

# Unsharp Mask Guided Filtering

Zenglin Shi, Yunlu Chen, Efstratios Gavves, Pascal Mettes, and Cees G. M. Snoek, *Senior Member, IEEE*

**Abstract**—The goal of this paper is guided image filtering, which emphasizes the importance of structure transfer during filtering by means of an additional guidance image. Where classical guided filters transfer structures using hand-designed functions, recent guided filters have been considerably advanced through parametric learning of deep networks. The state-of-the-art leverages deep networks to estimate the two core coefficients of the guided filter. In this work, we posit that simultaneously estimating both coefficients is suboptimal, resulting in halo artifacts and structure inconsistencies. Inspired by unsharp masking, a classical technique for edge enhancement that requires only a single coefficient, we propose a new and simplified formulation of the guided filter. Our formulation enjoys a filtering prior from a low-pass filter and enables explicit structure transfer by estimating a single coefficient. Based on our proposed formulation, we introduce a successive guided filtering network, which provides multiple filtering results from a single network, allowing for a trade-off between accuracy and efficiency. Extensive ablations, comparisons and analysis show the effectiveness and efficiency of our formulation and network, resulting in state-of-the-art results across filtering tasks like upsampling, denoising, and cross-modality filtering. Code is available at <https://github.com/shizenglin/Unsharp-Mask-Guided-Filtering>.

## I. INTRODUCTION

IMAGE filtering has been widely used to suppress unwanted signals (e.g., noise) while preserving the desired ones (e.g., edges) in image processing tasks like image restoration [1]–[3], boundary detection [4]–[6], texture segmentation [7]–[9], and image detail enhancement [10]–[12]. Standard filters, such as Gaussian filters and box mean filters, swiftly process input imagery but suffer from content-blindness, *i.e.*, they treat noise, texture, and structure identically. To mitigate content-blindness, guided filters [13]–[18], have received a great amount of attention from the community. The key idea of guided filtering is to leverage an additional guidance image as a structure prior and transfer the structure of the guidance image to a target image. By doing so, it strives to preserve salient features, such as edges and corners, while suppressing noise. The goal of this paper is guided image filtering.

Classical guided filtering, *e.g.*, [13], [19]–[21], performs structure-transferring by relying on hand-designed functions. Nonetheless, it is known to suffer from halo artifacts and structure inconsistency problems (see Fig. 1), and it may require a considerable computational cost. In recent years, guided image filtering has advanced by deep convolutional neural networks. Both Li *et al.* [22] and Hui *et al.* [23] demonstrate the benefits of learning-based guided filtering over classical guided filtering. These works and their follow-ups, *e.g.*, [24]–[26], directly predict the filtered output by means of feature fusion from the guidance and target images. Yet,

this implicit way of structure-transferring may fail to transfer the desired edges and may suffer from transferring undesired content to the target image [26], [27].

Pan *et al.* [27] propose an alternative way to perform deep guided filtering. Rather than directly predicting the filtered image, they leverage a shared deep convolutional neural network to estimate the two coefficients of the original guided filtering formulation [13]. While their approach obtains impressive filtering results in a variety of applications, we observe their network has difficulty disentangling the representations of the two coefficients, resulting in halo artifacts and structure inconsistencies, see Fig. 1. Building on the work of Pan *et al.* [27], we propose a new guided filtering formulation, which depends on a *single* coefficient and is therefore more suitable to be solved by a single deep convolutional neural network.

We take inspiration from another classical structure-transferring filter: unsharp masking [28]–[31]. From the original guided filter by He *et al.* [13] we first derive a simplified guided filtering formulation by eliminating one of its two coefficients. So there is only one coefficient left to be estimated for deciding how to perform edge enhancement, akin to unsharp masking. To arrive at our formulation, we rely on the filtering prior used in unsharp masking and perform guided filtering on the unsharp masks rather than the raw target and guidance images themselves. The proposed formulation enables us to intuitively understand how the guided filter performs edge-preservation and structure-transferring, as there is only one coefficient in the formulation, rather than two in [27]. The coefficient explicitly controls how structures need to be transferred from guidance image to target image and we learn it adaptively by a single network. To that end, we introduce a successive guided filtering network. The network obtains multiple filtering results by training a single network. It allows a trade-off between accuracy and efficiency by choosing different filtering results during inference. This leads to fast convergence and improved performance. Experimental evaluation on seven datasets shows the effectiveness of our proposed filtering formulation and network on multiple applications, such as upsampling, denoising, and cross-modality filtering.

## II. BACKGROUND AND RELATED WORK

In guided filtering, we are given an image pair  $(I, G)$ , where the image pair has been properly registered by default. Image  $I$  needs to be filtered, *e.g.*, due to the presence of noise or due to low resolution because it is captured by a cheap sensor. Guidance image  $G$  contains less noise and is of high resolution with sharp edges, *e.g.*, because it is captured by accurate sensors under good light conditions. The low-quality image  $I$  can be enhanced by filtering under the guidance of high-quality image  $G$ . Such a guided filtering process is defined

The authors are with the Informatics Institute of the University of Amsterdam, Amsterdam, the Netherlands.

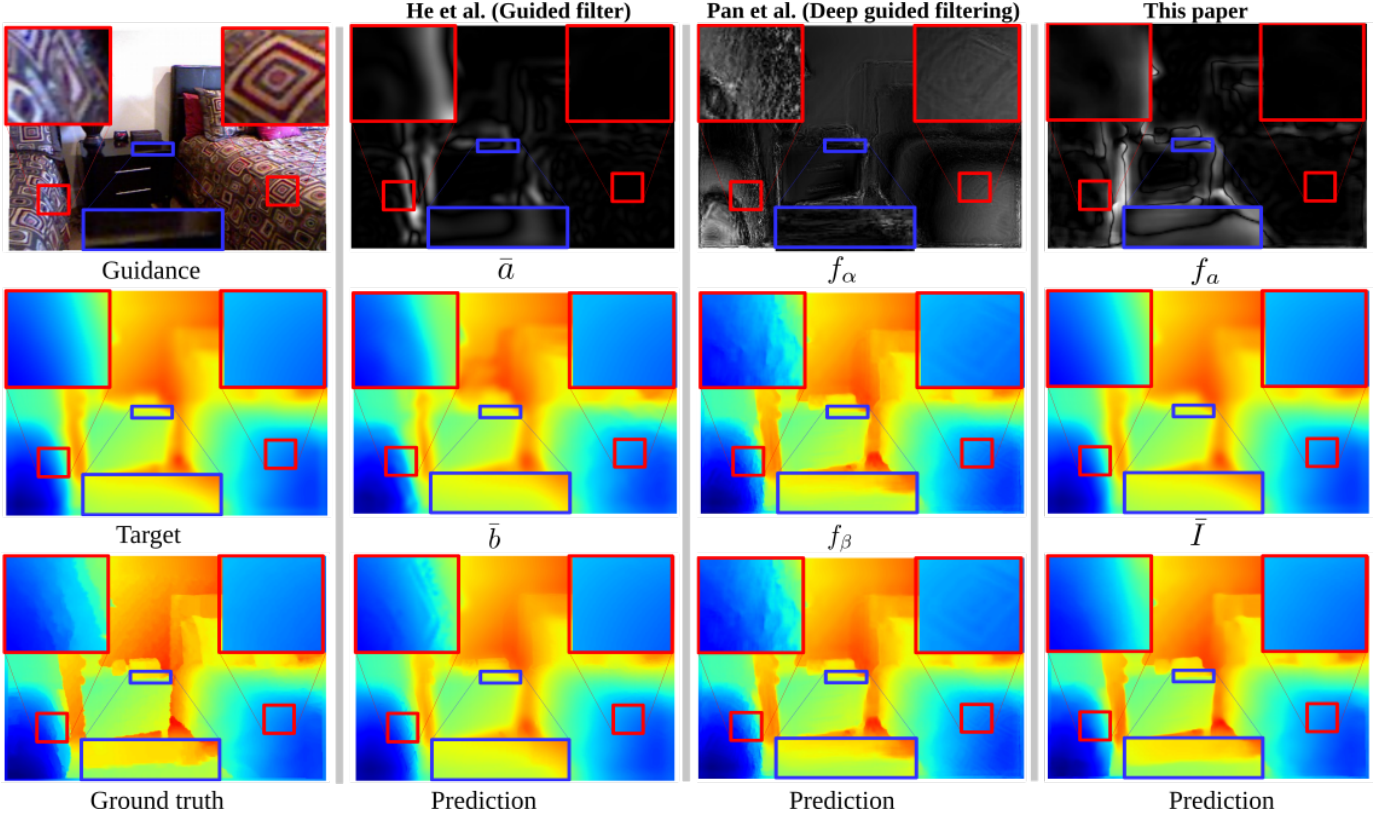


Fig. 1: **Motivation of this paper.** We show an example of depth upsampling ( $16\times$ ) using an RGB image as guidance. Both the conventional guided filter by He *et al.* [13] and the state-of-the-art deep guided filter by Pan *et al.* [27] explicitly estimate two coefficients, respectively  $(\bar{a}, \bar{b})$  and  $(f_\alpha, f_\beta)$ . In their current formulation, however, both methods are likely to over-smooth edges (compare edges in blue boxes) and transfer unwanted textures (compare highlighted details in red boxes). Our proposed guided filter, taking inspiration from unsharp masking, only requires learning a single coefficient  $f_a$  (notation details provided in Sections 2 and 3). As a result, we obtain a more desirable upsampling result, free of undesirable structures and textures from the guidance image.

as  $\hat{I} = \mathcal{F}(I, G)$ , where  $\mathcal{F}(\cdot)$  denotes the filter function and  $\hat{I}$  denotes the filtered output image. Below, we first review the benefits and weaknesses of classical guided filter functions and existing deep guided filter functions. We then present how our proposal can be an improved guided filtering solution by establishing a link to unsharp masking.

#### A. Classical guided filtering

The guided image filter [13] assumes that the filtered output image  $\hat{I}$  is a linear transform of the guidance image  $G$  at a window  $w_k$  centered at pixel  $k$ :

$$\hat{I}_i = a_k G_i + b_k, \quad \forall i \in w_k, \quad (1)$$

where  $a_k$  and  $b_k$  are two constants in window  $w_k$ . Their values can be obtained by solving:

$$E(a_k, b_k) = \sum_{i \in w_k} ((a_k G_i + b_k - I_i)^2 + \epsilon a_k^2). \quad (2)$$

Here,  $\epsilon$  is a regularization parameter penalizing large values for  $a_k$ . The optimal values of  $a_k$  and  $b_k$  are computed as:

$$a_k = \frac{\frac{1}{|w|} \sum_{i \in w_k} I_i G_i - \bar{I}_k \bar{G}_k}{\sigma_k^2 + \epsilon}, \quad (3)$$

$$b_k = \bar{I}_k - a_k \bar{G}_k. \quad (4)$$

Here,  $\bar{G}_k$  and  $\sigma_k^2$  are the mean and variance of  $G$  in  $w_k$ ,  $\bar{I}_k$  is the mean of  $I$  in  $w_k$ , and  $|w|$  is the number of pixels in  $w_k$ . For ease of analysis, the guidance image  $G$  and filtering target image  $I$  are assumed to be the same [13], although the general case remains valid. As a result, we can obtain:

$$a_k = \sigma_k^2 / (\sigma_k^2 + \epsilon), \quad b_k = (1 - a_k) \bar{G}_k. \quad (5)$$

Based on this, the regions and edges with variance ( $\sigma_k^2$ ) much larger than  $\epsilon$  are preserved, whereas those with variance ( $\sigma_k^2$ ) much smaller than  $\epsilon$  are smoothed. Hence,  $\epsilon$  takes control of the filtering. However, the value of  $\epsilon$  in the guided image filter [13] is fixed. As such, halos are unavoidable when the filter is forced to smooth some edges [13], [19]–[21]. An edge-aware weighted guided image filter is proposed in [20] to overcome this problem, where  $\epsilon$  varies spatially rather than being fixed:

$$\epsilon = \frac{\lambda}{\Gamma_G}, \quad \Gamma_G = \frac{\sigma_k^2 + \epsilon}{\frac{1}{N} \sum_{k=1}^N \sigma_k^2 + \epsilon}, \quad (6)$$

where  $\lambda$  and  $\epsilon$  are two small constants and  $\sigma_k^2$  is the variance of  $G$  in a  $3 \times 3$  window centered at the pixel  $k$ .  $\Gamma_G$  measures the importance of a pixel  $k$  with respect to the

whole guidance image. Kou *et al.* [19] propose a multi-scale edge-aware weighted guided image filter, in which  $\Gamma_G$  is computed by multiplying the variances of multiple windows. By taking the edge direction into consideration, Sun *et al.* [21] further improve the filter's performance. However, predefined parameters still remain in all these three methods.

