A Generic Deep Architecture for Single Image Reflection Removal and Image Smoothing

Qingnan Fan*[1] Jiaolong Yang[2] Gang Hua[2] Baoquan Chen[1,3] David Wipf[2]
[1]Shandong University [2]Microsoft Research [3]Shenzhen Research Institute, Shandong University
fqnchina@gmail.com, {jiaoyan,davidwip,ganghua}@microsoft.com, baoquan@sdu.edu.cn

Abstract

This paper proposes a deep neural network structure that exploits edge information in addressing representative low-level vision tasks such as layer separation and image filtering. Unlike most other deep learning strategies applied in this context, our approach tackles these challenging problems by estimating edges and reconstructing images using only cascaded convolutional layers arranged such that no handcrafted or application-specific image-processing components are required. We apply the resulting transferrable pipeline to two different problem domains that are both sensitive to edges, namely, single image reflection removal and image smoothing. For the former, using a mild reflection smoothness assumption and a novel synthetic data generation method that acts as a type of weak supervision, our network is able to solve much more difficult reflection cases that cannot be handled by previous methods. For the latter, we also exceed the state-of-the-art quantitative and qualitative results by wide margins. In all cases, the proposed framework is simple, fast, and easy to transfer across disparate domains.

1. Introduction

Inspired by the tremendous success of deep learning for large-scale visual recognition tasks like ILSVRC [28, 20], a variety of recent work has investigated deep neural networks for low-level computer vision tasks such as image denoising [27, 11], shadow removal [15], and image smoothing [38, 24]. Given that edges represent an important cue in addressing many of these problems, networks that can replace computationally-expensive or otherwise inflexible edge-aware filters naturally show promise.

For example, the underlying goal of image smoothing is to extract sparse salient structures, like perceptually important edges and contours, while minimizing the color differences in image regions with low amplitude. To approximate different edge-sensitive image smoothing filters which potentially have slow runtimes [2, 6, 8, 26, 37, 39, 42, 43] with deep networks, it has been proposed to first learn a salient gradient/weight map and then subsequently filter images via simpler, weighted optimization procedures [38] or iterative recursive processing techniques [24]. The above approaches focus on solving a single/major problem using a plain CNN model followed by more traditional, inflexible operations inspired by fixed filtering methods. Consequently, they are not fully extensible to implementing broader image smoothing effects or other significantly different problems such as image layer separation.

In this latter regard, one typical case where gradient domain statistics are relevant is in dealing with image reflections, that are often at least partially out of focus, when provided with a single image. When taking a photo through a glass window, the glare or reflection tends to distract the eye from the scene behind the glass. Many attempts to mitigate these effects, such as using a polarizer [19, 31], draping a large piece of black cloth over the lens and the glass to block ambient light from behind, or changing positions [22, 40, 41], are simply infeasible in many practical situations. Moreover, when taking photographs in airplane, museum, aquarium, or related environments, there is no other recourse but to shoot through the window. Consequently, it is common for photographers to simply widen the aperture of the camera and blur out the reflections.

To address this reflection removal problem from a computational perspective, traditional imaging models assume that the captured image $\mathbf{I}$ is a linear combination of a background layer $\mathbf{B}$ and a reflection layer $\mathbf{R}$, *i.e.*, $\mathbf{I} = \mathbf{B} + \mathbf{R}$. Obviously this is an ill-posed problem as there exist infinite feasible solutions, and hence most reflection removal algorithms require multiple input images [7, 30, 1, 19, 12, 22, 40, 41] or manual user interactions [21] to label reflection- and background-layer gradients, thus condensing the space of candidate solutions. However, one exploitable property in the reflection removal problem is that the gradients or perceptual structures of the two layers exhibit different distributions, since reflections often display a greater degree

*This work was done when Qingnan Fan was an intern at MSR.

of blurring. This then naturally leads us towards edge-based solutions, with data-driven network variants considered herein.

In this paper, we present a *Cascaded Edge and Image Learning Network (CEILNet)* that can be tailored to solve different image processing tasks such as layer separation (*e.g.*, reflection removal) and image filtering (*e.g.*, image smoothing). We rely on an overriding generic structure that is specialized in each instance via domain-specific edge information. The core framework operates in a very intuitive way. In brief, we separate the difficult task of directly predicting an image into two subproblems: (*i*) predicting the edge maps of the target images via a deeply supervised sub-network, and then (*ii*) reconstructing the target images by leveraging the predicted edge maps. These tasks are learned end-to-end by cascading two similar simple CNNs, and no hand-crafted modules are required. The edge map represents any color difference between each pair of adjacent pixels for task-specific target images, instead of sparse salient structures as in edge detection problems.

Of course, these objectives require ample training data to be feasible in practice. For image smoothing, this is not especially problematic provided sufficient computational resources are available for producing filter outputs across a corpus of images. However, for many layer separation tasks ground-truth instances are scarce. We therefore propose a novel weakly supervised learning method for training our reflection removal pipeline. This involves the use of images synthetically corrupted via reflections that mimic the physical properties of those found in natural scenes.

Our contributions can be summarized as follows:

- We propose a new, generic Cascaded Edge and Image Learning Network (CEILNet) that relies only on convolutional layers and is specifically designed to tackle edge-sensitive image processing tasks without resorting to any handcrafted, application-specific components. This structure is fast, extensible, and easy to reproduce, facilitating the seamless transfer to different low-level vision problems.
- We are the first to solve the challenging layer-separation problem of reflection removal from single images using deep learning techniques. We also propose a novel weakly supervised learning strategy combined with CEILNet.
- Beyond reflection removal, we demonstrated state-of-the-art visual and numerical performance using CEILNet on the image smoothing task, surpassing previous methods by a wide margin.

2. Related Work

Reflection Removal: Reflection removal is fundamentally an underdetermined problem and therefore requires prior knowledge or additional information to achieve any degree of success. Perhaps the most popular practical remedy is to use multiple input images, such as flash/non-flash image pairs [1], focus/defocus pairs [30], video sequences where background and reflection exhibit different motions [7, 34, 29, 9, 22, 33, 12, 40, 41], or those obtained through a polarizer at two or more orientations [19, 31, 29]. A few ambitious approaches attempt single image reflection removal, a far more difficult but practical scenario. In [21], manual annotation is required to guide an optimization-based layer separation. [32] compensates for the limited information by exploiting ghost cues, but this approach is not applicable beyond this somewhat specialized situation, or in the majority of practical cases. [35] leverages a multi-scale DoF computing strategy to separate reflection from background.

In terms of automatic reflection removal from a single image with minimal assumptions, the work most closely related to ours is [23]. This approach assumes the reflected layer is relatively blurry compared to the background scene, thus large gradients in it are strongly penalized in their optimization. However, we observe that the reflection in many real-world photographs, although indeed sometimes out of focus or blurry, is nonetheless produced by bright lights and often comprises the brightest portion of an image. The regional gradients associated with these reflections can therefore be quite large, violating the assumption in [23]. In this work, we synthesize a database of training samples that better capture the background and reflection statistics, and replace prior knowledge injected through explicit gradient penalization or energy minimization with a particular deep network to capitalize on this form of weak supervision. Empirically we will later show that indeed significant improvement is possible on real images.

Image Smoothing: Given the recent effectiveness of parallel computation through GPUs, and the strong learning capability of deep neural networks, replacing computationally-expensive, optimization-based smoothing filters with cheap neural modules has drawn a lot of attention [38, 24]. However, because accurately capturing smoothing effects with a fully convolutional deep network can be challenging, [38] trains a shallow CNN on the gradient domain followed by an optimized image reconstruction post-processing step with sensitive parameters tuned for each different smoothing filter. From a somewhat different perspective, by treating spatially-variant recursive networks as surrogates for a group of distinct filters, [24] combines sparse salient structure prediction implemented as CNN with image filtering in a hybrid neural network.

While significant differences exist, all of these prior methods lean on traditional optimization or filtering techniques at some point in their pipelines. Moreover, they are mostly applied to image smoothing using filter- or effect-

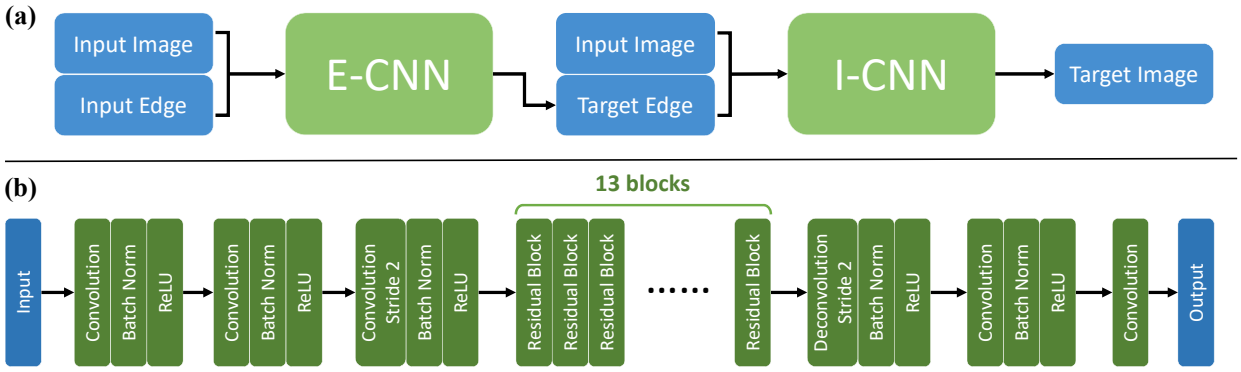

Figure 1. The proposed deep network architecture CEILNet. **(a)** The cascaded edge and image prediction pipeline. Two CNN networks, E-CNN and I-CNN, are used for edge prediction and image reconstruction, respectively. I-CNN takes the output of E-CNN as input, giving rise to an end-to-end and fully convolutional solution. **(b)** The detailed CNN structure shared by E-CNN and I-CNN.

dependent implementations without a universal, trainable parametric structure. This can potentially contribute to degraded performance since no single optimization or filtering strategy is likely to generalize to all different image smoothing effects. In contrast, our method learns a generic, fully-convolutional structure with no attendant postprocessing or otherwise fixed, filter-inspired structures. Empirical experiments demonstrate that this revised strategy outperforms the best existing work by a wide margin.

3. Network Structure

Our network consists of two cascaded sub-networks: an edge prediction network E-CNN and an image reconstruction network I-CNN. Figure 1 is a schematic description of the architecture, which is unchanged for both the reflection removal and image smoothing applications.

3.1. E-CNN: The Edge Prediction Network

When dealing with edge-sensitive image processing tasks like reflection removal and image smoothing, edges-related cues are naturally leveraged by many existing algorithms [21, 22, 40, 10, 38]. Similarly, given a source image $\mathbf{I}^s$, we apply a CNN to learn an edge map $\mathbf{E}^t$ of the target image $\mathbf{I}^t$ (*i.e.*, the background layer for reflection removal or the smoothed image for image smoothing). Note that the goal is to predict the edges of the *target image*, not the *input image*, and it is crucial not to confuse this procedure with conventional edge detection [3, 36].

In this work, our edge map is not binary, as we empirically found binary edge maps are less informative for the subsequent image reconstruction. Instead, we designed a simple but effective edge representation: the mean absolute color difference between a center pixel and its four-connected neighbors. Specifically, the edge map $\mathbf{E}$ of an image $\mathbf{I}$ is computed by:

$$\mathbf{E}_{x,y} = \frac{1}{4}\sum_{c}\big(|\mathbf{I}_{x,y,c} - \mathbf{I}_{x-1,y,c}| + |\mathbf{I}_{x,y,c} - \mathbf{I}_{x+1,y,c}| + |\mathbf{I}_{x,y,c} - \mathbf{I}_{x,y-1,c}| + |\mathbf{I}_{x,y,c} - \mathbf{I}_{x,y+1,c}|\big) \quad (1)$$

where x, y are the pixel coordinates and c refers to the channels in the RGB color space.

In order to ease the computation, we augment the source image $\mathbf{I}^s$ with its edge map $\mathbf{E}^s$ as an additional channel for input. The intuition behind is simple: either a reflection-free background layer or an image smoothed via a filtering process can be viewed as "simplified" versions of the original source images, and their edge maps are roughly "attenuated" versions of the source image edge maps. We observed that such an augmentation can not only lead to better results but also significantly accelerate the convergence during training. In summary, E-CNN approximates the following function f:

$$\mathbf{E}^t = f(\mathbf{I}^s, \mathbf{E}^s) \quad (2)$$

3.2. I-CNN: The Image Reconstruction Network

The second sub-network, I-CNN, is designed to reconstruct the target image $\mathbf{I}^t$ by learning how to process the input image $\mathbf{I}^s$ given the target edge map $\mathbf{E}^t$ predicted by E-CNN. In other words, it approximates the following function g:

$$\mathbf{I}^t = g(\mathbf{I}^s, \mathbf{E}^t) \quad (3)$$

The input image and the target edge are combined to be a 4-channel tensor as input, similar to E-CNN, hence their shared use of the same overall structure. Additionally, in the context of the edge-based image reconstruction step of image smoothing tasks, the I-CNN serves as a multi-purpose, data-driven substitution for traditional fixed filtering operations or optimization-based postprocessing structures.

1. Train E-CNN and I-CNN in parallel, with loss functions of Eq. 4 and Eq. 5 respectively.
2. Jointly train (fine-tune) E-CNN and I-CNN end-to-end, with loss in Eq. 6.

Figure 2. Our two-phase network training algorithm.

3.3. Details of CNN Layers

For simplicity, we employ the deep CNN structure shown in Fig. 1 (b) for both E-CNN and I-CNN. The two sub-nets only differ in the channel number of the final output, *i.e.*, 1 for E-CNN *vs*. 3 for I-CNN. In each case, we employ 32 convolutional layers with the same 3×3 kernel size (except for the third-to-last layer; see below). The intermediate 30 convolutional layers all have 64-dimensional input and output feature maps. The first 31 layers are followed by batch normalization (BN) and ReLU. To ensure better contextual information, we enlarge the receptive field by downsampling the internal feature map to half size and then upsampling it back by changing the stride of the third convolution layer to 2 and third-to-last convolution layer to deconvolution with stride 2 and kernel size 4×4. In this way, the receptive field is effectively enlarged without losing too much image detail, and meanwhile the computation cost is halved. For better performance and faster convergence, we implement the middle 26 convolution layers as 13 residual units [14] similar to [5].

Finally, to resolve the color attenuation issue [16, 17] observed in deep networks, we slightly magnify the predicted image $\mathbf{I}^t$ via $s_c \triangleq \arg\min_{s_c} \|\mathbf{I}_c^s - s_c \cdot \mathbf{I}_c^t\|_2^2$ and $\mathbf{I}_c^t \leftarrow s_c \cdot \mathbf{I}_c^t$. This global color correction is implemented as a parameter-free layer after I-CNN. Its computational cost is negligible.

4. Network Training

This section first presents our training pipeline, that applies independently of the data source. Later we describe application-specific means of generating training samples.

4.1. Training Details

We employ a two-phase network training algorithm shown in Fig. 2. Specifically, we first train the sub-networks separately with ground-truth images and their edge maps to ensure the best individual performances. We then fine-tune the entire network end-to-end, granting the two sub-nets more opportunity to cooperate accordingly.

The sub-nets are trained by minimizing the mean squared errors (MSE) of their predictions. Let the symbol $*$ denote ground truth, the loss for edge prediction is

$$l_E(\theta) = ||\mathbf{E}^t - \mathbf{E}^{t*}||_2^2. \tag{4}$$

For image prediction, we minimize not only the color MSE but also the discrepancy of gradients:

$$\begin{aligned} l_I(\theta) &= \alpha\, ||\mathbf{I}^t - \mathbf{I}^{t*}||_2^2 \\ &+\beta\, (||\nabla_x \mathbf{I}^t - \nabla_x \mathbf{I}^{t*}||_1 + ||\nabla_y \mathbf{I}^t - \nabla_y \mathbf{I}^{t*}||_1). \end{aligned} \tag{5}$$

The gradient discrepancy cost, though seemingly redundant, helps to prevent the deep convolutional network from generating blurry images [25]. In the joint training phase, we train the entire network by minimizing the loss:

$$l(\theta) = l_I(\theta) + \gamma\, l_E(\theta). \tag{6}$$

For all experiments across reflection removal and image smoothing, the loss coefficients are empirically set as $\alpha = 0.2,\ \beta = \gamma = 0.4$ (other selections produce similar results).

We initialize the convolution weights using the approach from [13] and train all networks using ADAM [18] with mini-batch size fixed at 1. When training the two sub-nets separately, the learning rate is set to 0.01 over the initial iterations, *e.g.*, 40 and 25 epochs for reflection and imaging smoothing tasks respectively. The entire network is then fine-tuned with the learning rate reduced to 0.001.

4.2. Training Data Generation

Reflection Image Synthesis: Real images with ground truth background layers are difficult to obtain. To generate enough training data, simply mixing two images with different coefficients (such as 0.8 for background and 0.2 for reflection) seems to be a straightforward and plausible compromise. Indeed, this strategy has been widely used in previous works [34, 29, 12, 23, 41] for analysis and quantitative evaluation. However, we found that networks trained on such images generalize poorly to real photographs. We therefore propose a novel synthesis method to better approximate real-world reflection.

As previously mentioned, we assume that the reflection is somewhat blurry relative to the background layer, which tends to be more sharp and clear. This is a valid assumption for many cases, as the camera is usually focused on the background target. Moreover, a photographer can easily widen the camera's aperture and blur out the reflections. A similar assumption is used by [23].

