Image smoothing based on global sparsity decomposition and a variable parameter

Xiang Ma¹, Xuemei Li^{1,2}, Yuanfeng Zhou¹, and Caiming Zhang^{1,2,3} (✉)

© The Author(s) 2021.

Abstract Smoothing images, especially with rich texture, is an important problem in computer vision. Obtaining an ideal result is difficult due to complexity, irregularity, and anisotropy of the texture. Besides, some properties are shared by the texture and the structure in an image. It is a hard compromise to retain structure and simultaneously remove texture. To create an ideal algorithm for image smoothing, we face three problems. For images with rich textures, the smoothing effect should be enhanced. We should overcome inconsistency of smoothing results in different parts of the image. It is necessary to create a method to evaluate the smoothing effect. We apply texture pre-removal based on global sparse decomposition with a variable smoothing parameter to solve the first two problems. A parametric surface constructed by an improved Bessel method is used to determine the smoothing parameter. Three evaluation measures: edge integrity rate, texture removal rate, and gradient value distribution are proposed to cope with the third problem. We use the alternating direction method of multipliers to complete the whole algorithm and obtain the results. Experiments show that our algorithm is better than existing algorithms both visually and quantitatively. We also demonstrate our method's ability in other applications such as clip-art compression artifact removal and content-aware image manipulation.

Keywords image smoothing; texture removal; global sparse decomposition; Bessel method

1 Introduction

1.1 The problem

Natural images usually contain texture and structure. The human visual system can easily understand the structure without being affected by texture. However, for the computer, because the texture can be complex, irregular, and anisotropic [1], it is a challenging task to remove texture from an image. The purpose of image smoothing is to remove the texture without destroying the structure as much as possible. Image smoothing is an important and widely used image processing technology; in tasks such as image segmentation, edge extraction, image enhancement, image decomposition, and artifact removal, it simplifies the problem immensely. Existing image smoothing algorithms can be roughly divided into three categories: filters based on local information, global optimization frameworks, and data-driven methods.

1.2 Filters based on local information

Bilateral filtering (BLF) [2] is a representative smoothing filter, which achieves smoothing by estimating the value of local patches by weighted average (Gaussian kernel estimation). After BLF was proposed, many improved versions [3, 4] appeared, mostly using modified Gaussian kernels. Among them, bilateral texture filtering (BTF) [5] can ensure high sharpness of edges, but flat regions lack regularity, and the visual effect is poor. Tree filtering [6] successfully mitigates the ringing phenomenon by constructing a tree structure, but if a misclassified pixel occurs, it causes the main edge to break down, resulting in a false boundary. Filters based on local information also include anisotropic filters [7], guided filters [8], an extremum smoothing algorithm [9], etc. Most

1 Shandong University, Jinan 250101, China. E-mail: X. Ma, sdu_max@163.com; X. Li, xmlisdu@sdu.edu.cn; Y. Zhou, yfzhou@sdu.edu.cn; C. Zhang, czhang@sdu.edu.cn (✉).

2 Shandong Co-Innovation Center of Future Intelligent Computing, Yantai 264025, China.

3 Digital Media Technology Key Lab of Shandong Province, Jinan 250014, China.

Manuscript received: 2021-01-25; accepted: 2021-03-01

of these filters are intuitive and simple, but are too dependent on local information and cause ringing.

1.3 Global optimization frameworks

Weighted least squares (WLS) [10] is a relatively robust image smoothing algorithm based on global optimization, particularly suitable for gradual image coarsening and edge preserving multi-scale detail extraction. Inspired by the difference-of-Gaussian (DoG) feature extraction algorithm [11, 12], relativity-of-Gaussian (RoG) [13] smooths the image by describing the relationship between Gaussian filters with different sizes. However, RoG has several problems, such as difficult parameter control and a tendency to local information loss. A well-known algorithm, total variation (TV) [14] performing regular optimization based on the L_1 norm, is often used in image denoising and restoration, but cannot effectively achieve smoothing. By approximately highlighting the image structure to control the number of non-zero gradients, L_0 gradient minimization (L_0) [15] algorithm has adequate protection for the main edges, but can easily lose original colors, so is lacking in aesthetics. L_0 gradient projection (L_0p) [16] overcomes the difficulty of parameter control of L_0 , without limiting the obvious pseudo-boundaries. To overcome the shortcomings of L_0p , algorithm in Ref. [17] restricts the smoothed image's gradient, only matching a few images. Compared with filtering methods, the optimization framework is more flexible but lacks local information protection, and can easily lose local weak edges. The relative total variation (RTV) [18] algorithm applies the relative norm to combine local filters with global optimization. Although the effect is sound, it may cause edge expansion and fail to protect local weak edges.

1.4 Data-driven methods

With the development of deep learning, data-driven image smoothing algorithms have appeared. However, as there is no ground truth in image smoothing, conventional supervised and semi-supervised learning methods cannot be readily used. The algorithms [19, 20] attempt to use a unified CNN framework to simulate the previous smoothing methods [10, 18], without getting rid of the limitations of the original algorithms. Although the algorithm (DVP) [21] optimizes the image smoothing process to improve

the effect to a certain extent by training parameters, generalization is always the barrier.

1.5 Contributions

Most filters based on local information are relatively intuitive and simple. Nonetheless, they tend to depend on the image's local information, resulting in ringing, edge expansion, and other phenomena. The optimization framework methods are flexible, but the regular terms' global selection can hardly protect local information. Generalization is still a limitation of the data-driven methods. Combining local filters and the global optimization framework can mitigate the problem to some extent, but is still not ideal, with, e.g., insufficient weak-edge protection.

