# Bilateral Texture Filtering

Hojin Cho
POSTECH

Hyunjoon Lee
POSTECH

Henry Kang
University of Missouri at St. Louis

Seungyong Lee
POSTECH

Input image | [Subr et al. 2009] | [Xu et al. 2012] | [Karacan et al. 2013] | Our method

**Figure 1:** *Bilateral texture filtering. In this example, the proposed method outperforms [Subr et al. 2009] in terms of both image structure preservation and texture smoothing. Compared to [Xu et al. 2012], our method effectively restores the shading on the object surface, and thus its original surface shape. [Karacan et al. 2013] overblurs some of the structure edges, which are preserved better in our scheme. Parameters: [Subr et al. 2009] ($k = 9$), [Xu et al. 2012] ($\lambda = 0.015, \sigma = 2$), [Karacan et al. 2013] ($k = 15, \sigma = 0.2$, Model 1), and our method ($k = 5, n_{itr} = 3$).*

## Abstract

This paper presents a novel structure-preserving image decomposition operator called *bilateral texture filter*. As a simple modification of the original bilateral filter [Tomasi and Manduchi 1998], it performs local patch-based analysis of texture features and incorporates its results into the range filter kernel. The central idea to ensure proper texture/structure separation is based on *patch shift* that captures the texture information from the most representative texture patch clear of prominent structure edges. Our method outperforms the original bilateral filter in removing texture while preserving main image structures, at the cost of some added computation. It inherits well-known advantages of the bilateral filter, such as simplicity, local nature, ease of implementation, scalability, and adaptability to other application scenarios.

**CR Categories:** I.4.3 [Image Processing and Computer Vision]: Enhancement—Smoothing; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

**Keywords:** bilateral filter, patch shift, texture smoothing, image decomposition

**Links:** DL PDF WEB

## 1 Introduction

Structure-preserving filtering is an essential operation with a variety of applications in computational photography and image analysis. Such an operation decomposes an image into prominent structure and fine-scale detail, making it easier for subsequent image manipulation such as tone mapping, detail enhancement, visual abstraction, scene understanding, and other tasks. Separating structure from detail often depends on measuring the size of local contrast, where structure is identified as pixels having relatively large contrast. However, when the fine-scale detail represents *texture*, as in Fig. 1, the conventional way of image decomposition may fail because texture often contains strong enough contrast to get confused with structure.

Many of the structure-preserving smoothing operators are based on local filtering [Perona and Malik 1990; Tomasi and Manduchi 1998; Fattal et al. 2007; Paris et al. 2011]. While these nonlinear filters are simple and intuitive to use, they are often ill-equipped to extract structure from texture due to having no explicit measure with which to distinguish the two. On the other hand, there are optimization-based [Yin et al. 2005; Aujol et al. 2006; Farbman et al. 2008; Subr et al. 2009; Buades et al. 2010; Xu et al. 2011; Xu et al. 2012] and patch-based [Karacan et al. 2013] solutions as well, some of which have been specifically designed to handle texture and thus outperform local filtering in terms of texture removal. However, they usually come with additional level of complexity and sophistication, which makes them harder to implement, accelerate, scale, or adapt.

In this paper, we present a novel method for nonlinear image decomposition based on a simple modification to bilateral filter [Tomasi and Manduchi 1998]. It is in essence a joint bilateral filter [Petschnigg et al. 2004; Eisemann and Durand 2004] that incorporates texture information (instead of color information) into the range filter kernel. We demonstrate that our method effectively removes texture while preserving structure, which the standard bilateral filter often fails to do. Being a simple extension to the popu-

lar bilateral filter, our method enjoys the benefits that come with it, such as simplicity, speed, ease of implementation, scalability, and adaptability. To distinguish the proposed method from its original formulation, we call it *bilateral texture filter*.

Our key idea to extract local texture without obscuring structure is *patch shift*, which for each pixel captures the texture information from the patch in the neighborhood that excludes prominent structure edges nearby and best represents the texture region containing the pixel. Patch shift in effect performs structure-preserving soft image segmentation of texture regions. The result of this operation is then used as the guidance image in our joint bilateral filtering. Therefore, the only additional step required over the standard bilateral filter is the computation of guidance image via patch shift, which can be achieved at a small computational cost.

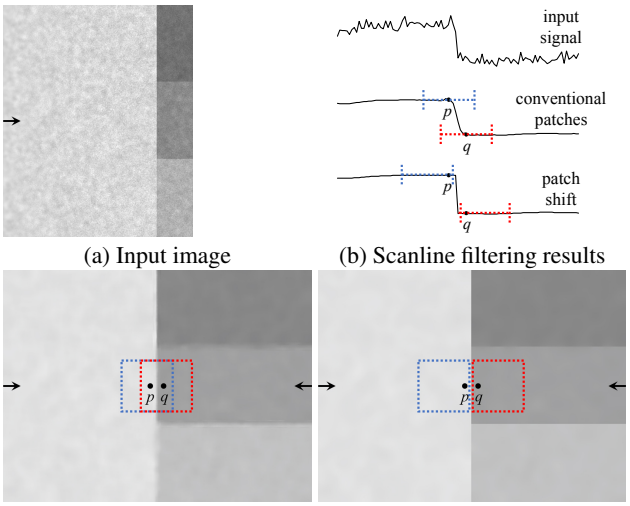
## 2 Related Work

Bilateral filter [Tomasi and Manduchi 1998] is one of the most widely used nonlinear operators for discontinuity-preserving image smoothing and decomposition. Its simplicity, effectiveness, and extendability led to its broader usage in other applications as well, such as tone mapping [Durand and Dorsey 2002], detail enhancement [Bae et al. 2006; Fattal et al. 2007], image editing [Oh et al. 2001; Chen et al. 2007], image upsampling [Kopf et al. 2007], mesh denoising [Jones et al. 2003; Fleishman et al. 2003], and artistic rendering [Winnemöller et al. 2006; Kang et al. 2009].

