

Supplementary Materials: R²SFD : Improving Single Image Reflection Removal using Semantic Feature Dictionary

Anonymous Authors

1 OVERVIEW

This supplementary is organized as follows. In Section 1, we provide implementation details of the proposed SFDNet architecture. In Section 2, we provide an analysis of the network computational complexity. In Section 3, we detail the training details for RAFENet and SFDNet. In Section 4, we provide an ablation study to analyse the impact of the number of images in the semantic feature dictionary (SFD). In Section 5, we provide a detailed analysis of the reflection aware features extracted by the RAFENet. In Section 6, we describe how to extended our algorithm to a user-in-the-loop approach. In Section 7, we provide some results on the similar images extracted by the proposed Coarse Semantic Search (CSS) module. We provide sample images from our dataset in Section 8 and some qualitative comparisons against state-of-the-art methods in Section 9.

2 NETWORK IMPLEMENTATION DETAILS

In this section, we provide additional details regarding the implementation of the proposed SFDNet. The proposed SFDNet architecture is shown in Fig. 1 for reference.

2.1 Encoder

The encoder consists of 2 Conv-Batchnorm-ReLU layers of stride 1 followed by 3 Conv-Batchnorm-ReLU layers of stride 2. All the convolutions used in the architecture have a kernel size of 3×3 . Each of the stride-1 convolutions have 32 output channels. The stride-2 convolutions have output channel sizes of 64, 128 and 256 respectively.

2.2 Decoder

The decoder consists of 3 Upsample-Conv-Batchnorm-ReLU layers followed by 2 Conv-Batchnorm-ReLU layers. We use bilinear interpolation for upsampling. All the convolutions have stride 1 and a kernel size of 3×3 . The first 3 convolutions have output channel sizes of 128, 64 and 32 respectively. The subsequent convolutions have output channel sizes of 32 and 3 respectively.

2.3 Residual Convolution Block (RCB)

Each of the RCB consists of 2 Conv-Batchnorm-ReLU layers of stride 1, kernel size 3×3 and output channel size of 256. The input to the RCB is added back to the output of the RCB.

2.4 Semantic Transfer Block

A schematic of the ST module is shown in Fig. 1. Let f_m denote the matched feature matrix \mathcal{F}_m reshaped back to $H \times W \times C$ dimension. We first concatenate the input features (f) and f_m along the channel dimensions. This is passed through a convolutional block consisting of two conv-batchnorm-reLU layers. Each of the convolutions have a kernel size of 3×3 and 256 outputs channels each. A gated

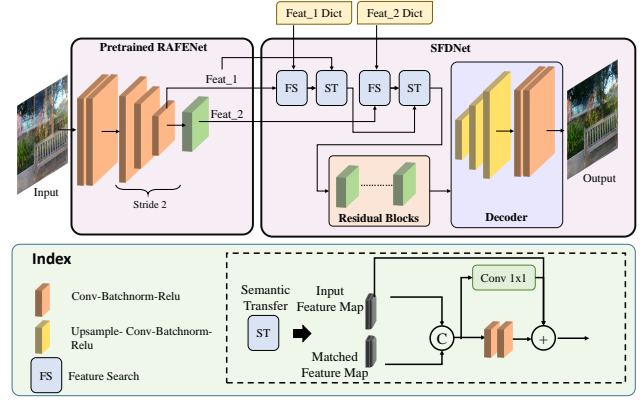


Figure 1: Semantic Feature Dictionary Network (SFDNet). The input is first passed through the pre-trained RAFENet. Next, the Feature Search (FS) module searches for the most similar features from the SFD ($Feat_1$ and $Feat_2$ dicts) for semantic transfer.

skip connection consisting of (1×1) convolution with 256 output channels is provided from the input to the output. The resultant feature map is then added back to f to learn the semantic transfer in a residual manner.

3 NETWORK COMPLEXITY ANALYSIS

In this section, we analyze the complexity of various components of the proposed method during inference. The VGG features required for Coarse Semantic Search (CSS) and the RAFF features are pre-computed for all the images in the image database. Since these are computed only once, their computations are asymptotically negligible over a large number of inferences. Hence we do not include this in the complexity analysis. We only analyze the components that needs to be executed for every inference, i.e, Coarse Semantic Search (CSS) and SFDNet. We also analyze the computations required for Feature Search (FS) and Semantic Transfer (ST) modules within the SFD. These results are summarized in Table 1. We use Multiply-Accumulate (MAC) operations to quantify the complexity of our methodology. We report the MAC operations required per output pixel.

We use 50000 images in our image database for Coarse Semantic Search (CSS). It can be observed that the computations required for CSS is negligible comparable to other modules in our pipeline. This is because every comparison in the CSS modules involves computing the cosine distance between two 1×1000 vectors, regardless of the input image shape. Hence our method is scalable to handle very large image databases. For example, increasing the size

Table 1: Complexity Analysis. M denotes million operations. $s = 10$ and the image database is of size 50000.

Component/Method	MACs/pixel
Coarse Semantic Search (CSS)	47
Feature Search	1.31M
Semantic Transfer	0.47M
Remaining components in SFDNet	1.89M
Total Computations	3.67M

of the database from 50000 to 1 million would increase the total computations only by $\sim 0.02\%$.

It can also be seen that Feature Search module accounts for $\sim 30\%$ of the total computations required for the model. These values are computed for $s = 10$, where s denotes the number of similar images selected from the image database. Increasing the value of s further would further increase the computations, with minimal improvement in accuracy of the model. Hence we used $s = 10$ for an optimal trade-off between computations and accuracy.

4 TRAINING DETAILS

We train the proposed RAFENet and SFDNet using PyTorch framework on a PC with NVIDIA Tesla V100 GPU and 32 GB RAM. The specific training details for these networks are detailed below.

4.1 RAFENet

We first pre-train the encoder backbone and the transmission branch of the RAFENet using synthetic training dataset for 250 epochs. The coefficient for cycle loss is set to zero and the reflection branch weights are frozen so that only the transmission branch is trained. Next, we unfreeze the weights for reflection branch and train for a further 250 epochs using all the three loss components. We used ADAM optimizer for training with an initial learning rate of 10^{-5} and a weight decay of 0.025.

