is to use pixel-neighborhood statistics to distinguish texture from structure and simultaneously find an optimal smoothing scale for each pixel. We show that unlike existing techniques, our method manages to achieve multiple conflicting goals, such as identifying/removing texture, preserving structure edges, protecting easy-to-miss features such as corners, and preventing oversharpening and/or overblurring artifacts (see Fig. 1).

2. Related Work

Edge-aware smoothing filters Anisotropic diffusion [PM90] averages neighboring pixels in an edge-aware manner by iteratively solving a diffusion equation based on local variations of pixel values. Bilateral filter [TM98] combines two Gaussian kernels, spatial and range, to average neighboring pixels without crossing edges. Guided filter [HST10] reduces gradient reversal near edges by using a local linear transformation kernel. Local Laplacian filter [PHK11] avoids halo artifacts by communicating information across multiple scales within Laplacian pyramid. The domain transform method [GO11] accelerates 2D edge-aware filtering by solving the problem in 1D space. One weakness of this line of filters is that they are not particularly well-equipped to filter out high-contrast detail or texture.

Kernel-based texture filters Recently, more sophisticated kernel-based filters have been proposed to perform structure-texture separation. Mode filter [KS10] and median filter [Wei06, MHW*13, ZXJ14] are capable of filtering out certain textures from images. To identify and remove texture components from an image, Karacan et al. [KEE13] use a patch-based covariance measure and Bao et al. [BSY*14] utilize a minimum spanning tree extracted from the image. Zhang et al. [ZSXJ14] employ bilateral filter guided by Gaussian-blurred input image to eliminate all image structures as well as textures smaller than a certain scale. Cho et al. [CLKL14] incorporate patch shift mechanism into conventional bilateral filter to perform structure-preserving texture filtering. Yang [Yan16] utilizes a semantic edge detector for iterative edge-preserving texture filtering. Many of these filters use a fixed smoothing scale on the entire image, which may result in overblurring or oversharpening when features of multiple scales coexist.

Global methods Some image/texture filters are driven by global minimization of certain objective functions. Farbman et al. [FFLS08] proposed weighted least squares (WLS) method that solves a large linear system to perform multiscale image decomposition. Subr et al. [SSD09] considered local extrema to distinguish fine-scale oscillations from real edges. L_0 gradient minimization [XLXJ11] restricts the number of non-zero gradients in the image for enhanced edge-aware filtering quality. Xu et al. [XYXJ12] minimize the relative total variation (RTV) measure of image for clearer separation of structure and texture. Ham et al. [HCP15] incorporate a Gaussian-blurred guidance image into their nonconvex optimization framework, and Bi et al. [BHY15] use L1 local flatness and global sparsity for image smoothing. Compared to kernel-based filtering, optimization-based techniques are harder to implement and accelerate.

3. Our Approach

Some of the recent texture filtering methods [CLKL14, ZSXJ14, HCP15] rely on intermediate guidance images to remove textures from images. The guidance image is typically generated via edge-preserving smoothing of the source image, and then used for a joint bilateral filtering to filter out texture. Given an input image I, the joint bilateral filter is defined as follows:

$$S_p = \frac{1}{k} \sum_{q \in N(p)} f(p,q)\, g(G_p, G_q)\, I_q, \tag{1}$$

where I_p, G_p, and S_p represent intensity/color values at the pixel p of the input, guidance, and filtered images, respectively, and $p = (p_x, p_y)$, $q = (q_x, q_y)$ are spatial coordinates of two pixels. $f(\cdot)$ and $g(\cdot)$ measure spatial and range distances between two pixels, for which Gaussian weighting functions are used [TM98, PSA*04]. The filtering process could be iterated depending on filtering objectives.

For the guidance-based structure-texture separation framework, the guidance image is crucial in determining the quality of filtering results. The properties of a good guidance image for structure-texture separation are: 1) being smooth within textured regions so the texture would be filtered out effectively; 2) showing clear discontinuities at structure edges and visually important features, however small they may be. In [ZSXJ14], the guidance image is first generated by Gaussian smoothing of the source image with certain scale, then updated via successive joint bilateral filtering operations. This method loses small-scale structures such as corners due to the initial Gaussian blurring. Cho et al. [CLKL14] proposed patch shift operation that preserves structure edges by finding alternative patches for pixels near the structure edges, but in a complex region containing multiple structures, confusion may arise during searching alternative patches and end up blurring the structures.

