Structure-Texture Decomposition of Images with Interval Gradient

Hyunjoon Lee¹, Junho Jeon¹, Junho Kim², and Seungyong Lee¹

¹POSTECH, Pohang, Korea
{crowlove, zwitterion27, leesy}@postech.ac.kr
²Kookmin 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

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, XLYX12, HBZW14] and filtering-based approaches [PHK11, KEE13, CLKL14, ZSXJ14].

Optimization-based approaches [ROF92, XLYX12, 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 \( \nabla T \) while preserving \( \nabla S \), and ii) re-
construction of the filtered image from the smoothed gradients, which not only preserves $\nabla S$ 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 $\nabla T$ from $\nabla S$ in the spatial support $\Omega$. We propose a gradient rescaling method based on the interval gradient to suppress $\nabla T$ while preserving $\nabla S$ 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] allows strong 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.

2.2. 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 measur-
In a 1D signal $I$, for a pixel $p$, one common gradient operator is the forward differentiation, defined as:

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

(1)

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

$$ (\nabla I^\Omega)_p = g^\Omega_l(I_p) - g^\Omega_r(I_p), $$

(2)

where $g^\Omega_l$ and $g^\Omega_r$ respectively represent left and right clipped 1D Gaussian filter functions defined by

$$ g^\Omega_l(I_p) = \frac{1}{k_l} \sum_{n \in \Omega(p)} w_\sigma(n-p-1) I_n, $$

$$ g^\Omega_r(I_p) = \frac{1}{k_r} \sum_{n \in \Omega(p)} w_\sigma(p-n) I_n. $$

(3)

where $w_\sigma$ is the clipped exponential weighting function with a scale parameter $\sigma$:

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

(4)

and $k_r$ and $k_l$ are normalizing coefficients defined as:

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

(5)

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

![Forward differentiation and interval gradient kernels. Left: Conventional forward difference kernel; right: our interval gradient kernel with $\sigma = 3$.](image)

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
that for a local window
The most important characteristic of the interval gradient is
3.3. Characteristics of interval gradient
paper unless otherwise stated.
smoothing textures. It is used to generate all examples in this
3.3. Characteristics of interval gradient
The most important characteristic of the interval gradient is

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

each other. The sign of \(|\nabla I_p|\) 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 can-
celled out because of the smoothing operators \(g^*_s\) and \(g^*_o\)
and \(|(\nabla I_p)|\) becomes much smaller than \(|(\nabla I)|\). 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 ten-
dency 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 tex-
ture, \(|(\nabla I_p)|\) is smaller than \(|(\nabla I)|\), although the sign of
\(|(\nabla I_p)|\) 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 im-
gages in the dataset given in [XYXJ12], using previous tex-
ture filtering methods [XYXJ12, ZSXJ14, CLKL14]. Then,
we measured the ratios between the magnitudes of the original
and interval gradients \((|(\nabla I_p)| / |(\nabla I)|)\) 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.

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.

Validation To validate the property of our interval gradient,
we conducted the following experiment. We filtered 200 im-
gages in the dataset given in [XYXJ12], using previous tex-
ture filtering methods [XYXJ12, ZSXJ14, CLKL14]. Then,
we measured the ratios between the magnitudes of the original
and interval gradients \((|(\nabla I_p)| / |(\nabla I)|)\) 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.

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.
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 \( \Omega_p \). With these objectives, we use the following equation to rescale the gradients of the input signal with the corresponding interval gradients:

\[
(\nabla' I)_p = \begin{cases} 
(\nabla I)_p \cdot w_p & \text{if } \text{sign}(\langle \nabla I \rangle_p) = \text{sign}(\langle \nabla \Omega I \rangle_p) \\
0 & \text{otherwise},
\end{cases}
\]

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

\[
w_p = \min \left(1, \frac{\|\nabla I\|_p + \varepsilon_s}{\|\nabla I\|_p + \varepsilon_s} \right),
\]

where \( \varepsilon_s \) is a small constant to prevent numerical instability. Too small values of \( \varepsilon_s \) would make the algorithm sensitive to noise, introducing unwanted artifacts to filtering results. The sensitivity to noise can be reduced by increasing \( \varepsilon_s \), but textures could not be completely filtered if \( \varepsilon_s \) 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 \( \langle \nabla \Omega I \rangle_p \), 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 \( \nabla S \) and suppressing \( \nabla T \). 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:

\[
R_p = \sum_{k=0}^{p-1} l_0 + (\nabla'I)_k, \ p \in \{0, 1, \ldots, N_p\},
\]

where \( l_0 \) 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:

\[
\arg \min_{a_p, b_p} \sum_{n \in \mathbb{N}(\cdot)} w_n \left( (a_p R_n + b_p - l_0)^2 + \varepsilon a_p^2 \right),
\]