Subsequent development of more sophisticated edge-preserving filters, including weighted least squares (WLS) [Farbman et al. 2008], edge-avoiding wavelets [Fattal 2009], local histogram filtering [Kass and Solomon 2010], local Laplacian filtering [Paris et al. 2011], domain transform [Gastal and Oliveira 2011], and $L_0$ gradient minimization [Xu et al. 2011], all basically share the same goal of smoothing fine-scale details without degrading image structures, although they are not explicitly designed to deal with texture. Subr et al. [2009], on the other hand, defined detail as oscillations between local extrema in order to distinguish small-scale yet high-contrast features, i.e., texture, from real edges.

Regular or near-regular textures may be identified and filtered by exploiting spatial relationship, frequency, and symmetry of texture features [Liu et al. 2004; Hays et al. 2006]. Total variation (TV) [Rudin et al. 1992] on the other hand has proven to work well on filtering arbitrary texture of irregular shapes by enforcing TV regularization constraints to preserve large-scale edges. The original formulation of TV regularization was further extended to achieve better quality, robustness, and efficiency [Yin et al. 2005; Aujol et al. 2006; Buades et al. 2010; Xu et al. 2012]. In particular, Xu et al. [2012] introduced the notion of relative total variation (RTV), a spatially-varying total variation measure that helps improve the quality of texture-structure separation.

Recently, Karacan et al. [2013] proposed a patch-based texture removal algorithm that uses the similarity measures based on a region covariance descriptor. Compared to the conventional pixel-based image decomposition methods, the use of covariance matrix associated with each patch in the neighborhood enables a more accurate description and identification of texture feature, leading to better performance in separating texture from structure. On the other hand, a patch-based approach is also prone to overblur the structure edges since the overlapping patches near an edge inevitably share similar statistics.




(a) Input image (b) Scanline filtering results

(c) Conventional patches (d) Patch shift

**Figure 2:** *Patch shift. Conventionally, texture feature is computed in a patch centered at each pixel, in which case the patches for two adjacent pixels should have a large overlap, reducing the feature discriminability. In contrast, patch shift finds a nearby patch that stays clear of a prominent structure edge. (b) Filtering of the scanline marked by arrows. (c) Filtered by [Karacan et al. 2013]. (d) Filtered with patch shift. The results in (b) show that our approach preserves structure edges, unlike the conventional approach.*

## 3 Key Idea: Patch Shift

Given a scalar-valued input image $I$, the bilateral filter [Tomasi and Manduchi 1998] computes an output image $J$ by

$$J_p = \frac{1}{k_p} \sum_{q \in \Omega_p} f(||q-p||) g(||I_q - I_p||) I_q, \qquad (1)$$

where $k_p$ is a normalizing term. The output $J_p$ at pixel $p$ is a weighted average of $I_q$ in the spatial neighborhood $\Omega_p$. The spatial kernel $f$ and the range kernel $g$ are typically Gaussian functions. The data-dependent weight $g$ is inversely proportional to the size of contrast between two pixels $p$ and $q$. This nonlinear weighting enables bilateral filter to blur small-scale intensity variations while preserving salient edges.

We extend the bilateral filter by substituting a texture description image $G$ in the range kernel $g$:

$$J_p = \frac{1}{k_p} \sum_{q \in \Omega_p} f(||q-p||) g(||G_q - G_p||) I_q, \qquad (2)$$

This is a texture-filtering variant of Eq. (1), and its success depends heavily on the design of $G$, which is also called *guidance image* in the context of joint bilateral filtering [Petschnigg et al. 2004; Eisemann and Durand 2004].

The value of $G_p$ can be defined by analyzing local image statistics [Manjunath and Ma 1996; Tuzel et al. 2006] in a rectangular *patch* $\Omega_p$ centered at $p$. When a patch contains both texture and structure, however, such local statistics may obscure the existence of salient edges or region boundaries. For example, two neighboring patches (each with size $k \times k$) centered at two adjacent pixels $p = (i, j)$ and $q = (i, j+1)$, respectively, must have a large overlap of size $k \times (k-1)$. Consequently, these local statistics should be similar even when $p$ and $q$ happen to be on the opposite sides

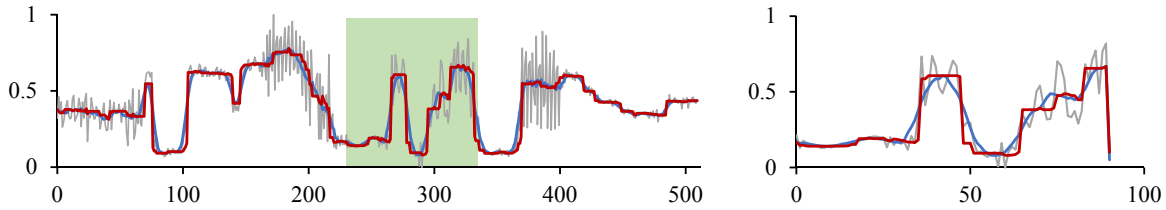



**Figure 3:** *1D example of guidance image computation using patch shift. The input signal is a scanline of the Barbara image.*

of an edge. While Karacan *et al.* [2013] alleviated this problem by including pixel position in computing the region covariance, it is often not enough to guard against edge blurring (Fig. 2c).

