Volume xx (200y), Number z, pp. 1–13

Structure-Texture Decomposition of Images with Interval Gradient

Hyunjoon Lee[1], Junho Jeon[1], Junho Kim[2], and Seungyong Lee[1]

[1]POSTECH, Pohang, Korea
{crowlove, zwitterion27, leesy}@postech.ac.kr
[2]Kookmin Univ, Seoul, Korea
junho@kookmin.ac.kr

Abstract

This paper presents a novel filtering-based method for decomposing an image into structures and textures. Unlike previous filtering algorithms, our method adaptively smooths image gradients to filter out textures from images. A new gradient operator, the interval gradient, is proposed for adaptive gradient smoothing. Using interval gradients, textures can be distinguished from structure edges and smoothly varying shadings. We also propose an effective gradient-guided algorithm to produce high quality image filtering results from filtered gradients. Our method avoids gradient reversal in the filtering results and preserves sharp features better than existing filtering approaches, while retaining simplicity and highly parallel implementation. The proposed method can be utilized for various applications that require accurate structure-texture decomposition of images.

Keywords: texture filtering, interval gradient, gradient-domain image decomposition, 1D filtering

Categories and Subject Descriptors (according to ACM CCS): 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

1. Introduction

In structure-texture decomposition, an image is decomposed as $I = S + T$, where I, S, and T represent the input image, structure elements, and texture details, respectively. In general, the structure-texture decomposition problem is formulated as finding an appropriate (or latent) structure S by suppressing texture details T in the input image I. Due to its usefulness in a broad range of image processing applications, several methods have been proposed to handle this problem, which in general can be divided into two categories: optimization-based [ROF92, Wei06, XLXJ11, XYXJ12, HBZW14] and filtering-based approaches [PHK11, KEE13, CLKL14, ZSXJ14].

Optimization-based approaches [ROF92, XLXJ11, XYXJ12, HBZW14] globally suppress the oscillating patterns induced from T, while guessing the structure image S as similar as possible to the input image I. Although they obtain high-quality results, these methods are comparably complex and cannot easily be parallelized, thus not allowing the algorithm to handle large images and used in interactive applications. Filtering-based algorithms [KEE13, CLKL14, ZSXJ14] try to design effective filter kernels to suppress T. Previous filtering approaches, however, often fail to accurately detect S for structure edges and corners.

In this paper, we propose a novel filtering-based method for structure-texture decomposition that generates high-quality results comparable to those of optimization-based approaches. Unlike most previous filtering-based approaches, we formulate the problem as $\nabla I = \nabla S + \nabla T$ in the gradient domain. Our approach directly manipulates gradients of the input image to filter out textures from the input. In this sense, our goal is to carefully design a gradient domain solution for structure-texture decomposition, where the desirable ingredients are i) structure-aware gradient smoothing, which suppresses ∇T while preserving ∇S, and ii) re-

(a) input image (b) filtering result (c) detail enhancement (d) close ups

Figure 1: *Structure-texture decomposition result. From the input image (a), textures are first filtered (b) and then enhanced (c). Our filtering result precisely preserves structure edges, and no artifact is introduced during the enhancement, as shown in (d).*

construction of the filtered image from the smoothed gradients, which not only preserves ∇S but also reflects the overall shape of the input I.

Our contribution is twofold. First, we define a new type of gradient operator, called the *interval gradient*, denoted by $\nabla_\Omega(\cdot)$, which is designed to distinguish ∇T from ∇S in the spatial support Ω. We propose a gradient rescaling method based on the interval gradient to suppress ∇T while preserving ∇S within the signal (Sec. 3).

Second, motivated by the guided image filter [HST10], we propose a novel gradient-based image filtering method that is suitable for structure-texture decomposition (Sec. 4). It inherits the characteristics of the guided image filter. The filtering results resemble the input images and at the same time preserve strong edges and gradients precisely, while detailed textures and repeated patterns are removed. In addition, the proposed method is composed of 1D local filters, similarly to [GO11], allowing extreme acceleration with modern GPUs. As a result, our method can decompose textures from images effectively and accurately, allowing the decomposed images to be utilized in various applications that require high-quality decomposition results, as shown in Fig. 1.

2. Related Work

2.1. Structure-texture image decomposition

Among the filtering-based approaches, the bilateral filter [TM98] is a representative kernel-based edge-preserving filter. Its range weight function allows data point pairs across edges to be clearly distinguished and thus not mixed. The guided filter [HST10] performs local linear transforms of a guidance image, a strategy that does not lead to gradient distortions near edges. The domain transform method [GO11] delivers a significant speedup in 2D filtering by reducing the dimensionality to one. These methods can produce high quality edge-preserving smoothing results but cannot easily remove strong textures.

Several methods have been proposed for removing strong textures within images. Subr *et al.* [SSD09] computed envelopes defined from local extrema and smoothed out texture oscillations by averaging the envelops. The histogram filter [KS10], median filter [Wei06, MHW*13, ZXJ14], and diffusion-based approaches [Wei98, VPS03] are capable of filtering textures to some extent, unless the textures contain large oscillating signals. Karacan et al. [KEE13] proposed a patch-based texture removal algorithm that uses similarity measures based on regional covariance. The bilateral texture filter [CLKL14] uses joint bilateral filtering with a guidance image generated via a patch shift mechanism. The rolling guidance filter [ZSXJ14] uses a Gaussian-blurred image as a guidance image in joint bilateral filtering to eliminate only the image structures that are smaller than a specific scale.

The total variation (TV) method [ROF92] effectively suppresses textures of arbitrary shapes by enforcing TV regularization constraints on the image. The weighted least squares (WLS) method [FFLS08] solves a large linear system to provide progressive/multiscale image coarsening and decomposition. L_0 gradient minimization [XLXJ11] globally optimizes the quality of filtering by controlling the number of non-zero gradients in the image. Xu et al. [XYXJ12] introduced relative total variation (RTV) which allows accurate identification and removal of texture regions. As compared to filtering-based approaches, optimization-based techniques [FFLS08, XLXJ11] are able to produce high-quality results; however, their computation costs are higher in general, even when the multigrid method [BFGS03, GWL*03] or a fast global solver based on separable application of a sequence of 1D subsystems [MCL*14] is applied.

2.2. Gradient domain image processing

Gradient domain analysis has been successfully utilized in various image processing applications, such as editing, composition, HDR image compression [FLW02, PGB03, Aga07, FHL*09, BZCC10]. We refer the reader to an excellent tutorial [AR07] for details. From an input image, the gradient information of each pixel is first computed by measuring intensity differences between the pixel and its neighborhood. Then, these gradients are manipulated according to application-specific purposes, while their local structures are preserved. The resultant image is then estimated based on the manipulated gradients through optimization, e.g., using the Poisson solver. Our local filtering approach is based on gradient domain processing specifically designed to handle the structure-texture decomposition problem.

Our approach has a connection to Retinex-based approaches for intrinsic image decomposition [TFA05, GJAF09, BST*14], in that both consider gradient-domain image decomposition. In the Retinex theory [LM71], an image could be decomposed into illumination and reflectance parts in the log-scale gradient domain, based on the assumption that illumination varies gradually but reflectance does abruptly. Retinex-based approaches use trained classifiers [TFA05] or global optimization [BST*14] to achieve illumination-reflectance decomposition in the gradient domain. Our approach tackles structure-texture decomposition in the gradient domain by defining interval gradients to handle complicated texture patterns.

3. Interval Gradient for Gradient Filtering

In structure-texture decomposition, an input image is decomposed into a structure image and a texture image. The structure part contains non-repeating edges and smooth shadings, while the texture part contains oscillating patterns and noise. The goal of the texture filtering is to find a structure image as similar as possible to the input image while removing textures. In this section, we first define the local characteristics of structure and texture in the 1D domain. Then we present our interval gradient operator, which is able to distinguish structure and texture from a given 1D signal.

