

Fast End-to-End Trainable Guided Filter

Huikai Wu, *Student Member, IEEE*, Shuai Zheng, Junge Zhang, *Member, IEEE*,
and Kaiqi Huang, *Senior Member, IEEE*

Abstract—Dense pixel-wise image prediction has been advanced by harnessing the capabilities of Fully Convolutional Networks (FCNs). One central issue of FCNs is the limited capacity to handle joint upsampling. To address the problem, we present a novel building block for FCNs, namely guided filtering layer, which is designed for efficiently generating a high-resolution output given the corresponding low-resolution one and a high-resolution guidance map. Such a layer contains learnable parameters, which can be integrated with FCNs and jointly optimized through end-to-end training. To further take advantage of end-to-end training, we plug in a trainable transformation function for generating the task-specific guidance map. Based on the proposed layer, we present a general framework for pixel-wise image prediction, named deep guided filtering network (DGF). The proposed network is evaluated on five image processing tasks. Experiments on MIT-Adobe FiveK Dataset demonstrate that DGF runs 10-100 times faster and achieves the state-of-the-art performance. We also show that DGF helps to improve the performance of multiple computer vision tasks.

Index Terms—Joint Upsampling, Guided Filtering, Pixel-wise Image Prediction, Model Acceleration, Fully Convolutional Networks

I. INTRODUCTION

DENSE pixel-wise image prediction is a fundamental image processing and computer vision problem and has a wide range of applications. In image processing, dense pixel-wise image prediction enables smoothing an image while preserving the edges [7]–[9], enhancing the details of an image [3], [10], transferring the style from a reference image [11], [12], dehazing the photos [4], [13]–[15], and retouching the images for global tonal adjustment [2]. In computer vision, pixel-wise image prediction not only addresses the problem of segmenting an image into semantic parts [16]–[18], but also helps to estimate depth from a single image [19], and detect the most salient object in an image [20], [21].

H. Wu and J. Zhang are with Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: huikai.wu@nlpr.ia.ac.cn, jgzhang@nlpr.ia.ac.cn

S. Zheng is with eBay Research. The work was conducted while the author was at the University of Oxford. E-mail: shuzheng@ebay.com

K. Huang is with Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China, and the CAS Center for Excellence in Brain Science and Intelligence Technology, 100190. E-mail: kqhuang@nlpr.ia.ac.cn

A preliminary version of this work [1] appeared in IEEE Conference on Computer Vision and Pattern Recognition, 2018. The code is available at <https://github.com/wuhuikai/DeepGuidedFilter>.

This work is funded by the National Key Research and Development Program of China (Grant 2016YFB1001004 and Grant 2016YFB1001005), the National Natural Science Foundation of China (Grant 61673375, Grant 61721004 and Grant 61403383) and the Projects of Chinese Academy of Sciences (Grant QYZDB-SSW-JSC006 and Grant 173211KYS-B20160008). The authors would like to thank Patrick Pérez and Philip Torr for their helpful suggestions.



Fig. 1. **Example Results of Deep Guided Filtering Network.** The top row shows the input images, and the bottom row presents the corresponding outputs. From left to right: image retouching [2], multi-scale detail manipulation [3], non-local dehazing [4], saliency detection [5], and depth estimation [6]. Best viewed in color.

Recent methods [22]–[24] usually employ Fully Convolutional Networks (FCNs) for these applications, achieving state-of-the-art performance. However, FCNs usually have a huge computational complexity and memory usage on high-resolution input images, which limits the deployment of pixel-wise image prediction algorithms in real-world applications. To accelerate FCNs, we present a general framework by following a coarse-to-fine fashion, which firstly downsamples the input image, executes the algorithm at low resolution, and then upsamples the result back to the original resolution. The main challenge is restoring the low-resolution output to the original resolution with rich details and sharp edges.

This challenge can be formulated as joint upsampling, which aims at generating a high-resolution output given the corresponding low-resolution one and a high-resolution guidance map. However, existing building blocks of FCNs have limited capability to handle such a problem. To enhance the ability of FCNs for joint upsampling, we propose to reformulate the widely used guided filter [25] into a fully differentiable building block, which can be (1) jointly trained with FCNs, (2) adapted for different tasks by learnable parameters, and (3) directly supervised by high-resolution ground truth.

To this end, we propose a novel building block for FCNs named guided filtering layer. Concretely, the original guided filter is expressed as a computational graph consisting of dilated convolutions and pointwise convolutions with learnable parameters, which can adaptively evolve for different tasks. A trainable transformation function is introduced into the proposed layer, which can generate a task-specific guidance map. As a result, all the parameters of a guided filtering layer can be learned in a data-driven manner through end-to-

end training. Moreover, such a layer can be easily integrated with a pre-defined FCN without extra efforts. By equipping FCNs with guided filtering layer, we present a general framework for pixel-wise image prediction tasks named Deep Guided Filtering Network (DGF), which can largely reduce the computational complexity and memory usage. The proposed framework can be widely employed for many image processing and computer vision tasks, as shown in Fig 1. Experiments show that DGF achieves the state-of-the-art performance in quality, speed, and memory usage.

In summary, the main contribution of this paper is that (1) we develop an end-to-end trainable guided filtering layer with learnable parameters and a trainable guidance map, which enhances the ability of FCNs for joint upsampling; (2) by combining with FCNs, the proposed layer significantly improves the state-of-the-art results in multiple image processing tasks, and runs 10-100 \times faster than the alternatives; and (3) additional experiments show that our approach generalizes well to many computer vision tasks and achieves significant improvements over baseline methods.

An early version of this paper [1] appeared in IEEE Conference on Computer Vision and Pattern Recognition (2018), to which we have made substantial extensions. The improvements are shown below: (1) [1] formulate the original guided filter into a series of spatially varying linear transformation matrices without any learnable parameters. In this paper, we reformulate the original guided filter into a block of dilated convolutions and pointwise convolutions with learnable parameters. Such a formulation enables guided filtering layer to fit a specific task through end-to-end training. (2) Based on the improved guided filtering layer, we further boosted the performance of DGF in five image processing tasks. (3) We conduct a systematic ablation study on five image processing tasks to analysis the influence of each hyper-parameter in DGF. (4) We demonstrate the upper bound of the proposed layer's ability in joint upsampling through a comprehensive experiment. (5) Both the training code and testing code are released for reproducing the experimental results in this paper and supporting further research as well as other applications.