These artifacts are, in a sense, due to the fixed scale parameter for the smoothing method that they use when generating the guidance image. Ideally, the source image would have to be smoothed more aggressively within textured regions and less around important edges and features in order not to blur them. In this paper, we address this issue by adaptively controlling the smoothing scale for guidance image generation. While adaptive scaling in image filtering is not new [EZ98, CT05, KFEA10], to the best of our knowledge, our method is the first filtering-based approach to apply adaptive scale to the problem of texture-structure separation. In the following sections, we describe a guidance image based texture-structure separation method with the collective flatness, a novel kernel scale measure based on directional relative total variation.

4. Adaptive Kernel Scale Estimation

Similarly to [ZSXJ14], we obtain the guidance image by filtering the source image. The difference is that we perform a Gaussian

filtering with adaptive kernel scale:

$$G_p = \sum_{q \in N(p)} g_{K_p}(p,q) I_q,$$

$$g_{K_p}(p,q) = \frac{1}{\sqrt{2\pi}K_p} \exp\left(-\frac{\|p-q\|^2}{2K_p^2}\right),$$

where K_p is an optimal kernel scale at a pixel p that is estimated as described in the following subsections.

4.1. Relative total variation (RTV)

To estimate K_p at each pixel, we need to measure how close the pixel is to the nearby structure edge that we want to preserve. The underlying principle is to enlarge K_p within a flat or textured region, and shrink K_p near structure edges and corners. In order to distinguish fine-scale oscillations from structure edges, first we employ relative total variation (RTV) metric [XYXJ12] for its robustness and discriminability. Given an image I, RTV along the x-axis is defined as follows:

$$\mathrm{RTV}_p^x = \frac{\sum_{q \in N(p)} g_\sigma(p,q) \; |(\partial_x I)_q|}{\left|\sum_{q \in N(p)} g_\sigma(p,q) \; (\partial_x I)_q\right| + \varepsilon}, \tag{2}$$

where $\partial(\cdot)$ is the discrete gradient operator and $g_\sigma(\cdot)$ is a Gaussian function of variance σ^2. ε is a small positive number. $N(p)$ is the spatial neighborhood of pixel p with the size of $3\sigma \times 3\sigma$. RTV_p^y is defined similarly along the y-axis. Simply put, RTV increases in a region of small-scale oscillations and decreases in a region containing a real edge. In that sense, RTV could be used for determining the kernel scale K_p at pixel p.

4.2. Directional RTV

One limitation of original RTV in Eq. (2) is that it often mistakes slanted sharp features such as corners for texture regions because they are considered as oscillations along both x- and y-axes. If we directly use RTV for kernel scale K_p, this mistake would result in overblurring of the fine-scale features due to the large-scale kernels assigned to them. To overcome this limitation, we define *directional relative total variation (dRTV)*:

$$\mathrm{dRTV}_p^\phi = \frac{\sum_{q \in N(p)} g_\sigma(p,q) \; |(\partial_\phi I)_q|}{\left|\sum_{q \in N(p)} g_\sigma(p,q) \; (\partial_\phi I)_q\right| + \varepsilon}, \tag{3}$$

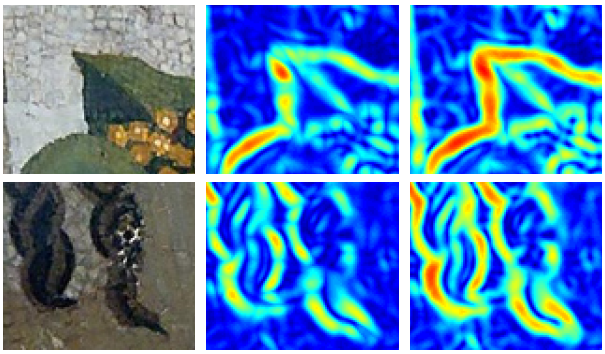
where $\partial_\phi(\cdot)$ is a directional partial derivative operator along angle ϕ:

$$\partial_\phi(\cdot) = \partial_x(\cdot)\cos\phi + \partial_y(\cdot)\sin\phi. \tag{4}$$

We find θ_p, which we call *structure direction*, as the direction that has the smallest oscillation. Usually such θ_p is the direction perpendicular to the nearby structure edge. At a corner, θ_p is the direction in which the corner's tip is pointing. In a texture region, θ_p can be an arbitrary direction. Since small-scale oscillation introduces high dRTV, the structure direction θ_p is obtained as:

$$\theta_p = \arg\min_\phi \; \mathrm{dRTV}_p^\phi \tag{5}$$

for $0 \le \phi < 2\pi$. In our implementation, we sampled and checked 12 different directions to efficiently identify structure direction θ_p.



input original RTV dRTV

Figure 2: *dRTV outperforms the original RTV in capturing salient features. Red color indicates small RTV value (considered as edge), while blue color indicates large RTV value (considered as texture).*

Using structure direction, dRTV can identify the structure edges as well as sharp corners which are easily missed by original axis-aligned RTV. Fig. 2 illustrates that dRTV outperforms the original RTV in capturing salient features, including slanted corners.