3.1. Structure and texture in 1D

In a 1D signal I, for a local window Ω_p centered at a pixel p, we define structure S as non-repeating salient edges that are smoothly increasing or decreasing signals within the window. Texture T is then defined as local oscillations added to the structure so that $I = S + T$.

Given the definition of a structure, we define that Ω_p is texture-free if and only if the signal is only increasing or decreasing but never oscillates within Ω_p. The signal can vary smoothly (smooth shading) or abruptly (structure edge). Although imperfect (*e.g.,* the signal can smoothly go up and then down without containing texture), this definition is effective in most cases. We can then determine whether a 1D signal I is texture-free by computing a measure based on this definition. Note that the RTV metric defined in [XYXJ12] is also minimized when the signal is increasing or decreasing. In this paper we present a novel operator, the interval gradient, to determine whether a signal is texture-free.

3.2. Definition of interval gradient

In a 1D discrete signal I, for a pixel p, one common gradient operator is the forward differentiation, defined as:

$$(\nabla I)_p = I_{p+1} - I_p. \tag{1}$$

It measures the difference between two adjacent signal values. In contrast, our interval gradient for pixel p is defined as

$$(\nabla_\Omega I)_p = g^r_\sigma(I_p) - g^l_\sigma(I_p), \tag{2}$$

where g^r_σ and g^l_σ respectively represent left and right clipped 1D Gaussian filter functions defined by

$$\begin{aligned} g^r_\sigma(I_p) &= \frac{1}{k_r} \sum_{n\in\Omega(p)} w_\sigma(n-p-1)\, I_n, \\ g^l_\sigma(I_p) &= \frac{1}{k_l} \sum_{n\in\Omega(p)} w_\sigma(p-n)\, I_n. \end{aligned} \tag{3}$$

w_σ is the clipped exponential weighting function with a scale parameter σ:

$$w_\sigma(x) = \begin{cases} \exp\left(-\frac{x^2}{2\sigma^2}\right) & \text{if } x \ge 0 \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

and k_r and k_l are normalizing coefficients defined as:

$$k_r = \sum_{n\in\Omega(p)} w_\sigma(n-p-1) \quad \text{and} \quad k_l = \sum_{n\in\Omega(p)} w_\sigma(p-n). \tag{5}$$

The kernel shape of the interval gradient is shown in Fig. 2.

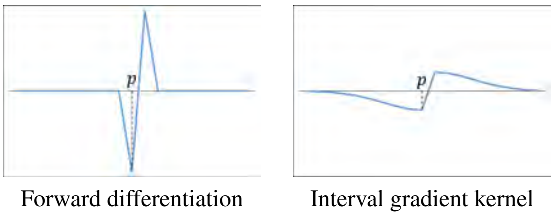


Figure 2: *Forward differentiation and interval gradient kernels. Left: Conventional forward difference kernel; right: our interval gradient kernel with* $\sigma = 3$.

Unlike forward differentiation, the interval gradient measures the difference between the weighted averages of the left and right parts of the signal around a pixel. It is also different from gradients of the smoothed signal, as discussed in Sec. 3.3.

For the shape of the interval gradient kernel, any type of kernel, such as concatenation of two box filters or derivatives of Gaussian, can be used if it is able to detect the average statistics of the left and right subregions of a pixel. Different kernels have different properties when smoothing textures, as discussed in Sec. 5. Our experiments showed that the proposed kernel shape in Eq. (4) preserves sharp features while

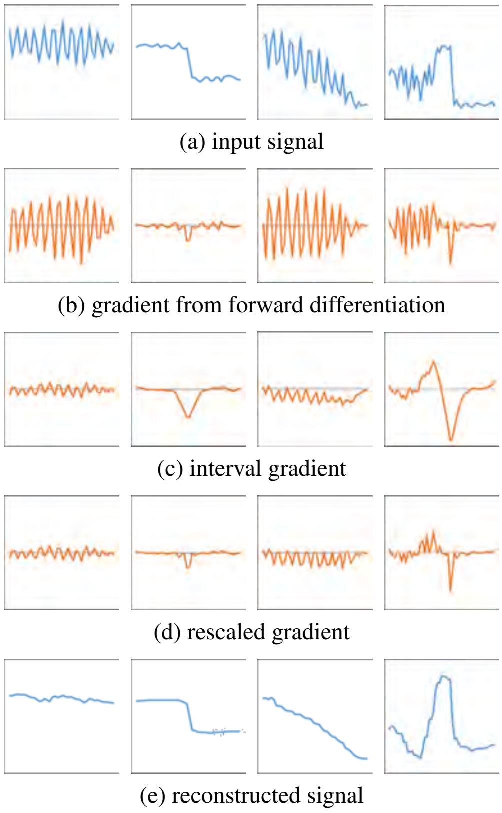


Figure 3: *Various types of signals and gradients. From left to right: texture, structure edge, texture with shading, and noisy valley near a structure edge.*

smoothing textures. It is used to generate all examples in this paper unless otherwise stated.

3.3. Characteristics of interval gradient

The most important characteristic of the interval gradient is that for a local window Ω_p, $|(\nabla I)_p| \leq |(\nabla_\Omega I)_p|$ if the window is texture-free. As defined in Sec. 3.1, Ω_p is texture-free if the signal is increasing or decreasing. For the increasing case, $I_{p+1} \leq g^r_\sigma(I_p)$ and $I_p \geq g^l_\sigma(I_p)$, making the above inequality true, and similar argument holds for the decreasing case. In addition, the sign of $(\nabla_\Omega I)_p$ captures the overall increasing/decreasing tendency of the signal within Ω_p.

These characteristics can also be shown by considering the interval gradient kernel in the gradient domain. We can show that a symmetric interval gradient kernel is equivalent to an unnormalized smoothing kernel in the gradient domain, where the sum of the kernel weights is greater than 1 (see Appendix). For a structure element p, $(\nabla_\Omega I)_p$ becomes bigger than $(\nabla I)_p$, since the smoothing kernel magnifies the gradient. For a texture region, $(\nabla_\Omega I)_p$ becomes smaller than $(\nabla I)_p$ as oscillating gradients with different signs cancel each other. The sign of $(\nabla_\Omega I)_p$ is the same as that of the weighted average of gradients, since scaling does not affect the sign.

Case by case analysis Four example cases are shown in Fig. 3. For a local window with texture, oscillations are canceled out because of the smoothing operators g^r_σ and g^l_σ and $|(\nabla_\Omega I)_p|$ becomes much smaller than $|(\nabla_I)_p|$. For a strong structure edge, the shape of interval gradients resembles that of the gradient of the smoothed edge. For a local window mixed with shading and textures, the magnitudes of interval gradients are smaller than those of pixel gradients but their signs indicate the overall increasing/decreasing tendency of the signal within the window. The rightmost column shows a noisy valley attached to a structure edge. In the noisy valley, similar to a mixture of shading and texture, $|(\nabla_\Omega I)_p|$ is smaller than $|(\nabla_I)_p|$, although the sign of $|(\nabla_\Omega I)_p|$ changes with overall shape of the valley. However, as the signal moves into the structure edge, the magnitudes of interval gradients increase and become larger than those of pixel gradients.

Validation To validate the property of our interval gradient, we conducted the following experiment. We filtered 200 images in the dataset given in [XYXJ12], using previous texture filtering methods [XYXJ12, ZSXJ14, CLKL14]. Then, we measured the ratios between the magnitudes of the original and interval gradients ($|(\nabla_\Omega I)_p \ / \ (\nabla I)_p|$) for all pixels in the two images, before and after filtering. As shown in Fig. 4, most ratio values are near to or greater than 1 for the filtered images, while the original images have many small ratio values because of high contrast textures and noise.