II. RELATED WORK

A. Joint Upsampling

The most related works to our method are along the direction of joint upsampling. Many algorithms have been developed to tackle this problem. Joint bilateral upsampling [26] applies a bilateral filter [27] to the high-resolution guidance map, resulting in a piecewise-smoothing high-resolution output. The underlying bilateral filter usually requires a large amount of computation resources. Thus, many methods [28]–[30] are presented to reduce the computation complexity. Build on joint bilateral upsampling, Barron *et al.* [31] present a new form of bilateral-space optimization that efficiently solves a regularized least-squares optimization problem to produce an output that is bilateral-smooth and close to the input. Gharbi *et al.* [32] first compute a description of the transformation from a highly compressed input to output. Then a high-fidelity approximation of the output can be constructed by applying

the recipe to the high-quality input. Similarly, bilateral guided upsampling [33] fits an image operator with a grid of local affine models on the low-resolution input/output pair firstly. The high-resolution output is then generated by applying the local affine model to the high-resolution input image. This method serves as a post-processing operation, while our approach can be jointly trained with the entire FCN. Deep bilateral learning [22] integrate bilateral filter with FCNs, which can be jointly learned through end-to-end training. However, this method requires producing affine coefficients before obtaining outputs, which lacks direct supervision from the ground truth. For computer vision tasks, the number of affine coefficients is usually very large, which becomes the bottleneck of performance and speed. Besides bilateral filter, guided filter [25] is also widely used in joint upsampling, which derived from a local linear model and computes the filtering output by considering the content of a guidance image. Compared to it, our method is formulated as a fully differentiable building block with learnable parameters, which can be jointly trained with FCNs and adaptively adjusted according to a specific task. Similarly, Yuan *et al.* [34] employ a locally-affine model to relate patches from low-resolution RAW images to high-resolution JPEG images.

The above methods are based on edge-preserving local filters. Differently, other methods [35]–[37] produce high-resolution outputs by optimizing manually designed objective functions involving all or many pixels. The objective functions typically consist of data terms and regularization terms like total variation (TV) [35], weighted least squares (WLS) [36], and scale map scheme [37]. Following these methods, Shen *et al.* [38] propose mutual-structure to reserve the structural information that is contained in both images. Similarly, Ham *et al.* [39] formulate the issue as a non-convex optimization problem, which is solved by the majorization-minimization algorithm. Compared to our method, the main drawbacks of these methods are (1) they rely on hand-designed objective functions, and (2) they are usually time-consuming.

B. Deep Learning based Image Filter

Recently, deep learning based methods are proposed in image processing tasks, which largely advanced the state-of-the-art performance. Such tasks include image denoising [40], image demosaicking [41], image deblurring [42], image matting [43], rain drop removal [44], image dehazing [45], and image colorization [46].

The above methods mainly focus on solving one specific image processing task. Differently, some other works [47]–[49] aim at approximating a general class of operators. Xu *et al.* [47] employ deep neural networks to approximate a variety of edge-preserving filters with a gradient-domain training procedure, while Liu *et al.* [48] combine a convolutional network and a set of recurrent networks to approximate various image filters.

Xu *et al.* [47] and Liu *et al.* [48] deploy neural networks to generate high-resolution output directly, accelerating the operation by dedicatedly designed network architectures. Similarly, Chen *et al.* [23] propose context aggregation networks to

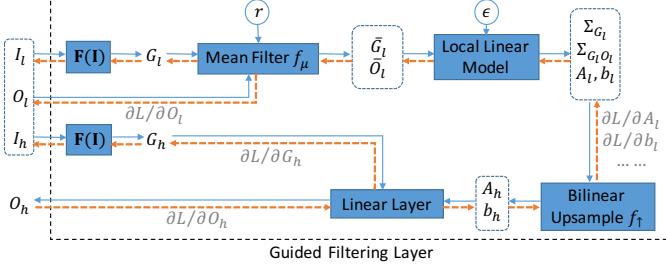


Fig. 2. Computation Graph of Guided Filtering Layer. Guided filtering layer takes low-resolution image I_l , high-resolution image I_h and low-resolution output O_l as inputs, generating the high-resolution output O_h . Compared to guided filter [25], the proposed layer is reformulated into a fully differentiable block and employs $F(I)$ to generate task-specific guidance map.

accelerate a wide variety of image processing operators, which performs superior to the prior works [33], [47], [48], [50], [51], achieving the best results regarding speed and accuracy. Our approach is complementary to this method, which can deliver comparable or better results and runs 10-100 \times faster.

Compared to all the related works, the proposed guided filtering layer can be end-to-end trained with the entire network and generalize well across different tasks ranging from image processing to computer vision, while achieving the state-of-the-art performance in both quality and speed.

III. GUIDED FILTERING LAYER

A. Problem Formulation

Given a high-resolution image I_h and the corresponding low-resolution output O_l , joint upsampling aims at generating a high-resolution output O_h that is visually similar to O_l and preserves the edges and details from I_h . In the literature of joint upsampling, guided filter [25] is one of the most widely used algorithms that has shown better performance regarding the trade-off between speed and accuracy.

B. Guided Filter Revisited

To address joint upsampling, guided filter [25] takes a low-resolution image I_l , the corresponding high-resolution image I_h , and a low-resolution output O_l as inputs, producing the high-resolution output O_h . Concretely, A_l and b_l are firstly obtained by minimizing a reconstruction error between \hat{O}_l and O_l , where \hat{O}_l subjects to a local linear model:

$$\hat{O}_l^i = A_l^k I_l^i + b_l^k, \forall i \in \omega_k. \quad (1)$$

ω_k is the k -th local square window on I_l , and I_l^i is the i -th pixel inside ω_k . A_h and b_h are then produced by upsampling A_l and b_l . The high-resolution output O_h is finally generated by a linear transformation model:

$$O_h = A_h * I_h + b_h, \quad (2)$$

where $*$ is element-wise multiplication.

Algorithm 1: Gradients for Guided Filtering Layer

Input : Low-resolution image I_l
 High-resolution image I_h
 Low-resolution output O_l
 Derivative for high-resolution output ∂O_h

Output: Gradients for all the inputs

- 1 $\partial b_l = \partial O_h \cdot \nabla_{b_l} f_\uparrow$
- 2 $\partial A_l = \partial O_h * G_h \cdot \nabla_{A_l} f_\uparrow - \partial b_l * \bar{G}_l$
- 3 $\partial \Sigma_{G_l O_l} = \partial A_l / (\Sigma_{G_l} + \epsilon)$
- 4 $\partial \bar{G}_l = \partial b_l - \partial \Sigma_{G_l O_l} * \bar{G}_l$
- 5 $\partial O_l = \partial \Sigma_{G_l O_l} \cdot \nabla_{G_l * O_l} f_\mu * G_l + \partial \bar{G}_l \cdot \nabla_{O_l} f_\mu$
- 6 $\partial G_l = -\partial b_l * A_l - \partial \Sigma_{G_l O_l} * \bar{O}_l - 2\partial \Sigma_{G_l} * \bar{G}_l$
 $+ 2\partial \Sigma_{G_l} \cdot \nabla_{G_l * G_l} f_\mu * G_l + \partial \bar{G}_l \cdot \nabla_{G_l} f_\mu$
- 7 $\partial I_l = \partial G_l \cdot \nabla_{I_l} F$
- 8 $\partial I_h = \partial O_h * A_h \cdot \nabla_{I_h} F$

C. Fully Differentiable Guided Filter

The original guided filter can only be employed as a post-processing operation, which is not differentiable and cannot be end-to-end trained with FCNs. To enhance the ability of FCNs for joint upsampling, we propose a novel building block by reformulating guided filter into a fully differentiable layer. Such a layer, named guided filtering layer, can be jointly trained with FCNs from scratch, and directly supervised by high-resolution targets.

The computation graph of guided filtering layer is shown in Figure 2. A_l and b_l are obtained by employing mean filter f_μ and local linear model to I_l and O_l , where f_μ is implemented as a box filter to reduce the computation complexity. A_h and b_h are then generated by bilinear upsampling f_\uparrow . O_h is finally produced by a linear layer taking A_h , b_h and I_h as inputs. r is the radius of f_μ and ϵ is the regularization term, which are set to be 1 and 1e-8 by default.

