Appendix

A.1 Network Architecture

In this section, we detail the network hyperparameters used in our model, focusing on the structure of the EmbedNet, the layer-specific parameters of the UNet, and the kernel sizes for both spatial and temporal denoising.

Input-Encoder Layers. Similar to previous works (Işık et al. 2021; Balint et al. 2023), our student model and teacher model both utilize a three-layer EmbedNet connected by Leaky ReLU activations. The student model is configured with 32 channels in the hidden layers, while to capture more comprehensive hidden layer information, our teacher model employs 256 channels.

U-Net. Similar to previous denoising works, we employ a standard U-Net architecture, utilizing max-pooling for downsampling, Leaky ReLU for activation functions, and concatenating skip connections. For the student model, we optimize performance by using a more compact architecture with channel counts as follows: c32c32d → c64c64d → c128c128d in the encoder, a bottleneck with 128 channels, and a decoder with channel counts: c64c64u → c32c32u.

In contrast, the teacher model is designed with a more extensive architecture to enhance the denoising capability by learning more detailed features. It employs larger channel counts in the encoder: c128c128d → c256c256d → c512c512d, a bottleneck with 512 channels, and a decoder with channel counts: c512c512u → c256c256u.

Pyramid Kernels. We adopt a pyramid kernel structure in both the student and teacher models, following the approach by NPPD (Balint et al. 2023). Similar to NPPD, we use average pooling with a size of 2x2 for downsampling. As described in the main text, the student model employs a kernel size of 5, while the teacher model utilizes progressively larger kernel sizes of 5, 7, and 9 in successive layers. For the upsampling layers, in contrast to NPPD, we don't predict upsampling weights and use them to guide the upsampling process for each pyramid layer, which is time-consuming. Instead, we adopt a bilinear upsampling strategy which has better time performance.

A.2 Analysis of the Pyramid Filter

In our work, we adopted the pyramidal filter proposed by Balint et al. (2023) as part of our network architecture. To evaluate the effectiveness of this approach, we conducted a series of experiments comparing the performance of networks utilizing the pyramid structure against those that directly predict parameters using the UNet. Fig. A.1 revealed that student models without the pyramid structure struggled to effectively remove noise, demonstrating a clear disadvantage in denoising capabilities. On the other hand, the inclusion of the pyramid structure did not significantly impact the runtime efficiency, indicating that it offers a favorable balance between performance and computational cost.

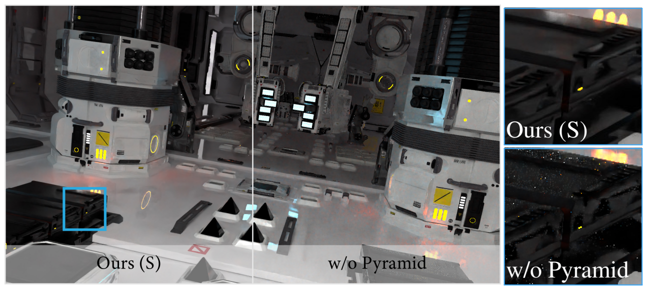

Figure A.1: **Ablation on the pyramid filter.** The student model without the pyramid structure struggles to effectively remove noise, as shown in the noisy conditions presented in the image, demonstrating a clear disadvantage in denoising performance.

A.3 Run Times

We use TensorRT version 8.6 to optimize CUDA kernels for inference of all neural network layers in our student model. The kernel fusion strategy followed the default settings, including simple ReLU fusions and automatic kernel tuning. At run time, the optimized kernel use tensor cores and enables mixed precision with FP16.

