Deep Flexible Structure Preserving Image Smoothing

Mingjia Li[*]
mingjiali@tju.edu.cn
Tianjin University

Yuanbin Fu[*]
yuanbinfu@tju.edu.cn
Tianjin University

Xinhui Li
lixinhui@tju.edu.cn
Tianjin University

Xiaojie Guo[†]
xj.max.guo@gmail.com
Tianjin University

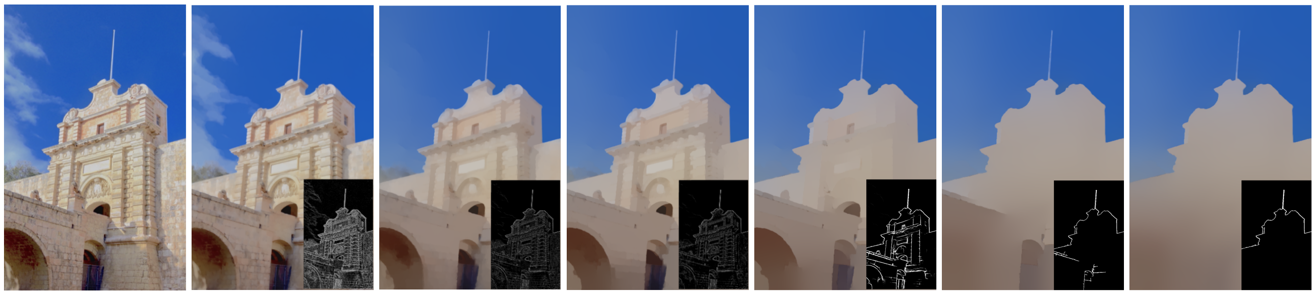


Figure 1: An example of our flexible structure preserving image smoothing. The proposed pipeline makes it easy to achieve diverse smoothing results without retraining the network whilst enjoying a small amount of network parameters and satisfactory visual quality. The edge guidance is shown on the corner of each result.

ABSTRACT

Structure preserving image smoothing is fundamental to numerous multimedia, computer vision, and graphics tasks. This paper develops a deep network in the light of flexibility in controlling, structure preservation in smoothing, and efficiency. Following the principle of divide-and-rule, we decouple the original problem into two specific functionalities, *i.e.*, controllable guidance prediction and image smoothing conditioned on the predicted guidance. Concretely, for flexibly adjusting the strength of smoothness, we customize a two-branch module equipped with a sluice mechanism, which enables altering the strength during inference in a fixed range from 0 (fully smoothing) to 1 (non-smoothing). Moreover, we build a UNet-in-UNet structure with carefully designed loss terms to seek visually pleasant smoothing results without paired data involved for training. As a consequence, our method can produce promising smoothing results with structures well-preserved at arbitrary levels through a compact model with 0.6M parameters, making it attractive for practical use. Quantitative and qualitative experiments are provided to reveal the efficacy of our design, and demonstrate its superiority over other competitors. The code can be found at https://github.com/lime-j/DeepFSPIS.

CCS CONCEPTS

• **Computing methodologies → Image manipulation**.

[*]Both authors contributed equally to this research.
[†]Corresponding author.

KEYWORDS

Image smoothing, Structure preserving, Deep learning

ACM Reference Format:
Mingjia Li, Yuanbin Fu, Xinhui Li, and Xiaojie Guo. 2022. Deep Flexible Structure Preserving Image Smoothing. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3547857

1 INTRODUCTION

Structure preserving image smoothing aims to suppress unwanted textural details while maintaining desired structures in natural images, which can ameliorate the performance of subsequent applications such as image stylization [5, 13], stereo matching [19, 30, 50], and optical flow [38, 46]. The problem is highly ill-posed, as there are infinite possible outputs for the same input. Given different images and tasks, hardly a sound way exists for determining which outputs are correct or best. Please see Fig. 1 for example. Usually, for obtaining a particular result among so many possible outputs, one may need to laboriously tweak the (hyper-) parameters of smoothing algorithms, *e.g.*, a numerical scalar adjusting the smoothing level/strength. Thus, for practical use, it is of great significance to develop a flexible scheme that allows users to intuitively customize their desired smoothing effects from various candidates.

Recently, deep learning techniques have considerably pushed forward the field of image filtering/smoothing, thanks to their strong capability of feature extraction. Our approach falls into this category as well. Objectively, learning to adapt or select the smoothing extent/level according to particular needs is still challenging for three main reasons: 1) It is difficult to change the smoothing degree when the network weights were frozen in the testing phase. How to adjust the weights or the generated feature maps of trained deep models for producing different effects in a handy manner is key to addressing this issue; 2) Aligned/paired ground-truth (reference) smoothed images with different smoothing strengths, are either tough to collect or even nonexistent, which prevents the network training from a supervised way; and 3) The relationship between natural images and desired smoothing outputs is complex, especially for images with diverse patterns and/or abundant color information. As a result, directly training a unified network to make the above happen may require a substantial amount of network parameters. Most existing deep learning approaches [6, 9, 14, 24, 31, 48] fail to resolve the problem of flexibly altering the smoothing effects in the testing phase. They merely produce results imitating one of existing optimization-based operators, like L0 [47], RTV [49], or muGIF [16]. For varying smoothing strengths, the network needs to be retrained with specifically produced (pseudo) ground truths. Several works [8, 11] for addressing this issue adopt a common network with dynamically generated weights to simulate the target operator with different selected configurations. In other words, during training, they are enslaved to producing multi-level smoothed results for each input image as paired data yet. In addition, the parameter for adjusting the smoothing strength exactly correspond to that of the imitated operator, the value of which is not restricted in a unified/fixed range. Although the work of [10] survives from manufacturing paired references for training in a supervised manner, it still suffers from the retraining procedure when tuning the configuration for different smoothing extents.

Contributions. This paper aims to build an efficient and effective deep learning framework, by jointly taking into account flexibility of adjusting the smoothing strength, promising visual quality, and computational efficiency. Our framework contains two main modules, *i.e.* adjuster and smoother. The motivation behind this logical partition is to decompose the original complex problem into two smaller sub-ones for restricting the solution space and developing compact models. Concretely, the adjuster responds for guidance prediction with respect to a strength parameter $\lambda \in [0, 1]$, which connects between λ and an intuitive guidance map that will be later used to indicate the smoothing procedure. The adjuster is a two-branch network equipped with a sluice mechanism, one branch of which is for extracting edge-related features while the other for texture-related features. Please notice that the guidance map can also be offered by users and be possibly generalized to other types of indicator. The high-quality smoothing results conditioned on the predicted/provided guidance can be obtained by our smoother, without (λ/guidance map, smoothed result) pairs required for training. Our contributions can be summarized as:

- We customize a smoothing/filtering framework that allows users to flexibly adjust the strength in the testing stage, by simply tuning a parameter λ in a fixed continuous range.
- Two decoupled modules are developed for model effectiveness and efficiency, *i.e.*, an adjuster for guidance prediction with respect to λ, and a smoother for conditional image smoothing, making our model attractive for practical use.
- We design several losses in cooperation with a UNet-in-UNet architecture to execute the training, making unpaired learning with high-quality smoothing results feasible.