Another limitation of classical guided image filters is their assumption that the target image and the guidance image have the same structure. In practice, there are also situations conceivable where an edge appears in one image, but not in the other. To address this issue, recent works [32]–[36] enhance guided image filters by considering the mutual structure information in both the target and the guidance images. These methods typically build on iterative algorithms that minimize global objective functions. The guidance signals are updated at each iteration to enforce the outputs to have similar structure as the target images. These global optimization methods generally use hand-crafted objectives that usually involve fidelity and regularization terms. The fidelity term captures the consistency between the filtering output and the target image. The regularization term, typically modeled using a weighted L2 norm [37], encourages the filtering output to have a similar structure as the guidance image. However, such hand-crafted regularizers may not transfer the desired structures from the guidance image to the filtering output [26], [27]. For this reason we prefer a guided filtering solution based on end-to-end deep representation learning.

### B. Deep guided filtering

Li *et al.* [22], [26] introduce the concept of deep guided image filtering based on an end-to-end trainable network. Two independent convolutional networks  $f_I$  and  $f_G$  first process the target image and guidance image separately. Then their outputs from the last layers are concatenated and forwarded to another convolutional network  $f_{IG}$  to generate the final output. We define the method as:

$$\hat{I} = f_{IG}(f_I(I) \oplus f_G(G)), \quad (7)$$

where  $\oplus$  denotes the channel concatenation operator. Several works follow the same definition but vary in their feature fusion strategies [23]–[25]. AlBahar *et al.* [24] introduce a bi-directional feature transformer to replace the concatenation operation. Su *et al.* [25] propose a pixel-adaptive convolution to fuse the outputs of networks  $f_I$  and  $f_G$  adaptively on the pixel level. Hui *et al.* [23] propose to perform multi-scale guided depth upsampling based on spectral decomposition, where low-frequency components of the target images are directly upsampled by bicubic interpolation and the high-frequency components are upsampled by learning two convolutional networks. The high-frequency domain learning leads to an improved performance. Wu *et al.* [38] alternatively combine convolutional networks and traditional guided image filters. Two independent convolutional networks  $f_I$  and  $f_G$  first amend the target image and the guidance image, and then feed their outputs into the traditional guided image filter  $\mathcal{F}_{IG}$  [13]:

$$\hat{I} = \mathcal{F}_{IG}(f_I(I), f_G(G)). \quad (8)$$

Rather than directly predicting the filtered image, Pan *et al.* [27] leverage deep neural networks to estimate the two coefficients of the original guided filtering formulation [13] based on a spatially-variant linear representation model, leading to impressive filtering results. It is defined as:

$$\hat{I} = f_\alpha(I, G) \odot G + f_\beta(I, G), \quad (9)$$

where  $f_\alpha$  and  $f_\beta$  are two convolutional networks, and  $\odot$  denotes element-wise multiplication. To reduce the complexity of learning, in the implementation Pan *et al.* [27] learn a single network and predict an output with two channels, one channel for  $f_\alpha$  and another channel for  $f_\beta$ . However, we observe that the shared network has difficulty disentangling the representations of the two coefficients, resulting in halo artifacts and structure inconsistencies. Differently, we propose a new guided filtering formulation, which depends on a *single* coefficient only and is therefore more suitable to be solved by a single deep convolutional neural network.

The aforementioned existing deep guided filtering works implicitly perform structure-transferring by learning a joint filtering network, usually resulting in undesired filtering performance [27], [39], [40]. Recent works [39]–[45] take inspiration from coupled dictionary learning [46], and incorporate sparse priors into their deep networks for explicit structure-transferring. Marivani *et al.* [40]–[43] propose a learned multimodal convolutional sparse coding network with a deep unfolding method for explicitly fusing information from the target and guidance image modalities. Deng *et al.* propose a deep coupled ISTA network with a multimodal dictionary learning algorithm [45], and a common and unique information splitting network with multi-modal convolutional sparse coding [39], for the sake of explicitly modeling the knowledge from the guidance image modality. Most of these works focus on guided image super-resolution. In this work, we propose an explicit structure-transferring method for general guided filtering problems. In particular, we propose a guided filtering formulation with a single coefficient, and we learn to estimate the coefficient to explicitly decide how to transfer the desirable structures from guidance image to target image. As we will demonstrate, this leads to more desirable filtering results.

### C. Unsharp masking

Our formulation is inspired by the classical sharpness enhancement technique of unsharp masking [28]–[31], which can be described by the equation:

$$\hat{I} = \lambda(I - \mathcal{F}_L(I)) + I, \quad (10)$$

where an enhanced image is represented by  $\hat{I}$ , an original image by  $I$ , an unsharp mask by  $(I - \mathcal{F}_L(I))$  where  $\mathcal{F}_L$  denotes a low-pass filter like Gaussian filters or box mean filters, and an amount coefficient by  $\lambda$  which controls the volume of enhancement achieved at the output. Essentially, guided filtering shares the same function of edges enhancement as unsharp masking by means of the structure-transferring from an additional guidance image. Based on this viewpoint, we derive a simplified guided filtering formulation from the original guided filter [13], with only one coefficient to be estimated, akin to the formulation of unsharp masking in Eq. (10).

### III. FILTERING FORMULATION

Here, we outline our new guided filtering formulation, in which only one coefficient needs to be estimated. Compared to estimating two coefficients ( $a$ ,  $b$ ) as in the original guided filtering formulation and subsequent deep learning variants, our formulation is more suitable to be solved with one single deep network. We start the derivation of our guided filtering formulation from the classical guided filter [13], summarized in Eq. (1), Eq. (3) and Eq. (4). In Eq. (1),  $\hat{I}$  is a linear transform of  $G$  in a window  $w_k$  centered at the pixel  $k$ . When we apply the linear model to all local windows in the entire image, a pixel  $i$  is involved in all the overlapping windows  $w_k$  that covers  $i$ . In this case, the value of  $\hat{I}_i$  in Eq. (1) is not identical when it is computed in different windows. So after computing  $(a_k, b_k)$  for all windows  $w_k$  in the image, we compute the filtered output image  $\hat{I}_i$  by averaging all the possible values of  $\hat{I}_i$  with:

$$\hat{I}_i = \frac{1}{|w|} \sum_{k \in w_i} (a_k G_i + b_k). \quad (11)$$

Similar in spirit to unsharp masking, summarized in Eq. (10), we want to maintain only the coefficient  $a$  to control the volume of structure to be transferred from guidance  $G$  to the filtered output image  $\hat{I}$ . Thus, we put Eq. (4) into Eq. (11) to eliminate  $b$ , and obtain:

$$\hat{I}_i = \frac{1}{|w|} \sum_{k \in w_i} a_k G_i + \frac{1}{|w|} \sum_{k \in w_i} (\bar{I}_k - a_k \bar{G}_k). \quad (12)$$

Next, we rewrite the formulation as:

$$\hat{I}_i = \frac{1}{|w|} \sum_{k \in w_i} a_k (G_i - \bar{G}_k) + \tilde{I}_i, \quad (13)$$

where  $\tilde{I}_i = \frac{1}{|w|} \sum_{k \in w_i} \bar{I}_k$ . Since  $\bar{G}_k$  is the output of a mean filter, it's assumed that  $\bar{G}_k$  is close to its mean in the window  $w_i$ . Next we rewrite Eq. (13) as follows

$$\hat{I}_i = \bar{a}_i (G_i - \tilde{G}_i) + \tilde{I}_i, \quad (14)$$

where  $\bar{a}_i = \frac{1}{|w|} \sum_{k \in w_i} a_k$ , and  $\tilde{G}_i = \frac{1}{|w|} \sum_{k \in w_i} \bar{G}_k$ . For convenience, we will omit subscript  $i$  in the following.

The formulation in Eq. (14) enables us to intuitively understand how the guided filter performs edge-preservation and structure-transferring. Specifically, the target image  $I$  is first smoothed to remove unwanted components like noise/textures, and the smoothing result is denoted by  $\tilde{I}$ . However, the smoothing process usually suffers from the loss of sharp edges, leading to a blurred output. To enhance the edges, an unsharp mask ( $G - \tilde{G}$ ) with fine edges generated from the guidance image  $G$  is added to  $\tilde{I}$  under the control of the coefficient  $a$ , leading to the structure being transferred from the guidance image to the filtered output image  $\hat{I}$ . Finally, we rewrite Eq. (14) to obtain a more general formulation for deep guided filtering:

$$\hat{I} = f_a(I_m, G_m) \odot G_m + \mathcal{F}_L(I), \quad (15)$$

where  $I_m = I - \mathcal{F}_L(I)$  and  $G_m = G - \mathcal{F}_L(G)$  denote the unsharp masks of the target image and the guidance image, which contain the structures of the guidance and the target images.  $\mathcal{F}_L$  denotes a linear shift-invariant low-pass filter like

the Gaussian filter or the box mean filter.  $f_a$  denotes the amount function, which controls the volume of structure to be transferred from the guidance image to the filtered output image. Next, we will elaborate on this function.

**Amount function  $f_a$ .** The output of  $f_a$  is the volume of the structure to be transferred from the guidance image to the filtered output image. Thus, the input of  $f_a$  should involve the structure of both the target and the guidance image, which together determine the output, i.e.,  $f_a(I_m, G_m)$ . Ideally,  $f_a$  should determine the structure-transferring in a pixel-adaptive fashion. It can be a manually designed function as the function  $a$  of the guided filter in Eq. (3). It also can be estimated by learning a deep neural network. Compared to hand-crafted functions, learnable functions are more flexible and allow for a better generalization to various image conditions.

**Successive filtering.** Successive operations of the same filter generally result in a more desirable output, thus we develop a successive guided filtering based on our formulation in Eq. (15). Instead of directly iterating the filtering output  $\hat{I}$ , we iterate the outputs of  $f_a$  as the function decides the effect of filtering. Let  $f_a^*$  be a composition of a set of basic functions  $\{f_a^{(l)}\}_{l=1}^L$ :

$$f_a^* = f_a^{(L)} \circ f_a^{(L-1)} \circ \dots \circ f_a^{(1)} \quad (16)$$

in which  $\circ$  denotes the function composition operation, such as  $(f \circ u)(\cdot) = f(u(\cdot))$ . With  $f_a^*$  we obtain a successive filtering formulation from Eq. (15),

$$\hat{I} = f_a^*(I_m, G_m) \odot G_m + \mathcal{F}_L(I). \quad (17)$$

In the next section we will detail how to implement our filters with deep convolutional neural networks.

