

IDF: Iterative Dynamic Filtering Networks for Generalizable Image Denoising

Supplementary Material

A1. Appendix

A1.1. Details Regarding Test Noise

To rigorously evaluate OOD denoising robustness, we benchmark eight noise categories: (1) real-world noise captured by smartphone and DSLR cameras; (2) Monte Carlo (MC)-rendered noise $spp \in \{64, 128\}$; (3) additive Gaussian noise with $\sigma \in \{15, 25, 50\}$; (4) spatially correlated Gaussian noise with $\sigma \in \{45, 50, 55\}$; (5) Poisson noise with $\alpha \in \{2.5, 3.0, 3.5\}$; (6) speckle noise with $\sigma \in \{0.02, 0.03, 0.04\}$; (7) salt-and-pepper noise with $p \in \{0.012, 0.016, 0.02\}$; and (8) mixture noise at levels $\{1, 2, 3, 4\}$. Following MaskedDenoising [8], Gaussian and spatial Gaussian noise levels are rescaled to $[0, 255]$, whereas the remaining noise levels are normalized to $[0, 1]$.

Further implementation details are provided in the subsequent subsections.

Spatial Gaussian Noise. Spatial Gaussian noise differs from standard Gaussian noise in that its values are spatially correlated rather than independent across pixels. This correlation often arises from sensor imperfections or smoothing effects during image processing. Following MaskedDenoising [8], we synthesize spatial Gaussian noise by convolving *i.i.d.* Gaussian noise with standard deviation σ using a 3×3 averaging filter.

Poisson Noise. Poisson noise, often referred to as photon or shot noise, originates from the quantized nature of light. Its variance is equal to the mean signal level, making it inherently signal-dependent. This type of noise is particularly prevalent in low-light conditions, where the photon count is low. We synthesize Poisson noise as follows: $\mathbf{I}_{\text{Noisy}} = \mathbf{I}_{\text{Clean}} + \mathbf{n} \cdot \alpha$, where \mathbf{n} is sampled from a Poisson distribution and α controls the noise magnitude.

Salt-and-Pepper Noise. Salt-and-pepper noise is an impulsive corruption characterized by random occurrences of extreme pixel intensities (*i.e.* pure black or white). This artifact commonly stems from transmission errors or sensor defects. Following MaskedDenoising [8], we synthesize salt-and-pepper noise with MATLAB’s `imnoise` function.

Speckle Noise. Speckle noise is a multiplicative perturbation commonly observed in coherent imaging modalities such as synthetic aperture radar (SAR) and ultrasound; it originates from the interference of multiple scattered wavefronts and manifests as granular texture. Following MaskedDenoising [8], we synthesize speckle noise with MATLAB’s `imnoise` function.

Mixture Noise. The mixture noise model combines several noise sources, including Gaussian, Poisson, speckle, and salt-and-pepper noise, to emulate real-world noise characteristics.

Table A1. Comparison of denoising performance based on model capacity. The number of parameters for DnCNN and Restormer are set to be similar to those of our method. The best results are highlighted in **bold**.

Methods	Params. (M)	Gaussian	Spatial Gaussian	SIDD
		PSNR↑/SSIM↑	PSNR↑/SSIM↑	PSNR↑/SSIM↑
DnCNN	0.04	19.74/0.4262	27.24/0.7966	28.93/0.6040
Restormer	0.04	20.11/0.4389	26.39/0.7572	27.39/0.5388
Ours	0.04	25.06/0.7547	27.78/0.8333	32.08/0.7578

Following MaskedDenoising [8], we synthesize this mixture by sequentially adding Gaussian noise with variance σ_g^2 , speckle noise with variance σ_{s1}^2 , Poisson noise scaled by α , salt-and-pepper noise with density d , and a second speckle component with variance σ_{s2}^2 .

We categorize the mixture noise into four levels, determined by their overall intensity and complexity:

Level 1: $\sigma_g^2 = 0.003$, $\sigma_{s1}^2 = 0.003$, $\alpha = 1$, $d = 0.002$, $\sigma_{s2}^2 = 0.003$,

Level 2: $\sigma_g^2 = 0.004$, $\sigma_{s1}^2 = 0.004$, $\alpha = 1$, $d = 0.002$, $\sigma_{s2}^2 = 0.003$,

Level 3: $\sigma_g^2 = 0.006$, $\sigma_{s1}^2 = 0.006$, $\alpha = 1$, $d = 0.003$, $\sigma_{s2}^2 = 0.006$,

Level 4: $\sigma_g^2 = 0.008$, $\sigma_{s1}^2 = 0.008$, $\alpha = 1$, $d = 0.004$, $\sigma_{s2}^2 = 0.008$.

Note that, for each level, the individual noise components are introduced in the prescribed sequence.

Monte Carlo-Rendered Image Noise. Noise produced by the stochastic Monte Carlo integration underlying physically based rendering results in sampling artifacts that reveal the finite number of rays used to approximate light transport. We evaluate the proposed method on the Monte Carlo noise benchmark from [15].

Real-World Noise. Real-world camera images contain complex noise arising from photon statistics, sensor readout, and in-camera ISP post-processing. To evaluate the robustness of IDF in these conditions, we evaluate it on diverse datasets captured with both smartphone and DSLR devices [1, 3, 54, 72].

A1.2. Additional Ablation Studies

Impact of Model Capacity on Generalization. To investigate whether reducing the number of model parameters alone can mitigate overfitting and enhance generalization, we use two conventional denoisers, DnCNN [82] and Restormer [79], with capacities adjusted to match our model. These variants are evaluated on Urban100 [25] corrupted with Gaussian noise ($\sigma = 50$) and spatial Gaussian noise

Table A2. OOD denoising results on various training noise types. The best results are highlighted in **bold**.

Training Noise Type	Model	Real-World (SIDD)	Spatial Gaussian ($\sigma = 55$)	Medical Imaging (LDCT)	Average
		PSNR↑/SSIM↑	PSNR↑/SSIM↑	PSNR↑/SSIM↑	PSNR↑/SSIM↑
Gauss. $\sigma = 15$ (Baseline)	CGNet	27.32/0.5359	25.41/0.6651	38.50/0.7878	30.41/0.6629
	MaskedDenoising	28.65/0.6043	26.72/0.7685	37.76/0.7880	31.04/0.7203
	Ours	32.08/0.7578	27.87/0.8049	44.45/0.9643	34.80/0.8423
(i) Gauss. $\sigma \sim \mathcal{U}(15, 50)$	CGNet	25.36/0.4298	25.09/0.6492	43.69/0.9254	31.38/0.6681
	MaskedDenoising	27.93/0.7320	25.07/0.7438	27.96/0.6870	26.99/0.7209
	Ours	32.39/0.7786	28.08/0.8056	44.88/0.9684	35.12/0.8509
(ii) Gauss. $\sigma \sim \mathcal{U}(15, 50)$ + Poisson $\alpha \sim \mathcal{U}(1, 4)$	CGNet	25.84/0.4440	25.48/0.6683	41.46/0.8737	30.93/0.6620
	MaskedDenoising	27.99/0.7433	25.03/0.7427	24.52/0.5447	25.85/0.6769
	Ours	32.95/0.8197	27.95/0.8004	44.60/0.9632	35.17/0.8611
(iii) Monte Carlo 16 spp	CGNet	27.31/0.5227	23.85/0.5861	44.70/0.9691	31.95/0.6926
	MaskedDenoising	31.45/0.7359	25.24/0.6657	44.49/0.9674	33.73/0.7897
	Ours	34.13/0.8414	26.10/0.7170	44.51/0.9625	34.91/0.8403

($\sigma = 55$), as well as on the SIDD real-world dataset. As reported in Table A1, simply reducing network size does not meaningfully alleviate overfitting because noise characteristics vary substantially across types and levels. In contrast, our framework achieves the best performance across all noise settings while remaining the most compact, thanks to its modules that dynamically adapt to OOD noise.