4.2 SFDNet

To train the proposed SFDNet, we first load the *reflection-aware* feature extractor weights pre-trained using RAFENet into SFDNet. The SFDNet training consists of three phases. In the first phase, the SFDNet is trained for 500 epochs using synthetic dataset. Next, we train the SFDNet for a further 500 epochs using the proposed LSRR training dataset. During these training phases, we set SFD size (s) to 1, to improve training time. The feature extractor weights are also fine-tuned in these phases. For the final phase of training, we set the size of SFD to 10. We also freeze the feature extractor weights so that SFD can be pre-computed for all the training images, for faster training. The SFDNet is then trained for another 1000 epochs using our LSRR training dataset. We used ADAM optimizer for training with an initial learning rate of 10^{-5} and a weight decay of 0.025.

5 IMPACT OF THE SIZE OF SFD

In this section, we analyze the impact of the number of images (s) used to create the Semantic Feature Dictionary on the output PSNR.

Table 2: Impact of the number of images (s) used to create SFD. The metrics are averaged over SIR2 [4], Berkeley [6] and our LSRR datasets. We use $s = 10$ for an optimal trade-off between output quality and computational complexity (shown in bold).

s	PSNR (dB)	SSIM	Million MACs (per pixel)
1	25.85	0.900	2.49
10 (Ours)	26.26	0.902	3.67
100	26.45	0.908	15.47
SRNet [1]	25.35	0.896	2.96
RAGNet [3]	24.90	0.885	3.28
IBCLN-F [2]	24.36	0.880	4.81
ERRNet-F [5]	23.39	0.872	9.46

We evaluate our model using feature dictionaries extracted from 1, 10 and 100 images respectively, and summarize the results in Table 2. It should be noted that these s images are chosen from our image database consisting of ~ 50000 images using the Coarse Semantic Search module (CSS). The cost of computation required for CSS is also included within Table 2 (million MACs/pixel). From Table. 2, it can be seen that the model performance increases with s . However, the gains in PSNR increases only marginally when s is very large (100), while the computational cost increases significantly. Hence, we choose $s = 10$ for an optimal trade-off between PSNR and model complexity.

We also provide comparisons against the state-of-the-art methods SRNet [1], RAGNet [3], IBCLN [2] and ERRNet [5] for reference. It can be seen that our proposed method using $s = 1$ (one reference image) outperforms all the methods on both PSNR and SSIM metrics. Further, it can be seen that the proposed method with $s = 1$ has the lowest computational complexity compared to all the state-of-the-art methods. Moreover, it can be seen that using $s = 10$ significantly outperforms all the methods while having similar computational complexity as that of RAGNet [3].

6 ANALYSIS OF RAPE FEATURES

In this section, we provide a more detailed analysis of the impact of *reflection-aware* (RAFE) features. First, we conduct an experiment to analyze the ability of our proposed RAFENet to extract reflection-aware features across varying strengths of reflections. For this experiment, we first created synthetic testing sets with varying reflection strengths using the method proposed by [5]. We vary the value of blending factor α (Eq. 1, Main Paper) from 0.0 to 1.0 to create testing sets with varying strengths of reflections. In this experiment, a lower value of α represent a lower strength of reflection, with 0 representing an image without any reflections. Next, we use the proposed RAFENet to extract features from the input corrupted with reflections (I) and the ground-truth transmission image (I_t). Let \mathcal{F}_I and \mathcal{F}_T denotes the features extracted from the input and ground truth images respectively. Next we define a metric to measure the reflection invariance of the features as follows:

$$\mathcal{D}_{rafe} = \|\mathcal{F}_I - \mathcal{F}_T\|_2 \quad (1)$$

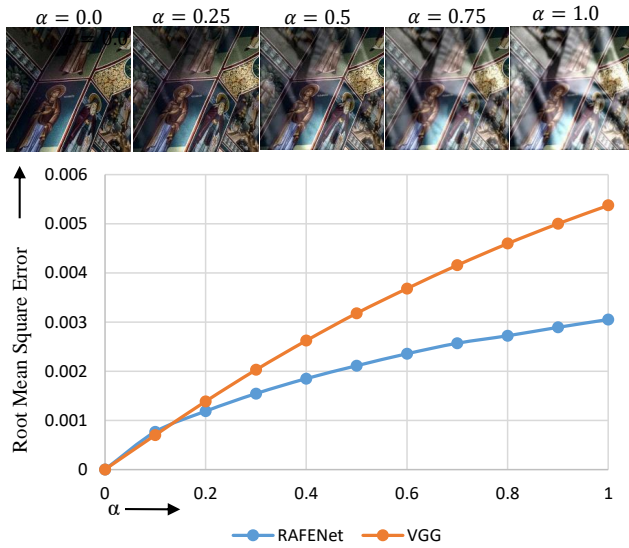


Figure 2: Reflection Invariance of extracted features. The features extracted using RAFENet is more reflection aware compared to the ones extracted using VGG, especially for higher strengths of reflections (α). We also provide sample images for different values of α .

Here $\|\cdot\|_2$ denotes the Root Mean Square Error. This metric measures the distance between features extracted from reflection corrupted input images and the reflection free groundtruth image. If the network is able to extract features in a reflection-aware manner, then the features extracted from both the images would be similar. Hence, a lower metric would correspond to a better ability of the network to generate reflection aware features. Next, we perform the same experiment using VGG features. Let \mathcal{D}_{vgg} denote the measure of reflection invariance of the features extracted using VGGNet. The results of this experiment are summarized in Fig. 2. From the figure, it can be seen that for very low values of α (Eg: 0.01, 0.1 etc) the value of \mathcal{D}_{rafe} is similar to that of \mathcal{D}_{vgg} . However, as the value of α increases, \mathcal{D}_{vgg} increases much faster than \mathcal{D}_{rafe} . This shows that RAFENet is able to extract reflection-aware features much better than conventional VGG based feature extractors in the presence of reflections.

We also conduct an experiment to justify our choice of using *Feat_1* and *Feat_2* as the reflection aware features. For this experiment, we created models where only *Feat_1* or *Feat_2* was used as the reflection aware features. The results are summarized in Table. 3. It can be seen that the proposed method using both *Feat_1* and *Feat_2* is able to generate better results compared to using only one set of features.

