Location-aware Single Image Reflection Removal

Zheng Dong\(^1\) Ke Xu\(^{2,4}\) Yin Yang\(^3\) Hujun Bao\(^1\) Weiwei Xu\(^1\) Rynson W.H. Lau\(^4\)
\(^1\)State Key Lab of CAD&CG, Zhejiang University \quad \(^2\)Shanghai Jiao Tong University \\
\(^3\)Clemson University \quad \(^4\)City University of Hong Kong
zhengdong@zju.edu.cn, kkangwing@gmail.com, 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 if a region is reflection-dominated or transmission-dominated, and it is used as a cue for the network to control the feature flow when predicting the reflection and transmission layers. We design our network as a recurrent network to progressively refine reflection removal results at each iteration. The novelty is that we leverage Laplacian kernel parameters to emphasize the boundaries of strong reflections. It is beneficial to strong reflection detection and substantially improves the quality of reflection removal results. Extensive experiments verify the superior performance of the proposed method over state-of-the-art approaches. Our code and the pre-trained model can be found at https://github.com/zdlarr/Location-aware-SIRR.

1. Introduction

Reflections often occur when images are 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. In reflection removal, an image \(I\) with reflections can be modeled as the weighted additive composition of a transmission layer \(T\) and a reflection layer \(R\). Precisely, following the alpha blending model in \([4, 23, 57]\), we express the composition procedure as:

\[
I = W \circ T + R,
\]

where \(W\) here is an alpha blending mask and \(\circ\) indicates the element-wise multiplication. This model is designed to approximate the complex physical mechanisms involved in forming images with reflections.

The task of single image reflection removal (SIRR) is to recover \(T\) from a given image \(I\). 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 \([21, 22]\), ghosting cues for thick glasses \([34]\), and relative smoothness that assumes the refection layer is smoother than the transmission layer \([24, 52]\). To disambiguate the restoration of \(T\) and \(R\) 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 \([21, 40, 42]\). 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. Therefore, such location information is expected to be beneficial for the neural networks to learn how the reflection information is encoded in the features. However, it is rarely investigated in deep learning-based SIRR...
methods [4, 15, 18, 23, 46, 47, 50, 57]. This might cause ambiguities when strong reflections appear. We observe that state-of-the-art SIRR methods typically fail to recover high-quality transmission layers in such cases.

This paper proposes a location-aware deep learning-based SIRR method for generic reflection removal. Our network incorporates a novel reflection detection module (RDM) to detect reflection-dominated regions via learning multi-scale Laplacian features. The output of RDM is a probabilistic reflection confidence map (RCMap), which controls the subsequent feature flow, resulting in significantly improved SIRR results. Although this detection-and-removal strategy has been explored in shadow analysis [3, 12, 13, 16, 31, 45] and rain removal [6, 17, 30, 51, 54, 55] tasks, they mainly exploit features learned in the RGB domain, which are different from our approach in two aspects. First, motivated by the relative smoothness prior in [24], which assumes that the characteristics of T and R in the gradient domain are different, we use the learned Laplacian kernel to emphasize the boundaries of strong reflections and suppress low-frequency reflections, which is beneficial to improve the quality of RCMaps. Second, RDM is trained without ground-truth reflection-dominated region masks. This can avoid the difficulty of defining or labeling reflection-dominated regions. The inverse RCMap, i.e., 1 - RCMap, can serve as the weight map \( W \) 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 removal-by-detection manner. It first detects the reflection-dominated regions based on the RCMap and then predicts the whole reflection layer by suppressing the transmission information. Afterwards, it restores the transmission layer by jointly leveraging the transmission-dominant regions and the predicted reflection layer. Such a network design decomposes the complicated SIRR problem into sub-problems and considers the mutual dependence between reflection and transmission. As illustrated in Fig. 1, our network can effectively remove strong 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 novel reflection detection module (RDM) to locate the reflection-dominated regions. It learns a group of multi-scale Laplacian kernel parameters to exploit reflection boundary information.
- Extensive experiments show that the proposed location-aware neural network outperforms state-of-the-art SIRR methods in removing strong reflections.

