Location-aware Single Image Reflection Removal

Zheng Dong[1], Yin Yang[2], Hujun Bao[1], Weiwei Xu[*1], Rynson W.H. Lau[3]
[1]Zhejiang University [2]Clemson University [3]City University of Hong Kong
zhengdong@zju.edu.cn, yin5@clemson.edu, {bao, xww}@cad.zju.edu.cn, rynson.lau@cityu.edu.hk

Abstract

This paper proposes a novel location-aware deep learning-based single image reflection removal method. Our network has a reflection detection module to regress a probabilistic reflection confidence map, taking multi-scale Laplacian features as inputs. This probabilistic map tells whether a region is reflection-dominated or transmission-dominated. The novelty is that we use the reflection confidence map as the cues for the network to learn how to encode the reflection information adaptively and control the feature flow when predicting reflection and transmission layers. The integration of location information into the network significantly improves the quality of reflection removal results. Besides, a set of learnable Laplacian kernel parameters is introduced to facilitate the extraction of discriminative Laplacian features for reflection detection. We design our network as a recurrent network to progressively refine each iteration's reflection removal results. Extensive experiments verify the superior performance of the proposed method over state-of-the-art approaches.

1. Introduction

Reflections often occur when an image is photographed through reflective and transparent media (*e.g.*, glass). Removing undesired reflections enhances the image quality and benefits many follow-up computer vision tasks, such as image classification, etc. In reflection removal, an image $\mathbf{I}$ with reflections can be modeled as the weighted additive composition of a transmission layer $\mathbf{T}$ and a reflection layer $\mathbf{R}$. Precisely, following the linear synthesis model in [5, 18, 45], we express the composition procedure as follows:

$$\mathbf{I} = \mathbf{W} \circ \mathbf{T} + \mathbf{R}, \tag{1}$$

where $\mathbf{W}$ here is extended to be a weight map that represents the variation of the attenuation of the transmission layer, an optical effect related to Fresnel's law.

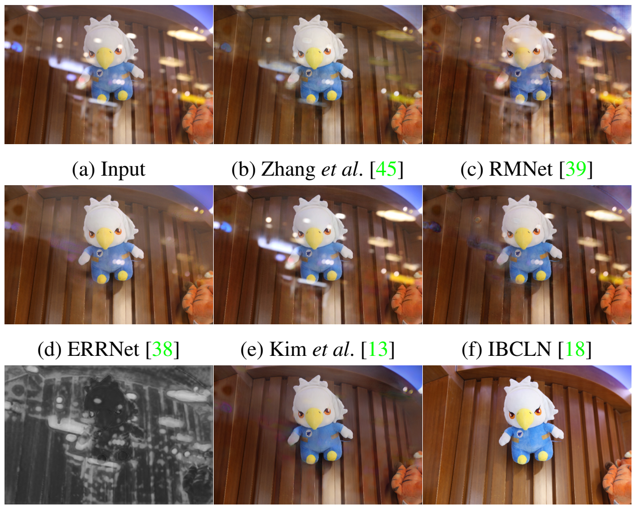
(a) Input (b) Zhang *et al.* [45] (c) RMNet [39]
(d) ERRNet [38] (e) Kim *et al.* [13] (f) IBCLN [18]
(g) RCMap (h) Ours (i) Ground truth

Figure 1. State-of-the-art methods (b,c,d,e,f) typically fail to recover high-quality transmission layers from strong reflections, *e.g.*, the highlights. Our method addresses this problem by learning the reflection confidence map (RCMap) for the detection of the reflection-dominated regions (g) and reflection removal (h). The input image (a) is obtained from [38].

The task of single image reflection removal (SIRR) is to recover $\mathbf{T}$ from a given image $\mathbf{I}$. It is an ill-posed problem since the number of unknowns is much more than the number of equations derived from Eq. 1. Therefore, priors are necessary to constrain the solution space, such as natural image gradient sparsity [16, 17], ghosting cues for thick glasses [27], and relative smoothness that assumes the refection layer is smoother than the transmission layer [20, 43]. To disambiguate the restoration of $\mathbf{T}$ and $\mathbf{R}$ in the gradient domain, several works propose first to determine the locations of reflection-dominated and transmission-dominated pixels, and then exploit different constraints at different locations to improve the reflection removal results [16, 33, 35]. While these methods are sensitive to the selection of hyperparameters, for instance, the commonly used gradient magnitude threshold, the detected location information is proven to be useful to handle strong reflections. However, such location information

*Corresponding author. The authors from Zhejiang University are affiliated with the State Key Lab of CAD&CG.

is not explicitly investigated in deep learning-based SIRR methods [5, 12, 13, 18, 38, 39, 42, 45]. In other words, there is no clue for the networks to be aware of how the information of reflections is encoded in features to facilitate the reflection removal. It might cause ambiguities when strong reflections (*e.g.*, reflected highlights) appear. We observe that state-of-the-art SIRR methods typically fail to recover high-quality transmission layers.

This paper proposes a location-aware deep learning-based SIRR method for generic refection removal. Our network incorporates a novel reflection detection module (RDM) to detect reflection-dominated regions via learning multi-scale Laplacian features. The usage of the Laplacian features here is motivated by the assumption that the characteristics of transmission and reflection layers in the gradient domain are different from each other, such as the relative smoothness prior in [20]. The output of RDM is a probabilistic reflection confidence map (RCMap), which controls the subsequent feature flow, resulting in significantly improved SIRR results. The inverse RCMap, *i.e.*, 1 - RCMap, can serve as the weight map $\mathbf{W}$ in Eq. 1 and be used to indicate the transmission-dominated regions. It motivates us to use Eq. 1 as a loss to control the RDM training.

Our proposed network iteratively restores the transmission layer from the corrupted input. In each iteration, we formulate the restoration process in a reflection removal-by-detection manner. It first detects the reflection-dominated regions based on the RCMap. Afterward, it predicts the whole reflection layer by suppressing the transmission information and restores the transmission layer by jointly leveraging the transmission-dominant regions and the predicted reflection layer. Such a network design is motivated by the alternating optimization strategy [2], which tries to decompose the complicated SIRR problem of Eq. 1 into easy-to-solve sub-problems and take the mutual dependence between reflection and transmission into consideration. As illustrated in Fig. 1, our network can effectively restore the transmission layer for an image with strongly reflected highlights.

In summary, the main contributions of our work are:

- We propose a novel SIRR method that iteratively restores the transmission layer from the corrupted input image. At each iteration, the restoration process is formulated in a removal-by-detection sequential manner.
- We propose a recurrent neural network, which incorporates a novel reflection detection module (RDM) to detect the reflection-dominated regions. It learns a group of multi-scale Laplacian kernel parameters to exploit reflection boundary information.
- Extensive experiments verify the superior performance of the proposed method over state-of-the-art approaches by a considerable margin.

2. Related Work

A variety of reflection removal methods, such as multi-view or video-based methods [6, 8, 19, 21, 26, 30, 31, 32, 41], dual-pixel sensor [23] and polarization-based [3, 15, 28] methods, have been proposed to reconstruct the background transmission layer through motion or optical cues. Since we focus on SIRR in this paper, we mainly review the works mostly related to ours in this section.