We expand on this assumption using a simple complementary observation. First, according to the Fresnel equation, we know that when incident light travels across media with different refractive indices (*e.g.*, glass and air) in front of some scene of interest, a portion of that light will be reflected back to the image plane. However, the actual visibility of this reflected light to the human eye or a camera depends on the relative intensity of light transmitted from the background scene. Therefore we may expect that only portions of the background layer transmitting modest light will be appreciably obstructed via a reflection layer, even if the latter is uniformly present across a scene. And yet in regions where reflections are apparent, their intensity can still

Randomly pick two natural images normalized to $[0, 1]$ as background $\mathbf{B}$ and reflection $\mathbf{R}$ respectively, then:

1. $\tilde{\mathbf{R}} \leftarrow gauss\_blur_{\sigma}(\mathbf{R})$ with $\sigma \sim \mathcal{U}(2, 5)$
2. $\mathbf{I} \leftarrow \mathbf{B} + \tilde{\mathbf{R}}$
3. $m \leftarrow mean(\{\mathbf{I}(\mathbf{x}, c) \mid \mathbf{I}(\mathbf{x}, c) > 1, \forall \mathbf{x}, \forall c = 1,2,3\})$
4. $\tilde{\mathbf{R}}(\mathbf{x}, c) \leftarrow \tilde{\mathbf{R}}(\mathbf{x}, c) - \gamma \cdot (m-1), \forall \mathbf{x}, \forall c$; γ set as 1.3
5. $\tilde{\mathbf{R}} \leftarrow clip_{[0,1]}(\tilde{\mathbf{R}})$
6. $\mathbf{I} \leftarrow clip_{[0,1]}(\mathbf{B} + \tilde{\mathbf{R}})$

Output $\mathbf{I}$ as the synthesized image with $\mathbf{B}$ as the ground-truth background layer.

Figure 3. Reflection image data synthesis for weakly-supervised learning. The subtraction and clipping operators allow for reflection intensities that can saturate and vanish in various regions.

be arbitrarily large (even if partially blurred) and so a purely additive model with a weakly scaled reflection component is not always physically plausible.

Based on the above observations, we develop a new method summarized in Fig. 3 to synthesize images with realistic background and reflection layers. One key difference from naive image mixing is that the brightness overflow issue is avoided not by scaling down the brightness, but by subtracting an adaptively computed value followed by clipping. In this way: (*i*) reflection-free regions are very likely to appear which is consistent with natural images, (*ii*) strong reflections can occur in other places, and (*iii*) the reflection contrast is better maintained. Also note that we randomly pick the σ of the Gaussian blur kernel between $[2, 5]$, in contrast to a fixed large value ($\sigma = 5$) tested in [23]. We are interested in handling a wider range of real cases, including cases with lesser blurry reflections. Figure 6 (top) displays 4 synthetic images generated by our method, and Fig. 5 shows a result comparison with naive image mixing. For more comparisons and details regarding the synthesis process, see the *supplemental material*.

Note that synthetically generated samples serve as a form of weak supervision, as we ultimately deploy the trained model on new real images containing natural reflections.

Generation of Smoothed Images: For image smoothing, our network is trained to approximate the effect of existing filters. The training and testing data will simply be the smoothed images generated by applying those filters to existing image databases. Various filters are tested in Sec. 5.

5. Experiments

This section first presents self-comparison experiments to analyze the importance of proposed network architecture design choices. We then evaluate the full CEILNet against the state-of-the-art algorithms on the single-image reflection removal and image smoothing tasks.

Table 1. Result comparison for the image smoothing task (learning an L_0 filter [37]). CEILNet outperformed Domain Transform (DT) [10] and simple I-CNNs without E-CNN by large margins.

	MSE	PSNR	SSIM
DT + input image edge	124.41	27.38	0.806
DT + pred. edge by E-CNN	51.26	31.17	0.964
DT + GT edge	45.67	31.66	0.971
I-CNN only	37.79	32.58	0.969
I-CNN only (64 layers)	31.86	33.33	0.973
I-CNN with input edge (64 layers)	22.50	34.86	0.979
CEILNet	**13.34**	**37.10**	**0.989**

5.1. Network Analysis

For simplicity, our analysis will be mainly based on the representative results of approximating L_0 smoothing [37]. These results were obtained on 100 PASCAL VOC test images (refer to Sec. 5.3 for training and testing details).

Is the target edge map from E-CNN helpful? To verify the importance of the target edge map for image reconstruction, we removed E-CNN and trained a simple I-CNN model without the predicted target edge or replacing the predicted target edge with the input image edge. Table 1 shows that I-CNN with predicted edge (*i.e.*, our CEILNet) outperformed I-CNN alone and I-CNN with input edge by significant margins, demonstrating the importance of target edge prediction. A visual comparison is shown in Fig. 4.

Similar results were obtained for reflection removal: the predicted background edges were found to be helpful for layer separation. Figure 5 shows a typical example.

Does simply stacking more layers in I-CNN suffice? Ideally, with enough depth, one may expect the network to handle target edge prediction implicitly without the need for an explicit E-CNN. We tried training a simple I-CNN with more convolutional layers. and found that the performance gets saturated quickly after more than 50 layers (a detailed figure is deferred to the supplementary material). Our CEILNet, *i.e.*, 32-layer E-CNN + 32-layer I-CNN, achieved much better results than a 64-layer simple I-CNN (as shown in Table 1) and a best-performing 70-layer one (PSNR 33.37 *vs*. 37.10 by CEILNet).

Is I-CNN better than a traditional method? To answer this question, we replaced I-CNN with the Domain Transform (DT) technique [10]. The predicted target edge map by E-CNN and the input image are fed to DT to output smooth images. We also tried the ground-truth target edge and the input image edges. Table 1 shows that I-CNN with predicted edge from E-CNN (*i.e.*, our CEILNet) outperformed all DT results by large margins. A visual comparison is presented in Fig. 4.

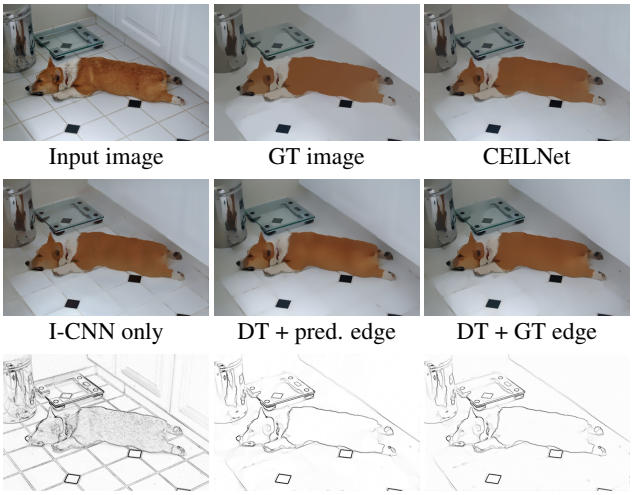

Figure 4. Qualitative comparison for the image smoothing task (learning an L_0 filter [37]). Our CEILNet generates a more satisfactory result than a simple I-CNN without E-CNN and than Domain Transform [10]. **Best viewed on screen with zoom.**

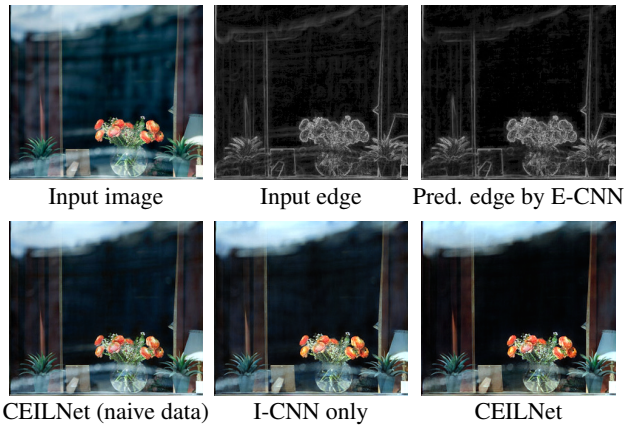

Figure 5. Qualitative reflection removal results on a real image. Our CEILNet removes more reflection and generates a clearer background image than a simple I-CNN without E-CNN, and than CEILNet trained with a naive image mixing strategy for data generation. **Best viewed on screen with zoom.**

For reflection removal, we also tried applying the layer separation algorithm in [22] with our predicted edges as input, but no satisfactory results were obtained.[1]

5.2. Reflection Removal

Training Data: We applied the method described in Sec. 4.2 to synthesize training data for the reflection removal task to accommodate our weakly-supervised learning pipeline. We used 17K natural images from the PASCAL VOC dataset [4] for the synthesis. These images were collected from Flickr, and represent a wide range of viewing conditions. Two natural images were used to generate one synthetic image containing a background layer and a reflection layer, resulting in 8.5K synthetic images in total. We split these images into a training set of 7,643 images and a test set with 850 images for quantitative comparison. The training images are also cropped to 224×224. The algorithm described in Fig. 2 was then applied, and we did not observe over-fitting in any of the training sub-tasks.

[1] [22] utilizes multiple images to identify background edges, which are used as prior to guide layer septation. Their septation algorithm did not work well with our edge maps as it assumes non-blurry reflections and requires binary edge maps.

Table 2. Quantitative comparison of our method with Li and Brown [23] on 100 synthetic images with reflection.

PSNR		SSIM	
[23]	Ours	[23]	Ours
15.50	**18.55**	0.786	**0.857**

Method Comparison: We tested our CEILNet against the state-of-the-art, single-image approach from [23]. For a quantitative comparison, we randomly selected 100 images in our test dataset, and evaluate the PSNR and SSIM metrics for the predicted $\mathbf{B}$ from both algorithms. The default parameters of [23] were used for evaluation. Table 2 shows that CEILNet significantly outperformed [23].

Figure 6 presents some qualitative results of our method compared against [23] on both synthetic and real images. The reflection image estimates are computed via $\mathbf{R} = \mathbf{I} - \mathbf{B}$. We tuned the parameters of [23] for each image to get the best visual result. It can be seen that [23] tends to generate a blurry reflection layer with brightness covering the whole image. It largely failed to remove less blurry, high contrast or partially present reflections. This is because [23] employs strong priors to penalize abrupt color transitions in $\mathbf{R}$ which, however, may be common in real cases. In contrast, our CEILNet is able to separate out the reflections reasonably well even if some of them are very bright and shiny, and without jeopardizing the reflection-free regions. More results and comparisons are deferred to the *supplementary material* due to space limitation.

5.3. Image Smoothing

Training Data: For image smoothing, we used the 17K natural images in the PASCAL VOC dataset as input, and generated the filtered images using existing image smoothing algorithms as the ground truth. These images are fed to the network without cropping. We also randomly pick 100 images in the PASCAL VOC dataset for testing. We again use the algorithm in Fig. 2 to train our CEILNet.

Method Comparison: We tested 8 image smoothing algorithms for the network to approximate, including bilateral filter (BLF) [26], iterative bilateral filter (IBLF) [8], rolling guidance filter (RGF) [42], RTV texture smoothing

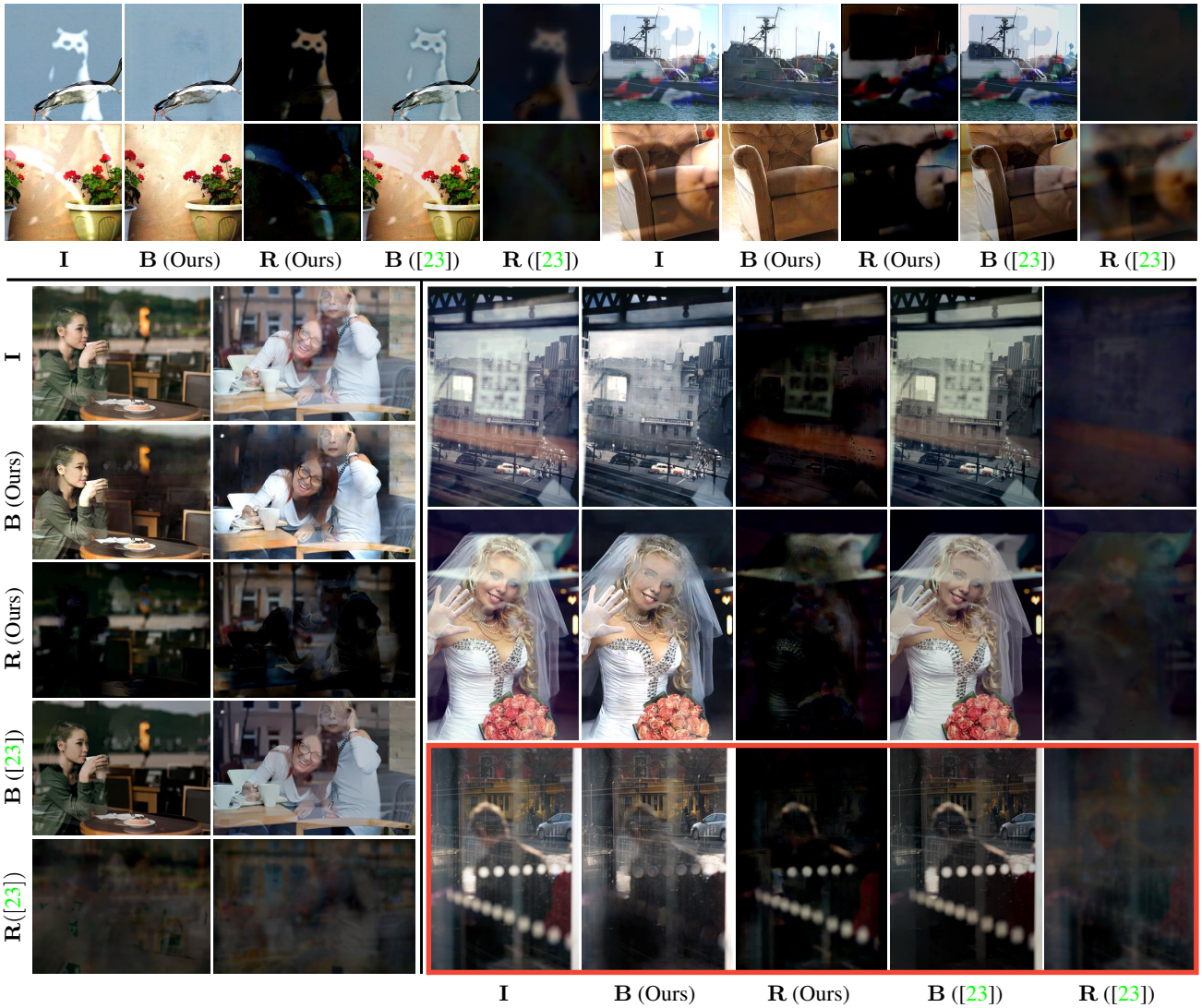

Figure 6. Qualitative results of the single image reflection removal task on synthetic (*top two rows*) and real (*bottom rows*) images. Visually inspected, our method can largely remove the reflection and produce reasonably good background images under various situations. The method of Li and Brown [23] clearly underperformed. The last example is a partial failure case for our method due to the strong reflection and weak transmitted light, but still the result is superior to [23]. **Best viewed on screen with zoom.**

Table 3. Quantitative comparison on the image smoothing tasks. We report the PSNR and SSIM metrics (larger is better) for 8 different smoothing filters, and compare our method with Xu *et al*. [38]. Average values are computed with the preceding 7 cases.

		BLF	IBLF	L_0	RGF	RTV	WLS	WMF	L_1	Ave.
PSNR	[38]	35.02	32.97	31.66	32.49	35.68	33.92	29.62		32.62
	Ours	**43.76**	**38.18**	**37.10**	**42.05**	**44.03**	**41.39**	**39.70**	**36.99**	**40.40**
SSIM	[38]	0.976	0.962	0.966	0.950	0.974	0.963	0.960		0.964
	Ours	**0.995**	**0.989**	**0.989**	**0.991**	**0.994**	**0.994**	**0.989**	**0.982**	**0.990**

Table 4. Running time comparison (in seconds). We compare the running time of our method against different traditional methods as well as deep learning based methods of Xu *et al*. [38] and Liu *et al*. [24] at various resolutions.

	BLF	IBLF	RGF	L_0	WMF	RTV	WLS	L_1	[38]	[24]	Ours
QVGA (320×240)	0.03	0.11	0.22	0.17	0.62	0.41	0.70	32.18	0.23	0.07	0.03
VGA (640×480)	0.12	0.40	0.73	0.66	2.18	1.80	3.34	212.07	0.76	0.14	0.12
720p (1280×720)	0.34	0.97	1.87	2.43	4.98	5.74	13.26	904.36	2.16	0.33	0.35

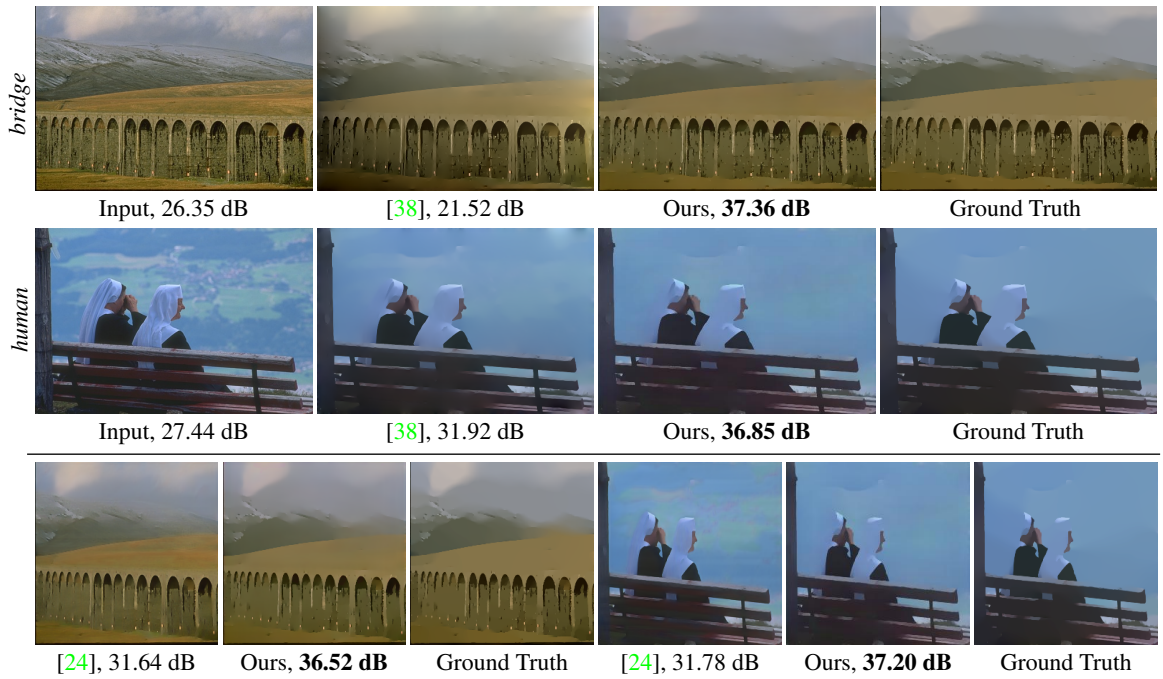

Figure 7. Qualitative results on the image smoothing task. All the methods are trained to approximate L_0 smoothing [37]. Top: Comparison with Xu *et al*. [38]. Bottom: Comparison with Liu *et al*. [24] on the 256×256 image size. Our results are visually much closer to the ground truth. The numbers show the PSNR values. **Best viewed on screen with zoom.**

Table 5. Comparison with Liu *et al*. [24] on image smoothing.

