Real-time Neural Denoising with Render-aware Knowledge Distillation

Mengxun Kong[1], **Jie Guo**[1*], **Chen Wang**[1], **Ye Yuan**[1], **Yanwen Guo**[1]

[1]Nanjing University, Nanjing 210023, China
{mengxunkong, chenwang}@smail.nju.edu.cn
{guojie, ywguo}@nju.edu.cn
yuanyeorange@gmail.com

Abstract

Monte Carlo ray tracing typically produces noises due to its stochastic nature. Currently, using a dedicated neural network trained on a large dataset has become a prevalent way to perform denoising. However, existing neural networks that produce high quality noise-free images are usually plagued by their low efficiency due to the large model size, preventing them from being deployed to real-time applications. To address this fundamental issue, we propose a render-aware knowledge distillation (RAKD) framework, specifically designed for neural network based Monte Carlo denoising methods. The framework is comprised of several key modules that help to better transfer rich learning representations from a well-performing but cumbersome teacher model to a compact and lightweight student model. Considering that rendering contents are dense and structural, we adopt a local-to-global knowledge distillation module that operates on both pixel level and image level. We also incorporate a render-aware parameter initialization strategy and construct an auxiliary unlabeled dataset to further improve the student model's performance and generalization ability. Experimental results demonstrate that RAKD achieves state-of-the-art denoising quality while upholding real-time performance, successfully tackling the computational constraints faced by resource-limited devices.

Introduction

Monte Carlo (MC) ray tracing can simulate a wide range of lighting effects by randomly sampling light paths and averaging their contributions in every pixel. Despite its significant computational cost, it has become the de facto standard in offline rendering scenarios (e.g., movie production) due to its accuracy and flexibility. Recent advances in powerful hardware and GPU ray tracing APIs (Sandy, Andersson, and Barré-Brisebois 2018; Harada 2020; Burgess 2020) bring real-time ray tracing (RTRT) into reach. Nonetheless, ray budgets are still limited, particularly on resource-limited devices. Existing RTRT algorithms can only afford very low sampling rates (typically 1˜4 spp), resulting in distracting noise.

Over the past decade, neural networks have demonstrated remarkable success in image denoising for Monte Carlo (MC) rendering, offering significant advancements in visual quality and computational efficiency (Zwicker et al. 2015; Huo and Yoon 2021), with architectures incorporating advanced techniques like transformer blocks (Lu, Xie, and Shen 2020; Yu et al. 2021), deformable convolution (Wei et al. 2021), and adversarial training (Xu et al. 2019; Lu et al. 2021; Yu et al. 2021). However, despite their impressive performance, the practical deployment of neural denoising methods in real-time graphical applications is often hindered by the high computational resources and memory requirements. To address this issue, research has increasingly focused on the development of lightweight neural models (Meng et al. 2020; Fan et al. 2021). These models, while beneficial in reducing computational load, typically achieve lower image quality due to their reduced model capacity and limited learning ability.

Monte Carlo denoising, particularly in the context of real-time applications, has seen a variety of approaches with both traditional and neural techniques. Traditional real-time denoising filters, such as those proposed by Schied et al. (2017); Zhdan (2021); Nvidia (2022b,a), continue to be prevalent in scenarios demanding high frame rates, where computational budgets are constrained. These methods, although non-machine-learning-based, excel in delivering performance under tight computational constraints.

In the real-time context, the focus has been on simplifying model size to meet performance demands. For instance, Meng et al. (2020) introduced a lightweight network that predicts a multi-scale pyramid to splat noisy data into a bilateral grid, while Işık et al. (2021) leveraged the pairwise affinity of per-pixel features for noise reduction and temporal stability. Additionally, Fan et al. (2021) proposed a network encoding of kernel weights to construct filter kernels using a hand-crafted decoder, and Thomas et al. (2022) combined supersampling and denoising within a U-Net inspired architecture. Most recently, Balint et al. (2023) introduced a novel pyramidal filter equipped with adaptable partitioning and upsampling stages.

While lightweight kernel prediction networks have demonstrated fast performance with decent denoising results, they often suffer from limited receptive fields and over-blurring issues. Our work builds upon these efforts by introducing a lightweight kernel prediction network that leverages knowledge distillation, enabling the network to learn

[*]Corresponding author

from the broad receptive fields and detail preservation capabilities of offline teacher denoisers. This approach aims to balance the trade-off between computational efficiency and image quality, making it suitable for real-time Monte Carlo rendering.

To address the above issue, we resort to knowledge distillation (KD) (Hinton, Vinyals, and Dean 2015) which is a promising model compression technique that transfers rich learning representations from a well-performing but cumbersome pre-trained teacher model to a compact and lightweight student model(Hinton, Vinyals, and Dean 2015). This approach has gained popularity for its ability to allow smaller models to perform at a level closer to that of larger models while requiring fewer computational resources, making it particularly useful in resource-constrained environments.

Knowledge distillation was first introduced by Hinton, Vinyals, and Dean (2015), demonstrating that the soft outputs (class probabilities softened with temperature) of a trained model could train a smaller model more effectively than hard labels alone. Beyond image classification tasks, KD has been successfully applied to more complex per-pixel tasks such as semantic segmentation and depth estimation. For instance, Liu et al. (2019, 2020) presented structured knowledge distillation schemes, including pair-wise and holistic distillation using GANs, to enforce consistency between the outputs of compact and cumbersome networks.

In the realm of image denoising, Young et al. (2021) proposed a Feature-Align layer that enforces attention to spatially varying noise, enhancing KD on RAW image denoising through a feature matching loss. Similarly, Zou et al. (2023) introduced LUNet, a lightweight image denoise network, alongside a distillation algorithm with retargeting supervision to improve efficiency. Other innovations include Shu et al. (2021)'s channel-wise distillation for dense prediction tasks, which differs from traditional spatial distillation methods.