The equations for propagating the gradients through guided filtering layer are shown in Algorithm 1. By formulating each operator into a differentiable function, the gradient of O_h back-propagates to O_l , I_l , and I_h through the computation graph, which enables both the joint training of FCNs and guided filtering layer with direct guidance from the high-resolution targets. As a result, FCNs can learn to generate a more suitable O_l for guided filtering layer to restore O_h .

D. Learn to Generate Task-Specific Guidance Map

In Section III-C, I_h , I_l and O_h , O_l are assumed to have the same number of channels. When the channel sizes are different, a transformation function is required to transform I_h and I_l into a guidance map with the same number of channels as O_h and O_l . Even when the channel sizes are the same, a guidance map better than I_h and I_l is necessary for higher performance. Existing methods usually manually design the transformation function for different tasks, requiring lots of efforts and attempts. On the contrary, since the proposed

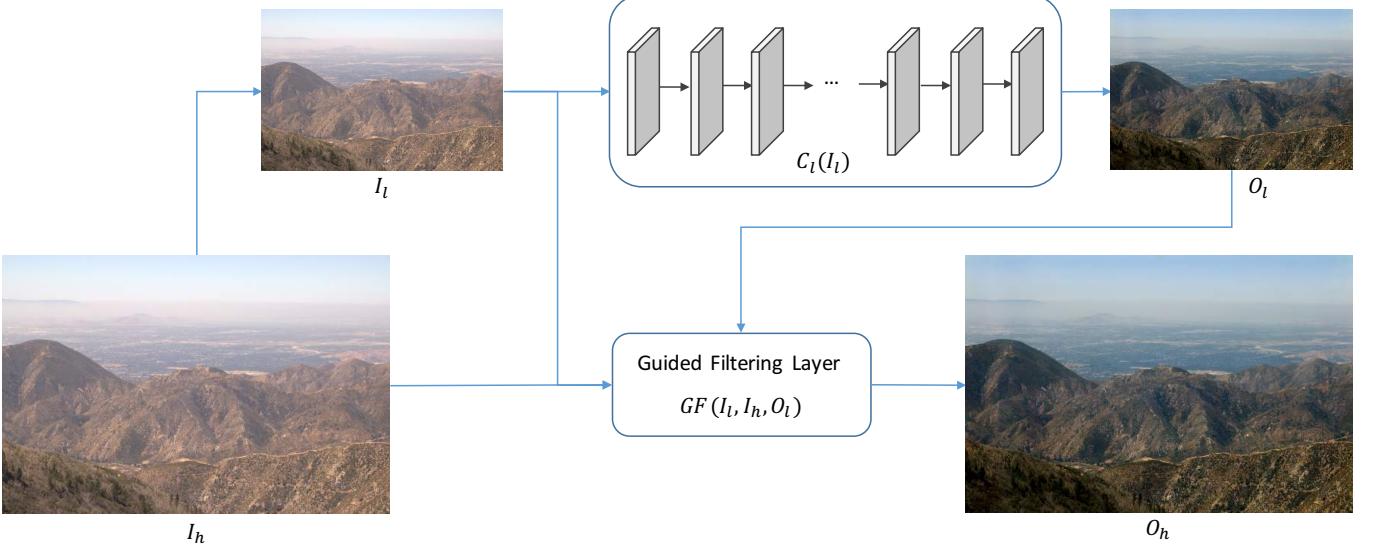


Fig. 3. **Framework Overview of Deep Guided Filtering Network.** Given the input image I_h , we first downsample it to obtain I_l . The corresponding low-resolution output O_l is then generated by the FCN $C_l(I_l)$. Finally, I_l , I_h and O_l are fed into guided filtering layer $GF(I_l, I_h, O_l)$ to generate the high-resolution output O_h .

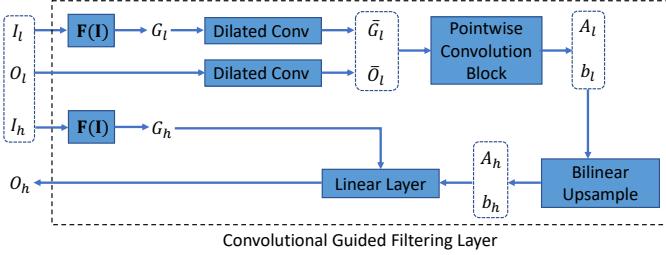


Fig. 4. **Computation Graph of Convolutional Guided Filtering Layer.** Dilated convolutions and a pointwise convolution block are introduced to replace mean filter and local linear model. With learnable parameters, such a layer can adaptively fit a specific task through end-to-end training.

guided filtering layer is fully differentiable, we can automatically learn a transformation function to generate more suitable, task-specific guidance maps by end-to-end training.

As shown in Figure 2, the transformation function $F(I)$ transforms I_h and I_l into task-specific guidance maps G_h and G_l . $F(I)$ is a FCN block composed of two convolution layers, between which are an adaptive normalization layer [23] and a leaky ReLU layer. The kernel size of both convolution layers is set to be 1×1 , and the channel size of the first convolution layer is set to be 16 by default.

E. Convolutional Guided Filtering Layer

Except $F(I)$, the proposed guided filtering layer is a parameter-free block, which behaves in the same manner for all different tasks. However, due to the huge differences between tasks, a single guided filtering layer without learnable parameters cannot perform well in all kinds of scenarios. To solve the problem, we introduce learnable parameters into guided filtering layer by replacing the non-parametric operations into convolution layers. As a result, the improved layer, convolutional guided filtering layer, becomes more powerful

for processing various applications, which can adaptively fit a specific task through end-to-end training.

The architecture of convolutional guided filtering layer is shown in Figure 4. Compared to that in Figure 2, dilated convolutions are introduced to replace mean filter f_μ , and a convolution block composed of pointwise convolutions takes the place of a local linear model. As for the hyper-parameters in Section III-C, ϵ is removed, and r represents the dilation rates in the dilated convolutions.

IV. DEEP GUIDED FILTERING NETWORK

Based on the proposed guided filtering layer, we present a general framework for pixel-wise image prediction tasks, named Deep Guided Filtering Network (DGF). By integrating the proposed layer with FCNs following a coarse-to-fine manner, DGF can generate high-resolution, edge-preserving outputs with a much lower computational cost and memory usage.

The architecture of DGF is shown in Figure 3. First, we downsample the original input image I_h to obtain the low-resolution input I_l . Then, a FCN $C_l(I_l)$ is applied to I_l , generating the corresponding low-resolution output O_l . Finally, the high-resolution output O_h is generated by guided filtering layer, taking I_l , I_h and O_l as inputs. The entire network is end-to-end trainable, which could be learned from scratch.