Impact of Diverse Training Noise Conditions on Generalization. We perform ablation studies to evaluate how well our method generalizes under varying training noise conditions. Table A2 summarizes the performance of IDF, MaskedDenoising [8], and the recent SOTA model CGNet [17], each trained under three distinct noise regimes (i–iii). All models are evaluated on challenging OOD noise scenarios: real-world sensor noise from SIDD [1], spatially correlated Gaussian noise on CBSD68 [61], and low-dose CT (LDCT) scans [50]. The LDCT benchmark, whose intensity distribution differs markedly from natural sRGB images, serves as an extreme test of OOD robustness. Because LDCT volumes are single-channel, each slice is replicated across three RGB channels to avoid architectural changes. Despite CGNet’s 119M parameters and MaskedDenoising’s mixed-noise curriculum, both methods suffer substantial performance drops in these settings. In contrast, IDF consistently preserves high fidelity across all benchmarks, and its accuracy improves as the training-noise configuration becomes more heterogeneous, highlighting its noise-invariant filtering capability. Please note that, for the LDCT evaluation, we constrain IDF to a single denoising iteration to avoid over-smoothing while maintaining computational efficiency.

Impact of Kernel Size. Table A3 analyzes how the patch-convolution kernel size K (see Equation 1) affects denoising performance on mixture and spatial Gaussian noise. A kernel of $K = 3$ shows the highest PSNR and SSIM on both noise types. Enlarging the kernel to $K = 5$ or $K = 7$ slightly degrades performance, indicating that small receptive fields capture essential local details without introducing excessive smoothing. These results show that a 3×3 kernel offers the best balance between detail preservation and noise removal within our framework.

Table A3. Comparison results on denoising performance depending on different kernel size K in Equation 1. The best results are highlighted in **bold**.

Kernel Size (K)	Mixture	Spatial Gaussian
	PSNR↑/SSIM↑	PSNR↑/SSIM↑
3	27.52/0.8405	27.78/0.8333
5	27.27/0.8366	27.58/0.8259
7	27.30/0.8356	27.41/0.8143

Table A4. Comparison results on denoising performance depending on different power normalization factor p in Equation 8. The best results are highlighted in **bold**.

Power Norm. (p)	Mixture	Spatial Gaussian
	PSNR↑/SSIM↑	PSNR↑/SSIM↑
1	27.01/0.8289	27.34/0.8231
2	27.24/0.8325	27.65/0.8293
3	27.52/0.8405	27.78/0.8333
4	27.50/0.8398	27.55/0.8199

Impact of the Power Normalization Factor. We perform an ablation study to investigate the impact of the power normalization factor p (see Equation 8) on denoising performance. The results are presented in Table A4 for two noise types: mixture noise and spatial Gaussian noise. Four different normalization factors, ranging from 1 to 4, are evaluated.

For mixture noise, performance consistently improves as the normalization factor increases from $p = 1$ to $p = 3$, with the highest PSNR observed at $p = 3$. A similar trend is noted for spatial Gaussian noise, where $p = 3$ also shows the best results among the evaluated settings. Increasing the normalization factor to $p = 4$ does not lead to further improvements; instead, it appears to reduce kernel diversity, which negatively impacts performance. These results indicate that a power normalization factor of $p = 3$ provides the most effective denoising performance across both noise types.

Impact of DIC threshold. Table A5 presents an ablation study on the influence of the DIC threshold κ (as defined

Table A5. Comparison results on denoising performance depending on different DIC threshold κ in Equation 9. We choose $\kappa = 0.015$ for main results and corresponding scores are highlighted in **bold**.

Threshold (κ)	Spatial Gaussian							
	$\sigma = 45$		$\sigma = 50$		$\sigma = 55$		Average	
	PSNR \uparrow /SSIM \uparrow	# Iterations	PSNR \uparrow /SSIM \uparrow	# Iterations	PSNR \uparrow /SSIM \uparrow	# Iterations	PSNR \uparrow /SSIM \uparrow	# Iterations
0.005	28.89/0.8557	8.02	28.25/0.8432	8.22	27.70/0.8298	8.58	28.28/0.8429	8.27
0.01	28.80/0.8524	7.40	28.15/0.8383	7.54	27.65/0.8275	8.04	28.20/0.8394	7.66
0.015	28.77/0.8511	7.10	28.12/0.8364	7.22	27.54/0.8217	7.44	28.14/0.8364	7.25
0.02	28.73/0.8497	6.60	28.10/0.8340	6.72	27.49/0.8182	6.86	28.11/0.8340	6.73
0.025	28.69/0.8472	6.00	28.06/0.8324	6.32	27.43/0.8154	6.50	28.06/0.8317	6.27
0.03	28.65/0.8455	5.66	27.99/0.8290	5.78	27.42/0.8152	6.34	28.02/0.8299	5.93

Table A6. Comparison results on denoising performance depending on the different number of training full iterations T in Figure 2. The best results are highlighted in **bold**.

# Iterations (T)	Mixture	Spatial Gaussian
	PSNR \uparrow /SSIM \uparrow	PSNR \uparrow /SSIM \uparrow
6	27.32/0.8326	27.48/0.8162
8	27.37/0.8390	27.67/0.8287
10	27.52/0.8405	27.78/0.8333
12	27.42/0.8389	27.86/0.8383

in Equation 9) on denoising performance under spatial Gaussian noise at varying noise levels ($\sigma = \{45, 50, 55\}$). The results include PSNR, SSIM, and the average number of iterations before termination.

As κ increases from 0.005 to 0.03, the average number of iterations decreases from approximately 8.27 to 5.93. This pattern is consistent across all noise levels, suggesting that a looser threshold prompts earlier termination of the iterative process, thereby reducing computational demands.

The findings reveal a clear trade-off: lower thresholds yield more iterations and slightly better denoising performance, whereas higher thresholds reduce computational cost at the expense of minor performance degradation. Based on this observation, we adopt $\kappa = 0.015$ as the default setting, as it offers a balanced compromise between performance and efficiency.

The selection of κ within our DIC mechanism directly affects both computational efficiency and denoising quality. For scenarios where computational constraints are important, a higher threshold may be preferable despite a modest performance loss. In contrast, for applications prioritizing maximum denoising fidelity, a lower κ value that allows additional iterations may be more suitable. This flexibility allows the framework to adapt to diverse practical requirements.