4.3. Flatness

Although dRTV in Eq. (3) can effectively distinguish even small structures from textures, it has limitation in accurately determining the kernel scale K_p. That is, a large value of dRTV_p^θ implies that pixel p belongs to a texture region but does not exactly mean that p is far away from a structure edge, as dRTV value is affected by the magnitudes of oscillations as well as the number of oscillations in a region. To resolve this limitation, we normalize dRTV values using a nonlinear transform and compute E_p^θ, which we call *flatness*:

$$E_p^\theta = \exp\left\{-\left(\frac{(1/\mathrm{dRTV}_p^\theta)^2}{2\sigma_e^2}\right)\right\}. \tag{6}$$

σ_e controls the flatness transition from structure edges to texture regions, which is fixed as 0.05 for sharp transition. The flatness value ranges in $[0,1]$, where 0 means a structure edge.

4.4. Collective flatness for kernel scale estimation

Fig. 3 shows that the flatness is high within flat or textured regions, and low around edges. Therefore, we should be able to define kernel scale as proportional to this measure. However, in Fig. 3b, flatness drastically reduces to zero at the pixels around an edge, which may result in overly conservative filtering near structure edges. Moreover, as shown in the right column of Fig. 3, the flatness inside a single stripe structure with textures becomes large because oscillations exist in any direction. This large flatness would introduce undesirable filtering artifacts as the kernel scales of pixels inside the stripe are larger than the strip width (Fig. 5). That is, although the flatness can distinguish texture regions from structures well, including edges and corners, it is not enough to determine an optimal kernel scale which leads exact amount of blurring.

To mitigate this problem, we gather information from a collec-

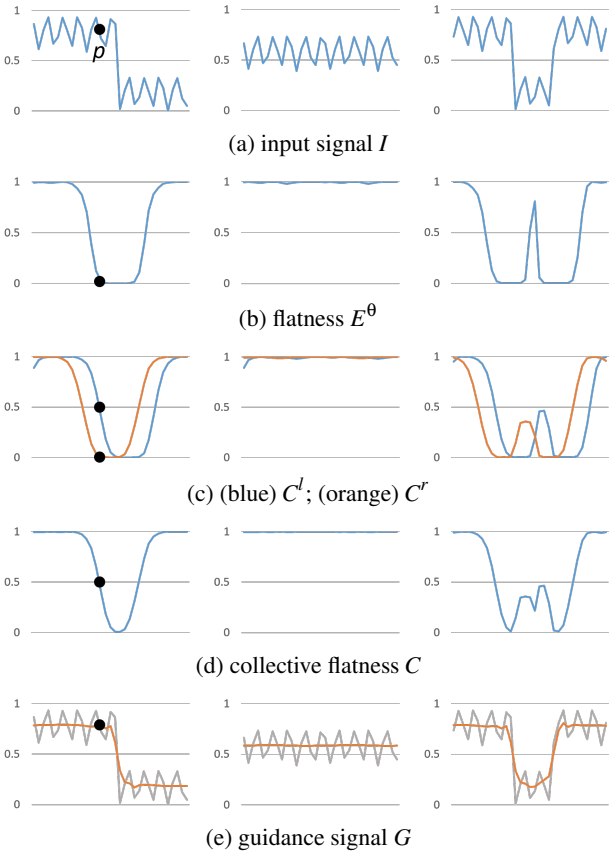


Figure 3: *1D synthetic examples (σ=5). (left) structure edge; (middle) texture region; (right) valley-like structure. In (b), texture regions have high flatness values while structure edges have low flatness values in the neighborhood. Our kernel scale estimation properly assigns small kernel scales to the pixels near structure edges, according to the estimated collective flatness in (d). Consequently, our guidance images in (e) effectively remove textures while preserving structure edges. Input signals are plotted with gray curves.*