Extensive experiments are conducted to demonstrate the efficacy of the proposed framework, reveal its superiority over other state-of-the-art alternatives, and show its potential to various applications.

2 RELATED WORK

The core idea of early image smoothing strategies is to average pixels within its local neighbourhood, with mean/Gaussian filters [44] and bilateral filter (BF) [35, 42] as representatives. Although these methods are simple and fast, due to the content-blindness, obvious defects often leave in visual results, such as gradient reversal artifacts and halos [2, 12]. Another attempt based on the scale-space theory, called rolling guidance filter (RGF) [51], takes a Guassian-blurred image as input, and then gradually seeks structural edges from the previous iteration to indicate the next round. Its main drawback comes from the inaccurate edge localization. Generally speaking, the above operators execute image smoothing by solely utilizing the local information, without consideration of global features.

To address the local issue, a variety of approaches have been proposed via treating the image smoothing as a global optimization problem. Among these schemes, l_0 gradient minimization [47] employs the ℓ_0 norm to regularize gradients to be sparse in filtered results. Further, the works of [3, 25, 27] improve the original ℓ_0 regularization terms by the ℓ_1 (or its variants) to clean up possible isolated spots. Xu *et al.* [49] designed a relative total variation (RTV) prior to handle complex textures. Besides the self-guided mode, several works adopt another signal with a different modality or shooting condition as reference to guide the optimization procedure. For instance, joint bilateral filter (JBF) [36] and guided image filter (GIF) [18] rely on the reference to determine either kernel weights or linear transform bases, which particularly favor the cases where the guidance image can provide more reliable information than the filtered image. Ham *et al.* [17] devised a static/dynamic filter (SD) to take advantage statically of the reference. However, these mentioned reference-guided methods lack of the ability to optimize/update the reference image. To overcome this problem, Guo *et al.* [16] proposed a simple yet effective filter, named mutually guided image filter (muGIF), to jointly optimize the two inputs. More recently, Liu *et al.* [28] developed a framework to perform image smoothing on tasks with different or even opposite requirements, by tuning (hyper-) parameters of a carefully-designed Hubel truncated loss. Though satisfactory smoothing effects, these optimization-based methods typically involve multiple iterations to converge, each of which requires computationally expensive operators, like the inverse of large matrices, limiting their applicability.

With the emergence of deep learning, as verified in [48] and [24], it is feasible to build deep networks for simulating traditional operators in a fully supervised manner. In this way, the inference could be greatly accelerated, because no online optimization is performed in the testing phase. We emphasize that, for the target problem, there is no well-defined best smoothing result for an image, and thus no so-called ground truth. To relieve the pressure of preparing (pseudo) ground truths, Fan *et al.* [10] investigated a strategy to directly learn from data without any supervision through optimizing the gradient-based objective function. However, this approach can only produce a certain filtering effect among multiple possible candidates, specifically to a given parameter setting. To make the filtering flexible, the work of [8] tries to dynamically generate the weights of the network according to different parameters during inference. It is capable to offer users with options of altering the

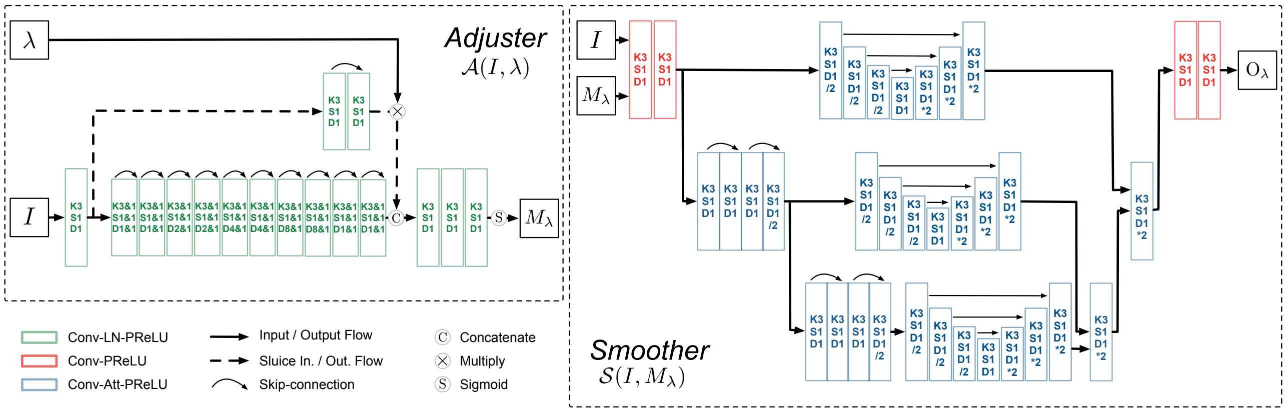


Figure 2: Illustration of our proposed framework. The letters K, S, and D indicate kernel size, stride and dilation rate of a convolution layer, respectively. Block with /2 denotes MaxPool2d(2) operation, while *2 represents upsampling operation via bi-linear interpolation. *Att* refers to the pixel-wise plus channel-wise attention layer [37].

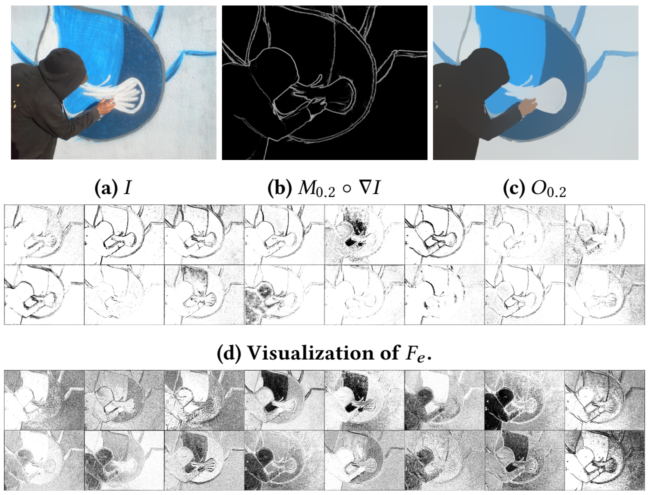
smoothing strength for diverse parameter configurations, without the need of retraining the network. Unfortunately, the paired filtered/smoothed images are still indispensable during training. Besides, by replacing the gradient prior with a semantic prior, Kim *et al.* [21] inserted a deep solver into the optimization algorithm of traditional operators, which contains the high-level information for discriminating structures and high-frequency textures. Even though allowing users to vary the smoothing degree by adjusting the parameters, it is essentially a global optimization-based method which is time-consuming due to multiple iterations required at the testing stage. Please notice that all the above mentioned deep-learning models demand either retraining the network for adjusting the filtering extent, or processing the natural images by existing operators for obtaining paired filtered images as (pseudo) ground truth, which is inconvenient to use in practice.

3 METHODOLOGY

This paper investigates a framework that decouples the structure preserving image smoothing problem into two logical components, *i.e.*, an adjuster for predicting guidance according to different demands, and a smoother for carrying out the smoothing in strict accordance with the predicted guidance, as depicted in Fig. 2. By this divide-and-conquer means, the two components can focus on their own jobs, which can thus be trained independently. Let I and O_λ denote the input and the smoothed result corresponding to the strength parameter λ, respectively. The adjuster $\mathcal{A}(I,\lambda)$ produces the guidance map M_λ controlled by λ, while the smoother $\mathcal{S}(I,M_\lambda)$ outputs O_λ conditioned on M_λ. The strength of smoothness λ is unified in the range from 0 (fully smoothing) to 1 (non-smoothing), making it easy to adjust. Since we the two modules have different functionalities, we made task-specific design for each of them. In what follows, we explain the two components in detail.