We overcome this limitation by introducing a novel method called *patch shift*. Assuming a $k \times k$ box representing a patch, each pixel $p$ has a total of $k^2$ patches in $I$ that contains $p$. Among these $k^2$ patches, we find the patch $\Omega_q$ that is least likely to contain a prominent structure edge. Once we have found $\Omega_q$ that has this property, we use the average intensity within this patch, denoted $B_q$, as the representative texture value $G_p$ at $p$ in Eq. (2). In a nutshell, patch shift finds the texture patch in the neighborhood that most likely stays clear of the structure edge (if present) and best represents the texture region that the pixel belongs to (Fig. 2d).

There are several possible choices for defining a texture measure in such a way that it ensures separation from structure edges. In this paper, we define texture as *fine-scale spatial oscillations* of signals, as in [Subr et al. 2009; Xu et al. 2012]. Let us assume for the time being that texture signal has smaller amplitude than the neighboring structure edge (this requirement will be lifted in Section 4), then we can simply measure the likelihood of containing structure edge for a patch $\Omega_q$ via its *tonal range* $\Delta(\Omega_q)$:

$$\Delta(\Omega_q) \quad = \quad I_{max}(\Omega_q) - I_{min}(\Omega_q). \tag{3}$$

where $I_{max}(\Omega_q)$ and $I_{min}(\Omega_q)$ denote the maximum and the minimum image intensities in $\Omega_q$, respectively. We then let patch shift select the patch with the minimum tonal range, which is to minimize the probability of involving a salient edge when computing texture feature.

Fig. 3 illustrates how patch shift works on a 1D signal, when tonal range is used as the texture measure. A patch is in this cas e defined as an interval of width $k$. For every $p$ in the input signal $I$, we precompute the average intensity $B_p$ within its own center patch $\Omega_p$. Then the texture signal $G_p$ at $p$ is obtained by copying $B_q$ at $q$ that has the smallest $\Delta(\Omega_q)$ in the neighborhood of $p$. Note that the patch shift process successfully flattens the (small) oscillations in texture regions without degrading the structure edges. In case $p$ is part of a thin texture region, there may be more than one structure edges in the neighborhood, in which case patch shift still tries its best to stay away from the biggest edge.

## 4 Algorithm

Our 2D filtering process is an extension of the 1D process described above. Given an input image $I$, we first apply $k \times k$ box kernel to compute the average image $B$. For each pixel $p$, we also compute the tonal range $\Delta(\Omega_p)$ in Eq. (3). We then obtain the guidance image $G$ via patch shift on each pixel. That is, we find the patch $\Omega_q$ whose $\Delta(\Omega_q)$ is the minimum among $k^2$ candidates, then copy $B_q$ to $G_p$. Finally we obtain the output image $J$ by applying joint bilateral filter on $I$, using $G$ as the guidance image. While this process generally performs well in terms of texture-structure decomposition, we make two modifications to improve the robustness of our scheme.

Eq. (3) suggests that patch shift may not work properly if the tonal range within a pure texture region is as large as (or larger than) the nearby structure edge. We resolve this by adapting Relative Total Variation (RTV) [Xu et al. 2012]. We define *modified Relative Total Variation (mRTV)* as

$$\text{mRTV}(\Omega_q) \quad = \quad \Delta(\Omega_q) \frac{\max\limits_{r \in \Omega_q} |(\partial I)_r|}{\sum\limits_{r \in \Omega_q} |(\partial I)_r| + \epsilon}, \tag{4}$$

where $|(\partial I)_r|$ denotes the gradient magnitude at pixel $r \in \Omega_q$ and $\epsilon$ is a small value to avoid division by zero. In our implementation, we use $|(\partial I)_r| = \sqrt{(\partial_x I)_r^2 + (\partial_y I)_r^2}$ and $\epsilon = 10^{-9}$. The tonal range $\Delta(\Omega_q)$ serves as a scale factor in Eq. (4) to reflect the absolute magnitude of the signal.

The mRTV value is relatively large in a structure patch containing only a few edges, and relatively small in a texture patch having frequent oscillations. For a $k \times k$ 2D patch, $\frac{\text{mRTV}(\Omega_q)}{\Delta(\Omega_q)}$ would be approximately $\frac{1}{k}$ for a horizontal or vertical step edge, and $\frac{1}{k^2}$ for a texture patch with full oscillations. Note that this is true even when the texture amplitudes are as large as (or larger than) the edges nearby. Therefore, the use of mRTV enables filtering of texture with arbitrarily large magnitudes. Our modified patch shift operation should now locate a pure texture patch among $k^2$ candidates by finding the patch $\Omega_q$ with the minimum mRTV value. Fig. 4h shows the mRTV values computed using Eq. (4), and Fig. 4c shows the guidance image $G$ obtained via mRTV-driven patch shift, which effectively restores structure edges from the blurred image $B$.

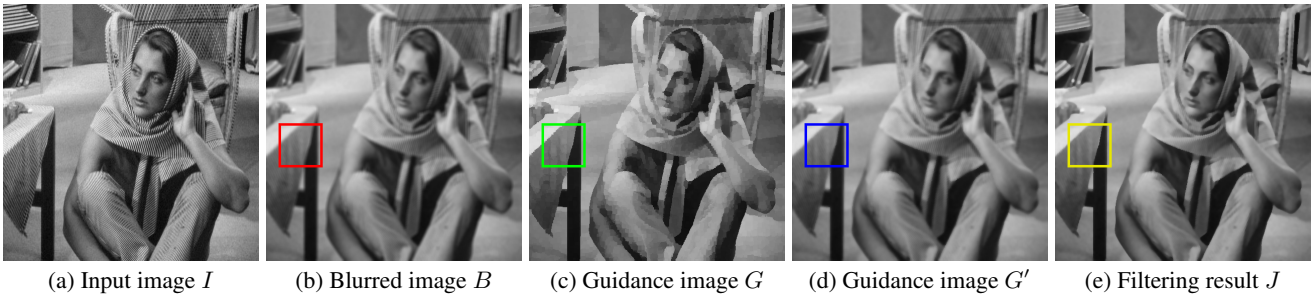
The mRTV values in a smooth or flat image region tend to be very small and thus may become sensitive to image noise. For example, in a smooth region where intensity changes gradually, small noisy peaks can be misinterpreted as edges, resulting in a wrong $B_q$ value being copied to $G_p$ and thus disrupting the gradual intensity variation. To prevent this, we examine the mRTV values of $\Omega_p$ and $\Omega_q$ when copying $B_q$ to $G_p$. If the two mRTV values are similar (meaning similar local statistics), $B_p$ is preferred over $B_q$ as the value of $G_p$. If and only if $\text{mRTV}(\Omega_q)$ is considerably smaller than $\text{mRTV}(\Omega_p)$ (meaning $\Omega_q$ is obviously more flat or homogeneous), $B_q$ is used for $G_p$.