	PSNR		SSIM	
	[24]	Ours	[24]	Ours
L_0	32.26	**36.62**	0.958	**0.986**
RGF	38.64	**40.80**	0.986	**0.989**
WLS	38.29	**40.27**	0.983	**0.992**
WMF	33.29	**37.75**	0.951	**0.986**
Ave.	35.64	**39.36**	0.966	**0.988**

(RTV) [39], weighted least square smoothing (WLS) [6], weighted median filter (WMF) [43], L_0 smoothing [37] and L_1 smoothing [2].

Table 3 presents the quantitative results of our method and [38] on the test set with 100 images. In can been seen that our network achieved much better results than [38] for all the 8 filters, on both the PSNR and SSIM metrics. We also compare our results with [24], whose models for 4 filters are publicly available. Note that at the time of writing, the latest code of [24] released by their authors cannot run on arbitrary image size due to some implementation constraints, so we use their default size of 256×256. Table 5 shows that our method also significantly outperformed [24] for all the 4 filtering algorithms.

Figure 7 presents two visual results of our method compared to others. It can be observed that the method of [24] generated obvious artifacts compared to the ground truth for both two cases, while [38] produced some unwanted color transitions in the right and bottom left regions of the "bridge" image, resulting in a PSNR even lower than the raw input image. In contrast, our results are visually more close to the ground truth. More results and discussions can be found in the *supplementary material*.

Running Time: We evaluate the running time of the eight traditional smoothing algorithms and the three deep learning based methods with respect to different image sizes on the same computer (NVIDA DGX-1). Table 4 shows that our method runs faster than others in most of the cases. It can approximate any traditional algorithm at over 8 fps for 640×480 images.

6. Conclusions and Future Work

We have proposed CEILNet, a generic deep architecture for edge-sensitive image processing. We provided the first learning-based solution to the challenging single image reflection removal problem using CEILNet and with the aid of a novel reflection image synthesis method. We have also significantly advanced the state-of-the-art in DNN-based image smoothing. Our future work includes testing CEILNet on more image processing tasks. Promising results for image denosing and inpainting have been obtained in our preliminary experiments.

Acknowledgement This work was partially supported by National 973 Program (2015CB352501), Shenzhen Innovation Program (JCYJ20150402105524053).

References

[1] A. Agrawal, R. Raskar, S. K. Nayar, and Y. Li. Removing photography artifacts using gradient projection and flash-exposure sampling. *ACM Transactions on Graphics (TOG)*, 24(3):828–835, 2005. 1, 2

[2] S. Bi, X. Han, and Y. Yu. An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics (TOG)*, 34(4):78, 2015. 1, 8

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, (6):679–698, 1986. 3

[4] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010. 6

[5] Q. Fan, D. Wipf, G. Hua, and B. Chen. Revisiting deep image smoothing and intrinsic image decomposition. *arXiv preprint. arXiv:1701.02965*, 2017. 4

[6] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics (TOG)*, 27(3), 2008. 1, 8

[7] H. Farid and E. H. Adelson. Separating reflections and lighting using independent components analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 262–267, 1999. 1, 2

[8] R. Fattal, M. Agrawala, and S. Rusinkiewicz. Multiscale shape and detail enhancement from multi-light image collections. *ACM Transactions on Graphics (TOG)*, 26(3), 2007. 1, 6

[9] K. Gai, Z. Shi, and C. Zhang. Blind separation of superimposed moving images using image statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 34(1):19–32, 2012. 2

[10] E. S. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. In *ACM Transactions on Graphics (TOG)*, volume 30, page 69. ACM, 2011. 3, 5, 6

[11] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics (TOG)*, 35(6):191, 2016. 1

[12] X. Guo, X. Cao, and Y. Ma. Robust separation of reflection from multiple images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2187–2194, 2014. 1, 2, 4

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015. 4

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 4

[15] S. H. Khan, M. Bennamoun, F. Sohel, and R. Togneri. Automatic shadow detection and removal from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 38(3):431–446, 2016. 1

[16] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, 2016. 4

[17] J. Kim, J. Kwon Lee, and K. Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1637–1645, 2016. 4

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015. 4

[19] N. Kong, Y.-W. Tai, and J. S. Shin. A physically-based approach to reflection separation: from physical modeling to constrained optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 36(2):209–221, 2014. 1, 2

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 1

[21] A. Levin and Y. Weiss. User assisted separation of reflections from a single image using a sparsity prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 29(9), 2007. 1, 2, 3

[22] Y. Li and M. S. Brown. Exploiting reflection change for automatic reflection removal. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2432–2439, 2013. 1, 2, 3, 6

[23] Y. Li and M. S. Brown. Single image layer separation using relative smoothness. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2752–2759, 2014. 2, 4, 5, 6, 7

[24] S. Liu, J. Pan, and M.-H. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 2, 7, 8

[25] T. Narihira, M. Maire, and S. X. Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *IEEE International Conference on Computer Vision (CVPR)*, pages 2992–2992, 2015. 4

[26] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *European Conference on Computer Vision (ECCV)*, pages 568–580, 2006. 1, 6

[27] J. S. Ren and L. Xu. On vectorization of deep convolutional neural networks for vision tasks. *AAAI Conference on Artificial Intelligence*, 2015. 1

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1

[29] B. Sarel and M. Irani. Separating transparent layers through layer information exchange. In *European Conference on Computer Vision (ECCV)*, pages 328–341, 2004. 2, 4

[30] Y. Y. Schechner, N. Kiryati, and R. Basri. Separation of transparent layers using focus. *International Journal of Computer Vision (IJCV)*, 39(1):25–39, 2000. 1, 2

[31] Y. Y. Schechner, J. Shamir, and N. Kiryati. Polarization and statistical analysis of scenes containing a semireflector. *JOSA A*, 17(2):276–284, 2000. 1, 2

[32] Y. Shih, D. Krishnan, F. Durand, and W. T. Freeman. Reflection removal using ghosting cues. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3193–3201, 2015. 2

[33] S. N. Sinha, J. Kopf, M. Goesele, D. Scharstein, and R. Szeliski. Image-based rendering for scenes with reflections. *ACM Transactions on Graphics (TOG)*, 31(4):100–1, 2012. 2

[34] R. Szeliski, S. Avidan, and P. Anandan. Layer extraction from multiple images containing reflections and transparency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 246–253, 2000. 2, 4

[35] R. Wan, B. Shi, T. A. Hwee, and A. C. Kot. Depth of field guided reflection removal. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 21–25. IEEE, 2016. 2

[36] S. Xie and Z. Tu. Holistically-nested edge detection. In *International Conference on Computer Vision (ICCV)*, pages 1395–1403, 2015. 3

[37] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via L_0 gradient minimization. In *ACM Transactions on Graphics (TOG)*, volume 30, page 174, 2011. 1, 5, 6, 8

[38] L. Xu, J. S. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *International Conference on Machine Learning (ICML)*, pages 1669–1678, 2015. 1, 2, 3, 7, 8

[39] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via natural variation measure. *ACM Transactions on Graphics (TOG)*, 2012. 1, 8

[40] T. Xue, M. Rubinstein, C. Liu, and W. T. Freeman. A computational approach for obstruction-free photography. *ACM Transactions on Graphics (TOG)*, 34(4):79, 2015. 1, 2, 3

[41] J. Yang, H. Li, Y. Dai, and R. T. Tan. Robust optical flow estimation of double-layer images under transparency or reflection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1410–1419, 2016. 1, 2, 4

[42] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *European Conference on Computer Vision (ECCV)*, pages 815–830, 2014. 1, 6

[43] Q. Zhang, L. Xu, and J. Jia. 100+ times faster weighted median filter. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 1, 8

Learning to Smooth Images

QINGNAN FAN, Shandong University
JIAOLONG YANG, Microsoft Researchs
DAVID WIPF, Microsoft Research
BAOQUAN CHEN, Shandong University
XIN TONG, Microsoft Research

Fig. 1. By minimizing the proposed spatially adaptive L_p norm flattening criterion and edge-preserving criterion, we are able to implement various completely different image smoothing effects in the same framework. Two edge-preserving smoothing examples are demonstrated in (a)-(d), two examples of blurring background out of saliency are shown in (e)-(h), and one texture removal example is displayed in (i)-(j).

Image smoothing is a basic and fundamental image processing tool. A variety of smoothing effects are developed for different applications, such as edge-preserving smoothing, texture removal, *etc.* In this paper, we implement multiple distinct image smoothing effects within a single framework. To accomplish that, we propose the spatially adaptive L_p norm flattening criterion, which applies dissimilar sparsity-inducing regularization in separate image regions according to user-desired edge map. To maintain some subtle edge information which are important but vulnerable to miss, we present an edge-preserving criterion that explicitly emphasizes the valuable image structures. Guided by different kinds of specified edge map, we are able to realize various smoothing effects via optimizing the proposed criterions. Instead of leveraging traditional numerical solver to optimize the above objective function, we introduce an efficient fully-convolutional deep neural network, which learns to generate the required smooth image and does not require the guided edge map for evaluation. Our deep learning based solver runs much faster than most image smoothing algorithms. We demonstrate the effectiveness and versatility of our learned image smoother on multiple image processing tasks, such as image abstraction, pencil sketching, detail enhancement, texture smoothing, background blur out of saliency *etc.*

2017. XXXX-XXXX/2017/10-ART1 $15.00
https://doi.org/0000001.0000001_2

ACM Reference format:
Qingnan Fan, Jiaolong Yang, David Wipf, Baoquan Chen, and Xin Tong. 2017. Learning to Smooth Images. 1, 1, Article 1 (October 2017), 12 pages.
https://doi.org/0000001.0000001_2

1 INTRODUCTION

Image smoothing is an essential tool to implement multiple different image processing and computer graphics applications. For example, edge preserving smoothing is required to generate some stylization effects like image abstraction, pencil sketching, *etc.* By extracting the detail layer from the input and edge-preserving smoothing image, the subtle image details can be enhanced for better visualization. Specialized smoothing measure [Xu et al. 2012] and framework [Zhang et al. 2014] are developed to remove the small-scale textures while maintaining the visually important image structures. More recently, the bokeh effects generated by the large aperture with professional lens has been approximated on the smart phones. Some famous app like Snapseed and smart phone producers like Huawei and Oppo have implemented similar functions to enable beautiful background blur effects. All the different applications demand specific and distinct design of the algorithm, for example, texture

removal requires to eliminate the distractive repetitive small-scale structures where color transition may be even significant, however edge-preserving smoothing will maintain such edges.

Various different smoothing effects can be implemented by minimizing the flattening criterion equipped with distinct L_p norm. For example, L_p norm (0<p<1) performs very well in achieving strong smoothing effects, but tends to create staircasing artifacts within moderate smooth regions by breaking them up into piecewise constant regions separated by spurious edges, while L_p norm (p=2) is able to generate blurry images and wipe out the edges.

To implement all the drastically different smoothing effects within a single framework, we propose a spatially variant L_p norm flattening criterion, which applies $L_{0.5}$ and L_2 norm in distinct image regions in accordance to a user specified edge map. For the example of edge-preserving image smoothing, $L_{0.5}$ norm is utilized among the whole image except for the over-sharpened patchy regions, which are covered by L_2 norm to blur the spurious edges.

As $L_{0.5}$ norm is employed, the smooth image becomes sharp, and the edges with conspicuous color transition are easily maintained well. By minimizing the flattening criterion, the nearby pixels with similar colors tends to be put even closer. Some slender edges whose two sides share similar color values tend to be over smoothed, but they still could be semantically important. To overcome the challenges, we present an edge-preserving criterion to explicitly highlight the desired valuable structures guided by the same edge map utilized in the flattening criterion.

To transform between different smoothing applications within the same framework, we only need to adjust the feeded edge map which indicates various smoothing effects in distinct regions.

Instead of leveraging traditional numerical solver to optimize the objective function, in this paper, we propose to employ an efficient fully-convolutional deep neural network which learns to minimize the proposed criterions. Note the benefits of our deep learning based solver lies in three aspects. *(a)*, the deep network is learned as an end-to-end mapping which characterizes the smoothing effects. It requires only the input image for evaluation since the guided edge map is incorporated into the criterions for only training. *(b)*, our deep learning based solver runs much faster than the other optimization based smoothing strategies, as only one forward pass is needed for testing instead of multiple iterations in traditional methods. *(c)*, deep learning provides a new pathway to optimize the objective functions which are too complex to find an efficient and effective traditional solver. The details of optimization process is represented in the deep network.

All in all, the contribution of this paper lies as follows,

- We introduce a new image smoothing measure, built on the mixture of a spatially adaptive L_p norm flattening criterion and an edge-preserving criterion guided by the same edge map. By minimizing the proposed criterions, our algorithm is able to achieve state-of-the-art various different image smoothing effects.
- We are the first to implement multiple different image smoothing effects within a single framework, such as edge-preserving smoothing, texture removal, background blur out of saliency, *etc.* The only variable responsible for the distinct smoothing effects is the edge map, which can be detected or even user designed.
- We propose a new deep learning based solution pipeline to the above image smoothing criterion. The optimization process is learned into the deep network which is represented as an end-to-end mapping for evaluation. The proposed solver is very efficient and doesn't require the guided edge map in the test stage.

2 RELATED WORK

During the last three decades, many image smoothing algorithms have been developed to accomplish different low-level vision and computer graphics problems, which varies from the classical filtering based method like anisotropic diffusion [Perona and Malik 1990] and bilateral filter [Tomasi 1998], to recent popular optimization based method like [Liu et al. 2017; Xu et al. 2011].

Filtering based approaches [Chen et al. 2007; Fattal 2009a; Gastal and Oliveira 2011; Kass and Solomon 2010; Paris and Durand 2006; Perona and Malik 1990; Weiss 2006] process the image in a local manner, and each pixel is determined by its spatial neighbourhood. One representative of these approaches is bilateral filter [Tomasi 1998], which replaces each pixel in the filtered image with a weighted mean of its surrounding pixels, with the weight sampled from a Gaussian distribution computed based on both spatial and color affinity. Bilateral filter has been widely applied in plenty of applications, such as image upsampling [Kopf et al. 2007], flash/non-flash image filtering [Petschnigg et al. 2004], tone mapping [Durand and Dorsey 2002], *etc.* Local filtering based methods are usually efficient, however it may produce gradient reversals and halo artifacts [Farbman et al. 2008].

The optimization based approaches usually define an energy function which contains a data constraint term that minimizes the pixel-wise difference between input and output images for seeking structure similarity, and a prior smoothness term that minimizes color difference among nearby pixels within the output image for pursuing strong flattening effects. To optimize such an objective function, a traditional numerical solver is usually utilized, for example, iterative reweighted least square (IRLS) is well-known for minimizing energy functions in non-quadratic forms such as L_p norm (0<p<1) [Min et al. 2014] and L_1 norm [Farbman et al. 2008]. Since numerical solvers are more robust, flexible, and are able to achieve the optimal smoothing result in a principled global manner, a variety of these papers have arisen in recent years.

[Farbman et al. 2008] advocates an edge-preserving operator by implementing a weighted least square optimization framework. This is designed for progressive coarsening of images and multi-scale detail extraction. [Min et al. 2014] follows a similar WLS scheme, but presents an efficient solution to the large linear systems by iteratively solving a sequence of one dimensional subsystems. Lately, [Liu et al. 2017] extends the previous work [Min et al. 2014] further by taking more general and larger two-dimensional neighborhood information into account, which achieves closer performance to the original WLS model, but are more time and memory efficient.

[Xu et al. 2011] minimizes the L_0 gradients, which globally controls a sparse set of gradient to approximate the prominent image

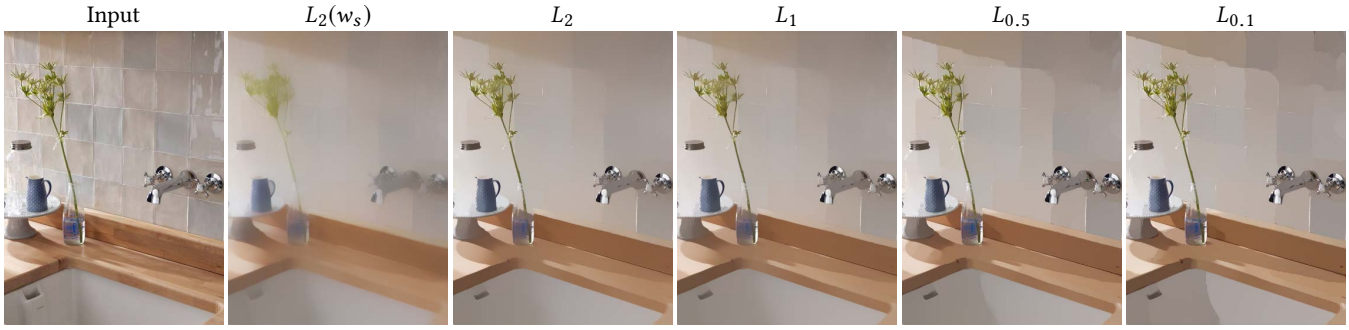

Fig. 2. Smooth images generated by different sparsity-inducing norms using the basic criterion and deep learning based solver described in our paper. While computing the gaussian weight in the basic flattening term, spatial affinity (w_s) is specified for the second image, while color affinity is used for the other smooth images. As the norm becomes smaller, the learned image becomes smoother, sharper and even more piece-wise constant, however there appears some spurious edges which are over sharpened. On the contrary, as the norm becomes larger, the image is less sharp and more blurry. Using spatial affinity in the gaussian weight shows stronger blurry effects.

structure. This approach is particularly effective in sharpening primary edges while eliminating a manageable degree of low-amplitude details. However, [Xu et al. 2011] tends to overlook or blur some important low-contrast edges due to its binary classification of the image gradients according to some global parameters.

Besides the edge-preserving smoothing approaches which tends to maintain all salient image structures, [Xu et al. 2012] presents specifically designed inherent variation and relative total variation measures optimized by IRLS to extract meaningful structure from over textured surfaces. Later on, [Zhang et al. 2014] introduces an effective scale-aware rolling guidance filter in order to preserve large-scale structures optimally.

