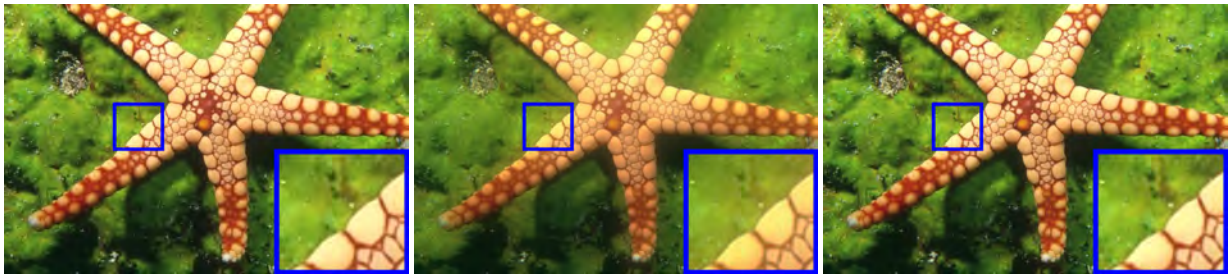
Iterative First-Order Reverse Image Filtering

Li Dong
Faculty of Electrical Engineering and Computer Science
Ningbo University, Ningbo, China
dongli@nbu.edu.cn

Jiantao Zhou
Department of Computer and Information Science
University of Macau, Macau, China
jtzhou@umac.mo

Cuiming Zou
School of Information Science and Engineering
Chengdu University, Chengdu, China
zoucuiming2006@163.com

Yulong Wang
School of Information Science and Engineering
Chengdu University, Chengdu, China
wangyulong6251@gmail.com



(a) Original (b) Filtered (c) Reverse Filtered

Figure 1: Reverse image filtering. (a) Original image. (b) Filtering the image using the BeFunky Photo Editor with their provided Smoothing service. (c) By only access the interface of the smoothing function (i.e., the provided smoothing service is treated as a black-box), our method could successfully remove the smoothing effects, and restore a sharpen image that is very close to the original image. Part of the image is enlarged for better visual comparison.

ABSTRACT

In this paper, we study an unconventional but practically meaningful problem *reverse image filtering*, which aims at removing the filtering effects with the given image filter. We propose an iterative first-order reverse image filtering algorithm based on the Newton-Raphson method. The convergence of the proposed iterative procedure is verified via extensive experiments. The experimental results show that our method could effectively reverse many prevalent image filters. Our method finds its practical usage in many applications, such as non-blind image deblurring and image details recovery. Furthermore, our method could probe the behaviors of black-box image filtering systems, which poses security concerns on the protection for the privately commercial image filters.

KEYWORDS

Reverse image filtering, Newton-Raphson method

ACM TURC-AIS 2019, May 17–19, 2019, Chengdu, China

ACM ISBN 978-1-4503-7158-2/19/05…$15.00
https://doi.org/10.1145/3321408.3326672

ACM Reference Format:
Li Dong, Jiantao Zhou, Cuiming Zou, and Yulong Wang. 2019. Iterative First-Order Reverse Image Filtering. In *TURC-AIS 2019: ACM TURC Conference on Artificial Intelligence and Security, May 17–19, 2019, Chengdu, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3321408.3326672

1 INTRODUCTION

Image filtering is one of the fundamental operations in image processing and computer vision systems. Over the past several decades, a variety of filters have been designed for fulfilling different purposes, e.g., image noise suppression [2, 4], texture removal [12, 13] and edge-perverse smoothing [5, 10].

Unlike the previous works, we here do not intend to develop a new image filter. Instead, we study an unconventional problem of reverse image filtering, which aims at

> *removing part or even all the filtering effects, with the* ***accessible*** *but* ***not necessarily known*** *image filter.*

To better understand this emerging problem, we illustrate in Fig.1 with a concrete example. The original image is retouched by the BeFunky Photo Editor using their provided Smoothing function[1]. As shown in Fig.1-(b), the photo editing system softens the original image by suppressing the fine details. Given the retouched image, one may want to recover the original image, i.e., reverse the filtering. One straightforward way is to compute the inverse function of the filter, and then apply it to the filtered image. However, the photo

[1] Available online: https://www.befunky.com/create/.

editor only provides the service interface, and we are definitely not clear how this smoothing function is implemented. It is therefore impossible to calculate the inverse of the filter explicitly. In contrast, without requiring the exact form of the filter, our method could successfully remove the smoothing effects, and restore a sharpen image that is very close to the original image (see Fig.1-(c)).