This strategy can be implemented by interpolating images $B$ and $G$ using the difference in mRTV values as blending weight. That is,

$$G'_p \quad = \quad \alpha_p G_p + (1 - \alpha_p) B_p, \tag{5}$$

where

$$\alpha_p = 2 \left( \frac{1}{1 + \exp(-\sigma_\alpha(\text{mRTV}(\Omega_p) - \text{mRTV}(\Omega_q)))} - 0.5 \right). \tag{6}$$



(a) Input image $I$ (b) Blurred image $B$ (c) Guidance image $G$ (d) Guidance image $G'$ (e) Filtering result $J$

(f) Close ups of regions (g) Tonal range $\Delta$ (h) mRTV (i) Alpha map $\alpha$ (j) Detail map $(I - J)$

**Figure 4:** *Overall process and intermediate images of our bilateral texture filtering (see Fig. 6 for color codes).*

**Algorithm 1** Bilateral texture filtering

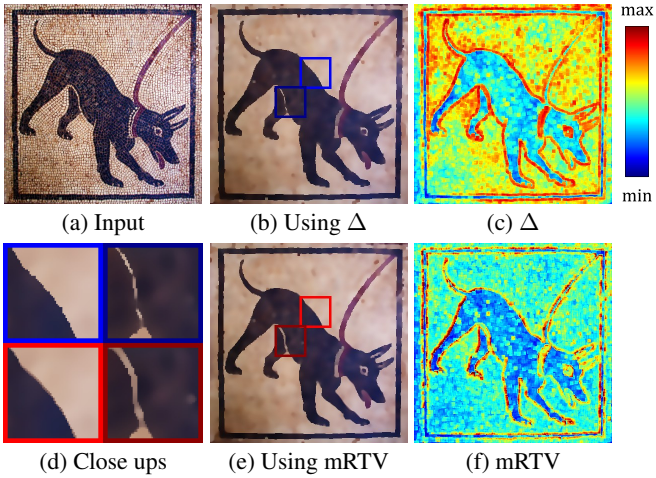
**Input:** image $I$
**Output:** texture filtered image $J$
  **for** $iter = 1 : n_{itr}$ **do**
    $B \leftarrow$ Uniform blurring of $I$
    mRTV $\leftarrow$ Compute Eq. (4) for each pixel $p$
    **for all** $p \in I$ **do**
      Find $q \in \Omega_p$ with minimum mRTV$_q$ ▷ *patch shift*
      $G_p \leftarrow B_q$
    **end for**
    $\alpha \leftarrow$ Compute Eq. (6) for each pixel $p$
    $G' \leftarrow \alpha G + (1 - \alpha)B$ ▷ *Eq. (5)*
    $J \leftarrow$ joint bilateral filtering of $I$ using $G'$ as guidance
    $I \leftarrow J$ ▷ *input for the next iteration*
  **end for**

The weight $\alpha_p \in [0, 1]$ is small inside smooth/texture regions, and large around edges (Fig. 4i). In Eq. (6), $\sigma_\alpha$ controls the sharpness of the weight transition from edges to smooth/texture regions, where a bigger $\sigma_\alpha$ means sharper transition. We use $\sigma_\alpha = 5k$ in our experiments. The interpolated image $G'$ is the modified guidance image (Fig. 4d) that we finally use in our joint bilateral filtering of Eq. (2) (Fig. 4e). Fig. 4j shows the detail map obtained by $I - J$.

For image denoising, a single iteration of bilateral filtering is often sufficient. However, texture may have spatial and/or range scales that are much bigger than that of noise. Therefore, depending on the input, more than one (usually $3 \sim 5$) iterations of bilateral texture filtering might be necessary to obtain a desired effect. Algorithm 1 summarizes our final algorithm.

## 5 Analysis

**Parameters** In our algorithm, there are practically only two parameters to control, $k$ (patch size) and $n_{itr}$ (number of iterations). $k$ basically determines the scale of texture to be removed. Fig. 5 shows results with various patch sizes. We used $k \in \{3, 5, 7, 9\}$ and $n_{itr} \in \{3, 4, 5\}$ for most examples in the paper. The joint bilateral filtering (the last step of our algorithm) comes with its original parameters; spatial kernel size $s \times s$, and spatial/range blur parameters $\sigma_s$ and $\sigma_r$. Spatial kernel size $s$, which may differ from the patch size $k$, determines the smoothness of the filter output. We set $s = 2k - 1$, $\sigma_s = k - 1$, and $\sigma_r = 0.05 \times \sqrt{c}$, where $c$ is the number of color channels of an image. The supplementary material contains results with different values of $s$ and the complete list of parameter values for all the examples in the paper.