A. Fully Convolutional Network $C_l(I_l)$

DGF is a general framework for pixel-wise image prediction tasks, which can remarkably reduce the computational complexity and memory usage of the underlying algorithms. Concretely, given a specific pixel-wise image prediction task, an FCN $C(I)$ can be designed to achieve excellent performance without considering the speed and memory cost. In order to obtain significant optimization on speed and memory,

we can simply drop $C(I)$ into the proposed framework DGF to serve as $C_l(I_l)$ without any other modifications. Since $C(I)$ processes the input images in low resolution rather than the original resolution, the speed, and memory usage can be largely improved. Moreover, the performance of our system is also comparable to the previous state-of-the-art one, thanks to the proposed guided filtering layer. This is due to that the proposed guided filtering layer significantly enhances the capabilities of FCNs in the task of joint upsampling.

B. Guided Filtering Layer

In this paper, there are four variants of DGF in total according to different configurations of the guided filtering layer.

- 1) DGF_s : The original guided filter [25] is employed as post-processing operation without any training. $C_l(I_l)$ is trained with low-resolution input/output pairs before inserted into DGF_s .
- 2) DGF_b : Guided filtering layer in Figure 2 is employed in DGF_b . $F(I)$ is an identity function when the inputs and outputs have the same number of channels. When the channel sizes are different, $F(I)$ transforms the inputs into a grey image by averaging along the channel axis. $C_l(I_l)$ and guided filtering layer are jointly trained from scratch under the supervision directly from the high-resolution targets.
- 3) DGF_b^c : Compared to DGF_b , the guided filtering layer is replaced by convolutional guided filtering layer in Figure 4.
- 4) DGF^c : Compared to DGF_b^c , $F(I)$ proposed in Section III-D is introduced, which can learn to generate task-oriented guidance maps without manually design. As a result, DGF^c is not only end-to-end trainable but can also fit different tasks better by adjusting the trainable convolution weights and the learnable $F(I)$.

C. Objective Function

DGF is trained end-to-end under the supervision directly from the high-resolution targets. Concretely, given the high-resolution output O_h and the corresponding target T_h , the objective function is defined as $L(O_h, T_h)$. The concrete formulation varies with different tasks. Usually, the objective function for training $C(I)$ can be directly employed to train DGF without any adjustment.

V. EXPERIMENTS: IMAGE PROCESSING TASKS

To show the effectiveness of our method, we employ DGF to clone five widely-used image processing operators. Concretely, the ground truth images are first generated by applying L_0 smoothing operator [7], detail manipulation operator [3], style transfer operator [11], non-local dehazing operator [4], and image retouching operator [2] to the input images. Then, the input/ground-truth pairs are used to train DGF in a supervised way to clone the corresponding image processing operator.

TABLE I
THE ARCHITECTURE OF $C_l(I_l)$ AND $F(I)$ FOR CLONING IMAGE PROCESSING OPERATORS. THE NEGATIVE SLOPE OF LEAKY RELU IS SET TO 0.2.

Layer	$C_l(I_l)$								$F(I)$	
	1	2	3	4	5	6	7	8	1	2
Kernel	3	3	3	3	3	3	3	1	3	1
#Channels	24	24	24	24	24	24	24	3	16	3
Dilation	1	1	2	4	8	16	1	1	1	1
Bias	\times	\checkmark	\times	\checkmark						
AdaptNorm	\checkmark	\times	\checkmark	\times						
Leaky ReLU	\checkmark	\times	\checkmark	\times						

A. Details of Five Image Processing Operators

1) L_0 Smoothing: L_0 smoothing [7] is effective for sharpening major edges while eliminating minor edges by the use of L_0 gradient minimization. To generate the ground truth images, we use the official implementation with the default parameters¹.

2) Detail Manipulation: Multi-scale detail manipulation [3] enhances an image by boosting features at multiple scales. Concretely, a three-level decomposition (coarse base level b and two detail levels d^1, d^2) of the **CIELAB** lightness channel is first constructed given the input image. The resulting image is then obtained by a non-linear combination of b, d^1 and d^2 . To generate the ground truth images, we first generate coarse-scale, medium-scale, and fine-scale images with the official implementation and the default parameters². The final output is then yielded by averaging the three images.

3) Style Transfer: Photographic style transfer [11] aims at transferring the photographic style of a reference image to the input image. To generate the ground truth images, we employ the official implementation with the default setting and the default reference image³. The generated outputs are grey images, which are transformed into **RGB** images as the ground truth.

4) Non-local Dehazing: Non-local dehazing [4] employs a non-local prior to remove the effects of atmospheric absorption and scattering in the input image. We use the official implementation with default parameters to generate ground truth images⁴.

5) Image Retouching: Image retouching aims at automatically improving the aesthetic quality of the input image by global tonal adjustment. Human experts are employed to generate the ground truth.

B. Details of DGF

We employ Context Aggregation Network (CAN) [23] as $C_l(I_l)$ for all the five image processing operators. The detailed architectures of $C_l(I_l)$ and $F(I)$ are shown in Table I. Adapt-Norm represents adaptive normalization proposed by Chen *et al.* [23]. Leaky ReLU is employed as the nonlinearity, of

¹<http://www.cse.cuhk.edu.hk/~leojia/projects/L0smoothing>

²<http://www.cs.huji.ac.il/~danix/epd>

³http://www.di.ens.fr/~aubry/code/matlab_fast_llf_and_style_transfer.zip

⁴<https://github.com/danberman/non-local-dehazing>

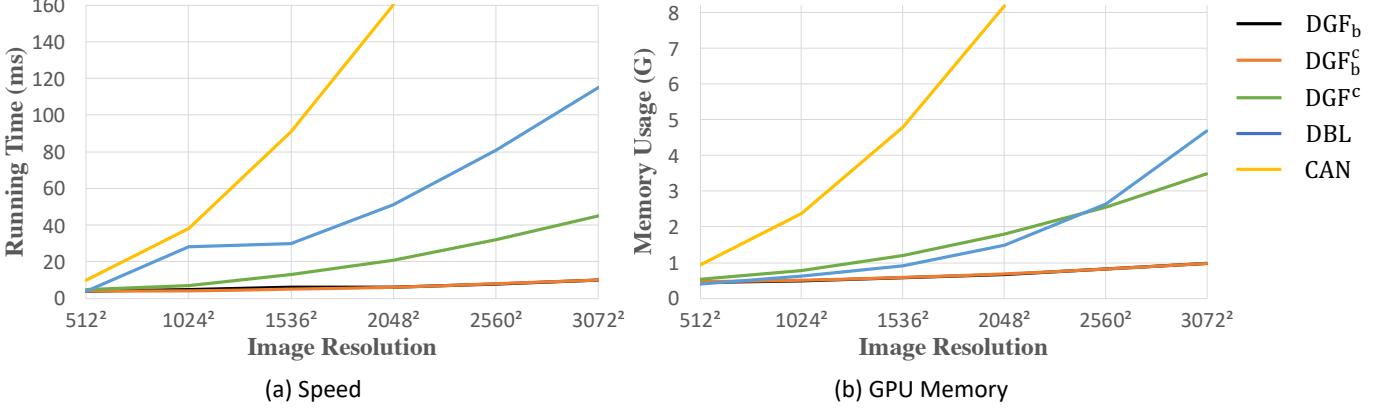


Fig. 5. Speed and Memory Usage Comparison on GPU Devices.

which the negative slope is set to 0.2. As for the objective function, we use L_2 loss by following the convention of previous works [22], [23].