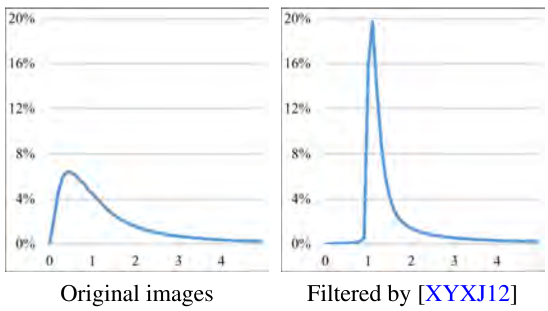


Figure 4: *(x-axis) gradient ratio, $|(\nabla_\Omega I)_p \ / \ (\nabla I)_p|$; (y-axis) percentage of pixels. The total number of pixels is around 100M. The image gradients along the horizontal direction were used for the comparison, and the result with the vertical gradients is similar. After texture filtering, almost all pixels have gradient ratios near to or greater than 1.*

Using interval gradients we can determine whether a local window of a given signal contains texture. In addition, we can determine the overall increasing/decreasing tendency of the signal. We present our gradient rescaling algorithm based on these characteristics in the following subsection.

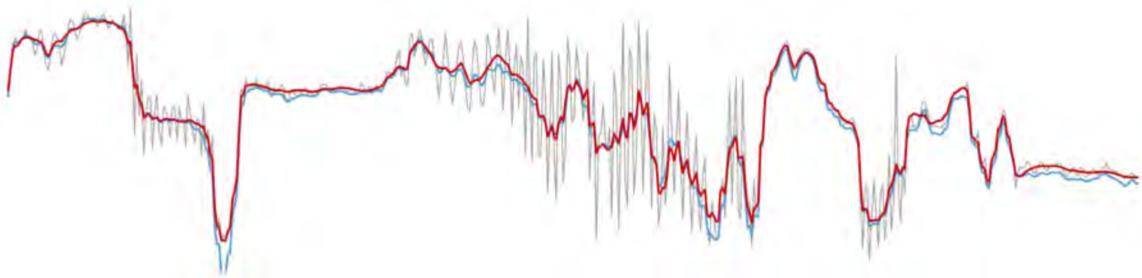


Figure 5: *Gradient-based guided filtering example. (gray) input signal; (blue) signal reconstructed directly from rescaled gradients; (red) signal after guided filtering. In the right part of the signal, the directly reconstructed signal becomes deviated from the original. This deviation error is adjusted by using the guided filtering, as shown by the red curve.*

3.4. Gradient rescaling with interval gradients

In order to produce a texture-free signal from an input signal I, the gradients within texture regions should be suppressed. Furthermore, the signal should be either increasing or decreasing for all local windows Ω_p. With these objectives, we use the following equation to rescale the gradients of the input signal with the corresponding interval gradients:

$$(\nabla' I)_p = \begin{cases} (\nabla I)_p \cdot w_p & \text{if } \operatorname{sign}((\nabla I)_p) = \operatorname{sign}((\nabla_\Omega I)_p) \\ 0 & \text{otherwise}, \end{cases} \tag{6}$$

where $(\nabla' I)_p$ represents the rescaled gradient, and w_p is the rescaling weight:

$$w_p = \min\left(1, \frac{|(\nabla_\Omega I)_p| + \varepsilon_s}{|(\nabla I)_p| + \varepsilon_s}\right), \tag{7}$$

where ε_s is a small constant to prevent numerical instability. Too small values of ε_s would make the algorithm sensitive to noise, introducing unwanted artifacts to filtering results. The sensitivity to noise can be reduced by increasing ε_s but textures could not be completely filtered if ε_s is too big. We set $\varepsilon_s = 10^{-4}$ in our implementation.

For structure edges and smoothly varying regions, gradients are not changed because $|(\nabla_\Omega I)_p| \geq |(\nabla I)_p|$, making w_p be 1. On the other hand, for textured regions with oscillating patterns and noise, $|(\nabla_\Omega I)_p| < |(\nabla I)_p|$ and the gradients are suppressed with w_p less than 1. In addition, $(\nabla I)_p$ is set to zero if the signs of the original and interval gradients are different. As our goal is to make a signal locally monotonic, we make $(\nabla I)_p$ follow the local increasing/decreasing tendency, which is estimated by the sign of $(\nabla_\Omega I)_p$, by setting $(\nabla I)_p$ to zero if it does not agree with the tendency.

As a result, the filtered gradients satisfy the first property specified in Sec. 1, preserving ∇S and suppressing ∇T. However, the textures may not be fully filtered for some regions, and stair-like effect can occur within a region containing both texture and smoothly varying shading, as shown in Fig. 3e. These remaining issues are handled in the following section.

4. Texture Filtering with Interval Gradient

4.1. 1D texture filtering

After gradient rescaling, we can reconstruct the filtered signal by simply accumulating the rescaled gradients. The accumulation result, however, can still contain small unfiltered oscillations and deviate from the original signal, since we have rescaled gradients locally (Figs. 3e and 5). We need a method for rectifying the reconstructed signal to remove remaining oscillations and to remedy the deviations from the original signal. Guided filtering [HST10] provides an appropriate solution for this task, as it filters the input signal to remove small noise and match the target signal. Guided filtering is also known to preserve edges and corners without introducing gradient distortions and oversharpened edges.

To obtain the filtering result of the input signal I, we first construct a temporary signal R from the rescaled gradients via accumulation:

$$R_p = \sum_{k=0}^{p-1} I_0 + (\nabla' I)_k,\ p \in \{0, 1, \ldots, N_p\}, \tag{8}$$

where I_0 is the leftmost value of I and N_p is the number of pixels in I. Then, for each reconstructed signal value R_p, the 1D guided filtering process seeks the best linear transform coefficients a_p and b_p that minimize:

$$\operatorname*{arg\,min}_{a_p, b_p} \sum_{n \in N(p)} w_n \left((a_p R_n + b_p - I_n)^2 + \varepsilon a_p^2 \right), \tag{9}$$

where w_n is the Gaussian weight with the scale parameter σ in Eq. (4), given as:

$$w_n = \exp\left(-\frac{(n-p)^2}{2\sigma^2}\right), \tag{10}$$

and ε is the smoothness parameter. The effects of the parameters σ and ε are discussed in Sec. 5. Closed form solutions

Algorithm 1 2D image filtering

Input: image I
Output: texture filtered image S
  while Eq. (14) is false **do**
    $(\nabla'_x I) \leftarrow$ rescaled gradients along x-axis ▷ Eq. (6)
    $(\nabla'_y I) \leftarrow$ rescaled gradients along y-axis
    for $i := 0 \ldots (N_d - 1)$ **do**
      $\sigma_i \leftarrow \sigma \cdot \sqrt{3}\frac{2^{N-i}}{\sqrt{4^N-1}}$ ▷ Adapted from [GO11]
      $S_x \leftarrow$ 1D filtering result along x-axis ▷ Eqs. (11)-(13)
      $I \leftarrow S_x$
      $S_y \leftarrow$ 1D filtering result along y-axis
      $I \leftarrow S_y$
    end for
  end while
  $S \leftarrow I$

exist for both a_p and b_p, as given in [HST10]:

$$a_p = \frac{g_\sigma\left((RI)_p\right) - g_\sigma\left(R_p\right) g_\sigma\left(I_p\right)}{g_\sigma\left(R_p^2\right) - g_\sigma\left(R_p\right)^2 + \varepsilon} \tag{11}$$

$$b_p = g_\sigma\left(I_p\right) - a_p \cdot g_\sigma\left(R_p\right), \tag{12}$$

where $g_\sigma(\cdot)$ represents 1D Gaussian smoothing with σ and RI is the elementwise multiplication of R and I. Then the filtered result S can be computed as:

$$S_p = \bar{a}_p R_p + \bar{b}_p, \tag{13}$$

where $\bar{a}_p = g_\sigma(a_p)$ and $\bar{b}_p = g_\sigma(b_p)$ are Gaussian smoothed coefficients. Fig. 5 shows an example of the guided filtering process; structural differences from the original as well as the remaining oscillations in the temporary signal are removed in the filtering.