3.1 Adjuster

3.1.1 Two-branch Architecture. The blueprint of our adjuster with a sluice mechanism is shown in the left part of Fig. 2. Given an input image I, we first pass it through a convolutional (*conv*) layer


(a) I **(b)** $M_{0.2}\circ\nabla I$ **(c)** $O_{0.2}$

(d) Visualization of F_e.

(e) Visualization of F_t.

Figure 3: (c) is the smoothed result of (a) conditioned on the predicted guidance (b) wrt $\lambda = 0.2$. F_e concentrates more on edges (d), while F_t prefers texture-related information (e).

to extract features F shared between the following two branches. Then, another 20 *conv* layers (deep branch) are equipped to learn edge-related information F_e, while only 2 extra *conv* layers (shallow branch) are employed to produce texture-related ones F_t. This is because that the edges or object-level boundaries are of higher semantic than the textural details or gradients, and thus spending more resources. Finally, F_e and $\lambda\cdot F_t$ are concatenated and fed into three decoding blocks to form the guidance M_λ. The parameter λ here can be regarded as a sluice for controlling the amount of textures conveyed by the shallow branch. When λ is close to 1, F_t makes a relatively evident impact on the final prediction. As λ decreases toward 0, F_e gradually stands out and F_t is equivalently suppressed. As a result, the larger (smaller) the λ, the denser (sparser) the M_λ. Figure 3 exhibits the visual comparison between F_e and F_t, corroborating the rationality of our two-branch design.

3.1.2 Two-stage Training Scheme. For the purpose of enforcing smoothing results to be perceptually meaningful, our adjuster desires multi-level edge-like information as guidance to the smoother. It is intuitive to draw support from the edge detection community. However, a huge gap exists between edge detection benchmarks (*e.g.*, NYUD [34] and BSDS500 [32] datasets) and our task, which impedes the intuitive plan. In other words, our mission involves not only a single level prediction corresponding to the ground-truth/annotated edge map[1] but a series of guidance maps related to different λ's in $[0, 1]$.

To shrink the gap and make the problem tractable, we alternatively introduce a two-stage training strategy. At the first stage, the shallow branch for modeling textural information is disabled. The adjuster is trained in a recurrent fashion with a pre-defined number ($R = 4$ in this work) of iterations corresponding to different discrete levels. Specifically, in the first round, the features F from the beginning layer is fed into the deep branch for obtaining the edge-related features $F_e^{(1)}$ and the guidance map $M^{(1)}$, respectively. Afterwards, $F_e^{(1)}$ is then taken as the input to the same branch to acquire $F_e^{(2)}$ and $M^{(2)}$. This process performs so on and so forth until the maximal number R of iterations is reached. The predicted guidance maps $M^{(r)}$ ($r \in \{1, 2, ..., R\}$) are trained on the BSDS500 dataset, and constrained by the guidance consistency loss detailed later in Sec. 3.1.4. This procedure enables our primitive adjuster to conduct multiple anchor guidance maps stemmed from the edges. Here we define the function of calculating the strength parameter λ, given a guidance map M, as follows:

$$\lambda := \sqrt{\Lambda(M, I)} \quad \text{with} \quad \Lambda(M, I) := \frac{\|M \circ \text{bin}(\nabla I)\|_1}{\|\text{bin}(\nabla I)\|_1}, \qquad (1)$$

where $\circ$ designates the Hadamard product, ∇ means the first order derivative filter, $\text{bin}(\cdot)$ is defined as binarization with a threshold of 0.005, and $\|\cdot\|_1$ stands for the ℓ_1 norm. Via such a simple function, the guidance map can be always connected to the numerical $\lambda \in [0, 1]$. Please notice that, for different images, the values of λ with respect to the R guidance maps generated from the first stage may change. The second stage is to learn a continuous mapping from λ to guidance map. The images in the BSDS500 are quite limited [29, 45], providing insufficient λ-guidance pairs to support a reasonable transition. Therefore, we additionally employ 10k images from the COCO dataset [23] without any annotated edge information. With the guidance maps predicted by the first-stage model, the volume of λ-guidance pairs is expanded on the additional images. In the sequel, all of the λ-guidance pairs produced from the BSDS500 and augmented from the COCO are used to better learn the transition.

3.1.3 Heuristic Component Drop. In nature, structure preserving image smoothing can be regarded as merging neighbor pixels having similar colors/intensities within different regions/objects into different components. The transition could be soft when the smoothing strengths are relatively low, where only textures and/or small regions/objects are involved to clean out. But, as the smoothing strength increases beyond a threshold τ, the merging happens between objects or large regions. Under the circumstances, each component ought to be considered as a whole, *i.e.*, a binary option to avoid the structure leakage/destruction issue. To this end, we adopt

[1]For the task of edge detection, the answer should be unique and well-defined.

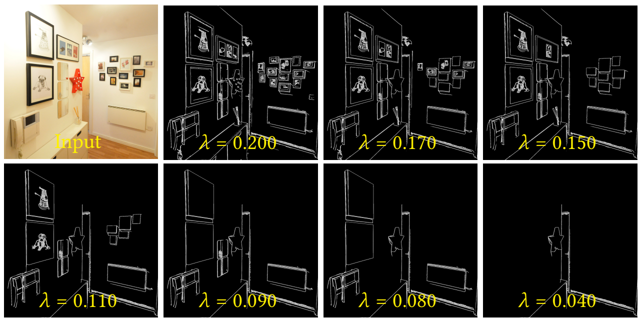


Figure 4: An example of our component drop.

a heuristic algorithm, called *component drop*. With a slight modification from [1], the elements in the (binarized) M_τ are organized into multiple sets of 8-connected components. In our experiments, τ is empirically set to 0.2 for all images, which works sufficiently well. Then, we select and drop the smallest components from M_τ by binary search until the remaining map is the closest one to the appointed λ. Please notice that the heuristic component drop is activated only when $\lambda \leq \tau$. Figure 4 shows the effectiveness of the proposed component drop.