7 TOWARDS A USER-GUIDED APPROACH

Our proposed method can easily be extended as user-guided approach for reflection removal. In this approach, the user can manually provide one or more reference images to aid the removal of reflection. This eliminates the need to automatically find similar images from a database. This would also allow the user to control the

Table 3: Analysis of RAFF Features. Evaluated on wildscene dataset

SI No.	Feature Extractor	PSNR	SSIM
1	RAFF(Only Feat_1)	26.28	0.9038
2	RAFF(Only Feat_2)	26.15	0.905
Ours	RAFF (Feat_1 & Feat_2)	26.62	0.9232

amount of reflection removal by provided different semantic scenes as a context. An example of this approach is shown in Figures 3 and 4.

In Fig. 3 we show an image example from LSRR dataset, with four different reference images provided by the user, and the corresponding outputs. It can be observed that the image restoration quality improves with increased semantic similarity. In Fig. 4, we analyse the impact, if a stereo image captured with the input image is provided as a reference image. It can be seen that the outputs obtained using a reflection-corrupted stereo reference image is almost able to match the output quality obtained by using a semantically similar image as reference. This is because, the reference image represents the same scene, and the network is able to extract relevant semantic content using the proposed reflection aware feature extraction.

8 ANALYSIS OF CSS MODULE

In this section, we provide a few image examples of the similar images retrieved by our Coarse Semantic Search module. These results are shown in Fig. 5. It can be seen that the CSS module is able to find images with similar semantic or texture content as that of the provided input image.

9 LSRR DATASET SAMPLES

We provide some sample images from our proposed LSRR dataset in Figs. 6 and 7. Our dataset consists of 2650 high resolution images with real reflections and their corresponding reflection-free ground truth images. The dataset was captured using a smartphone in various indoor and outdoor locations and in different lighting conditions. Our dataset consists of images with strong as well as weak reflection components for better variability and generalization. To the best of our knowledge, the proposed dataset is the largest dataset consisting of images with real reflection and their corresponding aligned ground truths.

10 RESULTS

In this section, we provide more comparative results against state of the art methods. In Figures 8 - 11, we provide comparisons on the proposed LSRR Dataset. We also provide comparisons on DSLR50 dataset [5] in Fig. 12 - 13 and on Berkeley dataset [6] in Fig.14.

REFERENCES

- [1] Zhikai Chen, Fuchen Long, Zhaofan Qiu, Juyong Zhang, Zheng-Jun Zha, Ting Yao, and Jiebo Luo. 2024. A Closer Look at the Reflection Formulation in Single Image Reflection Removal. *IEEE Transactions on Image Processing* (2024).
- [2] Chao Li, Yixiao Yang, Kun He, Stephen Lin, and John E Hopcroft. 2020. Single image reflection removal through cascaded refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3565–3574.



Figure 3: Reference Similarity Analysis. The network output can be controlled by changing the reference image. It can be seen that the reflection removal gets better as the similarity of the reference image increases

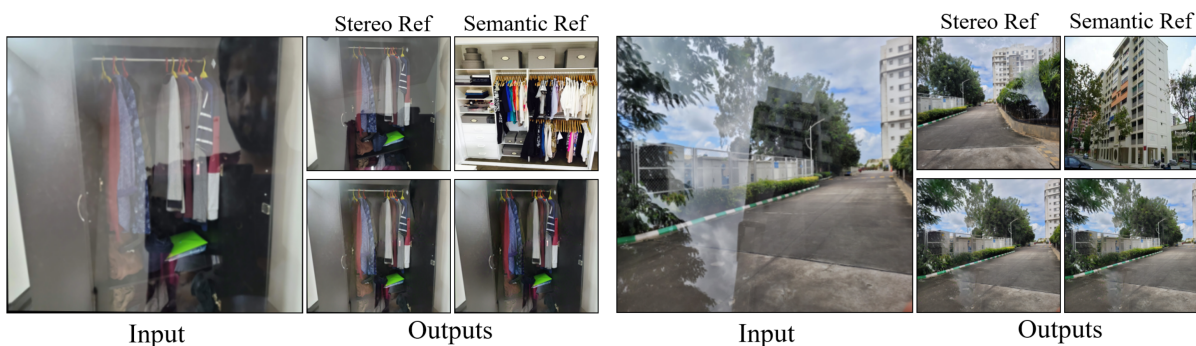


Figure 4: Stereo Reference Image. A stereo image captured along with the input image is provided as reference image (Top Row). The restoration quality is almost as good as that obtained using a semantic reference image.