Despite its success in various domains, knowledge distillation has yet to be widely adopted in real-time rendering applications. Our proposed render-aware knowledge distillation (RAKD) framework aims to bridge this gap by leveraging KD to significantly reduce the computational demands of neural denoising models, ensuring the high quality of denoised images even in real-time settings. By doing so, we aim to balance the trade-off between model efficiency and image quality, facilitating practical deployment in real-time Monte Carlo rendering.

The proposed RAKD framework is comprised of a local-to-global knowledge distillation module, a render-aware parameter initialization strategy, and an auxiliary unlabeled dataset. The local-to-global knowledge distillation module is designed to align the features of the teacher model and the student model on both pixel-level and image-level. This helps to preserve the structural information in the dense rendering contents as much as possible during knowledge distillation. The render-aware parameter initialization strategy tries to facilitate the training of the small-sized student model, making it converge fast to the optimal solution of the teacher model. In addition, an auxiliary unlabeled dataset is collected, aided by the teacher model, to improve the generalization ability of the student model. Experimental results on multiple benchmark scenes have demonstrated the effectiveness and superiority of the proposed RADK framework. In particular, it helps to achieve state-of-the-art Monte Carlo denoising quality with a very small neural network, guaranteeing real-time running performance.

Methodology

In this section, we unfold the details of our render-aware knowledge distillation (RAKD) framework, specially designed for solving the Monte Carlo denoising task in real time.

Overview

An overview of RAKD is shown in Fig. 1. As a typical KD framework, RAKD contains a computationally expensive teacher network and a lightweight student network. The teacher network has a large size and is designed to learn rendering-aware knowledge as much as possible from the training dataset. In contrast, the student network has much fewer trainable parameters than the teacher network. The key point lies in refining the knowledge transfer process from the teacher model to the student model. In this section, we first describe our denoise model architecture and then introduce our RAKD framework. Finally, we elucidate our implementation details. Our RAKD framework can be summarized into several strategies:

- We introduce a local-to-global knowledge distillation module so that we can distill structural rendering information both on pixel level and image level.
- We adapt render-aware knowledge initialization to provide knowledge for our student model at the beginning of our training and facilitate the training of our student model.
- We make use of auxiliary unlabeled dataset to expand our dataset in reasonable time. The auxiliary unlabeled dataset is rich in denoising knowledge, which is good for student's knowledge distillation process.

Teacher model and student model

Our teacher model and student model all follow the basic rule of kernel prediction, as described in Bako et al. (2017); Vogels et al. (2018), and use downsampling and upsampling approaches to generate denoising kernels. We further suppress noise by using a pyramid-layer architecture similar to (Balint et al. 2023) when using the predicted kernels to filter on the input radiance.

Our teacher network employs a complex architecture and extensive computational power to produce excellent denoising results. Our teacher network has a design of multi-layer architecture, which is composed of three layers. Each layer contains an EmbedNet and a kernel predictor to generate kernels. We let these layers generate different sizes of kernels to capture different scales of details. Bigger denoising kernels allow for better denoising quality while increasing

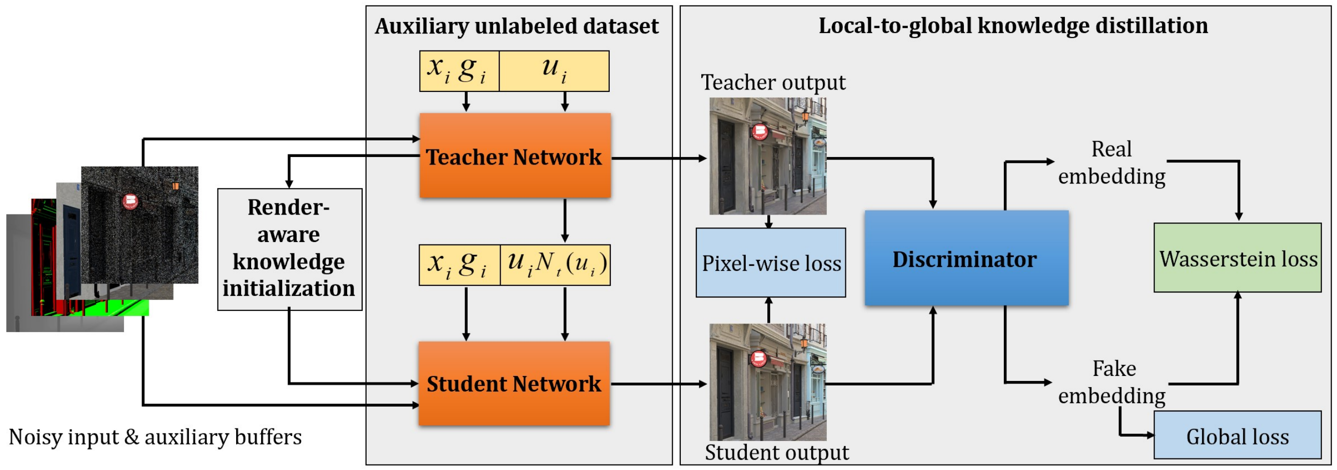

Figure 1: The proposed RAKD framework contains several knowledge distillation modules: a local-to-global knowledge distillation module, a render-aware knowledge initialization strategy, and an auxiliary unlabeled dataset. We compute pixel-wise loss between the denoised outputs of the student model and the teacher model. We also compute the global loss from the discriminator. These losses are used to guide the knowledge distillation process from the teacher model to the student model. The Wasserstein loss is used to train the discriminator.

computational cost (Fan et al. 2021), and our teacher network employs big denoise kernels. The kernel sizes range from 5 to 9 with a step size of 2.