To handle the ill-posed SIRR problem, researchers proposed different priors to constrain the solution space. Work in [17] leveraged the sparsity prior of gradients when decomposing an image into reflection and transmission layers. Observing that reflection layers are usually out of focus and appear to be more blurry than transmission layers, Li *et al*. [20] proposed to model the gradient distribution of these two layers with different probability distributions, resulting in good quality reflection removal results. When the thickness of the glass cannot be ignored, ghosting reflection could appear due to refraction. Shih *et al*. [27] employed a patch-based GMM prior distribution to model the natural image for reflection removal in this case. Some methods suppressed the blurry reflections using gradient-domain thresholding and image reconstruction methods [1, 43]. However, when there are strong reflection areas in the scene, these methods cannot effectively remove them and might smooth out the details in the transmission layer. It is also beneficial to separate the gradients of the original image into reflection gradients and transmission gradients to provide gradient-domain constraints to facilitate the layer separation in SIRR [16]. Detecting reflection-dominated regions in an image can be achieved through depth-of-field analysis [33, 35]. Our method is motivated by these methods but relies on deep learning to improve the robustness of reflection detection. Moreover, in contrast to the binary map in [33, 35], our RCMap is a probabilistic map used to suppress the unrelated features in reflection prediction.

Recently, deep learning-based SIRR methods have become popular. These methods are data-driven and try to learn task-specific features to solve the SIRR problem in feature space. Fan *et al*. [5] designed a deep neural network, CEIL-Net, to first regress the edge map of the transmission layer and then reconstruct the transmission layer. BDN [42] is also a two-stage network, where the reflection layer predicted in the first stage is used as auxiliary information to guide the transmission layer reconstruction in the second stage. IBCLN [18] proposed a recurrent network based on LSTM [10] units to refine the results of predicted reflection and transmission layers iteratively. CoRRN [37] proposed a network with a feature-sharing strategy and a statistic loss to remove the strong reflections within local regions. Face reflection removal has also been studied in [36]. Besides, various loss terms have been proposed to guided the deep

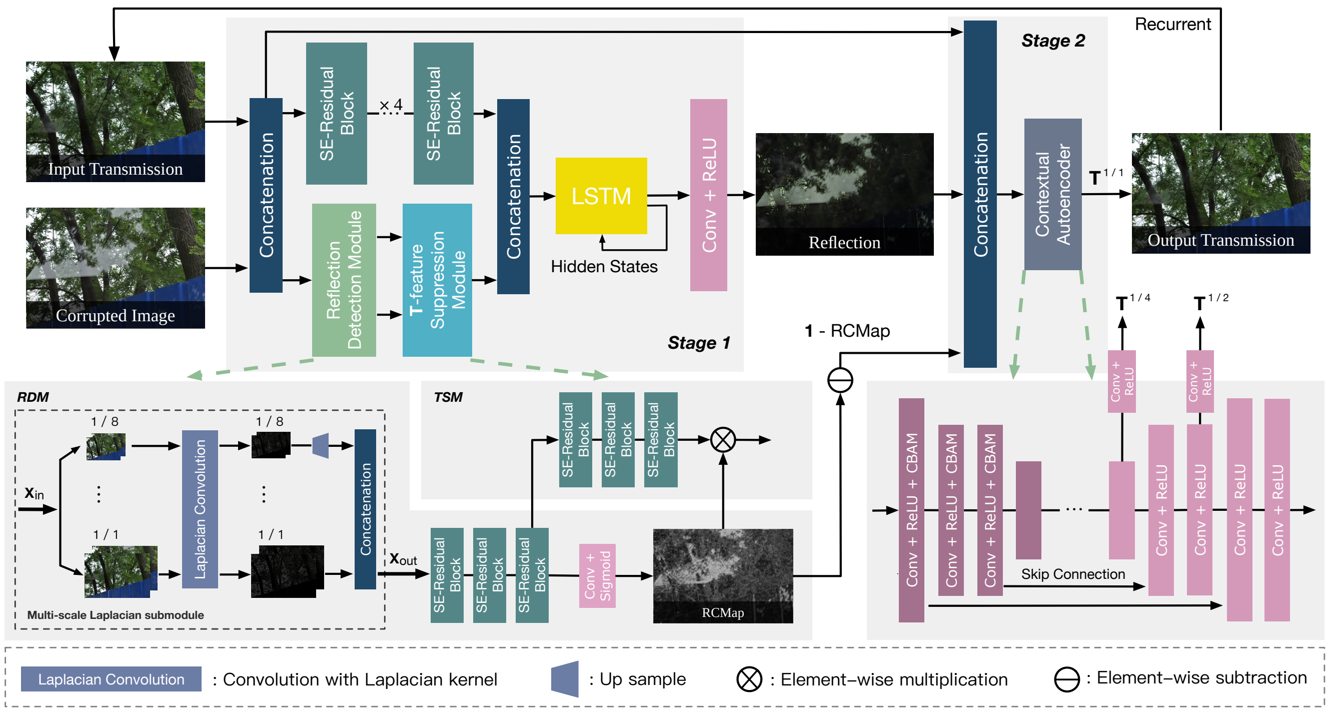

Figure 2. The architecture of our recurrent SIRR network. Stage 1: predict the RCMap and the reflection layer. “x4” indicates that the SE (Squeeze-and-Excitation) residual block [11] is repeated 4 times. Stage 2: predict the transmission layer. CBAM: Convolutional Block Attention Modules [40]. The output transmission image at iteration $i-1$ will be fed back to the network as the input of iteration i, and $\hat{\mathbf{T}}_0$ is initialized as $\mathbf{I}$.

neural networks to improve the SIRR results, such as perceptual loss based on VGG network, exclusion loss in gradient domain, and adversarial losses to prevent the blurring of reconstructed layers [13, 18, 38, 39, 42, 45].

The construction of the SIRR dataset is critical to the success of deep learning-based reflection removal methods. To this end, Jin *et al.* [12] proposed multiple data generation models. Wen *et al.* [39] proposed SynNet to generate images with reflections beyond linearity. Wei *et al.* [38] introduced an alignment-invariant loss to utilize the misaligned images as the real-world training dataset. Recently, Kim *et al.* [13] proposed a physics-based rending method to render images with reflections, the proposed method considers the reflection and refraction of light in glasses to obtain realistic rendering results.

3. Proposed Method

Our network is a recurrent network, as illustrated in Fig. 2. In each iteration i, our network takes the original image $\mathbf{I}$ and the transmission layer $\hat{\mathbf{T}}_{i-1}$ predicted in the previous iteration $i-1$ as inputs, and predicts the transmission layer $\hat{\mathbf{T}}_i$ to continue the iteration. $\hat{\mathbf{T}}_0$ is initialized as $\mathbf{I}$. The step-by-step refinement results of reflection removal are shown in Fig. 3. This design is inspired by the recurrent network for reflection removal in IBCLN [18], and we also use an LSTM [10] unit to aggregate information through iterations. However, our network is designed to lean towards the reconstruction of transmission and only iterates over the predicted transmission layer $\hat{\mathbf{T}}_i$.