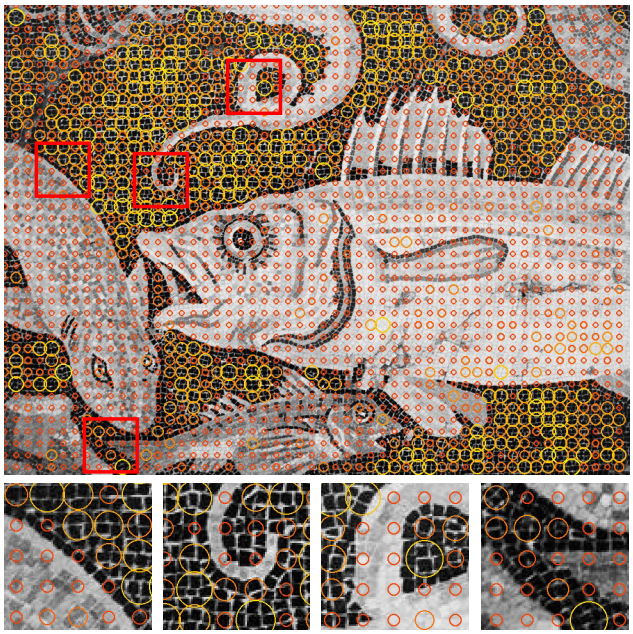


Figure 4: *Estimated kernel scales of grid sample points. A circle shows the size of the Gaussian filter kernel applied at a sample point ($\sigma = 4$).*

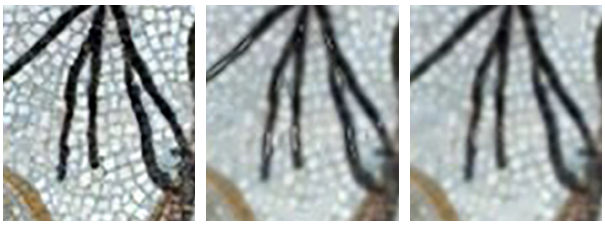


Figure 5: *Guidance images generated with adaptive kernel scales estimated using flatness and collective flatness. Collective flatness resolves the inverted filtering artifacts inside the stripe structures.*

tion of flatness values in the neighborhood. The *collective flatness* is computed as:

$$C_p = \max\left(C_p^l, C_p^r\right), \tag{7}$$

where C_p^l and C_p^r denote collective flatness from the left and right neighborhood of p, respectively. The left collective flatness C_p^l is obtained as:

$$C_p^l = \frac{1}{|N_l(p)|} \sum_{q \in N_l(p)} E_q^{\theta}, \tag{8}$$

where N_l denotes the left half of the 2D patch of size $\sigma \times \sigma$ rotated along the estimated structure direction θ_p. C_p^l is the average flatness of the pixels in the left neighborhood of p along the structure direction. Similarly, C_p^r is computed as the average flatness on the right half of the patch along the structure direction. Finally, the collective flatness C_p is the maximum of the two average flatness values.

Consider a pixel p in the left column of Fig. 3, where p is on the left side of the edge. The flatness E_p^{θ} of p itself is almost zero, but its left collective flatness C_p^l is larger than zero as the left neighbor pixels of p have non-zero flatness. In addition, as the distance of p from the edge increases, C_p^l also increases due to progressively more involvement of texture pixels in the left neighborhood. Consequently, we can use the value of C_p^l as an approximation of the distance from the edge. In this case, however, the right neighborhood of p contains the edge and C_p^r is smaller than C_p^l. Moreover, C_p^r does not decrease when p approaches the edge, and cannot well approximate the distance of p from the edge. In the case that pixel p is on the right side of the edge, C_p^r would be a better approximation of the distance from the edge than C_p^l. By taking the maximum of

(a) input (b) Cho et al. [CLKL14] (c) Zhang et al. [ZSXJ14] (d) our guidance image (e) our result

Figure 6: *Guidance images used for texture filtering. (a) input image, (b) Cho et al. [CLKL14] [$k = 7$], (c) Zhang et al. [ZSXJ14] [$\sigma_s = 4, \sigma_r = 0.1$], (d) our guidance image [$\sigma = 4, \sigma_r = 0.05$], (e) filtering result using the guidance image in (d). Compared to other methods, our guidance image better keeps small but salient features.*

C_p^l and C_p^r for the collective flatness C_p, we can handle both cases in a consistent way.

Now we estimate the kernel scale K_p using the collective flatness as follows:

$$K_p = \max\left(\sigma C_p, \delta\right), \tag{9}$$

where σ^2 is the variance of the Gaussian function used for computing dRTV in Eq. (3). Parameter δ is introduced to safeguard against signal oversharpening or aliasing ($\delta = 1$ by default). Note that K_p ranges in $[\delta, \sigma]$. Fig. 3d shows examples of kernel scale estimation for synthetic 1D signals, and Fig. 4 visualizes the estimated kernel scales for a real image. Fig. 5 demonstrates that the collective flatness C_p incorporates the neighborhood statistics and thus outperforms the original flatness E_p^θ in handling small-scale structures, such as narrow stripes.

