High-Quality Real-Time Rendering Using Subpixel Sampling Reconstruction

Boyu Zhang[1, 3], **Hongliang Yuan**[2, 3*]
[1]University of California, Los Angeles
[2]Xiaomi Cooperation
[3]Tencent AI Lab
bobo8496@ucla.edu, hercules.yuan@gmail.com

Abstract

Generating high-quality, realistic rendering images for real-time applications generally requires tracing a few samples-per-pixel (spp) and using deep learning-based approaches to denoise the resulting low-spp images. Existing denoising methods necessitate a substantial time expenditure when rendering at high resolutions due to the physically-based sampling and network inference time burdens. In this paper, we propose a novel Monte Carlo sampling strategy to accelerate the sampling process and a corresponding denoiser, subpixel sampling reconstruction (SSR), to obtain high-quality images. Extensive experiments demonstrate that our method significantly outperforms previous approaches in denoising quality and reduces overall time costs, enabling real-time rendering capabilities at 2K resolution.

Introduction

Rendering realistic images for virtual worlds is a key objective in many computer vision and graphics tasks (Huo and Yoon 2021; Xu et al. 2022; Huang et al. 2023; Li et al. 2023; Li, Ngo, and Nagahara 2023), with applications in animation production (Dahlberg, Adler, and Newlin 2019), VR/AR world generation (Overbeck et al. 2018), virtual dataset synthesis (Ge et al. 2022), etc. One widely used technique for this purpose is Monte Carlo (MC) sampling (Seila 1982), which is highly versatile but typically requires a large number of samples to achieve accurate results. Despite the relentless advancements in computational capabilities, the temporal expenditure for executing realistic rendering remains a practical constraint, with high-quality images often taking hours to generate. Using low samples-per-pixel (spp) can speed up this process but lead to visually distracting noise. To mitigate this issue, post-processing techniques have been developed, known as MC denoising, which normally have lower time costs than physically-based renderers and are widely used in modern game engines(Chaitanya et al. 2017; NVIDIA 2021; Xiao et al. 2020).

Most existing MC denoising methods (Edelsten, Jukarainen, and Patney 2019; Xiao et al. 2020; Chaitanya et al. 2017; Işık et al. 2021; Meng et al. 2020; Hasselgren et al. 2020; Fan et al. 2021) employ deep learning-based approaches to remove noise from images generated with more than 1-spp. While (Chaitanya et al. 2017; Meng et al. 2020; Fan et al. 2021; Thomas et al. 2022) attempt to develop methods to accelerate the overall process by working with low-sample data, they have yet to achieve real-time frame rates at high resolutions, as 1-spp remains time-consuming. Other approaches (Edelsten, Jukarainen, and Patney 2019; Xiao et al. 2020; Işık et al. 2021) focus on designing more efficient post-processing modules in the image space to handle noisy images, but they tend to produce aliased rendered pixels at low-sample images. Additionally, the complex network structures of these works impose heavy burdens on inference time.

To achieve real-time performance for the generation of high-resolution, realistic images, we introduce a novel MC sampling strategy, subpixel sampling. This strategy is designed to curtail the temporal demands of physically-based rendering. Complementing this, we also propose a denoising method subpixel sampling reconstruction (SSR), which is tailored to the subpixel sampling strategy.

Subpixel sampling. The subpixel sampling strategy generates images with less than 1-spp. To obtain this, we divide each frame at the target resolution into consecutive, non-overlapping tiles with size 2×2 and then compute only one ray-traced pixel per tile (we refer to it as 1/4-spp). This strategy allows us to use these reliable samples to interpolate the missing pixels with the GBuffers (OpenGL 1998) at the target resolution. We developed a vulkan-based (Sellers and Kessenich 2016) hybrid ray tracer to export datasets. By utilizing subpixel sampling, the cost of rendering time can be reduced by a third.

Subpixel sampling reconstruction. Our reconstruction contains two parts: a temporal feature accumulator and a reconstruction network. The former warps previous frames to align with the current frame at the target resolution and accumulates subpixel samples and GBuffers from the previous frame based on the temporal accumulation factor, which is computed according to the correlation of the current and previous frames, effectively expanding the perception field of pixels. Once subpixel samples are collected, we move on to the second component, our reconstruction network. This is a multi-scale U-Net (Ronneberger, Fischer, and Brox 2015) with skip connections, which enables us to reconstruct the desired high-resolution image.