In our network, each iteration is divided into two stages for restoring two layers sequentially, which is in a spirit similar to BDN [42]. However, we rely on RCMap to control the between-stage information flow. In the first stage, we predict the reflection layer $\hat{\mathbf{R}}_i$ and the reflection confidence map $\hat{\mathbf{C}}_i$ by taking $\mathbf{I}$ and $\hat{\mathbf{T}}_{i-1}$ as inputs. We denote the first stage as a function G_R that can be written as:

$$\hat{\mathbf{R}}_i, \hat{\mathbf{C}}_i = G_R(\mathbf{I}, \hat{\mathbf{T}}_{i-1}). \quad (2)$$

This stage mainly consists of two modules: reflection detection module (RDM) and transmission-feature suppression module (TSM). Specifically, RDM takes $\mathbf{I}$ and $\hat{\mathbf{T}}_{i-1}$ as inputs and predicts the map $\hat{\mathbf{C}}_i$ using features from a multi-scale Laplacian sub-module (MLSM). Next, TSM is used to suppress the Laplacian features within transmission-dominated regions via an element-wise multiplication between the features and $\hat{\mathbf{C}}_i$. Afterward, the suppressed features and the images features are concatenated as an LSTM [10] block's inputs to estimate $\hat{\mathbf{R}}_i$. In our work, $\hat{\mathbf{R}}_i$ is mainly used as a cue to improve the reconstruction of transmission layer.

In the second stage, we predict the transmission layer $\hat{\mathbf{T}}_i$

Code and the pre-trained model: https://github.com/zdlarr/Location-aware-SIRR

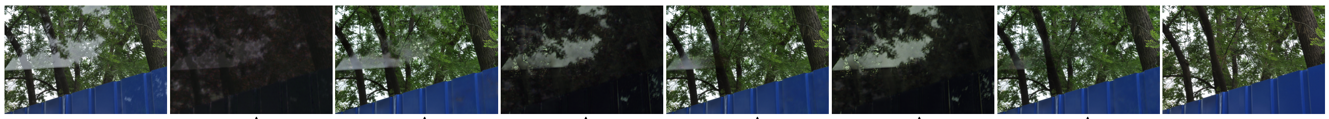
(a) $\mathbf{I}$ (b) $\hat{\mathbf{R}}_1$ (c) $\hat{\mathbf{T}}_1$ (d) $\hat{\mathbf{R}}_2$ (e) $\hat{\mathbf{T}}_2$ (f) $\hat{\mathbf{R}}_3$ (g) $\hat{\mathbf{T}}_3$ (h) $\mathbf{T}$

Figure 3. The gradual refinement of reflection removal results after each iteration. The input image $\mathbf{I}$ is taken in front of a window glass.

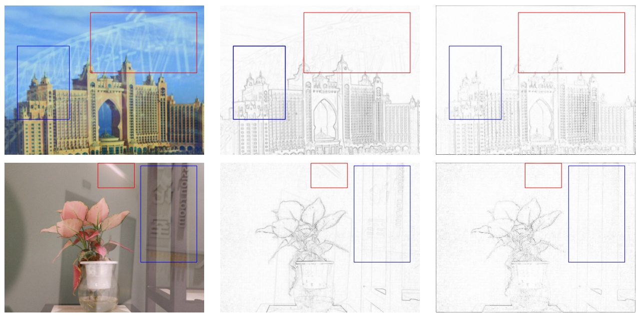
(a) Input (b) Inverse edge map (c) Inverse Laplacian map

Figure 4. Visualization of two inverse edge maps and inverse Laplacian maps along with their original images. For the inverse edge map, we first compute the edge map $\mathbf{E}$ ranging from 0 to 1 using the method in [5] and then obtain its inverse map by $1-\mathbf{E}$. Similar to the inverse Laplacian map, we first compute absolute Laplacian values through convolution with the Laplacian kernel $\mathbf{k}_L$, then divide each Laplacian value by the maximal absolute value of the image to obtain a map $\mathbf{L}$. The inverse Laplacian map is then set to $1-\mathbf{L}$. Consequently, the 0 value of the low-frequency signals in gradient domain is mapped to 1 in the inverse maps. Best view on screen and zoom in.

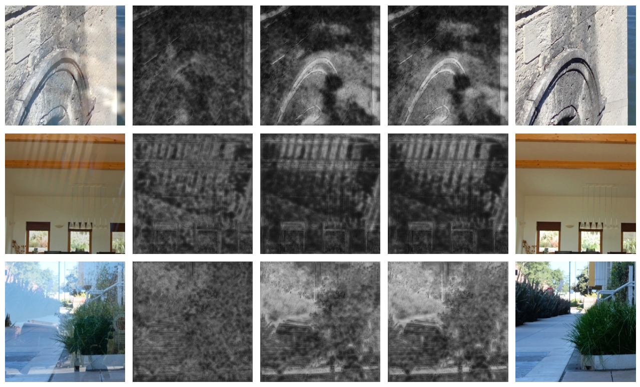
(a) $\mathbf{I}$ (b) Iter = 1 (c) Iter = 2 (d) Iter = 3 (e) $\mathbf{T}$

Figure 5. Visualization of the improvement of the reflection confidence map after each iteration. Note that the reflection-dominated regions are gradually obvious and accurate.

with the input of $\mathbf{I}$, $\hat{\mathbf{T}}_{i-1}$ as well as $\hat{\mathbf{R}}_i, 1-\hat{\mathbf{C}}_i$ computed in the first stage. We denote the second stage as a function G_T which can be described as:

$$\hat{\mathbf{T}}_i = G_T(\mathbf{I}, \hat{\mathbf{T}}_{i-1}, \hat{\mathbf{R}}_i, 1-\hat{\mathbf{C}}_i). \tag{3}$$

Notice that we utilize the inverse confidence map, *i.e.* $1-\hat{\mathbf{C}}_i$, in this stage. Since the transmission layer dominates at regions where $1-\hat{\mathbf{C}}_i$ have a high value, we expect this map to help the network learn weights to encode the reflection information in an adaptive manner, which should benefit the reconstruction of the transmission layer. For the network structure in this stage, we follow the contextual autoencoder network in [24], and additionally leverage CBAM (Convolutional Block Attention Module) [40] blocks after *Conv* and *ReLU* to compute the channel-wise and spatial attention. Please see supplementary materials for detailed network parameters.

Multi-scale Laplacian Features. We observe that the Laplacian operator, a second-order differential operator, can suppress the low-frequency reflections better. As illustrated in Fig. 4, low-frequency reflections are less obvious in the inverse Laplacian map than in the inverse edge map, it suggests that the Laplacian operator more effectively suppresses low-frequency reflections. Besides, it is desirable since we intend to detect strong reflections, and suppressed low-frequency reflections are relatively easier to remove. In contrast, strong reflections that have hard boundaries can not be suppressed by the Laplacian operator. It is possible that the difference between $\mathbf{I}$ and $\mathbf{T}$ caused by strong reflections become more obvious in the Laplacian domain. We assume it is beneficial to detect reflection-dominated regions and thus concatenate these two images to form $\mathbf{X}_{in} = [\mathbf{I}, \hat{\mathbf{T}}_{i-1}]$ as inputs to obtain multi-scale Laplacian features.

