Pixel-level Non-local Image Smoothing with Objective Evaluation

Zhi-Ang Liu, Ying-Kun Hou, Xian-Tong Zhen, Jun Xu, Ling Shao, Ming-Ming Cheng

***Abstract*—Recently, image smoothing has gained increasing attention due to its prerequisite role in other image processing tasks, e.g., image enhancement and editing. However, the evaluation of image smoothing algorithms is usually performed by subjective observation on images without corresponding ground truths. To promote the development of image smoothing algorithms, in this paper, we construct a novel Nankai Smoothing (NKS) dataset containing 200 images blended by versatile structure images and natural textures. The structure images are inherently smooth and naturally taken as ground truths. On our NKS dataset, we comprehensively evaluate 14 popular image smoothing algorithms. Moreover, we propose a Pixel-level Non-Local Smoothing (PNLS) method to well preserve the structure of the smoothed images, by exploiting the pixel-level non-local self-similarity prior of natural images. Extensive experiments on several benchmark datasets demonstrate that our PNLS outperforms previous algorithms on the image smoothing task. Ablation studies also reveal the work mechanism of our PNLS on image smoothing. To further show its effectiveness, we apply our PNLS on several applications such as semantic region smoothing, detail/edge enhancement, and image abstraction. The dataset and code are available at https://github.com/zal0302/PNLS.**

***Index Terms*—Image smoothing, benchmark dataset, performance evaluation, pixel-level non-local self similarity.**

I. Introduction

IMAGE smoothing is an important multimedia technology, aiming to decompose an image into a piece-wise smooth layer and a texture layer [11]. The smooth layer reflects the structural content of the image, while the texture layer presents the residual details in the image. The decomposed layers can be manipulated separately and recomposed in different ways to fulfill specific applications such as image enhancement [19], [42] and image abstraction [5], [51].

In the last decade, numerous image smoothing algorithms have been proposed from the perspectives of local filters [28], [31], [34], global filters [30], [52], [55], and deep filters [18], [24], [50], *etc*. Local filters smooth the input image by averaging pixel intensity in locally weighted manners [16], [31], [34]. They are computationally efficient, but produce gradient reversals and halo artifacts especially on edges [16]. Global filters [11], [52], [55] attenuate the reversals and artifacts by implementing optimization on the whole image in a principled manner. However, the global filters are usually time and memory consuming [56]. Deep filters [18], [24], [50] train smoothing networks with pairs of natural and "ground-truth" images. But the "ground-truth" images are usually generated by other image smoothing methods [49], [56], hindering the deep filters from domain generalized performance.

Fig. 1. Examples of our Nankai Smoothing (NKS) dataset.

Despite their promising performance, these image smoothing algorithms could hardly be objectively evaluated due to the lack of reasonable benchmarks. Although several datasets are collected for the image smoothing task [27], [49], [56], their "ground truths" are either generated by other image smoothing methods [49], [56] or blended with cartoon images and synthetic textures [27]. On one hand, the "ground truths" generated by existing smoothing methods are highly biased. Thai is, the metric results computed on "ground truths" do not reflect the smoothing performance of the smoothing method, but the closeness of its results to the "ground truths" generated by several smoothing methods. On the other hand, the algorithms trained on cartoon images and synthetic textures could not perform consistently well on smoothing out natural textures distinct from the training data.

To promote the development of image smoothing algorithms, in this work, we construct a novel Nankai Smoothing (NKS) dataset with 200 versatile images blended by structure images (ground truths) and natural textures. Some examples of our NKS dataset are illustrated in Figure 1. On our NKS dataset, we benchmark 14 popular image smoothing algorithms, and present an extensive performance analysis with commonly used metrics. Furthermore, we propose a Pixel-level Non-Local Smoothing (PNLS) method, based on the pixel-level non-local self similarity (NSS) prior of natural images [17]. Extensive experiments on several benchmark datasets (including our NKS) demonstrate that our PNLS achieves better performance on subjective visual quality (and

This work is supported by "the Fundamental Research Funds for the Central Universities", Nankai University (63201168) and Project (92022104).

ZA Liu, J Xu, and MM Cheng are with TKLNDST, College of Computer Science, Nankai University, Tianjin, China. YK Hou is with School of Information Science and Technology, Taishan University, Tai'an, China. XT Zhen and L Shao are with Inception Institute of Artificial Intelligence and Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE. J Xu (nankaimathxujun@gmail.com) is the corresponding author.

objective metrics) than previous image smoothing methods. To show its broad practicality, we apply our PNLS onto three image processing tasks: salient region smoothing, image detail/edge enhancement, and image abstraction, *etc*.

In summary, our major contributions are manifold:

- **We construct a Nankai Smoothing (NKS) dataset** containing 200 images blended by structure and texture images. The structure images are naturally taken as ground truths for evaluating image smoothing methods. We also benchmark 14 popular image smoothing algorithms on our NKS dataset.
- **We propose a novel Pixel-level Non-Local Smoothing (PNLS) method** by exploiting the pixel-level non-local self similarity (NSS) prior of natural images.
- **Experiments shows that our PNLS achieves promising image smoothing performance** on several benchmark datasets, via subjective evaluation and objective metrics. We also show the broad practicality of our PNLS by applying it on diverse image processing tasks.

The remainder of this paper is organized as follows. In §II, we review the related work. In §III, we introduce the Nankai Smoothing (NKS) dataset, and benchmark 14 popular image smoothing algorithms on it. We then present our PNLS smoothing method in §IV. In §V, we perform extensive experiments on several datasets to demonstrate the advantages of our PNLS over previous smoothing methods. We also provide more applications of our PNLS on several image processing tasks in §VI. The Conclusion is given in §VII.

II. Related Work

A. Image Smoothing Methods

Local filters explicitly filter each pixel as a weighted average of its neighborhood pixels in an one-step or iterative way. Bilateral filter (BF) [28] is a simple and intuitive method in this category, and widely applied in other image processing tasks [13], [21], [41]. However, since not all pixels have enough similar pixels around, the weighted average would be biased by outlier pixels, thus resulting gradient reversal artifacts [16]. It is also generalized to Joint Bilateral Filter (JBF) [31], in which the weights are computed upon another guidance image instead of the input image itself. The insights of resorting to a guidance image is later flourished in the Guided Filter (GF) [16]. GF has inspired numerous methods due to its $\mathcal{O}(N)$ complexity for an image with N pixels. However, it cannot resolve the ambiguity regarding whether or not to smooth certain edges. With the help of ℓ_0 gradient minimization [49] for correction, Su *et al.* [34] uses a degradation scheme to smooth small-scale textures and a joint bilateral filter for suppressing textures.

Global filters [11], [49], [51] attenuate the limitations of local filters such as gradient reversals and halo artifacts [11]. These methods solve an optimization function in a principled manner on the whole image. The function is usually consisted of a fidelity term for data fitting and a prior term for regularizing smoothness. Among these methods, Weighted Least Square (WLS) [11] adjusts the matrix affinities according to the image gradient and produces halo-free smoothing results. Later, a semi-global extension of WLS [25] is proposed to solve the linear system in a time and memory efficient manner. The ℓ_0 gradient minimization (L0) [49] globally controls the number of non-zero gradients which are involved in approximating the prominent structure of input image. However, one unavoidable problem is that they are prone to over-sharpen the edges while smoothing the details [11], [25], [49]. The Rolling Guidance Filtering (RGF) [54] filters images with the complete control of detail smoothing under a scale measure, employing BF for filtering with a rolling guidance implemented in an iterative manner. Zhou *et al.* [55] proposed an iterative optimization filter to selectively suppress the gradient on the features of smaller scales, while retaining the large-scale intensity variations in limited iterations. In short, global filters are usually computationally expensive, and often sacrifice the local edge-preserving effects for better global performance.

Deep filters mostly focus on accelerating while approximating state-of-the-art local or global filters such as BF [28] or WLS [11]. Deep Edge-Aware Filter (DEAF) [50] is a pioneering work in this category. It trains the network in the gradient domain, and reconstructs the filtered output from the refined gradients produced by the deep network. In [24], a hybrid neural network is proposed based on the recursive filters whose coefficients can be learned by a deep network. Li *et al.* [18] proposed a learning-based approach to construct a joint filter based on convolutional neural networks (CNNs). Shen *et al.* [33] introduces a convolutional neural pyramid to extract features of different scales, aiming at extracting larger receptive fields from input images. The work of [5] utilizes context aggregation networks to include more contextual information. Lu *et al.* [27] developed a structure and texture dataset and trained a texture and structure aware network, hence enabling their method the awareness. The work of [10] introduces an unsupervised learning CNN that facilitates generating flexible smoothing effects. One common issue is that all these approaches take the output of existing filters as "ground-truth", and hence can hardly outperform these "teacher" filters.