*Corresponding author.

The key points of our contribution can be summarized as follows:

- We propose a novel Monte Carlo sampling strategy termed as subpixel sampling, which significantly curtails the sampling time required for physically-based rendering to one-third.
- We introduce a denoising network, SSR, to reconstruct high-quality image sequences at real-time frame rates from rendering outcomes utilizing the subpixel sampling strategy.
- Our model yields superior results compared to existing state-of-the-art approaches and achieves real-time reconstruction performance of 2K resolution with 130 FPS.
- A realistic synthesised dataset is built through our subpixel sampling ray tracer. We will release the dataset and code for research purpose.

Related Work

Monte Carlo Denoising

Monte Carlo (MC) denoising techniques are extensively applied in the realm of rendering realistic images. Traditional best-performing MC denoisers were mainly based on local neighborhood regression models (Zwicker et al. 2015), includes zero-order regression (Rousselle, Knaus, and Zwicker 2012; Delbracio et al. 2014; Li, Wu, and Chuang 2012; Kalantari, Bako, and Sen 2015; Rousselle, Manzi, and Zwicker 2013; Moon et al. 2013), first-order regression (Bauszat, Eisemann, and Magnor 2011; Bitterli et al. 2016; Moon, Carr, and Yoon 2014) and even higher-order regression models (Moon et al. 2016).The filtering-based methods are based on using the auxiliary feature buffers to guide the construction of image-space filters. Most of the above methods run in offline rendering. To increase the effective sample count, real-time denoisers leverage temporal accumulation between frames over time to amortize supersampling (Yang et al. 2009), i.e. temporal anti-aliasing (TAA). The previous frame is reprojected according to the motion vector and blended with the current frame using a temporal accumulation factor, which can be constant (Schied et al. 2017; Mara et al. 2017; Meng et al. 2020) or changed (Schied, Peters, and Dachsbacher 2018) across different frames. The fixed temporal accumulation factor inevitably leads to ghosting and temporal lag. By adaptively setting the parameters, the temporal filter can rapidly adapt to temporal variations, efficiently responding to abrupt frame-to-frame changes. Yang et al. (Yang, Liu, and Salvi 2020) survey recent TAA techniques and provide an in-depth analysis of the image quality trade-offs with these heuristics. Koskela et al. (Koskela et al. 2019) propose a blockwise regression for real-time path tracing reconstruction and also do accumulation to improve temporal stability.

Deep Learning-Based Denoising

Recently, in the wake of advancements in powerful modern GPUs, numerous studies have leveraged CNN to construct MC denoisers. (Bako et al. 2017; Vogels et al. 2018) use deep CNN to estimate the local per-pixel filtering kernels used to compute each denoised pixel from its neighbors. Layer-based denoiser (Munkberg and Hasselgren 2020) designs a hierarchical kernel prediction for multi-resolution denoising and reconstruction. Owing to the substantial burdens of predicting large filtering kernels, these methods mostly target offline renderings. There are also other methods (Kuznetsov, Khademi Kalantari, and Ramamoorthi 2018; Xu et al. 2019; Gharbi et al. 2019; Yu et al. 2021; Back et al. 2022) that target denoising at more than 4 spp. To reduce the overhead of kernel prediction, Fan et al. (Fan et al. 2021) predict an encoding of the kernel map, followed by a high-efficiency decoder to construct the complete kernel map. Chaitanya et al. (Chaitanya et al. 2017) propose a recurrent connection based on U-Net (Ronneberger, Fischer, and Brox 2015) to improve temporal stability. Hasselgren et al. (Hasselgren et al. 2020) introduce a neural spatio-temporal joint optimization of adaptive sampling and denoising with a recurrent feedback loop. Hofmann et al. (Hofmann et al. 2021) also utilize the neural temporal adaptive sampling architecture to denoise rendering results with participating media. Xiao et al. (Xiao et al. 2020) presente a neural supersampling method for TAA, which is similar to deep-learned supersampling (DLSS) (Edelsten, Jukarainen, and Patney 2019). Meng et al. (Meng et al. 2020) denoise 1-spp noisy input images with a neural bilateral grid at real-time frame rates. Mustafa et al. (Işık et al. 2021; Thomas et al. 2022) adopte spatial kernels to filter the noisy image guiding by features. (Firmino, Frisvad, and Jensen 2023) designe adaptive sampling for optimizing MC denoising. (Balint et al. 2023) employe pyramid filters to recover renderings. Compared with these denoising frameworks targeting more than 1-spp, our approach is tailored to operate efficiently with 1/4-spp, cutting off rendering time expenditure.