Our denoise input is made up of low sample per pixel noisy input $\mathcal{L}_t$ and auxiliary features f_t at frame t which are computed at primary ray intersections, including albedo, normal, and depth. They are concatenated into a tensor with 10 channels.

As shown in Fig. 2, the input tensor is first sent into the embed net composed of several convolution layers to project the original input into a high-dimensional space, yielding the embedded feature vector e_t. The embedded feature vector is then sent into the U-Net-based kernel predictor to get denoising kernels k_t and temporal weight λ_t.

Then, we use the predicted kernels k_t to filter the noise radiance L_t through kernel convolution and get the denoised radiance O_t. We follow the state-of-the-art pyramid architecture which is proven to be good at avoiding low-frequency responses when filtering the input radiance. For each layer in our teacher model, we downsample the noise input into 4 pyramid layers and denoise the noise radiance at different resolutions. Then we upsample each pyramid layer's denoising results and combine them.

After each layer filters the original noise radiance, our teacher model uses a blending weight (BW) predictor to generate per-pixel weight maps for the denoising results of each layer. The denoising results are then blended according to the weight maps to produce the final output. Our BW predictor also uses a U-Net as its backbone. For our teacher network where there are three denoise layers, we operate the following blending step:

$$O_t = \sum_{i=1}^{3} w_i O_{ti} \tag{1}$$

where w_i is the weight map generated by our BW predictor.

We compute the accumulated per pixel noise $\mathcal{L}_t$ and auxiliary features f_t over time for temporal stability:

$$\mathcal{L}_t = \lambda_t \mathcal{L}_t + (1 - \lambda_t)\mathscr{W}_t \mathcal{L}_{t-1} \tag{2}$$
$$f_t = \lambda_t f_t + (1 - \lambda_t)\mathscr{W}_t f_{t-1} \tag{3}$$

where $\mathscr{W}_t$ denotes the warping operator, which uses motion vector to warp and bilinearly interpolate frame $t - 1$ to frame t. For the final output, we incorporate an extra temporal blend:

$$O_t = \lambda_t O_t + (1 - \lambda_t)\,\mathscr{W}_t O_{t-1} \tag{4}$$

Our student model is designed for real-time denoising. We choose the layer with kernel size of 5 in our teacher network as our student network for real-time requirements.

We further adjust the student's pyramid architecture for real-time requirements, where we reduce the channels in the embedding layer and U-Net.

Local-to-global knowledge distillation

Rendering contents are dense and structural, so we adopt a local-to-global knowledge distillation strategy. On the local level, we apply our local knowledge distillation, where we compute the per-pixel loss for the denoised output of our teacher model and student model.

Recent work successfully introduced GAN to the KD process for better and more stable learning (Micaelli and Storkey 2019; Fang et al. 2019; Liu et al. 2020). As for our global distillation, we introduce the generative adversarial network to our RAKD and perform distillation on image level.

In our work, GAN with self-attention layers (Zhang et al. 2019) is applied as a regularization part for the KD process. We treat our student network as the generator part of the GAN, whose denoised outputs are treated as fake samples,

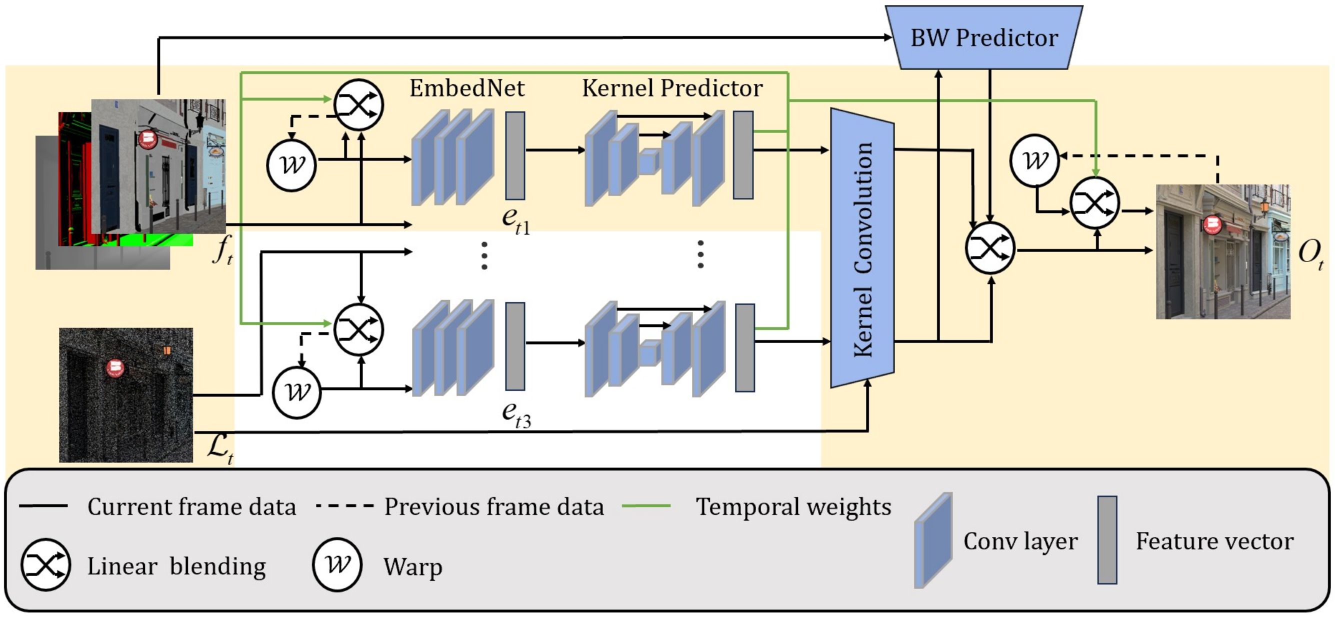

Figure 2: Network architecture of the teacher network and student network. The complete figure represents our multi-layered teacher network, which infers denoising results across various layers. The section highlighted with a yellow background represents our student network, which employs a single layer with a kernel size of 5. Note that there are differences in the number of channels between the EmbedNet and U-Net Kernel predictor components of our student and teacher networks.

and the denoised outputs of the teacher network are treated as real samples. We design a discriminator additionally, and the self-attention scheme is adopted for better capturing the space information.