The topic of reverse image filtering is a less explored area. To the best of our knowledge, the most relevant work might be Tao *et al.* [9]. They proposed a generic reverse filtering scheme called zero-order reverse filtering. The validity of this method was interpreted via Banach fixed point theorem. The experiments have shown their approach works well on a number of commonly used filters. However, the experimental results have also revealed that, for the most typical linear image filters that are based on convolution, e.g., Gaussian filter, zero-order reverse filtering is less effective or even failed, especially for the filter with the small support range.

As a well known fact, the convolution-based linear image filtering is a crucial building block of many prevalent image processing systems, from the simple Gaussian filtering to the complicated convolution neural network. Thus, in this work, we focus on the reverse of the convolution-based linear image filtering. Specifically, we propose an iterative first-order reverse image filtering algorithm based on the Newton-Raphson method. Our algorithm is simple but effective, and its convergence is validated via extensive experiments. Notably, we find that the seminal work [9] can be included into our method as a special case. The experiential results show that the proposed method could effectively reverse many commonly used image filters. The practical usage is demonstrated with non-blind image deblurring and image detail recovery applications.

Before moving on, we would like to clarify the difference between *reverse image filtering* with the conventional *image deconvolution*. First, traditional blind/non-blind image deconvolution methods often require some prior knowledge on the convolution kernel, e.g., the centro-symmetric property of the point spread function (i.e., the convolution kernel) [3]. Even for the blind image deconvolution, the kernel still shall be estimated beforehand. In contrast, our method could treat the image filtering process as a black-box and requires no prior information about the kernel. Second, most of existing image deconvolution methods only suitable for some specifc filters, e.g., motion bluring; while we experimentally find that, our method could even work properly for some image filtering operations that cannot be strictly expressed in convolutional form, e.g., image guided filter [6].

The rest of this work is organized as follows. In Section 2, we present the proposed iterative reverse image filtering method. In Section 3, the effectiveness of our algorithm is verified on the commonly used filters, along with the demonstration on two real applications. We finally discuss the limitations and make a conclusion in Section 4.

2 PROPOSED METHOD

In principle, an image filtering operation can be expressed as

$$\mathbf{Y} = f(\mathbf{X}), \tag{1}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are the original and the filtered images, respectively. The function $f(\cdot)$ is the image filter. It can be linear or non-linear, local or global.

Algorithm 1 Iterative first-order reverse image filtering

Input: Filtered image $\mathbf{Y}$, filter $f(\cdot)$, number of iterations N.
Output: Estimate of the original image $\hat{\mathbf{X}}$.

1: $\hat{\mathbf{X}}^{(0)} \leftarrow \mathbf{Y}$.
2: **for** $t = 0$ to $N-1$ **do**
3: $\quad \hat{\mathbf{X}}^{(t+1)} \leftarrow \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\mathbf{Y})\cdot\mathcal{F}(\hat{\mathbf{X}}^{(t)})}{\mathcal{F}\left(f(\hat{\mathbf{X}}^{(t)})\right)}\right)$.
4: **end for**
5: **return** $\hat{\mathbf{X}}^{(N)}$.

Our goal is to estimate the original image $\mathbf{X}$ without computing $f^{-1}(\cdot)$. To be more specific, we focus on the widely-used linear image filtering based on convolution operation, which can be written as

$$\mathbf{Y} = f(\mathbf{X}) = \mathbf{X} \otimes \mathbf{K}, \tag{2}$$

where $\mathbf{K}$ is the convolution mask, i.e., the kernel. Alternatively, (2) can be reformulated as a compact vector form $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{x}$ and $\mathbf{y}$ are the vectorized original and the filtered images, respectively; and $\mathbf{A}$ is the Toeplitz matrix describing the filtering process. In this context, our goal becomes to find the root of the following equation

$$f(\mathbf{x}) - \mathbf{y} = 0. \tag{3}$$

Clearly, in the case of convolution, the filter $f(\cdot)$ is a continuously differentiable function. We can thus apply the Newton-Raphson method [1] to successively solve (3) as follows