Method

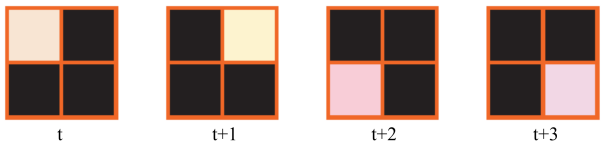
Subpixel Sampling

To mitigate the substantial computational cost associated with rendering in cases where the samples-per-pixel (spp) exceeds 1, we devise subpixel sampling that empowers us to produce images with 1/4-spp.

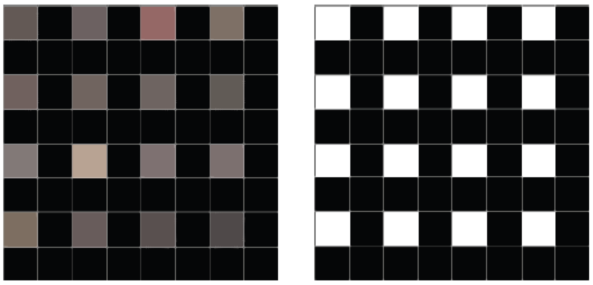
1/4-spp pattern Our strategy involves dividing each frame into non-overlapping 2×2 tiles and applying MC ray tracing methods to solve the rendering equation (Kajiya 1986) for one pixel in each tile. We term this process 1/4-spp pattern. To maintain data balance, we shift the sampling position to ensure that each pixel is sampled in the consecutive four frames at time steps t to $t + 3$, as illustrated in Fig. 1a.

GBuffers We leverage the rasterization pipeline to efficiently produce high-resolution GBuffers. In detail, we dump 1/4-spp RGB color $\mathbf{c} \in \mathbb{R}^3$ (Fig. 2a) and features $\mathbf{f} \in \mathbb{R}^{15}$. These features comprise four 3D vectors (albedo, normal, shadow, and transparent) and three 1D vectors (depth, metallic, and roughness), as shown in Figs. 2b to 2h.

Mask map As the sampled subpixels undergo ray tracing at a high resolution, their RGB values are reliable for the target resolution. In this context, we generate an additional mask map to denote reliable pixels. This map distinctly assigns a



(a) Subpixel sampling strategy



(b) Pixels (c) Mask map

Figure 1: (a) Sampling of a 2×2 tile from consecutive four frames. The sampled and unsampled pixels are drawn in color and in black (with value 0), respectively. (b) Pixels of a sub-patch example in a rendered image. (c) The corresponding mask map of patch (b) is depicted in white pixels with a value of 1, while black pixels indicate a value of 0.

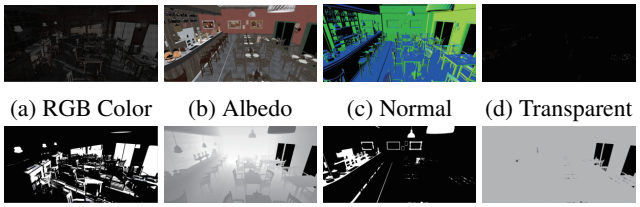
value of 1 to sampled positions and 0 to unsampled positions, as shown in Fig. 1c. It performs as a confidence map and is expected to guide our temporal feature accumulator to predict reasonable weights. To this end, we incorporate the mask map into the GBuffers.

Demodulation Similar to the previous approach (Chaitanya et al. 2017), we utilize the albedo (or base color) to demodulate the RGB image. Then, the resulting untextured irradiance x is transformed into log space using the natural logarithm function, i.e., $\ln(1 + x)$. However, our method differs in that once the untextured irradiance has been reconstructed, we re-modulate it using the accumulated albedo predicted by our temporal feature accumulator.