B. Image Smoothing Benchmarks

Datasets. There are several datasets of BSDS500 [29], DIV2K [1], MIT5K [3] originally collected for the image segmentation [40], super-resolution [20], and image denoising [32] tasks, but are also used to present image smoothing performance. But these images do not have corresponding smooth ground-truths. A dataset is published with the proposed image smoothing algorithm RTV [51], but similarly this dataset does not provide ground-truths. Zhu *et al.* [56] proposed a benchmark for image smoothing. Unfortunately, the ground-truth smoothed images are generated by existing smoothing algorithms. These "ground-truths" are prone to be subjective since in fact we could only evaluate the performance difference between a new algorithms and these handpicked existing algorithms. In our Nankai Smoothing (NKS) dataset,

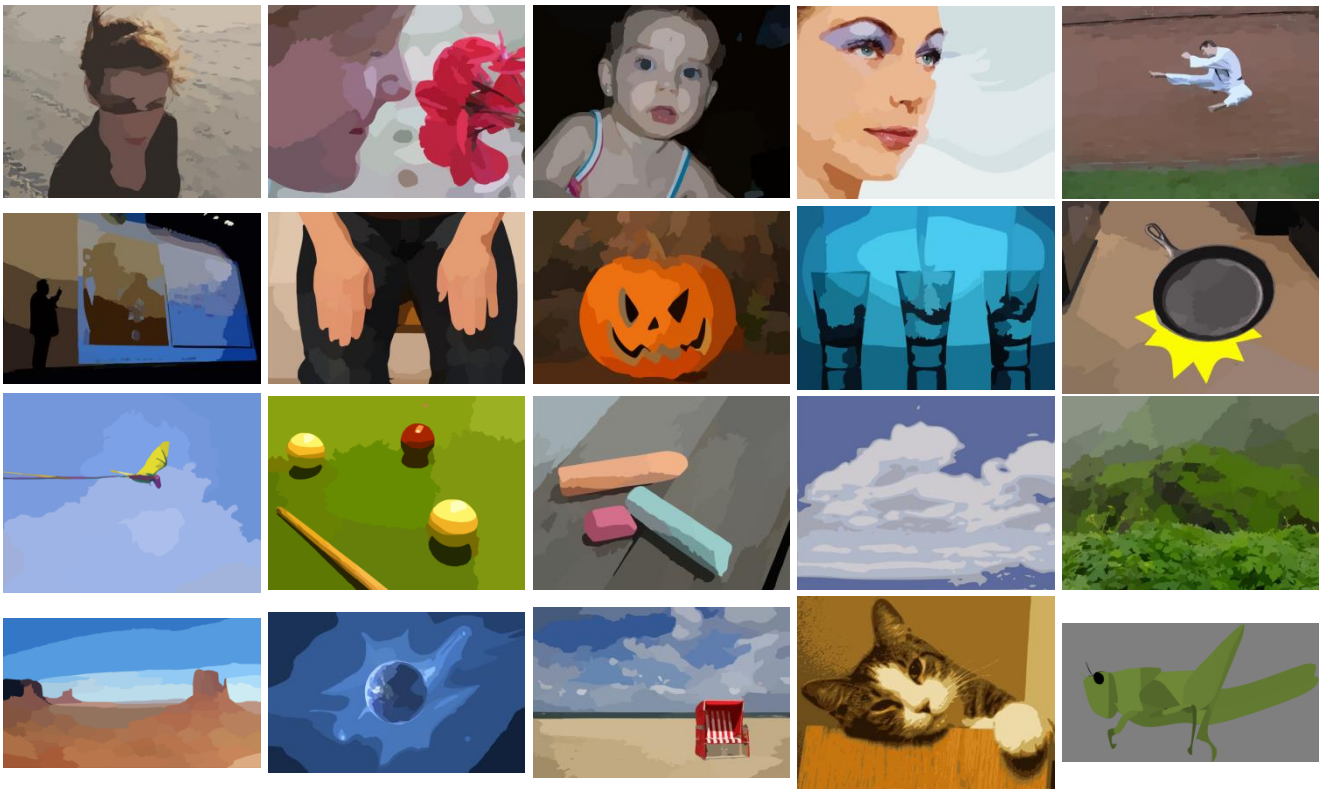


Fig. 2. The 20 structure images we used in our NKS dataset.

Fig. 3. The 10 natural texture images we used in our NKS dataset.

we collect the structure images as ground-truths and acquire samples by blending images.

Evaluation metrics. There are many meaningful evaluation metrics for image quality measurement. Peak-Signal-to-Noise-Ratio (PSNR) is an widely used objective metric in image restoration tasks, computing the error between the original image and the distorted image. However, PSNR focuses on the pixel-level difference between two images, while ignoring their similarity of visual characteristics [39]. To fill in this gap, the Structural Similarity index (SSIM) [39] was developed to comprehensively measure image similarity from the aspects of brightness, contrast and structure. SSIM takes into account the correlation of structural patches instead of pixels, and hence is more in line with human eye's judgment on image quality. Since not all pixels in an image are of the same importance, the Feature Similarity index (FSIM) [53] uses low-level features to evaluate the distance between the reference image and the distorted image. One common evaluation manner is to subjectively evaluate the smoothed image through visual quality, but this manner may lack accurate measurements. Although Weighted Root Mean Squared Error (RMSE) and Weighted Mean Absolute Error (WMAE) are used in [56], they may be stuck by the "ground truths" produced by existing smoothing methods. In this paper, we employ PSNR, SSIM [39] and FSIM [53] as the evaluation metrics due to their consistent performance with the visual perception of humans.

III. Proposed Nankai Smoothing Dataset

In this section, we develop a Nankai Smoothing (NKS) dataset for the image smoothing task, and evaluate 14 state-

of-the-art algorithms on it with three commonly used metrics, i.e., PSNR, SSIM [39], and FSIM [53].

A. Constructing the NKS Image Smoothing Dataset

Motivation. Since manually annotating the structure of an image is subjective and costly, extracting structural ground truths directly from natural images is difficult. In fact, image smoothing is very close to the image denoising task [22], [23], [37]: both aim to filter out small scale components (textures or noise) from the image. Despite the collection of ground truths is ambiguous for image smoothing [27], [56], constructing image denoising datasets [36], [44], [45] is explicitly feasible. That is, we add the noise to the clean image, and recover it from the synthetic noisy image by removing the noise [43], [46]–[48]. In evaluation, the corresponding clean images are naturally taken as the ground truths to compute the objective metrics such as PSNR and SSIM [39]. Similarly, in image smoothing we can blend structure and texture images to generate the test images, and the structure images can be recovered by removing the texture from the blended image.

Though previous work [27] had worked towards this direction, it suffers from two limitations: 1) the synthetic textures are not natural as real-world color images; 2) the dataset is not publicly released, making it difficult for others to evaluate novel smoothing methods on this dataset.

Collecting structure and texture images. As shown in Figure 2, we observe that vector images are smooth and can be reasonably taken as structure images, and construct the NKS dataset by blending vector images and texture images. Specifically, we search the key word of "vector" on the Pixabay website [7] and select 20 highly realistic vector images from thousands of free structure ones. In addition, we manually select 10 natural texture images from the Pixabay website [7]. The selected structure images and natural textures are shown in Figures 2 and 3, respectively. We observe that the vector images are smooth with clear structures, and thus can be taken as ground truths in image smoothing.

Generating blended images. To generate mixed structure and texture images, we blend each of the 20 structure images and each of the 10 natural textures in proper proportions.We set the proportion of structure images from 0.7 to 0.85 for ensuring every fused image is real enough. This process is closely similar to the generation of noisy images in image denoising [43], [48], except that we need to set proper proportions of structure and texture images to make the mixed image natural in visual quality. The reason is that, directly adding the structure and texture images together would result in overflow of pixel values, as well as unnatural looking of mixed images. The structure images are taken as the ground truths for the corresponding images blended by the structure images and the 10 natural textures. In this way, we collect overall $20 \times 10 = 200$ images in our NKS dataset, with 20 structure images as the ground-truths.

Dataset statistics. Our NKS dataset contains 200 images of versatile scenarios. We show the structure and texture images in Figures 2 and 3, from which one can see that our NKS dataset consists of diverse contents, such as Human (*e.g.*, children, women,...), Artifact (*e.g.*, pot, chalk,...), Landscape (*e.g.*, forest, beach,...), and Animal (*e.g.*, cat, insect,...), *etc*. We also present the statistics of our NKS dataset in Table I. All images are resized into the width of 512 with proportionally heights, by the default Matlab function "imresize". The numbers of different classes are 70 for Human, 50 for Artifact, 60 for Landscape, and 20 for Animal.

TABLE I
SUMMARY OF OUR NKS SMOOTHING DATASET.

Class	Size		Number	Structures	Textures
	Width	Height			
Human	512	340 $\sim$ 384	70	Vector	Carpet, Wood, ...
Artifact	512	343 $\sim$ 397	50		
Landscape	512	298 $\sim$ 384	60		
Animal	512	277 $\sim$ 384	20		

TABLE II
COMPARISON OF AVERAGE PSNR, SSIM [39], AND FSIM [53] BY 14 STATE-OF-THE-ART IMAGE SMOOTHING ALGORITHMS ON OUR NKS DATASET. THE AVERAGE RUNNING TIME (IN SECONDS) OF THESE METHODS (ON CPU OR *GPU*) IS REPORTED ON THE 110 IMAGES OF SIZE 512×384 IN OUR NKS DATASET. OUR METHOD WILL BE INTRODUCED IN §IV. "PUB." MEANS "PUBLICATION VENUES".