$$J_f(\hat{\mathbf{x}}^{(t)}) \cdot \hat{\mathbf{x}}^{(t+1)} = J_f(\hat{\mathbf{x}}^{(t)}) \cdot \hat{\mathbf{x}}^{(t)} + \mathbf{y} - f(\hat{\mathbf{x}}^{(t)}), \tag{4}$$

where $\hat{\mathbf{x}}^{(t)}$ denotes the estimate of $\mathbf{x}$ at the t-th iteration; and $J_f(\mathbf{x})$ is the Jacobian matrix that is evaluated at $\mathbf{x}$. Since the Jacobian matrix is the first-order partial derivatives of a vector-valued function, we therefore call our method the *iterative first-order reverse filtering*. Interestingly, we find that, if fix the Jacobian as unity matrix, (4) can be simplified as

$$\hat{\mathbf{x}}^{(t+1)} = \hat{\mathbf{x}}^{(t)} + \mathbf{y} - f(\hat{\mathbf{x}}^{(t)}), \tag{5}$$

which is exactly the zero-order reverse filtering [9]. In this light, [9] can be incorporated into our method as a special case.

However, the direct calculation of Jacobian requires extensive computation and huge memory resources. For instance, for an image of size 100×100, its Jacobian matrix is in size of $10^4 \times 10^4$. Fortunately, we notice the fact that

$$J_f(\mathbf{x}) \cdot \mathbf{x} = \lim_{\epsilon \to 0} \frac{f(\mathbf{x} + \epsilon\mathbf{x}) - f(\mathbf{x})}{\epsilon} = \mathbf{A}\mathbf{x} = f(\mathbf{x}). \tag{6}$$

Then, (4) can be simplified as

$$J_f(\hat{\mathbf{x}}^{(t)}) \cdot \hat{\mathbf{x}}^{(t+1)} = \mathbf{y}, \tag{7}$$

which is equivalent to

$$\hat{\mathbf{X}}^{(t+1)} \otimes \mathbf{K}^{(t+1)} = \mathbf{Y}. \tag{8}$$

Here $\mathbf{K}^{(t+1)}$ is the kernel used in $(t+1)$-th iteration, which can be derived from the estimate of $\mathbf{X}$ in the previous iteration. More specifically, $\mathbf{K}^{(t+1)}$ is the solution of the following optimization problem

$$\mathbf{K}^{(t+1)} = \arg\min_{\mathbf{K}} \|\hat{\mathbf{X}}^{(t)} \otimes \mathbf{K} - f(\hat{\mathbf{X}}^{(t)})\|_2^2, \tag{9}$$

Table 1: Comparison of the reversed image quality (PSNR in dB) with Tao *et al.* [9] on the BSD500 image dataset. For each cell of the last two rows, the top number is for [9], and the bottom one is for ours. The best results are highlighted in bold. N/A indicates that the PSNR value is smaller than 0, which means the method is failed.

Filters	GS	DK	LOG	MT	UMF	BF	GF	AWF	L0	MF
Init	24.89	22.97	5.12	23.88	30.35	31.74	31.44	35.93	36.17	29.39
Final	16.92	N/A	N/A	N/A	51.17	40.59	42.03	27.63	**37.06**	N/A
	211.68	**242.09**	**43.04**	**251.43**	**59.02**	**41.34**	**43.54**	**29.67**	34.14	19.04
Best	26.40	23.28	N/A	23.91	51.27	41.43	42.03	36.62	**37.87**	**29.32**
	214.37	**246.45**	**54.42**	**258.24**	**60.40**	**42.52**	**44.20**	**36.86**	36.89	26.34

which permits a closed-form solution

$$\mathbf{K}^{(t+1)} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}\left(f(\hat{\mathbf{X}}^{(t)})\right)}{\mathcal{F}(\hat{\mathbf{X}}^{(t)})}\right), \tag{10}$$

where $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ denote the 2D forward and inverse Fourier transforms, respectively. Note that here the division (and the multiplication used later) between Fourier transformed coefficients are element-wise. By plugging (10) into (8), one can readily obtain

$$\hat{\mathbf{X}}^{(t+1)} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\mathbf{Y}) \cdot \mathcal{F}(\hat{\mathbf{X}}^{(t)})}{\mathcal{F}\left(f(\hat{\mathbf{X}}^{(t)})\right)}\right). \tag{11}$$

From (11), we can observe that the updated estimate of $\hat{\mathbf{X}}^{(t+1)}$ merely depends on the previous estimate $\hat{\mathbf{X}}^{(t)}$ and the given filtered image $\mathbf{Y}$. In addition, (11) involves the image filter $f(\cdot)$ as a black-box. It does not need to know the exact form of $f(\cdot)$.