Subpixel Sampling Reconstruction

We designed subpixel sampling reconstruction (SSR) to recover temporally stable video from 1/4-spp image sequences at real-time frame rates. Fig. 3 shows the detailed architecture of SSR, which comprises two modules: the temporal feature accumulator (in green) and the reconstruction network (in blue).

Temporal Feature Accumulator The temporal feature accumulator module consists of two neural networks, each with two convolution layers that have a spatial support of 3×3 pixels. One network receives all features and mask of current frame as input and outputs reference embedding. The other computes embeddings for the current features $\boldsymbol{f}_t$ and warped previous features $\boldsymbol{f}_{t-1}$. These two embeddings are then pixel-wise multiplied to the reference embedding and then through softmax$(\cdot)$ to get α and β ($\alpha + \beta = 1$) blending factors for current features and previous features,



(a) RGB Color (b) Albedo (c) Normal (d) Transparent

(e) Shadow (f) Depth (g) Metallic (h) Roughness

Figure 2: Dumped buffers from our ray tracer.

respectively.

All features in Fig. 2 are accumulated through above process. Take untextured irradiance as an example, as illustrated in Fig. 4, we use the following equation to accumulate untextured irradiance $\mathbf{e}$ over the frame:

$$\mathbf{e}_t^a = \alpha \mathcal{W}(\mathbf{e}_{t-1}^a) + \beta \mathbf{e}_t, \tag{1}$$

where $\mathbf{e}_t^a$ is accumulated irradiance until t frame, $\mathbf{e}_t$ is irradiance for t frame. For the first frame, we set $\mathbf{e}_{t-1}^a$ to $\mathbf{e}_t$. $\mathcal{W}(\cdot)$ is a warping operator that reprojects previous frame to current one using motion vector.

The temporal feature accumulator serves a vital role in producing temporally stable results. Firstly, it can detect and remove disoccluded pixels and ghosting artifacts that traditional motion vectors cannot handle accurately. Secondly, since our input images are sparsely sampled, this module helps gather more finely sampled pixels across frames.

Reconstruction Network Our reconstruction network extends U-Net (Ronneberger, Fischer, and Brox 2015) with skip connections (Mao, Shen, and Yang 2016). In contrast to other U-Net-based denoising methods (Chaitanya et al. 2017), our approach predicts two coarse-scale images at the first two decoder stages rather than predicting dense features at these stages. This modification not only leads to faster inference but also results in high-quality images with superior quantitative metrics (see network ablation).

To generate a high-quality image for the current frame, we concatenate the current and accumulated features and feed them into our reconstruction network. Additionally, we input the warped denoised image from the previous frame, which enhances the temporal stability of image sequences (see network ablation). The reconstruction network consists of three encoder layers that produce three scale features.

Retaining temporal feedback at multiple scales is also a crucial step. To achieve this, we downsample the warped denoised image from the previous frame using a pool with a stride of two and pass it to each encoding stage. At the decoder stage, we concatenate the features and the warped denoised image at the same scale and feed them into a tile with two convolution layers. At the first two decoder stages, the image in RGB space is produced and upsampled. This upsampled image is then passed to the next decoder stage. The multi-scale feedback enables our network to own a sufficiently large temporal receptive field and efficiently generate high-quality, temporally stable results.