For the purpose of multi-scale Laplacian feature learning, we down-sample $\mathbf{X}_{in}$ to original size's $1/2, 1/4, 1/8$ by bi-linear interpolation [7]. The down-sampled results are denoted by $\mathbf{X}_2^{\downarrow}, \mathbf{X}_4^{\downarrow}, \mathbf{X}_8^{\downarrow}$ respectively. We utilize a convolution kernel with weights initialized to be a 3×3 Laplacian kernel, denoted by $\mathbf{k}_L = [0,-1,0;-1,4,-1;0,-1,0]$, to obtain the second derivative signal from $\mathbf{X}_{in}$. We allow the network to fine-tune the Laplacian kernel parameters to better extract Laplacian features, where the fine-tuned parameters are denoted as $\mathcal{L}ap$. During training, we utilize gradient clipping (0.25 in our experiments) to make sure that the learned kernel parameters stay close to the original kernel $\mathbf{k}_L$.

After the Laplacian convolution block in Fig. 2, the upsampling operation $U_j^{\uparrow}$ is applied to the multi-scale Laplacian feature maps to restore their original size, where j is the sampling rate. Precisely, given the images $\mathbf{X}_j^{\downarrow} (j = 2,4,8)$ and $\mathbf{X}_{in}$, the output features $\mathbf{X}_{out}$ can be written as:

$$\mathbf{X}_{out} = Concat(\mathcal{L}ap(\mathbf{X}_{in}), U_j^{\uparrow}(\mathcal{L}ap(\mathbf{X}_j^{\downarrow})))_{j=2,4,8}, \tag{4}$$

where $Concat$ is the concatenation operation.

Finally, taking $\mathbf{X}_{out}$ as input, RDM predicts the reflection confidence map from the Laplacian features. We employ three Squeeze-and-Excitation Residual Block (SE-ResBlocks) [11] to get efficient multi-channel Laplacian features, where each block comprises of three layers of SE-ResNet, then the PReLU function [9] is used to activate the features while keeping the negative values. These combined blocks denotes as $f_{\mathcal{L}ap}$. Thus, given the features $\mathbf{X}_{out}$, the map $\hat{\mathbf{C}}$ can be described as:

$$\hat{\mathbf{C}} = Sigmoid(Conv(f_{\mathcal{L}ap}(\mathbf{X}_{out}))). \tag{5}$$

Improvement of the predicted RCMaps for three training images along with their iteration is illustrated in Fig. 5.

Transmission-feature suppression module. While it is feasible to predict $\hat{\mathbf{R}}_i$ with RCMap $\hat{\mathbf{C}}_i$ as input, just like predicting $\hat{\mathbf{T}}_i$ in the second stage, we empirically found that suppressing the part of Laplacian features belong to transmission-dominated regions benefit the reflection layer prediction. It also leads to a relatively simple network design by concatenating the features computed with $\mathbf{X}_{in}$ and the suppressed Laplacian features as the input to the LSTM [10] block. That is, the same encoder-decoder network structure in the second stage is not used in the first stage to reduce the number of network parameters. Precisely, in this module, we employ three SE-ResBlocks to refine the Laplacian features and then multiply the features by $\hat{\mathbf{C}}_i$ for the purpose of transmission features suppressing.

4. Training Loss

In this section, we describe the five loss functions used in the training of our network. For clarity, we denote the ground-truth transmission and reflection layers by $\mathbf{T}, \mathbf{R}$ respectively, and the predicted transmission and reflection layers at iteration i as $\hat{\mathbf{T}}_\mathbf{i}, \hat{\mathbf{R}}_i$ respectively. The iterations number used in our recurrent network is denoted by N.

Composition Loss. The composition loss is proposed to guide the training of RDM to predict $\hat{\mathbf{R}}$ and $\hat{\mathbf{C}}$. Firstly, since the map $1 - \hat{\mathbf{C}}$ can serve as the weight map $\mathbf{W}$ in Eq. 1, we can compose an image by the following formula:

$$\hat{\mathbf{I}}_i = (1 - \hat{\mathbf{C}}_i) \circ \mathbf{T} + \mathbf{R}, \tag{6}$$

where $\circ$ is an element-wise production operation. Notice that $\mathbf{T}$ and $\mathbf{R}$ are the ground-truth images after Gamma correction [44]. We formulate the loss for $\hat{\mathbf{C}}$ as follows:

$$\mathcal{L}_{\hat{\mathbf{C}}} = \sum_{I,T,R \in \mathcal{D}} \sum_{i=1}^{N} \theta^{N-i} \mathcal{L}_{MSE}(\mathbf{I}, \hat{\mathbf{I}}_i), \tag{7}$$

where $\mathcal{L}_{MSE}$ indicates the mean squared error, θ is an attenuation coefficient to indicate the strength of supervision and we set it to 0.85.

Secondly, same as IBCLN [18], we adopt the linear synthesis model, *i.e.*, $\tilde{\mathbf{I}} = \alpha \cdot \tilde{\mathbf{T}} + \tilde{\mathbf{R}}$, to form a loss to guide the prediction of $\hat{\mathbf{T}}_i, \hat{\mathbf{R}}_i$. It has two forms: $\tilde{\mathbf{I}}_i = \alpha \cdot \mathbf{T} + \tilde{\mathbf{R}}_i$ and $\tilde{\mathbf{I}}_i = \alpha \cdot \tilde{\mathbf{T}}_i + \tilde{\mathbf{R}}_i$, where $\tilde{\mathbf{I}}_i, \tilde{\mathbf{T}}_i$ and $\tilde{\mathbf{R}}_i$ are the results of the inverse gamma correction of $\hat{\mathbf{I}}_i, \hat{\mathbf{T}}_i$ and $\hat{\mathbf{R}}_i$ respectively, and α is a known quantity for the synthesized images (see data augmentation in Sec. 5 for details). We use Eq. 7 to compute the loss for these two forms and denote the loss as $\mathcal{L}_{residual}$ as well. In short, our composition loss for the synthesized images is defined as:

$$\mathcal{L}_{comp} = \mathcal{L}_{\hat{\mathbf{C}}} + \mathcal{L}_{residual}. \tag{8}$$

Perceptual Loss. Same as ERRNet [38] and IBCLN [18], we use VGG-19 network [29] pre-trained on ImageNet [25] dataset to extract features for the computation of this loss. Our perceptual loss takes multi-scale images as inputs, and it can be written into:

$$\mathcal{L}_p = \sum_{T^j \in \mathcal{D}} \sum_{j=1,1/2,1/4} \gamma_j \mathcal{L}_{VGG}(\mathbf{T}^j, \hat{\mathbf{T}}_N^j), \tag{9}$$

where $\mathcal{L}_{VGG}$ denotes the mean square error between VGG features. We use three scales, *i.e.*, $j = 1, 1/2, 1/4$ and set the weights as: $\gamma_1 = 1, \gamma_3 = 0.8, \gamma_5 = 0.6$ respectively. For $\mathcal{L}_{VGG}$, we use the layers 'conv2_2', 'conv3_1', 'conv4_2' and 'conv5_2' as [38]. Fig. 2 shows how the network computes $\hat{\mathbf{T}}_N^j$.