GAN suffers from unstable gradient in training when we use Kullback-Leibler (KL) or Jensen-Shannon (JS) divergence; therefore, in our case, the Wasserstein distance (Gulrajani et al. 2017) is applied:

$$\mathcal{W}(p,q) = \sup_{\|f\|_L \le 1} \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)], \tag{5}$$

where the supremum is taken over all Lipschitz functions f with a Lipschitz constant not greater than 1.

Render-aware knowledge initialization

The weights of the student network are often initialized as a function of the weights of the teacher network in knowledge distillation (Lin et al. 2020; Wang et al. 2023). We introduce our render-aware knowledge initialization into our RAKD framework, allowing the knowledge about the rendering process learnt by our teacher model to be efficiently initialized in our student model. The U-Net-based kernel predictor in our denoising framework plays a core part in generating denoising kernels and temporal weights. We design the kernel predictor of our teacher and student models with composed downsampling and upsampling layers, which have different channels but the same number of layers.

Since our pre-trained teacher network and our student network share the same U-Net backbone, we conveniently extract parameters from the teacher and use them to initialize the student. Specifically, when extracting the corresponding layers' parameters from the teacher model with shape (h_t, w_t, k, k) to the student model's layers with shape (h_s, w_s, k, k), we use a step size of $\lfloor \frac{h_t}{h_s} \rfloor$ and $\lfloor \frac{w_t}{w_s} \rfloor$ in the two dimensions to extract the necessary parameters.

Auxiliary unlabeled dataset

The efficacy of neural networks has been proven to be heavily reliant on the quantity and quality of datasets. Previous works (Sun et al. 2017) generate a large amount of high *spp* data, ranging from 900 (Meng et al. 2020) to 65536 (Balint et al. 2023), and it takes much computational resource and time to generate numerous noise-free high *spp* data.

Inspired by the fact that an auxiliary unlabeled dataset is significant to the KD process in other pixel-wise tasks like dense prediction (Hu et al. 2023), we introduce an auxiliary unlabeled dataset to our denoising task. Our unlabeled dataset is generated by using our pre-trained teacher model to infer a large amount of noisy inputs and generate noise-free results.

Our dataset is composed of the labeled part $\mathcal{X}$ and the unlabeled part $\mathcal{U}$. The loss function for student model can be formulated as:

$$\begin{aligned} L = &\frac{1}{\mathcal{X}} \sum_{x_i, g_i \in \mathcal{X}} (\sigma l(N_s(x_i), N_t(x_i)) + \\ &(1-\sigma) l(N_s(x_i), g_i)) + \\ &\frac{1}{\mathcal{U}} \sum_{u_j \in \mathcal{U}} (l(N_s(u_j), N_t(u_j)) \end{aligned} \tag{6}$$

where l denotes our basic loss function, N_s and N_t denote our student network and teacher network respectively, and g_i denotes the high *spp* reference for noisy input x_i. σ is set to 0.5 in practice to balance the knowledge distillation process.

Then the overall spatial loss function for student can be formulated as:

$$L^{spatial} = L - \mu \frac{1}{|\mathcal{X}| + |\mathcal{U}|} \sum_{x_i \in \mathcal{X} \cup \mathcal{U}} \mathcal{W}^s(D(N_s(x_i))) \quad (7)$$

$$\mathcal{W}^s(D(N_s(x_i))) = E_{N_s(x_i) \sim p_s(N_s(x_i))} \left[D\left(N_s\left(x_i\right)\right] \right. \quad (8)$$

where L indicates the loss function in (6), D indicates the discriminator of our GAN, and $\mathcal{W}^s$ is part of the the Wasserstein distance in (5), which denotes our global loss for global knowledge distillation. $p_s\left(N_s\left(x_i\right)\right)$ denotes the distribution of our student model's denoised output. μ is set to 0.05 in practice to make these two parts comparable.

By leveraging an auxiliary unlabeled dataset, it is possible for us to greatly expand the dataset in reasonable time since our teacher model takes much less time to get noise-free results compared with time costing high *spp* reference render process. The student only learns from the teacher on the unlabeled dataset, which emphasizes the transfer of dark knowledge (Hinton, Vinyals, and Dean 2015) and thus optimizing the KD process.

Implementation details

Dataset We generate our labeled training dataset using 6 publicly available Tungsten scenes (Bitterli 2016), and our unlabeled dataset is generated using 4 publicly available Tungsten scenes and 4 publicly available PBRT (Pharr, Jakob, and Humphreys 2016) scenes. We use the Falcor renderer (Kallweit et al. 2022) to render our input noise, G-Buffer, and reference frames with 6000 samples per pixel. Our training dataset consists of a total of 600 10-frame sequences. We randomize the objects' materials and texture for diversity in each sequence.

Loss function We choose symmetric mean absolute percentage error (SMAPE) as our basic loss function following Munkberg and Hasselgren (2020), which can be formulated as:

$$E(r,d) = E_{x,y} \left[\frac{d_{x,y} - r_{x,y}}{|d_{x,y}| + |r_{x,y}| + \epsilon} \right] \quad (9)$$

where r denotes the reference image and d denotes the denoised image. We set ϵ to 0.001 in practice.

As for our GAN part, we use Wasserstein distance as discussed above. Our SMAPE loss also considers temporal loss where we use the motion vector to warp the last frame.