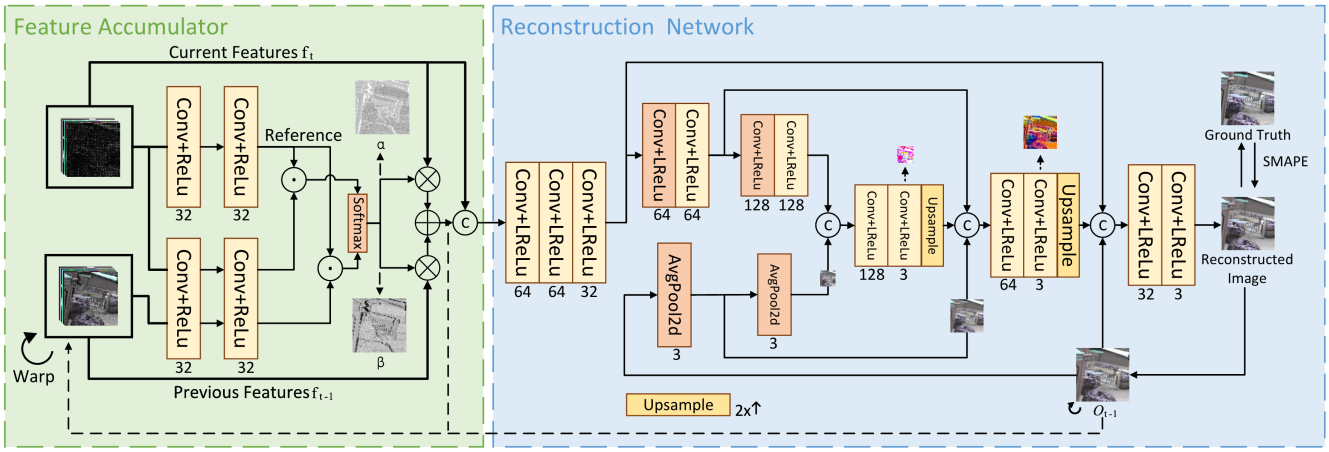


Figure 3: Subpixel sampling reconstruction consists of two modules: the temporal feature accumulator (left) and the reconstruction network (right) . The numbers under each network layer represent the output channels at corresponding layers. The operator $\odot$ denotes dot product between features. © indicates concatenation operation. $\oplus$ and $\otimes$ represent element-wise addition and multiplication, respectively. Note that all frames shown here are demodulated by albedo.

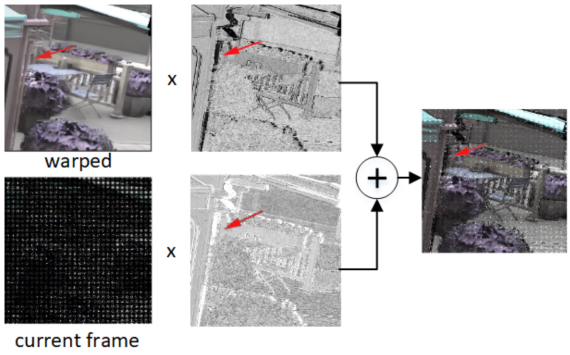
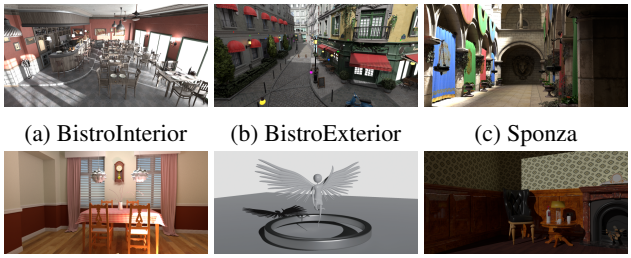


Figure 4: Illustration of accumulating untextured irradiance by our temporal feature accumulator. Warped irradiance by motion vector has ghosting artifacts (red arrow), which can be removed by giving lower weight in these areas.



(a) BistroInterior (b) BistroExterior (c) Sponza

(d) Diningroom (e) Angel (f) Warmroom

Figure 5: An overview of our generated dataset.

Loss

We use the symmetric mean absolute percentage error (SMAPE):

$$\ell(\mathbf{r},\mathbf{d}) = \frac{1}{3N}\sum_{p=1}^{p=N}\sum_{c=1}^{c=3}\frac{|\mathbf{d}_{p,c}-\mathbf{r}_{p,c}|}{|\mathbf{d}_{p,c}|+|\mathbf{r}_{p,c}|+\varepsilon}, \tag{2}$$

where N is the number of pixels and ε is a tiny perturbation, $\mathbf{d}$ and $\mathbf{r}$ are the denoised frame and the corresponding reference frame, respectively.