4.2. 2D Texture Filtering

So far, all the algorithm components, the interval gradient, gradient rescaling, and guided filtering, have been defined and applied in the 1D domain. To apply these components to a 2D image, we adopt the approach of alternating 1D filtering operations in the x and y directions, which was used for domain transform filtering [GO11]. We first compute the rescaled gradients in both x and y directions from the input image. At each iteration, we apply 1D filtering to the image along the x direction using Eqs. (11)-(13). The filtered result is then set as the input and the 1D filtering along the y direction is applied with the rescaled gradients computed at the beginning. As in domain transform filtering [GO11], these alternating steps are performed several times with decreasing spatial support σ_i. By taking only 1D operations, our 2D filtering can be performed efficiently.

A possible alternative approach to handle 2D images using our algorithm components would be to compute filtered gradients in the x and y directions using interval gradient and gradient rescaling, and then apply the Poisson reconstruction method [PGB03] to the filtered gradients. However, as shown in Fig. 6, such an approach is not very successful in our case, as the Poisson solver usually fails in the presence of noise and outliers, often producing over-smoothed results [AR07]. Remaining small oscillations from textures in the filtered gradients act as noise that does not favor a desirable output image, and a Poisson solver could return the textures patterns into the output. Moreover, the over-smoothing artifact of a Poisson solver blurs structural boundaries in the output.

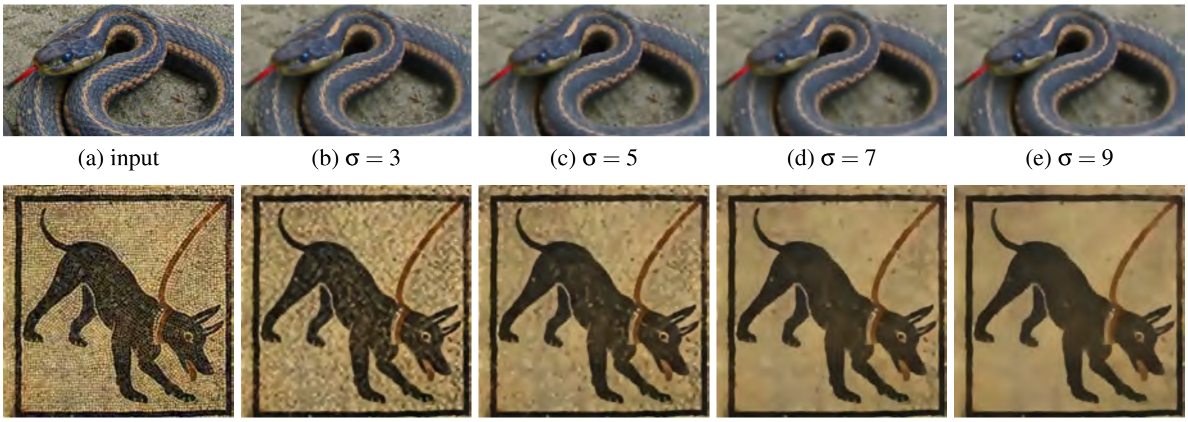
(a) input (b) Poisson (c) ours

Figure 6: *Comparison between Poisson reconstruction and our approach. Due to remaining oscillations in the filtered gradients, Poisson reconstruction generates an unpleasing result. With our guided filtering, such remaining oscillations are filtered out while structure edges are preserved.*

Filtering iterations Iterative filtering is necessary to fully remove high contrast textures from images [CLKL14]. To obtain the final result, the entire process, consisting of interval gradient computation, gradient smoothing, and iterative 1D filtering, is repeated several times, where the result obtained from the previous iteration is used as the input of the current iteration. For a stopping condition, we check whether the filtering iteration has been converged. As textures are removed, interval gradients become bigger, and the rescaling ratio in Eq. (7) becomes 1 for most pixels. Utilizing this, we measure the difference of rescaling ratios between the current and previous iterations. The iteration is stopped when the difference is less than a given threshold δ_t, so that

$$\max\left(\frac{1}{N}\sum_p\left(w_{p_x}^{t-1} - w_{p_x}^t\right)^2, \frac{1}{N}\sum_p\left(w_{p_y}^{t-1} - w_{p_y}^t\right)^2\right) < \delta_t, \tag{14}$$

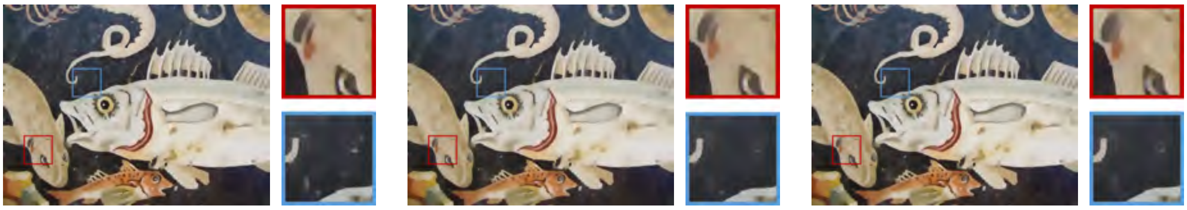
where $w_{p_x}^t$ and $w_{p_y}^t$ are the rescaling ratios for the x- and y-directional gradients of a pixel p at iteration t, respectively. Algorithm 1 summarizes the entire process of our 2D filtering method.



(a) input (b) $\sigma = 3$ (c) $\sigma = 5$ (d) $\sigma = 7$ (e) $\sigma = 9$

(f) input (g) $\varepsilon = 0.005^2$ (h) $\varepsilon = 0.01^2$ (i) $\varepsilon = 0.015^2$ (j) $\varepsilon = 0.02^2$

Figure 7: *Results with varying parameters. (top) bigger scale textures are removed as σ increases; (bottom) smoother results are obtained with bigger ε.*



(a) clipped Gaussian (b) box shaped (c) derivative of Gaussian

Figure 8: *Results using three different interval gradient kernels. The clipped Gaussian kernel preserves sharp features better, while the derivative of Gaussian kernel is able to effectively smooth high contrast textures. A box shaped kernel smooths textures moderately. All the parameters, except for the interval gradient kernel shape, are set the same for the three cases.*

5. Experimental Results

Parameters As in previous structure-preserving filtering methods, our method has two main control parameters, σ and ε. Parameter σ controls the scale of textures to be removed, and ε controls the smoothness of the result. In our experiments, we used $\sigma \in [2,\ 5]$ and $\varepsilon \in [0.01^2,\ 0.03^2]$. Fig. 7 shows filtering examples to demonstrate the effects of the parameters. For the convergence threshold δ_t in Eq. (14), too small values would bring in unnecessary filtering iterations and oversmooth the image, while bigger values could cause the filtering process to stop too early. We used a fixed value of 0.05^2 throughout our experiments. We found that more iterations (usually 5-8) are required to filter high contrast textures, while 3-5 iterations suffice for most cases.

Interval gradient kernel shapes Various kernels can be applied to compute interval gradients provided that they capture the signal difference between the two intervals in the left and right of a pixel. We tested three different kernels, clipped Gaussian, box shaped, and derivative of Gaussian (DoG) (Fig. 15). In principle, sharp features such as corners are preserved better if more weights are given near to the kernel center. On the other hand, high contrast textures can be effectively removed using the DoG kernel (Fig. 8). Results with the box shaped kernel show slightly inferior quality, but can be efficiently computed with integral images. In our experiments, we used clipped Gaussian kernels for most images, unless stated otherwise.

