Multi-view Self-supervised Disentanglement for General Image Denoising (Supplementary Material)

Contents

1. Introduction

This document provides supplementary materials for the main paper. Specifically, Section 2 presents thorough details of the model training used in our experiments followed by an ablation study of parameters. The supplemental denoising quantitative evaluation is presented in Section 3. Section 4 explores more low-level applications, including image super-resolution and inpainting, with comparison to the proposed MeD. Section 5 and Section 6 provide further details regarding the datasets used in our research and the methods for synthesising noise and downscale corruption. Finally, we present more qualitative results, with comparison to other methods, in Section 7.

2. Denoising Training Settings

For methods that use *Swin-Tx* model, *e.g*. N2C [11], MeD and N2N [10], we use the same set of hyperparameters for training. Prior to the formal experiment, we conducted some pilot experiments to test and select the final choice of hyperparameters on N2C. Following [23, 11], we use 48×48 random crops from DIV2K images. The training process is performed using a mini-batch size of 8 and undergoes a total of 500K iterations. We use Adam with β_1 = 0.9 and β_2 = 0.99 and learning rate of 10^{-4}, which decays every 100K with decay ratio 0.5.

Since the model is not the focus of this work, we use a simple 2-layer *Swin-Tx* for a fair comparison with other non-Transformer models, and is less likely to overfit the synthesised training noise distribution.

The influence of the size of the *Corruption Pool* on the performance of MeD and N2C [11] is demonstrated in Figure S1 (b). The experiments are started from only fixed Gaussian noise, then train on Gaussian noise with random sigma values. Finally, we expand the corruption pool from only Gaussian noise to more noise types and even with different types of down-scale and inpainting mask operations.

Table S1. Analysis of hyper-parameters for the loss terms.

Loss Hyperparameters for $[\mathcal{L}^{\mathcal{X}}, \mathcal{L}^{\mathcal{N}}, \mathcal{L}^{\mathcal{C}}, \mathcal{L}^{\mathcal{M}}]$	Gaussian Noise $\hat{\sigma}=25$	$\hat{\sigma}=50$	$\hat{\sigma}=75$
[1.0, 0.5, 0.5, 0.025]	31.18/ 0.8839	27.68/ 0.7765	25.74/ 0.7172
[0.5, 1.0, 0.5, 0.025]	31.04/ 0.8813	27.87/ 0.7788	25.86/ 0.7190
[0.5, 0.5, 1.0, 0.025]	30.85/ 0.8752	27.96/ 0.7794	25.92/ 0.7199
[1.0, 1.0, 1.0, 0.025]	**31.31/ 0.8876**	**28.05/ 0.7810**	**26.01/ 0.7216**
[1.0, 1.0, 1.0, 0.050]	31.29/ 0.8870	27.58/ 0.7659	24.29/ 0.6931
[1.0, 1.0, 1.0, 0.000]	31.20/ 0.8832	26.24/ 0.7391	23.78/ 0.6517

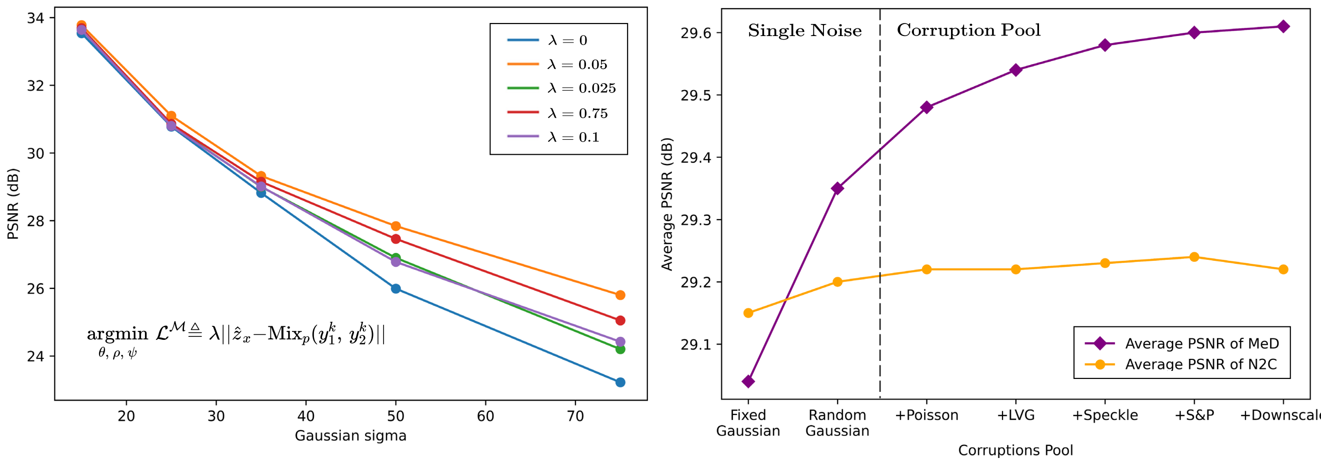
2.1. Hyperparameter Analysis

Prior to finalising the training procedure, we conducted experiments to analyse the impact of different hyperparameters associated with the loss terms in our model. Specifically, we tested varying the weighting factors $\mathcal{L}^{\mathcal{X}}$, $\mathcal{L}^{\mathcal{N}}$, $\mathcal{L}^{\mathcal{C}}$ and λ for the *Noise Reconstruction loss*, *Scene Reconstruction loss*, *Cross Compose loss*, and *Mix Scene reconstruction loss*, respectively. The analysis is shown in Figure S1 (a) and Table S1

Firstly, we conducted the experiment for analysing the value λ in Figure S1 (a). The orange (top) curve represents the performance of the optimal choice $\lambda = 0.05$.

$\mathcal{L}^{\mathcal{X}}$, $\mathcal{L}^{\mathcal{N}}$ and $\mathcal{L}^{\mathcal{C}}$ are tested from 0.5 to 1, with the best results obtained at $\mathcal{L}^{\mathcal{X}} = 1$, $\mathcal{L}^{\mathcal{N}} = 1$ and $\mathcal{L}^{\mathcal{C}} = 1$.

Based on these experiments, we selected hyperparameters of $\mathcal{L}^{\mathcal{X}} = 1$, $\mathcal{L}^{\mathcal{N}} = 1$, $\mathcal{L}^{\mathcal{C}} = 1$, and $\lambda = 0.025$ for all



(a) Analysis on λ for Bernoulli Manifold Mixture

(b) Analysis on the training Corruptions Pool

Figure S1. **Ablation experiments.** All models are tested with Gaussian noise removal on the CBSD68 dataset, unless otherwise specified. (a) MeD with Bernoulli Manifold Mixture Loss achieves the best performance at $\lambda = 0.05$ (the orange/top curve). (b) The performance of N2C and MeD is assessed while varying trained input corruptions. The findings suggest that MeD benefits much better from a diverse training corruption pool (the "+" sign in the horizontal axis indicates the further inclusion of a corruption in the corruption pool.).

Table S2. Supplementary quantitative comparison of different methods on CBSD68 dataset [12] for synthetic Gaussian noise. The experiments were conducted on fixed and random variance respectively. The best results are highlighted in **bold**, while the second best is underlined

Training Schema	Test $\hat{\sigma}$	Noisy/ Clean N2C [11]	Noisy/ Clean DBD$_4$ [8]	Noisy/ Noisy N2N [10]	Noisy/ Noisy N2S [3]	Noisy/ Noisy R2R [15]	Invariant Feature LIR [6]	Invariant Feature MeD
Gaussian $\sigma = 15$	15	33.21/ 0.9194	33.17/ 0.9179	33.16/ 0.9175	32.88/ 0.9099	31.19/ 0.8752	29.29/ 0.8118	**33.20/ 0.9181**
	25	28.04/ 0.7588	26.36/ 0.7473	26.56/ 0.7521	26.27/ 0.7456	23.51/ 0.5529	21.80/ 0.4802	**29.80/0.8401**
	50	19.89/ 0.3755	19.86/ 0.3741	19.68/ 0.3672	16.13/ 0.1978	16.09/ 0.2080	11.12/ 0.1217	**23.51/ 0.5529**
	75	17.16/ 0.2562	16.62/ 0.2314	14.59/ 0.1561	14.99/ 0.1642	14.09/ 0.1280	11.10/ 0.1042	**20.47/ 0.3870**
Gaussian $\sigma = 50$	15	30.69/ 0.8497	30.70/ 0.8478	30.80/ 0.8619	29.87/ 0.8267	28.15/ 0.7872	29.44/ 0.8011	**30.88/ 0.8799**
	25	29.81/ 0.8140	29.63/ 0.8182	29.54/ 0.8256	29.54/ 0.8256	28.89/ 0.8099	28.95/ 0.7967	**30.19/ 0.8218**
	50	28.56/ 0.7721	28.32/ 0.7765	28.30/ 0.7490	28.19/ 0.7802	27.80/ 0.7547	28.02/ 0.7682	**28.56/ 0.7835**
	75	22.60/ 0.5877	22.47/ 0.6042	22.45/ 0.5759	21.69/ 0.5433	21.42/ 0.5881	20.25/ 0.5368	**25.63/ 0.7372**

denoising training in our work. This provides an optimal balance between the different objectives.