Note most optimization based approaches are very time-intensive, since the solution to the objective function typically needs to solve a large linear system. Therefore, recently there have emerged a number of papers [Fan et al. 2017; Liu et al. 2016; Xu et al. 2015] dedicated to speeding up existing smoothing operators. They propose to learn a deep neural network by supervising the ground truth smooth images generated by previous methods, while our smoothing effects are learned by optimizing an objective function through a deep network in a specified unsupervised manner, which is completely different from theirs. Moreover, their algorithms fail to achieve satisfactory smoothing effects on some complex edge-aware smoothing algorithms such as [Xu et al. 2011] and [Bi et al. 2015].

Except for the classical image smoothing tasks devoted to eliminating the insignificant details among the whole image while preserving the primary structures, the scope of image smoothing can be spread out to a much larger domain. For example, [Shen et al. 2016] introduces an automatic portrait segmentation algorithm, which are post-processed to achieve some stylization effects. For example, blurring background outside of portrait to simulate the bokeh effect generated by a large aperture with professional lens. In this paper, we work on a much wider scope. We smooth the background outside the detected salient areas in the image. However, unlike previous method to operate in several consecutive independent steps, our algorithm incorporates the detected saliency information into the guided edge map, and implement the background blur effects into a single deep network. Similar background blur effects have also been implemented into many smart phones like Huawei and Oppo.

3 APPROACH

In this section, at first we introduce the proposed criterion we are minimizing to achieve image smoothing effects, which flatten image areas with soft color transitions while enforce structure similarity between the input and smooth image. Afterwards, we present our deep learning based solver to optimize the proposed criterion.

3.1 Basic Criterion

To enhance the high contrast edges and diminish insignificant details, our criterion contains a flattening term for regularization. In 2D image representation, we denote the input image by I, the output image by T. The input image I is transformed to YUV color space Y to compute the gaussian weight based on color affinity. The basic flattening term in the current position is written as follows,

$$\mathcal{E}_f = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_h(i)} w_{i,j}\cdot|T_i - T_j|^p \tag{1}$$

$$w_{i,j,r} = exp(-\frac{\sum_c (Y_{i,c} - Y_{j,c})^2}{2\sigma_r^2}) \tag{2}$$

$$w_{i,j,s} = exp(-\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{2\sigma_s^2}) \tag{3}$$

where σ_r and σ_s controls the color and spatial scale among the $h \times h$ nearby points $\mathcal{N}_h(i)$, c indicates the different color channels of Y, and N refers to the number of all the points among the three color channels in the image. Note the weight $w_{i,j}$ in Eq. 1 can be computed based on either color affinity $w_{i,j,r}$ or spatial affinity $w_{i,j,s}$. By minimizing the flattening criterion, neighbouring pixels which are close in color or spatial distance are pulled even closer to achieve smoothing effects.

Apart from the flattening term to remove unimportant details, we also employ the data term D_i to seek the structure similarity

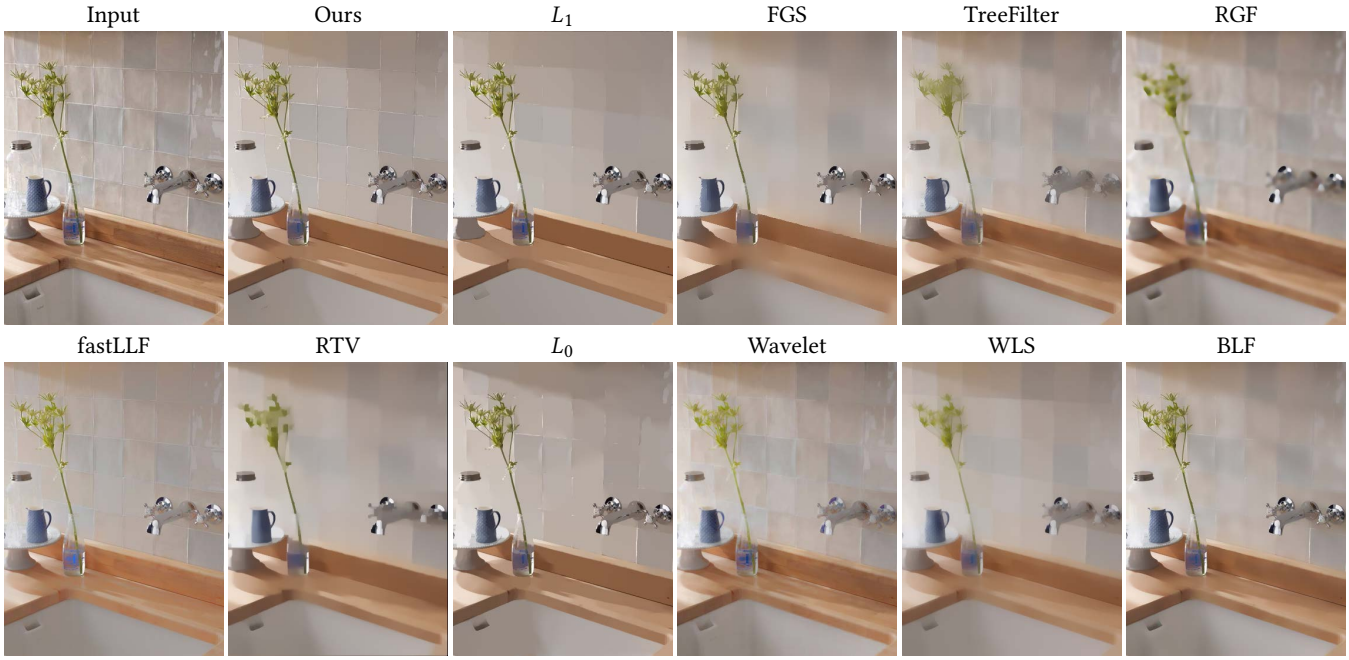

Fig. 3. Visual comparison between our deep learning based approach and traditional edge-preserving smoothing methods, for example, L_1 [Bi et al. 2015], FGS [Min et al. 2014], TreeFilter [Bao et al. 2014], RGF [Zhang et al. 2014], fastLLF [M. Aubry and Durand 2014], RTV [Xu et al. 2012], L_0 [Xu et al. 2011], Wavelet [Fattal 2009b], WLS [Farbman et al. 2008], BLF [Tomasi 1998].

from the input image, which is as follows,

$$\mathcal{E}_d = \frac{1}{N}\sum_{i=1}^{N} ||T_i - I_i||_2^2 \tag{4}$$

Therefore, the complete criterion for minimization is the weighted sum of these two terms,

$$\mathcal{E}_d + \lambda \cdot \mathcal{E}_f \tag{5}$$

where λ is a constant value to balance between the structure similarity and smoothness.

Similar objective functions have been introduced in many optimization based image smoothing papers [Bi et al. 2015; Farbman et al. 2008; Min et al. 2014; Rudin et al. 1992; Xu et al. 2011, 2012]. The main difference among these papers lies in the choice of designing the flattening term, for example, whether the surrounding pixels for computing weights are adjacent [Farbman et al. 2008; Min et al. 2014; Rudin et al. 1992; Xu et al. 2011, 2012] or in a larger neighbourhood [Bi et al. 2015; Liu et al. 2017], whether the weight is calculated based on color affinity [Bi et al. 2015; Farbman et al. 2008; Min et al. 2014], spatial affinity [Xu et al. 2012] or either both [Liu et al. 2017], whether the sparsity-inducing norm is L_2 [Farbman et al. 2008; Liu et al. 2017; Min et al. 2014], L_1 [Bi et al. 2015; Rudin et al. 1992; Xu et al. 2012] or L_0 [Xu et al. 2011].

Among so many possibilities in the flattening criterion, the sparsity-inducing norm plays a critical role in controlling the smoothness and sharpness effects. As shown in Figure 2, while the norm becomes smaller like $L_{0.1}$, the optimized image becomes sharper, smoother and even more piece-wise constant. And while the norm becomes larger like L_2, the image looks more blurry. By taking spatial affinity into consideration while computing the gaussian weight, the edges are mostly blurred.

We observe that not all the edges in the smooth image really matters to each specific application. For example, to pursue stronger edge-preserving smoothing effects with $L_{0.5}$ and $L_{0.1}$ norm, the output images are over sharpened and produce some spurious edges, *e.g.*, on the bottom of the smooth image some edges are incorrectly enhanced or even created while they do not exist in the input image. And in another application of texture smoothing, all the edges on the repetitive small-scale texture surfaces should be blurred while only meaningful structures should be preserved, which cannot be accomplished by the usual edge-preserving smoothing algorithms.

Moreover, some edges, such as the ones representing ceramic tiles with different colors on the wall, are very thin but semantically important. Among all the smooth images, such edges are mostly diminished, which make the different ceramic tiles less discriminative from each other. Based on the above observations, our criterions are designed.

3.2 Spatially Adaptive L_p Norm Flattening Criterion

To both seek strong smoothing effects and avoid undesired or spurious edges, we propose the spatially adaptive L_p norm flattening criterion. There are two sparsity-inducing norms employed totally. One is $L_{0.5}$ norm whose p value is in range $[0, 1]$, which is used to enforce the strong smoothing effects, while the other is L_2 norm used for blurring the unnecessary regions.

To help locate where to apply $L_{0.5}$ norm or L_2 norm, we calculate an edge map for the image first, of which each point is basically computed as the weighted sum of color difference among nearby pixels. The edge map of the output image is formulated as follows,

$$E_i(T) = \sum_{j \in \mathcal{N}(i)} |\sum_c (T_{i,c} - T_{j,c})| \tag{6}$$

where $\mathcal{N}(i)$ denotes any points within 2-pixel distance from the current point i, c denotes the three color channels of T. Accordingly, the guided input edge map is represented as $E_i(I)$, which is computed by the above equation initally, and modified further to adapt to distinct applications.

Afterwards, the spatially variant L_p norm is formulated as a function defined on the computed input and output edge map, p_i. The basic principal to determine the p value is that if an output edge point is relatively larger than an insignificant input edge point, we consider it as the undesired or spurious edge point, which is applied by L_2 norm, otherwise it's applied by $L_{0.5}$ norm. The mathematical L_p norm function is detailed below,

$$p_i = \begin{cases} 2 & \text{if } E_i(T) - E_i(I) > c_1 \text{ and } E_i(I) < c_2, \\ 0.5 & \text{otherwise.} \end{cases} \tag{7}$$

where c_1 and c_2 denotes thresholds which are constant values.

Finally the spatially adaptive L_p norm flattening criterion is formulated based on the above L_p norm function,

$$\mathcal{E}_f = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_h(i)} w_{i,j}(p_i) \cdot |T_i - T_j|^{p_i} \tag{8}$$

$$w_{i,j}(p_i) = \begin{cases} w_{i,j,s} & \text{if } p_i = 2, \\ w_{i,j,r} & \text{otherwise.} \end{cases} \tag{9}$$

where spatial affinity (w_s) is specified while computing the gaussian weight for L_2 norm, as it's devoted to blurring the unnecessary edges, and color affinity (w_r) is used for $L_{0.5}$ norm to seek strong edge-preserving smoothing effects.

For the example of edge-preserving smoothing tasks, the guided edge map stays unmodified as the way computed using Eq. 6. To accomplish this task, the image is covered with only $L_{0.5}$ norm in the beginning. As the trained deep network converges, some moderate gradient regions may be over sharpened with arisen spurious edges which do not exist in the input edge map. Following Eq. 7, L_2 norm is covered near the over-sharpened regions to blur these edges while other areas are still applied with $L_{0.5}$ norm. Note the spatially adaptive L_p norm map is continuously changing as the parameter of the deep network keeps being optimized along the training process. With this proposed criterion, we are able to achieve strong smoothing effects while overcome the patchy or staircasing artifacts.

With such a strategy, we are able to implement various different smoothing effects by modifying the guided edge map. For example, to simulate the large aperture effects by blurring the background, we can remove all the background edges by simply multiplying the input edge map with the detected saliency mask. Then according to Eq. 7, all the background areas will be covered by L_2 norm for blurry effects, while the salient foreground object keeps the original way.

3.3 Edge Preserving Criterion

As observed in Figure 2, some thin and meaningful edges, such as the ones that indicate the ceramic tiles, are diminished for all the smooth images, no matter what sparsity-inducing norm is used. In fact, the flattening criterion does not explicitly determine if an edge point should be preserved or not, instead it prefers to pull adjacent pixels in similar colors or within close distance even closer, and keep the pixels in different colors or at a far distance apart where salient edges are sharpened. However, regarding some thin edges of which the two sides exhibit similar colors, the flattening criterion does not preserve them well, even if they could be discriminative for semantic meaning.

To deal with the aforementioned challenge, we propose the edge-preserving criterion, which explicitly highlights the important edges by minimizing the quadratic difference between the selected input and target edge points,

$$\mathcal{E}_e = \frac{1}{N_e} \sum_{i=1}^{N_e} ||E_i(T) - E_i(I)||_2^2 \tag{10}$$

where N_e refers to the number of all the important edge points, where the edge-preserving criterion is only computed on.

Note The approach for selecting the important edge points is determined by the user. Any existing edge detector can be directly applied to our edge-preserving criterion. Users can even label the important edge points interactively according to their own preferences. To better demonstrate the effectiveness of our algorithm in maintaining some weak edges which are potentially important but vulnerable to lose, we design an edge detector described in the supplementary material, which already shows satisfactory results for most images (All the results in the paper are facilitated using our edge selection method). Note more sophisticated edge detectors can be explored, but it is outside the discussion of this paper.

Overall, the energy function for minimization is formulated as follows,

$$\mathcal{E} = \mathcal{E}_d + \lambda_f \cdot \mathcal{E}_f + \lambda_e \cdot \mathcal{E}_e \tag{11}$$

3.4 Implementation in Deep Learning

This objective function is implemented as three separate loss layers, which is differentiable to the optimized image, in deep learning framework.

The data criterion is realized in the common Mean Squared Error (MSE) criterion, which has been incorporated in many deep learning platform. It minimizes the quadratic different between the smooth image T predicted by our deep network and the input image I to seek structure similarity.

The flattening criterion calculates the weighted sum of the L_p difference between each pixel and its neighbouring pixels, which is much more resource-intensive. To tackle this challenge, this energy term is computed in parallel for all the points in the image with GPU. The flattening criterion takes the predicted smooth image T from our network as input, and does not require any manually labeled image as ground truth to optimize, since this criterion is designed for regularization.

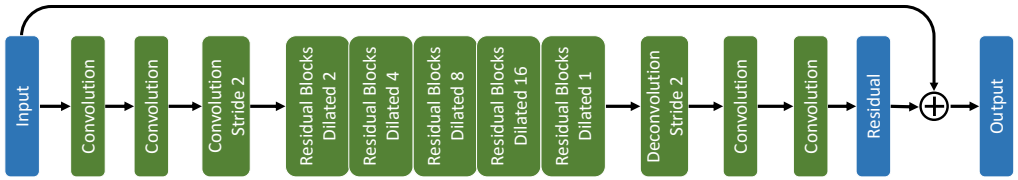

Fig. 4. The full picture of our dilated fully convolutional neural network (FCN). Our network contains 26 convolution layers, of which the middle 20 are organized into residual blocks and dilated convolutions with exponentially increasing skip steps. Skip connection is added for faster convergence.

The back propagation for the flattening criterion is computed as follows,

$$\frac{\partial \mathcal{E}_f}{\partial T} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_h(i)}(w_{i,j}(p_i)\cdot g(i,j,p_i) - w_{j,i}(p_j)\cdot g(j,i,p_j)) \quad (12)$$

$$g(i,j,p_i) = p_i \cdot |T_i - T_j|^{p_i-1} sign(T_i - T_j) \quad (13)$$

$g(j,i,p_j)$ can be computed similarly. Note while calculating the derivative of this flattening criterion to each point in the smooth image, we have to consider not only the condition that it is the center pixel among the surrounding window $(w_{i,j}(p_i)\cdot g(i,j,p_i))$, but also the condition that it is within the neighbourhood of other center pixels $(w_{j,i}(p_j)\cdot g(j,i,p_j))$.

The edge-preserving criterion computes the quadratic difference on the selected important edge points, and the corresponding back propagation is computed as

$$\frac{\partial \mathcal{E}_e}{\partial E(T)} = \frac{2}{N_e}\sum_{i=1}^{N_e}(E_i(T) - E_i(I)) \quad (14)$$

Later on, the gradients are back propagated to the smooth image from the edge map.

$$\frac{\partial E(T)}{\partial T_i} = \sum_{j\in\mathcal{N}(i)} sign(\sum_{c}(T_{i,c} - T_{j,c}))\cdot(\frac{\partial \mathcal{E}_e}{\partial E_i(T)} + \frac{\partial \mathcal{E}_e}{\partial E_j(T)}) \quad (15)$$

The above computed gradient can be applied to all the color channels in T.

Note as our algorithm doesn't require any specifically generated ground truth images for training the deep network, this is completely unsupervised learned.

3.5 Network Structure

In this section, we introduce our dilated fully convolutional network (FCN) for minimizing the aforementioned objective function, as illustrated in Figure 4.

Inspired by previous work [Yu and Koltun 2015] to enlarge receptive field explicitly with multiple dilated convolutions for semantic segmentation, and [Kim et al. 2016] to use very deep convolutional neural network equipped with residual learning for super-resolution, our network structure is designed with 26 convolutional layers with skip connection in order to learn to generate smooth images by optimizing the proposed criterion.

Specifically, of the 26 convolution layers, the third one downsamples the extracted feature maps by half via using stride of 2. The third one from the last is replaced by deconvolutional layer with stride 2 to recover the original image size. The middle 20 convolutional layers are organized as 10 residual blocks [He et al. 2016]. As image smoothing requires considering contextual information across relatively wide region, we explicitly increase the receptive field of CNN by replacing the original convolution with dilated convolution with exponentially increasing skip steps as shown in Figure 4. Every 2 consecutive residual blocks shares the same dilation factors and are grouped together.

All the convolutional layers contain kernels of size 3×3 and outputs 64 feature maps, followed by a batch normalization (BN) and ReLU layer, except for the last one, which reconstructs 3 channel residual image without following normalization and activation.

Note our dilated residual blocks are organized as an order with exponentially doubling step length, which guarantees in a $n \times n$ grid, any points can be reached by a random seed among the grid in logarithmic steps $log(n)$, which is very effective and efficient. This is also acknowledged as jump flooding used in parallel GPU implementation of Voronoi diagram [Rong and Tan 2006] and PatchMatch [Fan et al. 2015].