2. Related Work

A variety of reflection removal methods, such as multi-view or video-based [5, 8, 25, 27, 28, 31, 38, 39, 49], dual-pixel sensor-based [29] and polarization-based [2, 20, 35] methods, have been proposed to restore the transmission layer through motion or optical cues. Since we focus on SIRR in this paper, we mainly review the single image based methods below.

To handle the ill-posed SIRR problem, traditional optimization-based methods introduce different priors [21, 22]. Observing that reflection layers are usually out of focus and appear to be more blurry than transmission layers, Li et al. [24] introduced a relative smoothness prior to distinguish the gradients of the two layers with different probability distributions. Shih et al. [34] exploited ghosting cues to remove reflections when the thickness of the glass cannot be ignored. Huang et al. [14] proposed a wavelet transform based regularization method to separate ghosting patterns from background patterns. Multi-scale depth-of-field (DoF) analysis based methods [40, 42] were proposed to separate reflection from transmissions by detecting the reflection-dominated regions. However, the thresholds in these methods for determining the reflection regions are vulnerable to noise. In [1, 52], the Laplacian data fidelity term is used to suppress the blurry reflections. However, these two methods cannot effectively remove strong reflections and might smooth out the transmission layer’s details. In contrast, we leverage Laplacian features to emphasize the boundaries of strong reflections as a clue for the network to remove them.

Recently, many deep learning-based methods [4, 15, 18, 23, 46, 47, 50, 56, 57] were proposed to solve the SIRR problem by learning task-specific features. Fan et al. [4] designed a deep neural network, called CEIL-Net, to first regress the edge map of the transmission layer and then reconstruct the transmission layer. Yang et al. [50] proposed the BDN, a multi-stage network for estimating two layers sequentially, where the reflection layer predicted in the previous stage is used as auxiliary information to guide the transmission layer reconstruction in the next stage. Li et al. [23] proposed the IBCLN method, which is an LSTM based recurrent network for concurrently refining the results of the predicted reflection and transmission layers. Wan et al. [44] proposed a feature-sharing strategy and a statistical loss for removing the strong reflections within local regions. They further proposed to study the face reflection removal problem in [43]. Zhang et al. [56] proposed to explore the edge hints in the user-specified regions to separate the reflection and transmission layers. Some methods proposed to generate more representative training data. Jin et al. [15] proposed multiple data generation models. Wen et al. [47] proposed SynNet to generate images with reflections beyond linearity. Wei et al. [46] introduced an alignment-invariant loss to utilize the misaligned images as
the real-world training dataset. Kim et al. [18] proposed a physics-based rendering method to render realistic images with reflections. Unlike these methods, this paper explores how to incorporate the location information of reflections into the network for controlling the feature flow.

Many loss terms were used to boost the SIRR performance [18, 23, 46, 47, 50, 57], e.g., the VGG-based perceptual loss, exclusion loss in the gradient domain, and adversarial loss to prevent the blurring effects. In this paper, a new composition loss without using the GT reflection-dominated region masks is proposed to train our RDM.

### 3. Method

Our network is a recurrent network, as illustrated in Fig. 2. In each iteration $i$, our network takes the original image $I$ and the transmission layer $\hat{T}_{i-1}$ predicted in the previous iteration $i - 1$ as inputs, and predicts the transmission layer $\hat{T}_i$ to continue the iteration. $\hat{T}_0$ is initialized as $I$. The recurrent structure of our network is inspired by IBCLN [23]. However, different from jointly estimating the transmission and reflection layers in IBCLN, we design our model to reconstruct the transmission layer $\hat{T}_i$ conditioned on the RCMap and the restored reflection layer at each iteration. The step-by-step refinement results of reflection removal are shown in Fig. 3.

Like BDN [50], each iteration of our network is divided into two stages to restore two layers sequentially. However, we leverage RCMap to control the between-stage information flow. In the first stage, we predict the reflection layer $\hat{R}_i$ and RCMap $\hat{C}_i$ by taking $I$ and $\hat{T}_{i-1}$ as inputs. We denote the first stage as the function $G_R$:

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

This stage mainly consists of two modules: the reflection detection module (RDM) and the transmission-feature suppression module (TSM). Specifically, The RDM takes $I$ and $\hat{T}_{i-1}$ as inputs and predicts the confidence map $\hat{C}_i$ using features from a multi-scale Laplacian sub-module (MLSM). Next, the TSM is used to suppress the Laplacian features within transmission-dominated regions via an element-wise multiplication between the features and $\hat{C}_i$. Afterwards, the suppressed features and the image features are concatenated as the inputs of an LSTM [10] block to estimate $\hat{R}_i$. In our work, $\hat{R}_i$ and $\hat{C}_i$ are mainly used as cues to facilitate the reconstruction of transmission layers.