Impact of the Total Number of DID Block Full Iteration.

Table A6 presents an ablation study on the impact of the number of full training iterations (T) on denoising performance for two noise types: mixture and spatial Gaussian. As

shown in the table, varying the number of iterations leads to differences in PSNR and SSIM.

Denoising performance progressively improves as T increases from 6 to 10, reaching its peak at $T = 10$. A further increase to $T = 12$ results in a slight performance decline, indicating that excessive iterations may cause over-smoothing. Considering both noise types and the trade-off between performance and computational cost, we select $T = 10$ as the default setting. This choice strikes a balance between effective noise removal and detail preservation, while also highlighting the advantages of our DIC strategy in adapting the iterative denoising process based on image content and noise characteristics. These findings demonstrate the importance of selecting an appropriate T to achieve optimal performance across diverse noise scenarios without incurring unnecessary computational overhead.

Further Analysis on Inference Speed. In addition to the results presented in Table 4 of the main paper, we further evaluate the effectiveness of the proposed DIC in terms of computational efficiency. Specifically, we measure inference speed using high-resolution 4K images to evaluate whether IDF can be efficiently deployed in real-world scenarios.

While the runtime difference between the kernel-based DIC variant and the full-iteration version is minimal for 160×160 images, the benefit of DIC becomes significantly more evident at higher resolutions. On a single 4K frame, the DIC variant achieves a substantial 30% speed-up (0.383s vs. 0.548s).

These results highlight that IDF offers strong performance in both OOD denoising and inference efficiency, making it well-suited for practical deployment.

A1.3. DIC Algorithm

The overall inference algorithm of IDF is outlined in Algorithm A1. The noisy input image is iteratively denoised for up to T iterations. At each iteration, the dilation rate within the DID block is alternated to balance global and local context. Specifically, for odd-numbered iterations, the dilation rate is set to two to capture broader contextual regions. In contrast, during even-numbered iterations, the rate is reduced

Algorithm A1: Dynamic Iteration Control (DIC)**Require** : Input noisy image $\mathbf{I}_{\text{Noisy}}$, max iteration T

```

1  $\mathbf{I}_{\text{Clean}}^{(0)} \leftarrow \mathbf{I}_{\text{Noisy}}$ 
2 for  $t \leftarrow 1$  to  $T$  do
3    $\mathbf{y}^{(t)} \leftarrow \mathbf{I}_{\text{Clean}}^{(t-1)}$ 
4   if  $t = 1$  then
5      $\mathbf{I}_{\text{Res}}^{(t)} \leftarrow 0$ 
6   else
7      $\mathbf{I}_{\text{Res}}^{(t)} \leftarrow \hat{\mathbf{I}}_{\text{Clean}}^{(t-1)} - \hat{\mathbf{I}}_{\text{Clean}}^{(t-2)}$ 
8   end
9   dilation  $\leftarrow (t \bmod 2 = 1)$ 
10   $\mathbf{I}_{\text{Clean}}^{(t)} \leftarrow \text{DID-Block}(\mathbf{y}^{(t)}, \mathbf{I}_{\text{Res}}^{(t)}, \text{dilation})$ 
11  if criterion in Equation 9 is met then
12     $T \leftarrow t$ 
13  end
14 end
15 return Denoised image  $\hat{\mathbf{I}}_{\text{Clean}}^{(T)}$ 

```

Table A7. Comparison results on denoising performance depending on DIC strategies with equivalent average iterations. The results of Kernel-DIC are highlighted in **bold**.

Methods	Gaussian	Spatial Gaussian	SIDD
	PSNR \uparrow /SSIM \uparrow	PSNR \uparrow /SSIM \uparrow	PSNR \uparrow /SSIM \uparrow
No-DIC	31.89/0.9029	27.09/0.7949	29.42/0.6384
Image-DIC	31.75/0.9008	27.14/0.7976	29.85/0.6603
Kernel-DIC	31.79/0.9010	27.22/0.8025	30.09/0.6739

to one, enabling a focus on local details. If DIC is enabled and the stopping criterion is satisfied (see Equation 9), the iterative denoising process is terminated early.

A1.4. Additional Analysis on DIC

To accelerate inference efficiency, we propose the Dynamic Iteration Control (DIC) mechanism, which adaptively determines the number of denoising iterations based on image content and noise characteristics. To comprehensively evaluate DIC, we compare two variants in the following subsections. One variant, referred to as Image-DIC, utilizes the residual of the denoised images, while the other variant, termed Kernel-DIC, utilizes the residual of the predicted kernels, as described in Equation 9.

Comparison of Results with Equivalent Average Iterations. Table A7 compares the denoising performance of three strategies: No-DIC, Image-DIC, and Kernel-DIC, while maintaining an equivalent average number of iterations by manually adjusting the threshold κ in Equation 9 for each test dataset. On the Gaussian noise dataset with $\sigma = 15$, which represents an in-distribution scenario, No-DIC achieves the highest PSNR and SSIM. However, the differences compared to the DIC-based variants are marginal.

For the more challenging spatial Gaussian noise with

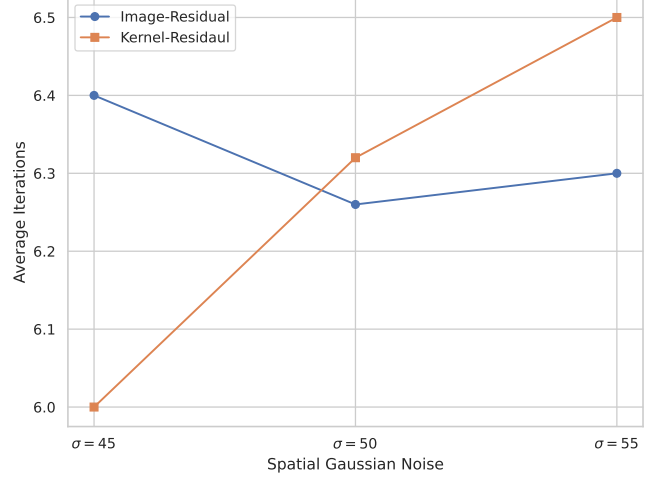


Figure A1. Comparison of results on averaged adaptive iterations with different levels of noise using image residual-based DIC (Image-DIC) and the kernel residual-based DIC (Kernel-DIC) approaches.

$\sigma = 55$, both DIC strategies lead to performance improvements. Image-DIC provides moderate gains, while Kernel-DIC offers further enhancement and outperforms No-DIC. It is worth noting that Urban100 is used for both synthetic noise settings. A similar trend is observed on the real-world SIDD dataset, where the use of DIC results in notable performance gains, with Kernel-DIC achieving the best results among the three.

These findings suggest that adaptive iteration strategies perform comparably to fixed-iteration methods under simple noise conditions. In contrast, under complex noise distributions, they provide clear benefits, with the kernel-residual-based DIC demonstrating the most consistent and robust improvements.