### IV. FILTERING NETWORK

There is a function  $f_a$  in our formulation, which governs a guided filtering coefficient. We propose to solve this function with a single convolutional neural network.

**Network for amount function  $f_a$ .** The function  $f_a$  decides how to transfer the structure of the guidance image to the filtered output image. There are two inputs  $G_m$  and  $I_m$  for this function. Two options are available for the network architecture. Like [23], [26], we can separately process these two inputs with two different sub-networks at first, and then fuse their outputs with another sub-network. Alternatively, we can concatenate these two inputs together and forward the concatenation into a single network, similar to the framework of [27]. Empirically, we find that the second option is easily implemented and achieves a desirable filtering accuracy and efficiency. Thus, we design the network of  $f_a$  with the second option in this work.

In our approach the unsharp masks, rather than the raw images themselves, are used as the inputs of the network. The unsharp mask is more sparse than the raw image itself, since most regions in the unsharp mask are close to zero. The learning of spatially-sparse data usually requires a large receptive field. The dilated convolution is a popular technique to enlarge the receptive fields in convolutional networks [47], [48]. We design our network by cascading four dilated convolutional layers with increasing dilation factors, where all the dilated



convolutional layers have the same channel number of 24 and the same kernel size of  $3 \times 3$ . Their dilation factors are set to  $\{1, 2, 4, 8\}$ . Leaky ReLU with a negative slope of 0.1 is used as an activation function after each dilated convolutional layer. Finally, a  $1 \times 1$  convolution layer generates the target output for  $f_a$ .

**Network for successive filtering.** To develop a network for the successive filtering formulation in Eq. (17), we consider the network of the basic functions  $\{f_a^{(l)}\}_{l=1}^L$  as a convolutional block, as shown in Fig. 2 (a). Then stacking this block multiple times results in a deep network for  $f_a^*$ . There are two outputs in the block. By concatenating its input and its feature maps from the last layer results in the first output. The concatenation output allows feeding the previous multi-level outputs to the following blocks, leading to improved performance. We develop the second output by using a convolutional layer on top of the last layer of the block, for the sake of balancing accuracy and efficiency.

Stacking more blocks results in higher accuracy at the expense of an increased computational complexity. To allow users to choose between accuracy and computational complexity, we generate filtering outputs from each block. If users want to obtain filtering results fast, the filtering results from the first blocks can be used. When the accuracy is more important, the filtering results from the later blocks can be used. In short, we obtain multiple filtering results while training one single network. The overall network architecture is visualized in Fig. 2 (b).

**Optimization.** During training, we are given  $N$  samples  $\{(I_n, G_n, Z_n)\}_{n=1}^N$ , with  $I_n \in \mathcal{I}$  the target image,  $G_n \in \mathcal{G}$  the guidance image and  $Z_n \in \mathcal{Z}$  the task-dependent ground-truth output image. Our goal is to learn the parameters  $\theta$  of the network for  $f_a$ . Two types of loss,  $L_1$  loss and  $L_2$  loss, have been widely used in deep guided filtering works. Early works, like [22], [23], have adopted a  $L_2$  loss, while recent works, like [24], [27], prefer a  $L_1$  loss because it is less sensitive to outliers and leads to less blurry results compared to a  $L_2$  loss [49], [50]. Following these recent works, we minimize the difference between filtered output image  $\hat{I}$  and its ground-truth  $Z$  using the  $L_1$  loss, which is defined as:

$$\mathcal{L}(I, G, Z; \theta) = \frac{1}{N} \sum_{n=1}^N \|\hat{I}_n(I_n, G_n; \theta) - Z_n\|_1. \quad (18)$$

## V. EXPERIMENTS

In this section, we provide extensive experimental evaluations. Section V-A introduces the experimental setup. Sections V-B, V-C, V-D, and V-E emphasize ablations, comparisons and analysis. Sections V-F, V-G, and V-H show further qualitative and quantitative results, and state-of-the-art comparisons on various applications, including upsampling, denoising, and cross-modality filtering.

### A. Experimental setup

**Image upsampling datasets.** We perform upsampling experiments on NYU Depth V2 [51], and Sintel optical flow

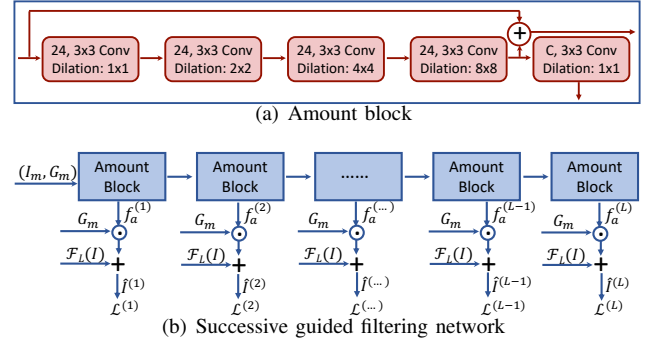


Fig. 2: **Network architecture for unsharp-mask guided filtering.** a) Dilated convolutional block for amount function  $f_a$ ; b) Network architecture for successive unsharp-mask guided filtering. Here,  $\odot$  denotes the element-wise product,  $\oplus$  denotes the concatenation operation,  $+$  denotes the element-wise sum. Leaky ReLU is used as activation function after each convolutional layer. There are  $(L - 4)$  amount blocks in the box, indicated with dots.  $\mathcal{L}$  denotes the loss function in Eq. (18). Here,  $G_m$  and  $\mathcal{F}_L(I)$  are shared by  $\{f_a^{(l)}\}_{l=1}^L$ .

[52]. For NYU Depth V2 we follow [26]. We use the first 1000 RGB/depth pairs for training and the remaining 449 pairs for testing, where each low-resolution depth image is generated by applying a nearest-neighbor interpolation. For Sintel, following [25], 908 samples, and 133 samples from clean pass are used for training, and testing, where each low-resolution flow image is generated by applying a bilinear interpolation.

**Image denoising datasets.** We perform denoising experiments also on NYU Depth V2 [51], as well as on BSDS500 [53]. For NYU Depth V2, we use the same split as for upsampling. BSDS500 contains 500 natural images. We train on the training set (200 images) and the validation set (100 images). We evaluate on the provided test set (200 images). Following [54], to train a blind Gaussian denoiser, random Gaussian noise is added to the clean training depth images in NYU Depth V2 and the clean RGB images in BSDS500, with a noise level  $\sigma \in [0, 55]$ . For testing, we consider three noise levels,  $\sigma = \{15, 25, 50\}$ . Thus, three separate noisy test images are generated for each original test image.

**Pre-processing.** For all datasets, we normalize the input images by scaling their pixel values to the range  $[0, 1]$ . According to Eq. (15), the guidance image  $G$  and target image  $I$  should have the same number of channels. In the depth/RGB dataset, the target is the 1-channel depth image. Thus, rgb2gray operation is used to transform a 3-channel RGB image into a 1-channel grayscale image as guidance. During training, we augment the images by randomly cropping  $256 \times 256$  patches. No cropping is performed during testing.

**Network and optimization.** The proposed network is optimized in an end-to-end manner. We implement the network with TensorFlow on a machine with a single GTX 1080 Ti GPU. The optimizer is Adam with a mini-batch of 1. We set  $\beta_1$  to 0.9,  $\beta_2$  to 0.999, and the initial learning rate to 0.0001. Optimization is terminated after 1000 epochs.

TABLE I: **Quantitative results** for image denoising on BSDS500 and depth upsampling on NYU Depth V2. When the functions  $a$  of the guided filter [13] and the weighted guided filter [20] are used for  $f_a$ , our filter is denoted by “Ours + GF” and “Ours + WGF”, respectively. Our filters perform at least as good as the baselines.

|               | Denoising (PSNR) $\uparrow$ |               |               | Upsampling (RMSE) $\downarrow$ |              |              |
|---------------|-----------------------------|---------------|---------------|--------------------------------|--------------|--------------|
|               | $\sigma = 15$               | $\sigma = 25$ | $\sigma = 50$ | $4\times$                      | $8\times$    | $16\times$   |
| Bicubic/Input | 24.61                       | 20.17         | 14.15         | 8.21                           | 14.03        | 22.48        |
| GF [13]       | 29.16                       | 26.47         | 23.82         | 7.25                           | 12.38        | 19.86        |
| Ours + GF     | 29.24                       | 26.59         | 23.84         | 7.18                           | <b>12.28</b> | <b>19.75</b> |
| WGF [20]      | <b>29.40</b>                | 26.92         | 23.98         | <b>7.17</b>                    | 12.33        | 19.79        |
| Ours + WGF    | 29.35                       | <b>26.96</b>  | <b>23.99</b>  | 7.18                           | 12.30        | 19.76        |

**Evaluation metrics.** To evaluate the quality of the predicted images we report four standard metrics: RMSE (Root Mean Square Error) for depth upsampling, EPE (End-Point-Error) for flow upsampling, PSNR (Peak Signal-to-Noise Ratio) for denoising, and SSIM (Structural Similarity Index Measure) for all applications.

### B. Unsharp-mask guided filtering without learning.

The goal of the first experiment is to validate our formulation as a valid guided filter. Here we do not rely on any deep learning for estimating  $f_a$ . Instead, we use the function  $a$  of the guided filter [13] and the weighted guided filter (WGF) [20] as  $f_a$ .  $\mathcal{F}_L(G)$  and  $\mathcal{F}_L(I)$  are generated by cascaded box mean filters. Using our formulation as a conventional guided filter, we provide qualitative and quantitative results to demonstrate that our filter performs as good as, or even better than the guided filter [13] and the weighted guided filter [20].

**Qualitative results.** The first example performs edge-preserving smoothing on a gray-scale image. The second example is about detail enhancement. For both, the target image and guided image are identical. Fig. 3 and Fig. 4 show the results of filtering, where we can see that our filter performs as good as the guided filter [13] in preserving structures. In the third example, we denoise a no-flash image under the guidance of its flash version to verify the effect of structure-transferring. The denoising results of our filter and the guided filter [13] in Fig. 5 are consistent and don’t have gradient reversal artifacts.

**Quantitative results.** Next, we compare our filter with the guided filter (GF) [13] and the weighted guided filter (WGF) [20] for natural image denoising on BSDS500 and depth upsampling on NYU Depth V2. There are two hyperparameters,  $r$  and  $\epsilon$  in GF and WGF. Grid-search is used to find the optimal hyperparameters. The results in Table I show that our filter performs at least as good as the guided filter [13] and the weighted guided filter [20], indicating that our formulation makes sense as a guided filter.