Figure 5: Examples of similar images obtained from the CSS module

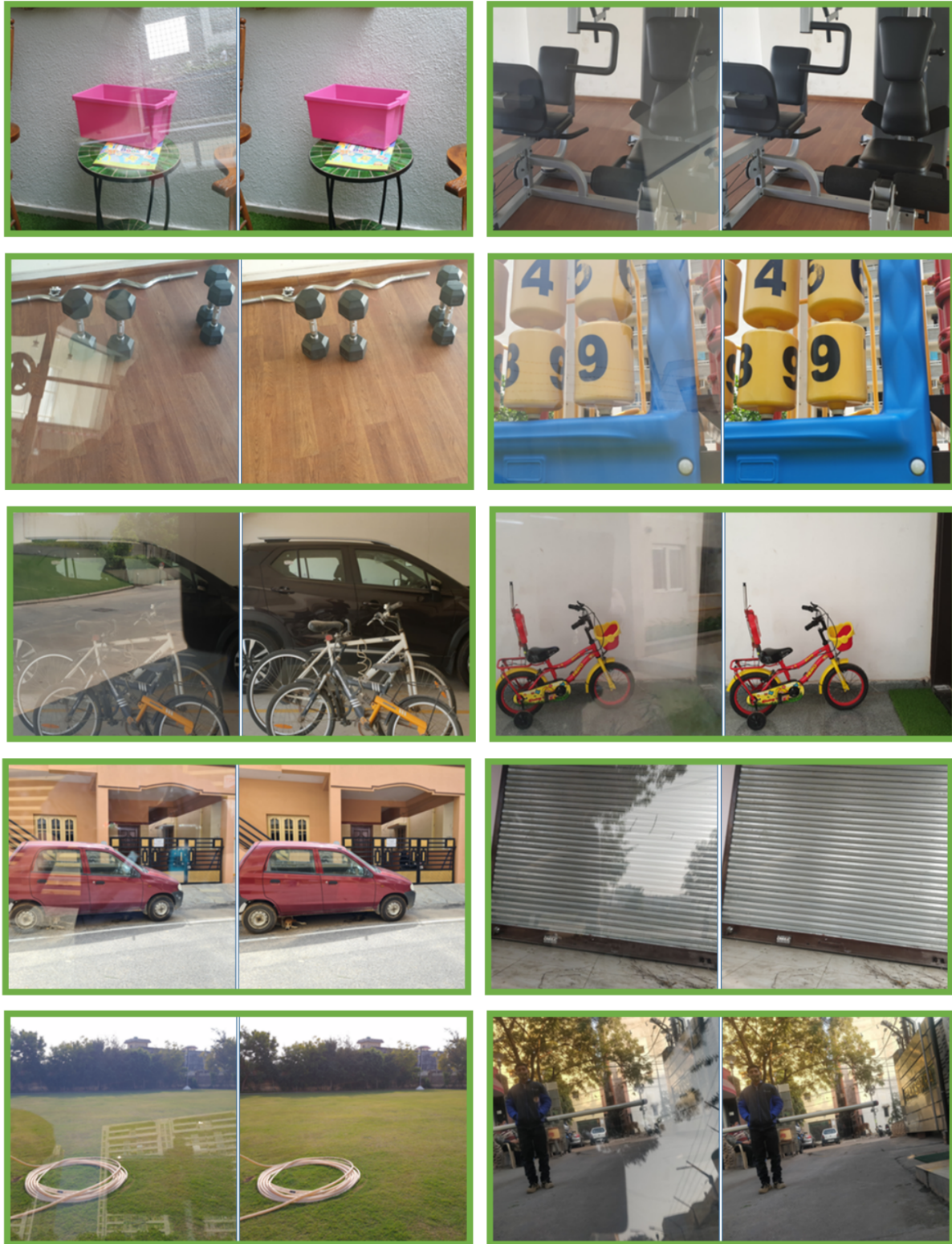


Figure 6: Samples from our LSRR dataset. Left image: Input; Right Image: ground truth



Figure 7: Samples from our LSRR dataset. Left image: Input; Right Image: ground truth