As for image smoothing task, the input and output image is highly correlated, but the color may be degenerated during many forward convolution operations in a deep neural network as observed in [Kim et al. 2016]. To overcome such a difficulty, we add a skip connection introduced in [Kim et al. 2016] from the input image following the output of the CNN layers, therefore the output smooth image becomes point-wise summation of the learned residual image and input image.

We also observe that for acceleration reasons, a variety of recent papers [Liu et al. 2016; Xu et al. 2015; ?] simulate the traditional image smoothing algorithms by learning a deep neural network with ground truth images generated by existing filters, which is essentially different from our task , which is dedicated to learning to optimize an image smoothing energy function in an unsupervised way.

3.6 Training Details

Since our framework doesn't require any careful design to generate ground truth smooth images for pixel-wise minimization, any natural images can be incorporated for training. To represent the generalization for the majority of natural images, we deploy around 17,000 images in the PASCAL VOC dataset [Everingham et al. 2010] as input images. These images were collected in the flicker photo-sharing website which represents high quality and a wide range of viewing conditions. These images are cropped to a reasonable size 224 × 224 to accelerate training process without influencing smoothness qualities.

Our network and objective function is implemented in Torch framework using mini-batch gradient descent, with batch size set as

2017-10-20 01:32 page 6 (pp. 1-12) Submission ID: 867-5309

1. The network parameters are randomly sampled using the method in [He et al. 2015]. The objective function is minimized using Adam [Kingma and Ba 2015] with learning rate set as 0.01. The training process takes only about 8 hours using Nvidia Geforce 1080 GPU.

Fig. 5. Demonstration of the effectiveness of the proposed spatially adaptive L_p norm flattening criterion. Each time only one sparsity-inducing norm is applied among the whole image to explore how much gap there is between a single norm and our hybrid norm. Two images with only $L_{0.5}$ norm are shown on the top, while two images with only L_2 norm are displayed on the bottom.

4 ALGORITHM ANALYSIS

To study the importance of the proposed criterion and deep-learning based solver, several critical questions are thoroughly discussed here.

Is the spatially adaptive L_p norm better than one single norm? To verify the necessity of the spatially adaptive L_p norm in achieving strong smoothing effects while avoiding the over-sharpened artifacts, we replace the spatially variant norm in the flattening criterion with only $L_{0.5}$ and L_2 norm respectively among the whole image.

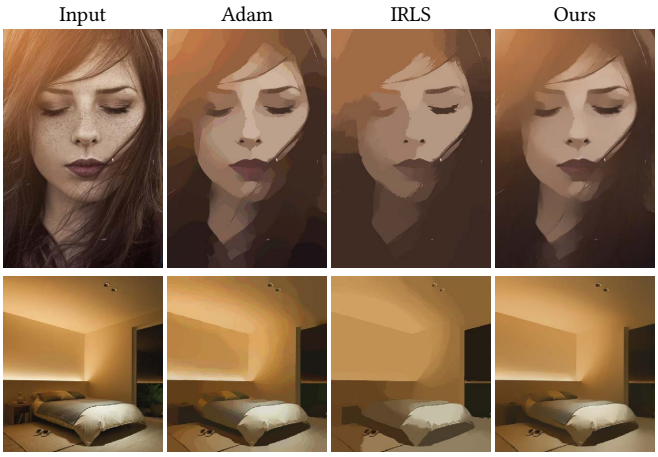

Fig. 6. Comparison between learned deep network and traditional numerical solvers. The popular gradient descent optimizer Adam [Kingma and Ba 2015] and iterative reweighted least square (IRLS) [Holland and Welsch 1977] are used for comparison.

While only $L_{0.5}$ norm is employed, it realizes strong smoothing effects but also creates staircasing artifacts within homogeneous regions by breaking them up into piecewise constant regions with spurious edges. For example in Figure 5, within the girl's cheek, the light transforms continuously from bright to dark, in order to seek strong sparsity with limited color possibilities, some edges which do not exist in the input image are enhanced or even created to separate the smooth region into different colors. Moreover, on the top of the cloud image, similar over sharpened side effects are also observed. These artifacts destroy the original image structure and hurt the appearance of the input image.

As shown on the right of Figure 5, after mixing $L_{0.5}$ and L_2 norm altogether in the flattening criterion, these strong smoothness effects are maintained while the artifacts are diminished. Note some slender hair lines are also preserved very well in our smooth image.

On the other hand, while applying L_2 norm only for the bottom two images in Figure 5, the text "**NAOMI**" on the train can be hardly seen clearly. The whole image looks very blurry and much less sharper than ours. It's the same case for the bottom image, details are not preserved well with only L_2 norm.

More discussions about different sparsity-inducing norms can be referred in [Bach et al. 2012]. The small L_p norm prefers smooth, sharp and piece-wise constant effects, while the large norm tends to generate more blurry results.

Is the edge-preserving criterion helpful? To answer this question, we remove the edge-preserving criterion in our objective function and retrain the deep network for comparison.

As shown in Figure 7, the above input image contains complex textures and noises in the wooden board where the three horizontal lines are easy to be observed by human beings but hard to be preserved with only the flattening criterion. The weak and slender boundaries between the ceramic tiles on the background wall are

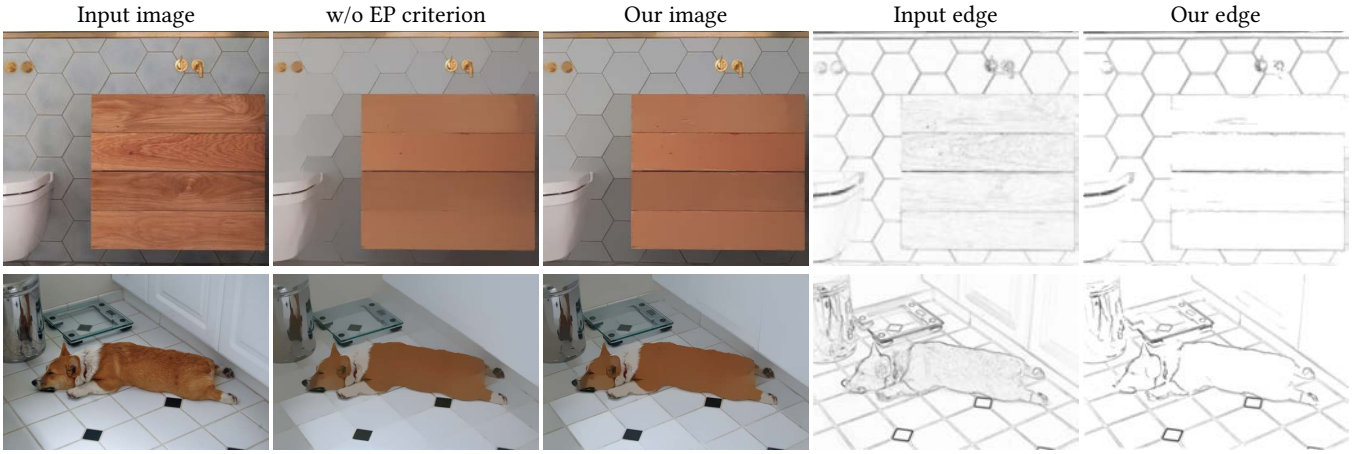

Fig. 7. Demonstration of the effectiveness of the proposed edge-preserving criterion. The smooth images without and with edge-preserving criterion is shown on the second and third column. We can see that our full algorithm maintains the structures very well. The guided edge map designed by ourselves are shown on the right, which are in accordance with preserved structure in the smooth image, which justifies the good learning ability of our deep network.

also prone to being missed. The input and our guided edge map are shown on the right. Without the edge-preserving criterion, the main image structures are mostly destroyed and the important edges are weakened in the smooth image. On the contrary, the smooth image generated with the full algorithm preserves the details very well. A similar example is shown in the bottom dog case, without assistance of the edge-preserving criterion, it's very vulnerable to lose the textures on the ground.

Note the main structure of the smooth image are basically in accordance with our selected edge map, which verifies the effectiveness of our edge-preserving term again. These guided edge maps are incorporated only on the training process, and not required for evaluation. From the consistance between our smooth image and selected edge map, we can see that our deep neural network indeed learns what kind of image structures we want it to learn.

Does the traditional numerical solver work better? For fair comparison, two popular classical solvers, gradient descent method Adam [Kingma and Ba 2015] and iterative reweighted least square (IRLS) [Holland and Welsch 1977], are discussed here. As our proposed criterion is differentiable to the smooth image, gradient descent method can be directly applied to solve this problem. IRLS has been well known for minimizing energy functions with non-quadratic forms, such as L_p norm (0<p<1) [Min et al. 2014] or L_1 norm [Xu et al. 2012] in the image smoothing tasks.

To utilize IRLS for optimization, a quadratic upper bound of the energy function has to be defined to minimize it. However, the proposed edge-preserving criterion is highly non-quadratic and cannot be transformed into a weighted L_2 form. Therefore in this section, we leave only the data term and the spatially adaptive L_p norm flattening term in the energy function for minimization. Note on the other hand, this demonstrates wide adaptability of our deep learning based solver to complex energy functions.

We compare the smooth images optimized by different solvers in Figure 6. The optimization process of gradient descent based method is very slow, which takes about 100 iterations for Adam to converge well in our task. Since the optimized image is gradually changing in a small step each time, there form many independent stripes which break up the image into many local smooth regions, which can be clearly observed in the shown image.

IRLS converges much faster than Adam, since in each step it finds the optimal solution to its quadratic upper bound. However it still takes about 10 iterations to generate stable smooth image. We observe similar artifacts in the images optimized by IRLS, which exhibit less stripes than Adam but they are still visible. Moreover we also find that the images tend to be over smoothed in some local regions.

Unlike traditional numerical solvers, our algorithm incorporates all the smoothing and edge-preserving effects altogether learned within a single deep neural network, which is more robust to outliers and only request one forward pass for evaluation. From the images shown in Figure 6, you can see our algorithm maintains the primary image structures well and shows satisfactory smoothing effects.

5 APPLICATIONS

In this section, we demonstrate the effectiveness and versatility of our image smoothing algorithm in a range of different low-level vision or computer graphics applications, such as image abstraction, pencil sketching, detail magnification, background blur, texture removal, *etc.*

5.1 Image Abstraction and Pencil Sketching

Image smoothing can be directly applied to some stylization systems. For example, [Winnemöller et al. 2006] proposes an efficient framework that abstracts imagery by simplifying low-amplitude regions, and increase contrast of visually important regions with difference-of-Gaussian edges.

To demonstrate the effectiveness of our algorithm in the simplest way following previous work [Farbman et al. 2008; Xu et al. 2011], we replace the iterative bilateral filter in [Winnemöller et al. 2006] with our learned edge-preserving image smoother, and the extracted

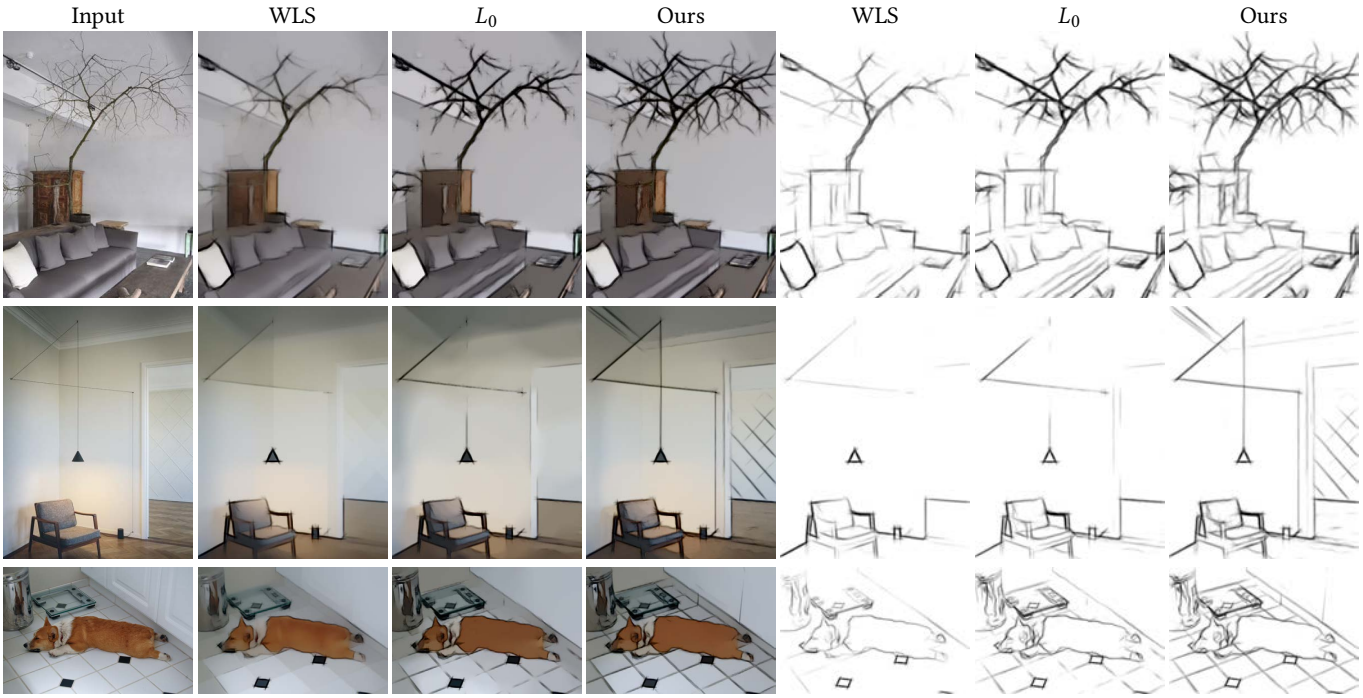

Fig. 8. Image abstractions (left) and pencil sketching (right) results. Compared to traditional smoothing algorithms, our learned edge-preserving image smoother maintains better subtle important structures.

lines are decorated with random sketches in different directions to generate a pencil drawing appearance.

Figure 8 demonstrates three image abstraction and pencil sketching examples. As shown in the tree image, the twigs are small, thin and is hardly visible. These important and meaningful edges are extracted from our smooth image, enhanced with random strokes and composed back to the smooth image to give an abstract look. Compared to our edge-preserving smoothing algorithms, the other traditional methods cannot preserve such edges well and destroy the original image structure.

Similar phenomenon can be observed in the bottom two images. The electronic lines of the ceiling lamp and the boundaries of ceramic tiles on the ground are also hard to be remained after applying traditional smoothing algorithms, while our algorithm is able to preserve a much better structure.

5.2 Detail Magnification

Edge-preserving flattening results can also serve as the piecewise smooth base layer for detail exaggeration tasks, where a residual detail layer is extracted by computing the difference between the original image and base layer. Afterwards, the decomposed details are enhanced by applying a sigmoid function and composed back to the base layer to generate the detail magnification effects. Note more sophisticated approach can be applied, but at the moment we only focus on demonstrating the value of our algorithm in the simplest way.

The ideal edge-preserving smoothing algorithm should neither blur or sharpen salient edges, since either one will result in ringing

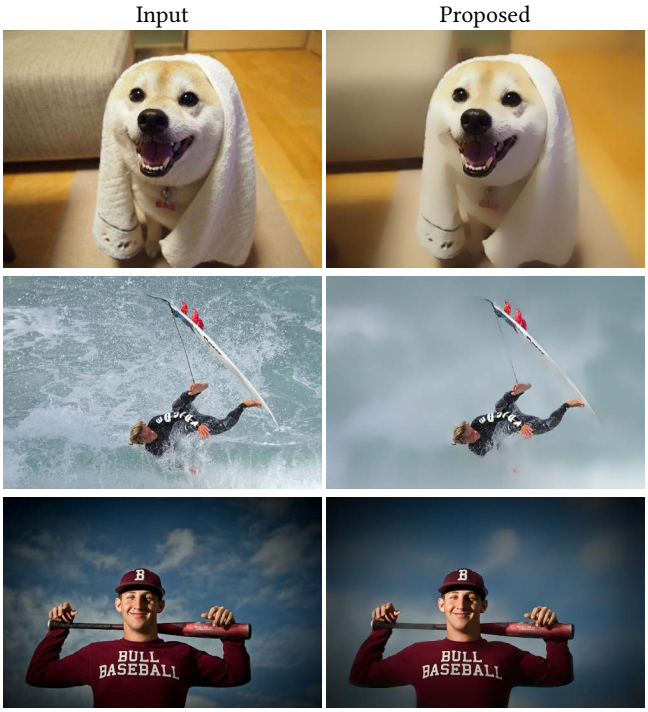

Fig. 9. Demonstration on blurring the background out of saliency.