3. Additional Denoising Evaluation

In this section, we present additional quantitative evaluations of our denoising results, which were not included in the main paper. The results are in Table S2 with additive Gaussian Noise, supplement to Table 1 in the main paper.

3.1. Generalisation on Unseen Noise Removal

To evaluate the generalisation ability of trained models on unseen noise removal, we conducted an additional experiment with more methods using only a single noisy image in Table S3. The results show that our method achieves comparable performance to state-of-the-art methods specially designed for Gaussian noise removal, and outperforms all compared methods on other noise types, while training only on the Gaussian noise. This further highlights the generalisation ability of our approach in handling unseen and unfamiliar noise distributions.

3.2. Further Analysis on Real-world Generalisation

In the main paper, we have shown that our method generalises well to real-world scenarios when trained only on synthetic data. Moreover, here we conduct experiments on training with a real-world dataset (SIDD without GT), and report the test results on SIDD and PolyU in Table S4.

Analysis: Comparing Table S4 and Table 4 in the main paper, it shows that all methods have significant improvement in performance on SIDD after training on SIDD, but little improvement on PolyU.

Table S3. Performance comparison of single-view approaches and Ours training on Gaussian noise and testing on various noise types.

Noise Type	DIP [17]	NAC [19]	S2S [16]	IDR [24]	Restormer [21]	MeD (Ours)
Gaussian, $\hat{\sigma} \in [25, 75]$	25.62/ 0.7017	27.13/ 0.7391	27.71/ 0.7622	28.52/ 0.8061	**29.10/ 0.8250**	28.45/ 0.8057
Speckle, $\hat{v} \in [25, 50]$	30.14/ 0.8574	31.55/ 0.8859	31.83/ 0.8980	28.62/ 0.8763	30.12/ 0.8557	**33.48/ 0.9115**
S&P, $\hat{r} \in [0.3, 0.5]$	28.62/ 0.7957	29.89/ 0.8741	30.57/ 0.9053	27.26/ 0.7544	23.09/ 0.6381	**30.84/ 0.9135**
AVG	28.13/ 0.7849	29.52/ 0.8330	30.04/ 0.8552	28.13/ 0.8123	27.44/ 0.7729	**30.92/ 0.8770**

Table S4. Train and test both on real-world datasets (PSNR/SSIM).

Method	MAC (G)	Supervised	Trained with	SIDD [1]	PolyU [20]
Restormer [21]	140.99	✓	Real (SIDD)	40.06/ 0.9601	36.38/ 0.9588
NAFNet [5]	63.6	✓	Real (SIDD)	**40.31**/ 0.9667	27.36/ 0.9225
N2N [11]	26.18	✗	Real (SIDD)	32.82/ 0.7297	36.22/ 0.9679
N2S [3]	26.18	✗	Real (SIDD)	30.98/ 0.6018	36.41/ 0.9721
CVF-SID [14]	77.86	✗	Real (SIDD)	34.71/ 0.9179	33.00/ 0.8768
MM-BSN [22]	339.46	✗	Real (SIDD)	37.37/ 0.9362	35.40/ 0.9484
MeD (Ours)	26.18	✗	Synthetic (NP)	35.81/ 0.8278	38.65/ 0.9855
MeD (Ours)	26.18	✗	NP + SIDD	**37.52/ 0.9434**	**38.91/ 0.9894**

Considering that collecting real data is expensive and sometimes infeasible compared to synthetic data and, as our following experiments show, generalising to new real datasets (real-to-real) is another issue (since the noise distributions are different), the model trained on synthetic noise data is more feasible and practical.

4. More Application Exploration

4.1. Experiment on Image Super-resolution

Figure S2 and Figure S3 show the qualitative results of our method for ×3 and ×4 super-resolution on Set5 dataset [4], compared with RCAN [25] and DASR [18]. It shows that our method achieves better performance than these methods, by using a corruption pool that contains both noise and down-scaling process.

4.2. Experiment on Image Inpainting

Evaluation is performed on Set11 [9]. Please see Figure S4 for two examples. It can be seen although our method is not designed for inpainting, we can still achieve better performance than sate-of-the-art methods such as DIP [17] and S2S [16].

5. Datasets

We used five different datasets to train and evaluate the denoising methods: DIV2K [2], CBSD68 [12], SIDD [1], CC [13], and PolyU [20].

DIV2K: DIVerse 2K resolution high-quality images [2] (DIV2K) contain 800 high-resolution images with a resolution of 2K or 4K. To train our denoising method, we added different types and levels of noise to the DIV2K dataset.

CBSD68: CBSD68 dataset [12] contains 68 colourful images with various levels of synthesising noise. These images were obtained from a range of sources, including natural scenes and synthetic images.

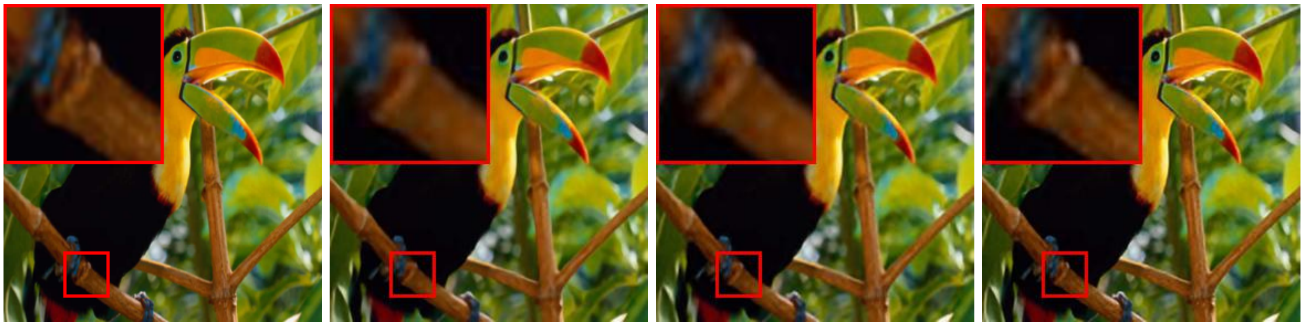
SIDD: The Smartphone Image Denoising Dataset [1] (SIDD) is a large-scale real-world dataset containing 24,000 images captured by smartphone cameras in ten scenes with varying lighting conditions. The ground truth images for the SIDD dataset are provided along with the noisy images in the dataset.

CC: Cross-Channel Image Noise Modeling [13] (CC) is another real-world dataset which contains 11 static scenes captured by three different consumer cameras. For each scene, it contains one temporal image and the precomputed temporal mean and covariance matrix data.

PolyU: PolyU dataset [20] is comprised of 40 different scenes captured by cameras. It contains the original image corrupted by realistic noise and the ground truth version which is obtained by averaging multiple exposures to remove the noise.

We also used Set5 dataset [4] and Set11 [9] to evaluate the super-resolution and image inpainting performances.