C. Experimental Setup

Our experiments are taken on MIT-Adobe FiveK Dataset [2], which contains 2,500/2,500 high-resolution photographs as the training/testing images. In the dataset, each photograph contains five annotations from five experts, which can be used as the ground truth for image retouching. Instead of all five annotations, we employ the annotation from expert A as the ground truth. As for the other four image processing operators, the ground truth images are generated following the instructions in Section V-A.

As for training, we first train the network for 150 epochs, with the input/target images resized to 512s⁵. To improve the generalization ability, we further train the network for 30 epochs, with training data randomly resized to a specific resolution between 512s and 1672s. As for I_l , the spatial resolution is 64s regardless of the resolution of I_h . Adam is employed as the optimizer, with learning rate set to 0.0001 and batch size set to 1.

Our primary baseline is Deep Bilateral Learning (DBL) [22], which shares a similar architecture to ours and achieves a good trade-off between quality and speed. Another strong baseline is CAN [23], which achieves state-of-the-art performance while runs reasonably fast. To ensure a fair comparison, we train the models using the official implementations and training procedures for both methods.

D. Experimental Results

1) *Running Time and Memory Usage:* The running time and memory usage are shown in Figure 5, which are measured on a workstation with Intel E5-2650 2.20GHz CPU and Nvidia Titan X (Pascal) GPU.

On GPU devices, both DGF_b and DGF_c take less than 10ms to process an image with resolution ranging from 512² to

⁵xs means the short side of an image is resized to x without changing the aspect ratio.

3072². DGF_c is slightly slower because of the usage of $F(I)$, but it still runs in real-time on images with resolution 3072². All the three variants of our method run much faster than CAN and DBL among all resolutions. Specifically, DGF_b, DGF_b, and DGF_c take 6ms, 6ms, and 21ms respectively for an image in 2048². CAN takes 160ms for an image in 2048², which are more than 25×, 25×, and 7× slower than our method. DBL takes 51ms in the same setting, which is slightly faster than CAN but more than 8× slower than DGF_b and DGF_c. The advantage of our method in speed is even more significant as the resolution grows.

For I_h with $h \times w \times n_I$ and O_h with $h \times w \times n_O$, the theoretical computational complexities of DGF_b, DGF_c, DGF_b, and DBL are $\mathcal{O}(n_O \times h \times w)$, $\mathcal{O}(n_O \times h \times w)$, $\mathcal{O}((n_I + n_O) \times h \times w)$ and $\mathcal{O}(n_I \times n_O \times h \times w)$ respectively.

As for memory usage, our method takes less GPU memory space than both baseline methods. CAN is the most memory inefficient method that takes nearly 10G GPU memory to process an image with resolution 2048². DGF_c takes a similar amount of memory space to that of DBL but grows slower as the resolution increases. DGF_b and DGF_b are the most memory efficient methods, which take less than 1G memory even on images with resolution 3072².

2) *Quantitative and Qualitative Comparison:* The performance of each method is evaluated on the test set of MIT-Adobe FiveK dataset with input/target images resized to 1024s. MSE, PSNR, and SSIM serve as the evaluation metrics.

As shown in Table II, our method achieves the state-of-the-art performance in style transfer, non-local dehazing, and image retouching, while obtaining comparable results in L_0 smoothing and multi-scale detail manipulation. Concretely, DGF_c achieves 26.17 dB in PSNR for style transfer, which improves over CAN and DBL by 4.86 dB and 2.85 dB respectively. Compared to DBL, our method outperforms it across all the five tasks in all three metrics by a large margin.

The qualitative results are shown in Figure 8⁶.

3) *The Role of Guided Filtering Layer:* To show the effect of convolutional guided filtering layer and $F(I)$, we replace O_l

⁶More qualitative results are shown in <http://wuhuikai.me/DeepGuidedFilterProject/#visual>.

TABLE II
QUANTITATIVE COMPARISON ON IMAGE PROCESSING TASKS. THE 1ST, 2ND AND 3RD METHODS ARE HIGHLIGHTED AS RED, GREEN, AND BLUE.

Method	L_0 Smoothing [7]			Detail Manipulation [3]			Style Transfer [11]			Non-local Dehazing [4]			Image Retouching [2]		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
Input	73	29.61	0.796	443	22.12	0.789	3534	13.28	0.521	2081	16.95	0.684	1507	18.44	0.727
CAN [23]	27	35.05	0.970	9	38.97	0.986	519	21.31	0.870	355	24.47	0.862	964	20.43	0.744
DBL [22]	39	32.35	0.896	75	29.84	0.924	354	23.32	0.834	502	23.27	0.852	1056	20.21	0.748
DJF [52]	90	29.40	0.937	100	28.99	0.927	383	22.73	0.856	649	21.04	0.724	1216	18.89	0.702
DGF _s	35	32.93	0.912	92	29.12	0.905	333	23.22	0.735	351	24.53	0.871	872	20.81	0.757
DGF _b	33	33.20	0.911	77	29.95	0.905	318	23.42	0.738	323	25.53	0.892	875	20.94	0.762
DGF ^c _b	32	33.39	0.917	69	30.48	0.916	305	23.61	0.751	293	25.76	0.896	855	21.04	0.760
DGF ^c	30	33.69	0.923	48	32.10	0.940	181	26.17	0.880	296	25.85	0.902	855	21.09	0.767

TABLE III
UPPER BOUND FOR OUR METHOD ON IMAGE PROCESSING TASKS.

Method	L_0 Smoothing [7]			Detail Manipulation [3]			Style Transfer [11]			Non-local Dehazing [4]			Image Retouching [2]		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
DGF _b	27	34.52	0.923	93	29.23	0.904	261	24.28	0.752	49	33.00	0.956	100	30.82	0.859
DGF ^c _b	25	34.93	0.925	71	30.35	0.917	245	24.55	0.761	34	34.35	0.964	94	30.97	0.857
DGF ^c	23	35.27	0.930	42	32.66	0.947	109	28.34	0.899	28	35.19	0.968	90	31.19	0.860

with low-resolution ground truth to generate O_h . The obtained result represents the performance upper bound of each DGF variant. As shown in Table III, by reformulating guided filtering layer into learnable convolution layers, DGF^c_b outperforms DGF_b in all five tasks. By further introducing $F(I)$ into the convolutional guided filtering layer, DGF^c achieves the best performance.

Similar results can be observed in Table II. By jointly end-to-end training, DGF_b achieves better performance on most tasks than DGF_s. Concretely, DGF_b improves 1 dB and 0.83 dB (PSNR) for non-local dehazing and detail manipulation. By reformulating into convolutional guided filtering layer, the performance is further improved by comparing DGF_b and DGF^c_b. By adding learnable $F(I)$, we gain significant improvements in several tasks, especially in tasks that are resolution-dependent. Table II shows that DGF^c increases PSNR by 2.56 dB and 1.62 dB compared to DGF^c_b for style transfer and detail manipulation.

DJF [52] is the state-of-the-art method for joint upsampling. To verify the effectiveness of our method, we replace guided filtering layer in DGF with DJF. Results in Table II show that our method outperforms DJF in all tasks. Besides, our method also runs much faster than DJF, which takes 9× less time than DJF on images with resolution 1024² (5ms v.s. 46ms).