(a) Input (b) Using $\Delta$ (c) $\Delta$

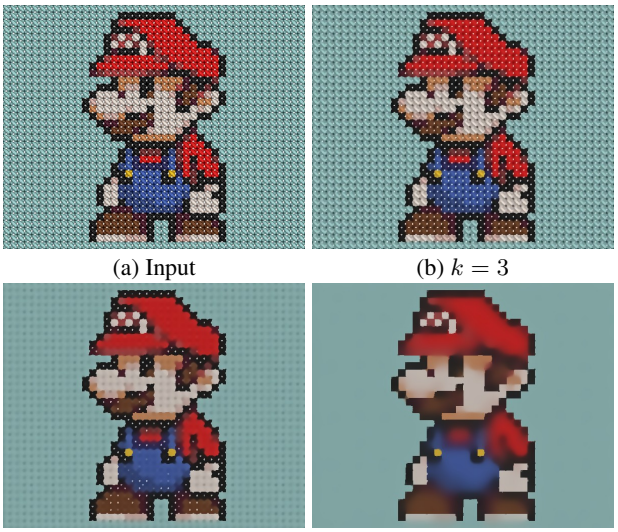
(d) Close ups (e) Using mRTV (f) mRTV

**Figure 6:** *Comparison of different texture measures. (b) and (e) show filtering outputs using tonal range $\Delta$ and mRTV, respectively. While their overall filtering results are comparable, (f) shows that mRTV delivers better texture-structure separation especially near edges. The $\Delta$ and mRTV values are normalized for visualization.*

**Texture-structure separation measure** We have proposed two measures to distinguish texture, i.e., fine-scale oscillations, from structure edges. For many of the images without large texture oscillations, the patch-wise tonal range $\Delta(\Omega)$ is sufficient to separate texture from structure edges. Our mRTV measure on the other hand shows better separation quality when texture signals are strong and noisy. Fig. 6 shows an example.

(a) Input (b) $k = 5$ (c) $k = 9$ (d) $k = 13$ (e) $k = 17$

**Figure 5:** *Results with various patch sizes. A bigger $k$ removes more (and bigger) textures, at the cost of losing some fine details. (top) Input image courtesy Duncan Hull. (bottom) Input image from flickr user Seattleye.*



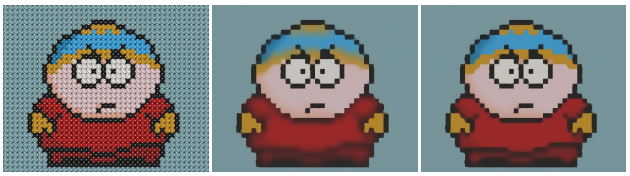
(a) Input (b) $k = 3$ (c) $k = 5$ (d) $k = 9$

**Figure 7:** *Multiscale texture filtering. From the input image (a), we progressively remove textures to get (b), (c), and (d). Input image courtesy Mark Delaney.*

**Method for initial blurring** While we have used a uniform box filter in generating the average (blurred) image $B$, other linear filter such as Gaussian blur may be used instead. Even a structure-preserving smoothing technique [Subr et al. 2009; Xu et al. 2011; Xu et al. 2012; Karacan et al. 2013] may be employed here, but at an unnecessary cost as the structure edges will be restored by patch shift anyway. The basic requirement for $B$ is to have the image texture properly smoothed out, and we found the box filter good enough for all of our experiments. Substituting other filters made little difference as shown in the supplementary material.

**Multiscale filtering** A multiscale extension of our algorithm should help deal with features of different scales. For multiscale bilateral texture filtering, we start from the initial scale with a small patch size $k$, then progressively move on to the next scale with an increased patch size $k'$ by using the previous output as input. Fig. 7 shows an example.

**Noise** Our method is robust against both Gaussian noise and salt-and-pepper noise (some results are in the supplementary material). Even a high-peak impulse noise on a random pixel would be significantly attenuated in the guidance image $G$ by substituting a pixel value from the blurred image $B$. Also, patch shift would stay away from this noise pixel when operating in the neighborhood.



(a) (b) (c)

**Figure 8:** *Color image filtering. (a) Input image. (b) Using grayscale guidance image. (c) Using color guidance image. Input image courtesy Mark Delaney.*

**Color image filtering** A color image (assuming RGB mode) is first converted to grayscale, from which we compute a grayscale guidance image $G'$. We then perform joint bilateral filtering on each color channel using $G'$. Most examples in this paper have been produced this way. Alternatively, we may exploit full color information from the start. For example, we first box filter the color image in each channel to construct $B$ in color. In the next step we compute mRTV separately in each channel, then add them together to form a scalar value. Patch shift is then performed with respect to this scalar measure, to construct an RGB guidance image $G'$ from the color values of $B$. Finally, joint bilateral filter is applied on each of RGB channels using $G'$. A CIELab mode image may be used and processed similarly. Explicit handling of color values may result in clearer separation between colored regions (Fig. 8).