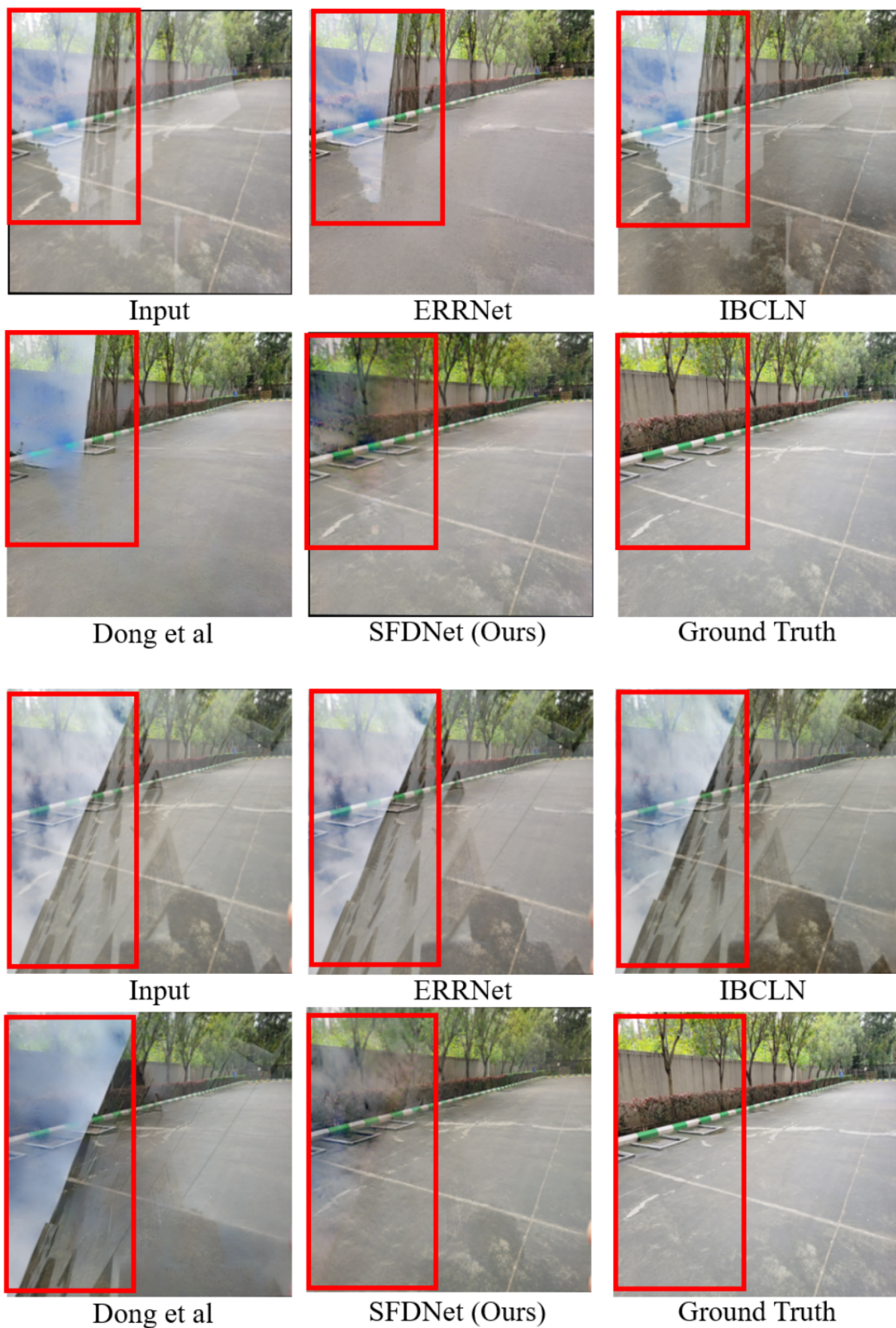


Figure 8: Comparisons against state-of-the-art methods on LSRR outdoor datasets

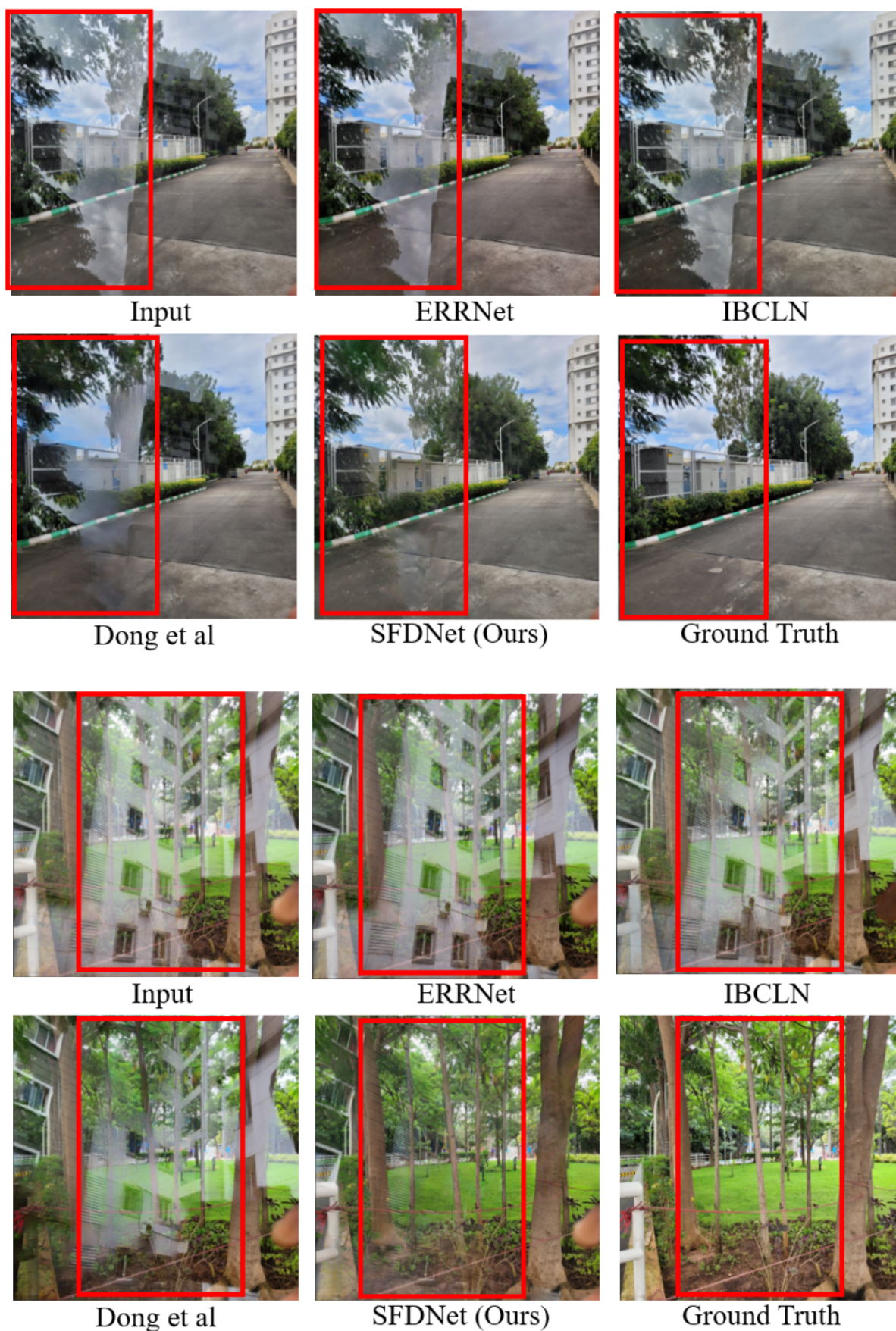


Figure 9: Comparisons against state-of-the-art