Handling color images For filtering color images, we use the gradient sums of color channels in the gradient rescaling step (Eqs. (6) and (7)), i.e.,

$$w_p = \min\left(1, \frac{\sum_{c\in\{r,g,b\}}\left|(\nabla_\Omega I^c)_p\right| + \varepsilon_s}{\sum_{c\in\{r,g,b\}}\left|(\nabla I^c)_p\right| + \varepsilon_s}\right). \qquad (15)$$

In the guided filtering process, the color bleeding artifact may occur if each channel is filtered independently. Guided filtering for color images is described in [HST10]: however, it requires a 3-by-3 matrix inversion for each pixel and that the target image be single channel. Instead, we use a simple approach to prevent the color bleeding artifact. Color bleed-

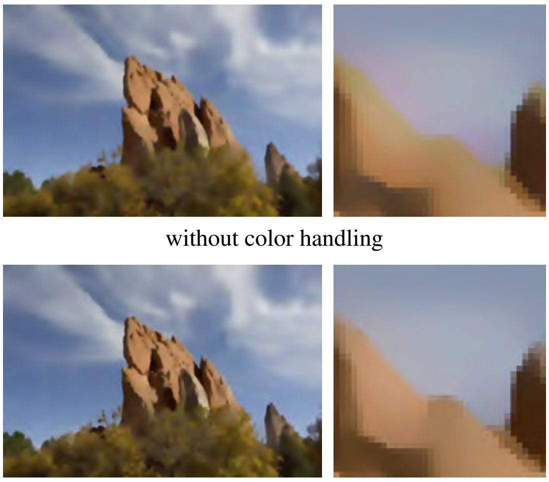


Figure 9: *Color handling example. In the top row, guided filtering is applied separately for each channel, and the red color channel is over-smoothed, causing color bleeding. Color bleeding can be prevented using Eqs. (16) and (17), as shown in the bottom row.*

ing happens when a_p in Eq. (11) of a specific color channel becomes very small as compared to those of the other channels during the filtering. In this case, the color channel is blurred while other channels are not. To prevent this, for each pixel we compute the minimum threshold of a_p as

$$a_p^{max} = \min\left(1, \max_{c\in\{r,g,b\}} a_p^c\right), \quad (16)$$

where a_p^c represents the scaling factor for a color channel c. The value of a_p^{max} is clamped by 1 to avoid unnecessary scale-up of a channel value. Then, we adjust a_p^c of each color channel using the threshold a_p^{max} as

$$a_p^c = \max\left(a_p^c,\, a_p^{max}\right). \quad (17)$$

Fig. 9 shows an example of color handling.

Timing data Table 1 shows the average computation time. Although our method is not significantly faster than previous approaches when running on a CPU, it is highly parallelizable because of its local nature. In addition, our method consists only of 1D filters, allowing the fast shared memory structure of CUDA to be used [NBGS08]. Our implementation using CUDA C++ can run around 400 times faster than the CPU version.

Comparison We compared our filtering method with a few previous approaches [XYXJ12,CLKL14,ZSXJ14]. For RTV [XYXJ12] and rolling guidance filter (RGF) [ZSXJ14], we used the authors' implementations publicly available on internet. For bilateral texture filtering (BTF) [CLKL14], we used our own implementation. Fig. 10 shows comparison examples. Similar to other methods, our method preserves the structure elements while textures are filtered out. Compared to BTF and RGF, our method preserves small features and edge gradients better. Our results show comparable quality to those of RTV with significantly less processing time, as shown in Table 1. As an optimization-based method, RTV flattens shading variations in the input image, and shading information can be lost in the filtering results although they appear sharper and cleaner. In contrast, our method preserves overall shading of images and retains important structure edge gradients. Additional filtering results of our method are shown in Fig. 11 and the supplementary material. Note that the original gradients of the structure edges in the input images are preserved well by our method and this may cause some structure edges to appear little blurry in our filtering results.

Table 1: *Timing data for a grayscale image of 600×800 pixels with a single filtering iteration, measured using our MATLAB (CPU) and CUDA C++ (GPU) implementations on a PC with Intel Core i7 CPU, 12GB RAM, and NVIDIA GeForce GTX 980 graphic card running Windows 8.1.*

Component	$\sigma = 3$	$\sigma = 5$	$\sigma = 7$
	CPU/GPU	CPU/GPU	CPU/GPU
Comp. ($\nabla' I$)	80ms/0.6ms	90ms/1ms	95ms/1.5ms
Comp. S	390ms/7.5ms	530ms/11ms	545ms/15ms
Total	470ms/8.1ms	620ms/12ms	640ms/16.5ms

Applications Because of the structure-texture decomposition property, the most intuitive application of our method would be detail enhancement. In enhancing details of images, it is crucial to precisely preserve structure edge gradients to achieve high-quality results, and this requirement prohibits many filtering-based approaches from being effectively used in detail enhancement. Our method does not distort the gradients of structure edges, even after removal of highly contrasting textures. With our method, high quality detail enhancement results can be obtained within seconds. Fig. 12 shows an example; more results can be seen in the supplementary material. As another application, Fig. 13 shows an example where our method is used for inverse halftoning. Although we do not provide specific examples, our method can also be used for other applications, such as noise removal, tone mapping, and texture replacement.

6. Conclusion

We proposed a novel filtering based structure-texture decomposition method. Rather than directly filtering image colors, our method manipulates image gradients to produce high quality filtering results with fewer artifacts than previous filtering-based approaches. We proposed a novel gra-