Overall, we may summarize the three main problems faced by image smoothing at present, and propose improvement schemes:

1. *The smoothing effect should be enhanced for richly textured images.* In general, with increasing image texture, current algorithms' smoothing effects become worse, as shown in Fig. 1. We set a global sparse regularity term to decompose the image into two parts and remove the textured part to improve performance.
2. *Inconsistency of smoothing in different parts of the image needs improvement.* Due to uneven illumination, contrast, and other factors in the image, smoothing parameters should differ across the image. Therefore, for locally adaptive parameters, patch-shift is used. More importantly, to determine the pseudo-boundary, we propose to conduct parametric surface fitting to allow

Fig. 1 Examples of image smoothing: (a, d) original images; (b, e) L_0 [15] smoothing results; (c, f) our results.

continuous parameter variation throughout the region and improve efficiency.

3. *A method of evaluating the smoothing effect is needed.* Without ground truth, image smoothness cannot be directly evaluated by PSNR, SSIM, or other conventional metrics. Comparing different algorithms based on visuals alone is too subjective, so there is an urgent need for quantitative measures as evaluation criteria. The difficulty with image smoothing lies in the algorithm's ability to distinguish between texture and edge, but the human eye can readily achieve this. Therefore, we can compare the edges extracted from the smoothing results with manually selected edges from the original image to evaluate the smoothing effect. Furthermore, image smoothing changes the gradient distribution, and the gradient is positively correlated with image smoothness, so we can compare the gradient distribution of the results to evaluate algorithms. Based on the above two points, we propose three measures: edge integrity rate, texture removal rate, and gradient value distribution to quantitatively evaluate the results using edges and gradients.

In summary, we combine local filters with the global optimization framework and propose an image smoothing algorithm based on global sparsity decomposition and a variable parameter. Firstly, the global sparse decomposition is used to pre-remove part of the texture to improve the smoothing performance. The variable parameter is then obtained via a parametric surface by patch-shift selection using the improved Bessel method to ensure localization and continuity. Finally, to limit image gradient variation through the L_1 norm, we achieve image smoothing. A flow chart is shown in Fig. 2.

The main structure of this paper is as follows:

Fig. 2 Flowchart of proposed algorithm.

Section 2 introduces the model framework used in our algorithm, as well as the global sparse decomposition, patch-shift parameter selection, and improved Bessel fitting. Section 3 describes in detail the solution of our algorithm based on alternating direction method of multipliers (ADMM). Section 4 shows the effects of selecting different parameters and compares the differences in visual effects and quantitative results between other algorithms and ours. Section 5 shows application of our algorithm to clip-art compression artifact removal and content-aware image manipulation. Finally, Section 6 summarizes the paper.

2 Problem formulation

2.1 Problem

This section considers how to solve the first two problems. Natural images can generally be described as

$$y = x + n$$

Here y , x , and n represent the original image, structure, and texture respectively. The goal of image smoothing is to obtain x from y . The global optimization framework can be described as

$$\hat{x} = \arg \min_x \frac{1}{2} \|y - x\|_2^2 + \lambda R(x) \quad (1)$$

where $\hat{x}$ is the result, the first term is a fidelity term, and λ is a smoothing parameter. $R(x)$ is the regularisation term, which is prior information and non-negative. We require $\lambda > 0$, otherwise $R(x)$ may not give the right guidance. For example, if $\lambda = -1$, the second term in Eq. (1) is $\| -\nabla x \|_2^2$ causing us to expect a larger gradient of x when finding the minimum, which runs counter to our intent of removing texture information.

2.2 Global sparse decomposition

For the first problem, we decompose the image into two parts: the low frequency part representing the structure and the high frequency part containing the texture. In image super-resolution and image reconstruction, the high frequencies are usually considered to be the missing information in the scaling process used to refine the result [22–24].

In contrast, high frequencies need to be removed during image smoothing. What needs to be made clear here is that we need to remove the texture beforehand and to ensure as much as possible that edges are not damaged. Therefore, global

sparse decomposition is used to ensure that the high frequencies are sparse and to reduce the loss of structural information. The algorithm can be described as

$$R_{str}(y) = \|y_H\|_1 + \frac{\kappa}{2} \sum_{d=1}^4 \|\nabla_d \otimes y_L\|_2^2$$

s.t. $y = f_L \otimes y_L + y_H$

where y_L and y_H represent low and high frequency information respectively. f_L is a low-pass filter, and $\otimes$ is the convolution operator. $f_L \otimes y_L$ is used to ensure the smooth component contains low-frequency information, so as to ensure that y_H approximately represents the texture. κ controls the smoothness: the larger κ , the more information y_H contains. ∇_d is the gradient in direction d , with $d \in \{1 = \text{horizontal}, 2 = \text{vertical}, 3 = 45^\circ, 4 = 135^\circ\}$. It is well known that the L_p norm can promote sparsity when $p \leq 1$. Here we use $\|y_H\|_1$ to force y_H to be the sparse component under the L_1 norm (L_1 norm is used to ensure convexity), making y_H contain only texture without destroying the structure.

We compare high frequency y_H with the gradient in Fig. 3, and label different colors according to the pixel values. Obviously, y_H is more sparse than the gradient. We further analyze this property in Fig. 4, and present the numerical distribution of gradient and y_H with different κ . It can be seen that the peak value of y_H is near 0, and the numerical distribution is closer to the Laplace distribution. This is since y_H is treated sparsely under the L_1 norm. Comparatively, the gradient's numerical distribution is fluctuating, and the peak is non-zero, containing much missing structural information from the image. Furthermore, it can be presumed that κ affects the sparsity of y_H : the larger κ , the sparser y_H . After removing y_H , we use y_L for smoothing.