Comparison of Image and Kernel-based DIC Results with Different Levels of Noise. In Figure A1, both Image-DIC and Kernel-DIC methods are evaluated under spatial Gaussian noise at varying magnitudes (*e.g.*, $\sigma = \{45, 50, 55\}$) to determine which residual feature, image or kernel, better reflects noise characteristics. As the noise level increases from 45 to 55, the average number of iterations required by the kernel-DIC (Kernel-Residual) increases linearly. In contrast, the iterations for the image-DIC (Image-Residual) remain largely unchanged across noise levels. These findings indicate that kernel-DIC is more responsive to noise magnitude, allowing for adaptive inference and improved computational efficiency in noise-dependent scenarios.

A1.5. Visualization of Predicted Kernels

To better understand how our model adapts to different noise characteristics, in Figure A2, we visualize the predicted kernel maps under a variety of noise types, including both

synthetic and real-world degradations. For each example, we select representative pixel locations (highlighted with red dots) and display their corresponding denoising kernels. The predicted kernels vary in shape depending on both the spatial structure and the underlying noise distribution. Notably, in flat or homogeneous regions, the kernels exhibit near-uniform weights, enabling effective averaging to suppress noise. In contrast, in textured or edge regions, the kernels become more anisotropic, preserving local structures. Importantly, all predicted kernels are normalized to sum to one, effectively functioning as content-adaptive averaging filters. This regularization improves stability and prevents over-amplification of noise, especially under OOD settings. These visualizations highlight how our model generalizes across noise domains by dynamically modulating its receptive behavior based on local context.

A1.6. Additional Analysis on Iterative Method

Visualization of the Iterative Refinement. To effectively demonstrate how denoising kernels and denoised images evolve over iterations, we present results from all ten iterations ($T = 10$). Specifically, the denoised images are shown in the upper row, while the corresponding averaged denoising kernels are displayed in the lower row for clarity. Gaussian and spatial Gaussian noise at levels $\sigma = 50$ and $\sigma = 55$, respectively, are applied to the CBSD68 [61] and Urban100 [25] datasets for synthetic noise removal. Additionally, the SIDD [1] and SIDD+ [3] datasets are used for evaluating real-world noise removal.

As illustrated in Figure A3 and Figure A4, even when the type and level of noise degradation remain the same, the denoising kernels differ between datasets. This variation reflects the model’s ability to adapt to specific image characteristics such as textures, and highlights its capacity for input-dependent filter generation. Moreover, the model exhibits sensitivity to different noise types, generating distinct kernels based on noise characteristics, even when the image content is unchanged.

Similarly, under real-world noise conditions (see Figure A5), the proposed framework dynamically adjusts to the unique properties of the input signal, demonstrating strong generalization capabilities in OOD scenarios. Notably, regardless of signal content or noise characteristics, the denoising kernels show progressive convergence across iterations, which reflects the stability and robustness of the iterative denoising process.

These observations confirm that the model is highly robust in OOD denoising tasks, consistently producing kernels that flexibly adapt to diverse image content, noise types, and intensity levels.

A1.7. Additional Denoising Results

Qualitative Comparison. We provide additional visual comparisons with other benchmark models for both synthetic and real-world noise removal in Figure A6 and Figure A7, respectively. For reference, the original clean images and their corresponding cropped regions of interest (ROI) are presented in Figure A8.

Quantitative Comparison. We further evaluate the generalization capability of our method by comparing its denoising performance with several benchmark models, including DnCNN [82], SwinIR [44], Restormer [79], CODE [88], and MaskedDenoising [8], across various synthetic noise types and levels. Results from CLIPDenoising [11] are also included for reference. A detailed comparison is provided in Table A8.

Table A8. Quantitative results of denoising performance on CBSD68, McMaster, Kodak24 and Urban100 with regard to varied synthetic OOD noises in terms of PSNR \uparrow and SSIM \uparrow . All methods are trained with Gaussian noise with a level of $\sigma = 15$. The symbol \dagger denotes that our model utilizes the proposed DIC during inference. The best and second-best results are highlighted in **bold** and underline.