methods on LSRR outdoor datasets

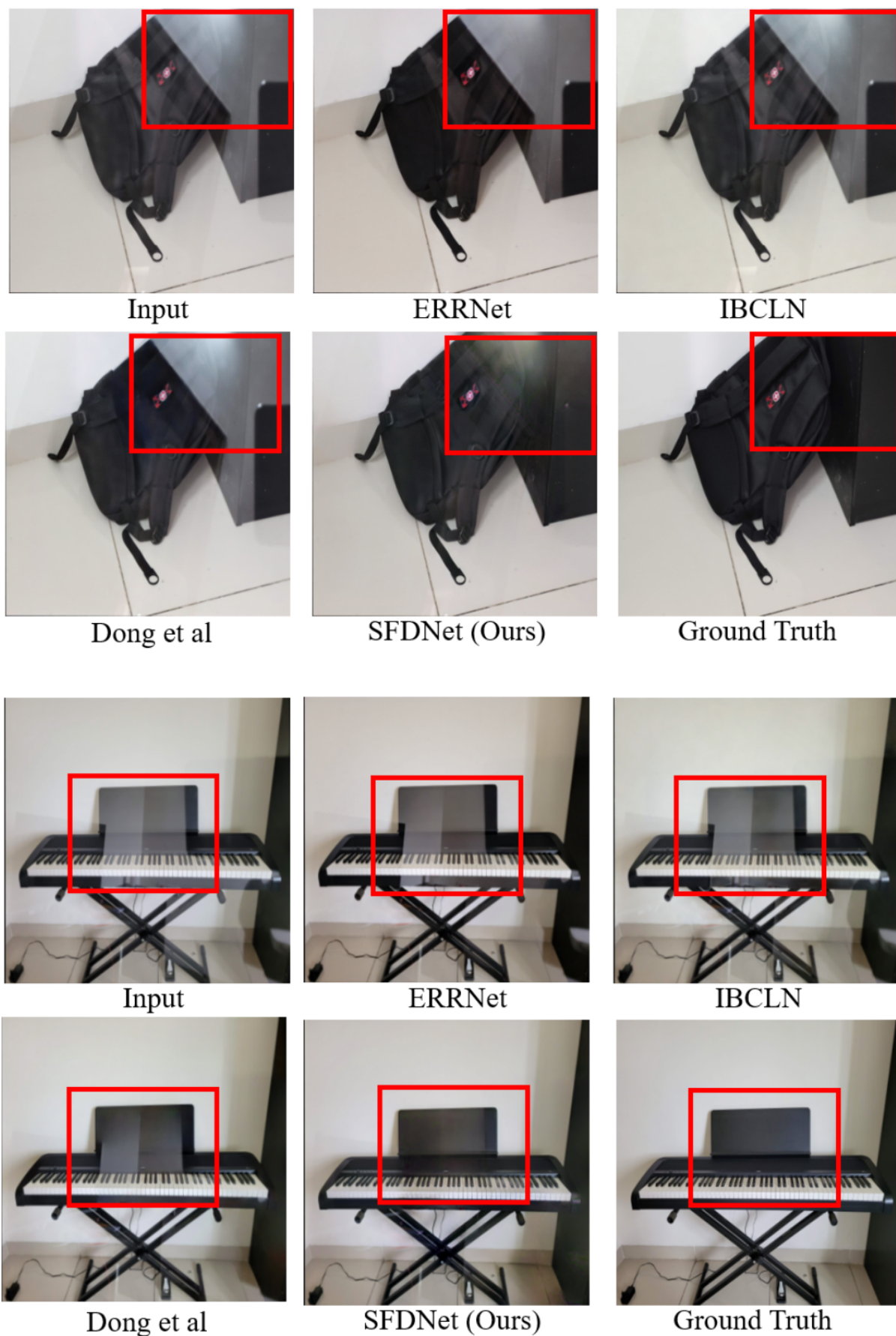


Figure 10: Comparisons against state-of-the-art methods on LSRR indoor datasets

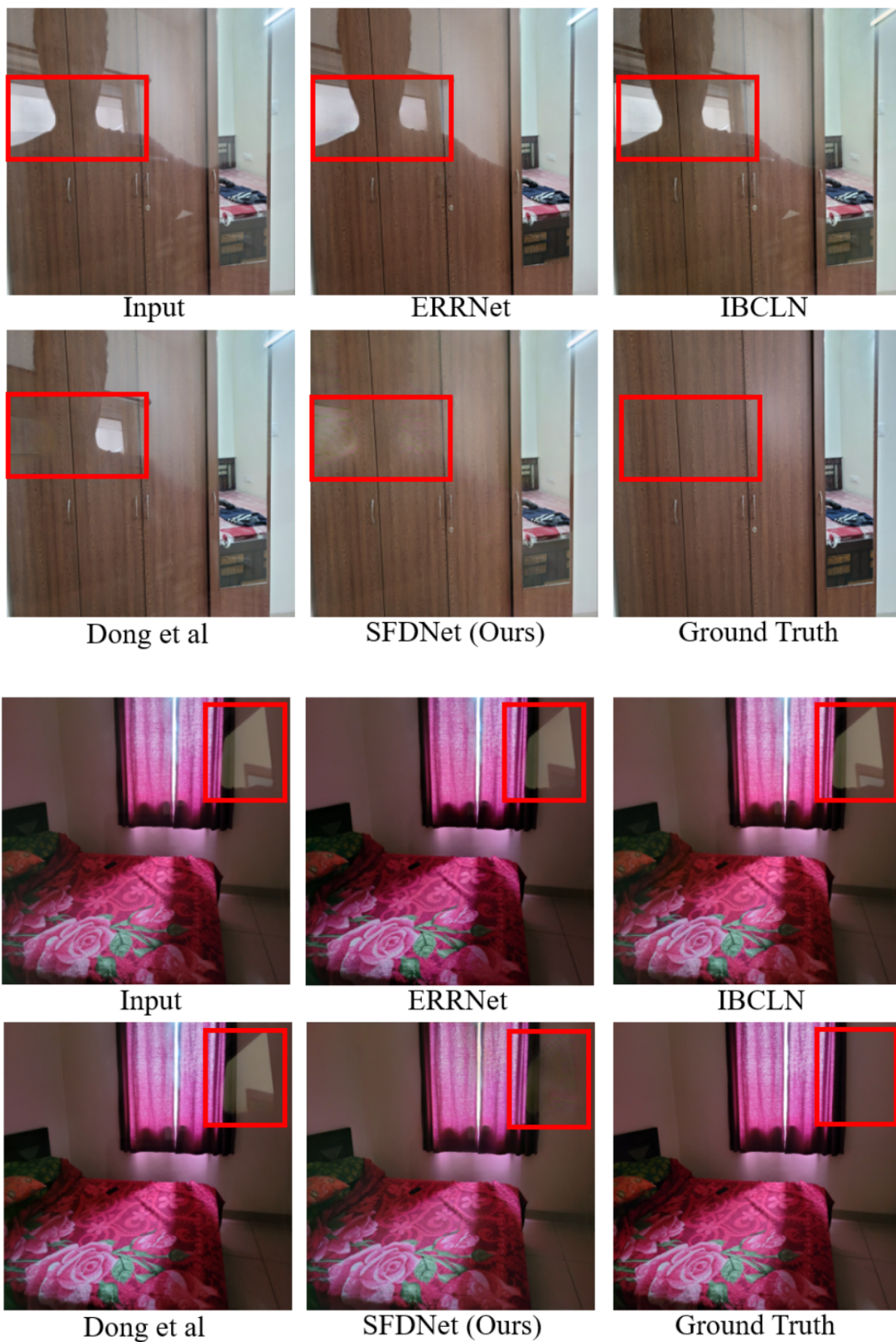