5. Texture Filtering Process

5.1. Guidance image generation

With the estimated kernel scale K, we apply varying scale isotropic Gaussian smoothing on I to generate the guidance image G:

$$G_p = \frac{1}{k} \sum_{q \in N(p)} \exp\left\{ -\frac{1}{2} \left(\frac{(p_x - q_x)^2 + (p_y - q_y)^2}{K_p^2} \right) \right\} I_q, \tag{10}$$

where k is the normalizing factor. As demonstrated in Fig. 3e, the adaptive kernel scale K ensures aggressive filtering of texture regions while preserving structure edges in the resulting guidance image. Fig. 6 compares guidance images generated by existing methods and ours.

5.2. Guidance-based texture filtering

The final output S is obtained by applying a joint bilateral filter to the source image I using the guidance image G:

$$S_p = \frac{1}{k} \sum_{q \in N(p)} g_{\sigma_s}(p, q)\, g_{\sigma_r}(G_p, G_q)\, I_q, \tag{11}$$

Algorithm 1 Scale-aware texture filtering

Input: image I
Output: filtering result S
  $S \leftarrow I$
  for $k := 0 \ldots (N-1)$ **do**
    $E^\theta \leftarrow$ flatness values from S ▷ Eq. (6)
    $K \leftarrow$ kernel scales ▷ Eq. (9)
    $G \leftarrow$ guidance image from S using K ▷ Eq. (10)
    $S \leftarrow$ joint bilateral filtering of I using G ▷ Eq. (11)
  end for

where $g_{\sigma_s}(\cdot)$ and $g_{\sigma_r(\cdot)}$ give spatial and range weights with corresponding Gaussian kernel sizes σ_s and σ_r, respectively. Note that σ_s may be different from σ used in Eq. (3). The process of guidance image computation and joint bilateral filtering can be iterated a few times to improve the filtering result. Algorithm 1 summarizes the entire process of our method, and its intermediate images are shown in Fig. 7.

6. Experimental Results

Kernel scale estimation Our method estimates the proper kernel scales of pixels to remove textures while preserving the structure edges and corners. We evaluated our kernel scale estimation method on various texture images. Our method shows satisfactory estimation results even on the complex regions that multiple structures coexist nearby. Visualization examples of kernel scale estimation can be found in Fig. 4 and the supplementary material. Note that the little intrusion of kernel to structure edge does not introduce overblurring because we use Gaussian filter kernel which gives smaller weights for kernel boundaries.

Parameters Our method has only one parameter σ when generating a guidance image. Parameter σ controls the maximum scale of texture to be removed. Since our method can identify the per-pixel smoothing scale, σ does not have much impact on pixels nearby structure edges. However, for pixels away from structure edges

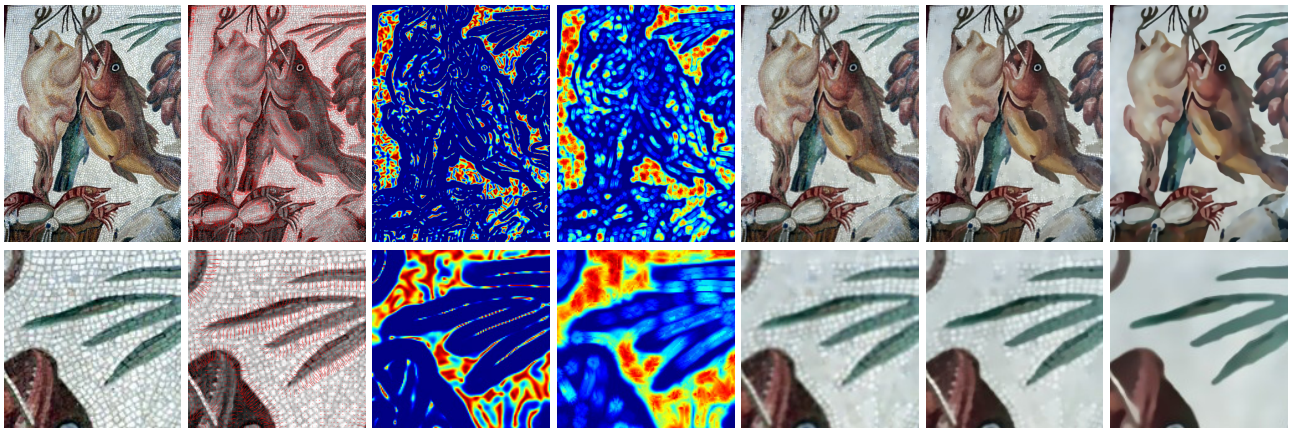


Figure 7: *Intermediate images of our filtering process. The lengths of red line segments in the second column show the values of directional flatness E^{θ}. In the third (E^{θ}) and fourth (C) columns, red color indicates large value while blue color indicates small value.*