Fig. 3 Comparison between image gradient and the proposed residual component y_H for $|\kappa| = 1$.

Fig. 4 Image gradient distribution and y_H distribution for different κ values.

2.3 Patch-shift parameter selection and surface fitting

The second problem occurs mainly because λ is a constant parameter. Even if we get the globally optimal solution, it may not be locally optimal. Separate parameter calculations for all points can validly solve this problem but are extremely time-consuming, so we propose a two-step parameter calculation algorithm, including patch-shift parameter selection and parametric surface fitting.

2.3.1 Patch-shift parameter selection

Patch-shift is an intuitive approach, where we set the values of patches by comparing them to global variations. To simply adjust the smoothness, a user adjustable parameter λ_G replaces the original λ . We define the local parameter $\lambda_{i,j}$ by

$$\begin{cases} \lambda_{i,j} = \chi_{i,j} \lambda_G \\ \chi_{i,j} = s \exp(-\sigma(y_L)/(\varepsilon + \sigma(\Omega_{i,j}))) \end{cases} \quad (2)$$

where $\Omega_{i,j}$ refers to the patch and (i,j) are its coordinates. $\sigma(\cdot)$ is the standard deviation operation. $\chi_{i,j}$ is the fluctuation rate. s is a simple adjustment factor. ε is a small value to prevent the denominator from being zero. As shown in Fig. 5, the smoother $\Omega_{i,j}$, the smaller and more rapidly decreasing $\chi_{i,j}$. However, due to patch-shift parameter selection discontinuity, the results show an obvious pseudo boundary at the junctions of patches, as shown in Fig. 6(d).

Fig. 5 Image gradient distribution and y_H distribution for different κ values.

Fig. 6 Comparisons of parametric surface with or without Bessel method: (a) original image; (b) smoothed result with improved Bessel method; (c) parametric surface with Bessel method; (d) smoothed result without improved Bessel method; (e) parametric surface without Bessel method.

2.3.2 Parametric surface fitting

To solve this problem, we propose a novel algorithm. We assign the patches' parameters to their center points to get a set of sample values. This grid of values can be considered to be a low-resolution surface, and we fit a parametric surface based on it, to give each pixel can get a unique parameter. After comparing various fitting methods, the Bessel method was chosen because: (i) the sample values calculated by Eq. (2) tend to fluctuate and cannot well-tuned by s alone. The Bessel method typically smooths the midpoint by passing only the starting and ending points of the sample values, which allows for easy parameter adjustment, and (ii) since Bessel method guarantees convexity, ensuring that λ correctly satisfies $\lambda > 0$.

Usually, the greatest similarity is found between adjacent points, so we propose a fitting method based on neighboring patches (n-patches), to allow more sample values to act on point p which needs to be found. While considering the computational difficulty, the 16 sample values closest to p are chosen to construct a parametric surface. Each 3×3 patch is called an n-patch, and 16 sample values constitute 4 n-patches, as shown in Fig. 7 (all red points construct one n-patch). We assume the window sliding step to be 1 for ease of illustration. $F_{i,j}(p)$ refers to the parametric surface of point p . (i,j) is the coordinate of the nearest sample value to the lower left of p and (p_i, p_j) is the coordinate of p . Moreover, we set $f_{i,j}(p)$ as the surface for each n-patch, and use the following function to fit the nine sample values:

$$f_{i,j}(p) = \sum_{h=-1}^1 \sum_{v=-1}^1 \varphi_h(m) \varphi_v(n) \chi_{i+h, j+v}, \quad 0 \leq m, n \leq 1$$

where $\chi_{i,j}$ refers to the sample values, $m = (p_i - i + 1)/2$, $n = (p_j - j + 1)/2$. Here $\varphi_c(t)$, $c \in \{-1, 0, 1\}$

Fig. 7 Bessel method.

are Bzier basis functions defined by

$$\varphi_{-1}(t) = (1-t)^2, \quad \varphi_0(t) = 2t(1-t), \quad \varphi_1(t) = t^2$$

and t is the distance from p to the reference point of $f_{i,j}(p)$ in the space (m, n) . All of $f_{i,j}(p)$, $f_{i+1,j}(p)$, $f_{i,j+1}(p)$, and $f_{i+1,j+1}(p)$ can give different parameters. However, we expect points with the same pixel values to have the same parameters in order to make the image smoother except at the edges, so a pixel-sensitive Gaussian weight considering pixel values is used to sum the four parameters. $F_{i,j}(p)$ can be defined as

$$F_{i,j}(p) = \frac{\sum_{h=0}^1 \sum_{v=0}^1 \omega_{i+h,j+v}(p) f_{i+h,j+v}(p)}{\sum_{h=0}^1 \sum_{v=0}^1 \omega_{i+h,j+v}(p)} \quad (3)$$

The weights in Eq. (3) are

$$\omega_{i,j}(p) = \beta_{i,j}(p)(1-m)(1-n)$$

$$\omega_{i+1,j}(p) = \beta_{i+1,j}(p)m(1-n)$$

$$\omega_{i,j+1}(p) = \beta_{i,j+1}(p)(1-m)n$$

$$\omega_{i+1,j+1}(p) = \beta_{i+1,j+1}(p)mn$$

with $\beta_{i,j}(p)$ defined as

$$\beta_{i,j}(p) = e^{-(P-P_{i,j})^2/(s\delta)}, \quad \delta = \sum_{i=0}^1 \sum_{j=0}^1 (P - P_{i,j})^2$$

P and $P_{i,j}$ are the pixel values at point p and the center point of $f_{i,j}(p)$, respectively. s is a adjustment factor. As shown in Fig. 8, ω makes the result smoother.