Figure 11: Comparisons against state-of-the-art methods on LSRR indoor datasets

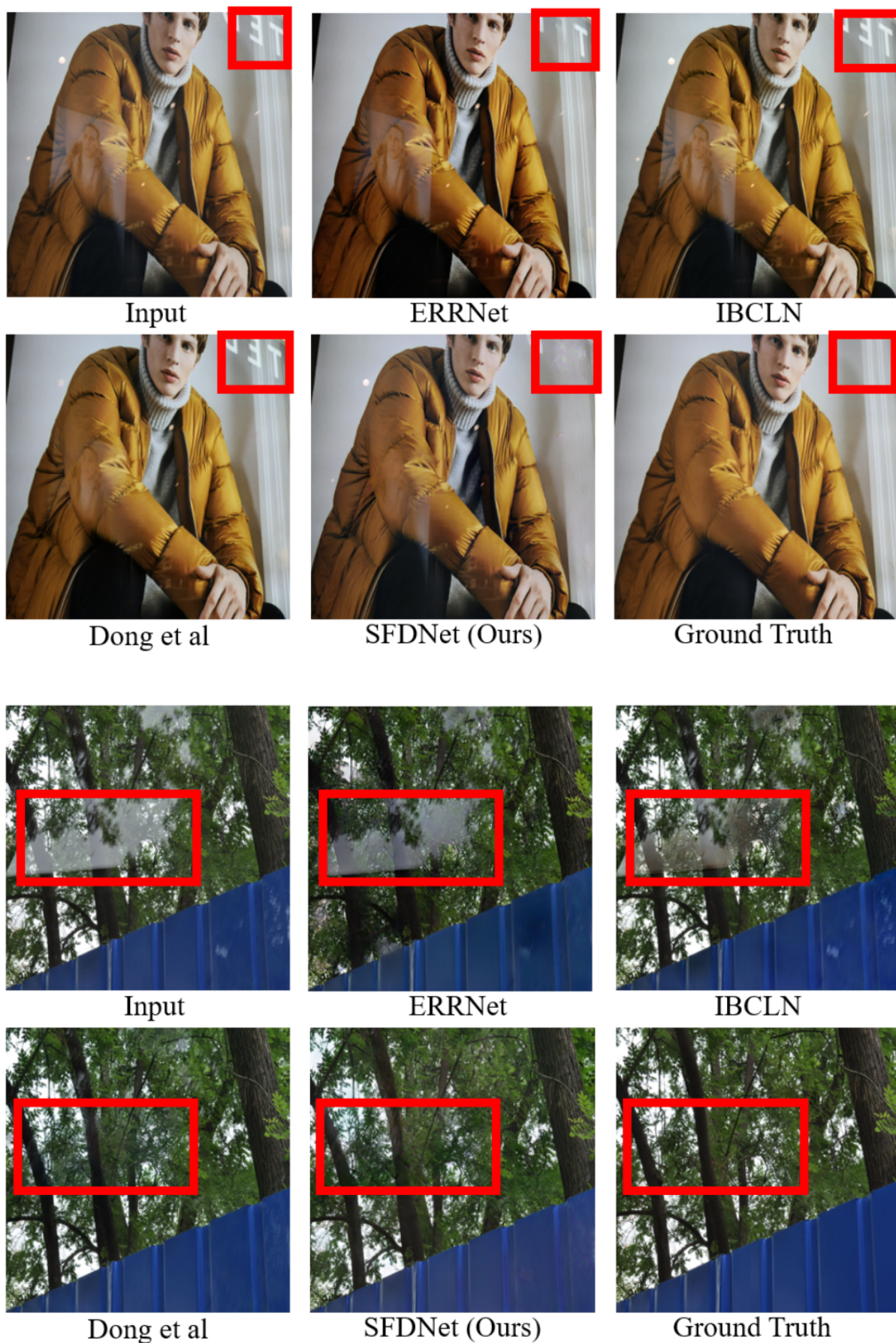


Figure 12: Comparisons against state-of-the-art methods on ERRNet DSLR50 dataset [5]