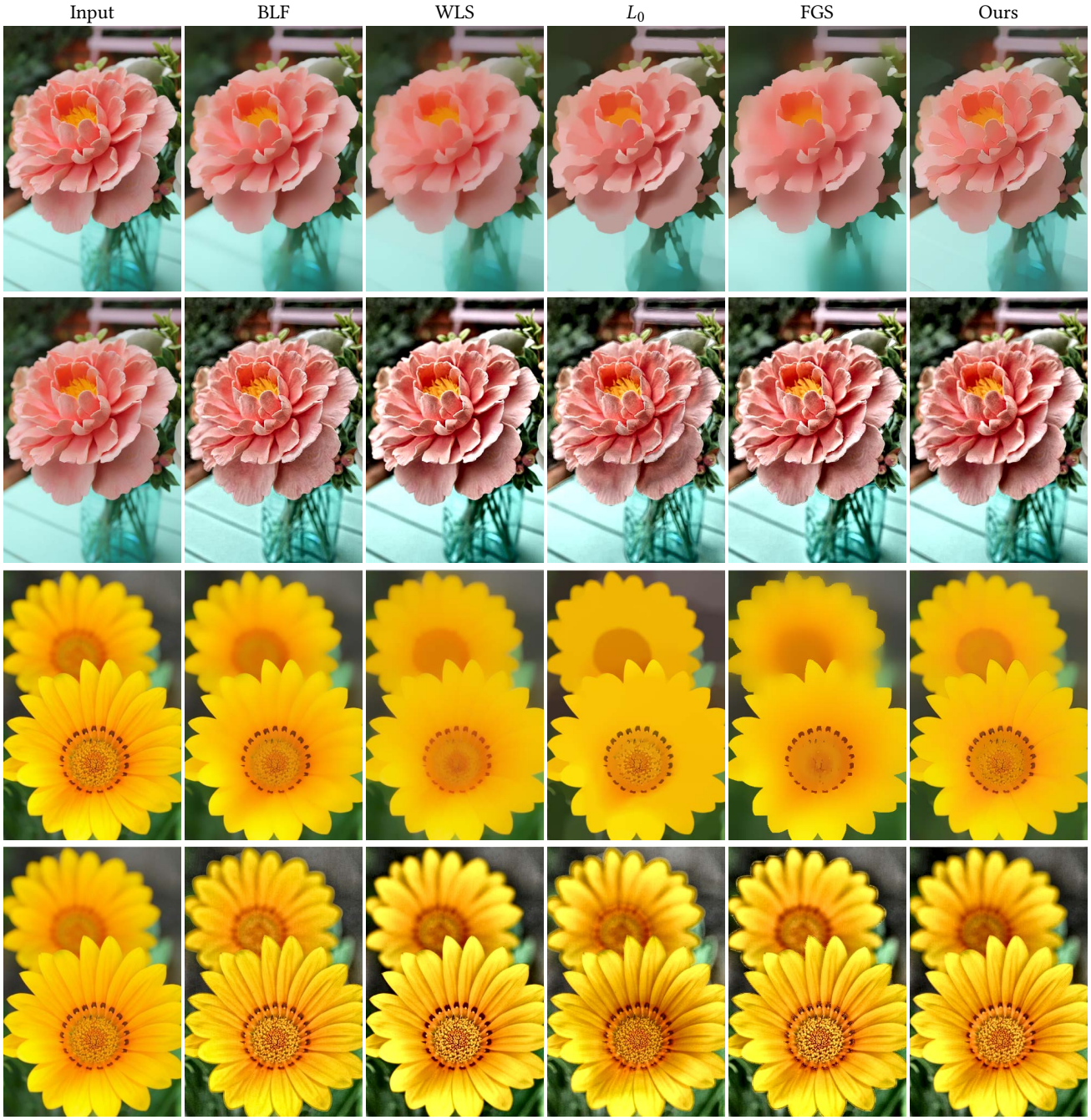

Fig. 10. Detail magnification compared with other traditional image smoothing algorithms, BLF [Tomasi 1998], WLS [Farbman et al. 2008], L_0 [Xu et al. 2011] and FGS [Min et al. 2014]. The top row demonstrates the smooth base layer, while the bottom one displays corresponding enhancement results. The first column shows the same two input images.

in the detail layer, which may cause halo or gradient reversals in the detail enhanced images. However, developing such an image smoothing filter is difficult, since it's usually tricky to determine what edges to preserve and how to maintain part of the edges while not to blur the others in a global controllable degree.

In our algorithm, to overcome this issue, we minimize the edge-preserving criterion to avoid the important edges to be blurred and optimize the spatially adaptive L_p norm criterion to eliminate the over-sharpened edges induced while seeking strong smoothing effects.

2017-10-20 01:32 page 10 (pp. 1-12) Submission ID: 867-5309

Two examples are shown in Figure 10. For the bottom one, a clear flower stays in the foreground while another blurry one is shown as the background. The traditional edge-preserving smoothing algorithms tend to sharpen the blurry boundaries of the background flower in the smooth image, where obvious gradient reversals are observed in the corresponding detail exaggerated images. We are able to maintain the structures well while not either enhancing or attenuating the edges that separate coarse scale image features in the smooth image, and no such artifacts exist in our detail enhanced image.

Fig. 11. Demonstration on blurring the background out of saliency.

5.3 Background Blur

One popular photographic technique is to blur the background in order to make foreground stand out. To implement such an effect, photographers usually need to capture the image using a professional camera and a lens with large aperture, and stay in a limited distance with the object which is out of focus from the background. However, such a camera and lens are neither portable to carry all the time, nor cheap enough to be accessible to everyone. Therefore, one possible solution to overcome such difficulties is to simulate the background blur effect with an existing image captured by our smart phones.

In this paper, we achieve this purpose by blurring the background with L_2 norm and smoothing the foreground with weak $L_{0.5}$ norm. In our implementation, the foreground is defined by the noticeable region generated using very recent state-of-the-art salient object detection paper [Zhang et al. 2017]. Note the primary modification from edge-preserving smoothing to background blur is just to mask out the guided edges in the background feeded into the proposed criterion, which is surprisingly easy to implement. Therefore, according to Eq. 7, the background region will be covered by L_2 norm, while the others stay the same as before. Note the whole process is incorporated into one single step, taken by a deep neural network which learns both saliency information and image smoothing effect.

A few examples are demonstrated in Figure 9 and 11. As can be seen, the main structure of foreground is maintained well, such as the hair and face in the surfing case of Figure 9, which is surrounded but not affected by the blurry background. The standing girl image in Figure 11 demonstrates an example that foreground and background stays too close to have enough depth difference for physical lens to generate bokeh effect, while ours does not encounter such an issue.

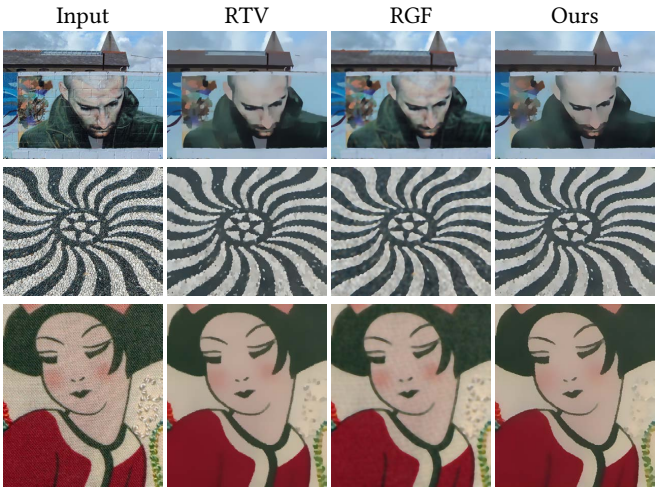

Fig. 12. Texture smoothing results compared with state-of-the-art image smoothing papers, RTV [Xu et al. 2012] and RGF [Zhang et al. 2014], which are dedicated on this specific task.

5.4 Texture Removal

Texture is usually composed of small-scale repeated identical or similar patterns, which distracts people from the large-scale meaningful structures.

To eliminate such textures, our proposed criterion can be easily modified to be well suited to it. In our implementation, we detect the texture edges applied with L_2 norm and cover the other regions with $L_{0.5}$ norm. Since these are repetitive patterns, such edges usually share similar colors on its two sides, which can be easily extracted. Similar to background blur application, we only need to modify the guided edge map feeded to the proposed criterions by masking out such edges, while leave the others unchanged.

Figure 12 shows three examples which demonstrates different kinds of patterns, graffiti, mosaic and embroidery. Our results all achieve satisfactory results compared to the traditional smoothing algorithms. As small sparsity-inducing norm is utilized, our results appears smoother and sharper.

REFERENCES

Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. 2012. Structured sparsity through convex optimization. *Statist. Sci.* 27, 4 (2012), 450–468.

Linchao Bao, Yibing Song, Qingxiong Yang, Hao Yuan, and Gang Wang. 2014. Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree. *IEEE Transactions on Image Processing* 23, 2 (2014), 555–569.

S. Bi, X. Han, and Y. Yu. 2015. An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 78.

Jiawen Chen, Sylvain Paris, and Frédo Durand. 2007. Real-time edge-aware image processing with the bilateral grid. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 103.

Frédo Durand and Julie Dorsey. 2002. Fast bilateral filtering for the display of high-dynamic-range images. In *ACM transactions on graphics (TOG)*, Vol. 21. ACM, 257–266.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88, 2 (2010), 303–338.

Qingnan Fan, David Wipf, Gang Hua, and Baoquan Chen. 2017. Revisiting Deep Image Smoothing and Intrinsic Image Decomposition. *arXiv preprint arXiv:1701.02965* (2017).

Qingnan Fan, Fan Zhong, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. 2015. JumpCut: non-successive mask transfer and interpolation for video cutout. *ACM Trans. Graph.* 34, 6 (2015), 195.

Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. 2008. Edge-Preserving Decompositions for Multi-Scale Tone and Detail Manipulation. *ACM Transactions on Graphics (TOG)* 27, 3 (2008).

Raanan Fattal. 2009a. Edge-avoiding wavelets and their applications. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 22.

Raanan Fattal. 2009b. Edge-avoiding wavelets and their applications. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 22.

Eduardo SL Gastal and Manuel M Oliveira. 2011. Domain transform for edge-aware image and video processing. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 69.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*. 1026–1034.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

Paul W Holland and Roy E Welsch. 1977. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods* 6, 9 (1977), 813–827.

Michael Kass and Justin Solomon. 2010. Smoothed local histogram filters. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 100.

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1646–1654.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)* (2015).

Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. 2007. Joint bilateral upsampling. In *ACM Transactions on Graphics (ToG)*, Vol. 26. ACM, 96.

Sifei Liu, Jinshan Pan, and Ming-Hsuan Yang. 2016. Learning Recursive Filters for Low-Level Vision via a Hybrid Neural Network. In *European Conference on Computer Vision (ECCV)*.

Wei Liu, Xiaogang Chen, Chuanhua Shen, Zhi Liu, and Jie Yang. 2017. Semi-Global Weighted Least Squares in Image Filtering. (2017).

S. Hasinoff J. Kautz M. Aubry, S. Paris and F. Durand. 2014. Fast Local Laplacian Filters: Theory and Applications. *ACM Transactions on Graphics* (2014).

Dongbo Min, Sunghwan Choi, Jiangbo Lu, Bumsub Ham, Kwanghoon Sohn, and Minh N Do. 2014. Fast global image smoothing based on weighted least squares. *IEEE Transactions on Image Processing* 23, 12 (2014), 5638–5653.

Sylvain Paris and Frédo Durand. 2006. A fast approximation of the bilateral filter using a signal processing approach. *Computer Vision–ECCV 2006* (2006), 568–580.

Pietro Perona and Jitendra Malik. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence* 12, 7 (1990), 629–639.

Georg Petschnigg, Maneesh Agrawala, Hugues Hoppe, Richard Szeliski, Michael Cohen, and Kentaro Toyama. 2004. Digital Photography with Flash and No-flash Image Pairs. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 664–672. https://doi.org/10.1145/1015706.1015777

Guodong Rong and Tiow-Seng Tan. 2006. Jump flooding in GPU with applications to Voronoi diagram and distance transform. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*. ACM, 109–116.

Leonid I Rudin, Stanley Osher, and Emad Fatemi. 1992. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena* 60, 1-4 (1992), 259–268.

Xiaoyong Shen, Aaron Hertzmann, Jiaya Jia, Sylvain Paris, Brian Price, Eli Shechtman, and Ian Sachs. 2016. Automatic portrait segmentation for image stylization. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 93–102.

Carlo Tomasi. 1998. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*. IEEE, 839–846.

Ben Weiss. 2006. Fast median and bilateral filtering. *Acm Transactions on Graphics (TOG)* 25, 3 (2006), 519–526.

Holger Winnemöller, Sven C Olsen, and Bruce Gooch. 2006. Real-time video abstraction. In *ACM Transactions On Graphics (TOG)*, Vol. 25. ACM, 1221–1226.

Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. 2011. Image smoothing via L_0 gradient minimization. In *ACM Transactions on Graphics (TOG)*, Vol. 30. 174.

Li Xu, Jimmy SJ. Ren, Qiong Yan, Renjie Liao, and Jiaya Jia. 2015. Deep Edge-Aware Filters. In *International Conference on Machine Learning (ICML)*. 1669–1678.

Li Xu, Qiong Yan, Yang Xia, and Jiaya Jia. 2012. Structure Extraction from Texture via Natural Variation Measure. *ACM Transactions on Graphics (TOG)* (2012).

Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* (2015).

Pingping Zhang, Dong Wang, Huchuan Lu, Hongyu Wang, and Xiang Ruan. 2017. Amulet: Aggregating Multi-level Convolutional Features for Salient Object Detection. (2017).

Qi Zhang, Xiaoyong Shen, Li Xu, and Jiaya Jia. 2014. Rolling Guidance Filter. In *European Conference on Computer Vision (ECCV)*. 815–830.

Revisiting Deep Image Smoothing and Intrinsic Image Decomposition

Qingnan Fan[*1] David Wipf[2] Jiaolong Yang[2] Gang Hua[2] Baoquan Chen[1,3]
[1]Shandong University [2]Microsoft Research [3]Shenzhen Research Institute, Shandong University
fqnchina@gmail.com, {davidwip, jiaoyan,ganghua}@microsoft.com, baoquan@sdu.edu.cn

Abstract

We propose an image smoothing approximation and intrinsic image decomposition method based on a modified convolutional neural network architecture with large receptive fields applied directly to the original color image. When training a deep model for these purposes however, it is quite difficult to generate edge-preserving images without undesirable color differences or boundary artifacts. To overcome these obstacles, we supervise intermediate outputs from different functional components during training. For example, we apply both image gradient supervision and a channel-wise rescaling layer that computes a minimum mean-squared error color correction. Additionally, to enhance piece-wise constant effects for image smoothing, we append a domain transform filter with a predicted refined edge map. The resulting deep model, which can be trained end-to-end, directly learns edge-preserving smooth images and intrinsic image decompositions without any special design, specialized auxiliary training data, or input scaling/size requirements. Moreover, our method shows much better numerical and visual results on both tasks and runs in comparable test time to existing deep methods.

1. Introduction

Image smoothing is a fundamental building block of many computer vision and graphics applications including texture removal, image enhancement, edge extraction, image abstraction, and beyond. The underlying goal of image smoothing, or edge preserving filtering, is to extract sparse salient structures like perceptually important edges and contours, while minimizing color differences in some input image of interest. A vast array of practical filtering algorithms have been proposed for this task using traditional image processing principles and energy functions that operate across both local and global image regions [5, 12, 13, 18, 26, 27, 32, 34, 35, 36]. However, often an expensive, impractical optimization problem must be solved per instance to produce the desired edge-aware filtered images [5, 18].

Recently, deep learning has demonstrated a remarkable ability to replace expensive optimization procedures with cheap computational modules that can be trained offline when given access to representative data, and later deployed at a fraction of the computational budget [15]. For example, given a sufficient training corpus obtained by passing input images through some preferred smoothing filter, we can attempt to learn a deep model that efficiently mimics the filter outputs [33]. Of course considerable care must be taken in designing such a network to reduce training data requirements and to ensure that the trained model is indeed capable of fast, robust inference on new images, and naive, black-box approaches are destined to fail. Perhaps even more importantly, a well-designed network for image filtering can also serve as the foundation for more complex, higher-level vision tasks, without the need for cumbersome integration with an iterative, energy-based filtering approach.

In this regard, one notable application of edge-preserving image smoothing is the computation of intrinsic image decompositions [5], which are useful for material recognition, image-based resurfacing, and relighting, among other things. The intrinsic image model assumes an ideal diffuse environment, in which an input image $\boldsymbol{I}$ is the pixel-wise product of an albedo image $\boldsymbol{A}$ and a shading image $\boldsymbol{S}$, i.e.,

$$\boldsymbol{I} \approx \boldsymbol{A} \odot \boldsymbol{S}. \tag{1}$$

The albedo or reflectance image indicates how object surface materials reflect light, while shading accounts for illumination effects due to geometry, shadows, and inter-reflections. Obviously, estimating such a decomposition is a fundamentally ill-posed problem as there exist infinitely many feasible decompositions. Fortunately though, prior information, often instantiated via specially tailored image smoothing filters or energy terms, allows us to constrain the space of feasible solutions [2, 4, 6, 9, 28, 37]. For example, the albedo image will usually be nearly piece-wise constant, with a finite number of levels reflecting a discrete set of materials and boundaries common to natural scenes. In contrast, the shading image is often assumed to be grayscale,

[*]This work was done when Qingnan Fan was an intern at MSR.

and is more likely to contain smooth gradations quantified by small gradients except at locations with cast shadows or abrupt changes in scene geometry [22].

Naturally, given access to ground truth intrinsic image decompositions, deep networks provide a data-driven candidate for solving this ill-posed inverse problem with fewer heuristic or hand-crafted assumptions. In this paper, we propose a flexible deep neural network architecture that is simultaneously capable of handling wide-ranging, generic image smoothing tasks, where the primary goal is an efficient implementation of existing edge-aware filtering effects, as well as intrinsic image decompositions, where now an augmented objective supports improved accuracy on a higher-level vision task. Our fully trainable, end-to-end network relies on three important ingredients:

Large Flexible Receptive Fields: Edge-aware filtering requires contextual information across a relatively wide image region, especially in the context of an end-to-end system. Additionally, accurate intrinsic image decompositions must account for the fact that distant pixels with the same reflectance on a large surface could have significant color differences due to the accumulation of smoothly-varying soft cast shadows [5]. Hence large receptive fields are essential for both tasks, and we choose to actualize them with maximal flexibility via a fully convolutional subnetwork with many layers.

Color Correction and Gradient Supervision: For image smoothing, the input and output color images should be highly correlated, but unfortunately the color may be attenuated during many forward convolutions in a deep network such as ours. To address this problem in the context of superresolution, [19] adds a skip connection directly from input to output images, which is not effective for image smoothing in our experience. Instead, we include a color correction layer which scales each channel of the image to minimize MSE color differences and naturally allows gradient backpropagation for training purposes. We also observe that supervising gradients of predicted images using an auxiliary objective helps preserve salient edges; however, this is notably different than direct operation in the gradient domain akin to most existing methods.

Domain Transform: To further enhance piece-wise constant effects favored by sparsity-based filtering methods [32, 5], we also leverage a domain transform [14, 8]. Practically speaking, in our system this can be viewed as a series of final, parameterless network layers that implicitly filter color information across the image in a warped domain. This warping is determined by edge maps derived from a preliminary smoothed image, which emerges from the lower convolutional and color correction layers described previously. Although the domain transform layers themselves do not actually increase the overall network capacity per se, they nonetheless alter the space of candidate filters and influence which image regions should be smoothed or not.

Overall, we propose an end-to-end system that can efficiently approximate a variety of computationally intensive image smoothing filters as well as infer related intrinsic image decompositions. For each of these tasks, our method demonstrates state-of-the-art performance both visually and numerically. We differentiate our design choices from existing methods in the next section.