## 6 Results

**Comparison with state-of-the-art** In Figs. 1 and 9, we compare our method with the state-of-the-art nonlinear image smoothing techniques that were specifically designed to perform texture removal [Subr et al. 2009; Xu et al. 2012; Karacan et al. 2013]. In generating results for these techniques, we used the implementations provided online by the authors and fine tuned the parameters manually. All the methods we tested generally succeeded in extracting prominent image structure while filtering out texture. As pointed out in [Xu et al. 2012; Karacan et al. 2013], however, we noticed that the method of Subr et al. [2009] often degrades image structures and exhibits blur artifacts, due to the difficulty of locating extrema in regions containing the mixture of texture and structure (Fig. 9b). The method of Xu et al. [2012] shows a robust performance in both texture removal and structure preservation/enhancement. As a byproduct of global optimization, however,

(a) Input (b) [Subr et al. 2009] (c) [Xu et al. 2012] (d) [Karacan et al. 2013] (e) Our method

**Figure 9:** *Comparison with previous methods on "Pompeii Fish Mosaic". Previous methods, (b)-(d), went through careful manual parameter tuning. Parameters: [Subr et al. 2009] ($k = 13$), [Xu et al. 2012] ($\lambda = 0.015, \sigma = 6$), [Karacan et al. 2013] ($k = 19, \sigma = 0.2$, Model 1), and our method ($k = 7, n_{itr} = 5$). Input image courtesy Chris Beckett.*

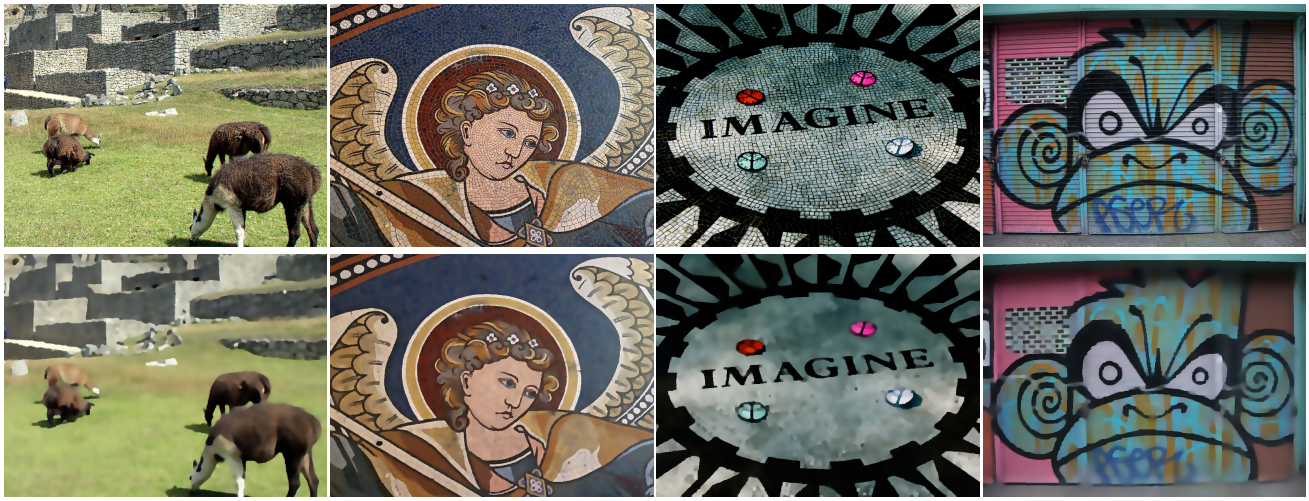



**Figure 10:** *More results of bilateral texture filtering. (top) Input images. (bottom) Our filtering results. (left to right) Input image courtesy flickr users YoTuT, Lawrence Rice, Alexander Kauschanski, and bixentro.*

oversmoothing of details may occur, which obscures the surface shading and makes the resulting image look somewhat flat. It also appears to be difficult with this method to eliminate texture located near a structure edge, possibly due to their strong edge preservation property (Fig. 9c). The covariance-based method proposed by Karacan *et al.* [2013] removes texture effectively while preserving edges and surface shading, but may oversmooth structure due to the inherent limitation of covariance descriptor in locating edges (Fig. 9d). On the other hand, our method consistently preserves both structure and shading information without leaving unprocessed texture (Fig. 9e). The supplementary material contains more comparisons using other input images. Fig. 10 shows additional results of our bilateral texture filtering.

**Timing data** Due to the added computation, our bilateral texture filtering runs several times slower than the original bilateral filter. With our unoptimized Matlab implementation, the total processing time for a single application takes about 1 to 2 seconds for a grayscale image of $800 \times 600$ pixels. We have also implemented our algorithm on GPU using C++ CUDA. For the same image resolution, our GPU version took about 2 to 3 milliseconds. This high performance with GPU is due to the local nature of our algorithm and its operations. See Table 1 for detailed timing statistics.

| Component | $k = 3$ | $k = 5$ | $k = 7$ |
|---|---|---|---|
| | CPU / GPU | CPU / GPU | CPU / GPU |
| Comp. mRTV | 0.181s / 0.454ms | 0.480s / 0.557ms | 0.822s / 0.879ms |
| Computing $B$ | 0.003s / 0.111ms | 0.003s / 0.168ms | 0.003s / 0.201ms |
| Patch shift | 0.157s / 0.212ms | 0.310s / 0.432ms | 0.497s / 0.749ms |
| Computing $\alpha$ | 0.007s / 0.196ms | 0.007s / 0.100ms | 0.007s / 0.125ms |
| JBF (Eq. (2)) | 0.077s / 0.200ms | 0.213s / 0.776ms | 0.363s / 1.455ms |
| Total | 0.425s / 1.173ms | 1.013s / 2.033ms | 1.692s / 3.409ms |

**Table 1:** *Timing data for a grayscale image of* $800 \times 600$ *pixels with a single iteration measured using our unoptimized Matlab (CPU) and C++ CUDA (GPU) implementations on a PC with Intel Core i7 CPU 950, 12GB RAM, and NVIDIA GeForce GTX 780 graphic card running Windows 7.*

**Applications** Our filter may be used to reduce image compression artifacts from cartoon images (Fig. 11). The proposed patch shift mechanism effectively identifies and suppresses the high fluctuation of noisy pixels near the strong edges, while preserving structure information. As a nonlinear edge-preserving filter, our method can be used for detail enhancement via layer decomposition. As shown in Fig. 12, the quality of our detail enhancement is comparable to that of the state-of-the-art methods. Fig. 13 shows an application of inverse halftoning, that aims to remove stipple dots from the halftone images. While our method is not tailored to solve this particular problem, it shows good performance in terms of re-