3.1.4 Loss Function. The objective of training the adjuster consists of guidance consistency, interpolation, and label regression losses. *Guidance consistency loss* is to enforce the predicted guidance to be consistent with a target signal. For the first stage training, the target signal at the beginning adopts $\nabla I^{(1)} := (1 - \alpha_0)\nabla I + M^{GT}$ with $\alpha_0 := 0.1$ and M^{GT} the ground-truth edge map, which is then updated as follows:

$$\forall r \in \{2, ..., R\} \ \nabla I^{(r)} := (1 - \alpha_r)\nabla I^{(r-1)} + \alpha_r \nabla I^{(r-1)} \circ M^{GT}, \quad (2)$$

where α_r increases along with the iteration step r by $\alpha_r \leftarrow 1.5 \cdot \alpha_{r-1}$. The above updating rule gradually suppresses textures and converges towards M^{GT}. The loss for the first stage training is:

$$\mathcal{L}_{\text{con}}^{1st} := \sum_{r=1}^{R} \|M^{(r)} \circ \overline{\nabla I^{(r)}}\|_1 + \beta \|\overline{M^{(r)}} \circ \nabla I^{(r)}\|_1, \qquad (3)$$

where $\overline{\nabla I^{(r)}} := \mathbf{1} - \nabla I^{(r)}$, $\overline{M^{(r)}} := \mathbf{1} - M^{(r)}$, and β is a parameter balancing the two terms. For the second stage, by treating the $M^{(r)}$ as reference, the guidance consistency loss turns out to be:

$$\mathcal{L}_{\text{con}} := \sum_{r=0}^{R+1} \|M_{\lambda^r} \circ \overline{M^{(r)}}\|_1 + \beta \|\overline{M_{\lambda^r}} \circ M^{(r)}\|_1, \qquad (4)$$

where $M^{(r)}$s, $r \in \{1, 2, ..., R\}$, are generated by the model trained at the first stage and M_{λ^r} denotes the guidance map predicted at the second stage with $\lambda^r := \sqrt{\Lambda(M^{(r)}, I)}$. In particular, $\lambda^{R+1} := \tau$, $M^{(R+1)} := M^{(R)}$, $\lambda^0 := 1$, and $M^{(0)} := \text{bin}(\nabla I)$. Moreover, β is set to 12 and 1 for the first and second stages, respectively.
Interpolation loss is capable to constrain the degree of output M_{λ^r} to be close to the corresponding λ^r simply by:

$$\mathcal{L}_{\text{intp}} := \sum_{r=0}^{R+1} \left(\lambda^r - \sqrt{\Lambda(M_{\lambda^r}, I)}\right)^2. \qquad (5)$$

Label regression loss consists of a cross entropy (CE) term and a dice term, as adopted by [7]. By denoting $M_{\lambda^r}(i)$ as the i-th element of

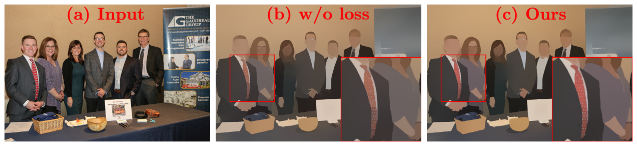


Figure 5: Visual comparison between without and with regularization on side-outputs by the loss.

M_{λ^r}, the class-balanced CE term and dice term are as follows:

$$\mathcal{L}_{\text{CE}} := -\sum_{r=0}^{R+1}\left(\sum_{i\in M_+}\log(M_{\lambda^r}(i)) - \sum_{i\in M_-}\log(1-M_{\lambda^r}(i))\right), \quad (6)$$

$$\mathcal{L}_{\text{dice}} := \sum_{r=0}^{R+1}\frac{\|M_{\lambda^r}\circ M_{\lambda^r}\|_1 + \|M^{(r)}\circ M^{(r)}\|_1}{2\cdot\|M_{\lambda^r}\circ M^{(r)}\|_1}, \quad (7)$$

where M_+ and M_- represent the edge pixel and non-edge pixel sets, respectively. Consequently, the label regression loss is given by:

$$\mathcal{L}_{\text{reg}} := \mathcal{L}_{\text{CE}} + \gamma_{\text{dice}}\mathcal{L}_{\text{dice}}, \quad (8)$$

where γ_{dice} (0.002 in our experiments) refers to a balance between the CE and dice terms.

As for the first stage training, only the guidance consistency loss $\mathcal{L}_{\text{con}}^{1st}$ is used. While for the second training stage, all of the interpolation, label regression and guidance consistency losses are involved, yielding:

$$\mathcal{L}_{\text{cont}} := \mathcal{L}_{\text{con}} + \gamma_{\text{intp}}\mathcal{L}_{\text{intp}} + \gamma_{\text{reg}}\mathcal{L}_{\text{reg}}, \quad (9)$$

where γ_{intp}, and γ_{reg} are the corresponding coefficients. In our experiments, we respectively set γ_{intp}, and γ_{reg} to 0.25 and 10.

3.2 Smoother

3.2.1 UNet-in-UNet Architecture. It is critical to have a large receptive field for acquiring better contextual information. Otherwise, the visual quality would be considerably poor when handling cases with large objects to be smoothed out. A straightforward way for enlarging receptive fields is to deepen the network, the drawback of which is that the redundant network parameters will be involved, and the gradient vanishing or explosion may occur. To effectively achieve the goal with the parameter space remarkably reduced, we alternatively employ a UNet-in-UNet architecture as shown in the right part of Fig.2. We advocate that each skip-connection in the vanilla UNet [39] can be also replaced with a (4-layer) U-shaped module, namely inner UNet, to further enlarge the receptive field. In detail, we progressively downsample the feature maps of previous layers on the encoding side to process features with smaller sizes, then upsample and fuse them with the feature maps produced by the inner UNet on the decoding side. The final outputs O_λ together with the side-outputs from those inner UNets are all regularized by the loss formulated in Eq. (13), which helps preserving the colors of input images (see Fig. 5). Besides, to stabilize the training, we adopt pixel-wise plus channel-wise attention [20, 37]. Figure 6 gives a comparison to show the rationale of our UNet-in-UNet structure in terms of suppressing large objects in comparison with a UNet without inner UNet in its skip-connection (Fig. 6(c)) and a Residual-in-UNet variant by substituting the inner UNet with residual blocks (see Fig. 6(b)).

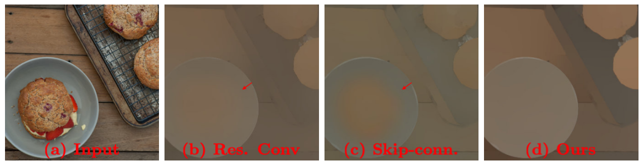


Figure 6: Visual comparison between different UNet-in-UNet configurations.

3.2.2 Loss Function. When training the smoother, we randomly sample $x \in [0,1]$ from a uniform distribution, then set $\lambda = x^{\lfloor 0.2j\rfloor+1}$ at the j-th epoch, to generate guidance map M_λ by the trained adjuster. The objective contains the truncated reconstruction loss, texture suppressing loss, and gradient preserving loss.

Truncated reconstruction loss preserves the structural similarity between the smoothed result O_λ and the original I, the penalty of which is in the following truncated manner:

$$\mathcal{L}_{\text{rec}} := \frac{1}{N}\sum_i \min[(O_\lambda(i) - I(i))^2, \lambda + \eta], \quad (10)$$

where N represents the total amount of pixels in the image. The threshold parameter η controls the position of truncation. In our experiments, empirically setting η to 0.5 works well. The truncation operation enables removing large components, which is vital especially when smoothing a large area that consists of components with dramatic differences in color.

Texture suppressing loss is proposed to constrain the desired smoothing result O_λ in the gradient domain, which can be expressed as:

$$\mathcal{L}_{\text{sup}} := \left\|\frac{\nabla O_\lambda}{M_\lambda \circ \nabla I + \epsilon}\right\|_2^2, \quad (11)$$

where $\|\cdot\|_2$ is the ℓ_2 norm, and ϵ is a small constant (0.005 in this work) to avoid zero denominator. If the gradients located on predicted edges (*i.e.*, large $|M_\lambda(i)\cdot\nabla I(i)|$ values), the penalties are small; otherwise the penalties get large.