Noise Types	Datasets	ClipDenoising	DnCNN	SwinIR	Restormer	CODE	MaskedDenoising	Ours \dagger	Ours
		PSNR \uparrow /SSIM \uparrow	PSNR \uparrow /SSIM \uparrow	PSNR \uparrow /SSIM \uparrow	PSNR \uparrow /SSIM \uparrow	PSNR \uparrow /SSIM \uparrow	PSNR \uparrow /SSIM \uparrow	PSNR \uparrow /SSIM \uparrow	PSNR \uparrow /SSIM \uparrow
Gaussian $\sigma = 15$	CBSD68	33.61/0.9273	33.64/0.9271	<u>34.00/0.9319</u>	<u>33.99/0.9319</u>	34.10/0.9339	30.78/0.8891	<u>32.24/0.8939</u>	32.18/0.8915
	McMaster	33.49/0.8932	34.30/0.9036	<u>34.95/0.9117</u>	34.83/0.9093	35.11/0.9276	30.90/0.8502	32.52/0.8776	32.11/0.8680
	Kodak24	34.27/0.9193	34.42/0.9209	34.99/0.9285	<u>34.98/0.9287</u>	34.95/0.9285	31.41/0.8833	32.93/0.8867	32.87/0.8839
	Urban100	32.77/0.9179	33.64/0.9264	<u>34.49/0.9338</u>	34.45/0.9338	34.59/0.9492	29.32/0.8995	31.52/0.8947	31.42/0.8929
Gaussian $\sigma = 25$	CBSD68	30.51/0.8718	24.89/0.6010	24.29/0.5639	27.51/0.7151	24.58/0.5631	28.20/0.8202	<u>29.64/0.8376</u>	29.80/0.8435
	McMaster	30.62/0.8319	25.63/0.5818	25.03/0.5468	28.07/0.6910	25.21/0.5395	28.99/0.7971	30.16/0.8218	<u>30.10/0.8195</u>
	Kodak24	31.40/0.8681	24.83/0.5388	24.26/0.5023	27.87/0.6731	24.57/0.5067	28.85/0.8004	<u>30.36/0.8296</u>	30.64/0.8410
	Urban100	30.05/0.8761	25.28/0.6476	24.66/0.6147	28.06/0.7530	24.99/0.6195	27.51/0.8419	<u>29.17/0.8534</u>	29.28/0.8571
Spatial Gaussian $\sigma = 45$	CBSD68	29.34/0.8488	28.19/0.7907	27.27/0.7391	24.14/0.6686	27.27/0.7353	28.13/0.8181	<u>29.01/0.8368</u>	29.20/0.8440
	McMaster	29.79/0.8236	<u>28.68/0.7707</u>	27.79/0.7189	23.93/0.6059	27.55/0.6890	28.43/0.7778	29.21/0.8035	<u>28.63/0.7860</u>
	Kodak24	29.97/0.8377	28.32/0.7591	27.34/0.7006	22.98/0.6192	27.41/0.7040	28.73/0.8105	<u>29.32/0.8140</u>	29.79/0.8315
	Urban100	29.38/0.8633	28.61/0.8148	27.64/0.7681	25.55/0.7013	27.54/0.7715	27.33/0.8425	<u>28.77/0.8511</u>	28.98/0.8619
Spatial Gaussian $\sigma = 50$	CBSD68	28.44/0.8263	26.98/0.7446	26.13/0.6918	23.72/0.6320	26.15/0.6894	27.43/0.7954	<u>28.31/0.8164</u>	28.51/0.8250
	McMaster	29.12/0.8047	27.52/0.7231	26.64/0.6678	23.49/0.5728	26.48/0.6439	27.82/0.7571	29.57/0.8256	<u>29.01/0.8095</u>
	Kodak24	29.08/0.8149	27.06/0.7063	26.17/0.6480	22.84/0.5807	26.26/0.6520	28.00/0.7854	<u>28.65/0.7915</u>	29.13/0.8128
	Urban100	28.56/0.8438	27.38/0.7740	26.47/0.7268	24.78/0.6688	26.41/0.7302	26.77/0.8224	<u>28.12/0.8364</u>	28.37/0.8483
Poisson $\alpha = 2.5$	CBSD68	29.89/0.8731	24.03/0.6261	23.67/0.6045	25.67/0.6941	23.99/0.6064	27.69/0.8024	<u>29.21/0.8425</u>	29.36/0.8477
	McMaster	30.88/0.8628	24.94/0.6627	24.50/0.6460	25.78/0.6939	24.81/0.5906	28.42/0.7224	<u>30.33/0.8542</u>	30.38/0.8565
	Kodak24	30.77/0.8655	23.94/0.5605	23.58/0.5406	25.96/0.6440	23.94/0.5471	28.28/0.7796	<u>30.08/0.8359</u>	30.30/0.8440
	Urban100	29.44/0.8840	23.61/0.6537	23.24/0.6390	25.30/0.7044	23.65/0.6395	26.85/0.8125	<u>28.87/0.8724</u>	28.94/0.8762
Poisson $\alpha = 3.0$	CBSD68	28.68/0.8457	21.36/0.5149	21.27/0.4988	23.53/0.6172	21.65/0.5023	25.79/0.7141	<u>28.15/0.8153</u>	28.37/0.8243
	McMaster	29.80/0.8429	22.27/0.5816	22.11/0.5725	23.59/0.6322	22.48/0.5163	26.59/0.6416	<u>29.44/0.8355</u>	29.54/0.8398
	Kodak24	29.56/0.8382	21.16/0.4448	21.09/0.4321	23.88/0.5636	21.49/0.4376	26.04/0.6685	<u>29.03/0.8068</u>	29.30/0.8213
	Urban100	28.22/0.8614	21.02/0.5664	20.94/0.5574	22.87/0.6267	21.39/0.5590	25.25/0.7341	<u>27.80/0.8517</u>	27.94/0.8596
Salt & Pepper $d = 0.012$	CBSD68	31.96/0.8900	26.63/0.7968	25.51/0.7654	25.89/0.7788	26.50/0.7727	30.49/0.8623	34.94/0.9355	<u>34.39/0.9256</u>
	McMaster	31.90/0.8633	25.51/0.7606	25.00/0.7420	25.33/0.7462	25.75/0.7149	30.10/0.7976	33.81/0.9059	<u>33.13/0.8948</u>
	Kodak24	32.62/0.8805	26.97/0.7764	25.75/0.7402	26.18/0.7547	26.92/0.7536	31.16/0.8619	35.54/0.9244	<u>35.13/0.9168</u>
	Urban100	31.50/0.9009	26.05/0.8146	25.15/0.7923	25.62/0.7993	26.48/0.8030	29.08/0.8802	33.36/0.9290	<u>32.94/0.9229</u>
Salt & Pepper $d = 0.016$	CBSD68	30.85/0.8700	25.18/0.7518	24.23/0.7174	24.57/0.7256	25.13/0.7269	30.13/0.8537	34.29/0.9285	<u>33.84/0.9191</u>
	McMaster	30.83/0.8377	24.09/0.7094	23.69/0.6878	24.01/0.6883	24.41/0.6679	29.68/0.7856	33.05/0.8953	<u>32.79/0.8897</u>
	Kodak24	31.48/0.8593	25.43/0.7249	24.42/0.6850	24.78/0.6932	25.47/0.7010	30.82/0.8532	34.91/0.9171	<u>34.58/0.9102</u>
	Urban100	30.57/0.8846	24.66/0.7730	23.89/0.7479	24.36/0.7537	25.13/0.7618	28.76/0.8713	32.87/0.9252	<u>32.53/0.9191</u>
Speckle $\sigma^2 = 0.02$	CBSD68	31.81/0.9038	29.72/0.8308	28.88/0.8100	29.15/0.8277	29.32/0.8143	29.91/0.8752	31.17/0.8910	<u>31.16/0.8897</u>
	McMaster	32.28/0.8703	30.32/0.8156	29.17/0.7946	28.89/0.8003	29.87/0.7573	30.47/0.8090	31.77/0.8798	<u>31.73/0.8785</u>
	Kodak24	32.69/0.9048	30.34/0.8173	29.39/0.7907	29.73/0.8129	30.04/0.8035	30.65/0.8739	<u>31.94/0.8836</u>	31.96/0.8828
	Urban100	30.94/0.9043	28.42/0.8130	27.50/0.7930	28.22/0.8100	28.03/0.7959	28.60/0.8832	30.39/0.8981	<u>30.31/0.8977</u>
Speckle $\sigma^2 = 0.03$	CBSD68	30.49/0.8863	26.68/0.7546	25.98/0.7363	26.84/0.7668	26.47/0.7440	29.00/0.8509	<u>29.97/0.8697</u>	30.03/0.8712
	McMaster	31.30/0.8578	27.19/0.7492	26.29/0.7325	26.82/0.7526	27.03/0.6946	29.69/0.7774	<u>30.92/0.8672</u>	30.93/0.8667
	Kodak24	31.40/0.8878	27.06/0.7232	26.21/0.6947	27.29/0.7382	26.86/0.7100	29.74/0.8479	<u>30.77/0.8619</u>	30.89/0.8656
	Urban100	29.69/0.8889	25.33/0.7398	24.68/0.7256	25.86/0.7529	25.24/0.7300	27.65/0.8466	<u>29.30/0.8840</u>	29.28/0.8852
Mixture Level 1	CBSD68	30.91/0.8930	27.43/0.7713	27.03/0.7476	28.44/0.8090	27.14/0.7372	29.08/0.8710	<u>30.27/0.8701</u>	30.32/0.8707
	McMaster	31.62/0.8664	27.88/0.7494	27.53/0.7306	28.54/0.7707	27.61/0.7137	29.85/0.8104	<u>31.05/0.8606</u>	31.06/0.8610
	Kodak24	31.72/0.8874	27.66/0.7296	27.26/0.7053	29.03/0.7872	27.44/0.6976	29.91/0.8663	<u>31.00/0.8627</u>	31.11/0.8658
	Urban100	30.40/0.8928	27.13/0.7692	26.73/0.7482	28.37/0.8091	26.98/0.7539	29.77/0.8799	<u>29.77/0.8794</u>	29.77/0.8798
Mixture Level 2	CBSD68	30.31/0.8816	25.86/0.6960	25.46/0.6668	27.42/0.7623	25.62/0.6582	28.44/0.8545	<u>29.68/0.8576</u>	29.76/0.8597
	McMaster	31.07/0.8537	26.43/0.6856	26.05/0.6658	27.48/0.7240	26.18/0.6583	29.36/0.8011	<u>30.58/0.8499</u>	30.61/0.8505
	Kodak24	31.15/0.8769	25.96/0.6414	25.57/0.6126	27.97/0.7321	25.77/0.6073	29.22/0.8446	<u>30.39/0.8486</u>	30.58/0.8557
	Urban100	29.81/0.8834	25.63/0.7060	25.21/0.6825	27.31/0.7668	25.48/0.6918	27.47/0.8645	<u>29.17/0.8691</u>	29.23/0.8711
Mixture Level 3	CBSD68	29.21/0.8569	23.20/0.5652	22.93/0.5375	25.11/0.6519	23.16/0.5332	26.95/0.7905	<u>28.69/0.8338</u>	28.83/0.8398
	McMaster	30.01/0.8264	23.77/0.5665	23.56/0.5510	25.24/0.6294	23.72/0.5367	27.99/0.7488	<u>29.71/0.8269</u>	29.78/0.8298
	Kodak24	30.08/0.8537	23.18/0.5028	22.92/0.4750	25.49/0.6054	23.16/0.4745	27.50/0.7634	<u>29.51/0.8292</u>	29.70/0.8375
	Urban100	28.70/0.8632	23.03/0.5976	22.78/0.5766	24.98/0.6720	23.07/0.5857	26.29/0.8093	<u>28.18/0.8503</u>	28.29/0.8550