4) *Cross Resolution Generalization*: In the main experiment, our method is evaluated on 1024s images. To show the generalization ability of DGF for processing images in different resolutions, the pre-trained DGF is directly employed on images in 512s, 1024s, 1536s, and 2048s without finetuning. As shown in Figure 6, our method performs equally well across different resolutions on all tasks except style transfer. The reason is that style transfer is highly resolution-dependent. Concretely, given a reference image with a fixed resolution, the styles of the outputs are different for input images with different resolutions.

5) *Ablation Study*: A series of experiments are taken in this section to validate the effect of each hyper-parameter in the proposed guided filtering layer.

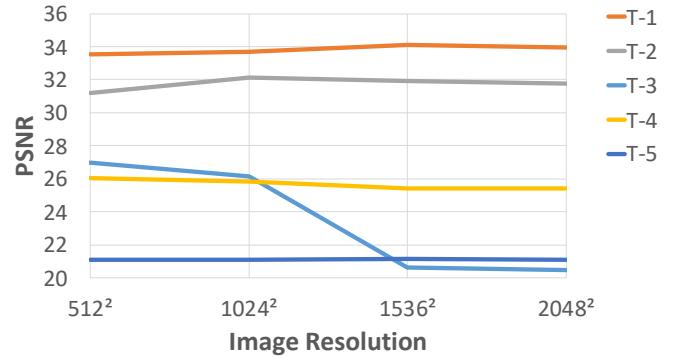


Fig. 6. CROSS RESOLUTION GENERALIZATION. T- x represents the x -th image processing task in Table II.

TABLE IV
SPEED AND MEMORY USAGE IN DIFFERENT RESOLUTIONS OF I_l .

Resolution	32	64	96	128	256	512
Running Time (ms)	6	6	6	7	7	19
Memory Usage (G)	0.67	0.68	0.70	0.73	0.87	1.41

The role of radius r is shown in Figure 7a. The performance drops quickly as r grows, and the default setting ($r = 1$) obtains the best PSNR score.

The effect of the resolution of I_l is shown in Figure 7b. For L_0 smoothing, multi-scale detail manipulation, and non-local dehazing, the performance grows as the resolution of I_l increases. For style transfer and image retouching, higher resolution is not always better. The corresponding running time and memory usage are shown in Table IV. When the resolution of I_l is 128 or 256, our method can not only achieve an excellent performance but also run very fast.

The function of $F(I)$ is also explored by varying the dilation rate. Figure 7c shows that increasing the dilation rate can improve the performance to a degree.

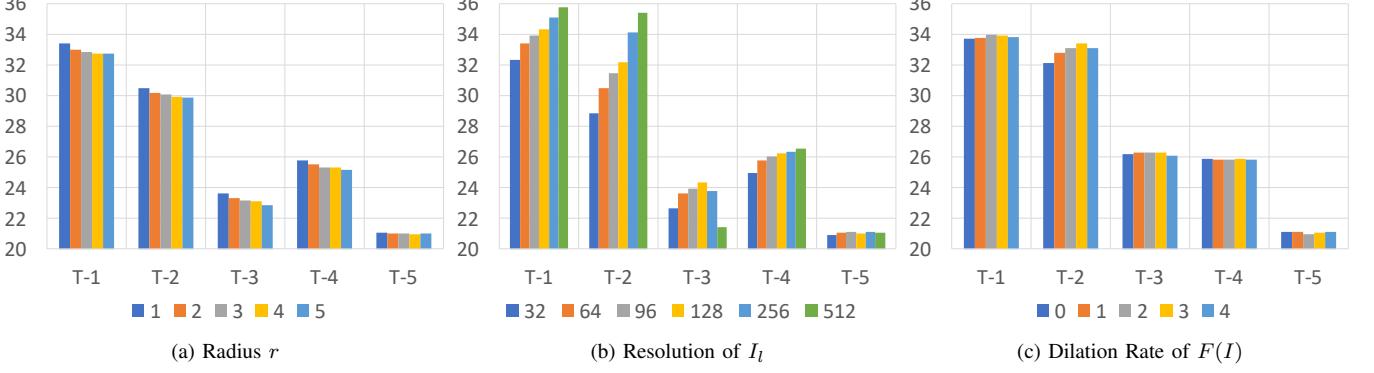


Fig. 7. **Ablation Study.** The performance (PSNR) with different (a) radius r of guided filtering layer, (b) resolution of I_l , and (c) dilation rate of $F(I)$. T- x represents the x -th image processing task in Table II. Best viewed in color.

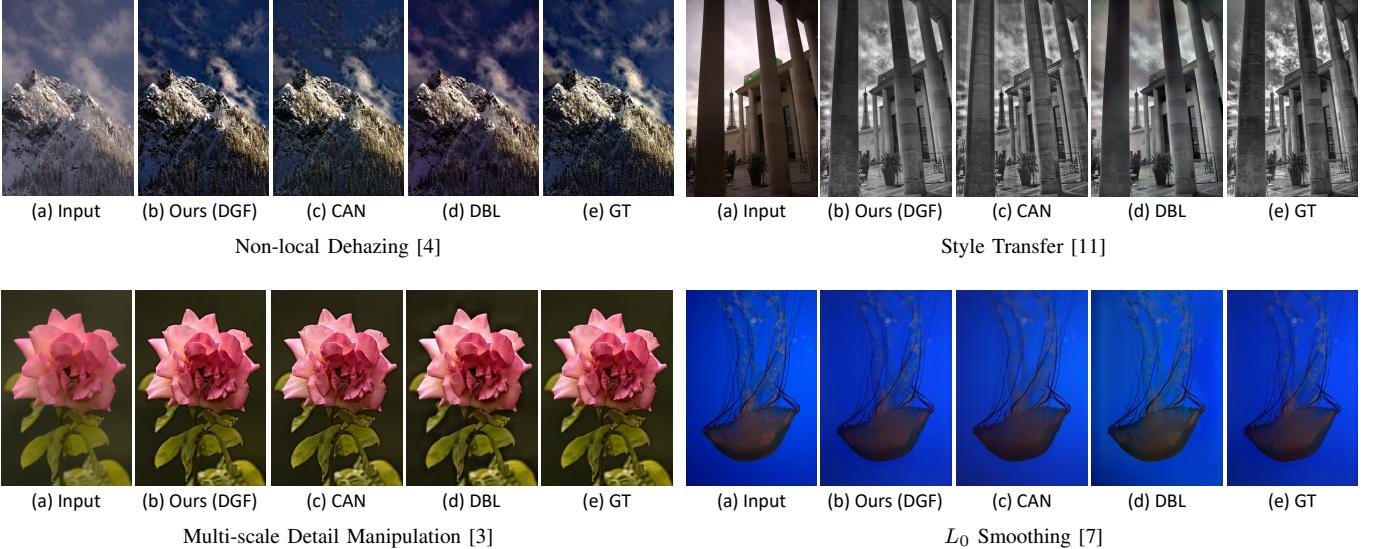


Fig. 8. **Qualitative Results in Image Processing.** Our method DGF^c is more visually appealing than other approaches. Best viewed in color.