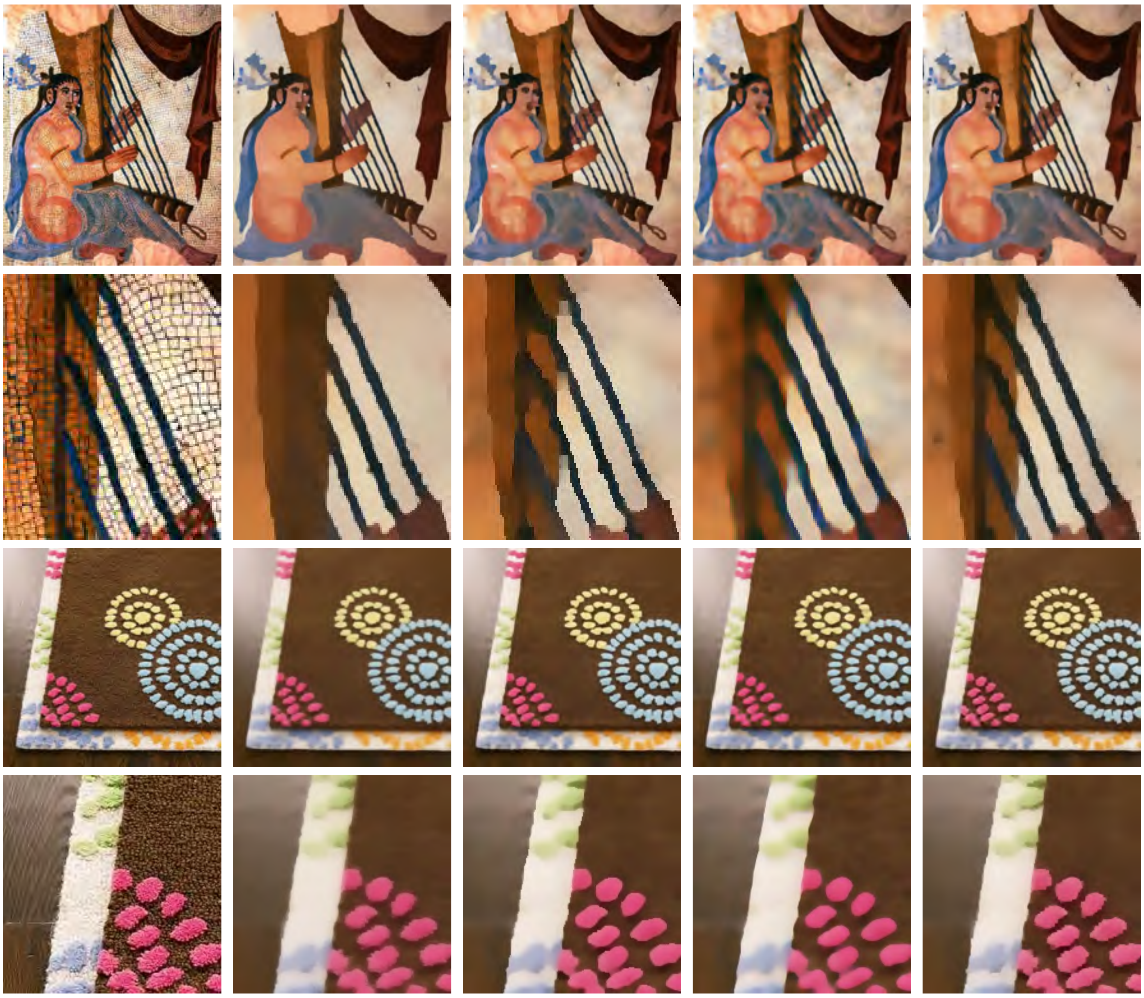


Figure 10: *Comparison of filtering results. Input images contain strong textures as well as sharp edges and smoothly varying shadings. As compared to other methods, our method removes textures while preserving image structures and shading better.*

dient operator, the *interval gradient*, which is a useful tool for effectively distinguishing the textures from the structures of an image. With our interval gradient operator and gradient guided image filtering, we obtained state-of-the-art structure-texture decomposition results, which can be utilized in various applications.

Limitations Similarly to previous approaches, our method requires fixed scale parameters and cannot handle textures with largely varying scales (Fig. 14a). As a local 1D filtering approach, our method cannot easily handle strong and irregular texture patterns. Since such textures can be considered as unique structures in local windows, sparse patterns may not be filtered (Fig. 14b). Our method attempts to preserve 1D increasing or decreasing signals as far as possible, and filtering may not be accurate around some complex texture boundaries (Fig. 14c). Lastly, our definition of the texture does not consider semantic information of the scene, and sometimes filtering results could be perceived nonintuitive in terms of the semantic information (Fig. 14d).

Future work Our possible future work would be to address these limitations. Ham *et al.* [HCP15] recently proposed a novel optimization based filtering method that can handle different scales of textures in the input image. It would also be interesting to find interactive applications based on our structure-texture decomposition method.

Acknowledgements This work was supported by Institute for Information & communications Technology Promotion (IITP) grant (R0126-16-1078) and the National Research Foundation of Korea (NRF) grant (NRF-

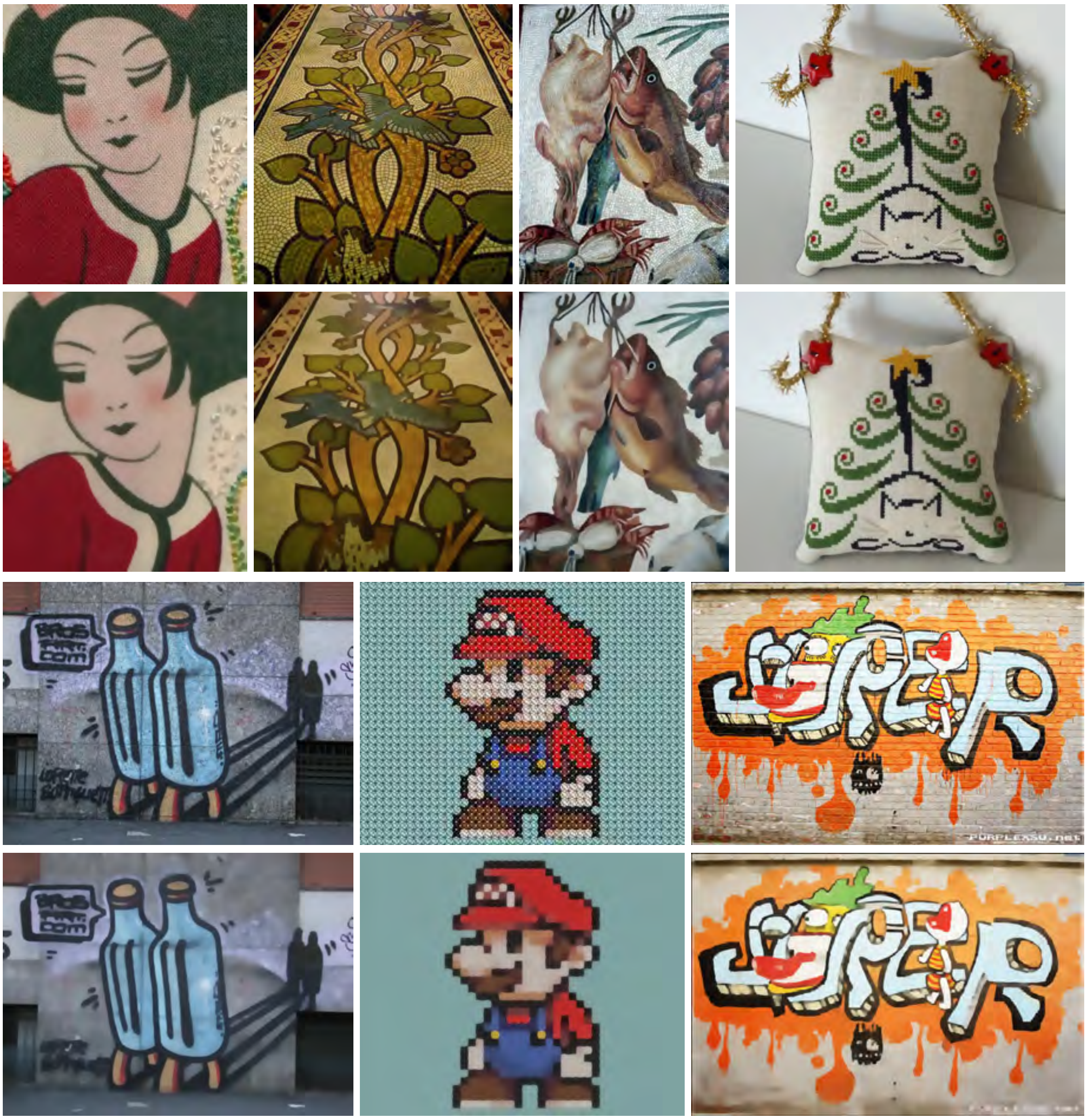


Figure 11: *Additional results of our structure-texture decomposition. (top) input images; (bottom) filtering results. The original gradients of structure edges in the input images are preserved well by our method, and it may make some structure edges appear a little blurry in the filtering results.*

2014R1A2A1A11052779) both funded by the Korea government (MSIP).