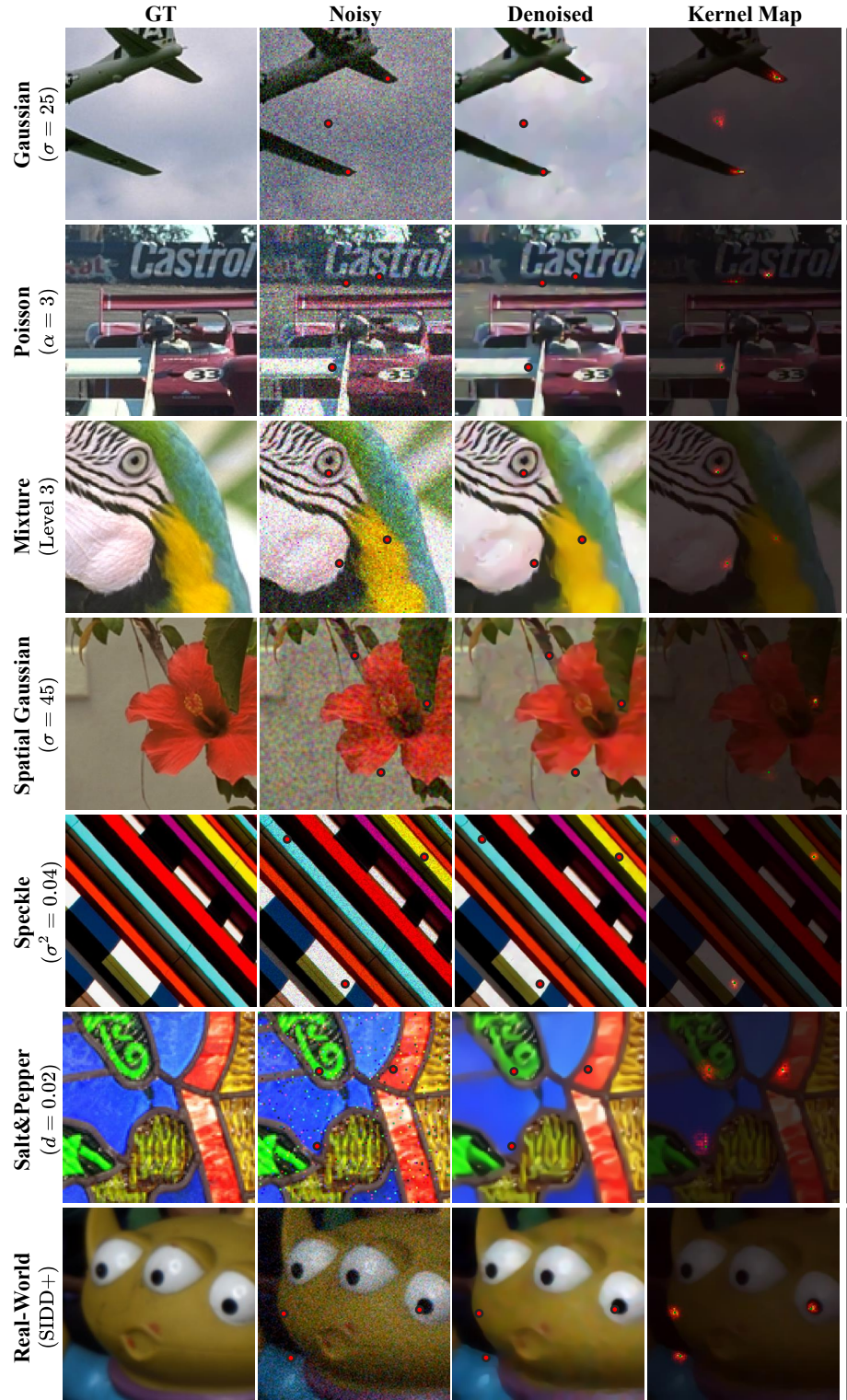


Figure A2. Visualizations of predicted kernel maps across diverse noise types. For each noise condition, we display the ground-truth (GT), noisy input, denoised output, and the predicted kernel map. Red dots in the **Noisy** and **Denoised** columns indicate reference pixels used to visualize the corresponding denoising kernels. All kernel maps are normalized to sum to one, functioning as adaptive averaging filters.