Figure 8: *Effect of the maximum kernel scale parameter σ. Bigger textures are removed as σ increases. As our method automatically identifies the smoothing scale per pixel, small but salient structures (antennas and legs of the bee) are well preserved even when large textures are smoothed out with a big σ.*

(e.g., texture regions), a bigger σ increases the estimated kernel scale, and leads to more aggressive texture removal and smoother results. We used $\sigma = 4$ in most cases of our experiments. Fig. 8 shows the effect of parameter σ. The joint bilateral filtering step requires two parameters, spatial and range kernel sizes σ_s and σ_r. We find that making σ_s bigger than σ produces smoother results, and we set $\sigma_s = 1.5\sigma$ throughout. σ_r ranges in $[0.05, 0.1]$ (large value for images with high-contrast textures). We used 5 iterations of our filtering process throughout, similarly to [ZSXJ14].

Visual comparison In Fig. 9, we compare our results with the state-of-the-art texture filtering techniques. Compared to existing kernel-based methods [CLKL14, ZSXJ14, BSY*14, Yan16], ours generally performs better in terms of preserving small but salient features/corners and making them look sharp. Our results are also comparable to optimization-based approaches [XYXJ12, HCP15] that are hard to accelerate with GPU implementation. The scale parameters for existing methods are set to filter out high-contrast mosaic textures in the background for visual comparison. For Zhang et al. [ZSXJ14], as the domain transform filter in the author-provided code causes color bleeding, we instead used our bilateral filter based implementation for fair comparison.

Handling large scale textures Since existing methods use fixed-scale kernels for structure-texture separation, they often fail to preserve salient but small scale features such as corners when *large-scale* textures should be removed. In contrast, our adaptive kernel scale estimation facilitates filtering of large-scale textures without hurting the important structures and features. Fig. 10 demonstrates that our method outperforms other methods in this regard. Note that, for fair comparison, scale and smoothness parameters for other methods have been modulated enough for removing the largest textures.

Analysis on iterative filtering As described in Section 5, our filtering process consisting of guidance image generation and joint bilateral filtering can be iterated to remove desired amount of textures. Since our iterative process is similar to RGF [ZSXJ14], we compare the behavior of progressive filtering in both methods. As shown in Fig. 11, RGF aggressively smooths out all the details smaller than a certain scale, and then recovers the remaining structures and some details iteratively. On the other hand, our method focuses more on identifying pixels nearby structure elements and smoothing them conservatively through multiple filtering iterations, so that such structure elements are better preserved regardless of their scales, while smoothing out even large scale textures.

(a) input (b) RTV [XYXJ12] (c) BTF [CLKL14] (d) RGF [ZSXJ14]

(e) TreeFiltering [BSY*14] (f) Static & Dynamic [HCP15] (g) Semantic Filter [Yan16] (h) our method

Figure 9: *Visual comparison of texture filtering results. (a) input image, (b) RTV [$\lambda = 0.01, \sigma = 6, \varepsilon_s = 0.02, n_{iter} = 4$], (c) BTF [$k = 5, n_{iter} = 5$], (d) RGF [$\sigma_s = 5, \sigma_r = 0.1, n_{iter} = 4$], (e) TreeFiltering [$\sigma = 0.02, \sigma_s = 5, \sigma_r = 0.05, n_{iter} = 5$], (f) Static & dynamic filtering [$\lambda = 200, \sigma = 2, steps = 10$], (g) Semantic filtering with recursive bilateral filter [$level_{smoothing} = 4.5$], (h) our method [$\sigma = 4, \sigma_r = 0.1, n_{iter} = 5$] (best viewed in the original resolution).*

Timing data Table 1 shows timing statistics of our method. Although the MATLAB-based implementation of the proposed method is relatively slow than previous methods due to the spatially varying Gaussian smoothing and joint bilateral filtering, the parallel nature of our algorithm allows a drastic speedup on GPU. By implementing the algorithm using CUDA C++, our method can be accelerated up to more than 10 times faster than the MATLAB version. Moreover, exploiting fast implementation strategies [DD02, PD06, GO11] for bilateral and Gaussian smoothing would provide additional drastic speedup.