Our loss combines two parts, the first one is computed on a sequence of 5 continuous frames, including spatial loss $\ell_s = \ell(\mathbf{r},\mathbf{d})$, temporal loss $\ell_t = \ell(\Delta\mathbf{r},\Delta\mathbf{d})$ where Δ is temporal gradient computed between two consecutive frames, relative edge loss $\ell_e = L_1(\frac{\nabla\mathbf{d}}{\mathbf{r}+\varepsilon},\frac{\nabla\mathbf{r}}{\mathbf{r}+\varepsilon})$, where gradient ∇ is computed using a High Frequency Error Norm (HFEN), an image comparison metric from medical imaging (Ravishankar and Bresler 2011). As suggested by Chaitanya et al. (Chaitanya et al. 2017), we assign higher weight to three loss functions (ℓ_s, ℓ_t and ℓ_e) of frames later in the sequence to amplify temporal gradients. For a training sequence of 5 images, we use (0.05, 0.25, 0.5, 0.75, 1). The second part is warped temporal loss $\ell_{wt} = \ell(\omega\mathbf{r},\omega\mathbf{d})$ where $\omega\mathbf{r} = r_4 - \mathcal{W}(r_3)$, $\mathcal{W}(\cdot)$ is a warping operator that reprojects previous frame to current one. We also include albedo loss $\ell_a = \ell(\mathbf{a}_{acc},\mathbf{a}_r)$. $\mathbf{a}_{acc}$ is accumulated albedo computed by our temporal feature accumulator. We only compute albedo loss on the last frame and warped temporal loss on the last two frames.

We use a weighted combination of these losses as the overall loss:

$$\ell = \lambda_s\ell_s + \lambda_t\ell_t + \lambda_e\ell_e + \lambda_w\ell_{wt} + \lambda_a\ell_a. \tag{3}$$

Experiments

Datasets and Metrics

Datasets As subpixel sampling is a novel strategy, the field currently lacks dedicated datasets specifically designed for this purpose. We utilized a vulkan-based (Sellers and

(a) Ours (b) Input (c) AFS (d) ANF (e) NSRR (f) RAE (g) SSR (h) Ref

Figure 6: Visual results on scenes BistroInterior, BistroExterior and Sponza.

Kessenich 2016) hybrid ray tracer to generate our subpixel sampling dataset. To optimize our approach for application in games and advanced virtual rendering, we conducted distinct training sessions for each 3D scene instead of collective training. This approach is in concordance with the paradigm utilized in NVIDIA DLSS (NVIDIA 2021). Since our input images were generated at 1/4-spp, a large number of images were imperative to train a robust denoiser. The training process was carried out across six scenes, see Fig. 5. The BistroInterior and BistroExterior (Lumberyard 2017) scenes contain more than one million triangles and transparency, diffuse, specular, and soft shadow effects. All scenes contain 100 to 1000 frames with a resolution of 1024×2048. We also rendered a validation set of 10 frames and a 50 frames test set for each scene. The ground truth image is rendered at 32768-spp for reference.

Metrics All comparison approaches are evaluated by three image quality metrics: peak signal to noise ratio (PSNR), structural similarity index (SSIM) (Wang et al. 2004), and root mean squared error (RMSE). Higher PSNR and SSIM imply superior performance, while lower RMSE indicates better.

Implementation Details

We randomly selected 5 consecutive frames for training each scene. To maximize the utilization of the GPUs, we also randomly cropped the inputs, including the noisy image and auxiliary features, to a resolution of 256x256. The kernel size is 3×3 at all layers. The weight coefficients for $\mathcal{L}_s$, $\mathcal{L}_t$, $\mathcal{L}_e$, $\mathcal{L}_w$, and $\mathcal{L}_a$ are 0.7, 0.1, 0.2, 0.4, and 5.0, respectively. We conducted all experiments using the PyTorch framework (Paszke et al. 2019) on 8 NVIDIA Tesla A100 GPUs. Adam optimizer (Kingma and Ba 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$ is used with the initial learning rate set to 1×10^{-4}. The learning rate is halved at one-third and two-thirds of the total number of iterations. We set batch size to 8 and trained our model for 200 epochs. Each scene required approximately 9 hours of training time.

We compare our method to several cutting-edge Monte Carlo denoising and reconstruction techniques, including the fastest-running method RAE (Chaitanya et al. 2017), ANF (Işık et al. 2021), which achieves the best denoising performance on more than 1-spp images, the offline method AFS (Yu et al. 2021), and the super-resolution approach NSRR (Xiao et al. 2020). Notably, while NSRR is primarily designed for super-resolution, it demonstrates adaptability for the sparse sampling task, as elucidated in the supplementary materials. Meanwhile, its practical applications in 3A game rendering further establish NSRR as a pertinent benchmark in our evaluation. We replicated all the methods using their default settings.