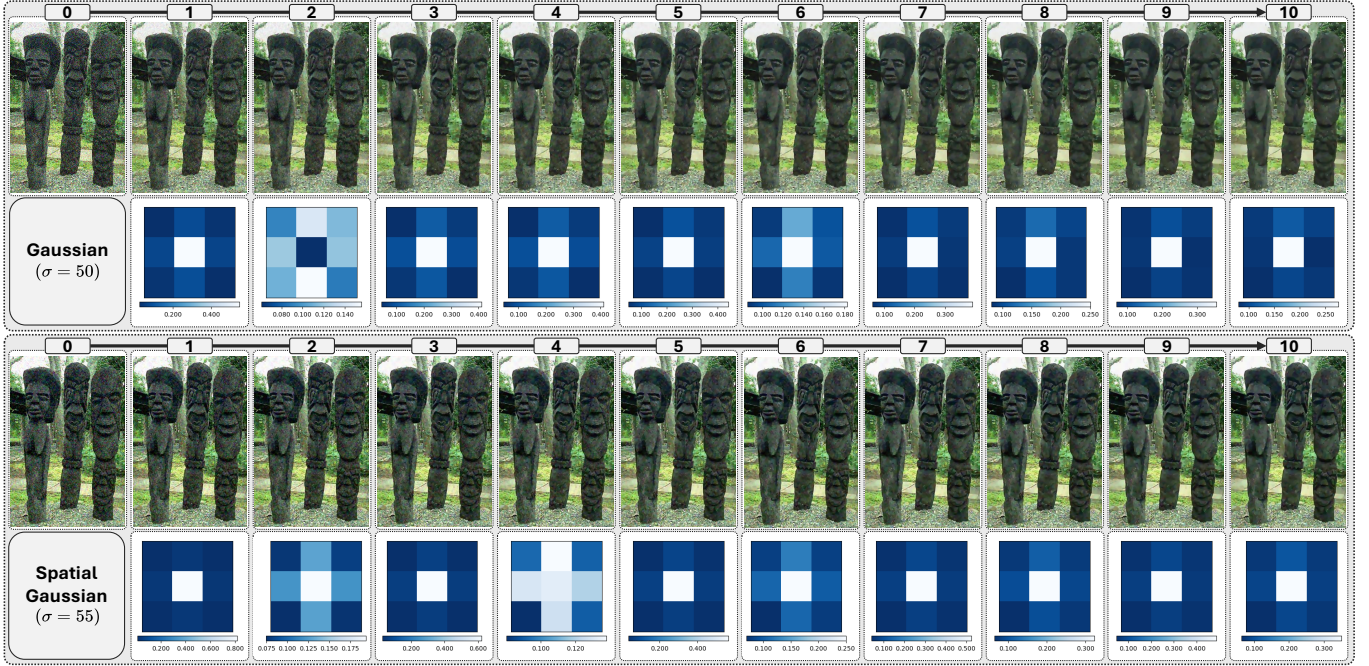


Figure A3. Comparison of results illustrating how denoising kernels and denoised images evolve through iterations, along with varying degradation types and levels on the CBSD68 [61] dataset. Please zoom in for a more detailed comparison.

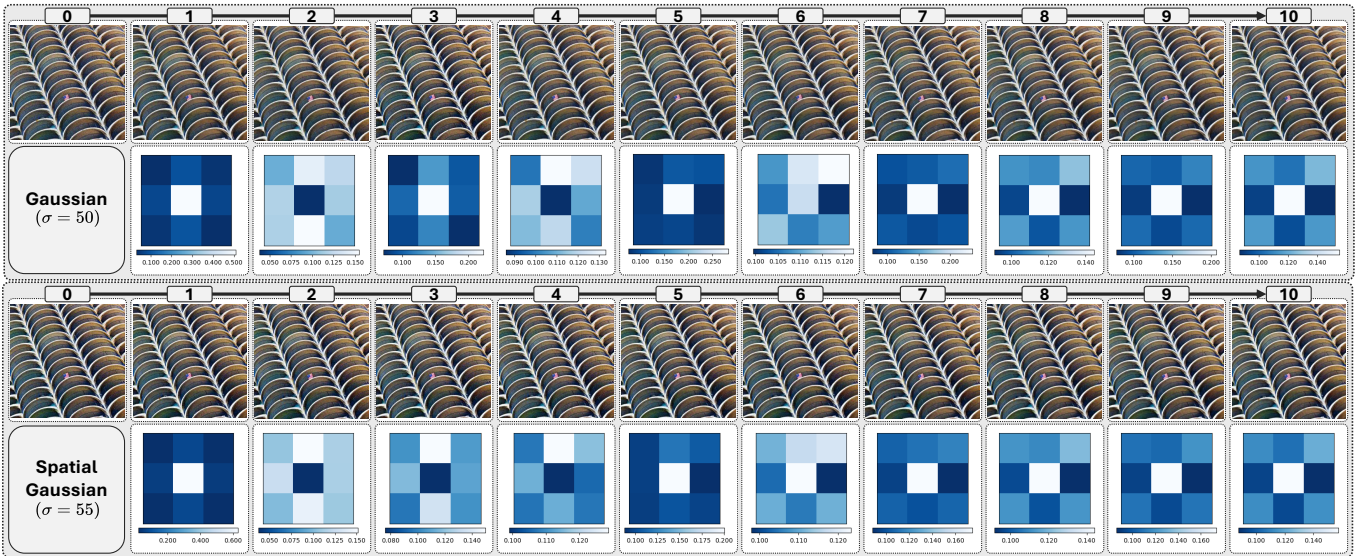


Figure A4. Comparison of results illustrating how denoising kernels and denoised images evolve through iterations, along with varying degradation types and levels on the Urban100 [25] dataset. Please zoom in for a more detailed comparison.

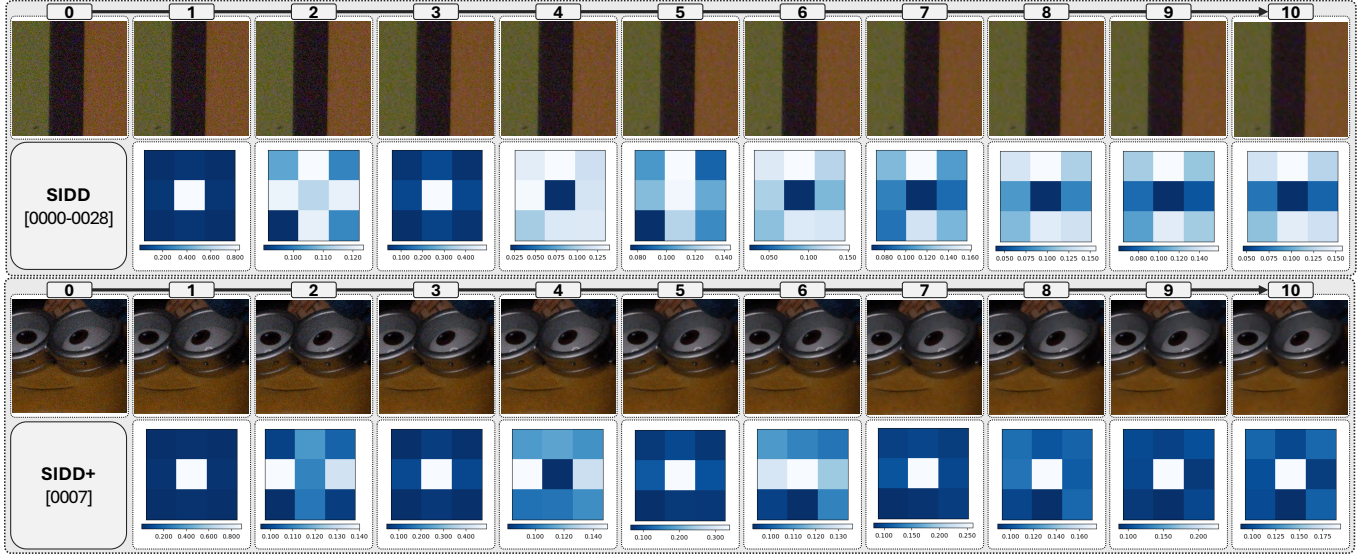


Figure A5. Comparison of results illustrating how denoising kernels and denoised images evolve through iterations, along with varying degradation types and levels on the real-world SIDD [1] and SIDD+ [3] dataset. Please zoom in for a more detailed comparison.