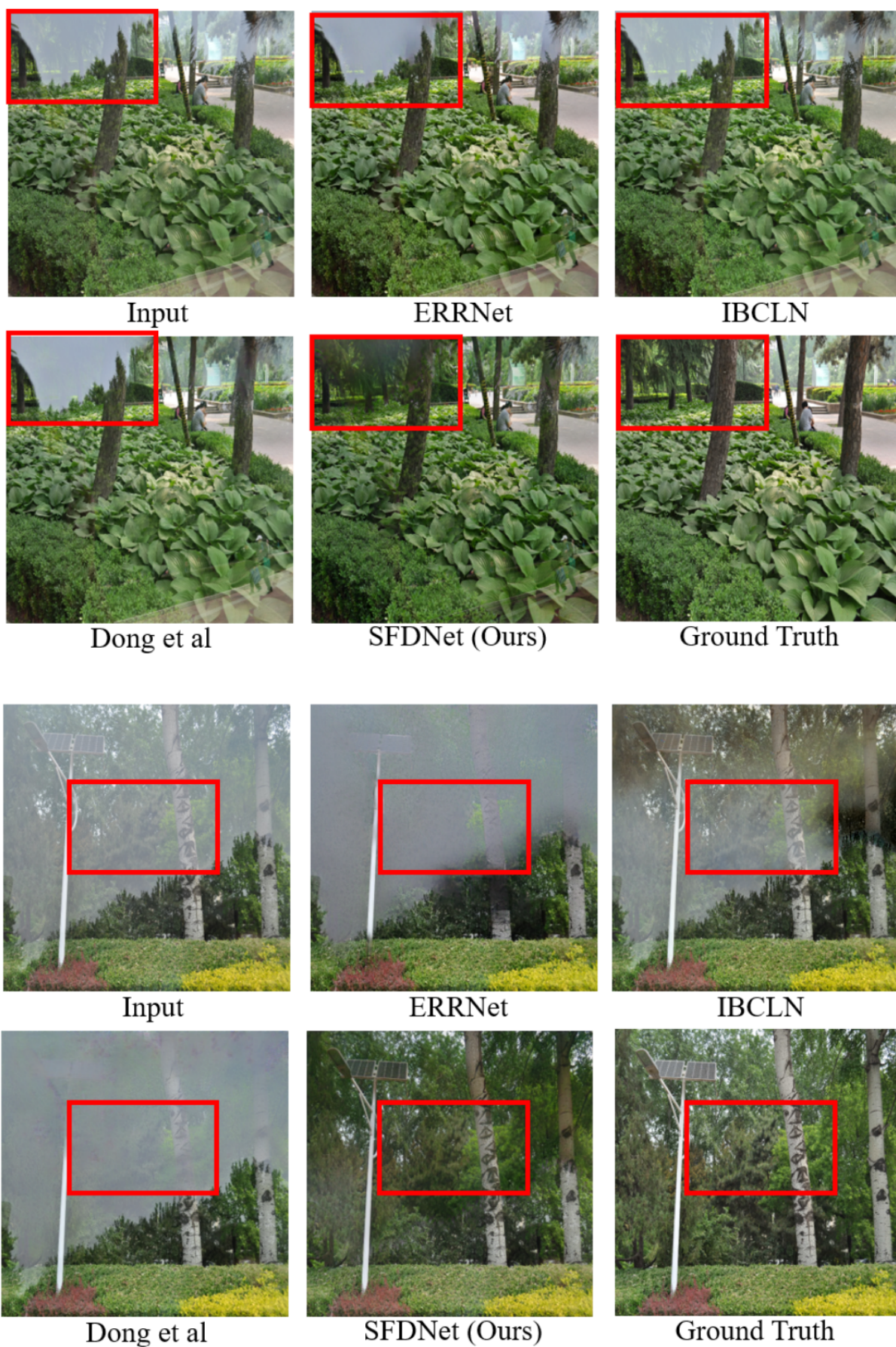


Figure 13: Comparisons against state-of-the-art methods on ERRNet DSLR50 dataset [5]

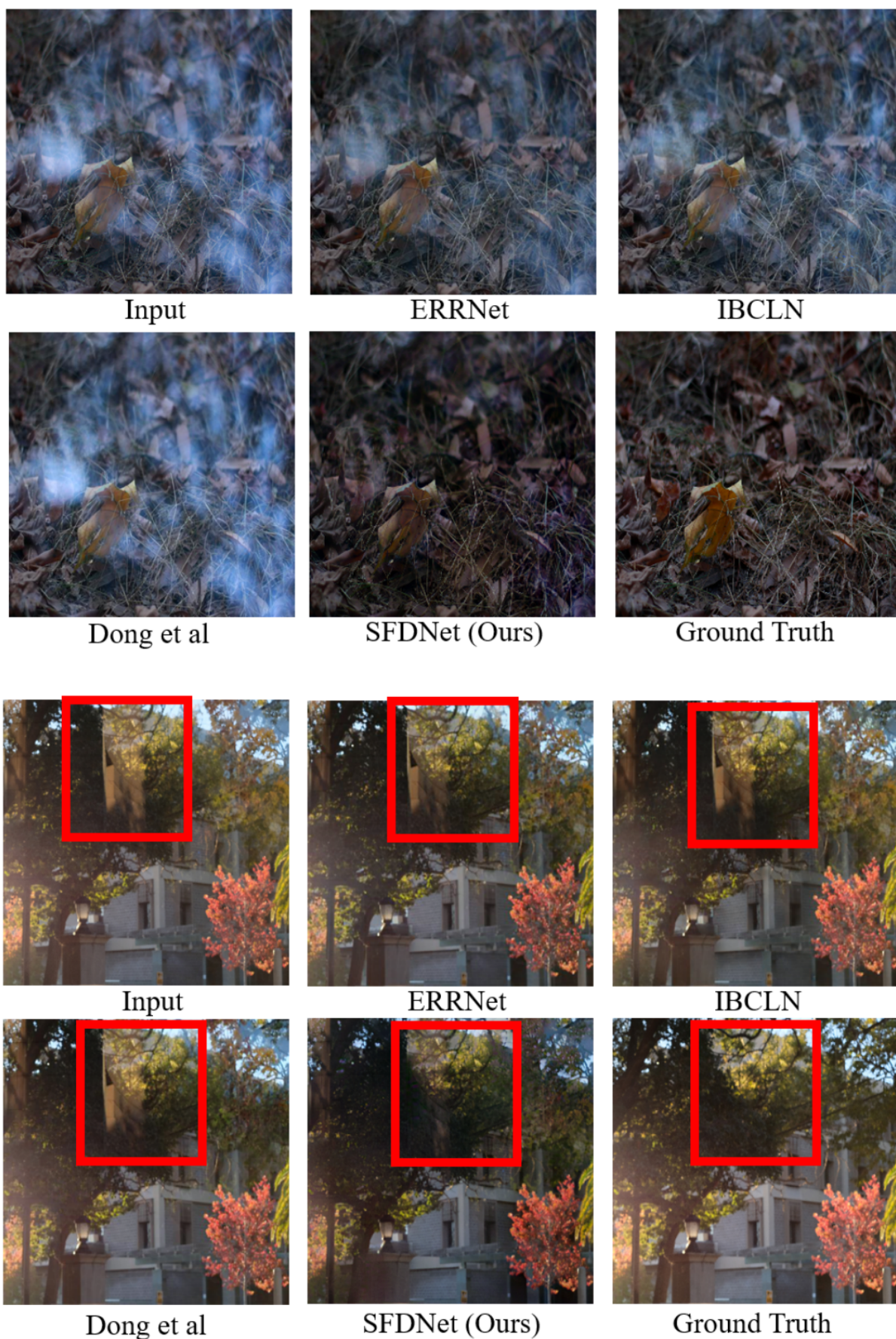


Figure 14: Comparisons against state-of-the-art methods on Berkeley dataset [6]

- [3] Yu Li, Ming Liu, Yaling Yi, Qince Li, Dongwei Ren, and Wangmeng Zuo. 2023. Two-stage single image reflection removal with reflection-aware guidance. *Applied Intelligence* 53, 16 (2023), 19433–19448.
- [4] Renjie Wan, Boxin Shi, Ling-Yu Duan, Ah-Hwee Tan, and Alex C. Kot. 2017. Benchmarking Single-Image Reflection Removal Algorithms. In *International Conference on Computer Vision (ICCV)*.
- [5] Kaixuan Wei, Jiaolong Yang, Ying Fu, David Wipf, and Hua Huang. 2019. Single image reflection removal exploiting misaligned training data and network enhancements. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8178–8187.
- [6] Xuaner Zhang, Ren Ng, and Qifeng Chen. 2018. Single image reflection separation with perceptual losses. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4786–4794.