### C. Unsharp-mask guided filtering with learning

Next, we assess the benefit of our formulation when  $f_a$  is learned by a neural network. We compare to four baselines: (i) DMSG [23], (ii) DGF [38] (iii) DJF [26], and (iv) SVLRM [27]. The experiments are performed on NYU Depth V2 [51]

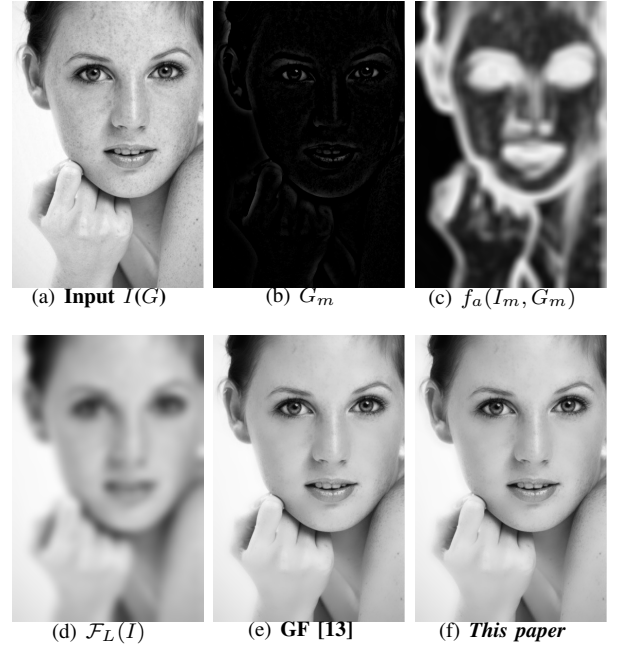


Fig. 3: **Edge-preserving filtering.** a) Target and guidance image  $I = G$ ; b)  $G_m$  containing the important structures. c)  $f_a(I_m, G_m)$  estimated by Eq. (3) with  $\epsilon = 0.05^2$ ; d)  $\mathcal{F}_L(I)$  obtained by a cascade of two box filters with radius  $r = 8$ ; e) The smoothing result obtained by the guided filter [13]; f) Our smoothing result. Both our filter and guided filter [13] can preserve good edges while removing noise.



Fig. 4: **Detail enhancement.** The parameters are  $r = 16$ ,  $\epsilon = 0.1^2$ . Our filter without learning, as defined in Eq. (14), performs as good as the guided filter [13].

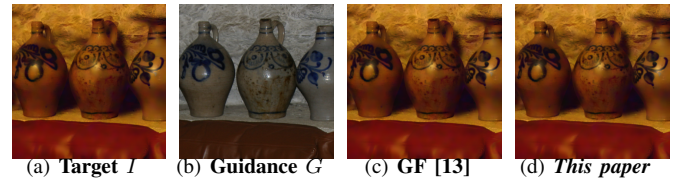


Fig. 5: **Flash/no-flash denoising.** The parameters are  $r = 8$ ,  $\epsilon = 0.2^2$ . Our filter without learning, as defined in Eq. (14), performs as good as the guided filter [13].

for depth upsampling ( $16\times$ ) and depth denoising ( $\sigma = 50$ ).

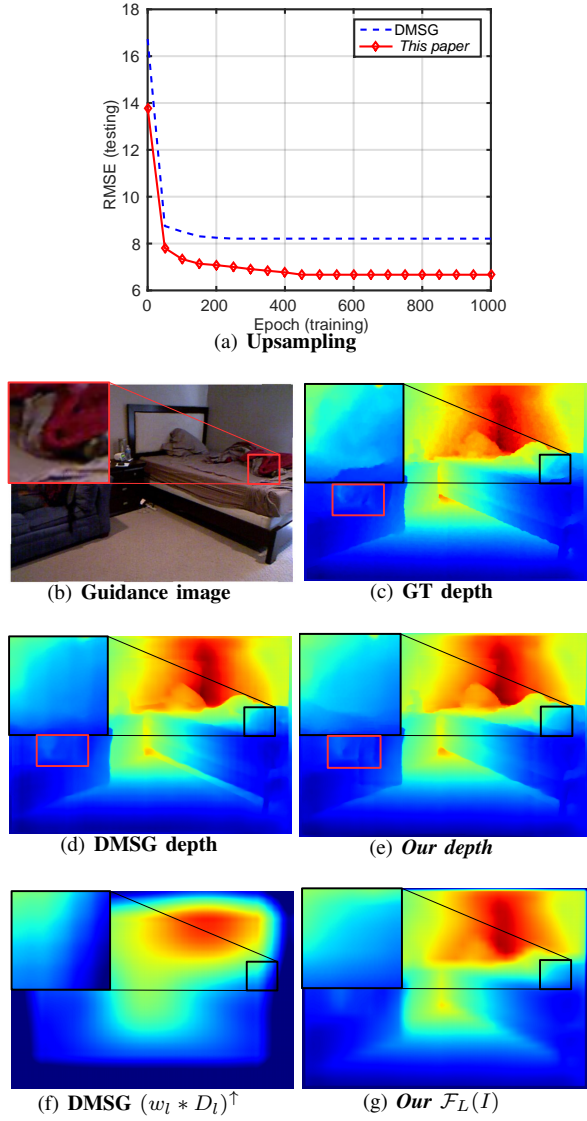


Fig. 6: **Comparison with DMSG [23]**. Compared to DGF, our approach better recovers finer edges as shown in the regions marked by the red boxes, and avoids producing artifacts as shown in the regions marked by the black boxes.

We compare these baselines separately. For each comparison, the network for  $f_a$  is the same as the network used in the compared method. We use a box mean filter with radius  $r = 8$  to obtain  $\mathcal{F}_L(I)$  and  $\mathcal{F}_L(G)$ . For depth upsampling ( $16\times$ ), we first upsample the low-resolution depth image by bicubic interpolation to obtain the target resolution for  $\mathcal{F}_L(I)$ . We also use the upsampled depth image as the input of the network, following [26], [27], [38]. One exception is the comparison with DMSG [23] which uses the original low-resolution depth image as the input of the network.

**Comparison with DMSG [23]**. Fig. 6 (a) demonstrates our approach achieves better upsampling results than DMSG [23] in terms of RMSE. DMSG performs depth upsampling based on spectral decomposition. Specifically, a low-resolution depth image is first decomposed into low-frequency components and high-frequency components. The low-frequency

components are directly upsampled by bicubic interpolation. The high-frequency components are upsampled by learning two convolutional networks. The first network is used to obtain multi-scale guidance. The other one performs multi-scale joint upsampling. The difference between our method and DMSG mainly lies in two aspects. First, we don't use the first network and just use the second network for amount function  $f_a$  to explicitly perform structure transfer instead of directly predicting the filtered output image. We find that our approach avoids halo effects more successfully, as shown in Fig. 6 (d) and (e). Second, Hui *et al.* use a Gaussian filter to smooth the low-resolution target depth image when generating its low-frequency components. After that, they upsample the low-frequency components by bicubic interpolation. However, this step is likely to produce artifacts, as shown in Fig. 6 (f). Since the network learning focuses on upsampling high-frequency components, the artifacts still remain in the final upsampling output, as shown in Fig. 6 (d). By contrast, we first upsample the low-resolution target depth image before smoothing. By doing so, the artifacts generated by bicubic interpolation can be removed by smoothing, as shown in Fig. 6 (g).

**Comparison with DGF [38]**. Wu *et al.* [38] learn two networks to amend the guidance and target images before feeding them to the guided filter [13]. The learned guidance and target images fit the guided filter [13] better than the original ones. However, DGF still suffers from the halo problem since its final filtering output is generated by the guided filter [13]. Our approach performs better than DGF [38] for both upsampling and denoising tasks, as demonstrated in Fig. 7 (a) and (b). As shown in Fig. 7 (e) and (g), the important edges are unavoidable to be smoothed because the structure-transferring is performed in an undesirable fashion. By contrast, our approach performs better on preserving and transferring important structures, as shown in 7 (f) and (h), as the amount function  $f_a$  is learned in a pixel-adaptive way through a deep neural network instead of designed manually.

**Comparison with DJF [26]**. The network in our method directly uses the unsharp masks of the target image and the guided image as inputs, and learns to estimate the amount function  $f_a$  for explicitly deciding how to transfer the desired structure from the guidance image to the target image. By contrast, the network in DJF [26] uses the original guidance image and target image as inputs, and directly predicts filtered output relying on feature fusion. The implicit structure transfer is likely to cause slow convergence and the unwanted contents to be transferred from guidance image to target image, as shown in Fig. 8. From Fig. 8 (a) and (b), we can see our approach converges faster and achieves better filtering performance than DJF [26] on both upsampling and denoising tasks.

**Comparison with SVLRM [27]**. Lastly, we compare to the state-of-the-art in deep guided filtering, namely SVLRM [27]. Here, we analyze two drawbacks of SVLRM [27], as illustrated in Fig. 9 (c-e). First,  $f_\alpha(I, G)$  and  $f_\beta(I, G)$  is likely to learn similar structure information. This is because they share the same training dependencies; such as input, network architecture and objective function. As a result,  $f_\alpha(I, G)$  can't



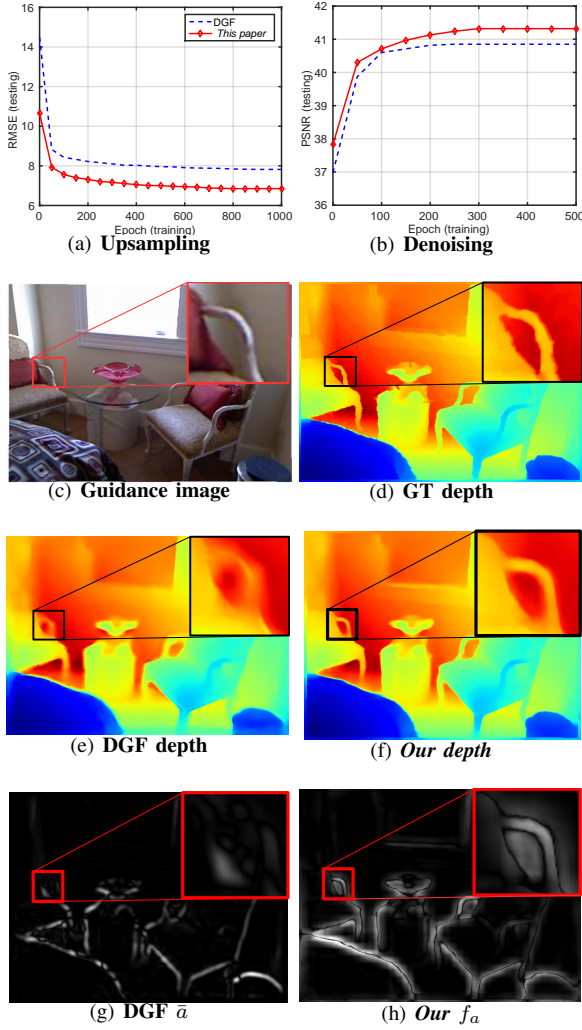


Fig. 7: **Comparison with DGF [38]**. The parameters of the guided filter [13] used in DGF are  $r = 4$ ,  $\epsilon = 0.1^2$ . Our learned amount function performs better on structure-transferring than the manually designed one as shown in the region marked by red boxes of (g) and (e). Thus, our filter reduces the over-smoothing of important edges as shown in the region marked by black boxes of (e) and (f).

transfer the desired structure from guidance  $G$  to the output image of  $f_\beta(I, G)$ . Second, SVLRM behaves like DJF [26] when the filtering performance is determined by  $f_\beta(I, G)$ . The implicit joint filtering causes slow convergence and the unwanted structures are transferred. By contrast, our approach focuses on estimating the amount function  $f_a$  for explicit structure transfer, leading to more desirable filtering results, as illustrated in Fig. 9 (h-j). Fig. 9 (a) and (b) demonstrate the better performance of our approach compared to SVLRM [27], for both upsampling and denoising.