After obtaining the parameters of all points, we combine them as χ_{y_L} . Under the control of λ_G , the final parameter can be expressed as

$$\lambda_{y_L} = \chi_{y_L} \lambda_G \quad (4)$$

Fig. 8 Results with or without ω : (a) original image; (b) smoothed result without ω ; (c) smoothed result with ω .

3 Efficient ADMM method for image smoothing

In order to improve efficiency, we combine the global sparse decomposition, the parametric surface, and the L_1 norm to give our model [25, 26]:

$$\arg \min_{x, y_L, y_H} \frac{1}{2} \|y - x\|_2^2 + \lambda_{y_L} \sum_d \|\nabla_d x\|_1 + \alpha \|y_H\|_1 + \frac{\kappa}{2} \sum_d \|\nabla_d \otimes y_L\|_2^2 \quad \text{s.t.} \quad y = f_L \otimes y_L + y_H \quad (5)$$

Here α and κ weigh the sparsity of y_H . λ_{y_L} controls the sparsity of gradient. Eq. (5) is non-differentiable and non-linear and is difficult to solve directly. So, we adapt ADMM to optimize this function iteratively. Two Lagrange constraints are added based on this strategy:

$$\arg \min_{x, y_L, y_H, T} \frac{1}{2} \|y - x\|_2^2 + \lambda_{y_L} \|T\|_1 + \alpha \|y_H\|_1 + \frac{\kappa}{2} \|\nabla \otimes y_L\|_2^2 + \frac{\gamma_1}{2} \|y - (f_L \otimes y_L + y_H) - \mu_1\|_2^2 + \frac{\gamma_2}{2} \|T - \nabla x - \mu_2\|_2^2 \quad (6)$$

For ease of writing, we omit d and replace the four-direction operator with ∇ . γ_1 and γ_2 are the parameters of the two Lagrange constraints. In practical, γ_1 and γ_2 are initialized to small positive values and are increased in each iteration to ensure convergence. μ_1 and μ_2 are Lagrange multipliers. T is the auxiliary parameter. Eq. (6) is convex, so we can update each parameter iteratively until convergence.

3.1 Solver

3.1.1 Computing y_L

Assuming all other parameters are fixed, we have

$$\arg \min_{y_L} \frac{\kappa}{2} \|\nabla \otimes y_L\|_2^2 + \frac{\gamma_1}{2} \|y - (f_L \otimes y_L + y_H) - \mu_1\|_2^2$$

The above problem can be solved directly by gradient descent and optimized by two-dimensional fast Fourier transform:

$$y_L = \mathcal{F}^{-1} \left(\frac{\gamma \overline{\mathcal{F}(f_L)} \mathcal{F}(y - y_H - \mu_1)}{\kappa \overline{\mathcal{F}(\nabla)} \mathcal{F}(\nabla) + \gamma_1 \overline{\mathcal{F}(f_L)} \mathcal{F}(f_L)} \right) \quad (7)$$

$\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ represent the fast Fourier transform (FFT) and its inverse (IFFT). $\overline{\mathcal{F}(\cdot)}$ denotes complex conjugation. We invert the matrix in the spatial domain by element multiplication in the frequency domain, making the operation more efficient.

3.1.2 Computing y_H

Consistent with the previous idea, we let $\Lambda = y - f_L \otimes y_L + \mu_1$, to give the problem for y_H :

$$\arg \min_{y_H} \alpha \|y_H\|_1 + \frac{\gamma_1}{2} \|y_H - \Lambda\|_2^2$$

This problem can be solved independently for each pixel i via simple soft-thresholding:

$$[y_H]_i = \text{sign}([y_H]_i) \max(0, [\Lambda]_i - \frac{\alpha}{\gamma_1}) \quad (8)$$

3.1.3 Computing T

Similarly, fixing the variables other than T in Eq. (6), the problem for T can be expressed as follows:

$$\arg \min_T \lambda_{y_L} \|T\|_1 + \frac{\gamma_2}{2} \|T - \nabla x - \mu_2\|_2^2$$

In the same way as for Eq. (8), we obtain

$$[T]_i = \text{sign}([T]_i) \max \left(0, [\nabla x + \mu_2]_i - \left\lfloor \frac{\lambda_{y_L}}{\gamma_2} \right\rfloor_i \right) \quad (9)$$

3.1.4 Computing x

After finding T , y_L , and y_H , the optimization of x can be described by

$$\arg \min_x \frac{1}{2} \|y - x\|_2^2 + \frac{\gamma_2}{2} \|T - \nabla x - \mu_2\|_2^2$$

The above problem also meets the requirements of gradient descent, and can be solved using:

$$x = \mathcal{F}^{-1} \left(\frac{\mathcal{F}(y) + \gamma_2 \overline{\mathcal{F}(\nabla)} \mathcal{F}(T - \mu_2)}{1 + \gamma_2 \overline{\mathcal{F}(\nabla)} \mathcal{F}(\nabla)} \right) \quad (10)$$

3.1.5 Updating μ_1 and μ_2

At the end of each iteration, the Lagrange multipliers must be updated:

$$\begin{cases} \mu_1 = \mu_1 + (f_L \otimes y_L + y_H - y) \\ \mu_2 = \mu_2 + (\nabla x - T) \end{cases} \quad (11)$$