Pixel and SSIM Loss. The pixel loss is used to penalize the pixel-wise difference between $\mathbf{T}$ and $\hat{\mathbf{T}}_i$. Here, we utilize l_1 loss which denotes as $\mathcal{L}_1$ to compute the absolute difference. We define the pixel loss as:

$$\mathcal{L}_{pixel} = \sum_{T \in \mathcal{D}} \sum_{i=1}^{N} \theta^{N-i} \mathcal{L}_1(\mathbf{T}, \hat{\mathbf{T}}_i), \tag{10}$$

where θ is set to 0.85 as well.

It is verified that the SSIM(structural similarity index) loss combined with l_1 loss perform better than l_2 loss in image restoration [46]. Therefore, we also adopt $\mathcal{L}_i^{SSIM} = 1 - SSIM(\mathbf{T}, \hat{\mathbf{T}}_i)$ in each iteration i as a loss term, which can be written into:

$$\mathcal{L}_{SSIM} = \sum_{T \in \mathcal{D}} \sum_{i=1}^{N} \theta^{N-i} \mathcal{L}_i^{SSIM}, \tag{11}$$

where the setting of θ is same as Eq. 10. We denote the mixture of SSIM and pixel loss as $\mathcal{L}_{mix}$ and define it as:

$$\mathcal{L}_{mix} = \alpha \mathcal{L}_{SSIM} + (1 - \alpha) \mathcal{L}_{pixel}, \tag{12}$$

where α is set to 0.84, same as the setting in [46].

Adversarial Loss. To improve the quality of the generated images, we further add an adversarial loss. We adopt a multi-layer discriminator network $\boldsymbol{D}$ to assess the quality of images and define the adversarial loss as:

$$\mathcal{L}_{adv} = \sum_{T \in \mathcal{D}} -log\, \boldsymbol{D}(\mathbf{T}, \hat{\mathbf{T}}). \tag{13}$$

Final loss. The final training loss can then be defined as:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{comp} + \lambda_2 \mathcal{L}_p + \lambda_3 \mathcal{L}_{mix} + \lambda_4 \mathcal{L}_{adv}. \tag{14}$$

Dataset (size)	Index (↑)	Methods						
		Zhang *et al*.-F [45]	BDN [42]	RMNet [39]	ERRNet-F [38]	Kim *et al*. [13]	IBCLN-F [18]	Ours
Postcard (199)	PSNR	21.497	20.460	19.833	22.374	23.055	23.421	**23.724**
	SSIM	0.870	0.858	0.872	0.889	0.871	0.864	**0.903**
Object (200)	PSNR	23.675	22.642	24.045	23.101	23.552	**24.416**	24.361
	SSIM	0.885	0.857	0.847	0.876	0.879	0.889	**0.898**
Wild(55)	PSNR	24.861	22.048	19.800	24.097	25.534	24.724	**25.731**
	SSIM	0.886	0.828	0.885	0.880	0.890	0.871	**0.902**
Zhang *et al*.(20)	PSNR	22.230	18.487	18.780	23.153	20.218	21.008	**23.338**
	SSIM	0.800	0.729	0.708	0.809	0.735	0.760	**0.812**
Li *et al*.(20)	PSNR	20.721	18.828	15.457	20.368	20.096	**23.695**	23.451
	SSIM	0.765	0.738	0.732	0.771	0.759	0.804	**0.808**
Average(494)	PSNR	22.752	21.374	21.315	22.810	23.298	23.882	**24.179**
	SSIM	0.871	0.844	0.851	0.875	0.866	0.868	**0.893**

Table 1. Quantitative comparison to state-of-the-art methods on real-world datasets. The best results are marked in red, and the second-best results are marked in blue.

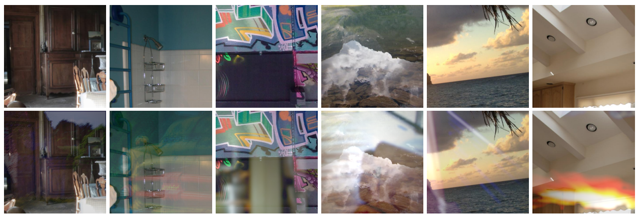
Figure 6. Examples of our synthesized images. Top: the transmission layers. Bottom: the synthesized images with reflections. First two columns: adding ghosting effect. Middle two columns: the blurred reflections. Last two columns: adding reflected highlights.

We set the weights for each loss in our experiments as follows: $\lambda_1 = 0.4, \lambda_2 = 0.2, \lambda_3 = 0.4, \lambda_4 = 0.01$.

5. Experiments

We have implemented our method using Pytorch [22] and tested it on a PC with an Nvidia Geforce RTX 2080 Ti GPU. To minimize the training loss, we adopt ADAM optimizer [14] to train our network for 60 epochs with learning rate $2e^{-4}$ and batch size 2. The β_1, β_2 in ADAM are set to 0.5 and 0.99, respectively. The network weights are initialized using a normal distribution (mean:0, variance: 0.02), and the number of recurrent iterations is set to 3, same as IBCLN [18]. In the inference stage, it takes about $0.068s$ for our method to process an input image of resolution 400×540.

Training dataset. Our training dataset consists of both synthesis and real-world data. For the synthesis data, we use the images dataset from [5]. This dataset has approximately 13700 transmission and reflection image pairs of size 256×256. With these pairs, we generate images with reflection using the linear synthesis model in [45]: $\mathbf{I} = \alpha \cdot \mathbf{T} + \mathbf{R}$, where α is randomly sampled in the interval $[0.7, 1.0]$ (see Fig. 6 for examples of synthesized images) and prepare triples, $\{\mathbf{I}, \mathbf{T}, \mathbf{R}\}$ for training. For the real-world data, there are a total of 290 real-world image pairs, $\{\mathbf{I}, \mathbf{T}\}$, in our dataset, including 200 pairs provided by the "Nature" dataset in IBCLN [18] and 90 pairs provided by Zhang *et al*. [45]. Same as [18], we feed the network with 4000 images in each epoch, including 2800 randomly sampled from the synthesis data and 1200 image patches of size 256×256 cropped from the real-world images.

Data augmentation. For sampled reflection images from the synthesis data, the data augmentation procedure first chooses whether to add ghosting effects or using a gray-scale version of reflection images with a probability of 0.2 or 0.3 respectively. The usage of gray-scale reflection images is an efficient acceleration strategy in the early training of reflection removal networks. The ghosting effect is used to simulate the reflective effect of thick glasses [4, 27], which can be formulated as: $\mathbf{R} = \beta \cdot \mathbf{H} \otimes \mathbf{K} \otimes \mathbf{R}$, where $\mathbf{K}$ is a Gaussian kernel, β is a reflection rate map of size $256 \times 256 \times 3$ cropped from a $560 \times 560 \times 3$ Gaussian map (sigma=3) [18], $\mathbf{H}$ is a two-pulse kernel(size=9, $peak_1 = 1 - \sqrt{\alpha}$, $peak_2 = \sqrt{\alpha} - \alpha$), $\otimes$ is the convolution operation. Second, if the ghosting effect is not chosen in the first step, all the images will be blurred with a Gaussian filter [18, 38, 45]. The filter kernel size is in the range of $[2, 5]$. We linearly increase the kernel range from $[2, 5]$ to $[0.8, 5.8]$ to cover more variations of the blurring degree of the reflection images. Specifically, the augmented reflection layer $\mathbf{R}$ is generated by $\mathbf{R} = \beta \cdot \mathbf{K} \otimes \mathbf{R}$, the reflection rate β is the same parameter as in adding a ghosting effect. We compose the processed reflection images with transmission images to synthesize $\mathbf{I}$ [45], and then perform gamma correction for $\mathbf{T}, \mathbf{R}$ and the composited image $\mathbf{I}$.