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

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

Notice that we utilize the inverse confidence map, i.e., $1 - \hat{C}_i$, in this stage. Since the transmission layer dominates at regions where $1 - \hat{C}_i$ have a high value, we expect this...
the following factors: 1/2, 1/4, 1/8. 

In this work, we utilize bi-linear interpolation with this two images to form reflection-dominated regions and thus concatenate such a behavior of the Laplacian operator is beneficial to detect reflection-dominated regions by the Laplacian operator. It is possible that the difference between \( I \) and \( T \) caused by strong reflections becomes more obvious in the Laplacian domain. We assume such a behavior of the Laplacian operator is beneficial to detect reflection-dominated regions and thus concatenate these two images to form \( \hat{X}_{in} = [I, \hat{T}_{i-1}] \) as inputs to obtain multi-scale Laplacian features.

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 evident in the inverse Laplacian map than in the inverse edge map, which suggests that the Laplacian operator suppresses low-frequency reflections more effectively. In contrast, strong reflections that have hard boundaries can not be suppressed by the Laplacian operator. It is possible that the difference between \( I \) and \( T \) caused by strong reflections becomes more obvious in the Laplacian domain. We assume such a behavior of the Laplacian operator is beneficial to detect reflection-dominated regions and thus concatenate these two images to form \( \hat{X}_{in} = [I, \hat{T}_{i-1}] \) as inputs to obtain multi-scale Laplacian features.

For the purpose of multi-scale Laplacian feature learning, we down-sample \( \hat{X}_{in} \) through scaling the height and width of input images using bi-linear interpolation with the following factors: 1/2, 1/4, 1/8 [7]. The down-sampled results are denoted by \( \hat{X}^1_{in}, \hat{X}^2_{in}, \hat{X}^4_{in} \) respectively.

We utilize a convolution kernel with weights initialized to be a \( 3 \times 3 \) Laplacian kernel, denoted by \( k_L = [0, -1, 0; -1, 4, -1; 0, -1, 0] \), to obtain the second derivative signal from \( \hat{X}_{in} \). We allow the network to fine-tune the Laplacian kernel parameters to better extract Laplacian features, where the fine-tuned parameters are denoted as \( \hat{L}ap \). During training, we utilize gradient clipping (0.25 in our experiments) to make sure that the learned kernel stays close to the original kernel \( k_L \).

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

\[
X_{out} = Concat(Lap(\hat{X}_{in}), \hat{U}^j_j(Lap(\hat{X}^j_{in})))_{j=2,4,8},
\]

where \( Concat \) is the concatenation operation.

Finally, taking \( X_{out} \) as input, RDM predicts the reflection confidence map \( \hat{C} \) 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 denoted as \( f_{Lap} \). Thus, given the features \( X_{out} \), the map \( \hat{C} \) can be described as:

\[
\hat{C} = Sigmoid(Conv(f_{Lap}(X_{out}))).
\]

The improvement of the predicted RCMaps for two synthesized images along with iterations is illustrated in Fig. 5.

Transmission-feature suppression module. In this module, we employ three SE-ResBlocks to refine the Laplacian features and then multiply the features by \( \hat{C} \) for the purpose of transmission features suppressing. Instead of pre-
dicting \( \hat{\mathbf{R}}_i \) with RCMap \( \hat{\mathbf{C}}_i \) as input, just like predicting \( \mathbf{T}_i \) in the second stage, we empirically found that suppressing the part of Laplacian features belong to transmission-dominated regions benefits the reflection layer prediction. It also leads to a relatively simple network design by concatenating the features computed with \( \mathbf{X}_{inv} \) 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.