3.2 Algorithm summary and complexity

The entire reconstruction process is outlined in Algorithm 1. The most time-consuming part is the solution of λ_{y_L} , which depends on the number of pixels N , the number of patches K , and the patch size k . In general, we do not reduce N during smoothing as doing so loses detail. However, K and k can directly affect the fitting effect, so we conducted a series of experiments to balance time and quality. We use N_{∇} to describe the smoothing effect:

Algorithm 1 Image smoothing based on global sparsity decomposition and variable parameter

Input:

Original image: y

ADMM parameters: $\mu_1, \mu_2, \gamma_1, \gamma_2$

Output:

Smoothed image: x

- 1: Initialization: $x = y, \mu_1 = 0, \mu_2 = 0$
- 2: **while** not converged **do**
- 3: Solve Subproblem y_L using Eq. (7);
- 4: Solve Subproblem y_H using Eq. (8);
- 5: Obtain adaptive parameter λ_{y_L} from y_L by Eq. (4);
- 6: Solve Subproblem T using Eq. (9);
- 7: Update x using Eq. (10);
- 8: Update μ_1 and μ_2 using Eq. (11);
- 9: **end while**

$$N_{\nabla}(x) = \sum_{n=1}^N C(|\nabla x_n|), \quad C(i) = \begin{cases} 0, & i = 0 \\ 1, & i > 0 \end{cases}$$

Assuming that images x_1 and x_2 are equally large, $N_{\nabla}(x_1) = N_{\nabla}(x_2)$ implies that the two images have the same smoothness. Our algorithm can reduce the gradient at each iteration, so the performance of different K or k can be evaluated by comparing how much time it takes to smooth the same image to the same N_{∇} . The experimental images used for this purpose were all taken from BSD500. As can be seen from Fig. 9(a), the time consumed is almost constant as k increases when N_{∇} is large enough. However, as N_{∇} decreases, the time spent becomes gradually positively correlated with k . Therefore, k can be adjusted according to specific needs. In this paper, we set $k = 3$. As image patches are selected in various ways, comparing K is confusing. Assuming that k is fixed, we replace K with the image patch move

Fig. 9 Impact of image patch choices: (a) patch size k ; (b) patch steps.

steps, which is interpreted as different percentages k . As shown in Fig. 9(b), the operation time decreases first and then increases as *steps* increase. The optimal value is about 0.3. Moreover, a decreasing difference between the λ_{y_L} of two adjacent iterations was witnessed during the experiment. So we set a strategy to reduce the number of λ_{y_L} calculations: after the 10th iteration, we calculate λ_{y_L} every five iterations. Experiments show that this strategy can not only ensure the correctness of our algorithm, but also effectively reduce the calculation time. In terms of convergence, Eq. (6) is convex. When the values of γ_1 and γ_2 are large enough, ADMM can ensure that the variables converge [27–30].

4 Experiments and discussion

In this section, the values of parameters in our algorithm are firstly discussed. Then we compare other algorithms with ours in terms of visual effects, and we create some images to evaluate the results quantitatively. Finally, the solution to the third problem is given.

4.1 Analysis of parameters

The size of f_L can directly affect the time spent on decomposition, so we conducted statistical experiments based on BSD500 to select the most efficient value. As shown in Fig. 10, the time consumed is relatively stable and has the lowest average when f_L is 6×6 . α affects the smoothness of y_L and the information contained in y_H . As can be seen from Fig. 11, image decomposition can effectively separate the high frequencies; y_L becomes smoother

Fig. 10 Impact of f_L on texture pre-removal.

Fig. 11 Decomposition results y_L for different α : (a) original image; (b) $\alpha = 20$; (c) $\alpha = 40$; (d) $\alpha = 60$; (e) $\alpha = 80$; (f) $\alpha = 100$.

as α increases. Here we set $\alpha = 5$. Figure 12 presents a set of smoothed results for different λ_G . It can be seen that the larger λ_G , the smoother the result. We set $\gamma_1 = \gamma_2$ and each iteration increases by 5% [31].

4.2 Comparison of visual effects

In order to demonstrate the effectiveness of our algorithm, we compare it to WLS [10], TV [14], tree filter [6], RoG [13], L_0 [15], RTV [18], and DSHFG [32]. All algorithms used were based on the code provided by the authors and with manual adjustment of parameters.

As shown in Fig. 13, WLS does not distinguish textures and edges well, nor does TV, and the whole result is very blurry. Tree filtering averages bilateral

Fig. 12 Smoothing results for different λ_G : (a) original image; (b) $\lambda_G = 0.001$; (c) $\lambda_G = 0.005$; (d) $\lambda_G = 0.01$; (e) $\lambda_G = 0.02$; (f) $\lambda_G = 0.025$.

Fig. 13 Comparison of results: (a) original image; (b) WLS [10] ($\lambda = 2, \alpha = 2$); (c) TV [14] ($\lambda = 0.08$); (d) tree [6] ($\sigma = 0.015$); (e) ROG [13] ($\lambda = 0.015, \sigma_1 = 1, \sigma_2 = 3$); (f) L_0 [15] ($\lambda = 0.035, \kappa = 2$); (g) RTV [18] ($\lambda = 0.02, \sigma = 3$); (h) DSHFG [32] ($\lambda = 0.02$); (i) ours ($\lambda_G = 0.02$).

weights and tree weights, but it does not protect all edges well. RoG uses several sets of Gaussian kernels with different weights to achieve texture removal, which can fully smooth the image globally, but some edges are not well protected. L_0 can better sharpen and protect the strong edges, but the effect when processing high-contrast texture images is poor, because it is difficult to distinguish textures based solely on gradients. RTV's local regularity can help it to achieve texture removal, but it cannot protect local weak edges well. DSHFG is an L_0 norm minimization smoothing algorithm based on image decomposition, which removes texture well, but it also loses local weak edges. In contrast, our algorithm can not only distinguish texture and structure well and remove

texture, but also effectively protects weak edge.