Gradient preserving loss further regularizes the structure of smoothing result to be similar with the edge guidance via:

$$\mathcal{L}_{\text{psrv}} := \|M_\lambda \circ \nabla I - \nabla O_\lambda\|_2^2. \quad (12)$$

The total loss of our smoother network $\mathcal{S}(I, M_\lambda)$ combines these three terms with two coefficients γ_{sup} and γ_{psrv}, written as follows:

$$\mathcal{L} := \mathcal{L}_{\text{rec}} + \gamma_{\text{sup}}\mathcal{L}_{\text{sup}} + \gamma_{\text{psrv}}\mathcal{L}_{\text{psrv}}. \quad (13)$$

In our experiments, γ_{sup} and γ_{psrv} are set to 4 and 0.1, respectively.

4 EXPERIMENTAL VALIDATION

4.1 Implementation Details

Our model is implemented in PyTorch. All of our experiments are carried out with an NVIDIA RTX 2080Ti GPU and an Intel Core i7-8700 3.20 GHz CPU. The Adam optimizer [22] is used for training the network, whose learning rate is set to 1e-4 at the beginning and linearly decays to 1e-6. Our model is insensitive to all the hyper-parameters, *e.g.*, when slightly adjusting the hyper-parameters, there is no obvious impact on the final visual effects. The adjuster and the smoother are trained independently. As previously noted, the data utilized for the first training stage of adjuster is from BSDS500 [32], while for the second stage of adjuster and the training of smoother, 10k images from the COCO [23] dataset are used.

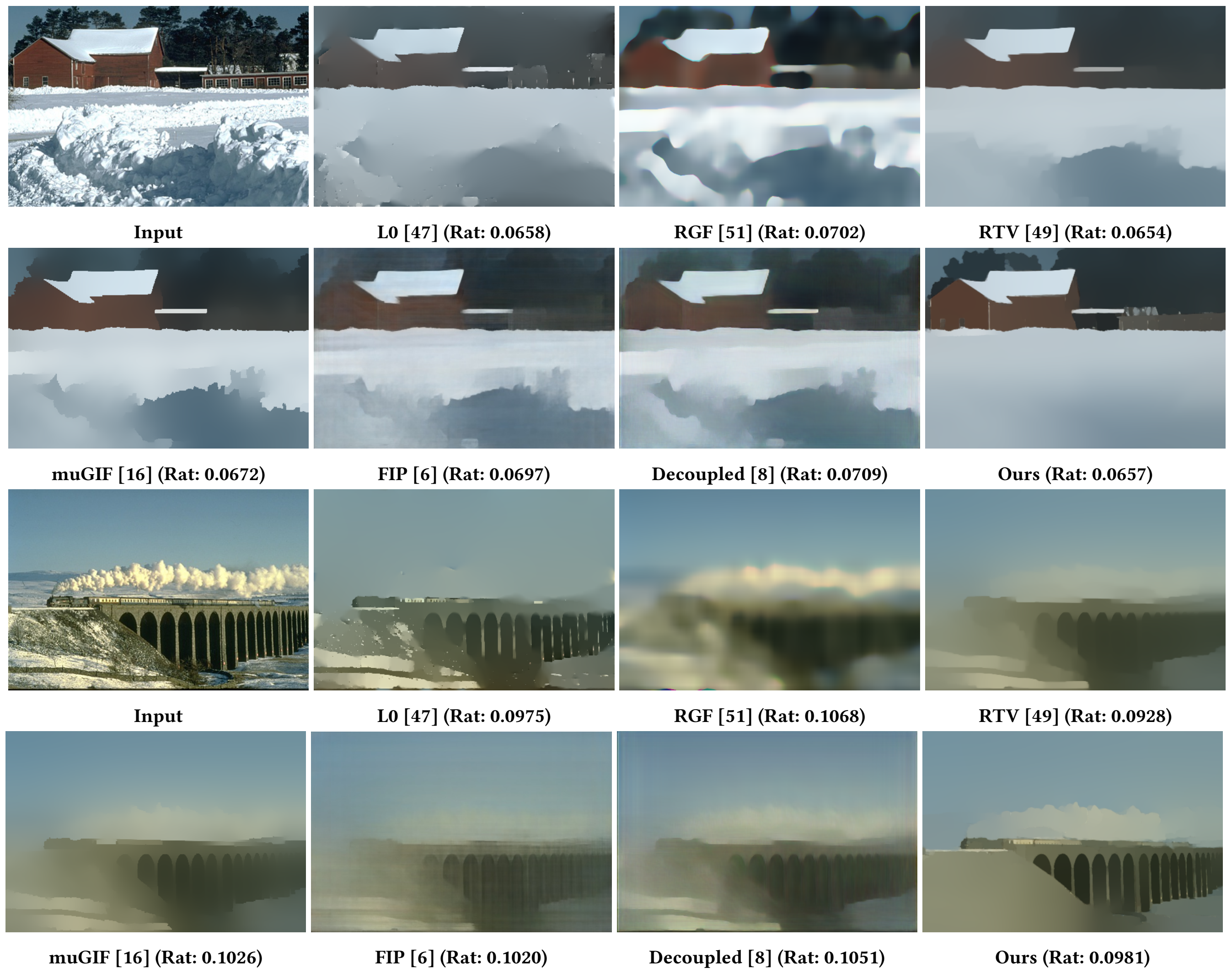


Figure 7: Visual comparison with state-of-the-art methods on image smoothing. The number in bracket behind each method indicates the ratio between the preserved gradient amount of the smoothed result to that of the original input image.

4.2 Performance Evaluation

4.2.1 Qualitative comparison. This part exhibits several visual comparisons between our method and other SOTA works. For the traditional methods, we select the L0 [47], RTV [49], RGF [51], and muGIF [16]. As for deep learning approaches, FIP [6] and decouple learning [8] are involved. We carefully tweak the (hyper-) parameters for each competitor to reach a similar smoothing level measured by the gradient ratio, denoted as *Rat*, for fair comparison. The references (pseudo ground truths) of FIP [6] and Decouple learning [8] are generated by the RTV method [49]. As shown in Fig. 7, the visual quality of the results by L0 and RGF is rather poor at this level of smoothing in structure preservation and texture removal. Though muGIF and RTV perform better, they still suffer from the structure leakage and the inability of smoothing out large objects/components. Moreover, the decoupled learning and FIP inherit the drawback of RTV, due to their imitation nature. Thanks to the semantic guidance learned from annotated edge labels and the strict adherence to the guidance, our method is capable to produce more visually-pleasing and semantically-meaningful results.

In addition, our method is flexible to embrace various types of user-defined edge maps, thanks to the principle of divide-and-rule. As a result, one can somehow edit the guidance map to fulfill his/her personalized demand. Given an input as shown in Fig. 8 (a), the guidance map in Fig. 8 (b) is synthesized by three kinds of edge respectively from Canny (left) [4], our adjuster (middle), and PiDiNet [41] (right). In (d), we manually remove the parterre on the ground from the guidance map produced by our adjuster. As can be seen, the results (c) and (e) firmly adhere to the guidance maps (b) and (d), respectively, which verify the flexibility in handling various guidance maps, and the strong smoothing consistency with the provided guidance, of our smoother. Moreover, a visual comparison between the ground-truth edge map of a sample from the BSDS500 dataset and the guidance map $M_{0.2}$ extracted by our adjuster is shown in Fig. 9 to further confirm the efficacy of our adjuster.