**Amount function  $f_a$ .** When we estimate the amount function  $f_a$  through a convolutional neural network, the network architecture plays an important role in filtering performance. We have compared our filtering formulation with several baselines when  $f_a$  is estimated by different networks used in

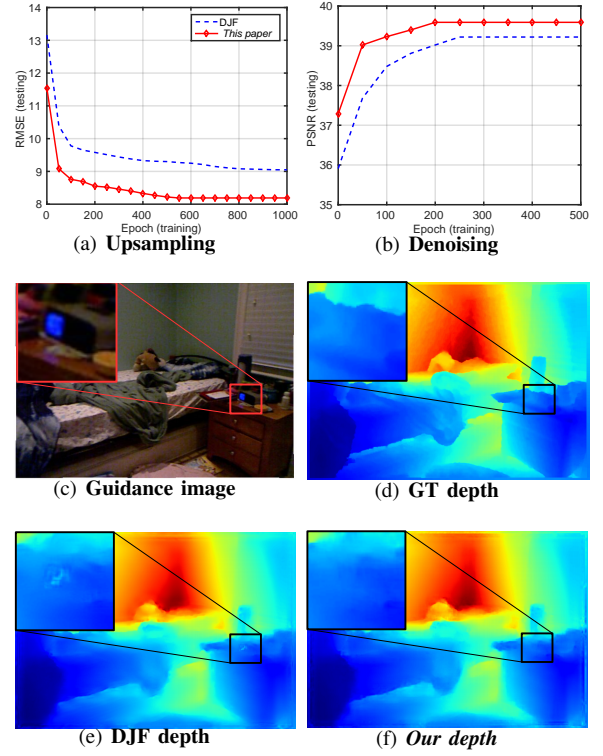


Fig. 8: **Comparison with DJF [26]**. Our approach avoids unwanted structures transferred from the guidance image to the target image as shown in the regions marked by the black boxes, leading to more desirable filtering results than DJF.

these baselines. Generally, we found deep networks perform better than shallow networks, *e.g.*, the network of SVLRM with a depth of 12 achieves an RMSE of 7.23 for upsampling ( $16\times$ ) and a PSNR of 40.27 for denoising ( $\sigma = 50$ ), better than the RMSE of 8.19 and the PSNR of 39.59 obtained by the network of DJF with a depth of 6. One explanation for this is the fact that the deep network has more ability to express complex functions than shallow ones.

In Eq.(15), we use the unsharp masks of the target image and guidance image as the input of the amount function network  $f_a$ . The raw target image and guidance image can also be the input. Next, we perform an experiment to study which input performs better. The network of DJF [26] is used for  $f_a$ . On NYU Depth V2, using the unsharp mask as input achieves an RMSE of 8.19 for upsampling ( $16\times$ ) and a PSNR of 39.59 for denoising ( $\sigma = 50$ ), better than the RMSE of 8.64 and the PSNR of 39.31 obtained by using the raw image as input. We find that using the unsharp mask as input not only achieves better filtering performance, but also converges faster because network learning can focus on extracting the desired structure without the interference of redundant signals from the smooth basis of the image.

**Smoothing filter  $\mathcal{F}_L$ .** To obtain the unsharp masks  $G_m$  and  $I_m$ , we need a smoothing filter for  $\mathcal{F}_L(I)$  and  $\mathcal{F}_L(G)$ . Next, we explore how the smoothing process affects the final filtering performance, we compare three different smoothing filters with different hyper-parameters on NYU Depth V2 for

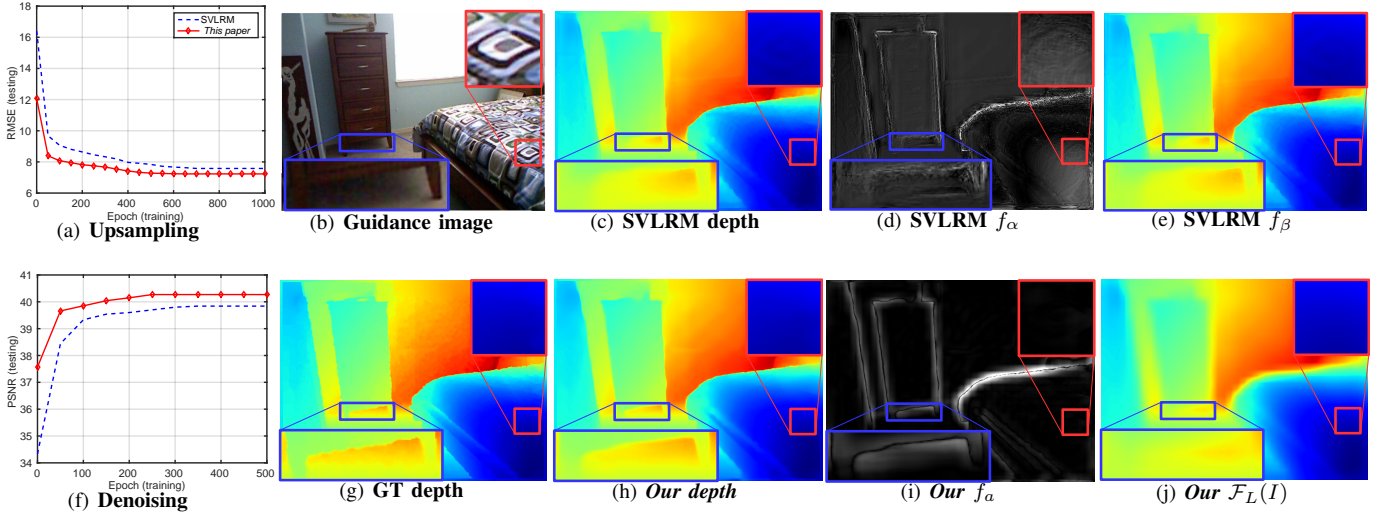


Fig. 9: **Comparison with SVLRM [27].** In SVLRM,  $f_\alpha$  and  $f_\beta$  are likely to learn the same structure information. In this case,  $f_\alpha$  can't transfer the structure desired by  $f_\beta$ , resulting from over-smoothing of edges as shown in the regions marked by the blue boxes of (d) and (e). When the filtering output is determined by  $f_\beta$ , SVLRM behaves like DJF [26], causing the transfer of unwanted contents from guidance image to target image as shown in the regions marked by the red boxes of (c-e). By contrast, our approach resolve the problems of SVLRM by explicitly learning structure-transferring, leading to more desirable filtering results as shown in regions marked by the blue and red boxes of (h-j).

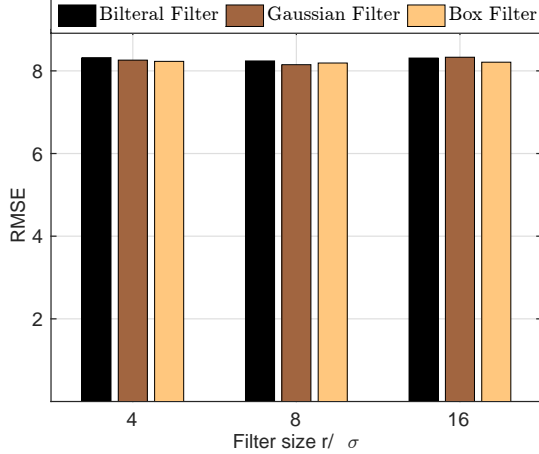


Fig. 10: **The effect of smoothing filter  $\mathcal{F}_L$  on NYU Depth V2 for depth upsampling ( $16\times$ ).** Our method is robust across smoothing filter type and size.

$16\times$  depth upsampling. The hyper-parameter is the filtering size  $r$  for the box filter, or the Gaussian variance  $\sigma$  for the Gaussian and bilateral filters. We use three different values: (4, 8, 16). The network of DJF [22] is used for  $f_\alpha$ . As shown in Fig. 10, our method is robust across filter type and size. We opt for the box filter with a filtering size of 8 throughout our experiments because it is simple, efficient and effective.

#### D. Successive filtering network

Next, we investigate the effect of the network we designed for our successive filtering formulation. There are multiple amount blocks used in the successive filtering network. We explore how the number of amount blocks  $L$  affects the

filtering performance on NYU Depth V2 for depth upsampling ( $16\times$ ) and depth denoising ( $\sigma = 50$ ).  $\mathcal{F}_L$  is a box mean filter with radius  $r = 8$ . For depth upsampling ( $16\times$ ), we first upsample the low-resolution depth image by bicubic interpolation to obtain the target resolution for  $\mathcal{F}_L(I)$  and network learning. We make two observations from the results shown in Table II. First, our model's filtering performance is consistently improved when increasing  $L$  from 1 to 5. Second, we can obtain multiple ( $L$ ) filtering results by training a single network. Each filtering result is as good as the result obtained by an independently trained network. The model's filtering performance doesn't improve a lot when  $L$  is increased from 4 to 5. Thus, we opt for  $L = 4$  in the following experiments.

#### E. Performance analysis.

Next, we analyze the performance of different deep guided filtering methods from three aspects: run-time performance, model parameters and filtering accuracy. For our methods, we use the successive filtering network with  $L = 4$ . Thus, we can obtain four filtering models by training a single network, indicated by Ours( $\hat{I}^{(1)}$ ), Ours( $\hat{I}^{(2)}$ ), Ours( $\hat{I}^{(3)}$ ) and Ours( $\hat{I}^{(4)}$ ). We perform upsampling ( $16\times$ ) with all methods on the testing datasets (499 RGB/depth pairs) of NYU Depth V2. We perform all the testings on the same machine with an Intel Xeon E5-2640 2.20GHz CPU and an Nvidia GTX 1080 Ti GPU. The average run-time performance on 499 images with the size of  $640 \times 480$  is reported in GPU mode with TensorFlow. From Table III, we can see that Ours( $\hat{I}^{(1)}$ ) has the fewest parameters (17 k), and Ours( $\hat{I}^{(4)}$ ) achieves the best filtering accuracy (6.07 RMSE). Ours( $\hat{I}^{(1)}$ ) achieves a competitive average run-time performance (31 ms) compared to the best one achieved by DJF [22] (29 ms).

TABLE II: **Ablation studies for our network** on NYU Depth V2 for depth upsampling ( $16\times$ , RMSE) and denoising ( $\sigma = 50$ , PSNR). our model's filtering performance is consistently improved when increasing  $L$  from 1 to 5.

|                 | L=1          |             | L=2          |             | L=3          |             | L=4          |             | L=5          |             |
|-----------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|
|                 | Upsampling ↓ | Denoising ↑ | Upsampling ↓ | Denoising ↑ | Upsampling ↓ | Denoising ↑ | Upsampling ↓ | Denoising ↑ | Upsampling ↓ | Denoising ↑ |
| $\hat{I}^{(1)}$ | 7.97         | 40.08       | 8.03         | 39.95       | 8.05         | 39.92       | 8.09         | 39.87       | 8.16         | 39.81       |
| $\hat{I}^{(2)}$ | n.a.         | n.a.        | 6.76         | 40.93       | 6.75         | 40.87       | 6.77         | 40.85       | 6.87         | 40.79       |
| $\hat{I}^{(3)}$ | n.a.         | n.a.        | n.a.         | n.a.        | 6.33         | 41.27       | 6.28         | 41.25       | 6.32         | 41.21       |
| $\hat{I}^{(4)}$ | n.a.         | n.a.        | n.a.         | n.a.        | n.a.         | n.a.        | 6.07         | 41.53       | 6.09         | 41.49       |
| $\hat{I}^{(5)}$ | n.a.         | n.a.        | n.a.         | n.a.        | n.a.         | n.a.        | n.a.         | n.a.        | 6.02         | 41.61       |

TABLE III: **Performance analysis.** on NYU Depth V2 for depth upsampling ( $16\times$ ). Our filtering models achieve competitive performance in terms of run-time, model parameters and filtering accuracy.

|                         | Run-time (ms) ↓ | Parameters (k) ↓ | Accuracy (RMSE) ↓ |
|-------------------------|-----------------|------------------|-------------------|
| DMSG <sup>†</sup>       | 36              | 534              | 8.21              |
| DJF <sup>†</sup>        | <b>29</b>       | 40               | 9.05              |
| DGF <sup>†</sup>        | 34              | 32               | 7.82              |
| SVLRM <sup>†</sup>      | 47              | 371              | 7.58              |
| Ours( $\hat{I}^{(1)}$ ) | 31              | <b>17</b>        | 8.09              |
| Ours( $\hat{I}^{(2)}$ ) | 45              | 38               | 6.77              |
| Ours( $\hat{I}^{(3)}$ ) | 58              | 59               | 6.28              |
| Ours( $\hat{I}^{(4)}$ ) | 66              | 85               | <b>6.07</b>       |

<sup>†</sup>Results from our reimplementation under the same settings as this work.