Applications Fig. 10 demonstrates the application of our method to detail enhancement. The optimization based approach [XYXJ12] shows best quality in terms of gradient preservation and artifact minimization. On the other hand, our result keeps shading information during the filtering process, and preserves small scale structures better than other kernel-based methods. As with many existing edge-aware filtering techniques, our method can be used to assist image abstraction and stylization. Inverse halftoning can also be handled with our method, where we can remove halftone dots

Component	$\sigma = 3$	$\sigma = 5$	$\sigma = 7$
	CPU/GPU	CPU/GPU	CPU/GPU
Comp. E, K	0.03s/26ms	0.04s/68ms	0.04s/112ms
Comp. G	0.08s/4ms	0.11s/8ms	0.12s/15ms
Comp. S	0.40s/11ms	1.05s/28ms	2.02s/53ms
Total	0.51s/0.041s	1.2s/0.1s	2.18s/0.18s

Resolution	400×300	800×600	1600×1200
Comp. E, K	15.53ms	55.3ms	205.03ms
Comp. G	1.89ms	6.39ms	21.50ms
Comp. S	6.09ms	21.59ms	77.41ms
Total	23.51ms	83.28ms	303.94ms

Table 1: *Timing data for varying maximum kernel scale parameter σ using a grayscale image of 800×600 pixels, and for varying image resolutions (GPU version) using $\sigma = 4$ with single filtering iteration. Times are measured using our MATLAB and CUDA implementations on a PC with Intel Core i7 CPU and NVIDIA GeForce GTX 980 graphic card running Windows 10.*

(a) input (b) RTV [XYXJ12] (c) BTF [CLKL14] (d) RGF [ZSXJ14] (e) our method

Figure 10: *Large-scale texture filtering (top) and detail enhancement (bottom) results. (a) input image, (b) RTV [XYXJ12] [$\lambda = 0.03, \sigma = 10, \varepsilon_s = 0.02, n_{iter} = 4$], (c) BTF [CLKL14] [$k = 6, n_{iter} = 5$], (d) RGF [ZSXJ14] [$\sigma_s = 10, \sigma_r = 0.1, n_{iter} = 10$], (e) our method [$\sigma = 18, \sigma_r = 0.04, n_{iter} = 8$]. For filtering large-scale textures, our method clearly outperforms previous methods in terms of preserving salient small-scale features such as corners (red box). As a result, our method enables the detail enhancement of an image containing large-scale textures without introducing halo artifacts near corners.*

from input images, revealing the underlying structures and shading (Fig. 12).

Additional results of our filtering method and comparisons with previous methods can be found in Fig. 13 and the supplementary material.

7. Conclusion

We have presented a novel structure-preserving image/texture filtering algorithm based on adaptive kernel scales. Our method relies on a collection of local statistics, *collective flatness*, to estimate per-pixel filter scale that is big enough to eliminate noise, texture, and clutter, but small enough to preserve all structure edges and corners. Our collective flatness measure and per-pixel kernel scale could be utilized as local statistical features for kernel based as well as optimized based image processing operations.

Our method is not without limitations. It assumes the oscillating property of textures and thus may not perfectly handle certain textures that would require explicit texture analysis or specific prior knowledge. Acceleration of the current implementation and its extension to video would also be an interesting future research direction as well.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) Grant (NRF-2014R1A2A1A11052779), Institute for Information and Communications Technology Promotion (IITP) Grant (R0126-16-1078), and Korea Creative Content Agency (KOCCA) Grant (APP0120150512002), funded by the Korea government (MSIP, MCST).

References

[BHY15] BI S., HAN X., YU Y.: An l 1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 78. 2

[BSY*14] BAO L., SONG Y., YANG Q., YUAN H., WANG G.: Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree. *IEEE Transactions on Image Processing 23*, 2 (2014), 555–569. 2, 6, 7

[CLKL14] CHO H., LEE H., KANG H., LEE S.: Bilateral texture filtering. *ACM Trans. Graphics 33* (2014), 128. 1, 2, 5, 6, 7, 8

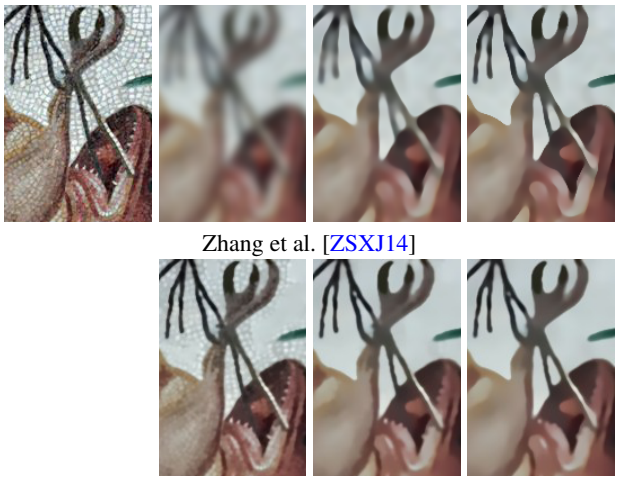


Figure 11: *Differently from Zhang et al. [ZSXJ14] that recovers structures after aggressively removing textures, our method conservatively and iteratively smooths pixels nearby salient features while smoothing out textures.*