4. Training Loss

In this section, we describe the four loss functions used in the training of our network. For clarity, we denote the ground-truth transmission and reflection layers by \( \mathbf{T}, \mathbf{R} \), the predicted transmission and reflection layers at iteration \( i \) as \( \hat{\mathbf{T}}_i, \hat{\mathbf{R}}_i \) respectively, and inverse gamma correction as a function \( g_{inv} \). 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 for predicting \( \hat{\mathbf{C}}_i \), and supervise \( \hat{\mathbf{T}}_i, \hat{\mathbf{R}}_i \) at each iteration using training images synthesized by the following linear alpha blending model in [57]:

\[
\tilde{\mathbf{I}} = \alpha \cdot \hat{\mathbf{T}} + \hat{\mathbf{R}},
\]

(6)

where \( \alpha \) is a scalar, \( \hat{\mathbf{T}} = g_{inv}(\mathbf{T}) \) and \( \hat{\mathbf{R}} = g_{inv}(\mathbf{R}) \).

Firstly, since the map \( 1 - \hat{\mathbf{C}}_i \) can serve as the weight map \( \mathbf{W} \) in Eq. 1, we can compose an image \( \tilde{\mathbf{I}}_i \) by the following formula using gamma corrected \( \mathbf{T} \) and \( \mathbf{R} \):

\[
\tilde{\mathbf{I}}_i = (1 - \hat{\mathbf{C}}_i) \circ \mathbf{T} + \mathbf{R},
\]

(7)

where \( \circ \) is an element-wise production operation. We formulate the loss for RCMap as follows:

\[
\mathcal{L}_C = \sum_{\mathbf{I}, \mathbf{T} \in \mathcal{D}} \sum_{i=1}^{N} \theta^{N-i} \mathcal{L}_{MSE}(\mathbf{I}, \tilde{\mathbf{I}}_i),
\]

(8)

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

Secondly, same as IBCLN [23], we adopt Eq. 6 to form a residual loss to guide the prediction of \( \hat{\mathbf{T}}_i, \hat{\mathbf{R}}_i \) in two forms: \( \tilde{\mathbf{I}}^{\circ}_i = \alpha \cdot g_{inv}(\mathbf{T}) + g_{inv}(\hat{\mathbf{R}}_i) \) and \( \tilde{\mathbf{I}}^{\circ}_i = \alpha \cdot g_{inv}(\hat{\mathbf{T}}_i) + g_{inv}(\hat{\mathbf{R}}_i) \), where \( \alpha \) is a known scalar used to synthesize training images. We use Eq. 8 to compute the difference between \( \tilde{\mathbf{I}}^{\circ}_i \) and \( g_{inv}(\mathbf{I}) \), and denote the loss as \( \mathcal{L}_{res} \). The composition loss for the synthesized images is defined as:

\[
\mathcal{L}_{comp} = \mathcal{L}_C + \mathcal{L}_{res}.
\]

(9)

Perceptual Loss. We use VGG-19 network [36] pretrained on ImageNet [32] dataset to extract features for the computation of our perceptual loss. This loss takes multiscale images as inputs and can be written into:

\[
\mathcal{L}_p = \sum_{\mathbf{T} \in \mathcal{D}} \sum_{j=1,3,5} \gamma_j \mathcal{L}_{VGG}(\mathbf{T}^j, \hat{\mathbf{T}}_N^j),
\]

(10)

where \( \mathcal{L}_{VGG} \) denotes the \( l_1 \) loss between VGG features. \( \hat{\mathbf{T}}_N^j \) indicates the output of the last \( j \)-th layer of the autoencoder in stage 2 at iteration \( N \), and \( \mathbf{T}^j \) indicates the ground truth that has the same scale as \( \hat{\mathbf{T}}_N^j \). We set \( \gamma_1 = 1, \gamma_3 = 0.8, \gamma_5 = 0.6 \) respectively. For \( \mathcal{L}_{VGG} \), we use the layers ‘convk,2’ \( (k = 1, 2, 3, 4, 5) \) of the standard VGG-19 net. 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 \( l_1 \) norm loss, denoted as \( \mathcal{L}_1 \), to compute the absolute difference. We define the pixel loss as:

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

where \( \theta \) 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 [58]. Therefore, we also adopt \( \mathcal{L}_{SSIM}^i = 1 - SSIM(\mathbf{T}, \hat{\mathbf{T}}_i) \) in each iteration \( i \) as a loss term, which can be written into:

\[
\mathcal{L}_{SSIM} = \sum_{\mathbf{T} \in \mathcal{D}} \sum_{i=1}^{N} \theta^{N-i} \mathcal{L}_{SSIM}^i,
\]

(12)

where the setting of \( \theta \) is same as Eq. 11. We denote the mixture of SSIM and pixel loss as \( \mathcal{L}_{mix} \) and define it as:

\[
\mathcal{L}_{mix} = \beta \mathcal{L}_{SSIM} + (1 - \beta) \mathcal{L}_{pixel},
\]

(13)

where \( \beta \) is set to 0.84, following the design in [58].

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

\[
\mathcal{L}_{adv} = \sum_{\mathbf{T} \in \mathcal{D}} -\log \mathcal{D}(\mathbf{T}, \hat{\mathbf{T}}).
\]

(14)

Overall Loss. Totally, our training loss is defined as:

\[
\mathcal{L} = \lambda_1 \mathcal{L}_{comp} + \lambda_2 \mathcal{L}_p + \lambda_3 \mathcal{L}_{mix} + \lambda_4 \mathcal{L}_{adv}.
\]

(15)

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

5. Experiments

We implement our method using PyTorch [28] on a PC with an Nvidia GeForce RTX 2080 Ti GPU. To minimize the training loss, we adopt ADAM optimizer [19] to train our network for 60 epochs with a learning rate \( 2e^{-4} \) and batch size 2. After 60 epochs, we reduce the learning rate to \( 1e^{-4} \) and add the unaligned dataset from ERNNet [46] to fine-tune our model. The \( \beta_1, \beta_2 \) 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 iteration number $N$ is set to 3, same as IBCLN [23]. Our network has 10.926M parameters, its FLOPs are 111.63G, which is comparable to IBCLN (130.88G) at each iteration. In the inference stage, it takes about 0.068s to process an input image of resolution $400 \times 540$.

**Training dataset.** Our training dataset consists of both synthetic and real-world data. For the synthetic data, we use the images dataset from [4]. This dataset has approximately 13700 image pairs of size $256 \times 256$. With these pairs, we adopt Eq. 6 and randomly sample $\alpha$ in [0.8, 1.0] to obtain $\tilde{I}$. We then apply gamma correction [53] to \{\tilde{I}, \tilde{T}, \tilde{R}\} to generate image triples \{I, T, R\}. For the real-world data, there are a total of 540 image pairs, \{I, T\}, in our dataset, including 200 pairs provided by the “Nature” dataset in IBCLN [23], 90 pairs provided by Zhang et al. [57] and 250 pairs provided by the unaligned dataset in ERRNet [46]. Following IBCLN [23], we feed the network with 4000 pairs (triples) in each epoch, including 2800 triples randomly sampled from the synthetic data and 1200 pairs of size $256 \times 256$ cropped from the real-world data. Moreover, our data augmentation can generate training images to cover more real-world reflection types. The details can be found in Sec.2 of the supplemental material.

### 5.1. Comparisons

To evaluate the performance of our method, we compare it to seven state-of-the-art SIRR methods, including Zhang et al. [57], BDN [50], RMNet [47], ERRNet [46], CoRRN [44], Kim et al. [18], and IBCLN [23]. 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 dataset 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 [47], as it requires additional alpha blending masks from a SynNet [47]. We also modify the code of Kim et al. [18] to compute SSIM in RGB space.

### Quantitative comparisons

Tab. 1 reports the performance comparisons on five real-world datasets. Datasets in the first three rows are all from $SIR^2$ constructed in [41], and the rest two datasets are from the evaluation set in Zhang et al. [57] and the “Nature” test dataset in Li et al. [23] respectively. It can be seen that our method is ranked as top-1 on the Postcard, Wild, and Zhang et al. datasets, top-2 on the Object (SSIM ranking) and Li et al. (PSNR ranking) datasets. Moreover, our method achieves the best average PSNR and SSIM scores. This verifies that our method can achieve superior performance in various real-world scenarios.