Temporal processing We adapted the idea of BackPropagation Through Time (BPTT) (Graves 2012) to improve our model's temporal stability. In detail, we train our model on 10-frame sequences and we compute temporal loss on adjacent frames.

We extend our SMAPE loss for adjacent frames like:

$$L^{temporal} = SMAPE((T_t - \mathscr{W} T_{t-1}), (O_t - \mathscr{W} O_{t-1})) \quad (10)$$

where T_t denotes the reference for frame t and O_t denotes the denoised output for frame t.

The temporal loss is then added to the spatial loss function (7):

$$L^{all} = L^{spatial} + \lambda L^{temporal} \quad (11)$$

λ is set to 0.1 to make these two parts comparable.

Training Our denoiser is implemented with PyTorch (Paszke et al. 2017). We use Adam optimizer (Kingma and Ba 2014) with a learning rate of 10^{-4} in the beginning, and we use a stepLR scheduler to decay our learning rate by a factor of 0.8 every 100 epochs. We take 256 x 256 patches on our dataset and augment them with rotations and flips. We use a batch size of 8 to train our model. It takes about 500 epochs for our teacher model to converge on our labeled dataset, costing about 3 days on a single RTX 4090 GPU, and it takes about 800 epochs for our student model to converge on the union of our labeled dataset and unlabeled dataset, costing about 5 days on a single RTX 4090 GPU. We train the GAN's discriminator together with our student model with the same training setup. We further optimize our model to run in real-time using TensorRT. [1]

Results

We evaluate the performance of our RAKD framework and two denoisers (the large teacher model and the small student model) on a set of test scenes, including the *Bistro* (Lumberyard 2017), *Zero-Day* (Winkelmann 2019), *Classroom* and *Dining room* (Bitterli 2016). We allow for a 10-frame warm-up phase at the start of every test sequence for our method and compared methods to accumulate temporal information. We utilize ACES tone mapping (Reinhard et al. 2023) to tonemap the output frame.

Comparisons with real-time methods

We compare our student model to state-of-the-art real-time denoisers: the Nvidia OptiX AI-accelerated Denoiser (ONND) version 7.5 with temporal (Hasselgren et al. 2020) and kernel-based (Bako et al. 2017) extensions enabled (Chaitanya et al. 2017), the weight-sharing model (WS) by Fan et al. (2021) and the neural partitioning pyramids model (NPPD) by Balint et al. (2023). We retrain the weight-sharing model and NPPD on our labeled dataset while using the pre-trained ONND included in OptiX. Since the NPPD model needs per-sample data as input, we adjusted the per-sample data encoding for fair comparison with our per-pixel data input approach. All comparisons were conducted on the NVIDIA RTX 4090 GPU. Our student model, leveraging TensorRT acceleration, denoises a full HD frame (1920×1080) in 10 ms, comparable to the 10 ms required by ONND, the 3-5 ms by the weight-sharing model, and the 30 ms by the NPPD model.

We compare different methods on commonly used per-pixel peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) metrics. The comparative results are presented in Table 1.

[1]https://developer.nvidia.com/tensorrt

Method	*Bistro*		*Zero-Day*		*Dining room*		*Classroom*	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Ours (S)	**25.36**	**0.832**	**26.95**	**0.897**	**35.02**	**0.974**	**30.62**	**0.946**
ONND	23.86	0.784	25.38	0.878	32.77	0.964	29.06	0.934
NPPD	24.33	0.792	25.99	0.879	33.66	0.965	30.08	0.939
WS	23.12	0.722	24.64	0.856	30.37	0.953	28.62	0.924
Ours (T)	**26.54**	**0.848**	**27.14**	**0.903**	**36.42**	**0.977**	**31.87**	**0.953**
OIDN	25.26	0.815	26.90	0.897	35.31	0.974	31.18	0.951

Table 1: Quantitative comparison of the baseline methods on 4 spp inputs. We show denoising results on the ***Bistro*, *Zero-Day*, *Classroom*** and ***Dining room*** scenes. Higher scores indicate better performance.

Figure A.2 detailed in Appendix A shows a visual comparison of real-time denoising performance for selected scenes. Other methods (ONND and WS) retain some high and low-frequency noise. Additionally, NPPD demonstrates relatively effective denoising but tends to over-blur high-frequency details at locations where the feature buffers fail to capture such information. Our student model is able to effectively remove noise while preserving lighting and texture details. In the red inset of the *Bistro* scene, Our method can effectively handle complex lighting and shading effects, whereas ONND, WS, and NPPD struggle with these aspects. In the *Dining room* scene, our method can faithfully preserve the intricate details of the plates. In contrast, ONND and WS fail to effectively remove the noise, while NPPD ends up over-smoothing the details. In the blue inset of the *Zero-Day1* scene, our method is able to effectively remove noise while preserving the intricate material and lighting details. In contrast, NPPD and WS fail to capture these fine details. While ONND is also able to handle the material details correctly, it is unable to reproduce the reddish glow emanating from the light sources in the scene.

Comparisons with offline methods

We compare our teacher model and student model with the state-of-the-art offline denoiser Intel Open Image Denoise (OIDN) (Áfra 2024) version 2.0.0. For comparison, we use the available pre-trained OIDN model. OIDN typically requires approximately 70ms to denoise a single 1920 × 1080 picture on Intel i9 CPU, a performance level that aligns with our teacher model's denoising time for images of the same size. Table 1 shows the quantitative comparison results.