	No.	Method	PSNR	SSIM	FSIM	Device	Time
Traditional Filters	1	BF [28]$_{ICCV'98}$	32.00	0.8478	0.8556	CPU	1.53
	2	WLS [11]$_{TOG'08}$	28.59	0.9011	0.9107	CPU	0.91
	3	EAW [12]$_{TOG'09}$	28.02	0.7953	0.8200	CPU	**0.02**
	4	L0 [49]$_{TOG'11}$	33.01	0.9249	0.9374	CPU	0.48
	5	RTV [51]$_{TOG'12}$	31.81	0.9206	0.9234	CPU	0.68
	6	GF [16]$_{TPAMI'13}$	32.09	0.8779	0.8672	CPU	**0.03**
	7	TF [2]$_{TIP'13}$	33.23	0.9186	0.9149	CPU	0.33
	8	FGS [30]$_{TIP'14}$	23.46	0.8368	0.7978	CPU	**0.03**
	9	RGF [54]$_{ECCV'14}$	32.51	0.9135	0.9128	CPU	0.22
	10	fastABF [14]$_{TIP'18}$	31.44	0.8977	0.8917	CPU	0.37
Deep Filters	11	LRNN [24]$_{ECCV'16}$	30.61	0.8666	0.8600	CPU	0.43
	12	FIP [5]$_{ICCV'17}$	32.03	0.8946	0.9061	*GPU*	0.45
	13	VDCNN [56]$_{TIP'19}$	33.38	0.9349	0.9395	*GPU*	1.47
	14	ResNet [56]$_{TIP'19}$	33.13	0.9354	**0.9434**	*GPU*	3.76
Ours	15	**PNLS (Fast)**	**33.45**	**0.9378**	0.9397	CPU	5.10
	16	**PNLS (Slow)**	**33.68**	**0.9420**	**0.9440**	CPU	78.68

B. Benchmarking Image Smoothing on our NKS dataset

Comparison methods. We evaluate 14 image smoothing algorithms on NKS dataset in total. These algorithms include 10 traditional filters: BF [28], WLS [11], EAW [12], GF [16], L0 [49], RTV [51], TF [2], FGS [30], RGF [54], fastABF [14] and 4 deep filters: LRNN [24], FIP [5], two baselines ResNet and VDCNN used in [56]. We employ the commonly used

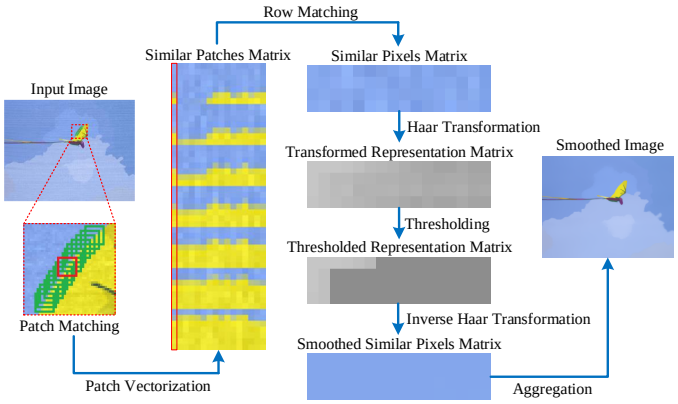


Fig. 4. **Flowchart of our PNLS smoothing method**. First, we select multiple reference patches for smoothing in the input image. For a separate reference patch (shown in the red bounding box), the Euclidean distance is used to perform patch matching and row matching to obtain the Similar Pixels Matrix. Then, the three channels of the Similar Pixels Matrix are transformed by using the Haar transformation, and the threshold is used for smoothing. Then the Thresholded Representation Matrices of the three channels are to inverse Haar transformation to obtain the Smoothed Similar Pixels Matrix. Finally, we aggregate it back into the image to complete the smoothing process of a single reference patch. The above process is performed for all selected reference patches to complete one iteration of image smoothing. We perform our PNLS several iterations to improve its performance.

metrics of PSNR, SSIM [39], and FSIM [53] to quantitatively evaluate the performance of the comparison methods on our NKS dataset. These metrics measure the distance between the smoothed images and the corresponding ground-truths.

Results. The comparison results are listed in Table II. One can see that, on the three metrics, TF [2] performs best among the traditional (local and global) filters, while the baselines VDCNN and ResNet in [56] performs better than the other two deep filters. We also report the average running time (in seconds) of different methods on the 110 images of size 512×384 in our NKS dataset. Specifically, FIP [5], two baselines ResNet and VDCNN used in [56] are tested on an NVIDIA GTX 1080 GPU and the other filters are tested on the Intel Core i7-6700K CPU. One can see that EAW [12], GF [16] and FGS [30] averagely take 0.02, 0.03, and 0.03 seconds to process a 512×384 image, much faster than the other filters.

IV. Proposed Pixel-level Non-local Smoothing

In this section, we present the proposed Pixel-level Non-Local Smoothing (PNLS) method, which is consisted of three steps: 1) searching non-local similar pixels (§IV-A); 2) estimating the smoothing threshold (§IV-B); and 3) smoothing by Haar transformation based thresholding (§IV-C). The flowchart of our PNLS is plotted in Figure 4. Note that we first transform an RGB image into the luminance-chrominance space [8], and get the corresponding YCbCr image. We search similar pixels and estimate the smoothing threshold in the Y channel. The similar pixels of the Cb and Cr channels are grouped according to results in the Y channel. Then, we perform image smoothing by threshold based Haar transformation on each channel. Finally, we transform the smoothed image in YCbCr space back to the RGB space.

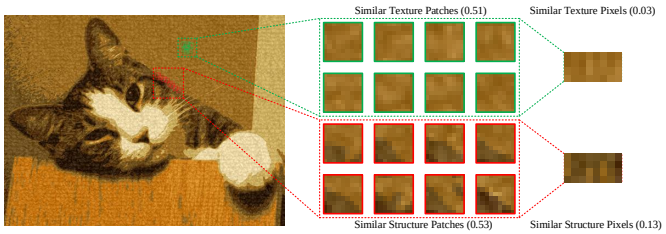


Fig. 5. **Importance of pixel-level smoothing.** The standard deviation (std) of similar patches belonging to the texture (0.51) and that of similar patches belonging to the structure (0.53) are very close. But the std of similar pixels in texture (0.03) is much smaller than that of similar pixels in structure area (0.13). Therefore pixel-level smoothing could well distinguish texture and structure area in a mixed image.

A. Searching Non-local Similar Pixels

For the input image $\boldsymbol{I} \in \mathbb{R}^{h \times w}$, we extract reference patches of size $m \times m$ with a step of s (horizontally and vertically) from its Y channel. For each reference patch, we first search its similar patches in a window of size $R \times R$. The similarity is measured by Euclidean distance. Then we reshape each similar patch into a vector $\boldsymbol{v}_r \in \mathbb{R}^{m^2}$ ($r = 1, ..., R^2$, $\boldsymbol{v}_1$ is the reference patch). We perform patch matching by selecting the q closest patches (including $\boldsymbol{v}_1$ itself) to $\boldsymbol{v}_1$. By stacking the q vectors in columns, we get the similar patches matrix $\boldsymbol{P} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_q] \in \mathbb{R}^{m^2 \times q}$.

However, patch matching could not well distinguish textures from structures. To illustrate this point, in Figure 5, we extract similar texture patches of size 9×9 from the texture area (green boxes) and the structure area (red boxes), respectively. The standard deviation (std) of similar patches in texture area and that of similar patches in structure area are very close, i.e., 0.51 and 0.53, respectively. To better distinguish structure and texture area, we propose to perform pixel-level row matching to extract similar pixels. We extract 4 similar pixel groups by row matching, as described in Figure 4, to form similar pixel matrix of size 4×8 from similar texture and structure patches, respectively. Figure 5 shows that the std of similar pixel groups in texture area (0.03) is much smaller than the std of similar pixel groups in structure area (0.13), indicating that pixel-level similarity well distinguishes the texture area from structure one. We will set a smoothing threshold to help smooth images accurately, which will be introduced in §IV-B.

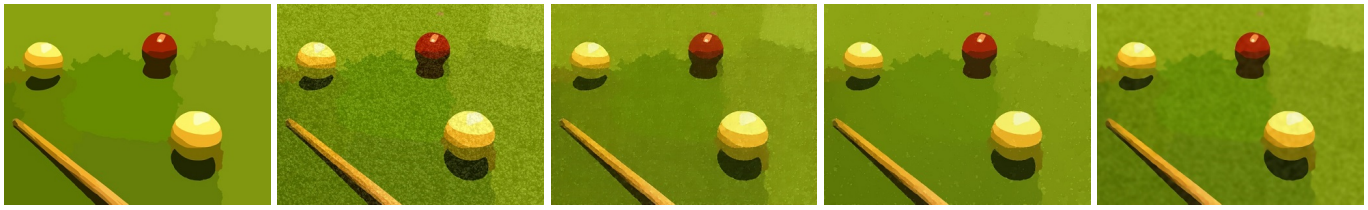
For row matching of similar pixel groups, we take the i-th row $\boldsymbol{v}^i \in \mathbb{R}^q$ ($i = 1, ..., m^2$) of $\boldsymbol{P}$ as the reference row, and calculate the Euclidean distances between the reference row $\boldsymbol{v}^i$ and the other rows $\{\boldsymbol{v}^j \in \mathbb{R}^q,\ j = 1, ..., m^2\}$, as follows:

$$d^{ij} = \|\boldsymbol{v}^i - \boldsymbol{v}^j\|_2. \tag{1}$$

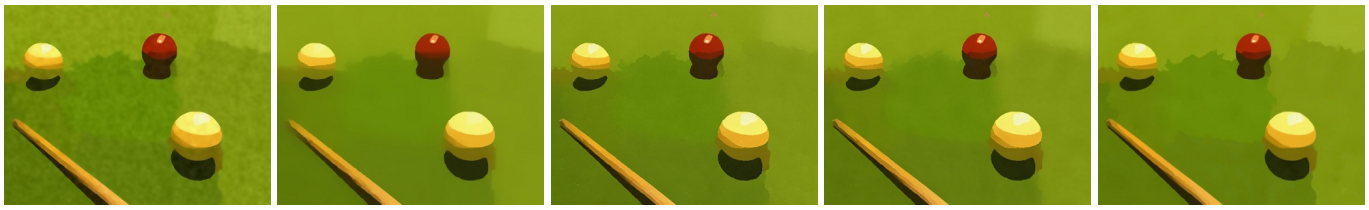
We select the p rows of pixels with the minimal distances to the reference row $\boldsymbol{v}^i$, and form the similar pixels matrix $\boldsymbol{S} = [\boldsymbol{v}^{i_1}, ..., \boldsymbol{v}^{i_q}] \in \mathbb{R}^{p \times q}$. Note that we have $\boldsymbol{v}^{i_1} = \boldsymbol{v}^i$ and $d^{ii_1} = 0$. The similar pixels matrices in Cb and Cr channels of the matrix $\boldsymbol{P}$ are extracted corresponding to the Y channel.