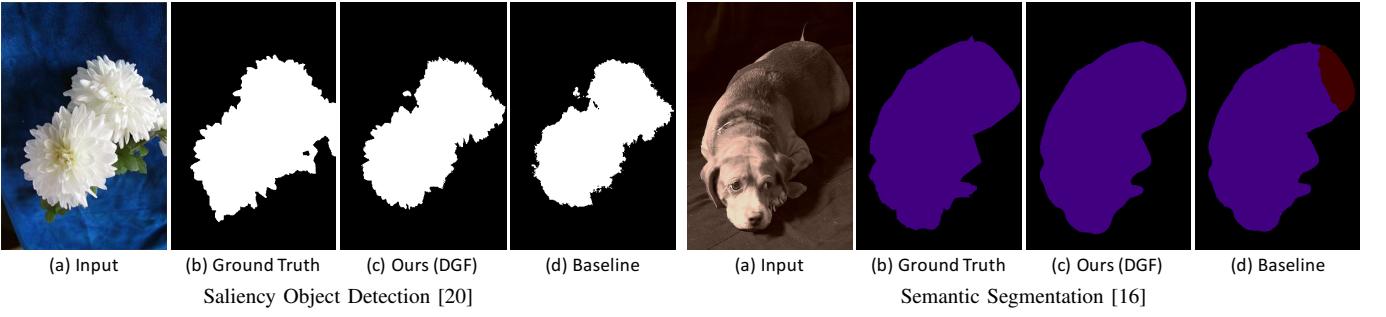


Fig. 9. **Qualitative Results in Computer Vision.** Best viewed in color.

VI. EXPERIMENTS: COMPUTER VISION TASKS

The proposed guided filtering layer can dramatically advance the performance of multiple image processing tasks in accuracy, speed, and memory usage. Moreover, our method can also be employed to replace the time-consuming conditional random field (CRF) in many computer vision applications. To evaluate the effectiveness of our method, we take an experiment on three computer vision tasks ranging from low-

level vision to high level-vision, namely depth estimation [19], saliency object detection [20], and semantic segmentation [16].

A. Details of Three Computer Vision Tasks

1) *Depth Estimation:* Depth estimation is proposed by Saxena *et al.* [19], which aims at predicting the depth at each pixel of an image with monocular cues. For this task, KITTI [53] is the most widely used dataset, which contains

42,382 rectified stereo pairs from 61 scenes. In this paper, 29,000/1,159 images from the official training set are used for training and evaluation, which covers 33 scenes. The remaining 28 scenes of the official training set contain 200 high-quality disparity images, which are used for testing in this paper.

2) *Saliency Object Detection*: Saliency object detection is used to detect the most salient object in an image, which is formulated as an image segmentation problem [20]. MSRA-B [54] and the official training/validation/test split [54] is used in our experiment.

3) *Semantic Segmentation*: Semantic segmentation aims at assigning each pixel of an image to one of the pre-defined labels [16]. To evaluate our method, PASCAL VOC 2012 benchmark [55] is used in this paper, which involves 20 foreground object classes and one background class. The original dataset contains 1,464, 1,449, and 1,456 pixel-wise labeled images for training, validation, and testing, respectively. The training set is further augmented by extra annotations [56], resulting in 10,582 images. We use the 10,582 augmented images for training and the 1,449 validation images for testing.

B. Details of DGF

When applying DGF to computer vision tasks, the high-resolution input image is directly processed by $C_l(I_l)$ without downsampling, generating the low-resolution output O_l . As for the architecture of $C_l(I_l)$, MonoDepth⁷ [6], DSS⁸ [5], DeepLab-V2⁹ [24] are employed for depth estimation, saliency detection, and semantic segmentation respectively. The corresponding training and testing procedures and loss functions are also used to train our network. As for the hyper-parameters of guided filtering layer, r and ϵ are determined by grid search on the validation set, as shown in Table V. Notably, a second guided filtering layer is applied in the saliency detection task to achieve better performance.

C. Main Results

The performances of our method and baseline methods are shown in Table VI. For depth estimation, DGF_s obtain 0.177 improvements in rms over the baseline. By end-to-end training and adding the learnable guidance map, we achieve the best performance (5.887) in rms. Similar results are obtained in saliency detection and semantic segmentation. F_β increases from 90.61% to 91.29% by applying the guided filtering layer to saliency detection. By replacing DGF_s with DGF, F_β further improves to 91.75%. For segmentation, DGF obtains 73.58% in mean IOU, which has an improvement of 1.79% compared to the baseline method.

We also compare our method with DenseCRF [57], which is commonly used in saliency detection and semantic segmentation. Experiments show that our method is comparable to DenseCRF in saliency detection, and obtains better performance in semantic segmentation. Besides, the proposed

⁷<https://github.com/mrharicot/monodepth>

⁸<https://github.com/wuhuikai/DeepGuidedFilter>

⁹<https://github.com/ish7/pytorch-deeplab-resnet>

TABLE V
HYPER-PARAMETERS OF GUIDED FILTERING LAYER.

Hyper-Params	1st guided filtering layer		2nd guided filtering layer	
	r	ϵ	r	ϵ
Depth	4	1e-2	-	-
Saliency	8	1e-2	8	1e-2
Segmentation	4	1e-2	-	-

TABLE VI
QUANTITATIVE COMPARISON ON COMPUTER VISION TASKS.

Method	Depth Estimation		Saliency Detection	Segmentation
	rms	log10	F_β	Mean IOU
Baseline	6.081	0.216	90.61%	71.79%
DenseCRF	-	-	91.87%	72.69%
DGF _s	5.904	0.211	91.29%	71.72%
DGF	5.887	0.209	91.75%	73.58%

layer performs at least 10 \times faster than DenseCRF. Averagely, our approach takes 34ms to process a 512² image, while DenseCRF takes 432ms.

Figure 9 shows the visual results of our method and baselines. The results obtained by our approach are better in preserving edges and details¹⁰.

VII. CONCLUSION

We present a novel building block for FCN, namely guided filtering layer, which aims at enhancing the ability of FCNs for joint upsampling. By formulating the guided filter into a fully differentiable module with learnable convolutional kernels, FCN-based pixel-wise image prediction approaches can benefit from end-to-end training and generate high-quality results. We further extend the proposed layer with a learnable transformation function, which makes it generalize well to different tasks by producing task-specific guidance maps. We integrate the guided filtering layer with FCNs and evaluate it on five image processing tasks and three computer vision tasks. Experiments show that the proposed layer could achieve state-of-the-art performance while taking 10-100 \times less computational cost. We also conduct a comprehensive ablation study, which demonstrates the contribution of each component as well as the hyper-parameters.

ACKNOWLEDGEMENT

This work is funded by the National Key Research and Development Program of China (Grant 2016YFB1001004 and Grant 2016YFB1001005), the National Natural Science Foundation of China (Grant 61673375, Grant 61721004 and Grant 61403383) and the Projects of Chinese Academy of Sciences (Grant QYZDB-SSW-JSC006 and Grant 173211KYS-B20160008). The authors would like to thank Patrick Pérez and Philip Torr for their helpful suggestions.

¹⁰More qualitative results are shown in <http://wuhuikai.me/DeepGuidedFilterProject/#visual>.