References

[Aga07] AGARWALA A.: Efficient gradient-domain compositing using quadtrees. *ACM Trans. Graphics 26*, 3 (2007), Article No. 94. 2

[AR07] AGRAWAL A., RASKAR R.: Gradient domain manipulation techniques in vision and graphics. In *Proc. ICCV 2007* (2007), Tutorials. 2, 6

[BFGS03] BOLZ J., FARMER I., GRINSPUN E., SCHRÖDER P.: Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. *ACM Trans. Graphics (Proc. SIGGRAPH 2003) 22*, 3 (2003), 917–924. 2

[BST*14] BONNEEL N., SUNKAVALLI K., TOMPKIN J., SUN D., PARIS S., PFISTER H.: Interactive intrinsic video editing. *ACM Trans. Graphics (Proc. SIGGRAPH 2014) 33*, 6 (2014), Article No. 197. 3

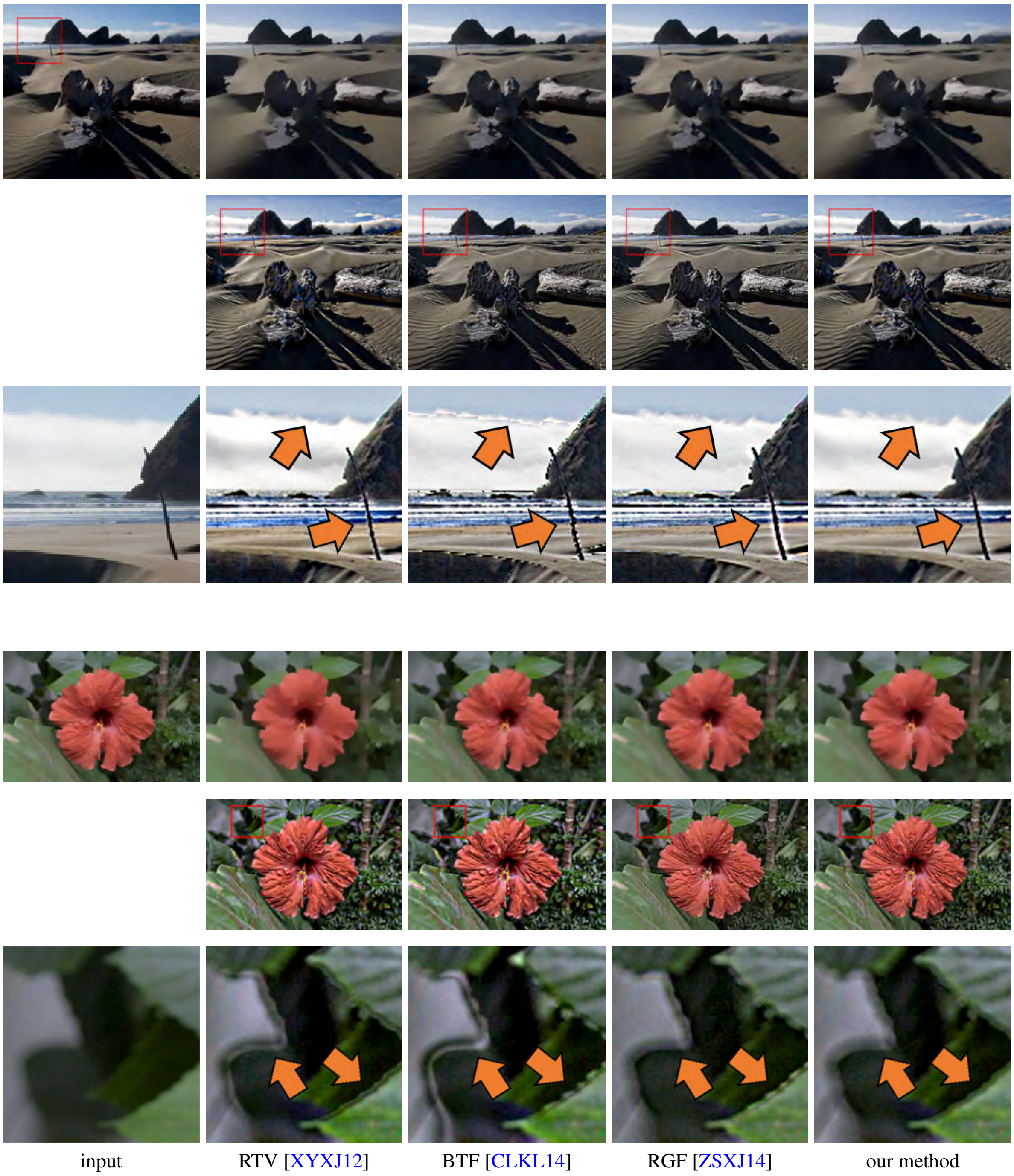


Figure 12: *Comparison of detail enhancement results. (top) input and texture-filtered results; (middle) detail enhancement results; (bottom) close-ups of detail enhanced images. Our method suffers less from gradient reversal artifacts such as halos and noise. Parameters: (top example) [XYXJ12] ($\lambda = 0.01$, $\sigma = 3$), [CLKL14] ($k = 2$, $n_{iter} = 2$), [ZSXJ14] ($\sigma_s = 3$, $\sigma_r = 0.1$, $n_{iter} = 5$), and our method ($\sigma = 2.5$, $\varepsilon = 0.015^2$); (bottom example) [XYXJ12] ($\lambda = 0.01$, $\sigma = 3$), [CLKL14] ($k = 3$, $n_{iter} = 2$), [ZSXJ14] ($\sigma_s = 4$, $\sigma_r = 0.1$, $n_{iter} = 5$), and our method ($\sigma = 2.5$, $\varepsilon = 0.02^2$).*

input [XYXJ12] [ZSXJ14] our method

Figure 13: *Inverse halftoning example. Our method can effectively remove halftoning artifacts from the input image, while preserving shading and strong edges.*

[BZCC10] BHAT P., ZITNICK C. L., COHEN M., CURLESS B.: Gradientshop: A gradient-domain optimization framework for image and video filtering. *ACM Trans. Graphics 29*, 2 (2010), Article No. 10. 2

[CLKL14] CHO H., LEE H., KANG H., LEE S.: Bilateral texture filtering. *ACM Trans. Graphics 33*, 4 (2014), Article No. 128. 1, 2, 4, 6, 8, 9, 11

[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. 2

[FHL*09] FARBMAN Z., HOFFER G., LIPMAN Y., COHEN-OR D., LISCHINSKI D.: Coordinates for instant image cloning. *ACM Trans. Graphics 28*, 3 (2009), Article No. 67. 2

[FLW02] FATTAL R., LISCHINSKI D., WERMAN M.: Gradient domain high dynamic range compression. *ACM Trans. Graphics 21*, 3 (2002), 249 – 256. 2

[GJAF09] GROSSE R., JOHNSON M. K., ADELSON E. H., FREEMAN W. T.: Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *Proc. ICCV 2009* (2009), pp. 2335–2342. 3

[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

[GWL*03] GOODNIGHT N., WOOLLEY C., LEWIN G., LUEBKE D., HUMPHREYS G.: A multigrid solver for boundary value problems using programmable graphics hardware. In *Proc. Graphics Hardware* (2003), pp. 102–111. 2

[HBZW14] HUA M., BIE X., ZHANG M., WANG W.: Edge-aware gradient domain optimization framework for image filtering by local propagation. In *Proc. CVPR* (2014), pp. 2838–2845. 1

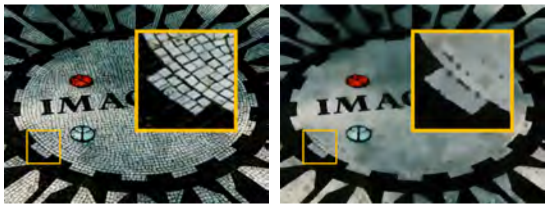
[HCP15] HAM B., CHO M., PONCE J.: Robust image filtering using joint static and dynamic guidance. In *Proc. CVPR* (2015), p. accepted for publication. 9

[HST10] HE K., SUN J., TANG X.: Guided image filtering. In *Proc. ECCV 2010* (2010), pp. 1–14. 2, 5, 6, 7

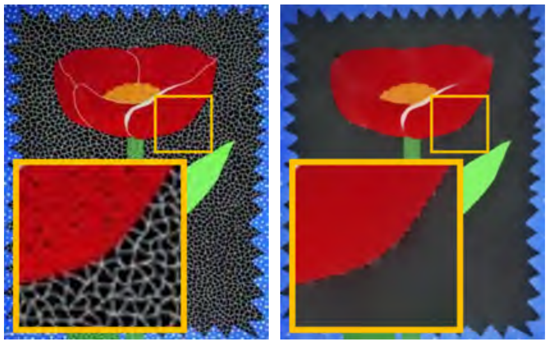
[KEE13] KARACAN L., ERDEM E., ERDEM A.: Structure-preserving image smoothing via region covariances. *ACM Trans. Graphics 32*, 6 (2013), 176:1–176:11. 1, 2