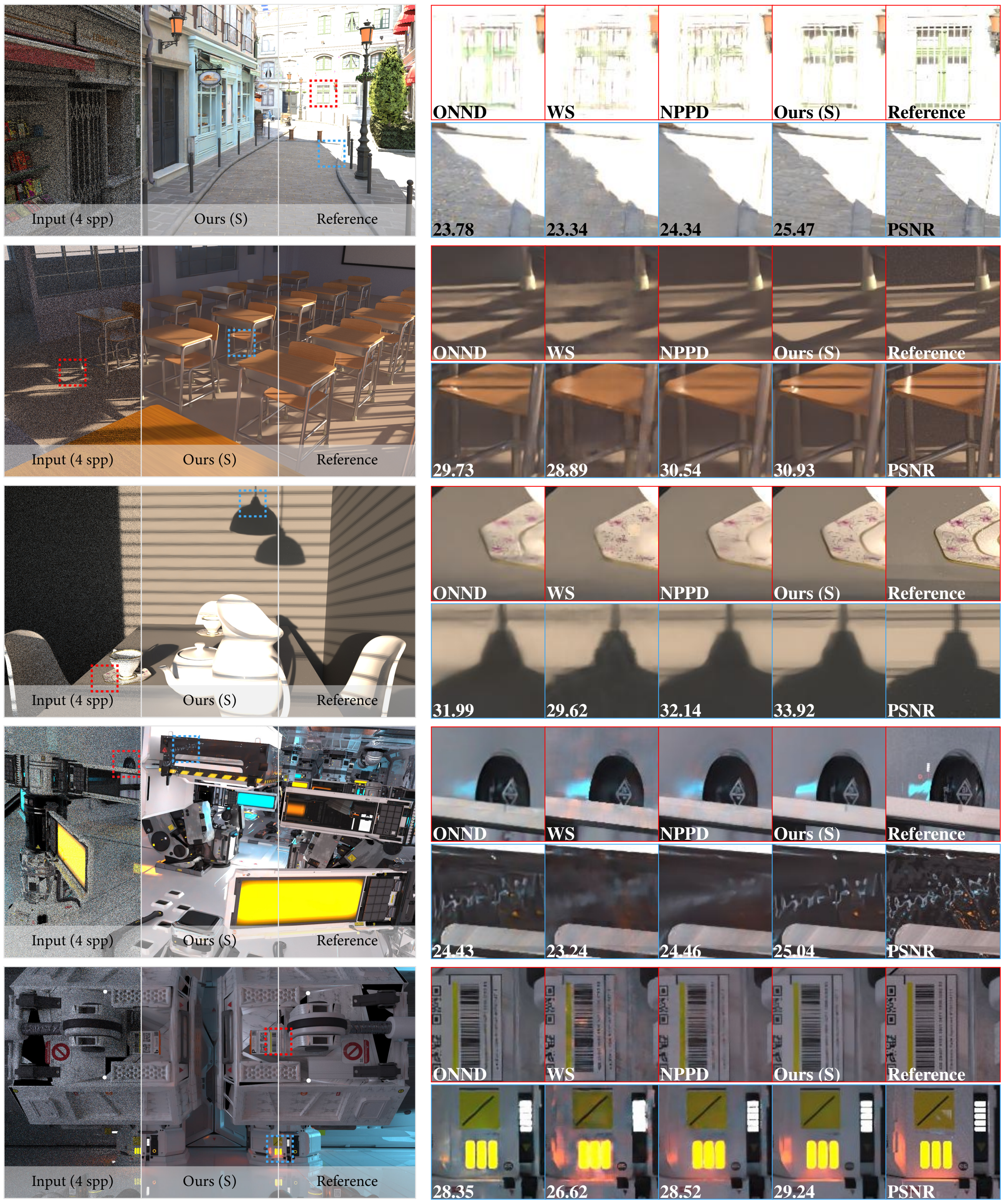

Figure A.2: Qualitative and quantitative evaluation of our student model (Ours (S)) on denoising input images rendered at 4 spp. We make comparisons with several real-time Monte Carlo denoising methods: ONND, WS and NPPD, on the ***Bistro*, *Classroom*, *Dining room*** and ***Zero-Day*** scenes.