Finally, we apply random rotation $(90^\circ, 180^\circ, 270^\circ)$ and flipping to all the loaded images to obtain the final training images as the network's inputs. The gamma correction is not executed on captured real-world images. After 60 epochs, we reduce the Gaussian kernel size to $[0.5, 3.5]$ and the learning rate to $3e^{-5}$ for the fine-tuning of our network on synthesized images with strong reflections.

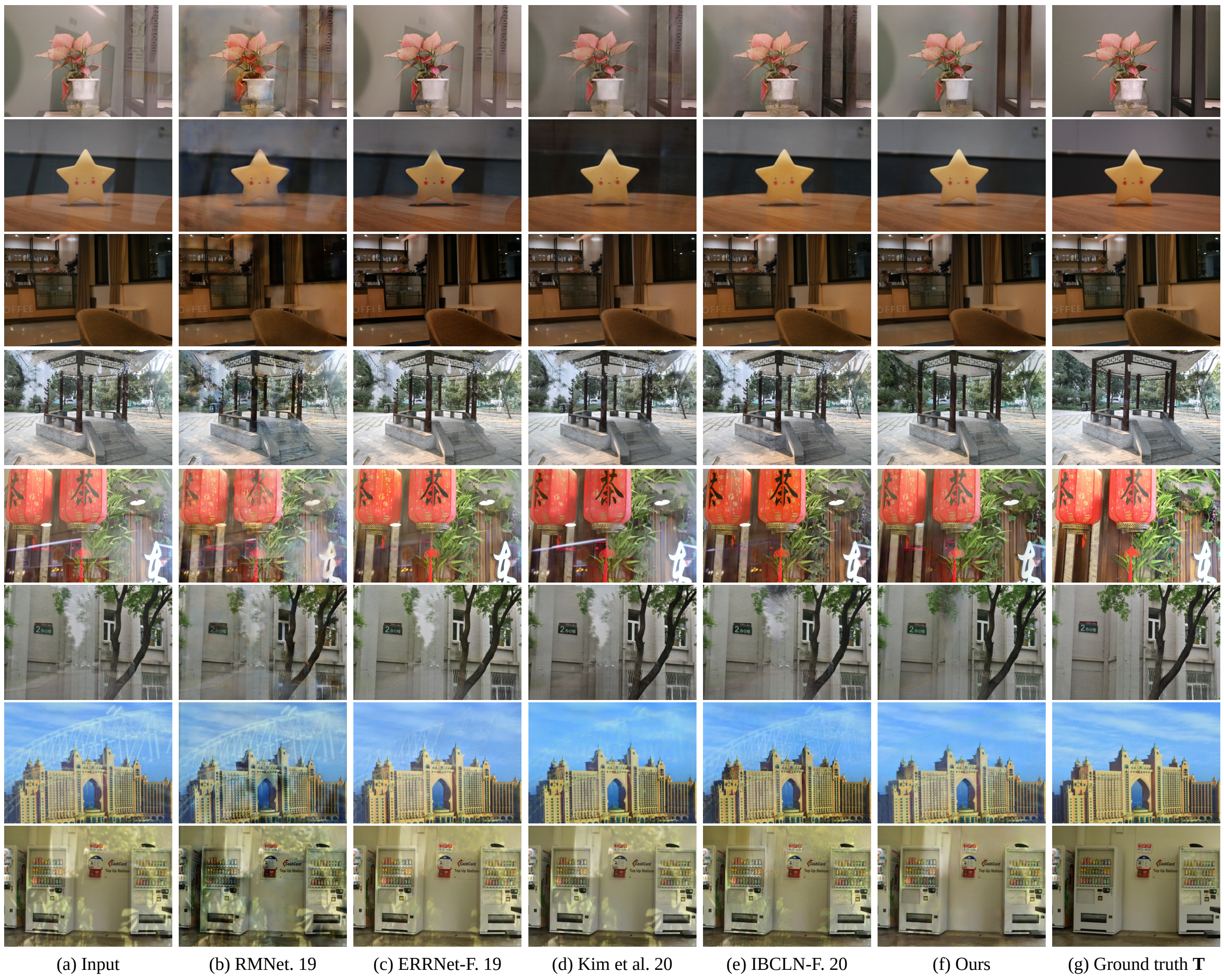

Figure 7. Qualitative comparisons between the proposed method and four latest state-of-the-arts.

5.1. Comparisons

To evaluate the performance of our method for SIRR, we compare it to six state-of-the-art methods based on deep learning, including Zhang *et al*. [45], BDN [42], RMNet [39], ERRNet [38], Kim *et al*. [13], and IBCLN [18]. We use PSNR and SSIM as metrics, where the higher metric value means better performance. For fair comparisons, we report their better performances either using their original trained models or using models fine-tuned with our training images if their training codes are available. The fine-tuned results are denoted with a suffix ”-F”. Note that we do not fine-tune the RMNet [39], as it requires additional alpha blending masks from a SynNet [39]. We also modify the code of Kim *et al*. [13] to compute SSIM in RGB space.

Quantitative comparisons. We quantitatively compare our model with the methods mentioned above on five real-world datasets and report the statistics in Tab. 1. The first three datasets are all from SIR^2 constructed in [34], and the rest two datasets are from the evaluation set in Zhang *et al*. [45] and the “Nature” test dataset in Li *et al*. [18] respectively. In Tab. 1, it can be seen that our method is ranked as top-1 on the Postcard, Wild, and Zhang *et al*. datasets and top-2 (PSNR ranking) on the Object and Li *et al*. datasets. However, for the top-2 PSNR ranking cases, our result is comparable to the top 1 methods. Moreover, our method achieves the best average PSNR and SSIM scores. This result verifies that our method can achieve superior performance in various real-world scenarios.