Set5 dataset: We use Set5 dataset [4] for super-resolution task. The Set5 dataset [4] consists of 5 high-quality images with different contents, including “baby”, “bird”, “butter-

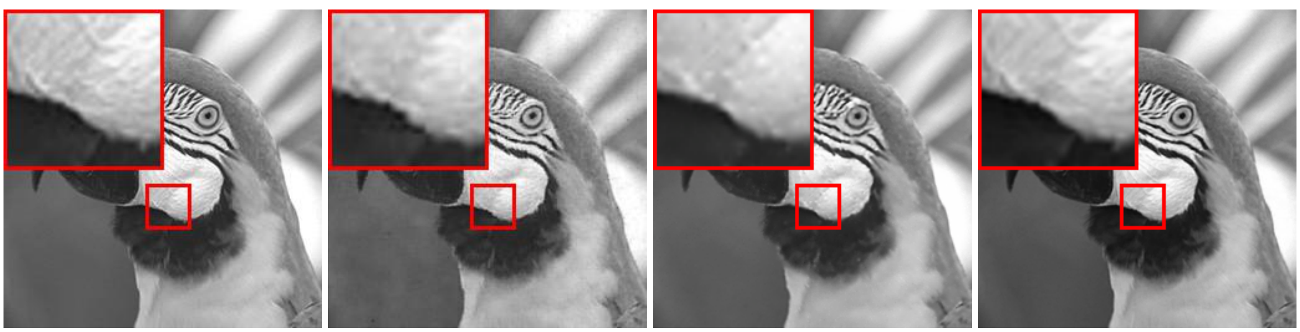


Set5 “Bird” [4]	RCAN [25]	DASR [18]	MeD (Ours)
PSNR/SSIM	34.89/ 0.9512	34.42/ 0.9364	**36.66/ 0.9747**

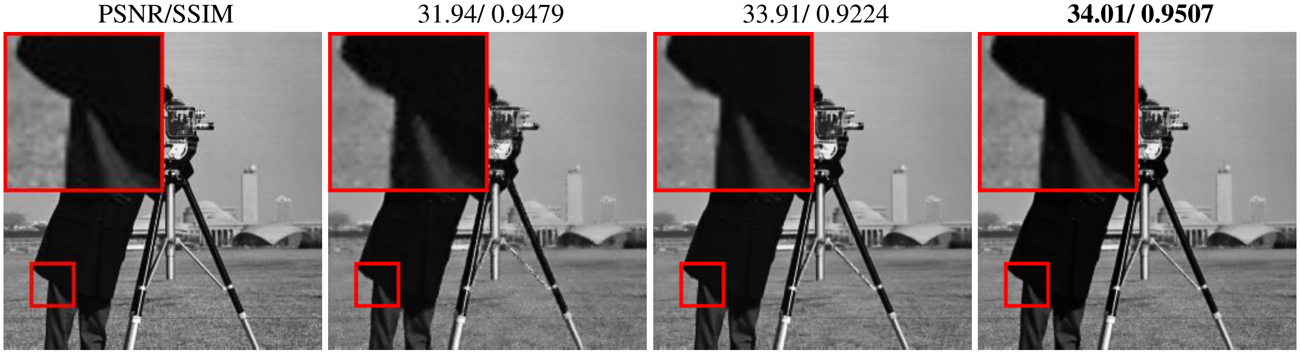
Figure S2. Visual comparison of image super-resolution (×3) methods on Set5 “Bird” [4] images.

Set 5 “Butterfly” [4]	RCAN [25]	DASR [18]	MeD (Ours)
PSNR/SSIM	30.91/ 0.9459	30.82/ 0.9527	**31.12/ 0.9636**

Figure S3. Visual comparison of image super-resolution (×4) methods on Set5 “Butterfly” [4] images.



Set 11 “Parrots” [4]	DIP [17]	S2S [16]	MeD (Ours)
PSNR/SSIM	31.94/ 0.9479	33.91/ 0.9224	**34.01/ 0.9507**



Set 11 “Cameraman” [4]	DIP [17]	S2S [16]	MeD (Ours)
PSNR/SSIM	30.97/ 0.9778	33.37/ 0.9355	**34.99/ 0.9478**

Figure S4. Visual comparison of image Inpainting methods on Set11 [9] images.

fly", "head", and "woman". Each of the images in the Set5 dataset has a magnifying factor of 2, 3, or 4, allowing us to evaluate the performance of our image super-resolution model across a range of magnification factors.

Set11 dataset: We compare our method (MeD) with DIP [17] and S2S [16] on image inpainting tasks using the Set11 dataset [9], which contains 11 grayscale images.

6. Synthesising Noisy Data and Downsampling Corruptions

We utilise the Pillow library[1] in Python to synthesise noisy data and perform downsampling corruptions.

6.1. Synthesising Noise

To evaluate the performance of our proposed algorithm, we synthesised noisy images using several types of noise models, including Gaussian, Local Variance Gaussian, Poisson, Speckle, and Salt-and-Pepper.
Remark: The original pixel value at position (i, j) in the image can be notated as $I(i, j)$ and the noisy pixel value can be notated as I_{noisy}.

Gaussian Noise: Gaussian noise is a type of additive noise that is commonly found in digital images. It is modelled as a normal distribution with zero mean and a standard deviation σ. To synthesise Gaussian noise, we added Gaussian-distributed noise to the original image. Specifically, we added Gaussian noise with zero mean and standard deviation σ to each pixel of the input image, where σ was set to 10. The noisy pixel value $I_{noisy}(i, j)$ is given by:

$$I_{noisy}(i, j) = I(i, j) + N(i, j), \tag{1}$$

$N(i, j)$ is a random variable generated from a Gaussian distribution with zero mean and standard deviation σ.

Local Variance Gaussian Noise: Local Variance Gaussian noise is a variant of Gaussian noise that takes into account the local variance of the image. In this case, we added Gaussian noise with different standard deviations to different local regions of the input image to achieve more realistic noise patterns. Specifically, the standard deviation of Gaussian noise for each pixel was calculated based on the local variance of its neighbouring pixels. The noisy pixel value $I_{noisy}(i, j)$ is given by:

$$I_{noisy}(i, j) = I(i, j) + N_L(i, j), \tag{2}$$

where $N_L(i, j)$ is a random variable generated from a Gaussian distribution with zero mean and standard deviation $\sigma_L(i, j)$, which is calculated as:

$$\sigma_L(i, j) = k * \sigma_{local}(i, j), \tag{3}$$

where $\sigma_{local}(i, j)$ is the local variance of the image at pixel (i, j), and k is a scaling factor that determines the strength of the noise.

Poisson Noise: Poisson noise is a type of noise that arises from the random nature of photon arrival in digital images. It is modelled as a Poisson distribution with parameter λ. To synthesise Poisson noise, we first modelled the image as a Poisson process and then generated noisy pixels based on this model. Specifically, the noisy pixel value $I_{noisy}(i, j)$ is given by:

$$I_{noisy}(i, j) = \min(255, \max(0, \text{Poisson}(\lambda(i, j)) + I(i, j))), \tag{4}$$

where $I(i, j)$ is the original pixel value at position (i, j), $\lambda(i, j)$ is the mean value of the Poisson distribution, and $\text{Poisson}(\lambda(i, j))$ is a random variable generated from a Poisson distribution with mean $\lambda(i, j)$.

Speckle Noise: Speckle noise is a type of multiplicative noise that is commonly found in ultrasound and radar images. It is modelled as a multiplicative noise with a uniform distribution between 0 and 1. To synthesise speckle noise, we multiplied each pixel of the original image with a random value drawn from a uniform distribution between 0 and 1. Specifically, the noisy pixel value $I_{noisy}(i, j)$ is given by:

$$I_{noisy}(i, j) = I(i, j) * U(0, 1), \tag{5}$$

where $U(0, 1)$ is a random variable drawn from a uniform distribution between 0 and 1.

Salt-and-Pepper Noise: Salt-and-pepper noise is a type of impulse noise that occurs when some pixels in the image are replaced with the maximum or minimum pixel value. It is modelled as a random process that replaces a certain percentage of the pixels in the image with either the maximum or minimum pixel value. Specifically, the noisy pixel value $I_{noisy}(i, j)$ can be calculated as follows:

$$I_{noisy}(i, j) = I(i, j) + S(i, j) - P(i, j), \tag{6}$$

where $S(i, j)$ and $P(i, j)$ are random variables that model the presence of salt-and-pepper noise, respectively. They are defined as follows:

$$S(i, j) = I_{max} * \text{Bernoulli}(p_s), \tag{7}$$

$$P(i, j) = I_{min} * \text{Bernoulli}(p_p), \tag{8}$$

where $\text{Bernoulli}(p)$ is a random variable that takes the value 1 with probability p and the value 0 with probability $1 - p$.