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

\[
w_n = \exp \left( -\frac{(n-p)^2}{2\sigma^2} \right),
\]

and \( \varepsilon \) is the smoothness parameter. The effects of the parameters \( \sigma \) and \( \varepsilon \) 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 $\triangleright$ Eq. (6)
    $\nabla_y I \leftarrow$ rescaled gradients along y-axis
    for $i := 0 \ldots (N_y - 1)$ do
        $\sigma_i \leftarrow \sigma \cdot \sqrt{3} \cdot 2^{i} \cdot \delta \downarrow$ Adapted from [GO11]
        $S_x \leftarrow$ 1D filtering result along x-axis $\triangleright$ Eqs. (11)-(13)
    end for
    $I \leftarrow S_x$
    $S_y \leftarrow$ 1D filtering result along y-axis
end while
$S \leftarrow I$

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

$$a_p = \frac{g\sigma((R)_{p}) - g\sigma(R_p) + g\sigma(I_p)}{g\sigma(R_{p})^2 - g\sigma(R_p)^2 + \varepsilon} \quad (11)$$

$$b_p = g\sigma(I_p) - a_p \cdot g\sigma(R_p), \quad (12)$$

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

$$S_p = \tilde{a}_p R_p + \tilde{b}_p, \quad (13)$$

where $\tilde{a}_p = g\sigma(a_p)$ and $\tilde{b}_p = g\sigma(b_p)$ 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 $\sigma$. 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.

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 $\delta_t$, so that

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

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.
Figure 7: Results with varying parameters. (top) bigger scale textures are removed as $\sigma$ increases; (bottom) smoother results are obtained with bigger $\epsilon$.

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, $\sigma$ and $\epsilon$. Parameter $\sigma$ controls the scale of textures to be removed, and $\epsilon$ controls the smoothness of the result. In our experiments, we used $\sigma \in [2, 5]$ and $\epsilon \in [0.01^2, 0.03^2]$. Fig. 7 shows filtering examples to demonstrate the effects of the parameters. For the convergence threshold $\delta_t$ 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 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.,

$$w_p = \min \left( 1, \frac{\sum_{c \in \{r, g, b\}} \left| \nabla G_p \right|^2 + \epsilon_s}{\sum_{c \in \{r, g, b\}} \left| \nabla I_c \right|^2 + \epsilon_s} \right).$$

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-
We compared our filtering method with a few previous approaches [XYXJ12, CLKL14, ZSXJ14]. For RTV [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. As another application, Fig. 12 shows an example; more results can be seen in 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.

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-
H. Lee, J. Jeon, J. Kim, & S. Lee / Structure-Texture Decomposition of Images with Interval Gradient


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.

gradient 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


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] (λ = 0.01, σ = 3), [CLKL14] (k = 2, niter = 2), [ZSXJ14] (σr = 3, σt = 0.1, niter = 5), and our method (σ = 2.5, ε = 0.015²); (bottom example) [XYXJ12] (λ = 0.01, σ = 3), [CLKL14] (k = 3, niter = 2), [ZSXJ14] (σr = 4, σt = 0.1, niter = 5), and our method (σ = 2.5, ε = 0.02²).
Figure 13: Inverse halftoning example. Our method can effectively remove halftoning artifacts from the input image, while preserving shading and strong edges.

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.


Let \( \{I_1, I_2, \ldots, I_{2n}\} \) be 1D signal values and \( \{g_1, g_2, \ldots, g_{2n-1}\} \) 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

\[
I_q = I_p + \sum_{k=p}^{q-1} g_k.
\]  

(18)

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

\[
\nabla \Omega I_n = \sum_{k=-n+1}^{n} w_k I_k - \sum_{k=1}^{n} w_k I_k.
\]  

(19)

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:

\[
\nabla \Omega I_n = \sum_{k=1}^{n} w_k (I_{2n-k+1} - I_k),
\]  

(20)

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

\[
\nabla \Omega I_n = \sum_{k=1}^{n} w_k \sum_{i=k}^{2n-k} g_i.
\]  

(21)

By rearranging in terms of \( g \), we obtain

\[
\nabla \Omega I_n = \sum_{k=1}^{2n-1} \tilde{w}_k g_k,
\]  

(22)

where \( \tilde{w} \) represents the symmetric weights with length of \( 2n-1 \), such that

\[
\tilde{w}_k = \begin{cases} 
    \sum_{i=1}^{k} w_k & k \in \{1, \ldots, n\} \\
    \sum_{i=n+1}^{2n-k-1} w_k & k \in \{n+1, \ldots, 2n-1\} \end{cases}
\]  

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