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

(a) varying texture scales



(b) strong irregular textures



(c) complex texture boundaries

(d) textures with semantics

Figure 14: *Limitation cases. (a) our method cannot easily handle textures with largely varying scales; (b) some strong, irregular textures may not be filtered since they can be considered as structures in local windows; (c) our method attempts to preserve monotonic signals in 1D, causing some complex texture boundaries are not accurately filtered. (d) our texture metric does not consider high-level semantic information, and some stripe patterns of zebras have not been smoothed.*

[LM71] LAND E. H., MCCANN J. J.: Lightness and retinex theory. *Journal of the Optical Society of Ameraica 61*, 1 (1971), 1–11. 3

[MCL*14] MIN D., CHOI S., LU J., HAM B., SOHN K., DO M. N.: Fast global image smoothing based on weighted least squares. *IEEE Trans. Image Processing 23*, 12 (2014), 5638–5653. 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

[NBGS08] NICKOLLS J., BUCK I., GARLAND M., SKADRON K.: Scalable parallel programming with cuda. *Queue 6*, 2 (2008), 40–53. 8

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM Trans. Graphics 22*, 3 (2003), 313 – 318. 2, 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

[ROF92] RUDIN L. I., OSHER S., FATEMI E.: Nonlinear total variation based noise removal algorithms. *Physica D 60* (1992), 259–268. 1, 2

[SSD09] SUBR K., SOLER C., DURAND F.: Edge-preserving multiscale image decomposition based on local extrema. *ACM Trans. Graphics (Proc. SIGGRAPH Asia 2009) 28*, 5 (2009), Article No. 147. 2

[TFA05] TAPPEN M. F., FREEMAN W. T., ADELSON E. H.: Recovering intrinsic images from a single image. *IEEE Trans. Pattern Analysis Machine Intelligence 27*, 9 (2005), 1459–1472. 3

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

[VPS03] VANHAMEL I., PRATIKAKIS I., SAHLI H.: Multiscale gradient watersheds of color images. *Image Processing, IEEE Transactions on 12*, 6 (2003), 617–626. 2

[Wei98] WEICKERT J.: *Anisotropic diffusion in image processing*, vol. 1. Teubner Stuttgart, 1998. 2

[Wei06] WEISS B.: Fast median and bilateral filtering. *ACM Trans. Graphics 25*, 3 (2006), 519–526. 1, 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, 4, 8, 9, 11, 12, 13, 14

[ZSXJ14] ZHANG Q., SHEN X., XU L., JIA J.: Rolling guidance filter. In *Proc. ECCV*, vol. 8691. 2014, pp. 815–830. 1, 2, 4, 8, 9, 11, 12

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

Appendix

Interval gradient kernel in gradient domain

Let $\{I_1, I_2, \ldots, I_{2n}\}$ be 1D signal values and $\{g_2, g_2, \cdots, g_{2n-1}\}$ be their gradients, such that $g_i = I_{i+1} - I_i$. Then, for each pair of pixels I_p and I_q where $p < q$, we can express I_q as

$$I_q = I_p + \sum_{k=p}^{q-1} g_k. \tag{18}$$

With a symmetric interval gradient kernel, the interval gradient of the n-th pixel is computed as:

$$\nabla_\Omega I_n = \sum_{k=n+1}^{2n} w_k I_k - \sum_{k=1}^{n} w_k I_k, \tag{19}$$

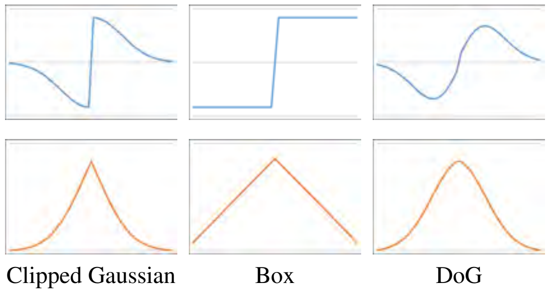


Figure 15: *(top) interval gradient kernels in the intensity domain; (bottom) corresponding equivalents in the gradient domain. For the kernels in the bottom row, the area under the curve is greater than 1 (not normalized) and their maximum values are always 1.*

where w represents symmetric kernel weights such that $w_{2n-k} = w_k$ for $k \in \{1, \ldots, n\}$.

Eq. (19) can be rearranged as:

$$\nabla_\Omega I_n = \sum_{k=1}^{n} w_k \left(I_{2n-k+1} - I_k\right), \tag{20}$$

and by substituting I_{2n-k+1} using Eq. (18) we obtain

$$\nabla_\Omega I_n = \sum_{k=1}^{n} w_k \sum_{i=k}^{2n-k} g_i. \tag{21}$$

By rearranging in terms of g, we obtain

$$\nabla_\Omega I_n = \sum_{k=1}^{2n-1} \bar{w}_k g_k, \tag{22}$$

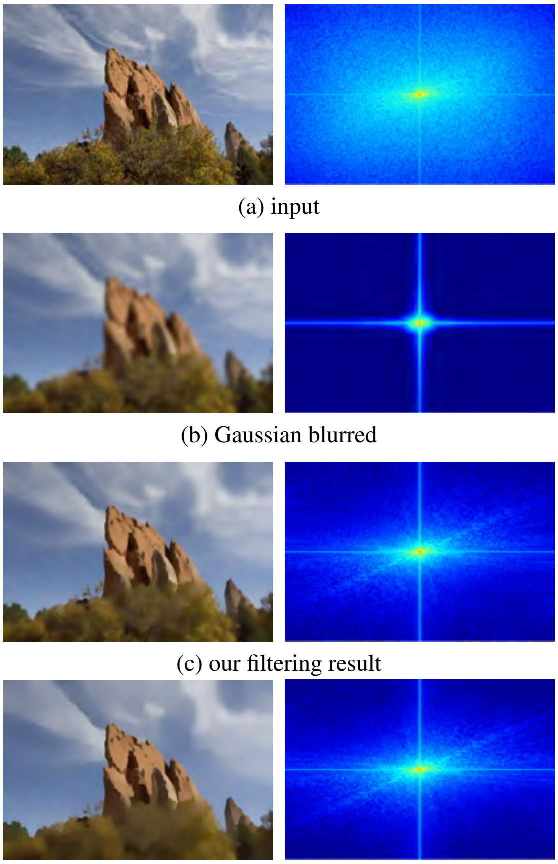
where $\bar{w}$ represents the symmetric weights with length of $2n-1$, such that

$$\bar{w}_k = \begin{cases} \sum_{i=1}^{k} w_k & k \in \{1, \ldots, n\} \\ \sum_{i=1}^{2n-k} w_k & k \in \{n+1, \ldots, 2n-1\} \end{cases}, \tag{23}$$

which becomes unnormalized smoothing of gradients. Fig. 15 shows interval gradient kernels and their corresponding smoothing kernels in the gradient domain.

Power spectrum analysis

Fig. 16 shows an example of the power spectrum analysis. Gaussian smoothing blurs not only textures but also structure edges, causing most high-frequency energies suppressed and flattened in the power spectrum. Our filtering method, on the other hand, smoothes high frequency textures but preserves sharp structure edges, resulting in the power spectrum where high-frequency energies have been reduced overall but still remain due to some from sharp structure edges. Fig. 16d shows that the power spectrum of the filtering result of the state-of-the-art optimization-based texture-filtering method [XYXJ12] is quite similar to ours.



(a) input

(b) Gaussian blurred

(c) our filtering result

(d) filtering result of [XYXJ12]

Figure 16: *Power spectrum analysis example. (left) input and smoothed images; (right) power spectrums of the images. Best viewed on a color screen.*