[1] https://pillow.readthedocs.io/en/stable/

Note that $S(i,j)$ and $P(i,j)$ are only added to the pixel value $I(i,j)$ with the respective set probabilities p_s and p_p. Therefore, the total percentage of pixels affected by salt and pepper noise is $p_s + p_p$.

6.2. Downscale Corruption

The down-scale corruption contains down-scale interpolation, including Bicubic, Lanczos, Bilinear and Hamming. We use OpenCV-Python for the down-scaling process.

7. Additional Qualitative Results

The following figures show the denoising comparison on both synthetic noise removal (Figure S5 – Figure S14) and denoising real noise data (Figure S15 – Figure S22).

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A high-quality denoising dataset for smartphone cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[2] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[3] Joshua Batson and Loic Royer. Noise2self: Blind denoising by self-supervision. In *International Conference on Machine Learning*, pages 524–533. PMLR, 2019.

[4] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012.

[5] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *European Conference on Computer Vision*, pages 17–33. Springer, 2022.

[6] Wenchao Du, Hu Chen, and Hongyu Yang. Learning invariant representation for unsupervised image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14483–14492, 2020.

[7] Rich Franzen. Kodak lossless true color image suite. *source: http://r0k. us/graphics/kodak*, 4(2), 1999.

[8] Clément Godard, Kevin Matzen, and Matt Uyttendaele. Deep burst denoising. In *Proceedings of the European conference on computer vision*, pages 538–554, 2018.

[9] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 449–458, 2016.

[10] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189*, 2018.

[11] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[12] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of IEEE International Conference on Computer Vision*, volume 2, pages 416–423. IEEE, 2001.

[13] Seonghyeon Nam, Youngbae Hwang, Yasuyuki Matsushita, and Seon Joo Kim. A holistic approach to cross-channel image noise modeling and its application to image denoising. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1683–1691, 2016.

[14] Reyhaneh Neshatavar, Mohsen Yavartanoo, Sanghyun Son, and Kyoung Mu Lee. Cvf-sid: Cyclic multi-variate function for self-supervised image denoising by disentangling noise from image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17583–17591, 2022.

[15] Tongyao Pang, Huan Zheng, Yuhui Quan, and Hui Ji. Recorrupted-to-recorrupted: unsupervised deep learning for image denoising. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2043–2052, 2021.

[16] Yuhui Quan, Mingqin Chen, Tongyao Pang, and Hui Ji. Self2self with dropout: Learning self-supervised denoising from single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1890–1898, 2020.

[17] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[18] Longguang Wang, Yingqian Wang, Xiaoyu Dong, Qingyu Xu, Jungang Yang, Wei An, and Yulan Guo. Unsupervised degradation representation learning for blind super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10581–10590, 2021.

[19] Jun Xu, Yuan Huang, Ming-Ming Cheng, Li Liu, Fan Zhu, Zhou Xu, and Ling Shao. Noisy-as-clean: Learning self-supervised denoising from corrupted image. *IEEE Transactions on Image Processing*, 29:9316–9329, 2020.

[20] Jun Xu, Hui Li, Zhetong Liang, David Zhang, and Lei Zhang. Real-world noisy image denoising: A new benchmark. *arXiv preprint arXiv:1804.02603*, 2018.

[21] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5728–5739, 2022.

[22] Dan Zhang, Fangfang Zhou, Yuwen Jiang, and Zhengming Fu. Mm-bsn: Self-supervised image denoising for real-world with multi-mask based on blind-spot network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4188–4197, 2023.

Figure S5. Visual comparison of image denoising methods on Kodak [7] images with Gaussian ($\sigma = 25$) + local variance Gaussian noise.

[23] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017.

[24] Yi Zhang, Dasong Li, Ka Lung Law, Xiaogang Wang, Hongwei Qin, and Hongsheng Li. Idr: Self-supervised image denoising via iterative data refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2098–2107, 2022.

[25] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European conference on computer vision*, pages 286–301, 2018.

Figure S6. Visual comparison of image denoising methods on Kodak [7] images with Gaussian ($\sigma = 25$) + local variance Gaussian noise.

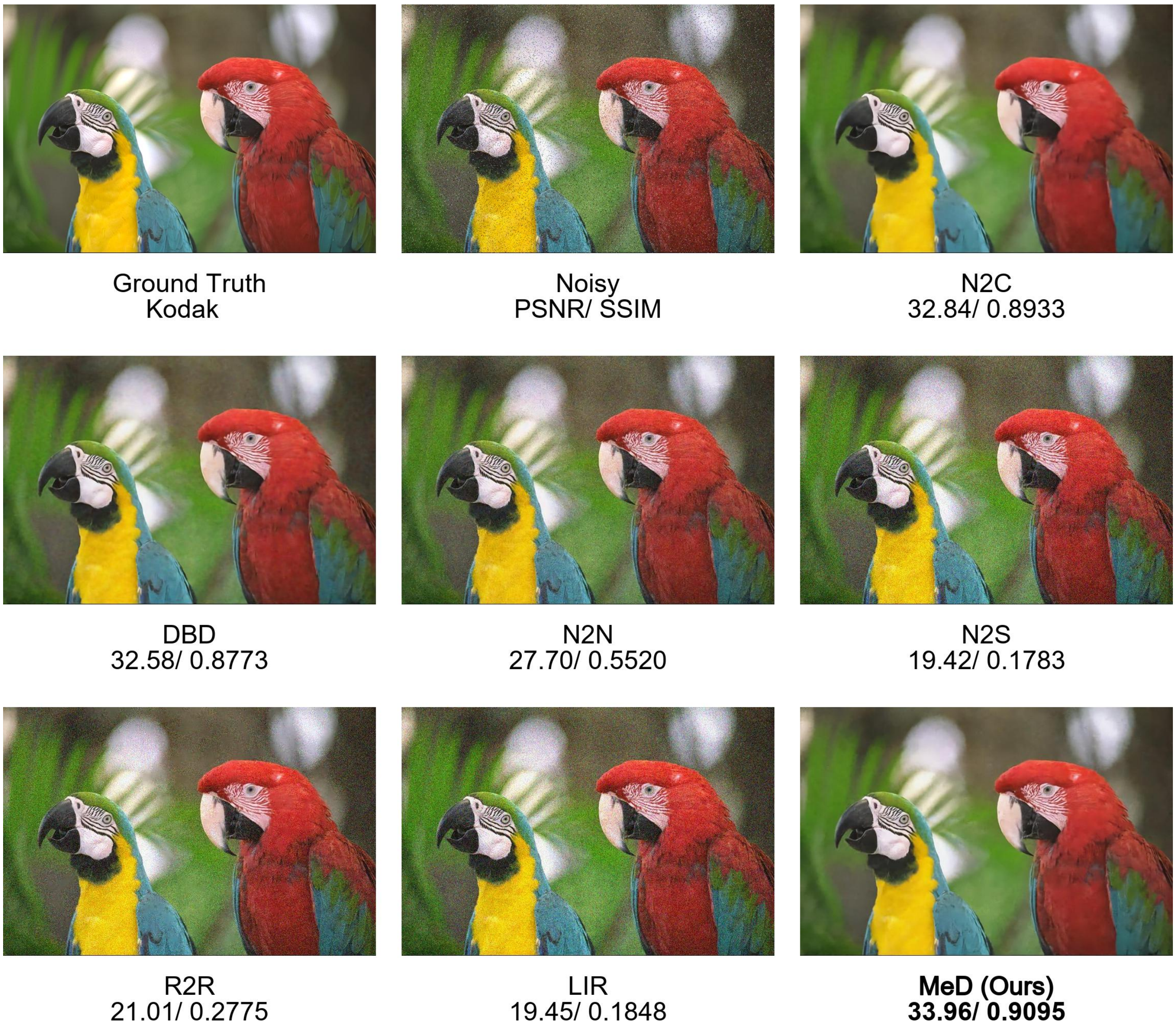


Figure S7. Visual comparison of image denoising methods on Kodak [7] images with Gaussian ($\sigma = 25$) + local variance Gaussian noise.