### Qualitative comparisons

Fig. 6 shows the reflection removal results of four existing models and ours. These images are from the “Nature” test dataset of IBCLN [23](rows 1-2), unaligned datasets of ERRNet [46](rows 3-5), and the benchmark datasets $SIR^2$ [41](rows 6-7). It can be seen that existing methods typically fail to remove large-area reflections and strong highlights. In contrast, our method can remove most undesirable reflections while preserving high-frequency details in the transmission layer. More qualitative results can be found in our supplemental material.

However, when the strong reflection regions are not correctly detected, our method still lacks cues to remove such reflections. For instance, there are remained highlights in the 4th row of our results in Fig. 6. As shown in Fig. 7, compared to the detected rod-shaped reflection (blue boxes), the two oval highlights (yellow boxes) are not completely detected. Hence, they still appear in the removal result. Although hard boundaries for these highlights exist in the inverse Laplacian map, our network still lacks enough context information to classify such oval highlights as reflections.

### 5.2. Ablation Study

To better analyze our network’s architecture and evaluate the importance of the loss functions, we perform ablation studies by manipulating the model components, removing or replacing loss functions. The statistics of PSNR and SSIM are obtained by evaluating the re-trained models in these experiments.
Evaluation of the network architecture. In Tab. 2, we first show that three modules in our network, namely RDM, TSM, and LSTM, all contribute to the SIRR performance (first three rows). We then test six different choices in the design of RDM, and the results are shown in the last six rows in Tab. 2. Our current design choices of RDM lead to the highest PSNR and SSIM scores. In addition, Fig. 8 illustrates the visual results of the ablation studies in Tab. 2.

Especially, to evaluate the effectiveness of Laplacian kernel initialization (LKI), we conduct an experiment by replacing LKI with random kernel initialization (RKI) using a Gaussian distribution (mean = 0, variance = 0.02) and cancel the gradient clipping. The PSNR/SSIM scores in the last second row of Tab. 2 show that LKI is superior to RKI. Moreover, a qualitative comparison is shown in the second and fourth columns in Fig. 9. The kernel learned by RKI tends to smooth out small-size reflections (yellow boxes) in its RCMap, leading to a downgraded reflection-removal result. In contrast, ours can successfully detect and remove the reflections. Moreover, the kernels learned by LKI and RKI are available in Sec.3 of the supplemental material.

Table 2. Network structure ablation study. w/o ∗: remove module ∗. M1 → M2: change M1 to M2. SLSM: Single-scale Laplacian submodule. GC: gradient clipping. Edge: edge features, namely the partial derivatives of the image with reference to x, y (refer to Fig. 4). Ĉ from Ift: disable MLSM in RDM and predict RCMap using the extracted features of [I, ĈI] before LSTM block. Fix MLSM: disable the fine-tuning of Laplacian kernel parameters.
occasionally generates GT RCMaps (the third column in Fig. 10) that incorrectly label some transmission-dominated regions with the reflection-dominated regions, even with relative intensity method (the last column in Fig. 10, \( \beta = 0.1 \)). We hypothesize that it is why our model based on \( \mathcal{L}_C \) surpasses the other two variants that use Eq. 16 and RKI in PSNR/SSIM scores, as shown in the last three rows of Tab. 3. Besides, the third column of Fig. 9 illustrates that the RCMap obtained under the supervision of \( \mathcal{L}_C^B (\beta = 0.3) \) contains mislabeling errors (red boxes), resulting in damages to non-reflection regions when estimating \( \hat{T} \). In Sec.5 of the supplemental material, we evaluate the detection accuracy further and show that our method can well protect non-reflection regions.

### 6. Conclusion

We develop a location-aware SIRR network to improve the quality of SIRR results substantially. Its key feature is that we leverage learned Laplacian features that can emphasize the strong reflections’ boundaries to locate and remove strong reflections, such as reflected highlights. The network has an RDM that takes multi-scale Laplacian features as inputs to detect reflections roughly. In the future, we plan to simplify our network’s design further to save the number of parameters and improve its inference speed on the mobile computing platform.

### Acknowledgements

We thank the reviewers for their constructive comments. Weiwei Xu is supported by NSFC (No. 61732016). This work was partially supported by a GRF grant from RGC of Hong Kong (Ref.: 11205620).
References