2. Relationship with Existing Models

Image Smoothing: Recently, there have been multiple attempts [33, 25] to imitate the behavior of a large, representative family of image smoothing filters using a single, data-driven approximation framework capable of accelerated inference speeds. For example, [33] proposed a 3-layer network for first predicting prominent gradients, followed by a separate post-processing step involving filter-specific tuning parameters to reconstruct the smoothed image estimate. A potential limitation however, is that gradient or salient structure prediction errors can easily propagate to surrounding areas in the pixel domain during the reconstruction step, reducing visual quality. Moreover, practical image smoothing filters favor varying degrees of edge preservation, spanning the gamut from blurry to piece-wise constant images. Purely gradient-based processing tends to struggle more generating images on the blurry end of this scale, where continuous color transitions may be required. For these reasons, our approach operates entirely in the original color image domain, a gradient supervision term notwithstanding.

A second approach from [25] involves a hybrid network composed of several convolutional layers followed by a set of recurrent network layers. In brief, the CNN portion of the model takes the original image as input and computes a set of weights that are presumably reflective of salient image structures. The recurrent network layers then take these weights, as well as the original image, and act as a form of data-driven guided filter to produce the desired smoothing effect. To a certain degree, our approach can be viewed as the antithesis of this model. Whereas we explicitly parameterize an image model and only later refine it using gradient supervision and a parameterless domain transform modulated via a small subnetwork, the method from [25] fully parameterizes the weights of a recurrent guided filter array, which acts much like a domain transform via the analysis from [8], but then applies these filters only to multi-scale versions of the original raw images. A downside of the later strategy is that it requires downsampling the original images to multiple scales, and at least in its present form, will not work with image sizes that are not multiples of 16 (provided code actually only works with special image sizes). Additionally, any mismatch between the computed weights

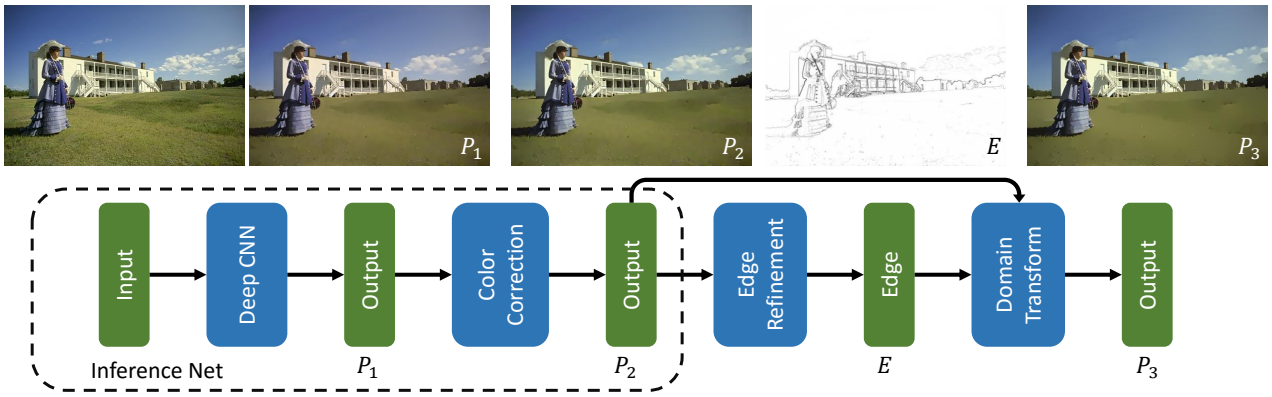

Figure 1. **Basic Network Structure**. For smoothing/filtering tasks, an input image is fed into an inference network (Deep CNN + channel-wise rescaling layer for color correction), followed by a domain transform modulated using an edge refinement step. For intrinsic image decompositions, this pipeline is modified slightly via separate albedo and shading paths as described in the text.

and legitimate salient edge structure will necessarily propagate to wider regions in the predicted output image since the recurrent filters have no mechanism to compensate.

Intrinsic Image Decomposition: To move beyond the traditional filtering approaches mentioned in Section 1 to data-driven deep models capable of producing scene-level intrinsic image decompositions, we require images with known ground truth. The MPI-Sintel intrinsic images benchmark [7] provides an important step in this direction, and several existing deep network pipelines have been built upon it. First, [31] learns a two-scale convolutional network to directly predict both albedo and shading images. However, the specific architecture, which closely resembles that from [10] developed for predicting depth and surface normals, involves intermediate feature maps at 1/32 scale such that significant detail information may be compromised. A second more recent method from [20] trains a joint model to learn depth maps and intrinsic images with activations coupled across iterations. Unlike our approach, the resulting pipeline operates in the gradient domain and requires separate post-processing steps, as well as additional guidance from labeled depth images. Additionally, [24] leverages a generative adversarial network (GAN) to learn intrinsic images from a special resynthesized dataset, but does not provide models or results from existing MPI-Sintel benchmarks for comparison. Finally, [29] trains an encoder-decoder CNN to learn albedo, shading and specular images; however, this approach does not apply to scene-level images as our approach does.

3. Proposed Model

Our proposed model consists of three main parts: an inference net, an edge refinement step, and a domain transform as illustrated in Figure 1. We will now detail each component in turn.

3.1. Inference Net

Unlike previous work that primarily attempts to explicitly predict sparse image gradients [33, 20] or related weight maps [25], the primary parameterized workhorse of our model is an inference net designed to directly predict filtered images in an end-to-end manner. This network contains 20 convolutional neural network (CNN) layers followed by a color correction layer and two gradient computation layers. These are designed as follows:

CNN Layers: Of the 20 CNN layers, the middle 18 are all of the equivalent size $64\times64\times3\times3$, meaning that for each of 64 output feature maps, 3×3 spatial regions of all 64 input feature maps are filtered. In contrast, the first layer takes the input image and represents it as 64 feature maps, while the last layer transforms the multi-channel representation back to the original image space (3 color channels). In both cases the kernel sizes are also 3×3. Additionally, each convolutional layer is followed by batch normalization (BN) and ReLU layers except for the last one. Finally, to accelerate the training process, we retrofit the middle 16 convolutions with 8 modified residual blocks (see the supplementary file for details of these modifications and network structure).

As observed in [33], a fully convolutional network trained directly in the original image domain may tend to generate very blurry images and unwanted details, while gradient domain supervision is more sensitive to the changes in sharp edges that comply with human perception of contrast more than absolute color values. But we argue that this limitation of the former is largely due to unsufficient receptive field size and flexibility. After all, image smoothing and intrinsic image decomposition techniques determine the color values of each pixel in the context of a large surrounding area. Especially for albedo images, small color difference between nearby pixels can accumulate, producing significant differences between distant pixels with complex connecting pathways even when they share the

same reflectance. This renders the estimation problem quite difficult if we are restricted to either small or insufficiently parameterized receptive fields. Therefore, instead of using 3 convolutional layers of large kernel size (16×16, 1×1 and 8×8) as in [33], we adopt 20 convolutional layers with 3×3 kernels, which allows the learned regression function to more accurately account for distant contextual information [30].

Color Correction Layer: Although image smoothing attempts to remove small gradients, it should still generally preserve a high degree of correlation between input and output colors. With so many convolutional layers however, such correlations may eventually weaken without some additional source of long-term memory that reflects the preservation of the original color schema. For this purpose, we include an additional color correction layer to scale each channel independently as follows. Let $\boldsymbol{P}_{1_c}$ denote the output of the 20 layer CNN module associated with color channel c, while $\boldsymbol{I}_c$ is the corresponding input image. We then compute

$$\alpha_c = \arg\min_{\alpha_c} \|\boldsymbol{I}_c - \alpha_c \boldsymbol{P}_{1_c}\|_2 = \frac{\boldsymbol{P}_{1_c} \cdot \boldsymbol{I}_c}{\boldsymbol{P}_{1_c} \cdot \boldsymbol{P}_{1_c}} \qquad (2)$$

via a dedicated scaling layer (where $\cdot$ indicates a dot product) and output the updated image prediction $\boldsymbol{P}_{2_c} = \alpha_c \boldsymbol{P}_{1_c}$. As a small caveat, we also threshold values of α_c outside of the range $[0.7, 1.3]$ during training when the predicted images may still be radically different from $\boldsymbol{I}_c$. Note that network gradients are easily backpropagated through such a layer. Also for testing, we discard any such threshold. A related idea has been incorporated into a classical, optimization-based intrinsic image decomposition [16], but this represents a decidedly different context from our CNN model. In any event, a channel-wise color correction layer is a simple but very effective addition to our pipeline, and we find that it reduces negative color difference effects by a wide margin.

Gradient Supervision: Most image smoothing algorithms seek to more-or-less achieve a piece-wise constant result. Certainly the albedo component of an intrinsic image decomposition problem is well-approximated by such an effect. But smoothing a large region into a single color is a challenging task for convolution since the kernel weights are shared over the whole feature map, but local color values can vary dramatically across regions. To help mitigate this problem, we supervise the x and y gradients $\nabla_x \boldsymbol{P}_2$,$\nabla_y \boldsymbol{P}_2$ from the color-normalized images $\boldsymbol{P}_2$ (across all color channels) using a standard auxiliary MSE loss function.

3.2. Edge Refinement and Domain Transform

The images generated by the inference net described above already demonstrate excellent numerical performance. But we observe that some modest color changes still exist in regions that piece-wise constant filters like [5, 32, 34] are able to smooth away. Inspired by [14, 8], we address this lingering issue via a domain transform guided by refined edge estimates. The domain transform is an edge-preserving processing step that admits efficient implementation via separable 1-D recursive filtering layers applied across rows and columns in an image. Two inputs are required: the raw input signal to be filtered, which corresponds with our predicted image $\boldsymbol{P}_2$ from the previous layer, and a domain transform density map that differentiates which regions to filter and which to leave unaltered.

This mapping is produced by a small edge refinement subnetwork in our pipeline. The core idea originates from [14], where the density map is simply the original input image gradient. Later [8] experimented with more sophisticated predicted edge maps applied in conjunction with image segmentation scores. However, their pipeline addresses a completely different task, and the embedded, application-specific edge map requires the concatenation of features culled from intermediate CNN layers. In contrast, since we already generate reasonable edge-preserving images from our base inference net, we do not require this degree of sophistication. Hence our proposed edge refinement subnetwork operates as follows. (Further technical details about how to implement domain transforms within a neural network architecture, including gradient backpropagation steps, can be found in [8].)

The edge refinement module consists of two stages. The first computes, for every point in the input $\boldsymbol{P}_2$, the averaged absolute differences between each of its four neighboring pixels, which is then further averaged across all color channels to produce a single value. The resulting map is next refined using a very shallow network containing 3 convolutional layers. The feature maps for the first two convolution layers have 64 channels. The kernel size for the 1st and 3rd layer is 15×15, zero padded with 7 pixels on each side to maintain spatial dimensions, while the kernel size for the 2nd layer is 1×1, which computes a weighted average of processed pixel vectors for the smoothing operation. The first two convolution layers are followed with ReLU units, while the third is not.

3.3. Modifications for Intrinsic Images

Finally we conclude this section by describing special accommodations for intrinsic image decompositions. Here the inference net is split into two branches after the middle convolutional layer in order to predict albedo and shading images simultaneously. We also observe that (1) can be leveraged as a useful prior for intrinsic image decomposition, enforcing that the estimated decomposition must recombine to approximate the original image $\boldsymbol{I}$ (to the extent that the diffusive, Lambertian model is accurate). There-

fore we include an additional term that penalizes deviations of the estimated $\boldsymbol{A} \odot \boldsymbol{S}$ from $\boldsymbol{I}$, which helps to balance the error between albedo and shading images.

Additionally, intrinsic image decompositions often require potentially larger receptive fields since accumulated soft shadows sharing the same albedo could cover a wide region. Consequently we extend the depth to 42 convolutional layers and downsample intermediate features maps by $1/2$ which saves half the training time. And because albedo and shading images need not have a very close color similarity to the input image, we can remove the color normalization layer for training an intrinsic image decomposition network. For the detailed structure of this modified network, please refer to the supplementary material.

4. Training Details

We begin with the basic image smoothing network depicted in Figure 1. The embedded inference net requires supervision on 4 outputs: the predicted image $\boldsymbol{P}_1$ directly from the CNN, the color-normalized image $\boldsymbol{P}_2$, and computed gradients in x and y directions, i.e., $\nabla_x \boldsymbol{P}_2$, $\nabla_y \boldsymbol{P}_2$. With the $*$ symbol denoting ground truth, the inference net objective terms become

$$\begin{aligned} \boldsymbol{l}_1(\theta) &= \boldsymbol{w}_1 ||\boldsymbol{P}_1 - \boldsymbol{P}_1^*||_2^2 + \boldsymbol{w}_2 ||\boldsymbol{P}_2 - \boldsymbol{P}_2^*||_2^2 \\ &+ \boldsymbol{w}_3 (||\nabla_x \boldsymbol{P}_2 - \nabla_x \boldsymbol{P}_2^*||_2^2 + ||\nabla_y \boldsymbol{P}_2 - \nabla_y \boldsymbol{P}_2^*||_2^2) \end{aligned} \quad (3)$$

where θ denotes the parameter set of the network layers and ℓ_2 norm penalties are chosen since they favor high peak signal-to-noise ratios (PSNR), a common evaluation criteria for image estimation problems. After training inference net in isolation, we then learn an independent network for edge refinement, whose output is denoted $\boldsymbol{E}$ with corresponding loss function $||\boldsymbol{E} - \boldsymbol{E}^*||_2^2$. Finally, we jointly train the whole network using

$$\boldsymbol{L}(\theta) = \boldsymbol{l}_1(\theta) + \boldsymbol{w}_3 ||\boldsymbol{E} - \boldsymbol{E}^*||_2^2 + \boldsymbol{w}_4 ||\boldsymbol{P}_3 - \boldsymbol{P}_3^*||_2^2, \quad (4)$$

where $\boldsymbol{P}_3$ is the output of the domain transform, and noting that, as a gradient proxy we retain the same weighting parameter $\boldsymbol{w}_3$ for $\boldsymbol{E}$ as used in Eq. 3 for coordinate-wise gradients.

For the intrinsic image network, the albedo and shading branches share the same loss function as introduced above, referred to as $\boldsymbol{L_a}(\theta)$ and $\boldsymbol{L_s}(\theta)$, except for the removal of supervision on the color-normalized image $\boldsymbol{P}_2$ and with analogous gradient supervision now denoted to $\boldsymbol{P}_1$. We also include supervision that reflects the constraint from Eq. 1, giving the final cost

$$\boldsymbol{L}(\theta) = \boldsymbol{L_a}(\theta) + \boldsymbol{L_s}(\theta) + \boldsymbol{w}_5 ||\boldsymbol{A}_1 \odot \boldsymbol{S}_1 - \boldsymbol{A}^* \odot \boldsymbol{S}^*||_2^2, \quad (5)$$

where $\boldsymbol{A}_1$ and $\boldsymbol{S}_1$ are the estimates obtained from the inference net analogous to $\boldsymbol{P}_1$.

We implement all networks within the Torch framework using mini-batch gradient descent, with batch size fixed at 16. The energy function weights from $\boldsymbol{w}_1$ to $\boldsymbol{w}_5$ are empirically determined as 0.2, 0.1, 0.35, 0.2, and $0.2/255$ respectively. Note that these weights are used for all smoothing filters and all intrinsic image decompositions, and hence are quite robust; manual tuning in an application-specific manner is not required. Convolutional layers are initialized using the methods from [17]. To train inference net, we use ADAM [21] to accelerate convergence. The learning rate is initially set to 0.01 and then reduced to 0.001 to generate further improvement. For edge refinement, we use simple stochastic gradient descent [23] with learning rate set to 0.01. And for fine-tuning the whole network together, we again use ADAM with a small learning rate of $1e$-5.

5. Experimental Results

This section showcases the numerical and visual advantages of our approach. Further experiments and thorough ablation studies are deferred to the supplementary file.

5.1. Image Smoothing

Dataset: As a low-level vision task, many corpora of natural images are suitable for training and evaluating smoothing algorithms. Here we use the around 17000 natural images from public PASCAL VOC dataset [11] as input, and the corresponding images filtered by traditional smoothing algorithms as the labels/supervision. These images were collected from the the flickr photo-sharing website and are in a wide range of viewing conditions, the same data source used by [33] with which we compare. We randomly crop images at a size of 224×224 pixels for a single image to generate about 17,000 patches. To evaluate our performance, we randomly pick 100 images to form our VOC test split. Additional test results using the BSDS500 dataset [1] are similar (see supplementary).

Comparisons: We approximate 8 different image smoothing filters including the bilateral filter (BLF) [27], iterative bilateral filter (IBLF) [13], L_0 smoothing (L_0) [32], rolling guidance filter (RGF) [35], RTV texture smoothing (RTV) [34], weighted least square smoothing (WLS) [12], weighted median filter (WMF) [36] and L_1 smoothing (L_1) [5]. Compared to 7 public models released by [33], we show much better results in Table 1. Likewise, Table 3 presents comparisons with the recent work from [25]. However, their released code at the time of this writing, which is built upon a multiscale decomposition, cannot run on arbitrary image size, so we test on their default image size 256×256. Even though we do not apply any special tuning for this size, or train on multiscale images as in Table 3, we still achieve better results on average in their setting. And our consistently higher SSIM values indicate that our learned filters are quite effective in generating perceptually good images. Addi-

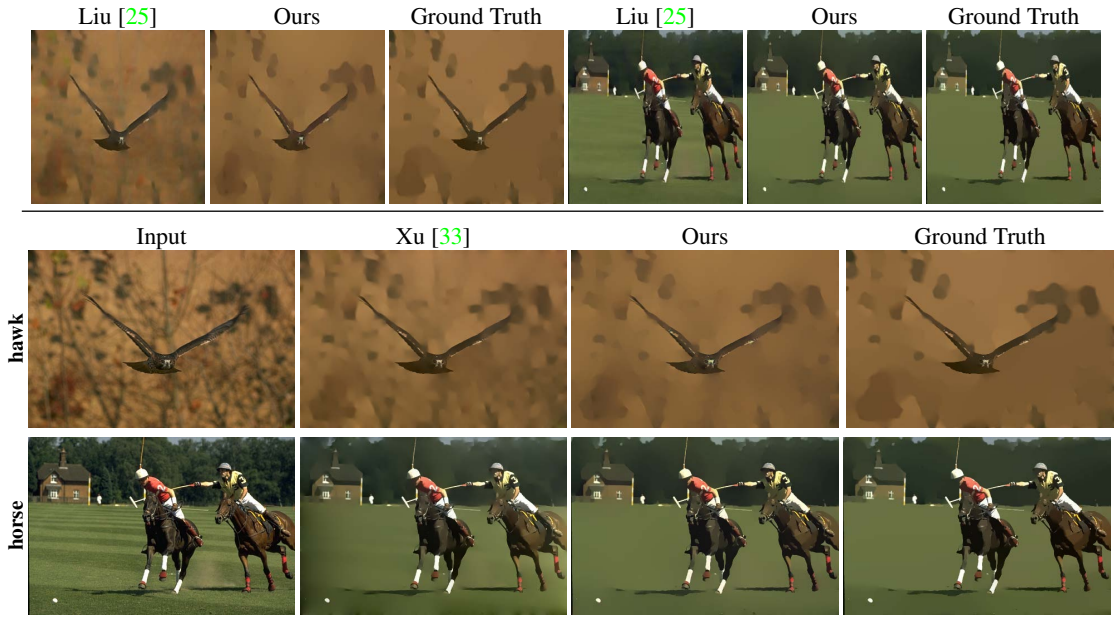

Figure 2. **Qualitative Comparison on Image Smoothing Task**. *Top*: Comparison with Liu *et al.* [25] on the 256×256 image size. *Bottom*: Comparison with Xu *et al.* [33]. All the results are trained on L_0 filter. Zoom in to watch the difference.