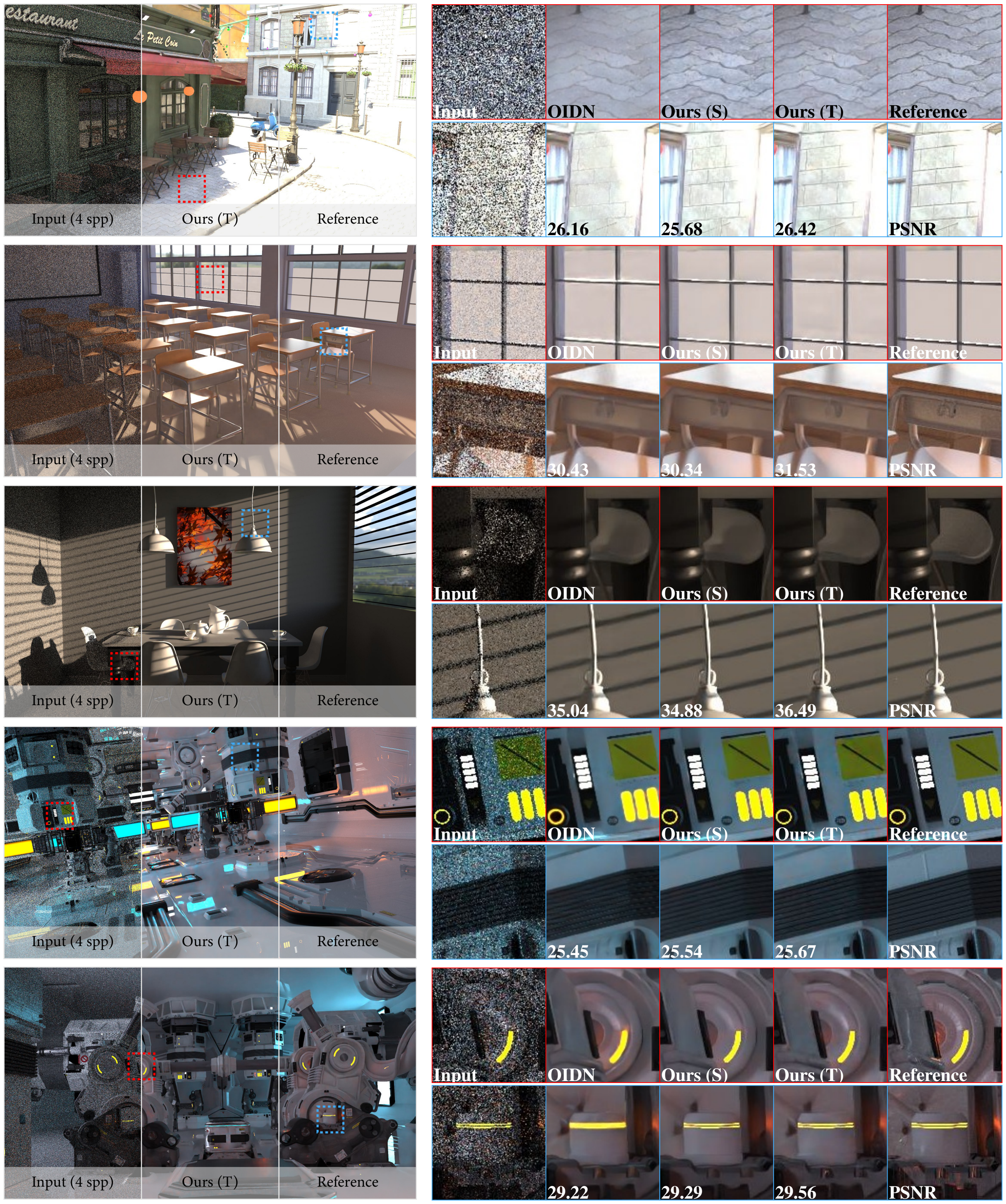

Figure A.3: Qualitative and quantitative comparisons of our two models: Ours (T) and Ours (S) against the state-of-the-art offline Monte Carlo denoising method: OIDN, on the ***Bistro*** **,** ***Classroom*****,** ***Dining room*** and ***Zero-Day*** scenes.