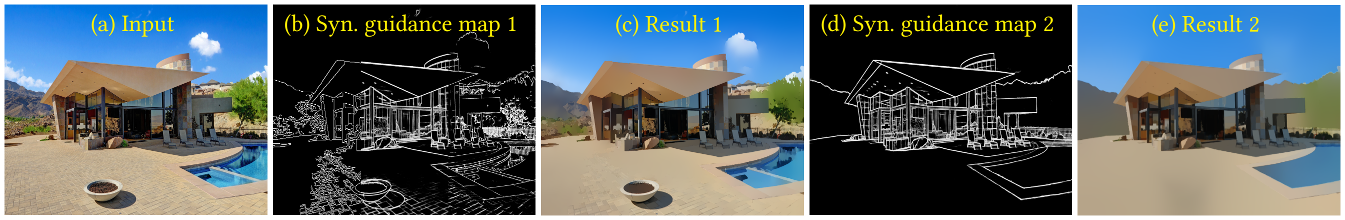


Figure 8: An illustration to the flexibility of our model.

Table 1: Runtime comparison on processing one 480p image *in seconds*. GPU costs are marked with †.

Method	L0	RGF	SD	RTV	L1	realLS
Time	1.3426	0.4246	1.0664	1.8052	94.5964	0.2969
Method	**muGIF**	**enBF**	**DEAF**	**FIP**	**Decouple**	**Ours**
Time	1.9451	0.1710	0.3261†	0.033†	0.038 †	0.057†

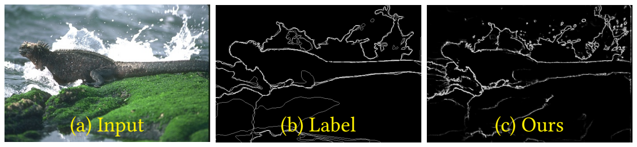


Figure 9: A comparison between the manually-annotated label and our predicted guidance map.

4.2.2 Running-time comparison. The time costs of different methods during inference are shown in Tab. 1. Traditional methods including L0 [47], RTV [49], RGF [51], SD [17], L1 [3], muGIF [16], realLS [27], and enBF [26], as well as deep learning approaches including DEAF [48], FIP [6], and Decouple Learning [8] are involved in the comparison. The implementations of competitors are all provided by their authors. The running time is measured by averaging over 100 images from the BSDS500 validation set (resized to 480p). As reported in Tab. 1, our method is more efficient than those traditional operators, but slightly fall behind FIP and Decouple by about 0.02s. By further taking the smoothing quality and the flexibility into consideration, our model seems to be a better choice than the others for practical use.

4.2.3 Ablation Study. To reveal the effectiveness of our asymmetric two-branch design in the adjuster, a symmetric design, say the two branches having the same depth are adopted for comparison. It can be seen from Fig. 10, our asymmetric design (a deeper branch and a shallower one), can extract structural edges more precisely with less parameters. As for our two-stage training of adjuster, Fig. 11 shows the difference between with and without the two-stage training scheme. We can observe from (e) and (f) that the adjuster extracts inaccurate guidance without the pairs generated at the first stage, which leads to noticeable artifacts in the final results, as shown in (b). Moreover, to verify the validity of our UNet-in-UNet design, we replace the inner UNet between the encoder and decoder with 8 residual-blocks (Fig. 6(b)) and skip-connection (Fig. 6(c)), then retrain the model with the same guidance produced by the adjuster. Our proposed architecture can suppress undesired textures and maintain primary structures better than the other options (failing to clearly remove large objects as indicated by the red arrows, and suffering from the unpleasant color shift issue).

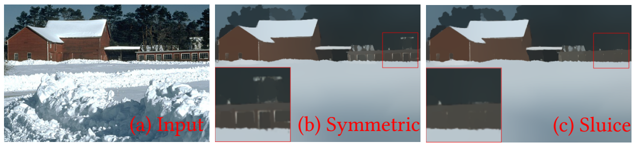


Figure 10: Ablation Study on our sluice design.

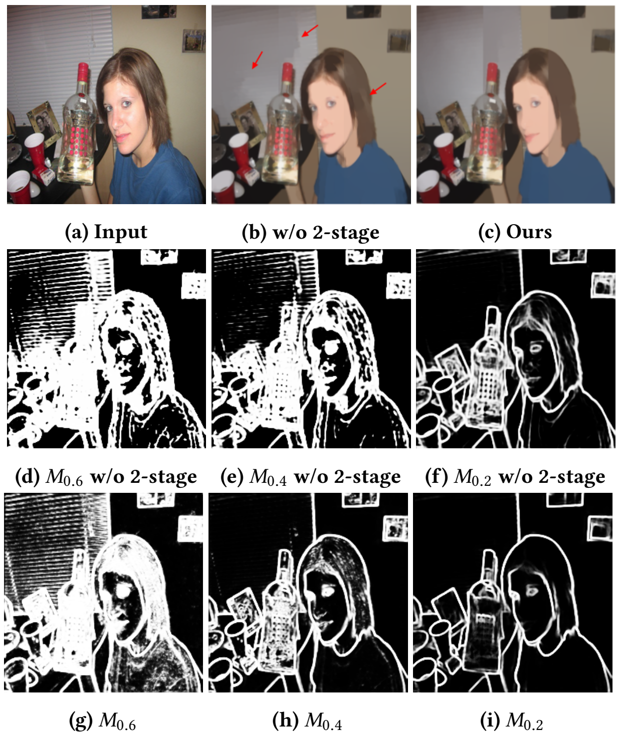

(a) Input
(b) w/o 2-stage
(c) Ours
(d) $M_{0.6}$ w/o 2-stage
(e) $M_{0.4}$ w/o 2-stage
(f) $M_{0.2}$ w/o 2-stage
(g) $M_{0.6}$
(h) $M_{0.4}$
(i) $M_{0.2}$

Figure 11: Ablation study on two-stage adjuster training.

4.3 Applications

4.3.1 Guided Depth Upsampling. Following previous literature [16, 28], we first pollute the depth images from Middlebury benchmarks with noise, then downsample them at four scales {2, 4, 8, 16} to simulate the degradation of depth images. The recovery process (upsampling and denoising) can be achieved by taking a registered high-resolution RGB image as reference. The competitors are reference-guided filters, including MSJF [40], GIF [18], muGIF [16], and SD [17]. The filtering is directly guided by the registered RGB images. While for our method, the edges extracted from the RGB images by the adjuster are taken as guidance. Notice that existing deep learning methods do not have the flexibility to utilize the guidance image, and thus are excluded from the comparison. Table 2 reports

Table 2: Quantitative comparison on the noisy simulated ToF data in terms of MAE. The best results are highlighted in bold, while the second-best results are underlined.