Figure A.3 detailed in Appendix A shows a visual comparison of offline denoising performance for selected scenes. While OIDN exhibits reasonably effective denoising capabilities, it struggles to handle the intricate texture details of materials and the complex lighting and shading effects in the scenes. In contrast, the teacher model, with the aid of the GAN discriminator during training, has a stronger ability to preserve fine details. Furthermore, through the knowledge distillation training process, the student model can learn and inherit the detail-preserving capabilities of the teacher to a certain extent. In the red inset of the *Bistro* and the blue inset of the *Zero-Day1* scenes, the teacher model can effectively preserve the intricate details of object boundaries. Through the knowledge distillation process, the student model has also learned to some extent the capability of boundary preservation, exhibiting reasonably good denoising performance. In contrast, OIDN struggles when dealing with multiple overlapping distant objects, often resulting in blurring or color confusion issues. In the *Classroom* and *Dining room* scenes, we can also observe that the student model has effectively learned the light and shadow handling capabilities of the teacher model. As a result, it is able to accurately reproduce the visual effects of shadows cast by objects such as windows and lamps, and the denoising performance approaches that of the offline OIDN method.

	Method	Ours (S)	w/o D	w/o G	w/o I
Bistro	PSNR	**25.36**	24.58	24.93	25.22
	SSIM	**0.8322**	0.7715	0.8196	0.8149
Zero-Day	PSNR	**26.95**	25.97	26.69	26.67
	SSIM	**0.8972**	0.8804	0.8930	0.8925
Classroom	PSNR	**30.62**	26.54	29.64	29.85
	SSIM	**0.9461**	0.9049	0.9370	0.9380
Dining room	PSNR	**35.02**	32.43	33.07	33.80
	SSIM	**0.9739**	0.9615	0.9667	0.9688

Table 2: Quantitative comparison for ablation study. Our complete knowledge distillation framework is compared with (1) training the student model only on the labeled dataset (the column ”w/o D”), (2) training the student model only with our local per-pixel loss, without GAN regularization and global loss (the column ”w/o G”), and (3) replacing our render-aware knowledge initialization strategy with the Xavier initialization strategy (the column ”w/o I”). Metrics are averaged on the ***Bistro*, *Zero-Day*, *Classroom*** and ***Dining room*** scenes. Higher scores indicate better performance.

Ablation on different knowledge distillation choices

We perform ablation experiments to validate the necessity of our proposed distillation strategies. We train our student model only on our labeled dataset, we train our student model only with the local pixel-wise distillation, without GAN regularization and global loss and we train our student model with Xavier initialization (Glorot and Bengio 2010) instead of using our render-aware knowledge initialization to initialize it respectively. Table 2 shows metrics averaged on our test scenes and Figure 3, Figure 4, and Figure 5 show visual comparisons of our ablation experiments.

Performance analysis

Our local-to-global knowledge distillation includes both local pixel-level distillation and global image-level distillation. The global distillation is achieved through the GAN, which is crucial to the KD process and serves as regularization for the KD process. Without the GAN's discriminator,

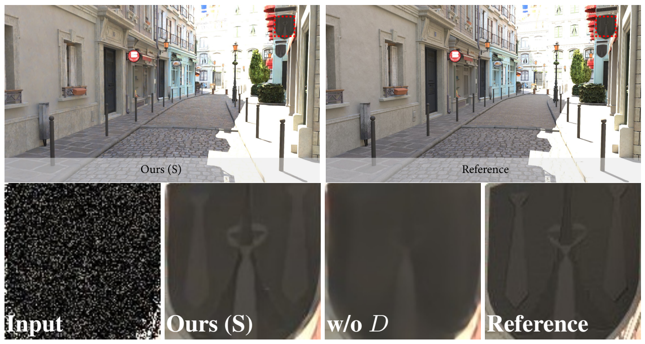

Figure 3: **Ablation on the auxiliary unlabeled dataset.** Our complete framework is compared with training the student model only on our labeled dataset, without the unlabeled dataset, on the ***Bistro*** scene.

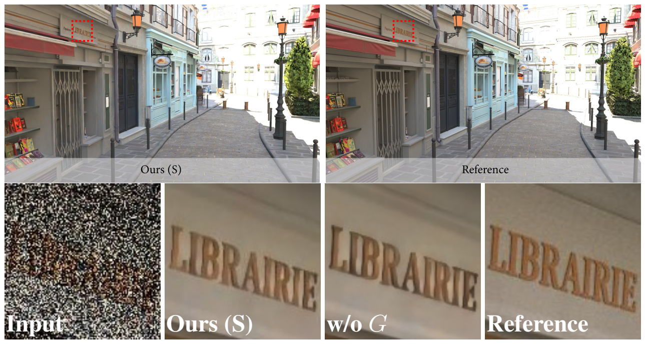

Figure 4: **Ablation on the local-to-global knowledge distillation module.** Our complete framework is compared with training the student model only using our local per-pixel loss, without the regularization from GAN, on the ***Bistro*** scene.

our student model tends to overfit the training dataset and exhibits poor generalization. Visual comparison can be found in Figure 4, where training our student model solely with local pixel-wise loss, the edges in the results can be blurry as the student learns inadequately from the teacher.

Our render-aware knowledge initialization aims to provide a better initialization for the student. When using a common initialization method such as Xavier, our student may converge to a suboptimal position. Our proposed render-aware knowledge initialization strategy requires a similar backbone for the teacher and the student. It is elegant yet useful according to the results.

Our auxiliary unlabeled dataset is crucial for the training of our student model. Training on the unlabeled dataset emphasizes more on the denoising results of our teacher model, which is necessary for the KD process. We argue that the teacher's denoising performance plays an important role, and we should refrain from using an unlabeled dataset to avoid potential side effects in scenes where the teacher struggles to denoise effectively.