4.3 Quantitative comparison based on created images

To quantitatively evaluate the results of different algorithms using PSNR, we manually constructed several texture images, as shown in Fig. 14. In order to show the poor generalization of data-driven methods, VDCNN [33] and ResNet [33] are added to the control group. Smoothing results are shown in Fig. 15 and Table 1. It can be seen that most algorithms except TV remove texture well, but there are some artificial textures that have not been removed in Fig. 15(d) and Fig. 15(e). The PSNR shows that our algorithm is better.

Fig. 14 Created images: (a) is a simple created image, and the others are images with different artificial textures.

Table 1 Quantitative comparison based on created images (PSNR)

	TV [14]	RTV [18]	DSHFG [32]	VDCNN [33]	ResNet [33]	Ours
Fig. 14(b)	21.2890	30.7315	27.2117	29.7807	29.6161	31.4621
Fig. 14(c)	21.4065	31.0806	27.6249	30.1331	29.9596	31.8956
Fig. 14(d)	20.6971	27.9442	25.9466	26.9227	26.6810	28.2501
Avg.	21.1309	29.9188	26.9277	28.9455	28.7522	30.5359

Fig. 15 Comparison using created images: (a) TV [14] ($\lambda = 0.1$); (b) RTV [18] ($\lambda = 0.015, \sigma = 3$); (c) DSHFG [32] ($\lambda = 0.01$); (d) ResNet [33]; (e) VDCNN [33]; (f) ours ($\lambda_G = 0.02$). Rows 1–3 correspond to Figs. 14(b)–14(d), respectively.

4.4 Quantitative comparison with the proposed evaluation method

We propose three evaluation metrics in terms of edge and gradient distribution to overcome the dependence on ground truth.

4.4.1 Edge integrity rate and texture removal rate

Conventional edge extraction algorithms cannot reasonably distinguish texture and edges, as shown in Fig. 16(b). So we manually indicate the real edges and present the edge integrity rate and the texture removal rate to evaluate the smoothing effect. The edge integrity rate evaluates the degree of protection of edges. The texture removal rate evaluates the level of texture removal. They are defined as follows:

$$EI = \frac{EE(x) \odot GT(y)}{GT(y)}, TR = \frac{EE(x) \oplus GT(y)}{EE(y)} \quad (12)$$

where EI and TR represent edge integrity rate and texture removal rate. $EE(x)$ and $GT(y)$ are the extracted edges from smoothing results and hand-drawn ground truth. The operators $\odot$ and $\oplus$ mean XNOR and XOR, respectively. While the author of RTV provides the texture image we experimented

with, the manually indicated edges are too coarse, and we redrew them.

Let us first observe the visual differences between the several algorithms: see Fig. 17. Smoothing can result in simplifying edges. WLS and RTV do a good job of removing textures, but they also cause some missing edges. DSHFG can preserve relatively intact edges. However, DSHFG loses some weak edges, such as the flower-like edge at the bottom left of the image.

Tables 2 and 3 show the edge integrity rate and texture removal rate for all algorithms and demonstrate that our algorithm outperforms the others. The average edge integrity rate of all algorithms is lower than 50%. This is because while the human eye can distinguish texture and edges, it can not be easy to determine the exact location of pixel-level edges. The boundaries obtained by smoothing algorithms are typically 1–3 pixels wide, while the labeled data are generally wider than 3 pixels, a problem we address in our next study. As shown in Table 3, the texture removal rates of RTV, DSHFG, and our method are relatively good, with some even exceeding 99%. The effects evaluated by the two indexes are consistent

Fig. 16 Comparison of edges before and after smoothing: (a) original images; (b) original edges; (c) ground truth; (d) edges after smoothing.

Fig. 17 Comparison of edge extraction: (a) original image; (b) ground truth; (c) WLS [10] ($\lambda = 2, \alpha = 2$); (d) RTV [18] ($\lambda = 0.015, \sigma = 3$); (e) DSHFG [32] ($\lambda = 0.01$); (f) ours ($\lambda_G = 0.02$).

Table 2 Comparison of edge integrity rate

	TV [14]	RTV [18]	DSHFG [32]	Ours
01_06.jpg	0.0730	0.6116	0.5758	0.6018
01_15.jpg	0.0860	0.4422	0.3854	0.5034
01_22.jpg	0.1685	0.3945	0.3650	0.4205
02_01.jpg	0.1192	0.4094	0.3760	0.4807
04_08.jpg	0.1400	0.5093	0.4957	0.5050
07_15.jpg	0.0690	0.4153	0.4593	0.5315
07_30.jpg	0.1666	0.5575	0.5197	0.5915
07_34.jpg	0.2391	0.4631	0.5472	0.4951
12_15.jpg	0.1675	0.3350	0.3309	0.3510
12_53.jpg	0.2175	0.2800	0.2772	0.3254
Avg.	0.1997	0.4385	0.4174	0.4905

Table 3 Comparison of texture removal rate

	TV [14]	RTV [18]	DSHFG [32]	Ours
01_03.jpg	0.1897	0.8450	0.7684	0.9012
01_07.jpg	0.3450	0.9979	0.9466	0.9968
01_09.jpg	0.3927	0.9339	0.9558	0.9950
01_25.jpg	0.4533	0.9417	0.9268	0.9608
02_16.jpg	0.3338	0.7933	0.7982	0.8348
11_12.jpg	0.4882	0.8957	0.8052	0.9350
12_26.jpg	0.4480	0.9511	0.9040	0.9745
13_02.jpg	0.4465	0.8712	0.8711	0.9247
13_05.jpg	0.6277	0.9879	0.9500	0.9907
13_17.jpg	0.4800	0.9868	0.9657	0.9932
Avg.	0.4126	0.8622	0.8287	0.8978

with our visual conclusions as a whole, indicating that these two indexes can provide a good quantitative comparison of image smoothing.