Figure S8. Visual comparison of image denoising methods on Kodak [7] images with Gaussian ($\sigma = 50$) + local variance Gaussian noise.

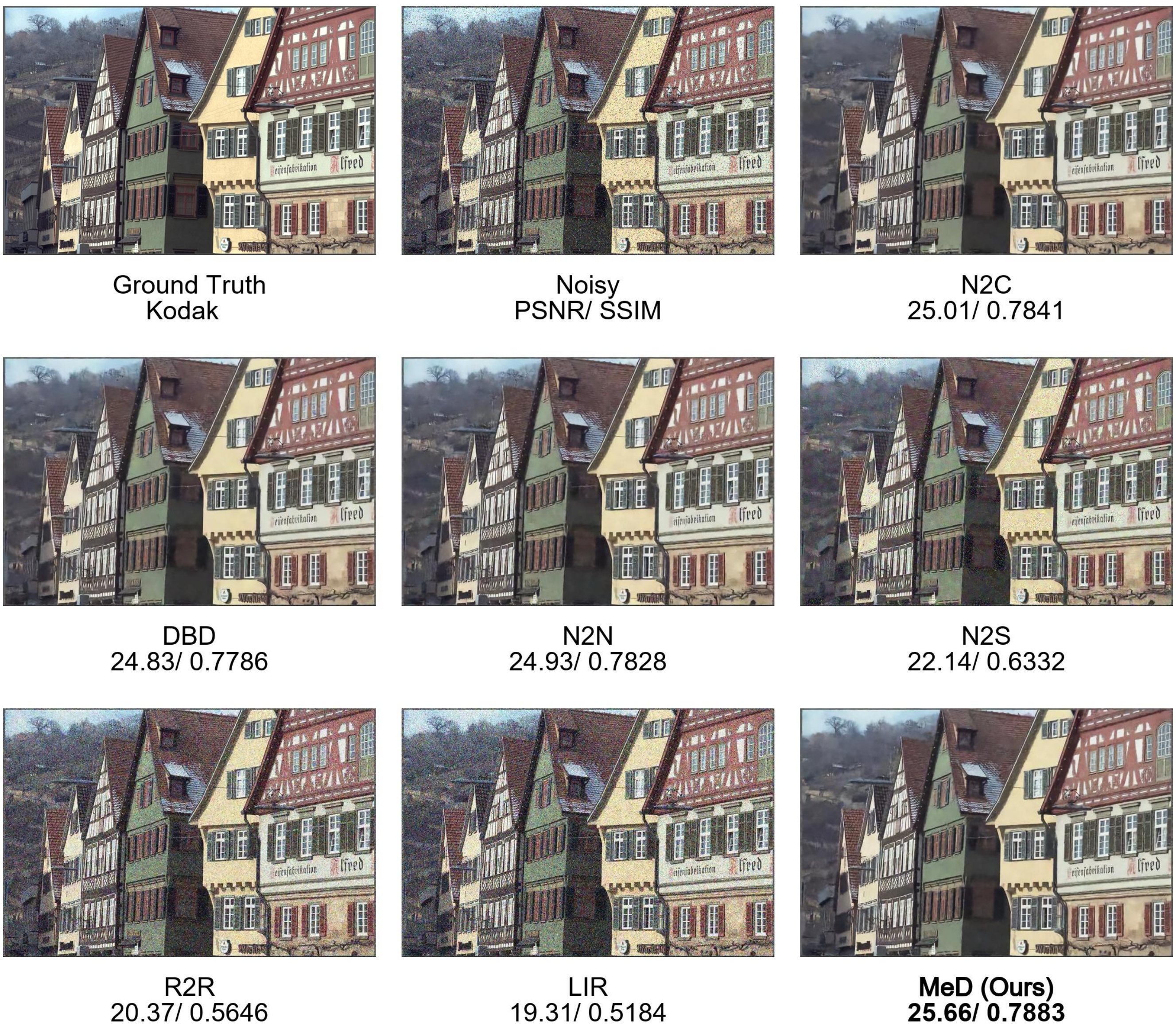


Figure S9. Visual comparison of image denoising methods on Kodak [7] images with Gaussian ($\sigma = 50$) + local variance Gaussian noise.

Figure S10. Visual comparison of image denoising methods on Kodak [7] images with Gaussian ($\sigma = 50$) + local variance Gaussian noise.

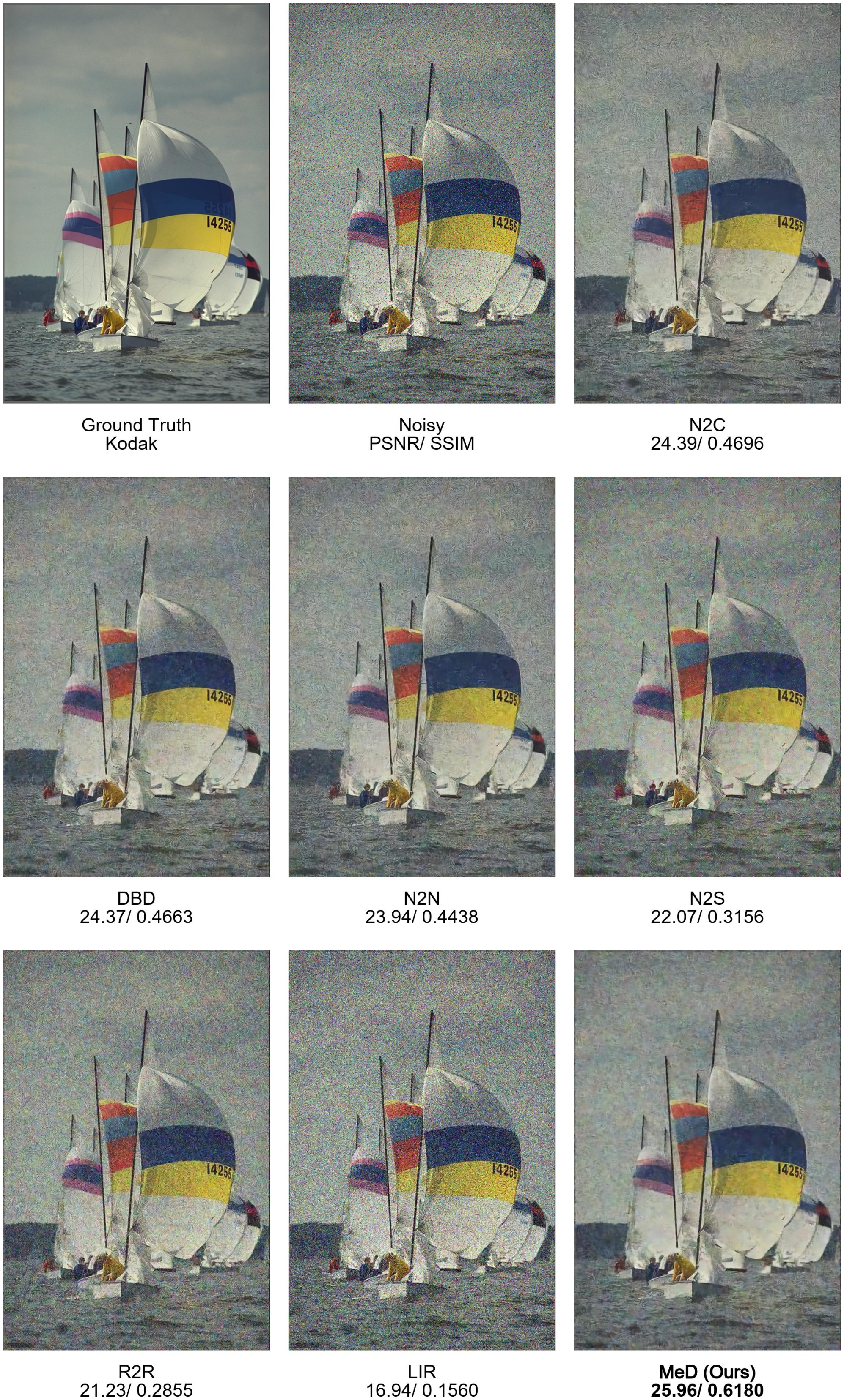


Figure S11. Visual comparison of image denoising methods on Kodak [7] images with Gaussian ($\sigma = 75$) + local variance Gaussian noise.