The whole procedure is summarized in Algorithm 1. To boot up the algorithm, $\hat{\mathbf{X}}^{(0)}$ is initialized as $\mathbf{Y}$. The only required algorithmic parameter is the number of iterations. In our experiments, the default setting of this parameter is 20. This setting will be experimentally justified in Section 3.1.

Remarks on the convergence of Algorithm 1: Essentially, our method solve the root-finding problem in the context of $f(\cdot)$ being convolution, which based on the Newton-Raphson method shown in (4). The convergence property of Newton-Raphson method has been well studied, and the detailed theoretical analysis can be found in [1]. We thus omit this analysis here. In Section 3.2, we validate the convergence of the Algorithm 1 through extensive experiments.

3 EXPERIMENTAL RESULTS

All the experiments are conducted on a PC with Intel Core i5-4570 3.2 GHz CPU and 4G RAM. In the following, we first verify the effectiveness of the proposed method on some common filters, and then demonstrate the practical usage with two real applications. **The source code is publicly available at: https://github.com/nbudongli/reversefiltering**.

3.1 Evaluation on Some Common Filters

We collect all the 500 images from the Berkeley segmentation dataset [8] as the test image set BSD500. In order to observe how the quality of the reversed image varies along with iteration, the iteration number N in Algorithm 1 is first intentionally set as a large value, e.g., 50.

Our method is evaluated and compared with the relevant work Tao *et al.* [9] on 10 commonly used image filters, which includes Gaussian Filter (GS), Disk Filter (DK), Laplacian of Gaussian Filter (LOG), Motion Blurring (MT) and Unsharp Masking Filter (UMF), Bilateral Filter (BF) [10], Guided Filter (GF) [6], Adaptive Wiener Filter (AWF) [7], L0 Smooth (L0) [11] and Median Filter (MF) [7]. Note that the first five filters perform linear image filtering based on convolution operation[2]. while the last five filters are more sophisticated and may beyond the image convolution. Nevertheless, we also include the last five filters in the experiments to test the applicable scope of our proposed method.

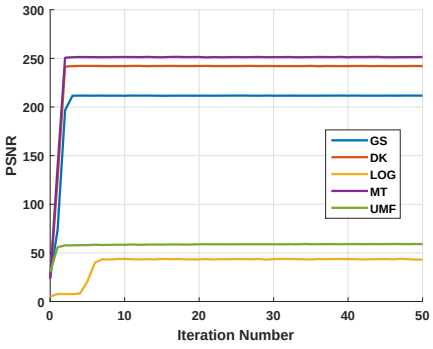
The PSNR value is adopted as the metric to measure the difference between two images. In the following, we report three PSNRs: 1) the initial PSNRs (**Init**), which indicates the PSNR between the filtered image (i.e., the image before applying our method) and the original image; 2) the final PSNRs (**Final**), which indicates the PSNR between the reversed image (i.e., the image after applying our method for 50 iterations) and the original image; 3) the best PSNRs (**Best**) attained over the entire iterative process. The quantity **Best** is provided here because we observe the PSNR values yielded by some filters may oscillate.

The results are complied in Table 1, from which we can make several observations. First, for the convolution-based image filters such as GS, DK, LOG, MT and UMF, our method could effectively removing the filtering affects. In particular, for the filters GS, DK and MT, the PSNRs achieved by our method are all larger than 200dB, which implies that the restored image is nearly the same as the original one. In contrast, Tao *et al.* [9] often fail to reverse the filtering (denoted by N/A). Second, for the two sophisticated filters BF and GF, our method could also recover the original image to some extent, with slightly better results (improves around 1dB) compared with that of Tao *et al.* [9]. Finally, for the filters consisting of discontinuous operations such as MF, AWF and L0, both Tao *et al.* [9] and our method cannot reverse the image filtering. We claim that those filters cannot be reversed by our algorithm.