4.4.2 Gradient value distribution

Image smoothing is about eliminating as much redundant texture as possible, which leads to gradients in the sparse direction. Thus, gradient value distribution can also be used to describe smoothness. On the premise of ensuring that structure is not destroyed, the more the distribution tends to 0, the sparser the gradient, and the better the smoothing effect. As shown in Fig. 18, the peak values of the gradient for all algorithms lie around 0, indicating that the gradients of smoothed images tend to be sparse. Other than TV, our algorithm has the highest sparsity. From a visual inspection, it can be seen that TV destroys structural information, which leads to extreme sparsity.

In summary, our algorithm outperforms the others in visual performance and is supported by the three suggested metrics: edge integrity rate, texture removal rate, and gradient value distribution.

Fig. 18 Gradient value distribution: (a) average for all images; (b) distribution for 01_03.jpg.

5 Applications

5.1 Clip-art compression artifact removal

Image processing operations such as compression or super-resolution can distort images such as cartoons and clip-art, and generate pseudo boundaries that traditional denoising algorithms cannot remove. As can be seen in Fig. 19, image smoothing can effectively solve this problem, and our method provides better results than with others.

Fig. 19 Smoothing for artifact removal: (a) original image; (b) L_0 ; (c) ROG; (d) RTV; (e) DSHFG; (f) ours.

Fig. 20 Content-aware image manipulation: (a, c) original images; (b) background blurring; (d) foreground enhancement.

5.2 Content-aware image manipulation

Our proposed method can be combined with image significance detection [34] to realize content-aware image manipulation by dividing the image into foreground and background and processing them separately to achieve foreground enhancement or background blurring.

6 Conclusions and limitations

In summary, we have made targeted improvements to three current problems in image smoothing. We

enhance smoothing performance on richly-textured images by pre-removing textures based on global sparse decomposition. By parameter adaptation based on patch-shift and parametric surface fitting through an improved Bessel method, we adapt to the inconsistency of different parts of the image. Three evaluation metrics are proposed to quantitatively evaluate smoothing performance, avoiding dependence on ground truth. Comparisons with existing algorithms demonstrate that our algorithm works better. However, our algorithm also has limitations. We do not solve the problem of training pairs, so it cannot train a convolutional network intuitively. If this problem is solved, the smoothing quality can be further improved, which is what we hope to do next.

Acknowledgements

This work was supported by NSFC Joint Fund with Zhejiang Integration of Informatization and Industrialization under Key Project (U1609218).

References

- [1] Sun, Y. J.; Schaefer, S.; Wang, W. P. Image structure retrieval via L minimization. *IEEE Transactions on Visualization and Computer Graphics* Vol. 24, No. 7, 2129–2139, 2018.
- [2] Tomasi, C.; Manduchi, R. Bilateral filtering for gray and color images. In: *Proceedings of the International Conference on Computer Vision*, 839–846, 1998.
- [3] Chen, J. W.; Paris, S.; Durand, F. Real-time edge-aware image processing with the bilateral grid. *ACM Transactions on Graphics* Vol. 26, No. 3, 103, 2007.
- [4] Weiss, B. Fast median and bilateral filtering. *ACM Transactions on Graphics* Vol. 25, No. 3, 519–526, 2006.
- [5] Cho, H.; Lee, H.; Kang, H.; Lee, S. Bilateral texture filtering. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 128, 2014.
- [6] Bao, L. C.; Song, Y. B.; Yang, Q. X.; Yuan, H.; Wang, G. Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree. *IEEE Transactions on Image Processing* Vol. 23, No. 2, 555–569, 2014.
- [7] Perona, P.; Malik, J. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 12, No. 7, 629–639, 1990.
- [8] He, K. M.; Sun, J.; Tang, X. O. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 35, No. 6, 1397–1409, 2013.
- [9] Subr, K.; Soler, C.; Durand, F. Edge-preserving multiscale image decomposition based on local extrema. *ACM Transactions on Graphics* Vol. 28, No. 5, Article No. 147, 2009.
- [10] Farbman, Z.; Fattal, R.; Lischinski, D.; Szeliski, R. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics* Vol. 27, No. 3, Article No. 67, 2008.
- [11] Lindeberg, T. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of Applied Statistics* Vol. 21, Nos. 1–2, 225–270, 1994.
- [12] Mikolajczyk, K.; Schmid, C. An affine invariant interest point detector. In: *Proceedings of the 7th European Conference on Computer Vision-Part I*, 128–142, 2002.
- [13] Cai, B. L.; Xing, X. F.; Xu, X. M. Edge/structure preserving smoothing via relativity-of-Gaussian. In: *Proceedings of the IEEE International Conference on Image Processing*, 250–254, 2017.
- [14] Rudin, L. I.; Osher, S.; Fatemi, E. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena* Vol. 60, Nos. 1–4, 259–268, 1992.
- [15] Xu, L.; Lu, C. W.; Xu, Y.; Jia, J. Y. Image smoothing via L gradient minimization. *ACM Transactions on Graphics* Vol. 30, No. 6, Article No. 174, 2011.
- [16] Ono, S. L gradient projection. *IEEE Transactions on Image Processing* Vol. 26, No. 4, 1554–1564, 2017.
- [17] Ono, S. Edge-preserving filtering by projection onto L gradient constraint. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 1492–1496, 2017.
- [18] Xu, L.; Yan, Q.; Xia, Y.; Jia, J. Y. Structure extraction from texture via relative total variation. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 139, 2012.
- [19] Chen, Q. F.; Xu, J.; Koltun, V. Fast image processing with fully-convolutional networks. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2516–2525, 2017.
- [20] Xu, L.; Ren, J.; Yan, Q.; Liao, R.; Jia, J. Deep edge-aware filters. In: *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37, 1669–1678, 2015.
- [21] Kim, Y.; Ham, B.; Do, M. N.; Sohn, K. Structure-texture image decomposition using deep variational priors. *IEEE Transactions on Image Processing* Vol. 28, No. 6, 2692–2704, 2019.
- [22] Zhao, M.; Zhang, H. X.; de Sun, J. A novel image retrieval method based on multi-trend structure descriptor. *Journal of Visual Communication and Image Representation* Vol. 38, 73–81, 2016.