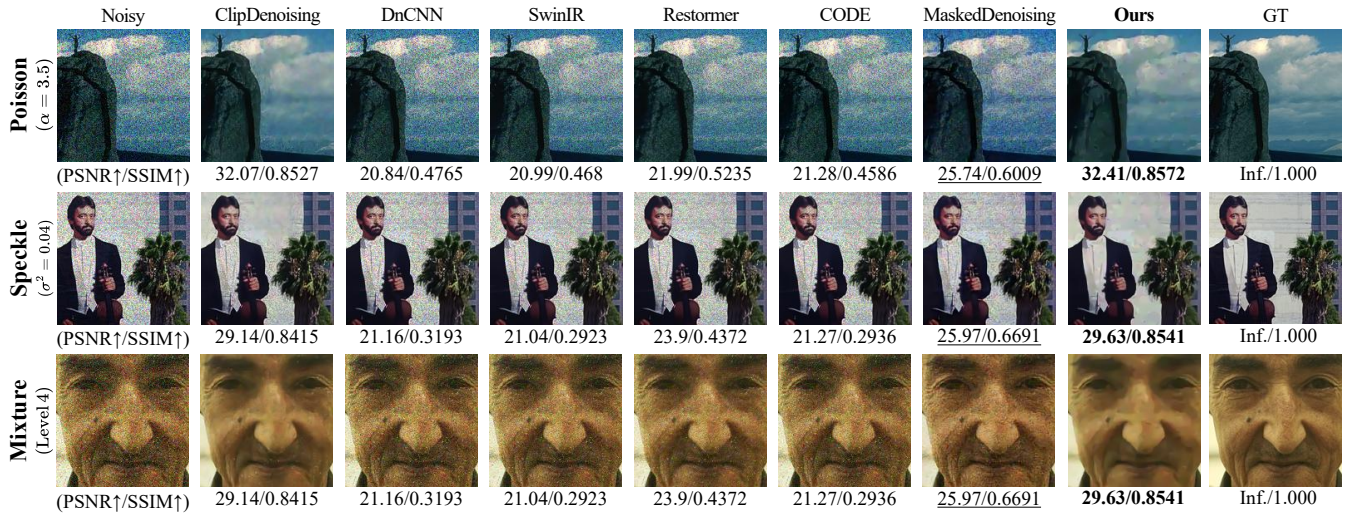


Figure A6. Qualitative results of denoising performance on synthetic OOD noise in terms of PSNR \uparrow /SSIM \uparrow . During training, none of the methods are exposed to the noise types present in the test set. Please zoom in for a more detailed comparison.

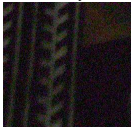


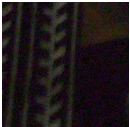

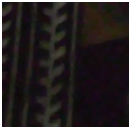
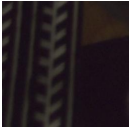



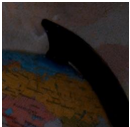


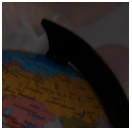
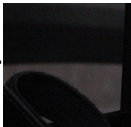
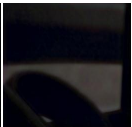
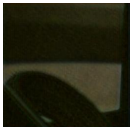
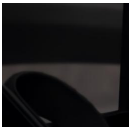

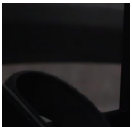
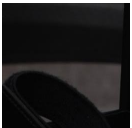
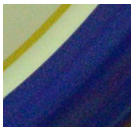
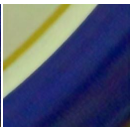
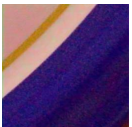
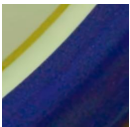
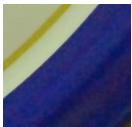
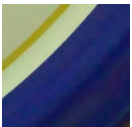

	Noisy	ClipDenoising	Restormer	CODE	MaskedDenoising	Ours	GT
SIDD							
	(PSNR↑/SSIM↑)	29.51/0.6699	<u>28.67/0.5998</u>	27.43/0.5531	28.15/ <u>0.6249</u>	32.15/0.9079	Inf./1.000
SIDD+							
	(PSNR↑/SSIM↑)	36.87/0.8849	25.94/0.622	<u>35.76/0.8531</u>	28.66/0.6665	37.11/0.9112	Inf./1.000
PolyU							
	(PSNR↑/SSIM↑)	32.43/0.7153	27.52/0.6942	<u>38.81/0.9201</u>	35.61/0.8253	41.26/0.9778	Inf./1.000
Nam							
	(PSNR↑/SSIM↑)	35.21/0.9292	22.72/0.7145	<u>35.65/0.9343</u>	33.39/ <u>0.9468</u>	37.35/0.9714	Inf./1.000

Figure A7. Qualitative results of denoising performance on real-world OOD noise in terms of PSNR↑/SSIM↑. During training, none of the methods are exposed to the noise types present in the test set. Please zoom in for a more detailed comparison.

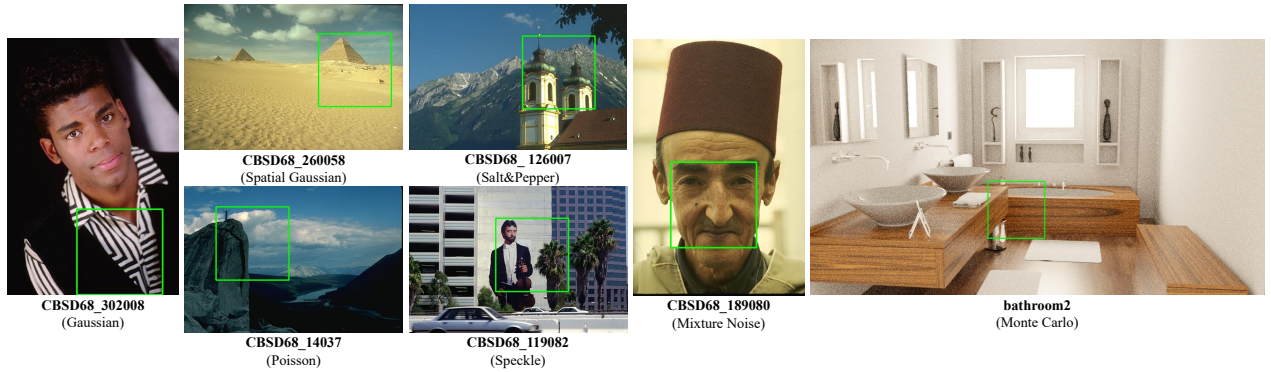


Figure A8. The original clean image and cropped region-of-interest (ROI) from the CBSD68 [61] and Monte Carlo rendering [15] dataset used for the qualitative evaluation of synthetic noise removal.