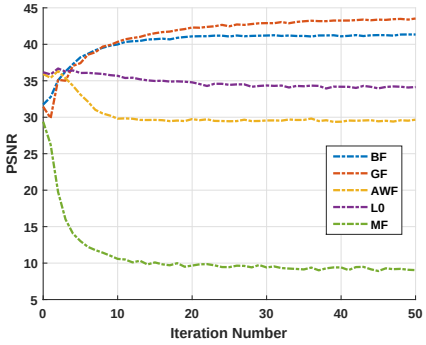
3.2 Convergence of Algorithm 1

To investigate the convergence properties of our method on different filters, we compute and record the PSNR value at each iteration.

[2]All those filters can be found in software MATLAB. Please refer to the functions `imfilter` and `imsharpen` for more details.



(a)



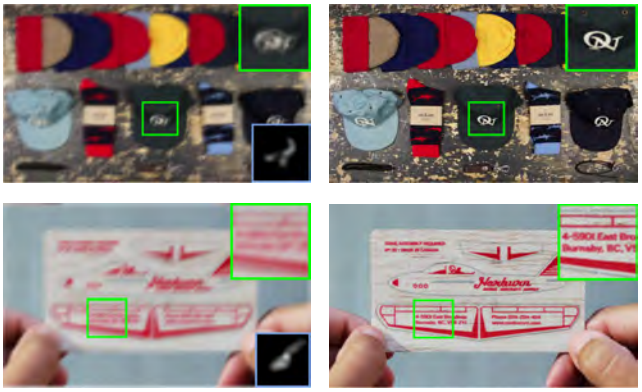
(b)

Figure 2: PSNR-iteration curves for 10 common filters. (a) Results for Gaussian Filter (GS), Disk Filter (DK), Laplacian of Gaussian Filter (LOG), Motion Blurring (MT) and Unsharp Masking Filter (UMF). (b) Results for Bilateral Filter (BF) [10], Guided Filter (GF) [6], Adaptive Wiener Filter (AWF) [7], L0 Smooth (L0) [11] and Median Filter (MF) [7].

The PSNR-iteration curves are shown in Fig.2. It can be seen that our method often converges after around 20 iterations. Specifically, in Fig.2-(a), one can observe that, for the filters GS, DK, LOG, MT and UMF, the PSNR values can be significantly improved in several iterations, and converge in less 10. This verifies the convergence of the Algorithm 1 on those filters. For the filters BF, GF, AWF, L0 and MF, our method still converges, but typically after 20 iterations (see Fig.2-(b)). As we already analyzed in above, for the filter BF and GF, our method is only partially effective. This can be observed from the fact that the converged PSNRs only improve around 10dB compared with the initial PSNRs. For the filters AWF, L0 and MF, our method also converges, but with a general decrement of PSNRs. This phenomenon again shows that our method cannot be applied to reverse those filters.

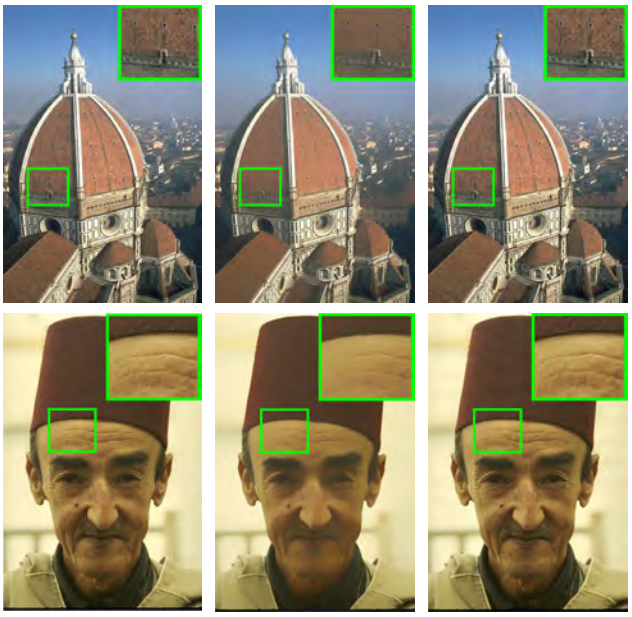
3.3 Real Applications

In this subsection, we illustrate the practical usage of our method with two real applications. The first application is the non-blind image deblurring. With the provided or estimated image kernel, one can readily construct the blurring process, which plays the role of the filtering function $f(\cdot)$ in Algorithm 1. As shown in Fig.3, by



(a) Input Image (b) Our Results

Figure 3: Non-blind image deblurring using our method. (a) Input images with the blur kernels (shown in the right-bottom rectangle). (b) The restored sharp images. Part of the image is enlarged for better visual comparison.