TABLE IV: **Depth upsampling** for  $2\times$ ,  $4\times$ ,  $8\times$  and  $16\times$  on NYU Depth V2. The depth values are measured in centimeter, and a boundary with 6 pixels is excluded for evaluation. We outperform alternative filters for almost all settings.

|                         | 2×          |               | 4×          |               | 8×          |               | 16×         |               |
|-------------------------|-------------|---------------|-------------|---------------|-------------|---------------|-------------|---------------|
|                         | RMSE ↓      | SSIM ↑        | RMSE ↓      | SSIM ↑        | RMSE ↓      | SSIM ↑        | RMSE ↓      | SSIM ↑        |
| DMSG [23]               | -           | -             | 3.78        | -             | 6.37        | -             | 11.16       | -             |
| DJF [26]                | -           | -             | 3.38        | -             | 5.86        | -             | 10.11       | -             |
| bFT [24]                | -           | -             | 3.35        | -             | 5.73        | -             | 9.01        | -             |
| PAC [25]                | -           | -             | 2.39        | -             | 4.59        | -             | 8.09        | -             |
| FWM [55]                | -           | -             | 2.16        | -             | 4.32        | -             | 7.66        | -             |
| SVLRM [27]              | -           | -             | <b>1.74</b> | -             | 5.59        | -             | 7.23        | -             |
| DMSG <sup>†</sup>       | 2.12        | 0.9957        | 3.43        | 0.9864        | 4.19        | 0.9814        | 8.21        | 0.9607        |
| DGF <sup>†</sup>        | 2.29        | 0.9940        | 3.18        | 0.9897        | 4.78        | 0.9776        | 7.82        | 0.9568        |
| DJF <sup>†</sup>        | 1.37        | 0.9972        | 2.85        | 0.9934        | 4.48        | 0.9801        | 9.05        | 0.9548        |
| SVLRM <sup>†</sup>      | 1.28        | 0.9975        | 2.62        | 0.9946        | 3.96        | 0.9835        | 7.58        | 0.9616        |
| Ours( $\hat{I}^{(1)}$ ) | 2.02        | 0.9963        | 2.90        | 0.9925        | 4.23        | 0.9839        | 8.09        | 0.9563        |
| Ours( $\hat{I}^{(2)}$ ) | 1.65        | 0.9971        | 2.61        | 0.9938        | 3.82        | 0.9851        | 6.77        | 0.9657        |
| Ours( $\hat{I}^{(3)}$ ) | 1.34        | 0.9974        | 2.40        | 0.9940        | 3.65        | 0.9857        | 6.28        | 0.9690        |
| Ours( $\hat{I}^{(4)}$ ) | <b>1.21</b> | <b>0.9976</b> | 2.33        | <b>0.9949</b> | <b>3.58</b> | <b>0.9863</b> | <b>6.07</b> | <b>0.9706</b> |

<sup>†</sup>Results from our reimplementation under the same settings as this work.

### F. Depth and flow upsampling

Tables IV and V show results for upsampling a depth image or optical flow image, under the guidance of its RGB image. We have noted that the existing works use different training settings and evaluation protocols. For fair comparison, we reimplement the main baseline methods under our experimental settings. Our filter performs well, especially on Sintel and the larger upsampling scales on NYU Depth. Different from the related works [23]–[27], [38], our model learns an amount function  $f_a$  to explicitly decide how to transfer the desired structure from guidance image to target image. Thus, our model can be more effective and efficient to learn the desired output. Fig. 11 show our ability to better recover finer edges.

TABLE V: **Flow upsampling** for  $2\times$ ,  $4\times$ ,  $8\times$  and  $16\times$  on Sintel. We outperform alternative filters for almost all settings.

|                         | 2×          |               | 4×          |               | 8×          |               | 16×         |               |
|-------------------------|-------------|---------------|-------------|---------------|-------------|---------------|-------------|---------------|
|                         | EPE ↓       | SSIM ↑        | EPE ↓       | SSIM ↑        | EPE ↓       | SSIM ↑        | EPE ↓       | SSIM ↑        |
| DJF [26]                | -           | -             | 0.18        | -             | 0.44        | -             | 1.04        | -             |
| PAC [25]                | -           | -             | 0.11        | -             | 0.26        | -             | 0.59        | -             |
| FWM [55]                | -           | -             | 0.09        | -             | 0.23        | -             | 0.55        | -             |
| DMSG <sup>†</sup>       | 0.14        | 0.9928        | 0.24        | 0.9895        | 0.41        | 0.9811        | 0.96        | 0.9560        |
| DGF <sup>†</sup>        | 0.11        | 0.9942        | 0.13        | 0.9934        | 0.31        | 0.9842        | 0.78        | 0.9692        |
| DJF <sup>†</sup>        | 0.10        | 0.9951        | 0.17        | 0.9927        | 0.43        | 0.9837        | 1.04        | 0.9547        |
| SVLRM <sup>†</sup>      | 0.09        | 0.9957        | 0.16        | 0.9921        | 0.36        | 0.9845        | 0.98        | 0.9567        |
| Ours( $\hat{I}^{(1)}$ ) | 0.06        | 0.9988        | 0.11        | 0.9936        | 0.36        | 0.9851        | 0.86        | 0.9684        |
| Ours( $\hat{I}^{(2)}$ ) | 0.05        | 0.9990        | 0.07        | 0.9942        | 0.29        | 0.9859        | 0.68        | 0.9734        |
| Ours( $\hat{I}^{(3)}$ ) | 0.03        | 0.9991        | 0.05        | 0.9943        | 0.18        | 0.9864        | 0.52        | 0.9754        |
| Ours( $\hat{I}^{(4)}$ ) | <b>0.03</b> | <b>0.9991</b> | <b>0.04</b> | <b>0.9947</b> | <b>0.16</b> | <b>0.9867</b> | <b>0.45</b> | <b>0.9773</b> |

<sup>†</sup>Results from our reimplementation under the same settings as this work.

TABLE VI: **Depth image denoising** on NYU Depth V2. Our filter achieves the best results for all settings.

|                         | $\sigma = 15$ |               | $\sigma = 25$ |               | $\sigma = 50$ |               |
|-------------------------|---------------|---------------|---------------|---------------|---------------|---------------|
|                         | PSNR ↑        | SSIM ↑        | PSNR ↑        | SSIM ↑        | PSNR ↑        | SSIM ↑        |
| DGF <sup>†</sup>        | 45.52         | 0.9650        | 43.96         | 0.9579        | 40.15         | 0.9422        |
| DJF <sup>†</sup>        | 46.06         | 0.9633        | 43.58         | 0.9462        | 39.24         | 0.8826        |
| SVLRM <sup>†</sup>      | 47.35         | 0.9722        | 44.38         | 0.9524        | 39.84         | 0.8891        |
| Ours( $\hat{I}^{(1)}$ ) | 46.02         | 0.9627        | 43.62         | 0.9474        | 39.87         | 0.9112        |
| Ours( $\hat{I}^{(2)}$ ) | 46.92         | 0.9704        | 44.53         | 0.9586        | 40.85         | 0.9310        |
| Ours( $\hat{I}^{(3)}$ ) | 47.30         | 0.9732        | 44.93         | 0.9635        | 41.25         | 0.9422        |
| Ours( $\hat{I}^{(4)}$ ) | <b>47.45</b>  | <b>0.9750</b> | <b>45.09</b>  | <b>0.9664</b> | <b>41.53</b>  | <b>0.9488</b> |

<sup>†</sup>Results from our reimplementation under the same settings as this work.

### G. Depth and natural image denoising

Our formulation also allows for standard filtering without a guidance image by simply making  $G$  identical to  $I$  in Eq. (17). Intuitively, such a setup defines a structure-preservation filter, while the guided variant defines a structure-transfer filter. Next, we evaluate the ability to remove Gaussian noise from depth and natural images. For depth image denoising, its RGB image is used as guidance. For natural image denoising, the target and guidance image are the same RGB image. The quantitative results are shown in Tables VI and VII. We obtain the best PSNR and SSIM scores on both datasets for all three noise levels. Fig. 12 and Fig. 13 highlight our ability to better remove noise and preserve finer edges compared to other filters.

### H. Cross-modality filtering

Finally, we demonstrate that our models trained on one modality can be directly applied to other modalities. Here, we use the models trained with RGB/depth image pairs for the joint upsampling of bw/color and RGB/saliency image pairs, and the joint denoising of RGB/NIR and flash/no-flash



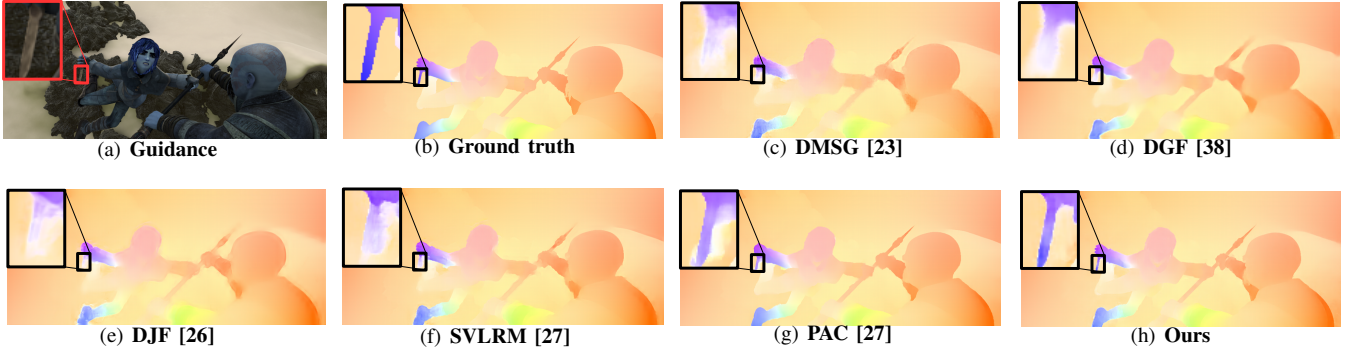


Fig. 11: **Optical flow upsampling** ( $16\times$ ) on Sintel. We are able to maintain sharp and thin edges.