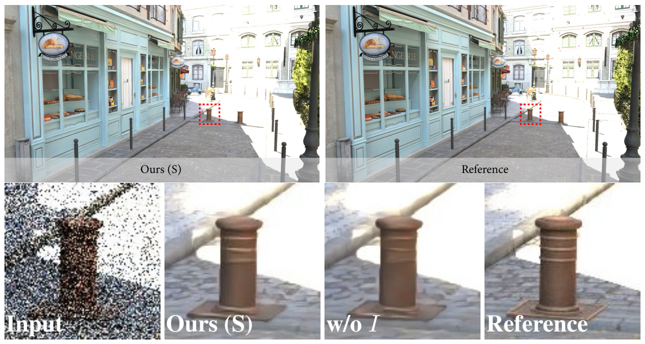

Figure 5: **Ablation on render-aware knowledge initialization.** Our complete framework is compared with training the student model only using Xavier initialization, without render-aware knowledge initialization, on ***Bistro*** scene.

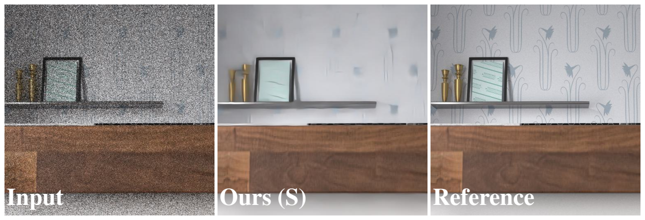

Figure 6: **Failure case.** Our method may fail when facing scenes featuring specular reflection. Here, we see the scene rendered through a mirror.

Limitations

Our method has several limitations. Firstly, the proposed denoiser heavily relies on primary ray intersections, and its performance may decline when facing specular objects, as shown in Fig. 6. A possible solution would include splitting diffuse and specular radiance in the input (Bako et al. 2017). Secondly, while we use temporal loss for temporal stability, our dataset consists of 10-frame sequences, shorter than recent works that use 64-frame sequences (Balint et al. 2023). Access to a broader range of datasets with longer sequences is useful for further improving temporal stability.

Conclusion

In conclusion, we have proposed the first knowledge distillation framework, named RAKD, for real-time Monte Carlo denoising. Considering the dense and structural rendering contents, several special designs are incorporated into this framework: a local-to-global knowledge distillation module, a render-aware knowledge initialization strategy, and an auxiliary unlabeled dataset. Aided by this framework, we can distill complex render information both on pixel level and image level. Attributing to the small size of the student model, we achieve real-time performance on neural denoising and succeed in retaining high-frequency details as compared with previous work. Our proposed knowledge distillation framework is not restricted to a particular denoising backbone and holds promise for future applications in real-time denoising efforts.

References

Áfra, A. T. 2024. Intel® Open Image Denoise. https://www.openimagedenoise.org.

Bako, S.; Vogels, T.; Mcwilliams, B.; Meyer, M.; NováK, J.; Harvill, A.; Sen, P.; Derose, T.; and Rousselle, F. 2017. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Trans. Graph.*, 36(4).

Balint, M.; Wolski, K.; Myszkowski, K.; Seidel, H.-P.; and Mantiuk, R. 2023. Neural Partitioning Pyramids for Denoising Monte Carlo Renderings. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701597.

Bitterli, B. 2016. Rendering resources. Https://benedikt-bitterli.me/resources/.

Burgess, J. 2020. RTX on the NVIDIA Turing GPU. *IEEE Micro*, 40(2): 36–44.

Chaitanya, C. R. A.; Kaplanyan, A. S.; Schied, C.; Salvi, M.; Lefohn, A.; Nowrouzezahrai, D.; and Aila, T. 2017. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)*, 36(4): 1–12.

Fan, H.; Wang, R.; Huo, Y.; and Bao, H. 2021. Real-time Monte Carlo Denoising with Weight Sharing Kernel Prediction Network. *Computer Graphics Forum*, 40(4): 15–27.

Fang, G.; Song, J.; Shen, C.; Wang, X.; Chen, D.; and Song, M. 2019. Data-Free Adversarial Distillation. *CoRR*, abs/1912.11006.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256. JMLR Workshop and Conference Proceedings.

Graves, A. 2012. *Long Short-Term Memory*, 37–45. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-24797-2.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30.

Harada, T. 2020. Hardware-Accelerated Ray Tracing in AMD Radeon ProRender 2.0. https://gpuopen.com/learn/radeon-prorender-2-0/.

Hasselgren, J.; Munkberg, J.; Salvi, M.; Patney, A.; and Lefohn, A. 2020. Neural Temporal Adaptive Sampling and Denoising. *Computer Graphics Forum*, 39(2): 147–155.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531.

Hu, J.; Fan, C.; Jiang, H.; Guo, X.; Gao, Y.; Lu, X.; and Lam, T. L. 2023. Boosting lightweight depth estimation via knowledge distillation. In *International Conference on Knowledge Science, Engineering and Management*, 27–39. Springer.

Huo, Y.; and Yoon, S. 2021. A survey on deep learning-based Monte Carlo denoising. *Comp. Visual Media*, 7: 169–185.

Işık, M.; Mullia, K.; Fisher, M.; Eisenmann, J.; and Gharbi, M. 2021. Interactive Monte Carlo denoising using affinity of neural features. *ACM Trans. Graph.*, 40(4).

Kallweit, S.; Clarberg, P.; Kolb, C.; Davidovič, T.; Yao, K.-H.; Foley, T.; He, Y.; Wu, L.; Chen, L.; Akenine-Möller, T.; Wyman, C.; Crassin, C.; and Benty, N. 2022. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lin, Y.; Li, Y.; Wang, Z.; Li, B.; Du, Q.; Xiao, T.; and Zhu, J. 2020. Weight distillation: Transferring the knowledge in neural network parameters. *arXiv preprint arXiv:2009.09152*.

Liu, Y.; Chen, K.; Liu, C.; Qin, Z.; Luo, Z.; and Wang, J. 2019. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2604–2613.