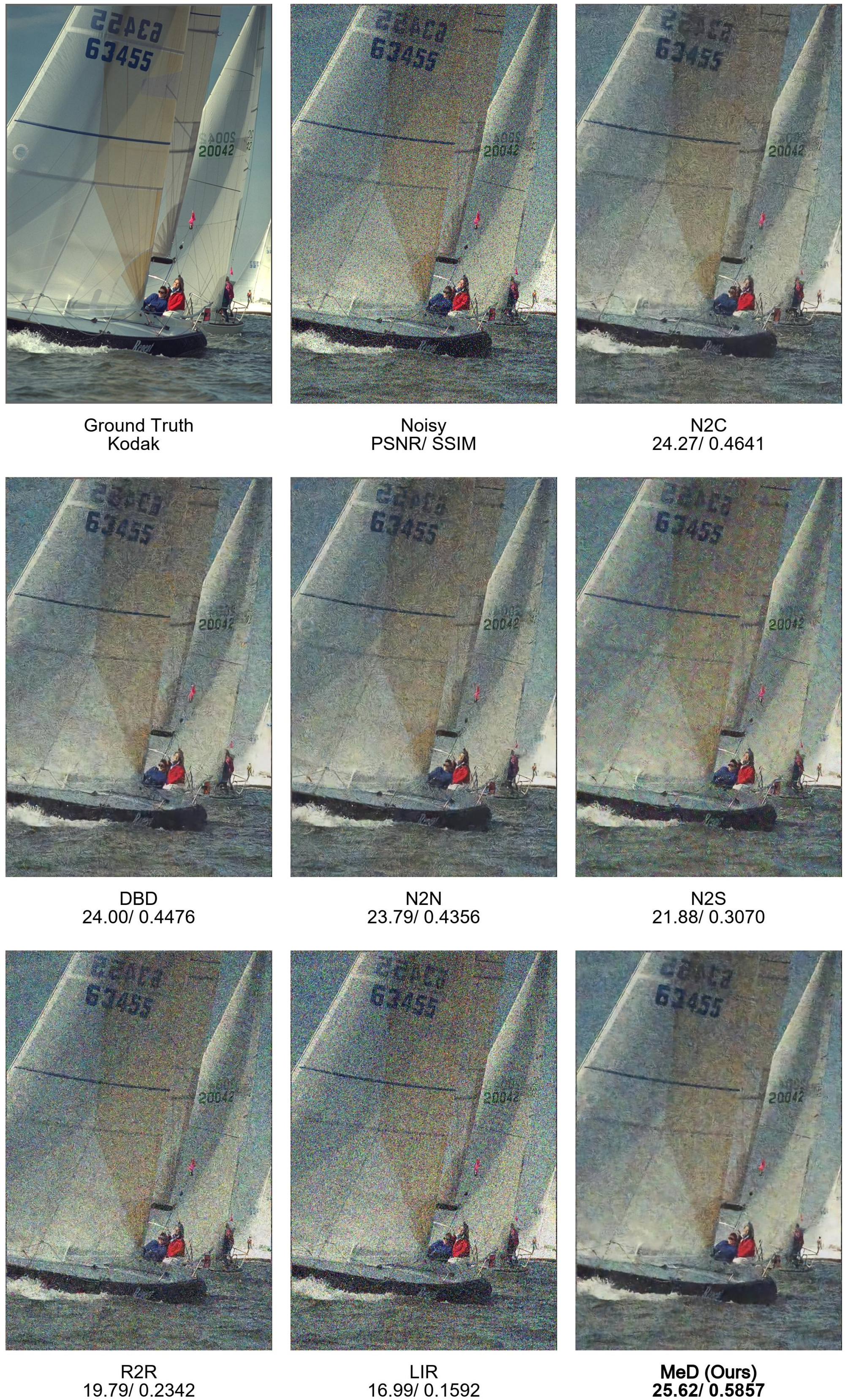


Figure S12. Visual comparison of image denoising methods on Kodak [7] images with Gaussian ($\sigma = 75$) + local variance Gaussian noise.

Figure S13. Visual comparison of image denoising methods on Kodak [7] images with Gaussian ($\sigma = 75$) + local variance Gaussian noise.

Figure S14. Visual comparison of image denoising methods on Kodak [7] images with local variance Gaussian + Poisson noise.

Figure S15. Visual comparison of image denoising methods on real noisy image dataset SIDD [1] example images with real noise.

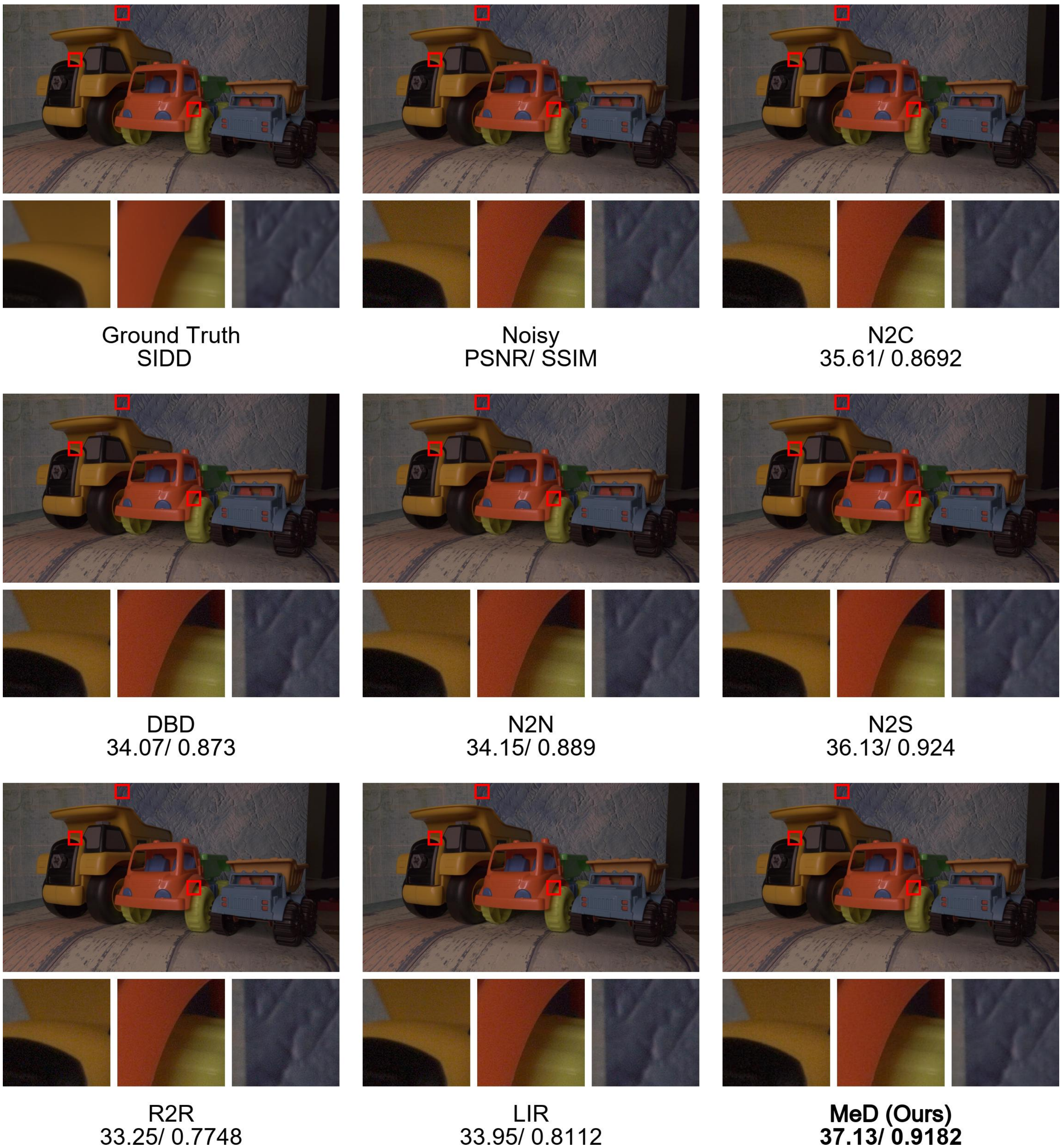


Figure S16. Visual comparison of image denoising methods on real noisy image dataset SIDD [1] example images with real noise.