- [23] Zhang, F.; Li, J. J.; Liu, P. Q.; Fan, H. Computing knots by quadratic and cubic polynomial curves. *Computational Visual Media* Vol. 6, No. 4, 417–430, 2020.
- [24] Liu, X. X.; Zhang, Y. F.; Bao, F. X.; Shao, K.; Sun, Z. Y.; Zhang, C. M. Kernel-blending connection approximated by a neural network for image classification. *Computational Visual Media* Vol. 6, No. 4, 467–476, 2020.
- [25] Wang, Y. L.; Yang, J. F.; Yin, W. T.; Zhang, Y. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences* Vol. 1, No. 3, 248–272, 2008.
- [26] Gu, S. H.; Xie, Q.; Meng, D. Y.; Zuo, W. M.; Feng, X. C.; Zhang, L. Weighted nuclear norm minimization and its applications to low level vision. *International Journal of Computer Vision* Vol. 121, No. 2, 183–208, 2017.
- [27] Eckstein, J.; Bertsekas, D. P. On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming* Vol. 55, Nos. 1–3, 293–318, 1992.
- [28] Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning* Vol. 3, No. 1, 1–122, 2011.
- [29] Zhang, R.; Kwok, J. Asynchronous distributed ADMM for consensus optimization. In: Proceedings of the 31st International Conference on International Conference on Machine Learning, Vol. 32, II-1701–II-1709, 2014.
- [30] Sun, D. L.; Févotte, C. Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 6201–6205, 2014.
- [31] Zhang, M. L.; Desrosiers, C. High-quality image restoration using low-rank patch regularization and global structure sparsity. *IEEE Transactions on Image Processing* Vol. 28, No. 2, 868–879, 2019.
- [32] Ma, G. H.; Zhang, M. L.; Li, X. M.; Zhang, C. M. Image smoothing based on image decomposition and sparse high frequency gradient. *Journal of Computer Science and Technology* Vol. 33, No. 3, 502–510, 2018.
- [33] Zhu, F. D.; Liang, Z. T.; Jia, X. X.; Zhang, L.; Yu, Y. Z. A benchmark for edge-preserving image smoothing. *IEEE Transactions on Image Processing* Vol. 28, No. 7, 3556–3570, 2019.
- [34] Jian, M. W.; Zhang, W. Y.; Yu, H.; Cui, C. R.; Nie, X. S.; Zhang, H. X.; Yin, Y. Saliency detection based on

directional patches extraction and principal local color contrast. *Journal of Visual Communication and Image Representation* Vol. 57, 1–11, 2018.

Xiang Ma is currently a Ph.D. student in the School of Software, Shandong University, Jinan. He received his master degree from Shandong University in 2020. His research interests include image processing and information visualization.

information visualization.

Xuemei Li received her master and doctor degrees from Shandong University in 2004 and 2010, respectively. She is currently an associate professor in the School of Software, Shandong University, and a member of the GDIV Lab. She is engaged in research on geometric modeling, CAGD, image processing, and

professor with the School of Software, Shandong University, where he is also a member of the GDIV Lab. His current research interests include computer graphics, information visualization, and image processing.

Yuanfeng Zhou received his master and Ph.D. degrees from the School of Software, Shandong University, in 2005 and 2009, respectively. He held a post-doctoral position with the Graphics Group, Department of Computer Science, University of Hong Kong, from 2009 to 2011. He is currently a

and 1984, respectively, and his Dr.Eng. degree in computer science from the Tokyo Institute of Technology, Japan, in 1994. From 1997 to 2000, he held a visiting position at the University of Kentucky, USA. His research interests include CAGD, CG, information visualization, and medical image processing.

Caiming Zhang is a professor and doctoral supervisor of the Software College at Shandong University. He is now also the dean of the Digital Media Research Institute at Shandong University of Finance and Economics. He received his B.S. and M.E. degrees in computer science from Shandong University in 1982

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made.

The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material

is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Other papers from this open access journal are available free of charge from <http://www.springer.com/journal/41095>. To submit a manuscript, please go to <https://www.editorial-manager.com/cvmj>.