		BLF	IBLF	L_0	RGF	RTV	WLS	WMF	L_1	Ave.
PSNR	Xu [33]	35.02	32.97	31.66	32.49	35.68	33.92	29.62		32.62
	Ours	38.34	35.07	33.56	38.62	36.02	38.03	35.75	32.43	36.48
SSIM	Xu [33]	0.976	0.962	0.966	0.950	0.974	0.963	0.960		0.964
	Ours	0.987	0.980	0.981	0.987	0.983	0.985	0.980	0.950	0.983

Table 1. **Quantitative Comparison on Image Smoothing Task**. We report PSNR and SSIM error metrics (larger is better) on 8 different smoothing filters compared with Xu *et al.* [33]. The average value is computed among the front 7 filters. Our results outperform our competitor by a large margin.

tionally, we note that WMF can generate relatively blurry images compared to other filters with the parameterization used in these experiments taken from [33], and yet our network handles this case quite well in both tables relative to other approaches. This is likely because previous methods more-or-less directly rely on sparse structures and may have trouble reproducing the softer transitions required by this WMF filter. Visual comparisons with the WMF filter are in the supplementary material.

We next display some visual comparisons in Figure 2 using the L_0 filter, which produces effects more on the piecewise constant end of the smoothing spectrum. Here we observe that both [33] and [25] have some trouble reproducing constant regions, and [33] has difficulty with erroneous gradient estimates propagating from image boundaries (see horse image). Finally, we evaluate the testing time with traditional edge-aware filters and learned deep networks all on the same computer with public codes and models. Our network achieves favourable speeds and is much faster than most traditional image smoothing methods (see Table 2).

5.2. Intrinsic Image Decomposition

Datasets: We follow two recent state-of-the-art deep learning based methods [31, 20] and evaluate our algorithm on the MPI-Sintel dataset [7] that facilates scene-level quantitative comparisons. It consists of 890 images from 18 scenes with 50 frames each (except for one that contains 40 images). Due to limited images in this dataset, we randomly crop 10 different patches of size 300×300 from one image to generate 8900 patches. Like [31], we use two-fold cross validation to obtain all 890 test results with two trained models. We evaluate our results on both a *scene split*, where half the scenes are used for training and the other half for testing, and an *image split*, where all 890 images are randomly separated into two parts. Clearly the *scene split* is less susceptible to inflated results from overfitting, since images in the same sequence has a big overlap with each other while images in different scenes do not.

While investigating the MPI-Sintel dataset online, we noticed that there are actually two sources for the input and albedo images. The first one is obtainable by emailing

methods	BLF	IBLF	RGF	L_0	WMF	RTV	WLS	L_1	Xu [33]	Liu [25]	Ours
QVGA (320×240)	0.03	0.11	0.22	0.17	0.62	0.41	0.70	32.18	0.23	0.07	0.06
VGA (640×480)	0.12	0.40	0.73	0.66	2.18	1.80	3.34	212.07	0.76	0.14	0.20
720p (1280×720)	0.34	0.97	1.87	2.43	4.98	5.74	13.26	904.36	2.16	0.33	0.51

Table 2. **Running Time Comparison**. Timing comparisons against different traditional filters and deep learning-based counterparts from Xu *et al.* [33] and Liu *et al.* [25] at various resolutions.

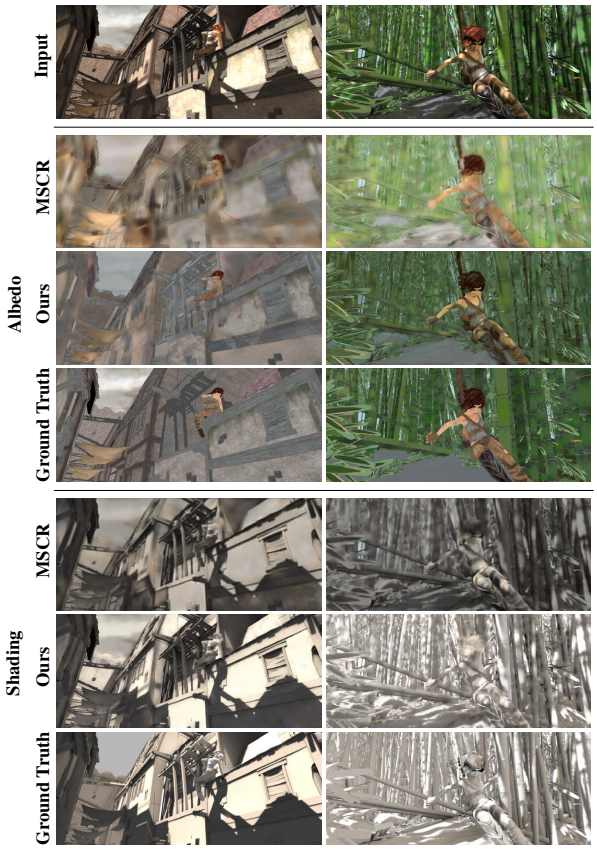

Figure 3. **Qualitative Comparison on main MPI-Sintel Benchmark**. The visual results are evaluated on the model trained on the more difficult *scene split*.

the authors of [7] directly, which we call main MPI-Sintel dataset given that it appears to be used by most previous methods [9, 31, 24]. But there is also a second, ostensibly lesser-known set that can be partially downloaded from their official web page (but also requires emailing to obtain full ground-truth). We refer to this version as the auxilliary MPI-Sintel dataset, which is used by [20]. Defective images in these data stemming from rendering issues are removed for evaluation purposes following previous methods [31].

	PSNR		SSIM	
	Liu [25]	Ours	Liu [25]	Ours
L_0	32.26	33.97	0.958	0.980
RGF	38.64	38.15	0.986	0.987
WLS	38.29	37.29	0.983	0.986
WMF	33.29	36.68	0.951	0.982
Ave.	35.64	36.52	0.966	0.984

Table 3. Comparison with recent work Liu *et al.* [25] on image smoothing task. Experiments are conducted using their four released trained filters.

Finally, to test performance on real images where scene-level ground-truth is unavailable, we also use the 220 images in the MIT intrinsic dataset [16] as in [31]. This data contains only 20 different objects, each of which has 11 images. To compare with previous methods, we train our model using 10 objects via the split from [3], and evaluate the results on images from the remaining objects.

Comparison: To quantitatively evaluate the performance, we use three criteria, mean-squared error (MSE), local mean-squared error (LMSE), and the dissimilarity version of the structural similarity index (DSSIM) as proposed in [9]. First, in Tables 4 and 5, our model shows much better results on the MPI-Sintel data. Note that the very recent work JCNF [20] benefits from training on additional depth map data, and as a gradient domain method needs to execute a separate post-processing step to reconstruct images. MSCR [31] also benefits from additional training data from the MIT intrinsic dataset.

We show 2 groups of qualitative results trained on the more difficult *scene split* in Figure 3. Note that we are able to predict much sharper and relatively high-quality results even though we have only around 500 training images (and no outside training information as with other methods), and training and testing data do not have any overlap. For more visual results of *image split* or *scene split*, please refer to the supplementary material.

Next, Table 6 presents relative performance using MIT intrinsic data. Here we observe that our approach is significantly better than the other deep networks [29, 31] even though [31] utilizes additional MPI-Sintel data and [29] incorporates additional rendered, large-scale single-object intrinsic images to assist in training. Note that [3] uses a number of specialized priors appropriate for this simplified object-level data, while end-to-end CNN approaches like ours and [31] have less advantage here due to limited training data (110 images). Moreover, [3] is not competitive on other more complex, scene-level data types as

		MSE			LMSE			DSSIM		
		albedo	shading	average	albedo	shading	average	albedo	shading	average
image split	Retinex [16]	0.0606	0.0727	0.0667	0.0366	0.0419	0.0393	0.2270	0.2400	0.2335
	Barron *et al.* [3]	0.0420	0.0436	0.0428	0.0298	0.0264	0.0281	0.2100	0.2060	0.2080
	Chen *et al.* [9]	0.0307	0.0277	0.0292	0.0185	0.0190	0.0188	0.1960	0.1650	0.1805
	MSCR [31]	0.0100	0.0092	0.0096	0.0083	0.0085	0.0084	0.2014	0.1505	0.1760
	Ours	0.0067	0.0060	0.0063	0.0041	0.0042	0.0041	0.1050	0.0783	0.0916
scene split	MSCR [31]	0.0190	0.0213	0.0201	0.0129	0.0141	0.0135	0.2056	0.1596	0.1826
	Ours	0.0181	0.0175	0.0178	0.0122	0.0118	0.0120	0.1674	0.1382	0.1528

Table 4. **Quantitative Comparison on main MPI-Sintel Benchmark**. We evaluate our results using both scene and image splits across three standard error rates of intrinsic images on the main MPI-Sintel dataset.

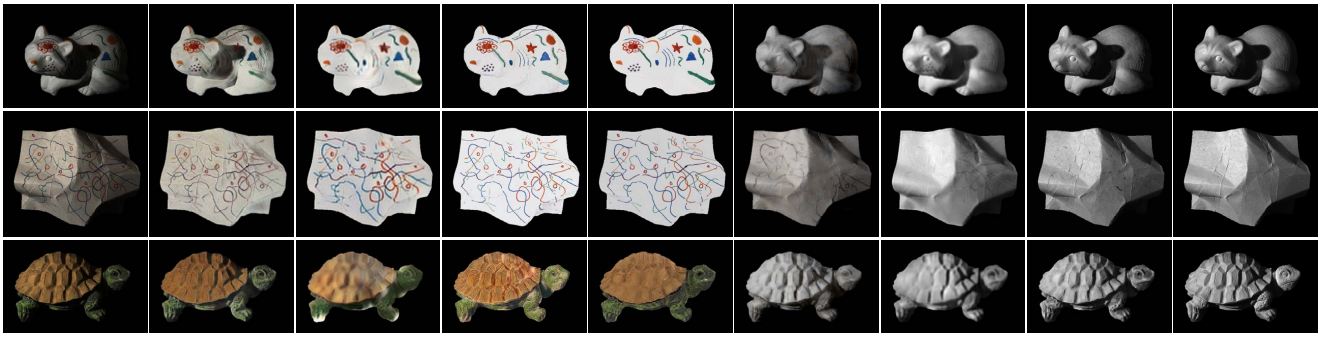

Figure 4. **Qualitative Comparison on MIT Intrinsic Benchmark**. Compared with Shi *et al.* [29] and MSCR [31] on Barron *et al.*'s test split of MIT intrinsic dataset, our algorithm achieves the best/sharpest results.

shown in Table 4. In Figure 4, our predicted images are also much sharper and more accurate than the other deep learning methods. Finally, we should note that [38] trains a model to predict relative reflectance orderings between image patches and integrates the learned prior within an existing optimization framework for real-world intrinsic image decomposition. We tested their trained model only on the MIT intrinsic data due to the slow running time, and it was not comparable to other methods.

6. Conclusion

This paper proposes a deep network for image smoothing and intrinsic image decomposition tasks. Design features include a deep CNN with large receptive fields operating in the original color domain, a color normalization step, gradient supervision, and a domain transform for refining final image estimates. The resulting pipeline produces edge-preserving smooth images or intrinsic decompositions without any preprocessing, postprocessing, application-specific tuning parameters, or special requirements of input image size. We have also numerically demonstrated state-of-the-art performance on important representative benchmarks.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011. 5

[2] J. T. Barron and J. Malik. Intrinsic scene properties from a single rgb-d image. *CVPR*, 2013. 1

[3] J. T. Barron and J. Malik. Shape, illumination, and reflectance from shading. *PAMI*, 37(8):1670–1687, 2015. 7, 8, 9

[4] S. Bell, K. Bala, and N. Snavely. Intrinsic images in the wild. *ACM Trans. on Graphics (SIGGRAPH)*, 33(4), 2014. 1

[5] S. Bi, X. Han, and Y. Yu. An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics*, 34(4):78, 2015. 1, 2, 4, 5

[6] N. Bonneel, K. Sunkavalli, J. Tompkin, D. Sun, S. Paris, and H. Pfister. Interactive intrinsic video editing. *ACM Transactions on Graphics (TOG)*, 33(6):197, 2014. 1

[7] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *ECCV*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 3, 6, 7

[8] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In *CVPR*, 2016. 2, 4

		MSE			LMSE			DSSIM		
		albedo	shading	average	albedo	shading	average	albedo	shading	average
image split	JCNF [20]	0.0070	0.0090	0.0080	0.0060	0.0070	0.0065	0.0920	0.1010	0.0970
	Ours	0.0043	0.0057	0.0050	0.0032	0.0042	0.0037	0.1012	0.0827	0.0919

Table 5. **Quantitative Comparison on auxilliary MPI-Sintel Benchmark**. Similar results to Table 4, except using the auxiliary data. Note that JCNF [20] is only trained and tested on the *image split* of MPI-Sintel dataset, hence our exclusion of the scene split here.

	MSE			LMSE
	albedo	shading	average	total
Zhou *et al.* [38]	0.0252	0.0229	0.0240	0.0319
Barron *et al.* [3]	0.0064	0.0098	0.0081	0.0125
Shi *et al.*[29]	0.0216	0.0135	0.0175	0.0271
MSCR [31]	0.0207	0.0124	0.0165	0.0239
Ours	0.0127	0.0085	0.0106	0.0200

Table 6. **MIT Intrinsic Data Results**. Performance of various methods on Barron *et al.*'s test set [3]. LMSE is computed using an error metric specifically designed for this data [16]. Note also that Barron *et al.*'s approach [3] relies on specialized priors and masked objects particular to this dataset. This approach does not work well on scene level data (see Table 4).

[9] Q. Chen and V. Koltun. A simple model for intrinsic image decomposition with depth cues. In *ICCV*, 2013. 1, 7, 8

[10] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, pages 2650–2658, 2015. 3

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, June 2010. 5

[12] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2008)*, 27(3), Aug. 2008. 1, 5

[13] R. Fattal, M. Agrawala, and S. Rusinkiewicz. Multiscale shape and detail enhancement from multi-light image collections. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3), Aug. 2007. 1, 5

[14] E. S. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. In *ACM Transactions on Graphics (TOG)*, volume 30, page 69. ACM, 2011. 2, 4

[15] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *ICML*, pages 399–406, 2010. 1

[16] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *ICCV*, pages 2335–2342. IEEE, 2009. 4, 7, 8, 9

[17] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015. 5

[18] L. Karacan, E. Erdem, and A. Erdem. Structure-preserving image smoothing via region covariances. *ACM Trans. Graph.*, 32(6):176:1–176:11, 2013. 1

[19] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 2

[20] S. Kim, K. Park, K. Sohn, and S. Lin. Unified depth prediction and intrinsic image decomposition from a single image via joint convolutional neural fields. In *ECCV*, 2016. 3, 6, 7, 9

[21] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 5

[22] E. H. Land and J. J. McCann. Lightness and retinex theory. *J. Opt. Soc. Am.*, pages 1–11, 1971. 2

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5

[24] L. Lettry, K. Vanhoey, and L. Van Gool. Darn: a deep adversial residual network for intrinsic image decomposition. *arXiv preprint arXiv:1612.07899*, 2016. 3, 7

[25] S. Liu, J. Pan, and M.-H. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *ECCV*, 2016. 2, 3, 5, 6, 7

[26] S. H. J. K. M. Aubry, S. Paris and F. Durand. Fast local laplacian filters: Theory and applications. *ACM Transactions on Graphics*, 2014. 1

[27] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *ECCV*, pages 568–580, 2006. 1, 5

[28] L. Shen and C. Yeo. Intrinsic images decomposition using a local and global sparse representation of reflectance. In *CVPR*, pages 697–704. IEEE, 2011. 1

[29] J. Shi, Y. Dong, H. Su, and S. X. Yu. Learning non-lambertian object intrinsics across shapenet categories. *CVPR*, 2017. 3, 7, 8, 9

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4

[31] M. M. Takuya Narihira and S. X. Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *ICCV*, 2015. 3, 6, 7, 8, 9

[32] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via L_0 gradient minimization. In *SIGGRAPH Asia*, 2011. 1, 2, 4, 5

[33] L. Xu, J. S. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *ICML*, pages 1669–1678, 2015. 1, 2, 3, 4, 5, 6, 7

[34] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via natural variation measure. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2012. 1, 4, 5

[35] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *ECCV*, pages 815–830, 2014. 1, 5

[36] Q. Zhang, L. Xu, and J. Jia. 100+ times faster weighted median filter. In *CVPR*, 2014. 1, 5

[37] Q. Zhao, P. Tan, Q. Dai, L. Shen, E. Wu, and S. Lin. A closed-form solution to retinex with nonlocal texture constraints. *PAMI*, 34(7):1437–1444, 2012. 1

[38] T. Zhou, P. Krahenbuhl, and A. A. Efros. Learning data-driven reflectance priors for intrinsic image decomposition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3469–3477, 2015. 8, 9