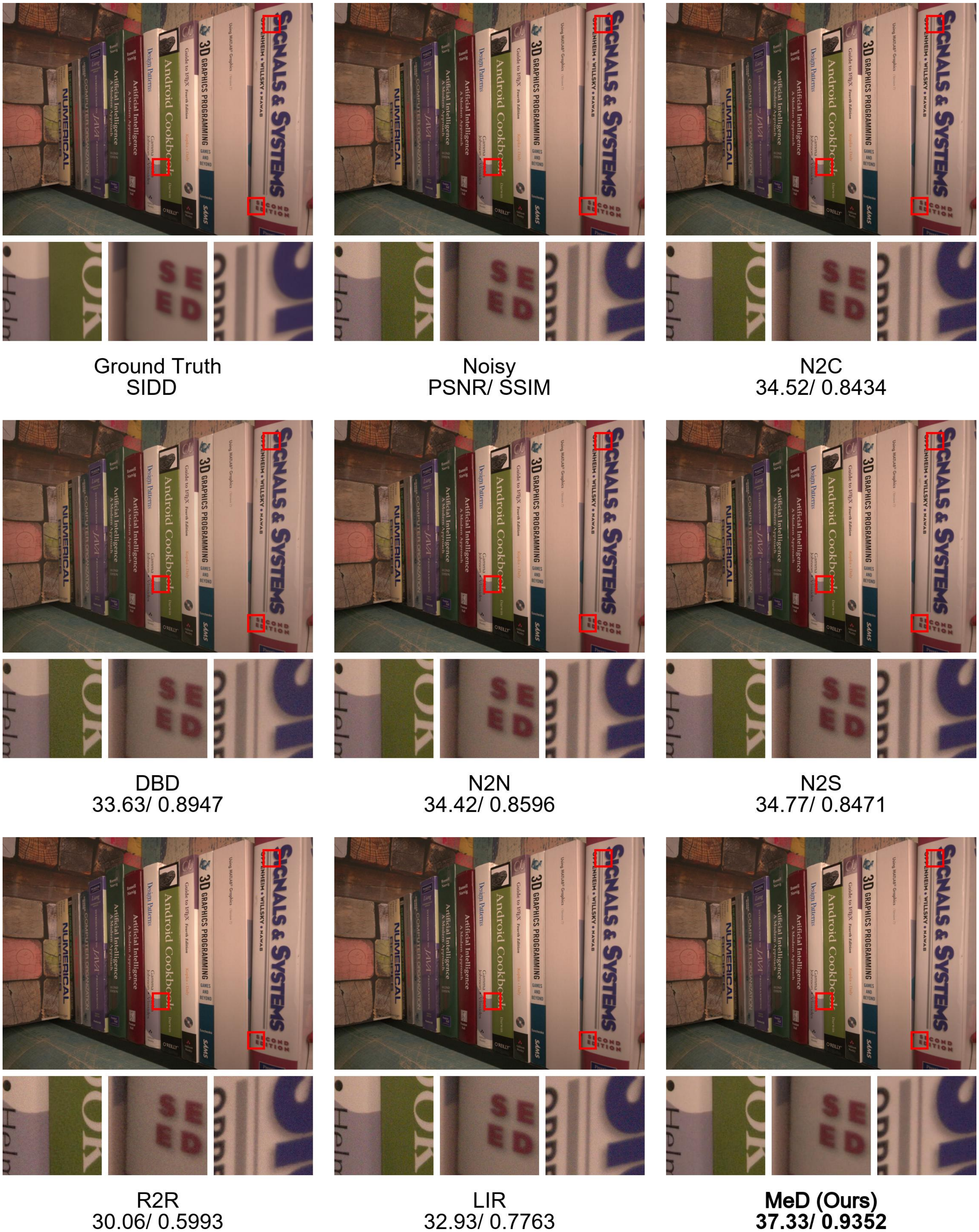


Figure S17. Visual comparison of image denoising methods on real noisy image dataset SIDD [1] example images with real noise.

Figure S18. Visual comparison of image denoising methods on real noisy image dataset SIDD [1] example images with real noise.

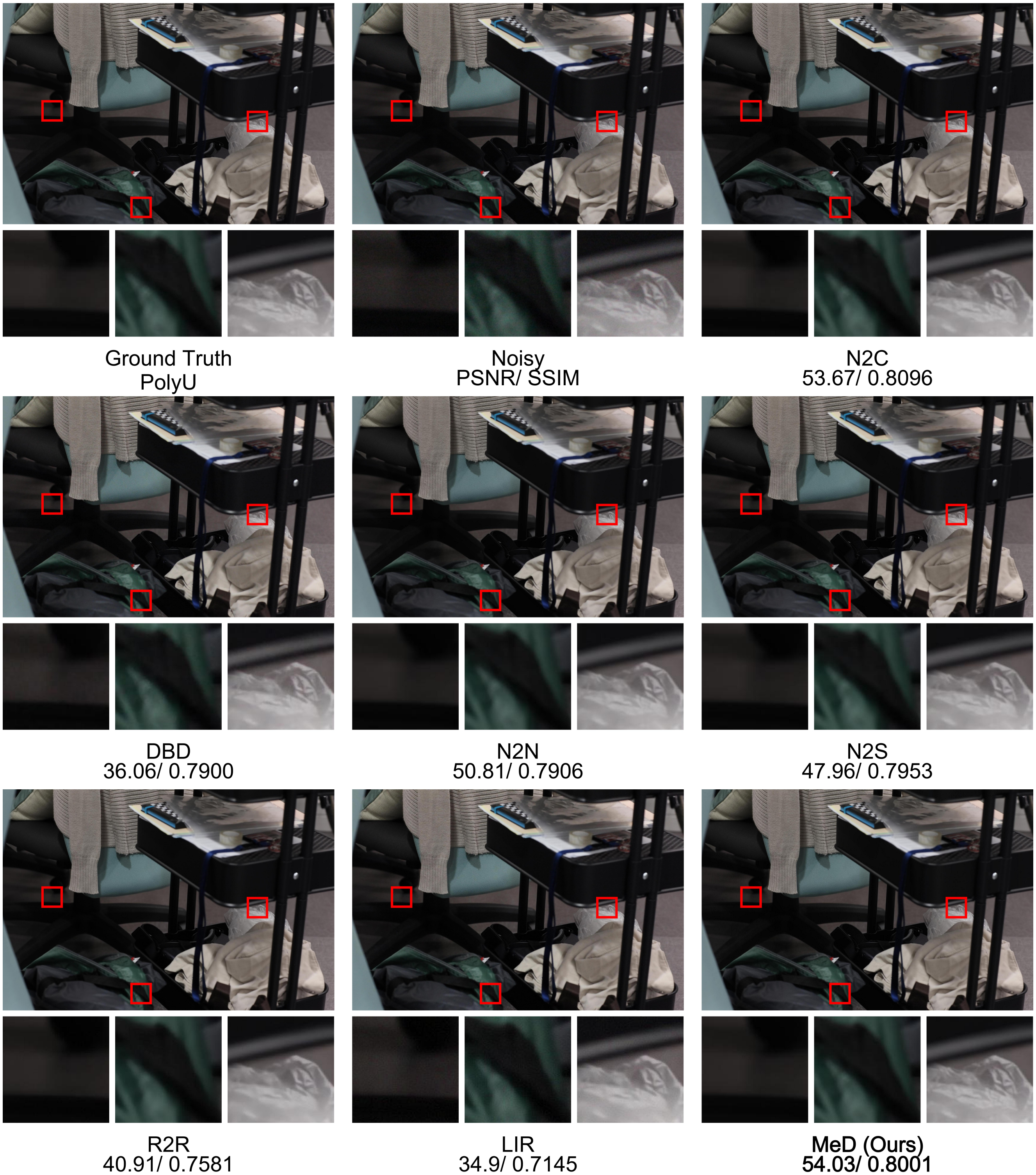


Figure S19. Visual comparison of image denoising methods on real noisy image dataset PolyU [20] example images with real noise.