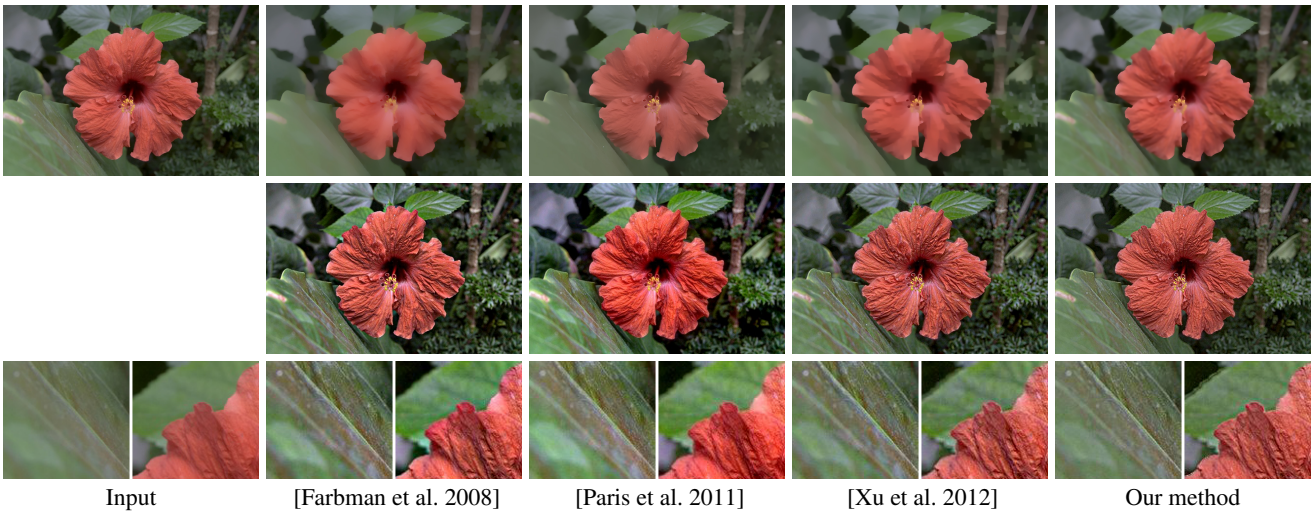



**Figure 12:** *Detail enhancement result compared with previous methods. Input and filtered images are in the top row, and the detail-enhanced results are in the middle with close-ups in the bottom. Parameters: [Farbman et al. 2008] ($\lambda = 1, \alpha = 1.2$), [Paris et al. 2011] ($\alpha = 2, \sigma_r = 0.4$), [Xu et al. 2012] ($\lambda = 0.015, \sigma = 0.5$), and our method ($k = 3, n_{itr} = 5$).*

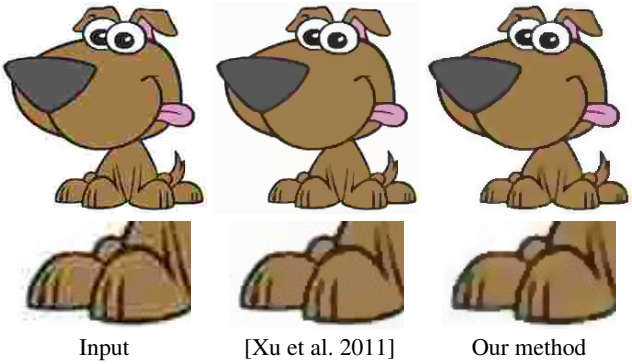



**Figure 11:** *Cartoon JPEG artifact removal by [Xu et al. 2011] and our method. Input image from [Xu et al. 2011].*

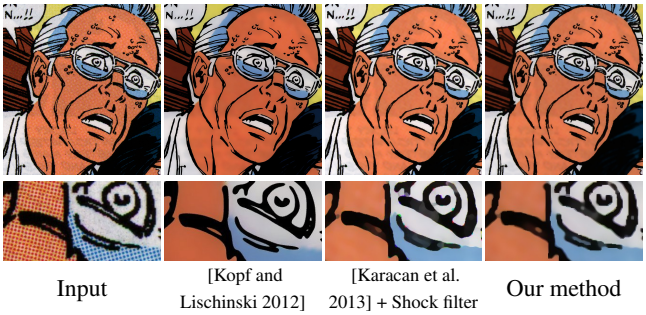



**Figure 13:** *Inverse halftoning. The method of [Kopf and Lischinski 2012] is dedicated to this problem and produces the best result. Compared to [Karacan et al. 2013], our method preserves well the shape of the original black lines. Input image © Marvel Comics.*

moving dots while keeping important structure edges without the need for post-processing such as shock filtering.

## 7 Discussion and Future Work

Our bilateral texture filter retains the simplicity of the original bilateral filter, yet provides significantly enhanced performance in separating texture details from image structures. We expect this simplicity, efficiency, and effectiveness to open up interesting application possibilities. The proposed patch shift mechanism plays a key role in our method as it finds appropriate texture/smooth patch for each pixel that is needed to generate a guidance image. Patch shift is a general concept and does not depend on any specific definition of texture feature. Therefore, its usefulness and applicability could be further explored in a larger context of research on image processing.

**Limitations** Although we designed the mRTV measure to handle textures with strong oscillations, our method may still have trouble with extreme variations inside a texture region (Fig. 14, top). Another limitation case is the mixture of large-scale textures with obscure borders between them. Since patch shift depends on identifying structural edges, our method would fail in this case (Fig. 14, bottom).