Liu, Y.; Shu, C.; Wang, J.; and Shen, C. 2020. Structured knowledge distillation for dense prediction. *IEEE transactions on pattern analysis and machine intelligence*, 45(6): 7035–7049.

Lu, Y.; Fu, S.; Zhang, X. H.; et al. 2021. Denoising Monte Carlo renderings via a multi-scale featured dual-residual GAN. *Vis Comput*, 37: 2513–2525.

Lu, Y.; Xie, N.; and Shen, H. T. 2020. DMCR-GAN: Adversarial Denoising for Monte Carlo Renderings with Residual Attention Networks and Hierarchical Features Modulation of Auxiliary Buffers. In *SIGGRAPH Asia 2020 Technical Communications*, SA '20. New York, NY, USA: Association for Computing Machinery. ISBN 9781450380805.

Lumberyard, A. 2017. Amazon Lumberyard Bistro, Open Research Content Archive (ORCA). http://developer.nvidia.com/orca/amazon-lumberyard-bistro.

Meng, X.; Zheng, Q.; Varshney, A.; Singh, G.; and Zwicker, M. 2020. Real-time Monte Carlo Denoising with the Neural Bilateral Grid. In Dachsbacher, C.; and Pharr, M., eds., *Eurographics Symposium on Rendering - DL-only Track*. London, UK: The Eurographics Association. ISBN 978-3-03868-117-5.

Micaelli, P.; and Storkey, A. J. 2019. Zero-shot Knowledge Transfer via Adversarial Belief Matching. *CoRR*, abs/1905.09768.

Munkberg, J.; and Hasselgren, J. 2020. Neural Denoising with Layer Embeddings. volume 39, 1–12.

Nvidia. 2022a. NVIDIA Real-Time Denoisers. https://developer.nvidia.com/rtx/ray-tracing/rt-denoisers.

Nvidia. 2022b. ReLAX: A Denoiser Tailored to Work with the Re-STIR Algorithm.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch.

Pharr, M.; Jakob, W.; and Humphreys, G. 2016. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 3rd edition.

Reinhard, E.; Stark, M.; Shirley, P.; and Ferwerda, J. 2023. *Photographic tone reproduction for digital images*, 661–670. Association for Computing Machinery.

Sandy, M.; Andersson, J.; and Barré-Brisebois, C. 2018. DirectX: Evolving Microsoft's Graphics Platform. Game Developers Conference 2018.

Schied, C.; Kaplanyan, A.; Wyman, C.; Patney, A.; Chaitanya, C. R. A.; Burgess, J.; Liu, S.; Dachsbacher, C.; Lefohn, A.; and Salvi, M. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*, HPG '17. New York, NY, USA: Association for Computing Machinery. ISBN 9781450351010.

Shu, C.; Liu, Y.; Gao, J.; Yan, Z.; and Shen, C. 2021. Channel-wise knowledge distillation for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5311–5320.

Sun, C.; Shrivastava, A.; Singh, S.; and Gupta, A. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, 843–852.

Thomas, M. M.; Liktor, G.; Peters, C.; Kim, S.; Vaidyanathan, K.; and Forbes, A. G. 2022. Temporally Stable Real-Time Joint Neural Denoising and Supersampling. *Proc. ACM Comput. Graph. Interact. Tech.*, 5(3): Article 21.

Vogels, T.; Rousselle, F.; McWilliams, B.; Röthlin, G.; Harvill, A.; Adler, D.; Meyer, M.; and Novák, J. 2018. Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (TOG)*, 37(4): 1–15.

Wang, X.; Weissweiler, L.; Schütze, H.; and Plank, B. 2023. How to Distill your BERT: An Empirical Study on the Impact of Weight Initialisation and Distillation Objectives. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 1843–1852. Toronto, Canada: Association for Computational Linguistics.

Wei, X.; Huang, H.; Shi, Y.; Yuan, H.; Shen, L.; and Wang, J. 2021. End-to-End Adaptive Monte Carlo Denoising and Super-Resolution. *CoRR*, abs/2108.06915.

Winkelmann, M. 2019. Zero-Day, Open Research Content Archive (ORCA). https://developer.nvidia.com/orca/beeple-zero-day.

Xu, B.; Zhang, J.; Wang, R.; Xu, K.; Yang, Y.-L.; Li, C.; and Tang, R. 2019. Adversarial Monte Carlo denoising with conditioned auxiliary feature modulation. *ACM Trans. Graph.*, 38(6).

Young, L. D.; Reda, F. A.; Ranjan, R.; Morton, J.; Hu, J.; Ling, Y.; Xiang, X.; Liu, D.; and Chandra, V. 2021. Feature-Align Network with Knowledge Distillation for Efficient Denoising. arXiv:2103.01524.

Yu, J.; Nie, Y.; Long, C.; Xu, W.; Zhang, Q.; and Li, G. 2021. Monte Carlo Denoising via Auxiliary Feature Guided Self-Attention. *ACM Transactions on Graphics*, 40(6): 13.

Zhang, H.; Goodfellow, I.; Metaxas, D.; and Odena, A. 2019. Self-attention generative adversarial networks. In *International conference on machine learning*, 7354–7363. PMLR.

Zhdan, D. 2021. *ReBLUR: A Hierarchical Recurrent Denoiser*. Berkeley, CA: Apress.

Zou, B.; Zhang, Y.; Wang, M.; and Liu, S. 2023. Toward Efficient Image Denoising: A Lightweight Network with Retargeting Supervision Driven Knowledge Distillation. In *Advances in Computer Graphics: 39th Computer Graphics International Conference, CGI 2022, Virtual Event, September 12–16, 2022, Proceedings*, 15–27. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-031-23472-9.

Zwicker, M.; et al. 2015. Traditional Denoising Methods in Rendering. *ACM Transactions on Graphics*, 34(4): 1–10.