Qualitative comparisons. Fig. 7 presents the reflection removal results of different models on corrupted images. These images are from the benchmark datasets SIR^2 [34](rows 7-8), unaligned datasets of ERRNet [38] (rows 4-6), the “Nature” test dataset in Li *et al*. [18](rows 1-3). It can be seen that there are difficult cases, such as a large area of reflections and strong highlights, that are not

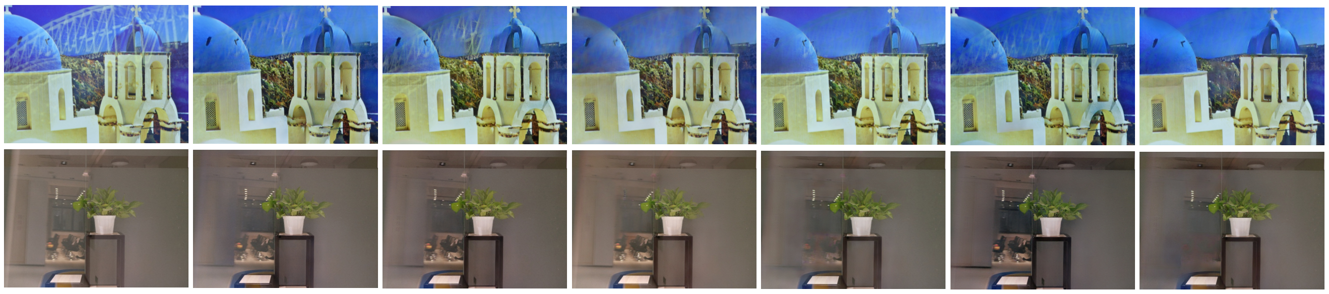
(a) Input (b) Without RDM & TSM (c) Without TSM (d) RCMap from $\mathbf{I}_{ft}$ (e) Fix MLSM (f) Laplacian $\rightarrow$ Edge (g) Ours

Figure 8. Visualization of reflection removal results according to the ablation study in Tab. 2.

Model	*SIR*^{2} [34]		Zhang *et al.*[45]		Li *et al.*[18]	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
w/o RDM & TSM	23.237	0.881	22.784	0.800	22.607	0.778
w/o TSM	23.304	0.882	22.139	0.805	23.088	0.799
w/o LSTM	23.808	0.887	21.040	0.760	22.721	0.794
$\hat{\mathbf{C}}$ from $\mathbf{I}_{ft}$	22.595	0.888	21.747	0.792	22.283	0.792
MLSM $\rightarrow$ SLSM	23.089	0.879	21.708	0.796	23.065	0.800
Fix MLSM	23.640	0.891	22.951	0.809	22.646	0.802
Laplacian $\rightarrow$ *Edge*	23.612	0.892	22.381	0.808	23.234	0.798
Ours	**24.117**	**0.901**	**23.338**	**0.812**	**23.451**	**0.808**

Table 2. Network structure ablation study. w/o RDM & TSM: remove RDM and TSM. w/o TSM: remove TSM. w/o LSTM: remove the LSTM block. $\hat{\mathbf{C}}$ from $\mathbf{I}_{ft}$: disable MLSM in RDM and predict the RCMap using the extracted features of $\mathbf{I}$ and $\mathbf{T}$ before LSTM block. MLSM $\rightarrow$ SLSM: Change multi-scale Laplacian to the single scale Laplacian. Fix MLSM: disable the fine-tuning of Laplacian kernel parameters. *Laplacian* $\rightarrow$ *Edge*: Change Laplacian features to Edge features, namely the partial derivatives of the image with respect to x and y according to Fig. 4, in RDM.

Model	*SIR*^{2} [34]		Zhang *et al.* [45]		Li *et al.* [18]	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
w/o $\mathcal{L}_{pixel}$	23.365	0.888	22.113	0.788	21.587	0.788
w/o $\mathcal{L}_{SSIM}$	15.653	0.673	17.652	0.746	15.197	0.642
w/o $\mathcal{L}_{P}$	22.725	0.880	22.219	0.804	22.778	**0.812**
w/o $\mathcal{L}_{comp}$	23.774	0.894	22.453	0.807	23.036	0.801
w/o $\mathcal{L}_{adv}$	23.571	0.889	22.480	0.803	23.240	0.800
Complete	**24.117**	**0.901**	**23.338**	**0.812**	**23.451**	**0.808**

Table 3. Loss term ablation study. We remove each of the five loss terms and evaluate the corresponding re-trained model to check its influence on the reflection removal results.

well handled by other methods. In contrast, our method can remove most undesirable reflections while keeping the high-frequency details of the transmission layer at the same time. However, when the strong reflection regions are not correctly detected, our method still lacks information to remove such reflections. For instance, there are remained reflected highlights in our result in the fifth row of Fig. 7. As shown in Fig. 9, the two oval strong highlights are not totally detected in RCMap for this image. Hence, they still appear in the removal result. Although there are hard boundaries in the inverse Laplacian map for these highlights, we speculate that the error is because there is no enough context

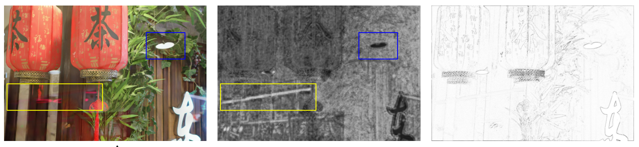
(a) $\hat{\mathbf{T}}$ (b) RCMap (c) Inverse Laplacian map

Figure 9. A failure case. The oval reflected highlights are not correctly detected for the corrupted image in the fifth row of Fig. 7.

information for the network to classify such oval highlights as reflections. Besides, this image is un-aligned with the transmission image, but we did not train our network on the un-aligned reflection removal dataset from ERRNet [38].

5.2. Ablation Study

To better analyze the architecture of our network and evaluate the importance of the loss functions, we perform the ablation study by changing the model structure and removing each loss function. The statistics of PSNR and SSIM are obtained by evaluating the re-trained models in these experiments. In Tab. 2, we firstly show the effectiveness of RDM, TSM, and LSTM blocks in our network (first three rows). We can found that each module contributes to the SIRR performance. Secondly, we test four different choices in the design of RDM (last four rows, see captions of Tab. 2 for the descriptions). It can be seen that our current design choice of RDM leads to the highest PSNR and SSIM scores. In addition, the visualization of the reflection removal results in the evaluation of network structure is shown in Fig. 8. In Tab. 3, we show that each loss term contributes to the network's performance. We speculate that the large drop of PSNR and SSIM after removing $\mathcal{L}_{SSIM}$ is because the weight of SSIM loss is much larger than pixel loss in $\mathcal{L}_{mix}$. Removing $\mathcal{L}_{comp}$ also degrades the performance but not as large as $\mathcal{L}_{SSIM}$. Thus, with the rest loss terms, the network can learn to predict RCMap in a self-supervised manner to some extent.

6. Conclusion

We proposed a location-aware SIRR network in this paper to improve the quality of SIRR results substantially. The network has an RDM to detect reflections roughly. The predicted RCMap is used to control the features flow of the network for restoring the transmission and reflection layers. Such design enables the network to learn to encode reflection information adaptively and is beneficial to removing strong reflections, such as reflected highlights. In the future, we plan to simplify our network's design further to save the number of parameters and improve the inference speed for its application on the mobile computing platform.