B. Estimation of Smoothing Threshold

As shown in Figure 5, pixels in similar patches of the structure in image would suffer from larger stds than those



(a) Ground Truth | (b) Input Image 27.26/0.4367/0.6491 | (c) FIP [5] 32.74/0.8250/0.9037 | (d) L0 [49] 35.00/0.8861/0.9540 | (e) RGF [54] 34.45/0.8914/0.9097



(f) TF [2] 35.47/0.9104/0.9195 | (g) RTV [51] 33.67/0.9112/0.9611 | (h) ResNet [56] 34.03/0.8989/0.9704 | (i) VDCNN [56] 34.90/0.9110/0.9671 | (j) Ours **36.90/0.9309/0.9803**

Fig. 6. Comparison of smoothed images and PSNR(dB)/SSIM/FSIM results by different methods on the image "*S15_T01*" from our NKS dataset. The best results are highlighted in **bold**.

in the texture area. Then a threshold is essential to determine whether or not (and the extent to which) smooth each similar pixels matrix for distinguishing structure and texture. Since the pixels in the p rows of S are very close, we can consider the std as the energy estimation due to texture changes and compute it as

$$\sigma = \frac{1}{m^2(p-1)\sqrt{q}} \sum_{t=2}^{p} \sum_{i=1}^{m^2} d^{ii_t}. \tag{2}$$

To perform consistent image smoothing, we set a global threshold as the average σ of all similar pixels matrices.

C. Smoothing by Haar Transformation based Thresholding

In §IV-A, we have obtained a set of similar pixels matrices $\boldsymbol{S} \in \mathbb{R}^{p\times q}$ and the threshold σ. We then employ the Haar transformation [15] for thresholding similar pixels matrices. Specifically, we utilize the Haar transformation with lifting scheme [9], [35], which includes vertical transformation matrix $\boldsymbol{H}_l \in \mathbb{R}^{p\times p}$ and horizontal transformation matrix $\boldsymbol{H}_r \in \mathbb{R}^{q\times q}$. In order to perform Haar transformation, we set p, q to be powers of 2. The detailed transformation process is provided in the *Supplementary File*. Thus, we get the transformed representation matrix $\boldsymbol{T} \in \mathbb{R}^{p\times q}$:

$$\boldsymbol{T} = \boldsymbol{H}_l \boldsymbol{S} \boldsymbol{H}_r. \tag{3}$$

By using the smoothing threshold, we could restore the element in i-th ($i = 1, ..., q$) row, j-th ($j = 1, ..., m^2$) column of the transformed representation matrix $\boldsymbol{S}$ via

$$\hat{\boldsymbol{T}} = \boldsymbol{T} \odot \mathbb{I}_{\{|\boldsymbol{T}| \geq \lambda\sigma^2\}}, \tag{4}$$

where $\odot$ means element-wise production, $\mathbb{I}$ is the indicator function, and λ is the parameter controlling the extent to threshold. According to the wavelet theory [35], the elements in the last two rows of $\boldsymbol{T}$ (except the 1-st column) belongs to high frequency bands of the Haar transformation, and these elements are texture information. We directly set these elements as zero in $\hat{\boldsymbol{T}}$:

$$\widetilde{\boldsymbol{T}}(i,j) = \hat{\boldsymbol{T}}(i,j) \odot \mathbb{I}_{\{\text{if } i=1,\ldots,q-2 \text{ or } j=1\}}, \tag{5}$$

where $\widetilde{\boldsymbol{T}}(i,j)$ and $\hat{\boldsymbol{T}}(i,j)$ are the elements in i-th ($i = 1, ..., q$) row, j-th ($j = 1, ..., m$) column of matrices $\widetilde{\boldsymbol{T}}$ and $\hat{\boldsymbol{T}}$, respectively. We then use the vertical inverse Haar transformation matrix $\boldsymbol{H}_{il} \in \mathbb{R}^{p\times p}$ and horizontal inverse Haar transformation matrix $\boldsymbol{H}_{ir} \in \mathbb{R}^{q\times q}$ on thresholded representation matrix $\widetilde{\boldsymbol{T}}$. The detailed inverse transformation process is also provided in the *Supplementary File*. Then we could get the smoothed similar pixels matrix $\widetilde{\boldsymbol{S}}$ without texture via

$$\widetilde{\boldsymbol{S}} = \boldsymbol{H}_{il} \widetilde{\boldsymbol{T}} \boldsymbol{H}_{ir}. \tag{6}$$

Finally, the smoothed similar pixels matrices are aggregated to the corresponding positions in the original image. The above is the detailed process of image smoothing based on Haar transformation techniques [15].

D. Iterative Smoothing Scheme

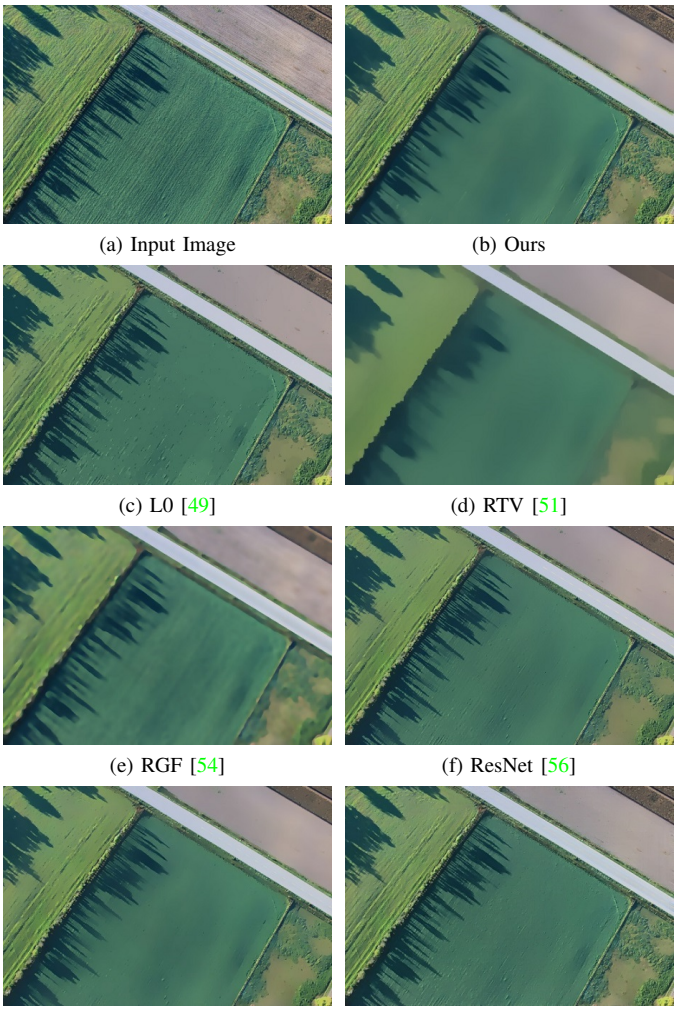
For better performance, we apply the above smoothing process for $N = 10$ iterations. Experiments show that our PNLS method with $N = 10$ achieves satisfactory smoothing results (please refer to §V for more details).

E. Complexity Analysis

The proposed PNLS method contains three parts: 1) in §IV-A, the complexity of patch matching is $\mathcal{O}(whR^2m^2/s^2)$, while the complexity of row matching is $\mathcal{O}(whqm^4/s^2)$; 2) in §IV-B, the complexity for smoothing threshold estimating is $\mathcal{O}(whm^2p/s^2)$; 3) in §IV-C, the complexity for Haar transformation based thresholding is $\mathcal{O}(whpqm^2/s^2)$. Since the above process iterates N times, the complexity of our PNLS is $\mathcal{O}(whm^2N/s^2 \cdot \max\{R^2, qm^2, pq\})$.