(a) Original (b) Filtered (c) Recovered

Figure 4: Recover the image details that are suppressed by bilateral image filtering. (a) Original image. (b) Filtered image. (c) Recovered image. Part of the image is enlarged for better visual comparison.

applying our algorithm to those blurred images, we could restore the underlying sharp image structures (e.g., the small characters).

Another practical application is to recover the details suppressed by the image filter. To illustrate this, we take the widely-used bilateral filtering as an example. As can be seen from Fig.4-(b), the filtered image appears to be much smoother than the original one. Many textural structures and fine-details turn out to be invisible. Interestingly, our method could bring back such *lost* fine-details.

This can be observed from texture of the roof and the forehead wrinkles of the old man in Fig.4-(c).

4 CONCLUSION

In this work, we study an unconventional but interesting problem: reverse image filtering. We tackle this problem by employing the Newton-Raphson method, and propose an iterative first-order reverse image filtering algorithm. The convergence of the proposed method is verified via extensive experiments. The experimental results show that our method could significantly outperforms the previous zero-order reverse filtering technique, especially on the reversing the convolution-based linear image filtering task. The practical usage of our method is illustrated on two real applications, and we believe the proposed method could be applied to many other potential applications, e.g., image forensics.

Although our method has demonstrated the effectiveness on reversing many image filters, it still has several limitations. First, our method works well for the convolution-based linear filtering, but for some filters consisting of many discontinuous operations, e.g., the median filter, the effectiveness of our method would degrade or even completely fail. Second, the image filter can be treated as a black-box in our method, but it still requires the filter available as a deterministic component, which might not be satisfied sometimes.

REFERENCES

[1] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C. Sagastizábal. 2013. *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media.

[2] A. Buades, B. Coll, and J.-M. Morel. 2005. A non-local algorithm for image denoising. In *Proc. IEEE Conf, Comput. Vis. Pattern Recognit.*, Vol. 2. 60–65.

[3] T. Chan and C.-K. Wong. 1998. Total variation blind deconvolution. *IEEE Trans. Image Process.* 7, 3 (1998), 370–375.

[4] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. [n. d.]. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Trans. Image Process.* 16, 8 ([n. d.]).

[5] E. S. Gastal and M. M. Oliveira. 2011. Domain transform for edge-aware image and video processing. In *ACM Trans. Graph.*, Vol. 30. 69.

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

[7] J.S. Lim. 1990. Two-dimensional signal and image processing. *Englewood Cliffs, NJ, Prentice Hall, 710 p.* (1990).

[8] David M., Charless F., Doron T., and Jitendra M. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. IEEE Int. Conf. Comput. Vis.*, Vol. 2. 416–423.

[9] X. Tao, C. Zhou, X. Shen, J. Wang, and J. Jia. 2017. Zero-order Reverse Filtering. In *Proc. IEEE Int. Conf. Comput. Vis.*

[10] C. Tomasi and R. Manduchi. 1998. Bilateral filtering for gray and color images. In *Proc. IEEE Int. Conf. Comput. Vis.* 839–846.

[11] L. Xu, C. Lu, Y. Xu, and J. Jia. 2011. Image smoothing via L0 gradient minimization. *ACM Trans. Graph.* 30, 6 (2011), 174.

[12] L. Xu, Q. Yan, Y. Xia, and J. Jia. 2012. Structure extraction from texture via relative total variation. *ACM Trans. Graph.* 31, 6 (2012), 139.

[13] Q. Zhang, X. Shen, L. Xu, and J. Jia. 2014. Rolling guidance filter. In *Proc. Eur. Conf. Comput. Vis.* Springer, 815–830.