Figure 12: *Inverse halftoning.*

[CT05] CHOUDHURY P., TUMBLIN J.: The trilateral filter for high contrast images and meshes. In *ACM SIGGRAPH 2005 Courses* (2005), p. 5. 2

[DD02] DURAND F., DORSEY J.: Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graphics 21*, 3 (2002), 257–266. 6

[EZ98] ELDER J., ZUCKER S.: Local scale control for edge detection and blur estimation. *IEEE Trans. Pattern Analysis Machine Intelligence 20*, 7 (1998), 699–716. 2

[FFLS08] FARBMAN Z., FATTAL R., LISCHINSKI D., SZELISKI R.: Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graphics 27*, 3 (2008), 67:1–67:10. 1, 2

[GO11] GASTAL E. S. L., OLIVEIRA M. M.: Domain transform for edge-aware image and video processing. *ACM Trans. Graphics 30* (2011), 69:1–69:12. 2, 6

[HCP15] HAM B., CHO M., PONCE J.: Robust image filtering using joint static and dynamic guidance. In *Proc. CVPR 2015* (2015), IEEE, pp. 4823–4831. 2, 6, 7

[HST10] HE K., SUN J., TANG X.: Guided image filtering. In *Proc. ECCV 2010* (2010), pp. 1–14. 1, 2

[KEE13] KARACAN L., ERDEM E., ERDEM A.: Structure-preserving image smoothing via region covariances. *ACM Trans. Graphics 32*, 6 (2013), 176:1–176:11. 1, 2

[KFEA10] KATKOVNIK V., FOI A., EGIAZARIAN K., ASTOLA J.: From local kernel to nonlocal multiple-model image denoising. *International Journal of Computer Vision 86* (2010), 1–32. 2

[KS10] KASS M., SOLOMON J.: Smoothed local histogram filters. *ACM Trans. Graphics 29*, 4 (2010), 100:1–100:10. 2

[MHW*13] MA Z., HE K., WEI Y., SUN J., WU E.: Constant time weighted median filtering for stereo matching and beyond. In *Proc. ICCV 2013* (2013), pp. 49–56. 2

[PD06] PARIS S., DURAND F.: A fast approximation of the bilateral filter using a signal processing approach. In *Proc. ECCV 2006* (2006), pp. 568–580. 6

[PHK11] PARIS S., HASINOFF S. W., KAUTZ J.: Local Laplacian filters: Edge-aware image processing with a Laplacian pyramid. *ACM Trans. Graphics 30*, 4 (2011), 68:1–68:12. 1, 2

[PM90] PERONA P., MALIK J.: Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Analysis Machine Intelligence 12*, 7 (1990), 629–639. 2

[PSA*04] PETSCHNIGG G., SZELISKI R., AGRAWALA M., COHEN M., HOPPE H., TOYAMA K.: Digital photography with flash and no-flash image pairs. *ACM Trans. Graphics 23* (2004), 664–672. 2

[SSD09] SUBR K., SOLER C., DURAND F.: Edge-preserving multiscale image decomposition based on local extrema. *ACM Trans. Graphics 28*, 5 (2009), 147:1–147:9. 2

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Proc. ICCV 1998* (1998), pp. 839–846. 1, 2

[Wei06] WEISS B.: Fast median and bilateral filtering. *ACM Trans. Graphics 25*, 3 (2006), 519–526. 2

[XLXJ11] XU L., LU C., XU Y., JIA J.: Image smoothing via L0 gradient minimization. *ACM Trans. Graphics 30*, 5 (2011), 174:1–174:12. 1, 2

[XYXJ12] XU L., YAN Q., XIA Y., JIA J.: Structure extraction from texture via relative total variation. *ACM Trans. Graphics 31*, 6 (2012), 139:1–139:10. 1, 2, 3, 6, 7, 8

[Yan16] YANG Q.: Semantic filtering. In *Proc. CVPR 2016* (2016), IEEE, pp. 4517–4526. 2, 6, 7

[ZSXJ14] ZHANG Q., SHEN X., XU L., JIA J.: Rolling guidance filter. In *Proc. ECCV*, vol. 8691. 2014, pp. 815–830. 1, 2, 5, 6, 7, 8, 9

[ZXJ14] ZHANG Q., XU L., JIA J.: 100+ times faster weighted median filter. In *Proc. CVPR 2014* (2014), pp. 2830–2837. 2

Figure 13: *Additional results. (top) input images; (bottom) texture filtering results.*