V. Experiments

In this section, we first compare the proposed Pixel-level Non-Local Smoothing (PNLS) method with competitive methods on several image smoothing benchmark datasets. We also



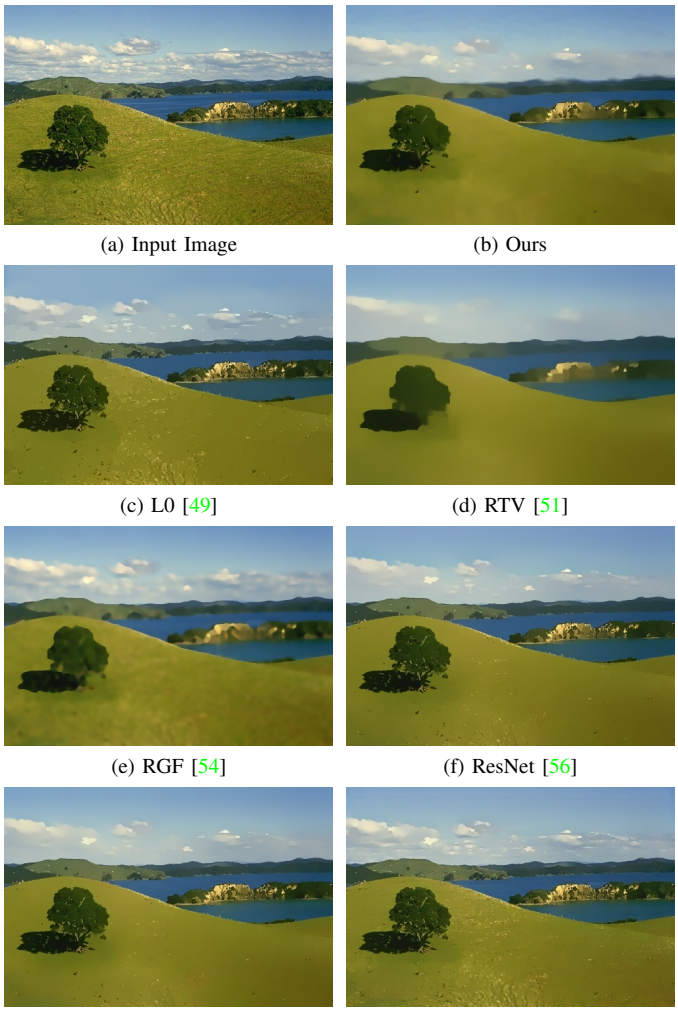
(a) Input Image (b) Ours

(c) L0 [49] (d) RTV [51]

(e) RGF [54] (f) ResNet [56]

(g) VDCNN [56] (h) FIP [5]

Fig. 7. Comparison of smoothed images by different methods on the image "*0261*" from the DIV2K dataset [1].



(a) Input Image (b) Ours

(c) L0 [49] (d) RTV [51]

(e) RGF [54] (f) ResNet [56]

(g) VDCNN [56] (h) FIP [5]

Fig. 8. Comparison of smoothed images by different methods on the image "*0251*" from the dataset [56].

perform comprehensive ablation studies to gain deeper insights into the proposed PNLS method. More comparison results of visual quality can be found in the *Supplementary File*.

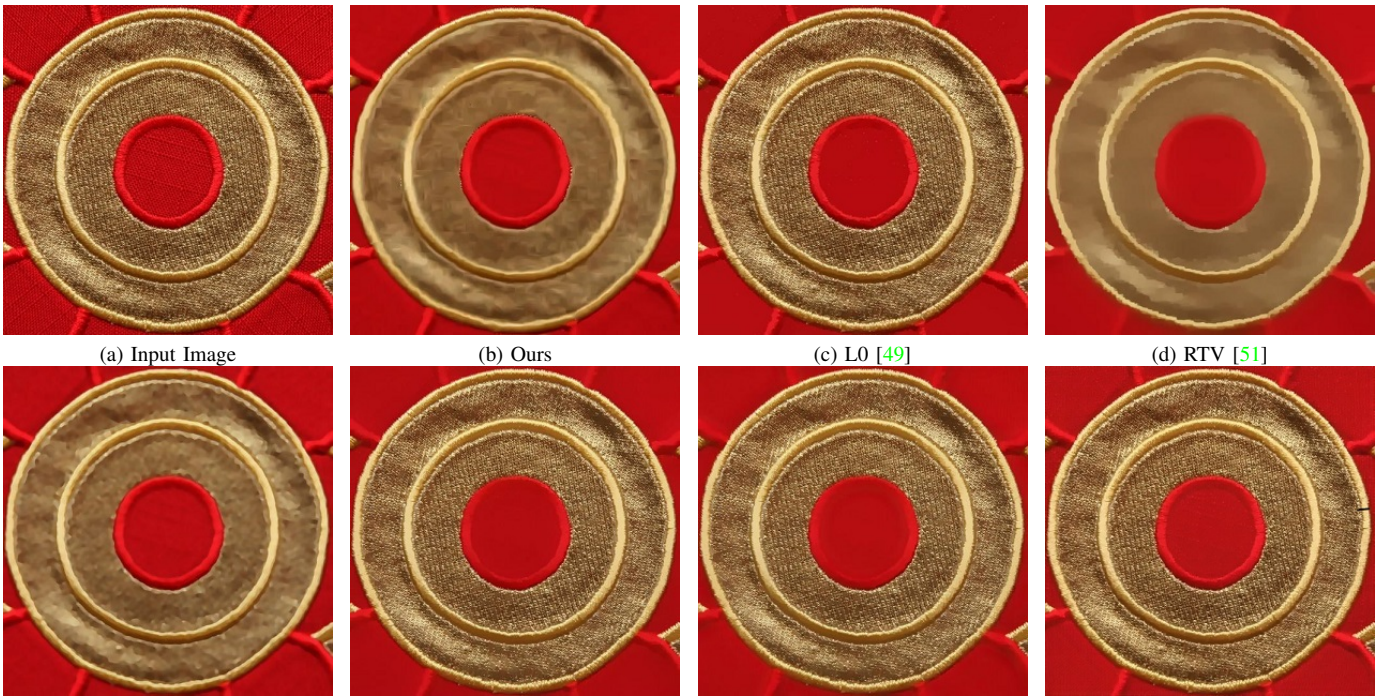
A. Implementation Details

Parameter settings. As shown in Table II, we have two versions of PNLS: fast PNLS (No. 15) and slow PNLS (No. 16). The parameters of our fast PNLS include the size $R = 15$ of searching window, the step $s = 4$ for extracting neighbor reference patches, the patch size $m = 4$, the iteration number $N = 10$, the threshold parameter $\lambda = 0.4$. For our slow PNLS, the parameters are almost the same with those of fast PNLS, but the step is set as $s = 1$ (rather than $s = 4$ in fast PNLS) to extract more reference patches.

Comparison methods. We compare the proposed PNLS method with 14 state-of-the-art image smoothing methods: BF [28], WLS [11], EAW [12], GF [16], L0 [49], RTV [51], TF [2], FGS [30], RGF [54], fastABF [14], LRNN [24], FIP [5], the two baselines ResNet and VDCNN in [56]. For every comparison method, we download its original code from the corresponding authors' website, and perform experiments with its default parameter settings. The comparisons are evaluated on PSNR, SSIM [39], FSIM [53], and visual quality.

Datasets. We evaluate our PNLS with the comparison image smoothing methods on the DIV2K dataset [1] (1000 high-resolution RGB images with diverse contents), the images used in RTV [51], and the Edge-Preserving image Smoothing (EPS) dataset [56] (500 images). Note that the images in the DIV2K [1] and RTV [51] datasets do not have corresponding ground truth images, while the ground truths of the EPS dataset [56] are constructed by manually selecting the smoothed images generated by seven existing state-of-the-art image smoothing algorithms.

Evaluation. On the three datasets of [1], [51], [56], we evaluate the performance of different smoothing methods by subjectively comparing the visual quality of smoothed images. On our NKS dataset with structure images as ground truths. Thus we evaluate the performance of different smoothing algorithms qualitatively on visual quality, and quantitatively on PSNR, SSIM [39] and FSIM [53].



(a) Input Image (b) Ours (c) L0 [49] (d) RTV [51]

(e) RGF [54] (f) ResNet [56] (g) VDCNN [56] (h) FIP [5]

Fig. 9. Comparison of smoothed images by different methods on the image "*11_11*" from the dataset in [51].

B. Comparison Results

As shown in Table II, the slow version of our PNLS achieves the best results of PSNR/SSIM/FSIM on the NKS dataset. Our fast PNLS achieves comparable objective performance when compared to the slow PNLS. Besides, it greatly improves the running time, and needs averagely 5.10 seconds to process a 512×384 image. In Figures 6, 7, 8, and 9, we compare the visual results of our PNLS method with the other state-of-the-art image smoothing methods. We observe that our PNLS method preserves the structures of the image contents while smoothing out the textures of the images.

In Figure 9, RTV [51] well erases the textures between the middle circles, but destroy the structure of the red background pattern, making it very blurry. The other methods do not even fully remove the textures betweent the middle circles. Our PNLS ensures the integrity and sharpness of the background pattern while well removing the textures. Note that our PNLS outperforms the deep learning based smoothing networks in [56], even though they are trained by plenty of images with "ground truths". In Figure 6, we compare both the visual quality (subjective metric) and PSNR/SSIM/FSIM (objective metrics) results of different image smoothing methods on the image "*S15_T01*" ("15" is the index of structure image, while "01" is the index of texture one).