**Future work** These limitations serve as motivations for future work. For instance, the mRTV measure could be further enhanced so that a broader range of textures could be handled within our framework. As for the patch shift mechanism, a logical next step would be to explore the design of a structure-adaptive patch type/shape, which could possibly lead to enhanced quality of texture-structure separation. Developing a video texture filtering framework would be an interesting future extension as well.

## Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments. This work was supported in part by Basic Science Research Program of NRF (2013R1A1A2011692, 2012-0008835) and IT/SW Creative Research Program of NIPA (2013-H0503-13-1013).

## References

AUJOL, J.-F., GILBOA, G., CHAN, T., AND OSHER, S. 2006. Structure-texture image decomposition–modeling, algorithms, and parameter selection. *International Journal of Computer Vision 67*, 1, 111–136.

**Figure 14:** *Limitations. (top to bottom) Input image courtesy flickr user funkblast and limonada.*


BAE, S., PARIS, S., AND DURAND, F. 2006. Two-scale tone management for photographic look. *ACM Trans. Graphics 25*, 3, 637–645.

BUADES, A., LE, T. M., MOREL, J.-M., AND VESE, L. A. 2010. Fast cartoon + texture image filters. *IEEE Trans. Image Processing 19*, 8, 1978–1986.

CHEN, J., PARIS, S., AND DURAND, F. 2007. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graphics 26*, 3, 103:1–103:9.

DURAND, F., AND DORSEY, J. 2002. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graphics 21*, 3, 257–266.

EISEMANN, E., AND DURAND, F. 2004. Flash photography enhancement via intrinsic relighting. *ACM Trans. Graphics 23*, 673–678.

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

FATTAL, R., AGRAWALA, M., AND RUSINKIEWICZ, S. 2007. Multiscale shape and detail enhancement from multi-light image collections. *ACM Trans. Graphics 26*, 3, 51:1–51:9.

FATTAL, R. 2009. Edge-avoiding wavelets and their applications. *ACM Trans. Graphics 28*, 3, 22:1–22:10.

FLEISHMAN, S., DRORI, I., AND COHEN-OR, D. 2003. Bilateral mesh denoising. *ACM Trans. Graphics 22*, 3, 950–953.

GASTAL, E. S. L., AND OLIVEIRA, M. M. 2011. Domain transform for edge-aware image and video processing. *ACM Trans. Graphics 30*, 4, 69:1–69:12.

HAYS, J., LEORDEANU, M., EFROS, A. A., AND LIU, Y. 2006. Discovering texture regularity as a higher-order correspondence problem. In *Proc. ECCV 2006*, 522–535.

JONES, T. R., DURAND, F., AND DESBRUN, M. 2003. Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graphics 22*, 3, 943–949.

KANG, H., LEE, S., AND CHUI, C. 2009. Flow-based image abstraction. *IEEE Trans. Visualization and Computer Graphics 15*, 62–76.

KARACAN, L., ERDEM, E., AND ERDEM, A. 2013. Structure-preserving image smoothing via region covariances. *ACM Trans. Graphics 32*, 6, 176:1–176:11.

KASS, M., AND SOLOMON, J. 2010. Smoothed local histogram filters. *ACM Trans. Graphics 29*, 4, 100:1–100:10.

KOPF, J., AND LISCHINSKI, D. 2012. Digital reconstruction of halftoned color comics. *ACM Trans. Graphics 31*, 6, 140:1–140:10.

KOPF, J., COHEN, M., LISCHINSKI, D., AND UYTTENDAELE, M. 2007. Joint bilateral upsampling. *ACM Trans. Graphics 26*, 3, 96.

LIU, Y., LIN, W.-C., AND HAYS, J. 2004. Near-regular texture analysis and manipulation. *ACM Trans. Graphics 23*, 3, 368–376.

MANJUNATH, B. S., AND MA, W. Y. 1996. Texture features for browsing and retrieval of image data. *IEEE Trans. Pattern Analysis Machine Intelligence 18*, 8, 837–842.

OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-based modeling and photo editing. In *Proc. ACM SIGGRAPH 2001*, ACM Press, New York, NY, USA, 433–442.

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

PERONA, P., AND MALIK, J. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Analysis Machine Intelligence 12*, 7, 629–639.

PETSCHNIGG, G., SZELISKI, R., AGRAWALA, M., COHEN, M., HOPPE, H., AND TOYAMA, K. 2004. Digital photography with flash and no-flash image pairs. *ACM Trans. Graphics 23*, 664–672.

RUDIN, L. I., OSHER, S., AND FATEMI, E. 1992. Nonlinear total variation based noise removal algorithms. *Physica D 60*, 259–268.

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

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proc. ICCV 1998*, 839–846.

TUZEL, O., PORIKLI, F., AND MEER, P. 2006. Region covariance: A fast descriptor for detection and classification. In *Proc. ECCV 2006*, 589–600.

WINNEMÖLLER, H., OLSEN, S. C., AND GOOCH, B. 2006. Real-time video abstraction. *ACM Trans. Graphics 25*, 3, 1221–1226.

XU, L., LU, C., XU, Y., AND JIA, J. 2011. Image smoothing via L0 gradient minimization. *ACM Trans. Graphics 30*, 5, 174:1–174:12.

XU, L., YAN, Q., XIA, Y., AND JIA, J. 2012. Structure extraction from texture via relative total variation. *ACM Trans. Graphics 31*, 6, 139:1–139:10.

YIN, W., GOLDFARB, D., AND OSHER, S. 2005. Image cartoon-texture decomposition and feature selection using the total variation regularized L1 functional. *Variational, Geometric, and Level Set Methods in Computer Vision 3752*, 73–84.