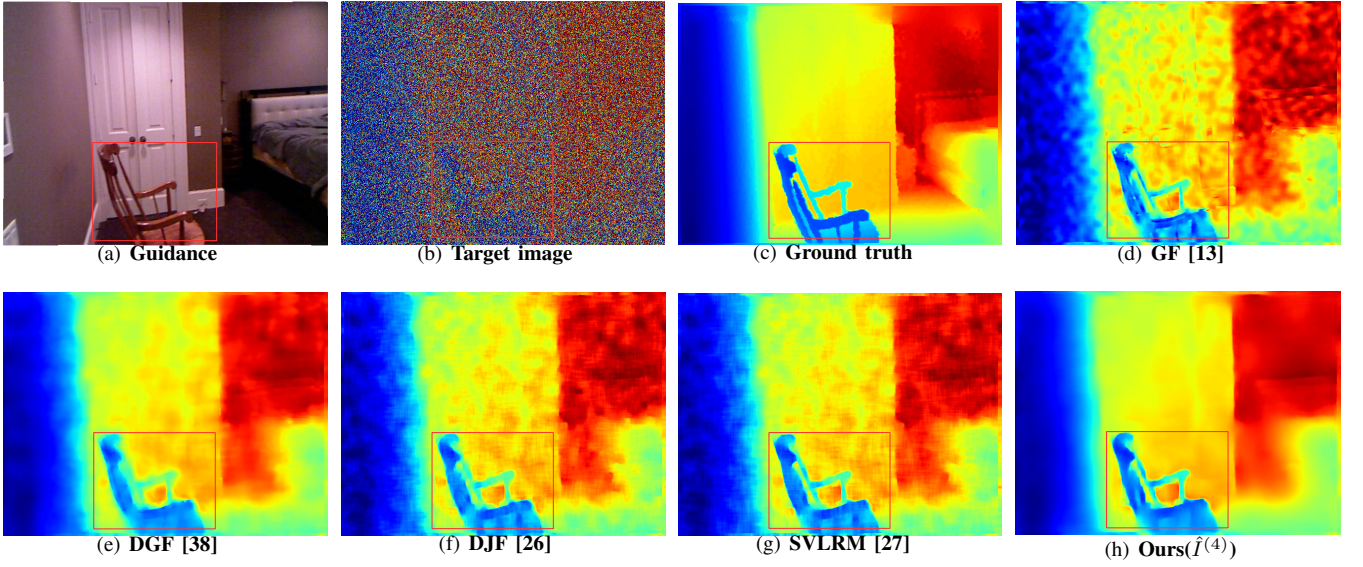


Fig. 12: **Depth denoising result** ( $\sigma = 50$ ) on NYU depth v2. Our filter better preserves edges and removes noise.

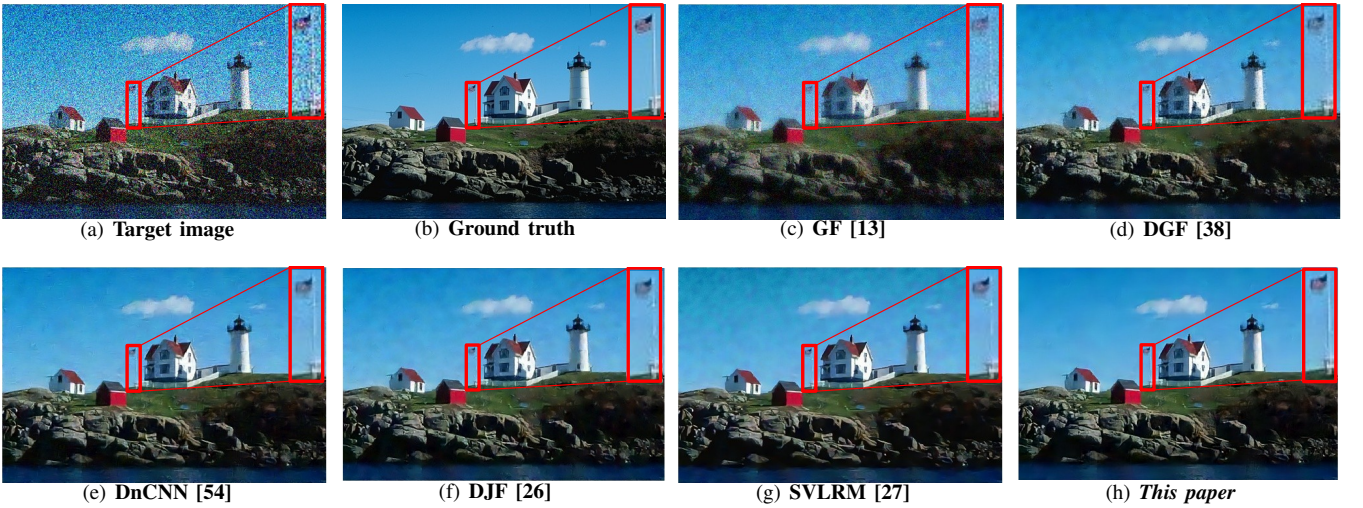


Fig. 13: **Denoising result** ( $\sigma = 50$ ) on BSDS500. Our filter better preserves finer edges with fewer noise artifacts.

image pairs. For our method, we use the model of  $Ours(\hat{I}^{(4)})$ . Following [26], for the multi-channel target image, *i.e.*, no-

flash image, we apply the trained models independently for each channel. For the single-channel guidance image, *i.e.*, NIR

TABLE VII: **Natural image denoising** on BSDS500. Our filter achieves the best results for all settings.

|                         | $\sigma = 15$   |                 | $\sigma = 25$   |                 | $\sigma = 50$   |                 |
|-------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
|                         | PSNR $\uparrow$ | SSIM $\uparrow$ | PSNR $\uparrow$ | SSIM $\uparrow$ | PSNR $\uparrow$ | SSIM $\uparrow$ |
| DnCNN <sup>†</sup>      | 33.12           | 0.9166          | 30.48           | 0.8618          | 27.10           | 0.7495          |
| DGF <sup>†</sup>        | 31.89           | 0.9010          | 29.56           | 0.8409          | 26.31           | 0.7229          |
| DJF <sup>†</sup>        | 33.10           | 0.9168          | 30.49           | 0.8646          | 27.09           | 0.7496          |
| SVLRM <sup>†</sup>      | 33.01           | 0.9202          | 30.60           | 0.8712          | 27.37           | 0.7686          |
| Ours( $\hat{I}^{(1)}$ ) | 33.41           | 0.9272          | 30.80           | 0.8758          | 27.43           | 0.7716          |
| Ours( $\hat{I}^{(2)}$ ) | 33.65           | 0.9341          | 31.05           | 0.8868          | 27.73           | 0.7829          |
| Ours( $\hat{I}^{(3)}$ ) | 33.76           | 0.9354          | 31.16           | 0.8890          | 27.87           | 0.7895          |
| Ours( $\hat{I}^{(4)}$ ) | <b>33.79</b>    | <b>0.9361</b>   | <b>31.19</b>    | <b>0.8906</b>   | <b>27.91</b>    | <b>0.7938</b>   |

<sup>†</sup>Results from our reimplementation under the same settings as this work.

TABLE VIII: **Cross-modality filtering** for joint upsampling ( $4\times$ ) on bw/color and RGB/saliency pairs. Our filter achieves the best results for all settings.

|                    | bw/color          |                 | RGB/saliency         |                 |
|--------------------|-------------------|-----------------|----------------------|-----------------|
|                    | RMSE $\downarrow$ | SSIM $\uparrow$ | F-measure $\uparrow$ | SSIM $\uparrow$ |
| GF <sup>†</sup>    | 11.51             | 0.6054          | 0.685                | 0.5365          |
| DGF <sup>†</sup>   | 11.17             | 0.6041          | 0.701                | 0.5431          |
| DJF <sup>†</sup>   | 10.96             | 0.6046          | 0.697                | 0.5378          |
| SVLRM <sup>†</sup> | 10.78             | 0.6074          | 0.699                | 0.4588          |
| <b>Ours</b>        | <b>10.39</b>      | <b>0.6095</b>   | <b>0.705</b>         | <b>0.6087</b>   |

<sup>†</sup>Results from our reimplementation under the same settings as this work.

TABLE IX: **Cross-modality filtering** for joint denoising ( $\sigma = 25$ ) on Flash/no-flash and RGB/NIR pairs. Our filter achieves the best results for all settings.

|                    | Flash/no-flash  |                 | RGB/NIR         |                 |
|--------------------|-----------------|-----------------|-----------------|-----------------|
|                    | PSNR $\uparrow$ | SSIM $\uparrow$ | PSNR $\uparrow$ | SSIM $\uparrow$ |
| GF <sup>†</sup>    | 29.43           | 0.7675          | 27.22           | 0.6971          |
| DGF <sup>†</sup>   | 28.11           | 0.7246          | 26.40           | 0.6436          |
| DJF <sup>†</sup>   | 29.53           | 0.7407          | 27.56           | 0.6826          |
| SVLRM <sup>†</sup> | 30.27           | 0.7584          | 28.33           | 0.7084          |
| <b>Ours</b>        | <b>30.76</b>    | <b>0.7699</b>   | <b>28.95</b>    | <b>0.7226</b>   |

<sup>†</sup>Results from our reimplementation under the same settings as this work.

image, we replicate it three times to obtain a 3-channel guidance image.

**Joint upsampling.** To speed up the translation from one image to another image, one strategy is to perform translation at a coarse resolution and then upsample the low-resolution solution back to the original one with a joint image upsampling filter. Here, we demonstrate that our models act as joint upsampling filters well on bw/color and RGB/saliency image pairs translation tasks. For bw/color translation, we use 68 bw images from BSD68 dataset [56], and the colorization model proposed by Lizuka *et al.* [57] is used as translation model. For RGB/saliency translation, we use 1000 RGB image from ECSSD dataset [58], and the saliency region detection model proposed by Hou *et al.* [59] as translation model. The input images, *i.e.*, bw images and RGB images, are first downsampled by a factor of  $4\times$  using nearest-neighbor interpolation, and then are feed into the translation models to generate the output images. After that, we recover the output

images to the original resolution under the guidance of the original input images by various joint upsampling methods. Table VIII shows the quantitative results, and we can see that our model achieves the best performance for both two tasks. The joint upsampling pipeline performs more than two times faster than direct translation on the GPU mode. We also provide the qualitative results in Fig. 14 and 15 to show that the proposed model better recovers finer details.

**Joint denoising.** We introduce two datasets for the joint denoising experiments. **Flash/no-flash** [60] consists of 120 image pairs, where the no-flash image is used for denoising under the guidance of its flash version [13], [26], [61], [62]. **Nirscene1** [63] consists of 477 RGB/NIR image pairs in 9 categories, where the RGB image is used for denoising under the guidance of its NIR version [26], [62], [64]. Table IX shows that our filter has a better denoising ability than alternatives. Fig. 16 and Fig. 17 provide qualitative results, which shows that our model better preserves important structures while removing noises.

## VI. CONCLUSION

In this paper, we have introduced a new and simplified guided filter. With inspiration from unsharp masking, our proposed formulation only requires estimating a single coefficient, in contrast to the two entangled coefficients required in current approaches. Based on the proposed formulation, we introduce a successive guided filtering network. Our network allows for a trade-off between accuracy and efficiency by choosing different filtering results during inference. Experimentally, we find that the proposed filtering method better preserves sharp and thin edges, while avoiding unwanted structures transferred from guidance. We furthermore show that our approach is effective for various applications such as upsampling, denoising, and cross-modal filtering.

## REFERENCES

- [1] M. Elad and A. Feuer, "Superresolution restoration of an image sequence: adaptive filtering approach," *IEEE Transactions on Image Processing*, vol. 8, no. 3, pp. 387–395, 1999.
- [2] M. R. Banham and A. K. Katsaggelos, "Spatially adaptive wavelet-based multiscale image restoration," *IEEE Transactions on Image Processing*, vol. 5, no. 4, pp. 619–634, 1996.
- [3] S. P. Awate and R. T. Whitaker, "Unsupervised, information-theoretic, adaptive image filtering for image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 3, pp. 364–376, 2006.
- [4] W.-Y. Ma and B. S. Manjunath, "Edgeflow: a technique for boundary detection and image segmentation," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1375–1388, 2000.
- [5] Y. Kang, C. Roh, S.-B. Suh, and B. Song, "A lidar-based decision-making method for road boundary detection using multiple kalman filters," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4360–4368, 2012.
- [6] M. Jacob and M. Unser, "Design of steerable filters for feature detection using canny-like criteria," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1007–1019, 2004.
- [7] D. Dunn and W. E. Higgins, "Optimal gabor filters for texture segmentation," *IEEE Transactions on Image Processing*, vol. 4, no. 7, pp. 947–964, 1995.
- [8] T. P. Weldon, W. E. Higgins, and D. F. Dunn, "Efficient gabor filter design for texture segmentation," *Pattern Recognition*, vol. 29, no. 12, pp. 2005–2016, 1996.
- [9] T. Randen and J. H. Husoy, "Texture segmentation using filters with optimized energy separation," *IEEE Transactions on Image Processing*, vol. 8, no. 4, pp. 571–582, 1999.