C. User Study on Our NKS dataset

Though our NKS dataset has ground truths, it is more convincing to subjectively evaluate the comparison methods by highly controlled user study, as did by previous work [56]. To perform user study together with our algorithm, we select seven state-of-the-art methods: L0 [49], RTV [51], RGF [54], FIP [5], TF [2] and the two baselines ResNet and VDCNN in [56]. We perform user study with the help of 80 randomly selected undergraduate and postgraduate students in Nankai University, and uniformly use our NKS dataset for evaluation. As shown in the Figure 10, we have given the input image and the ground truth in every page. What the user needs to do is only to choose the smoothed image that, in his or her opinion, is closest to the ground truth (top right image). The voting results in the Figure 11 show that our PNLS has won the most choices (206 votes), better than the second one ResNet [56] (186 votes) and the third one L0 [49] (163 votes).

D. Ablation Study

Here, we conduct deeper examinations on how the parameters influence our PNLS. All experiments are performed on our NKS smoothing dataset. Our PNLS has 5 major parameters, the searching field size $R = 15$, the step $s = 4$ of extracting reference patches, the patch size $m = 4$, the iteration number $N = 10$, the threshold $\lambda = 0.4$. We study the individual influence on our PNLS of each parameter while fixing the others. The PSNR, SSIM [39], and FSIM [53] results are summarized in Table III. We observe that, by increasing the searching field size R, the performance of our PNLS increases because more patches could be matched in one time. Increasing the step s in our PNLS will decrease the number of reference patches to be processed, and thus naturally decrease the quantitative performance of our PNLS (but also speed up the running time). The smoothing strength of our PNLS increases with the increase of iteration number N. Our PNLS is very robust to the changes of parameter m or λ. Additionally, Table III shows that the iteration number N is the most sensitive parameter to the performance of our PNLS. To study how different N influence our PNLS, in Figure 12,

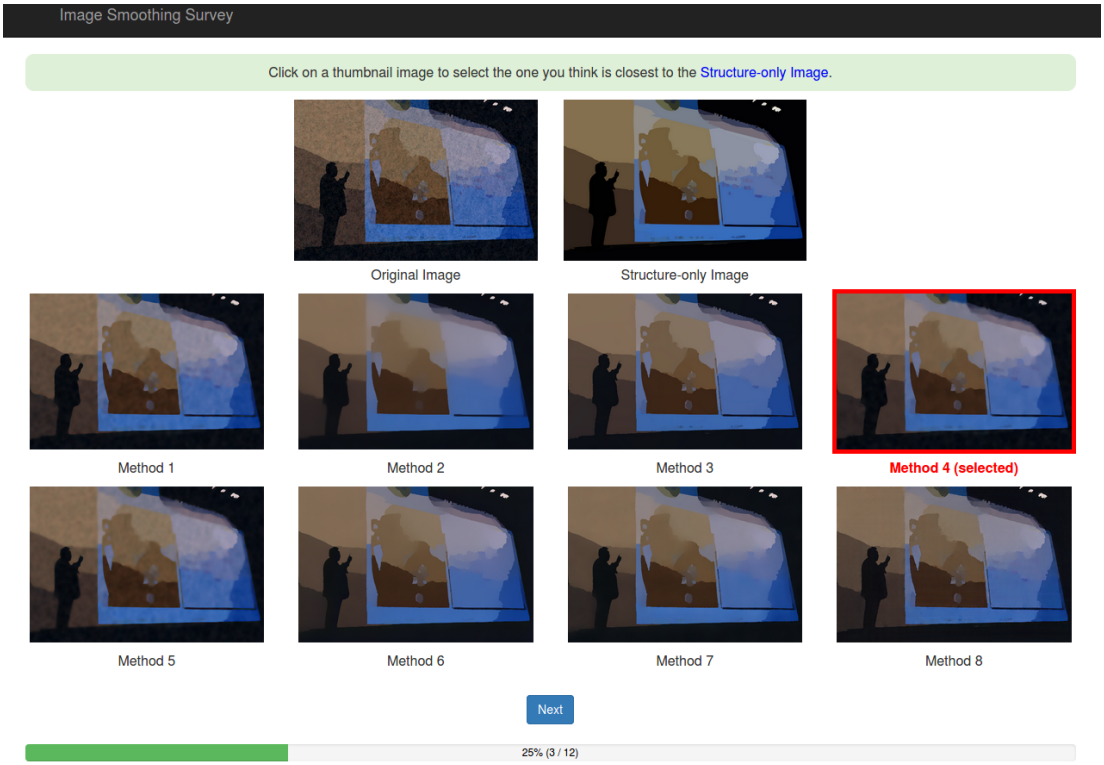


Fig. 10. **Interface of the system used in our user study experiments**. Each user has 12 rounds of voting. During each round of voting, the user selects one of the images smoothed by eight different methods that he/she thinks is closest to the structure image. The user only needs to click the selected image and then click the next button to proceed to the next round of voting.

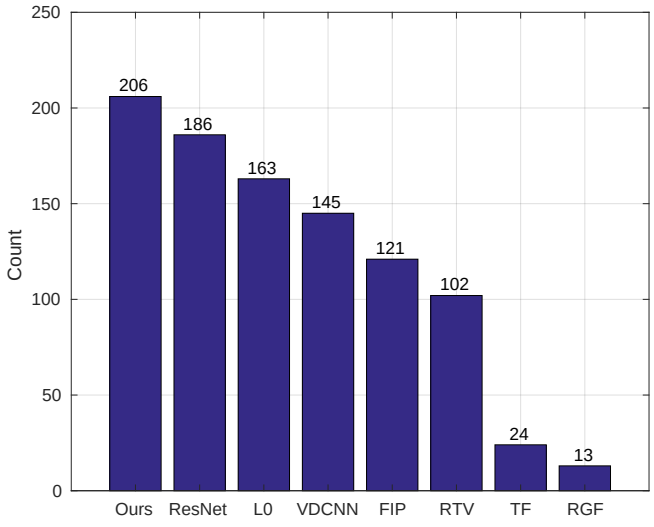


Fig. 11. Histogram of votes by 80 users for different methods.

we compare the visual quality and PSNR/SSIM/FSIM results by PNLS with different N, on the image "*S12_T06*" from our NKS dataset. One can see that our PNLS performs consistently well with $N = 5, 10, 15, 20$, demonstrating the robustness of our PNLS over different numbers of iterations.

VI. Applications

We apply the proposed Pixel-level Non-Local Smoothing (PNLS) method on 4 image processing tasks: semantic region smoothing, image detail enhancement, image edge enhancement, and image abstraction.

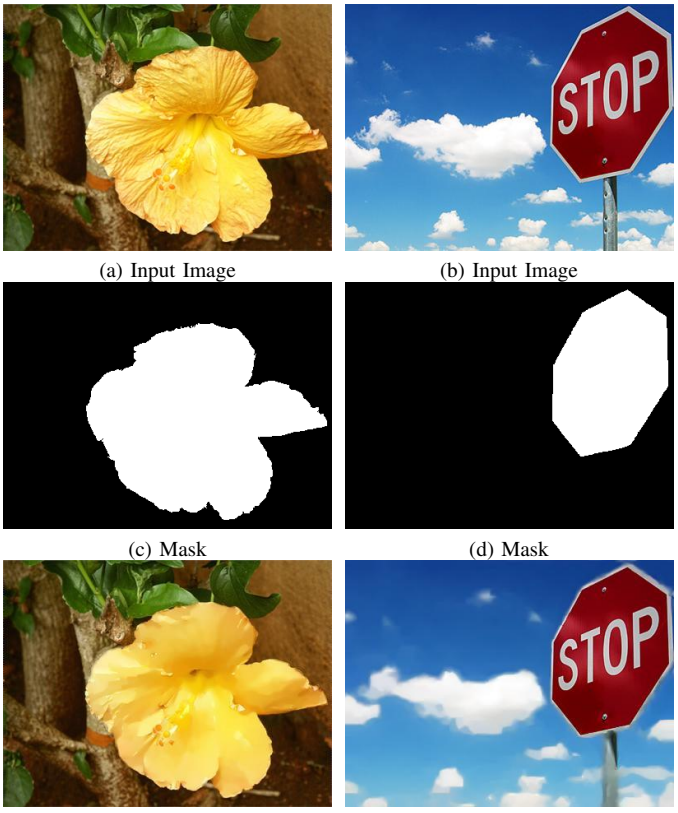
Semantic region smoothing is mainly to smooth only the foreground or background region of an image while leaving the other region as is. In this task, we first predict the foreground

TABLE III
Average PSNR (dB), SSIM and FSIM of our PNLS with different parameters over the NKS dataset. We change one parameter at a time to assess its individual influence on our PNLS. "↑" means that higher is better.