Method	Art				Book				Dolls				Laundry				Moebius				Reindeer			
	2×	4×	8×	16×	2×	4×	8×	16×	2×	4×	8×	16×	2×	4×	8×	16×	2×	4×	8×	16×	2×	4×	8×	16×
MSJF	2.15	2.64	3.00	4.55	1.14	2.75	2.95	3.13	1.43	1.74	2.71	4.39	2.56	3.28	3.36	4.09	1.19	2.25	3.14	3.77	1.11	2.35	2.57	4.73
GIF	1.39	1.85	3.01	4.83	0.81	1.13	1.90	2.29	1.09	1.89	2.66	3.43	1.20	1.82	2.81	4.15	1.30	2.07	3.06	4.12	1.19	1.84	2.72	3.50
muGIF	0.94	1.45	**1.89**	3.35	0.85	**0.92**	**1.29**	2.43	**0.60**	**0.78**	1.26	2.59	**0.61**	0.83	1.61	2.50	**0.73**	1.05	1.19	2.15	**0.67**	0.88	1.79	2.75
SD	0.89	1.49	2.77	4.38	0.77	0.95	1.44	2.67	0.79	0.85	1.41	2.45	0.65	0.81	1.71	3.76	0.85	1.19	1.72	2.83	0.81	1.13	1.67	3.75
Ours	**0.88**	**1.21**	**1.89**	**3.16**	**0.73**	0.94	1.35	**2.10**	0.64	0.80	**1.22**	**2.33**	0.64	**0.76**	**1.39**	**2.40**	0.76	**0.98**	**1.16**	**2.14**	0.73	**0.87**	**1.49**	**2.46**

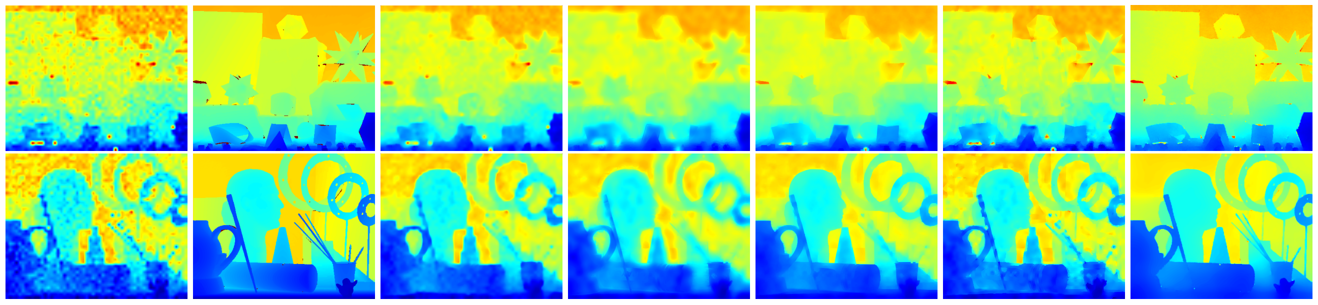


Figure 12: Visual comparison on the 8× Art and Moebius cases. Best viewed in color.

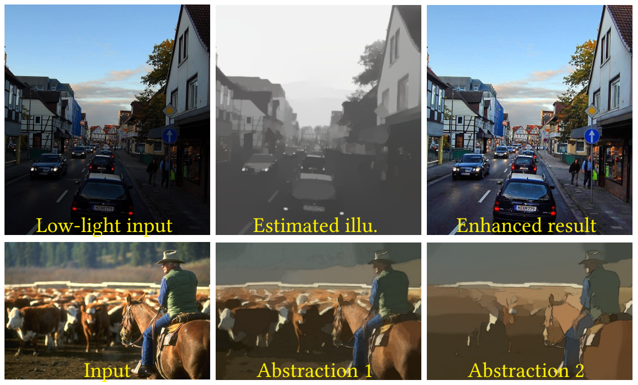


Figure 13: Visual results of low-light enhancement (upper row), and multi-level image abstraction (lower row).

the quantitative results. It can be seen that our method achieves the best/second-best performance in all the cases, thanks to the semantically-meaningful edge guidance from the adjuster. Figure 12 displays the visual results, as can be seen from which, our method can produce better visual effects against the other competitors in preserving sharp depth edges and noise suppression.

4.3.2 Low-light Enhancement. Our model can also be applied to low-light enhancement. We follow [15] to extract initial illumation map then refine it by structure-preserving smoothing. The performance is testified on the LIME [15] and VV datasets in terms of NIQE[33]. Compared to RTV [49], RGF [51], muGIF [16], FIP [6] and Decouple [8], our method can produce reasonable illumination maps, and better enhanced results in NIQE, as shown in Fig. 13 and Tab. 3. Please notice that we do not involve any task-specific modification. More sophisticated designs can be introduced for better performance.

Table 3: Quantitative comparison on the Low-light enhancement task in terms of NIQE↓.

Datasets	RTV	RGF	muGIF	FIP	Decouple	Ours
LIME	3.8586	3.8779	3.7881	3.9079	3.8170	**3.5936**
VV	5.2232	5.3961	5.1941	5.4858	5.1434	**5.1384**

4.3.3 Image Abstraction. Image abstraction is to simplify a complex image to a easy-to-understand one, by suppressing insignificant details while highlighting salient structures. For example, Winnemoller *et al.* [43] proposed to adopt iterative Gaussian filtering to eliminate unnecessary details, and enhance the primary edges extracted by difference-of-Gaussian (DOG). Different from previous approaches, we replace the Bilateral Filters (BF) with our method, and also use our adjuster instead of DOG to extract and highlight primary edges. The second row in Fig. 13 shows two image abstraction results with respect to $\lambda = 0.2$ and $\lambda = 0.7$.

5 CONCLUSION

In this paper, we have proposed a flexible structure-preserving image smoothing/filtering framework. To shrink the solution space, we decomposed the original task into two sub-problems, *i.e.*, guidance generation (adjuster) and conditional smoothing (smoother). Our framework jointly takes the flexibility, visual quality, and efficiency into account, making it attractive for practical use. The advantages of our method against other competitors have been verified by various experiments. We foresee that a wide range of multimedia applications can benefit from our proposed framework.

ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China under Grant 62072327.

REFERENCES

[1] Stefano Allegretti, Federico Bolelli, and Costantino Grana. 2019. Optimized Block-Based Algorithms to Label Connected Components on GPUs. *TPDS* 31, 2 (2019), 423–438.

[2] Soonmin Bae, Sylvain Paris, and Frédo Durand. 2006. Two-scale tone management for photographic look. *TOG* 25 (07 2006), 637–645.

[3] Sai Bi, Xiaoguang Han, and Yizhou Yu. 2015. An L1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *TOG* 34, 4 (2015), 78.

[4] John Canny. 1986. A computational approach to edge detection. *TPAMI* 8, 6 (1986), 679–698.

[5] Chen Cao, Shifeng Chen, Wei Zhang, and Xiaoou Tang. 2011. Automatic motion-guided video stylization and personalization. In *ACM MM*. 1041–1044.

[6] Qifeng Chen, Jia Xu, and Vladlen Koltun. 2017. Fast image processing with fully-convolutional networks. In *ICCV*. 2516–2525.

[7] Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. 2018. Learning to Predict Crisp Boundaries. In *ECCV*, Vol. 11210. 570–586.