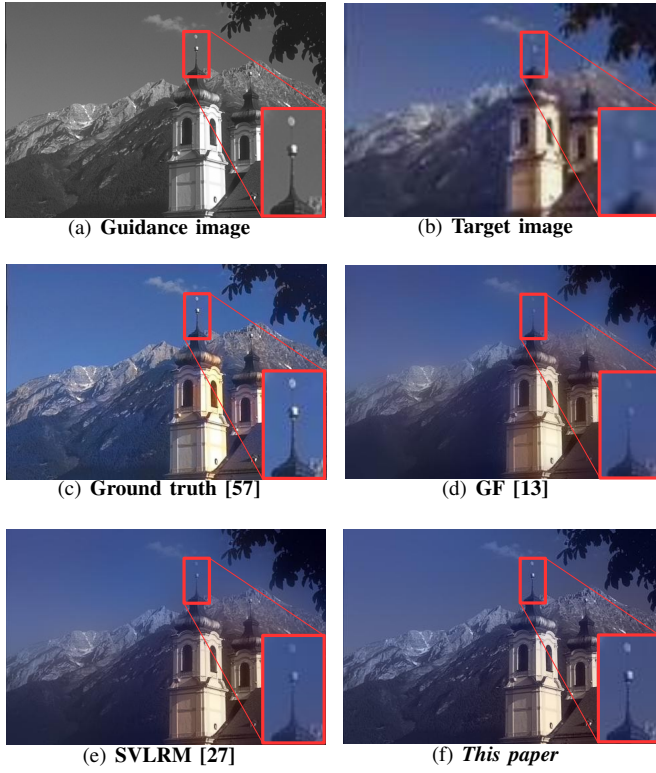


Fig. 14: **Joint upsampling result** ( $4\times$ ) on bw/color. Our filter recovers more desirable details.



Fig. 16: **Joint denoising result** ( $\sigma = 25$ ) on Flash/no-flash. Our filter better removes noises with less blurring.

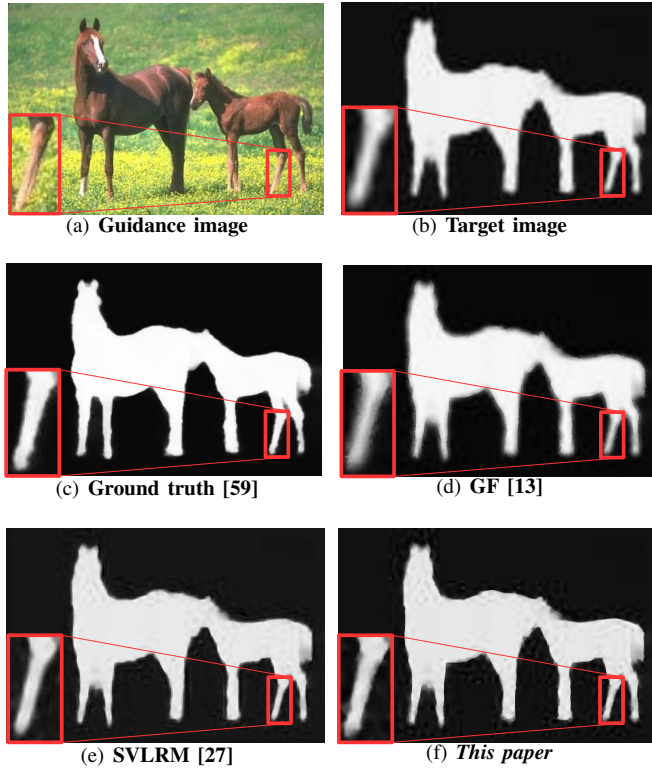


Fig. 15: **Joint upsampling result** ( $4\times$ ) on RGB/saliency. Our filter recovers sharper edges.



Fig. 17: **Joint denoising result** ( $\sigma = 25$ ) on RGB/NIR. Our filter better preserves finer details with fewer noises.

- [10] T. C. Aysal and K. E. Barner, "Quadratic weighted median filters for edge enhancement of noisy images," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3294–3310, 2006.
- [11] X. Guo, Y. Li, and H. Ling, "Lime: Low-light image enhancement via illumination map estimation," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 982–993, 2016.
- [12] D. Min, J. Lu, and M. N. Do, "Depth video enhancement based on weighted mode filtering," *IEEE Transactions on Image Processing*, vol. 21, no. 3, pp. 1176–1190, 2011.
- [13] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 6, pp. 1397–1409, 2012.
- [14] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint bilateral upsampling," *ACM Transactions on Graphics*, vol. 26, no. 3, p. 96, 2007.
- [15] C. C. Pham, S. V. U. Ha, and J. W. Jeon, "Adaptive guided image filtering for sharpness enhancement and noise reduction," in *PSIVT*, 2011.
- [16] M.-Y. Liu, O. Tuzel, and Y. Taguchi, "Joint geodesic upsampling of depth images," in *CVPR*, 2013, pp. 169–176.
- [17] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand, "Deep bilateral learning for real-time image enhancement," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–12, 2017.
- [18] J. Xie, R. S. Feris, and M.-T. Sun, "Edge-guided single depth image super resolution," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 428–438, 2015.
- [19] F. Kou, W. Chen, C. Wen, and Z. Li, "Gradient domain guided image filtering," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 4528–4539, 2015.
- [20] Z. Li, J. Zheng, Z. Zhu, W. Yao, and S. Wu, "Weighted guided image filtering," *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 120–129, 2014.
- [21] Z. Sun, B. Han, J. Li, J. Zhang, and X. Gao, "Weighted guided image filtering with steering kernel," *IEEE Transactions on Image Processing*, vol. 29, pp. 500–508, 2019.
- [22] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep joint image filtering," in *ECCV*, 2016.
- [23] T.-W. Hui, C. C. Loy, and X. Tang, "Depth map super-resolution by deep multi-scale guidance," in *ECCV*, 2016.
- [24] B. AlBahar and J.-B. Huang, "Guided image-to-image translation with bi-directional feature transformation," in *ICCV*, 2019.
- [25] H. Su, V. Jampani, D. Sun, O. Gallo, E. Learned-Miller, and J. Kautz, "Pixel-adaptive convolutional neural networks," in *CVPR*, 2019.
- [26] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Joint image filtering with deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1909–1923, 2019.
- [27] J. Pan, J. Dong, J. S. Ren, L. Lin, J. Tang, and M.-H. Yang, "Spatially variant linear representation models for joint filtering," in *CVPR*, 2019.
- [28] K. Morishita, S. Yamagata, T. Okabe, T. Yokoyama, and K. Hamatani, "Unsharp masking for image enhancement," Dec. 27 1988, US Patent 4,794,531.
- [29] A. Polesel, G. Ramponi, and V. J. Mathews, "Image enhancement via adaptive unsharp masking," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 505–510, 2000.
- [30] G. Deng, "A generalized unsharp masking algorithm," *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1249–1261, 2010.
- [31] W. Ye and K.-K. Ma, "Blurriness-guided unsharp masking," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4465–4477, 2018.
- [32] X. Shen, C. Zhou, L. Xu, and J. Jia, "Mutual-structure for joint filtering," in *ICCV*, 2015.
- [33] R. J. Jevnisek and S. Avidan, "Co-occurrence filter," in *CVPR*, 2017.
- [34] X. Guo, Y. Li, J. Ma, and H. Ling, "Mutually guided image filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
- [35] B. Ham, M. Cho, and J. Ponce, "Robust guided image filtering using nonconvex potentials," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 1, pp. 192–207, 2018.
- [36] H. Yin, Y. Gong, and G. Qiu, "Side window filtering," in *CVPR*, 2019.
- [37] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 1–10, 2008.
- [38] H. Wu, S. Zheng, J. Zhang, and K. Huang, "Fast end-to-end trainable guided filter," in *CVPR*, 2018.
- [39] X. Deng and P. L. Dragotti, "Deep convolutional neural network for multi-modal image restoration and fusion," *IEEE transactions on pattern analysis and machine intelligence*, 2020.
- [40] I. Marivani, E. Tsiliogianni, B. Cornelis, and N. Deligiannis, "Multimodal deep unfolding for guided image super-resolution," *IEEE Transactions on Image Processing*, vol. 29, pp. 8443–8456, 2020.
- [41] —, "Learned multimodal convolutional sparse coding for guided image super-resolution," in *ICIP*. IEEE, 2019, pp. 2891–2895.
- [42] —, "Joint image super-resolution via recurrent convolutional neural networks with coupled sparse priors," in *ICIP*. IEEE, 2020, pp. 868–872.
- [43] —, "Multimodal image super-resolution via deep unfolding with side information," in *EUSIPCO*. IEEE, 2019, pp. 1–5.
- [44] —, "Designing cnns for multimodal image super-resolution via the method of multipliers," in *EUSIPCO*. IEEE, 2021, pp. 780–783.
- [45] X. Deng and P. L. Dragotti, "Deep coupled ista network for multimodal image super-resolution," *IEEE Transactions on Image Processing*, vol. 29, pp. 1683–1698, 2019.
- [46] P. Song, X. Deng, J. F. Mota, N. Deligiannis, P. L. Dragotti, and M. R. Rodrigues, "Multimodal image super-resolution via joint sparse representations induced by coupled dictionaries," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 57–72, 2019.
- [47] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *ICLR*, 2015.
- [48] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.
- [49] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab, "Robust optimization for deep regression," in *ICCV*, 2015.
- [50] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR*, 2017, pp. 1125–1134.
- [51] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.
- [52] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *ECCV*, 2012.
- [53] 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 *ICCV*, 2001.
- [54] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.
- [55] X. Xu, Y. Ma, and W. Sun, "Learning factorized weight matrix for joint filtering," in *ICML*, 2020.
- [56] S. Roth and M. J. Black, "Fields of experts," *International Journal of Computer Vision*, vol. 82, no. 2, p. 205, 2009.
- [57] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 1–11, 2016.
- [58] J. Shi, Q. Yan, L. Xu, and J. Jia, "Hierarchical image saliency detection on extended cssd," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 4, pp. 717–729, 2015.
- [59] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. H. Torr, "Deeply supervised salient object detection with short connections," in *CVPR*, 2017.
- [60] S. He and R. W. Lau, "Saliency detection with flash and no-flash image pairs," in *ECCV*, 2014.
- [61] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, "Digital photography with flash and no-flash image pairs," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 664–672, 2004.
- [62] Q. Yan, X. Shen, L. Xu, S. Zhuo, X. Zhang, L. Shen, and J. Jia, "Cross-field joint image restoration via scale map," in *ICCV*, 2013.
- [63] M. Brown and S. Süssstrunk, "Multispectral SIFT for scene category recognition," in *CVPR*, 2011.
- [64] X. Wang, F. Dai, Y. Ma, J. Guo, Q. Zhao, and Y. Zhang, "Near-infrared image guided neural networks for color image denoising," in *ICASSP*, 2019.