	Value	11	13	15	17	Margin
R	PSNR↑	33.30	33.39	33.45	33.48	0.18
	SSIM↑	0.9358	0.9371	0.9378	0.9379	0.0021
	FSIM↑	0.9378	0.9392	0.9397	0.9396	0.0019
	Value	1	2	3	4	Margin
s	PSNR↑	33.68	33.65	33.59	33.45	0.23
	SSIM↑	0.9420	0.9415	0.9403	0.9378	0.0042
	FSIM↑	0.9440	0.9436	0.9424	0.9397	0.0043
	Value	4	5	6	7	Margin
m	PSNR↑	33.45	33.41	33.32	33.27	0.18
	SSIM↑	0.9378	0.9391	0.9388	0.9382	0.0013
	FSIM↑	0.9397	0.9416	0.9413	0.9406	0.0019
	Value	5	10	15	20	Margin
N	PSNR↑	33.31	33.45	33.08	32.68	0.77
	SSIM↑	0.9206	0.9378	0.9377	0.9355	0.0172
	FSIM↑	0.9166	0.9397	0.9409	0.9387	0.0243
	Value	0.3	0.4	0.5	0.6	Margin
λ	PSNR↑	33.49	33.45	33.31	33.13	0.36
	SSIM↑	0.9361	0.9378	0.9378	0.9370	0.0017
	FSIM↑	0.9365	0.9397	0.9402	0.9396	0.0037

(a) Ground Truth (b) Input Image 27.96/0.5277/0.7008 (c) $N = 5$ 34.41/0.9286/0.9224 (d) $N = 10$ 34.51/0.9511/0.9579 (e) $N = 15$ 33.89/0.9469/0.9569 (f) $N = 20$ 33.29/0.9408/0.9505

Fig. 12. Comparison of smoothed images and PSNR(dB)/SSIM/FSIM results by our PNLS with parameter N in different values, on the image "*S12_T06*" from our NKS dataset.



(a) Input Image (b) Input Image (c) Mask (d) Mask (e) Foreground Smoothing (f) Background Smoothing

Fig. 13. **Semantic region smoothing** by our PNLS method of two images from the MSRA-B SOD dataset [38].

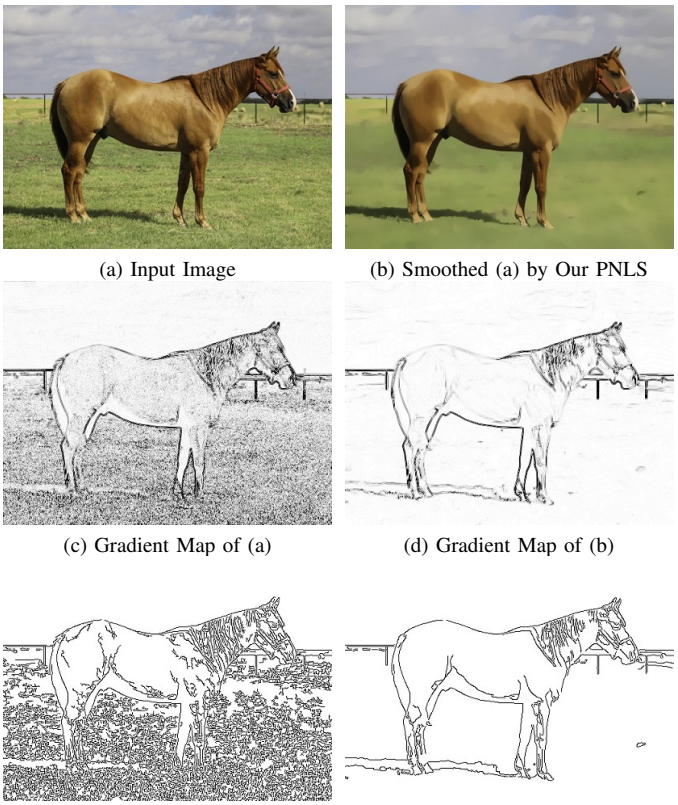
(a) Input Image (b) Input Image (c) Smoothed Image by Our PNLS (d) Smoothed Image by Our PNLS (e) Enhanced Image (f) Enhanced Image

Fig. 14. **Illustration of image detail enhancement** implemented by our PNLS smoothing method.

mask that separates the foreground and background of the image, and then smooth the region we are interested in. As shown in Figure 13, the foreground (upper row) or background (lower row) is smoothed while the background region is left as it is. Here, we use the salient ground truths as the corresponding foreground masks. In practice, we employ the famous method of [6] to predict the salient region.

Image detail enhancement aims at enhancing the details of an image while avoiding producing artifacts (gradient reversals or halos). For the input image, our PNLS decomposes it into a base layer (the smoothed image) and a detail layer (the removed textures). Then we enlarge the detail layer by 3 times while leaving the base layer as it is. The enhanced image is obtained by adding the enlarged detail layer back to the base layer. In Figure 14, we present the input images, the smoothed images by our PNLS, and the enhanced images on the images "*0347*" and "*0484*" from [56]. We observe that the images with enhanced details looks very natural when compared to the input images. This demonstrates that our PNLS well preserves the structure of input images while removing the details.

Image edge enhancement. As shown in Figure 15 (b), the boundary of the horse and the lawn are still very clear when compared to the input image (Figure 15 (a)). Here, we use a Laplacian operator and a Canny edge detector [4] to compute the gradient maps and edge maps of Figures 15 (a) and (b), respectively. The boundary in the gradient map (c) of the input image (a) is difficult to distinguish due to the interference of the textures in the surrounding area. The edge map (e) extracted by the Canny edge detector [4] is also seriously affected by the texture in (a). As shown in Figures 15 (d) and (f), our PNLS method smooths out unimportant textures (please refer to the textures in horse and the lawn in Figure 15 (b)). Thus, the Laplacian operator and the Canny edge detector [4]



(a) Input Image (b) Smoothed (a) by Our PNLS

(c) Gradient Map of (a) (d) Gradient Map of (b)

(e) Edge Map of (a) (f) Edge Map of (b)

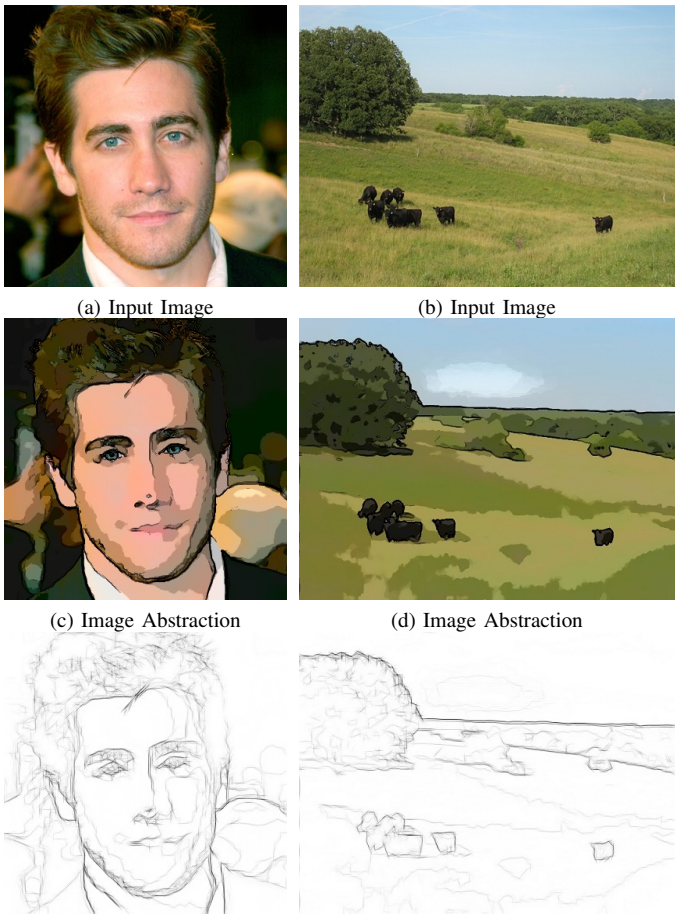
Fig. 15. **Illustration of edge enhancement and extraction results**. Our PNLS smoothing method suppresses the texture details, and strengthen structural edges of the input image (a).

are enabled to extract clear gradient map (Figure 15 (e)) and reliable edge map (Figure 15 (f)), respectively, of the smoothed image (Figure 15 (b)).

Image abstraction. Our PNLS method can also be applied into the image abstraction task. The visual results are shown in Figure 16. As suggested by [41], we perform image abstraction by replacing bilateral filtering [28] with our PNLS, and extract cartoon-style abstractions (c) and (d) of the input images (a) and (b), respectively. In addition, we also used the method of [26] to generate pencil sketching results (e) and (f) of the abstract images (c) and (d), respectively. One can see that our PNLS can help obtain promising image abstraction and pencil sketching results, due to its capability of capturing accurate structure of images.

VII. Conclusion

Despite the progress of algorithms, proper benchmark datasets in image smoothing community are in urgent requirements. In this paper, we constructed a Nankai Smoothing (NKS) dataset to affiliate the comparison of image smoothing algorithms. With our NKS dataset, we benchmark 14 popular image smoothing algorithms. Besides, we also proposed a Pixel-level Non-Local Smoothing (PNLS) method, by utilizing the pixel-level non-local self similarity prior of natural images. Our PNLS achieved better qualitative and (or) quantitative performance than the other competing methods on several benchmark datasets (including our NKS dataset). Extensive



(a) Input Image (b) Input Image

(c) Image Abstraction (d) Image Abstraction

(e) Pencil Sketching (f) Pencil Sketching

Fig. 16. **Illustration of image abstraction and pencil sketching results**, in which our PNLS method removes the texture details.

parameter analysis validated the robustness of our PNLS on image smoothing. We further validated the broad practicality of the proposed PNLS method on the tasks of semantic region smoothing, detail/edge enhancement, and image abstraction.

This work can be extended in at least two directions. First, we can further speed up our PNLS method to fulfill practical applications. Second, we can construct a larger image smoothing dataset, consisting of training and test images, to better benchmark deep learning based smoothing networks.