Figure S20. Visual comparison of image denoising methods on real noisy image dataset PolyU [20] example images with real noise.

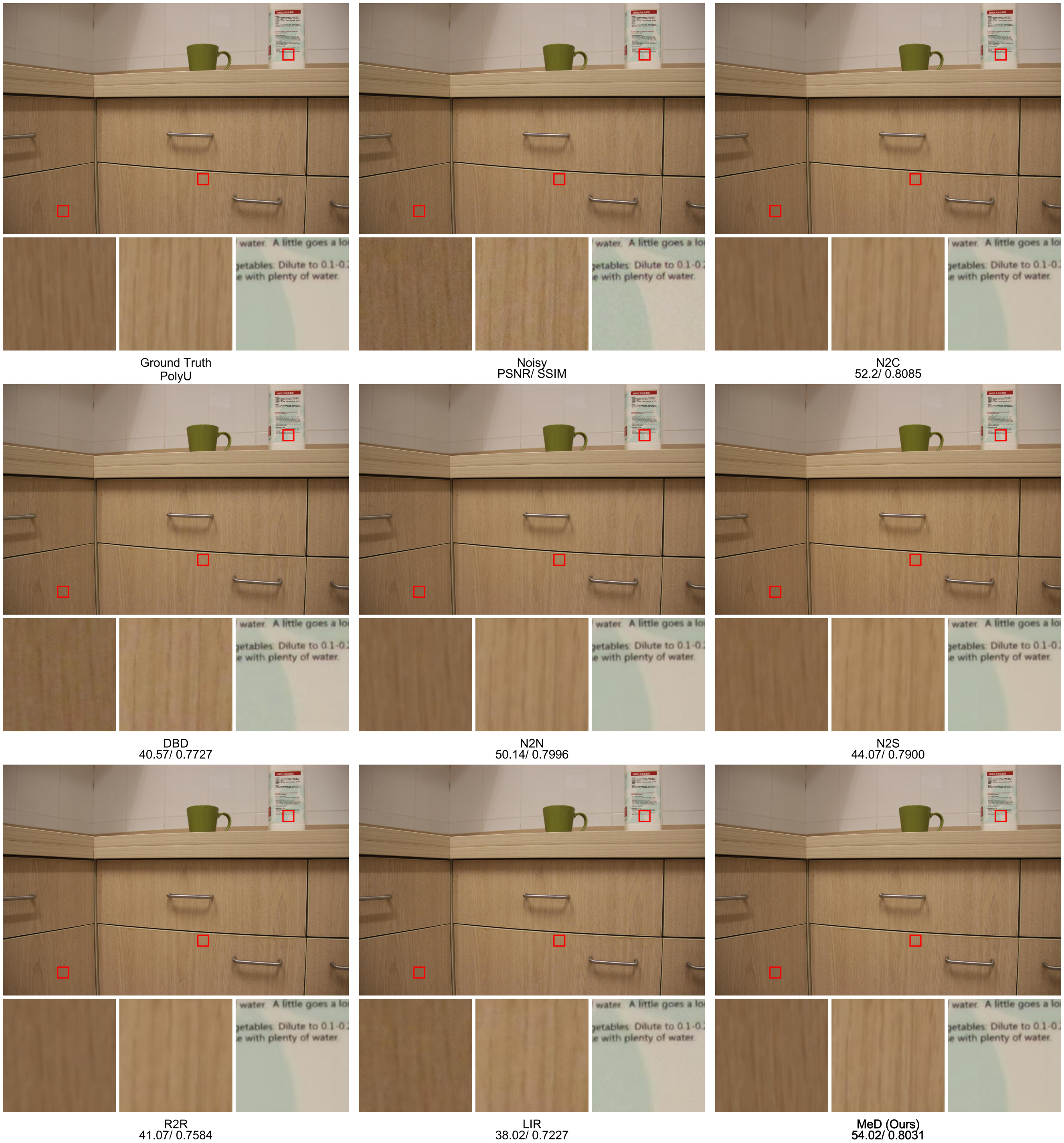


Figure S21. Visual comparison of image denoising methods on real noisy image dataset PolyU [20] example images with real noise.

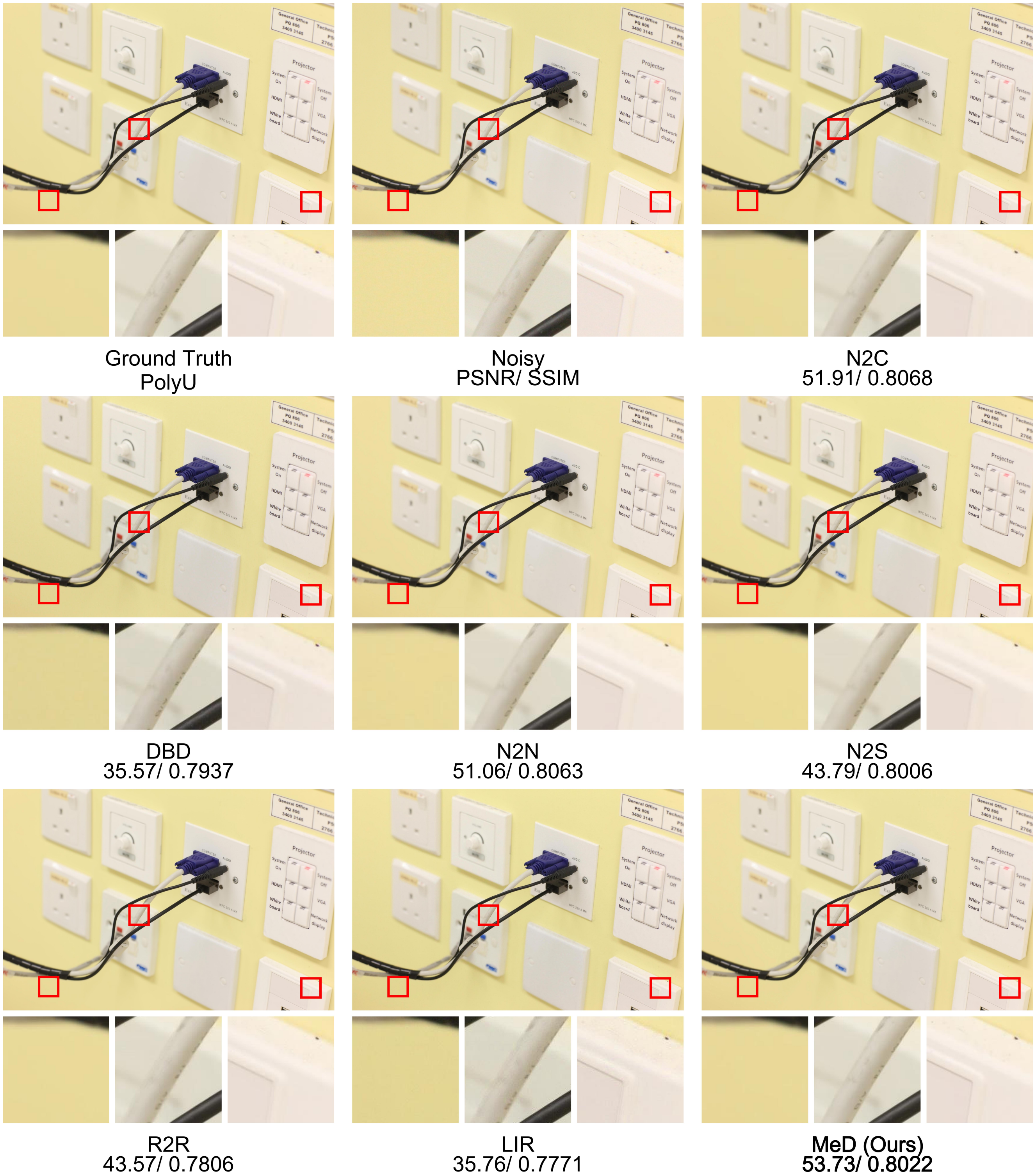


Figure S22. Visual comparison of image denoising methods on real noisy image dataset PolyU [20] example images with real noise.