References

[1] Nikolaos Arvanitopoulos, Radhakrishna Achanta, and Sabine Susstrunk. Single image reflection suppression. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4498–4506, 2017. 2

[2] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011. 2

[3] Yakun Chang and Cheolkon Jung. Single image reflection removal using convolutional neural networks. *IEEE Trans. Image Process.*, 28(4):1954–1966, 2018. 2

[4] Zhixiang Chi, Xiaolin Wu, Xiao Shu, and Jinjin Gu. Single image reflection removal using deep encoder-decoder network. *arXiv preprint arXiv:1802.00094*, 2018. 6

[5] Qingnan Fan, Jiaolong Yang, Gang Hua, Baoquan Chen, and David Wipf. A generic deep architecture for single image reflection removal and image smoothing. In *Int. Conf. Comput. Vis.*, pages 3238–3247, 2017. 1, 2, 4, 6

[6] Kun Gai, Zhenwei Shi, and Changshui Zhang. Blind separation of superimposed moving images using image statistics. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(1):19–32, 2011. 2

[7] Rafael C Gonzalez, Richard Eugene Woods, and Steven L Eddins. *Digital image processing using MATLAB*. Pearson Education India, 2004. 4

[8] Xiaojie Guo, Xiaochun Cao, and Yi Ma. Robust separation of reflection from multiple images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2187–2194, 2014. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Int. Conf. Comput. Vis.*, pages 1026–1034, 2015. 5

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2, 3, 5

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7132–7141, 2018. 3, 5

[12] Meiguang Jin, Sabine Süsstrunk, and Paolo Favaro. Learning to see through reflections. In *2018 IEEE International Conference on Computational Photography (ICCP)*, pages 1–12. IEEE, 2018. 2, 3

[13] Soomin Kim, Yuchi Huo, and Sung-Eui Yoon. Single image reflection removal with physically-based training images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5164–5173, 2020. 1, 2, 3, 6, 7

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[15] Chenyang Lei, Xuhua Huang, Mengdi Zhang, Qiong Yan, Wenxiu Sun, and Qifeng Chen. Polarized reflection removal with perfect alignment in the wild. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1750–1758, 2020. 2

[16] A. Levin and Y. Weiss. User assisted separation of reflections from a single image using a sparsity prior. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(9):1647–1654, 2007. 1, 2

[17] Anat Levin, Assaf Zomet, and Yair Weiss. Learning to perceive transparency from the statistics of natural scenes. In *Adv. Neural Inform. Process. Syst.*, pages 1271–1278, 2003. 1, 2

[18] Chao Li, Yixiao Yang, Kun He, Stephen Lin, and John E Hopcroft. Single image reflection removal through cascaded refinement. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3565–3574, 2020. 1, 2, 3, 5, 6, 7, 8

[19] Yu Li and Michael S Brown. Exploiting reflection change for automatic reflection removal. In *Int. Conf. Comput. Vis.*, pages 2432–2439, 2013. 2

[20] Yu Li and Michael S Brown. Single image layer separation using relative smoothness. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2752–2759, 2014. 1, 2

[21] Ajay Nandoriya, Mohamed Elgharib, Changil Kim, Mohamed Hefeeda, and Wojciech Matusik. Video reflection removal through spatio-temporal optimization. In *Int. Conf. Comput. Vis.*, pages 2411–2419, 2017. 2

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[23] Abhijith Punnappurath and Michael S Brown. Reflection removal using a dual-pixel sensor. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1556–1565, 2019. 2

[24] Rui Qian, Robby T Tan, Wenhan Yang, Jiajun Su, and Jiaying Liu. Attentive generative adversarial network for raindrop removal from a single image. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2482–2491, 2018. 4

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015. 5

[26] Bernard Sarel and Michal Irani. Separating transparent layers through layer information exchange. In *Eur. Conf. Comput. Vis.*, pages 328–341. Springer, 2004. 2

[27] YiChang Shih, Dilip Krishnan, Fredo Durand, and William T Freeman. Reflection removal using ghosting cues. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3193–3201, 2015. 1, 2, 6

[28] Christian Simon and In Kyu Park. Reflection removal for in-vehicle black box videos. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4231–4239, 2015. 2

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[30] Sudipta N Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. Image-based rendering for scenes with reflections. *ACM Trans. Graph.*, 31(4):1–10, 2012. 2

[31] Chao Sun, Shuaicheng Liu, Taotao Yang, Bing Zeng, Zhengning Wang, and Guanghui Liu. Automatic reflection removal using gradient intensity and motion cues. In *ACM Int. Conf. Multimedia*, pages 466–470, 2016. 2

[32] Richard Szeliski, Shai Avidan, and Padmanabhan Anandan. Layer extraction from multiple images containing reflections and transparency. In *IEEE Conf. Comput. Vis. Pattern Recog.*, volume 1, pages 246–253. IEEE, 2000. 2

[33] Renjie Wan, Boxin Shi, Ling-Yu Duan, Ah-Hwee Tan, Wen Gao, and Alex C Kot. Region-aware reflection removal with unified content and gradient priors. *IEEE Trans. Image Process.*, 27(6):2927–2941, 2018. 1, 2

[34] Renjie Wan, Boxin Shi, Ling Yu Duan, Ah Hwee Tan, and Alex C. Kot. Benchmarking single-image reflection removal algorithms. In *Int. Conf. Comput. Vis.*, 2017. 7, 8

[35] Renjie Wan, Boxin Shi, Tan Ah Hwee, and Alex C Kot. Depth of field guided reflection removal. In *IEEE Int. Conf. Image Process.*, pages 21–25. IEEE, 2016. 1, 2

[36] Renjie Wan, Boxin Shi, Haoliang Li, Ling-Yu Duan, and Alex C Kot. Face image reflection removal. *Int. J. Comput. Vis.*, pages 1–15, 2020. 2

[37] Renjie Wan, Boxin Shi, Haoliang Li, Ling-Yu Duan, Ah-Hwee Tan, and Alex Kot Chichung. Corrn: Cooperative reflection removal network. 2019. 2

[38] Kaixuan Wei, Jiaolong Yang, Ying Fu, David Wipf, and Hua Huang. Single image reflection removal exploiting misaligned training data and network enhancements. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8178–8187, 2019. 1, 2, 3, 5, 6, 7, 8

[39] Qiang Wen, Yinjie Tan, Jing Qin, Wenxi Liu, Guoqiang Han, and Shengfeng He. Single image reflection removal beyond linearity. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3771–3779, 2019. 1, 2, 3, 6, 7

[40] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Eur. Conf. Comput. Vis.*, pages 3–19, 2018. 3, 4

[41] Tianfan Xue, Michael Rubinstein, Ce Liu, and William T Freeman. A computational approach for obstruction-free photography. *ACM Trans. Graph.*, 34(4):1–11, 2015. 2

[42] Jie Yang, Dong Gong, Lingqiao Liu, and Qinfeng Shi. Seeing deeply and bidirectionally: A deep learning approach for single image reflection removal. In *Eur. Conf. Comput. Vis.*, pages 654–669, 2018. 2, 3, 6, 7

[43] Yang Yang, Wenye Ma, Yin Zheng, Jian-Feng Cai, and Weiyu Xu. Fast single image reflection suppression via convex optimization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8141–8149, 2019. 1, 2

[44] Leonid P Yaroslavsky. *Digital picture processing: an introduction*, volume 9. Springer Science & Business Media, 2012. 5

[45] Xuaner Zhang, Ren Ng, and Qifeng Chen. Single image reflection separation with perceptual losses. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4786–4794, 2018. 1, 2, 3, 6, 7, 8

[46] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for neural networks for image processing. *arXiv preprint arXiv:1511.08861*, 2015. 5