REFERENCES

- [1] H. Wu, S. Zheng, J. Zhang, and K. Huang, “Fast end-to-end trainable guided filter,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [2] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, “Learning photographic global tonal adjustment with a database of input / output image pairs,” in *CVPR*, 2011.
- [3] Z. Farbman, R. Fattal, and D. Lischinski, “Edge-preserving decomposition for multi-scale tone and detail manipulation,” *ACM TOG*, vol. 27, 2008.
- [4] D. Berman, T. Treibitz, and S. Avidan, “Non-local image dehazing,” in *CVPR*, 2016.
- [5] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr, “Deeply supervised salient object detection with short connections,” in *CVPR*, 2017.
- [6] C. Godard, O. Mac Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in *CVPR*, 2017.
- [7] L. Xu, C. Lu, Y. Xu, and J. Jia, “Image smoothing via l0 gradient minimization,” *ACM TOG*, vol. 30, 2011.
- [8] L. Xu, Q. Yan, Y. Xia, and J. Jia, “Structure extraction from texture via relative total variation,” *ACM TOG*, vol. 31, 2012.
- [9] Q. Zhang, X. Shen, L. Xu, and J. Jia, “Rolling guidance filter,” in *ECCV*, 2014.
- [10] K. Subr, C. Soler, and F. Durand, “Edge-preserving multiscale image decomposition based on local extrema,” *ACM TOG*, vol. 28, 2009.
- [11] M. Aubry, S. Paris, S. W. Hasinoff, J. Kautz, and F. Durand, “Fast local laplacian filters: theory and applications,” *ACM TOG*, vol. 35, pp. 1–14, 2014.
- [12] S. Bae and S. P. F. Durand, “Two-scale tone management for photographic look,” *ACM TOG*, 2006.
- [13] K. He, J. Sun, and X. Tang, “Single image haze removal using dark channel prior,” *IEEE TPAMI*, vol. 33, 2011.
- [14] R. Fattal, “Single image dehazing,” *ACM TOG*, vol. 27, 2008.
- [15] ——, “Dehazing using color-lines,” *ACM TOG*, vol. 34, 2014.
- [16] X. He, R. S. Zemel, and M. . Carreira-Perpinan, “Multiscale conditional random fields for image labeling,” in *CVPR*, 2004.
- [17] J. Shotton, J. Winn, C. Rother, and A. Criminisi, “Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation,” in *ECCV*, 2006.
- [18] Y. Cheng, R. Cai, Z. Li, X. Zhao, and K. Huang, “Locality-sensitive deconvolution networks with gated fusion for rgb-d indoor semantic segmentation,” in *CVPR*, 2017.
- [19] A. Saxena, S. H. Chung, and A. Y. Ng, “Learning depth from single monocular images,” in *NIPS*, 2005.
- [20] T. Liu, J. Sun, N.-N. Zheng, X. Tang, and H.-Y. Shum, “Learning to detect a salient object,” in *CVPR*, 2007.
- [21] G. Li and Y. Yu, “Visual saliency based on multiscale deep features,” in *CVPR*, 2015.
- [22] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand, “Deep bilateral learning for real-time image enhancement,” *ACM TOG*, vol. 36, no. 4, p. 118, 2017.
- [23] Q. Chen, J. Xu, and V. Koltun, “Fast image processing with fully-convolutional networks,” in *ICCV*, 2017.
- [24] 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,” in *arXiv*, 2016.
- [25] K. He, J. Sun, and X. Tang, “Guided image filtering,” *IEEE TPAMI*, vol. 35, no. 6, pp. 1397–1409, 2013.
- [26] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, “Joint bilateral upsampling,” *ACM TOG*, vol. 26, no. 3, 2007.
- [27] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in *ICCV*, 1998.
- [28] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, “Gaussian kd-trees for fast high-dimensional filtering,” *ACM TOG*, vol. 28, pp. 1–21:12, 2009.
- [29] A. Adams, J. Baek, and A. Davis, “Fast high-dimensional filtering using the permutohedral lattice,” in *Eurographics*, 2009.
- [30] E. S. L. Gastal and M. M. Oliveira, “Adaptive manifolds for real-time high-dimensional filtering,” *ACM TOG*, vol. 31, no. 4, p. 33, 2012.
- [31] J. T. Barron and B. Poole, “The fast bilateral solver,” in *ECCV*, 2016.
- [32] M. Gharbi, Y. Shih, G. Chaurasia, J. Ragan-Kelley, S. Paris, and F. Durand, “Transform recipes for efficient cloud photo enhancement,” *ACM TOG*, 2015.
- [33] J. Chen, A. Adams, N. Wadhwa, and S. W. Hasinoff, “Bilateral guided upsampling,” *ACM TOG*, vol. 35, no. 6, p. 203, 2016.
- [34] L. Yuan and J. Sun, “High quality image reconstruction from raw and jpeg image pair,” in *ICCV*, 2011.
- [35] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D: nonlinear phenomena*, 1992.
- [36] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, “Edge-preserving decompositions for multi-scale tone and detail manipulation,” in *ACM TOG*, 2008.
- [37] 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.
- [38] X. Shen, C. Zhou, L. Xu, and J. Jia, “Mutual-structure for joint filtering,” in *ICCV*, 2015.
- [39] B. Ham, M. Cho, and J. Ponce, “Robust image filtering using joint static and dynamic guidance,” in *CVPR*, 2015.
- [40] H. C. Burger, C. J. Schuler, and S. Harmeling, “Image denoising: Can plain neural networks compete with bm3d?” in *CVPR*, 2012.
- [41] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, “Deep joint demosaicking and denoising,” *ACM TOG*, 2016.
- [42] L. Xu, J. S. Ren, C. Liu, and J. Jia, “Deep convolutional neural network for image deconvolution,” in *NIPS*, 2014.
- [43] X. Shen, X. Tao, H. Gao, C. Zhou, and J. Jia, “Deep automatic portrait matting,” in *ECCV*, 2016.
- [44] D. Eigen, D. Krishnan, and R. Fergus, “Restoring an image taken through a window covered with dirt or rain,” in *ICCV*, 2013.
- [45] W. Ren, S. Liu, H. Zhang, J. Pan, X. Cao, and M.-H. Yang, “Single image dehazing via multi-scale convolutional neural networks,” in *ECCV*, 2016.
- [46] 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 TOG*, 2016.
- [47] L. Xu, J. Ren, Q. Yan, R. Liao, and J. Jia, “Deep edge-aware filters,” in *ICML*, 2015.
- [48] S. Liu, J. Pan, and M.-H. Yang, “Learning recursive filters for low-level vision via a hybrid neural network,” in *ECCV*, 2016.
- [49] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *CVPR*, 2017.
- [50] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *ECCV*, 2016.
- [51] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *CVPR*, 2017.
- [52] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang, “Deep joint image filtering,” in *ECCV*, 2016.
- [53] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *CVPR*, 2012.
- [54] H. Jiang, J. Wang, Z. Yuan, Y. Wu, N. Zheng, and S. Li, “Salient object detection: A discriminative regional feature integration approach,” in *CVPR*, 2013.
- [55] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.”
- [56] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, “Semantic contours from inverse detectors,” in *ICCV*, 2011.
- [57] P. Krähenbühl and V. Koltun, “Efficient inference in fully connected crfs with gaussian edge potentials,” in *NIPS*, 2011.