[8] Qingnan Fan, Dongdong Chen, Lu Yuan, Gang Hua, Nenghai Yu, and Baoquan Chen. 2019. A general decoupled learning framework for parameterized image operators. *TPAMI* 43, 1 (2019), 33–47.

[9] Qingnan Fan, Jiaolong Yang, Gang Hua, Baoquan Chen, and David P. Wipf. 2017. A Generic Deep Architecture for Single Image Reflection Removal and Image Smoothing. In *ICCV*. 3258–3267.

[10] Qingnan Fan, Jiaolong Yang, David Wipf, Baoquan Chen, and Xin Tong. 2018. Image smoothing via unsupervised learning. *TOG* 37, 6 (2018), 1–14.

[11] Xiao-Nan Fang, Miao Wang, Ariel Shamir, and Shi-Min Hu. 2019. Learning explicit smoothing kernels for joint image filtering. *Computer Graphics Forum* 38, 7 (2019), 181–190.

[12] Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. 2008. Edge-preserving decompositions for multi-scale tone and detail manipulation. *TOG* 27, 3 (2008), 1–10.

[13] Eduardo S. L. Gastal and Manuel M. Oliveira. 2011. Domain transform for edge-aware image and video processing. *TOG* 30, 4 (2011), 69.

[14] Michaël Gharbi, Gaurav Chaurasia, Sylvain Paris, and Frédo Durand. 2016. Deep joint demosaicking and denoising. *TOG* 35, 6 (2016), 1–12.

[15] Xiaojie Guo, Yu Li, and Haibin Ling. 2017. LIME: Low-Light Image Enhancement via Illumination Map Estimation. *TIP* 26, 2 (2017), 982–993.

[16] Xiaojie Guo, Yu Li, Jiayi Ma, and Haibin Ling. 2020. Mutually guided image filtering. *TPAMI* 42, 3 (2020), 694–707.

[17] Bumsub Ham, Minsu Cho, and Jean Ponce. 2017. Robust guided image filtering using nonconvex potentials. *TPAMI* 40, 1 (2017), 192–207.

[18] Kaiming He, Jian Sun, and Xiaoou Tang. 2013. Guided image filtering. *TPAMI* 35, 6 (2013), 1397–1409.

[19] Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. 2013. Fast cost-volume filtering for visual correspondence and veyond. *TPAMI* 35, 2 (2013), 504–511.

[20] Qiming Hu and Xiaojie Guo. 2021. Trash or Treasure? An Interactive Dual-Stream Strategy for Single Image Reflection Separation. In *NeurIPS*. 24683–24694.

[21] Youngjung Kim, Bumsub Ham, Minh N Do, and Kwanghoon Sohn. 2018. Structure-texture image decomposition using deep vriational priors. *TIP* 28, 6 (2018), 2692–2704.

[22] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[23] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *ECCV*, Vol. 8693. 740–755.

[24] Sifei Liu, Jinshan Pan, and Ming-Hsuan Yang. 2016. Learning recursive filters for low-level vision via a hybrid neural network. In *ECCV*, Vol. 9908. 560–576.

[25] Wei Liu, Xiaogang Chen, Chuanhua Shen, Zhi Liu, and Jie Yang. 2017. Semi-global weighted least squares in image filtering. In *ICCV*. 5862–5870.

[26] Wei Liu, Pingping Zhang, Xiaogang Chen, Chunhua Shen, Xiaolin Huang, and Jie Yang. 2020. Embedding bilateral filter in least squares for efficient edge-preserving image smoothing. *TCSVT* 30, 1 (2020), 23–35.

[27] Wei Liu, Pingping Zhang, Xiaolin Huang, Jie Yang, Chunhua Shen, and Ian Reid. 2020. Real-time image smoothing via iterative least squares. *TOG* 39, 3 (2020), 28.

[28] Wei Liu, Pingping Zhang, Yinjie Lei, Xiaolin Huang, Jie Yang, and Ian D. Reid. 2020. A generalized framework for edge-preserving and structure-preserving image smoothing. In *AAAI*. 11620–11628.

[29] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. 2017. Richer convolutional features for edge detection. In *CVPR*. 5872–5881.

[30] Jian-Guang Lou, Hua Cai, and Jiang Li. 2005. A real-time interactive multi-view video system. In *ACM MM*. 161–170.

[31] Kaiyue Lu, Shaodi You, and Nick Barnes. 2017. Deep texture and structure aware filtering network for image smoothing. In *ECCV*. 229–245.

[32] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*. 416–425.

[33] Anish Mittal, Rajiv Soundararajan, and Alan C. Bovik. 2013. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Lett.* 20, 3 (2013), 209–212.

[34] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. 2012. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*. 746–760.

[35] Pietro Perona and Jitendra Malik. 1990. Scale-space and edge detection using anisotropic diffusion. *TPAMI* 12, 7 (1990), 629–639.

[36] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. 2004. Digital photography with flash and no-flash image pairs. *TOG* 23, 3 (2004), 664–672.

[37] Xu Qin, Zhilin Wang, Yuanchao Bai, Xiaodong Xie, and Huizhu Jia. 2020. FFA-Net: Feature Fusion Attention Network for Single Image Dehazing. In *AAAI*. 11908–11915.

[38] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. 2015. Epicflow: edge-preserving interpolation of correspondences for optical flow. In *CVPR*. 1164–1172.

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, Vol. 9351. 234–241.

[40] Xiaoyong Shen, Chao Zhou, Li Xu, and Jiaya Jia. 2015. Mutual-structure for joint filtering. In *ICCV*. 3406–3414.

[41] Zhuo Su, Wenzhe Liu, Zitong Yu, Dewen Hu, Qing Liao, Qi Tian, Matti Pietikainen, and Li Liu. 2021. Pixel Difference Networks for Efficient Edge Detection. In *CVPR*. 5117–5127.

[42] Carlo Tomasi and Roberto Manduchi. 2002. Bilateral filtering for gray and color images. In *ICCV*. 839–846.

[43] Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. 2006. Real-time video abstraction. *TOG* 25, 3 (2006), 1221–1226.

[44] Andrew P. Witkin. 1983. Scale-space filtering. In *IJCAI*. 1019–1022.

[45] Saining Xie and Zhuowen Tu. 2015. Holistically-nested edge detection. In *CVPR*. 1395–1403.

[46] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. 2012. Motion detail preserving optical flow estimation. *TPAMI* 34, 9 (2012), 1744–1757.

[47] Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. 2011. Image smoothing via L0 gradient minimization. *TOG* 30, 6 (2011), 1–12.

[48] Li Xu, Jimmy SJ Ren, Qiong Yan, Renjie Liao, and Jiaya Jia. 2015. Deep edge-aware filters. In *ICML*. 1669–1678.

[49] Li Xu, Qiong Yan, Yang Xia, and Jiaya Jia. 2012. Structure extraction from texture via relative total variation. *TOG* 31, 6 (2012), 139.

[50] Kuk-Jin Yoon and In So Kweon. 2006. Adaptive support-weight approach for correspondence search. *TPAMI* 28, 4 (2006), 650–656.

[51] Qi Zhang, Xiaoyong Shen, Li Xu, and Jiaya Jia. 2014. Rolling guidance filter. In *ECCV*. 815–830.