References

[1] E. Agustsson and R. Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *in Proc. IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, July 2017. 2, 7

[2] L. Bao, Y. Song, Q. Yang, H. Yuan, and G. Wang. Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree. *IEEE Trans. Image Process.*, 23(2):555–569, 2013. 4, 5, 6, 7, 8

[3] V. Bychkovsky, S. Paris, E. Chan, and F. Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *in Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, 2011. 2

[4] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986. 10

[5] Q. Chen, J. Xu, and V. Koltun. Fast image processing with fully-convolutional networks. In *in Proc. IEEE Int. Conf. Comput. Vis.*, pages 2497–2506, 2017. 1, 2, 4, 5, 6, 7, 8

[6] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. S. Torr, and S.-M. Hu. Global contrast based salient region detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(3):569–582, 2015. 10

[7] Clker-Free-Vector-Images. Pixabay. https://pixabay.com/users/clker-free-vector-images-3736/. Accessed: 2019-11-02. 4

[8] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Trans. Image Process.*, 16(8):2080–2095, 2007. 5

[9] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *Journal of Fourier Analysis and Applications*, 4(3):247–269, 1998. 6

[10] Q. Fan, J. Yang, D. Wipf, B. Chen, and X. Tong. Image smoothing via unsupervised learning. *ACM Trans. Graph.*, 37(6), 2018. 2

[11] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graph.*, 27(3):67, 2008. 1, 2, 4, 7

[12] R. Fattal. Edge-avoiding wavelets and their applications. *ACM Trans. Graph.*, 28(3):22, 2009. 4, 5, 7

[13] R. Fattal, M. Agrawala, and S. Rusinkiewicz. Multiscale shape and detail enhancement from multi-light image collections. *ACM Trans. Graph.*, 26(3), July 2007. 2

[14] R. G. Gavaskar and K. N. Chaudhury. Fast adaptive bilateral filtering. *IEEE Trans. Image Process.*, 28(2):779–790, 2018. 4, 7

[15] A. Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371, Sep 1910. 6

[16] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(6):1397–1409, 2013. 1, 2, 4, 5, 7

[17] Y. Hou, J. Xu, M. Liu, G. Liu, L. Liu, F. Zhu, and L. Shao. Nlh: A blind pixel-level non-local method for real-world image denoising. *IEEE Trans. Image Process.*, 29:5121–5135, 2020. 1

[18] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep joint image filtering. In *in European Conference on Computer Vision.*, pages 154–169, 2016. 1, 2

[19] Z. Liang, J. Xu, D. Zhang, Z. Cao, and L. Zhang. A hybrid l1-l0 layer decomposition model for tone mapping. In *in Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, June 2018. 1

[20] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee. Enhanced deep residual networks for single image super-resolution. In *in Proc. IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, pages 136–144, 2017. 2

[21] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang. Noise estimation from a single image. In *in Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, volume 1, pages 901–908, 2006. 2

[22] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang. Non-local recurrent network for image restoration. In *in Proc. Adv. Neural Inform. Process. Syst.*, pages 1673–1682, 2018. 4

[23] D. Liu, B. Wen, J. Jiao, X. Liu, Z. Wang, and T. S. Huang. Connecting image denoising and high-level vision tasks via deep learning. *IEEE Trans. Image Process.*, 29:3695–3706, 2020. 4

[24] S. Liu, J. Pan, and M.-H. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *in European Conference on Computer Vision.*, pages 560–576, 2016. 1, 2, 4, 7

[25] W. Liu, X. Chen, C. Shen, Z. Liu, and J. Yang. Semi-global weighted least squares in image filtering. In *in Proc. IEEE Int. Conf. Comput. Vis.*, pages 5861–5869, 2017. 2

[26] C. Lu, L. Xu, and J. Jia. Combining sketch and tone for pencil drawing production. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, pages 65–73, 2012. 11

[27] K. Lu, S. You, and N. Barnes. Deep texture and structure aware filtering network for image smoothing. In *in European Conference on Computer Vision.*, September 2018. 1, 2, 4

[28] R. Manduchi and C. Tomasi. Bilateral filtering for gray and color images. In *in Proc. IEEE Int. Conf. Comput. Vis.*, 1998. 1, 2, 4, 7, 11

[29] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *in Proc. IEEE Int. Conf. Comput. Vis.*, volume 2, pages 416–423, July 2001. 2

[30] D. Min, S. Choi, J. Lu, B. Ham, K. Sohn, and M. N. Do. Fast global image smoothing based on weighted least squares. *IEEE Trans. Image Process.*, 23(12):5638–5653, 2014. 1, 4, 5, 7

[31] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. *ACM Trans. Graph.*, 23(3):664–672, 2004. 1, 2

[32] D. Ren, W. Zuo, D. Zhang, J. Xu, and L. Zhang. Partial deconvolution with inaccurate blur kernel. *IEEE Trans. Image Process.*, 27(1):511–524, 2018. 2

[33] X. Shen, Y.-C. Chen, X. Tao, and J. Jia. Convolutional neural pyramid for image processing. *arXiv preprint arXiv:1704.02071*, 2017. 2

[34] Z. Su, X. Luo, Z. Deng, Y. Liang, and Z. Ji. Edge-preserving texture suppression filter based on joint filtering schemes. *IEEE Trans. Multimedia*, 15(3):535–548, 2013. 1, 2

[35] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, 3(2):186 – 200, 1996. 6

[36] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin. Deep learning on image denoising: An overview. *arXiv preprint arXiv:1912.13171*, 2019. 4

[37] C. Tian, Y. Xu, Z. Li, W. Zuo, L. Fei, and H. Liu. Attention-guided cnn for image denoising. *Neural Networks*, 2020. 4

[38] J. Wang, H. Jiang, Z. Yuan, M.-M. Cheng, X. Hu, and N. Zheng. Salient object detection: A discriminative regional feature integration approach. *Int. J. Comput. Vis.*, 123(2):251–268, 2017. 10

[39] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004. 3, 4, 5, 7, 8

[40] Z. Wang, J. Xu, L. Liu, F. Zhu, and L. Shao. Ranet: Ranking attention network for fast video object segmentation. In *in Proc. IEEE Int. Conf. Comput. Vis.*, Oct 2019. 2

[41] H. Winnemöller, S. C. Olsen, and B. Gooch. Real-time video abstraction. *ACM Trans. Graph.*, 25(3):1221–1226, July 2006. 2, 11

[42] J. Xu, Y. Hou, D. Ren, L. Liu, F. Zhu, M. Yu, H. Wang, and L. Shao. Star: A structure and texture aware retinex model. *IEEE Trans. Image Process.*, 2020. 1

[43] J. Xu, Y. Huang, M.-M. Cheng, L. Liu, F. Zhu, X. Hou, and L. Shao. Noisy-as-clean: Learning unsupervised denoising from the corrupted image. *arXiv preprint arXiv:1906.06878*, 2019. 4

[44] J. Xu, H. Li, Z. Liang, D. Zhang, and L. Zhang. Real-world noisy image denoising: A new benchmark. *arXiv preprint arXiv:1804.02603*, 2018. 4

[45] J. Xu, L. Zhang, and D. Zhang. External prior guided internal prior learning for real-world noisy image denoising. *IEEE Trans. Image Process.*, 27(6):2996–3010, June 2018. 4

[46] J. Xu, L. Zhang, and D. Zhang. A trilateral weighted sparse coding scheme for real-world image denoising. In *in European Conference on Computer Vision.*, 2018. 4

[47] J. Xu, L. Zhang, D. Zhang, and X. Feng. Multi-channel weighted nuclear norm minimization for real color image denoising. In *in Proc. IEEE Int. Conf. Comput. Vis.*, 2017. 4

[48] J. Xu, L. Zhang, W. Zuo, D. Zhang, and X. Feng. Patch group based nonlocal self-similarity prior learning for image denoising. In *in Proc. IEEE Int. Conf. Comput. Vis.*, pages 244–252, 2015. 4

[49] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via l_0 gradient minimization. *ACM Trans. Graph.*, 30(6):174, 2011. 1, 2, 4, 6, 7, 8

[50] L. Xu, J. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *International Conference on Machine Learning*, pages 1669–1678, 2015. 1, 2

[51] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via relative total variation. *ACM Trans. Graph.*, 31(6):139:1–139:10, 2012. 1, 2, 4, 6, 7, 8

[52] F. Zhang, L. Dai, S. Xiang, and X. Zhang. Segment graph based image filtering: Fast structure-preserving smoothing. In *in Proc. IEEE Int. Conf. Comput. Vis.*, pages 361–369, 2015. 1

[53] L. Zhang, L. Zhang, X. Mou, and D. Zhang. Fsim: A feature similarity index for image quality assessment. *IEEE Trans. Image Process.*, 20(8):2378–2386, 2011. 3, 4, 5, 7, 8

[54] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *in European Conference on Computer Vision.*, pages 815–830, 2014. 2, 4, 6, 7, 8

[55] Z. Zhou, B. Wang, and J. Ma. Scale-aware edge-preserving image filtering via iterative global optimization. *IEEE Trans. Multimedia*, 20(6):1392–1405, 2018. 1, 2

[56] F. Zhu, Z. Liang, X. Jia, L. Zhang, and Y. Yu. A benchmark for edge-preserving image smoothing. *IEEE Trans. Image Process.*, 28(7):3556–3570, 2019. 1, 2, 3, 4, 5, 6, 7, 8, 10