Time Analysis

Rendering To showcase the efficiency of our subpixel sampling, we test the rendering time of each stage on the NVIDIA RTX 3090 GPU at a resolution of 1024×2048, see Tab. 2. The subpixel sampling strategy significantly reduced the sampling time from 12.79ms to 4.35ms, resulting

Method	BistroInterior		BistroExterior		Sponza		Diningroom		Warmroom		Angel		Ave	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
AFS	22.86	.7650	24.60	.8071	25.50	.8119	25.41	.8637	29.55	.8021	22.06	.8601	25.00	.8183
ANF	23.20	.7583	22.14	.7201	23.98	.8219	22.23	.7226	30.91	.8774	25.86	.8813	24.72	.7969
NSRR	23.87	.8104	25.54	.8538	24.93	.8113	27.17	.8843	36.40	.9740	34.94	.9804	28.81	.8857
RAE	24.03	.8351	24.11	.8006	27.74	.8898	29.87	.9007	34.32	.9675	29.18	.9161	28.21	.8849
SSR	**28.99**	**.8945**	**29.97**	**.9121**	**31.79**	**.9410**	**32.48**	**.9375**	**37.34**	**.9799**	**38.04**	**.9876**	**33.10**	**.9421**

Table 1: Quantitative comparison results on six scenes. We choose four baseline methods to compare with our SSR method. The best result is in bold, and the second-best is underlined in each column.

Strategy	R (ms)	T&S (ms)	Sampling (ms)	Overall(ms)
w-SS	0.85	1.72	4.35	**6.92**
w/o-SS	0.85	1.72	12.79	15.36

Table 2: Average rendering time of six scenes. R implies the rendering stage rasterization, and T&S stands for rendering transparent and shadow stage. w-SS denotes rendering with our subpixel sampling (1/4-spp), while w/o-SS means without it (1-spp).

Methods	1024×2048		1024×1080	
	Time (ms)	FPS	Time (ms)	FPS
AFS	41.8	24	25.6	39
ANF	33.0	30	19.8	51
NSRR	34.5	29	21.7	46
RAE	10.4	96	6.22	160
SSR	**7.6**	**131**	**4.56**	**220**

Table 3: Comparison results of inference time. Our SSR achieves 130 frames per second (FPS) at 2K resolution and 220 FPS at 1080p resolution.

in a 34% reduction in time. With the employment of subpixel sampling, the average total rendering time is 6.92 ms, compared to 15.36 ms without it, resulting in an approximate 3× improvement.

Reconstruction We also conducted an evaluation of the inference time for SSR and compared it against other methods. The comparison was carried out using the same frame for each scene, and the average results are presented in Tab. 3, which shows the average inference time for all six scenes at 1024 × 2048 and 1024 × 1080 resolution using an NVIDIA Tesla A100. Our SSR is capable of reaching a remarkable 130 FPS when operating at 2K resolution and 220 FPS at 1080p images. At both resolutions, SSR provides a frame rate improvement of approximately 37% compared to the previously fastest method.

Tab. 2 and Tab. 3 show the time of rendering and reconstruction respectively, while their combined cost is displayed in Fig. 7.

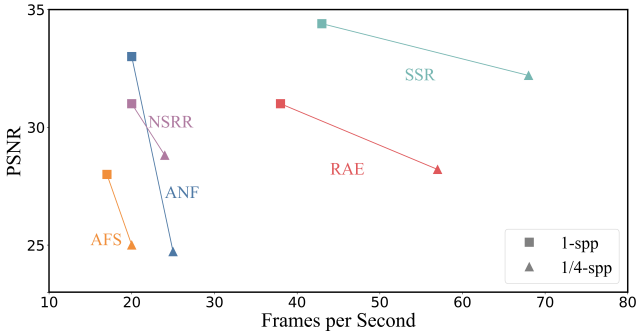


Figure 7: Speed-quality comparison on the 1-spp and 1/4-spp scenes at resolution 1024×2048, where higher PSNR and FPS (top right) is most desirable.

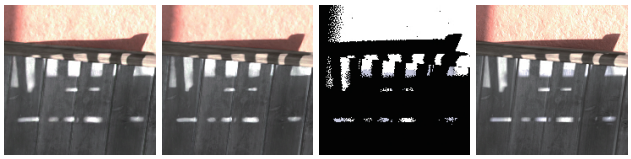
Quantitative Evaluation

Quantitative comparison results are shown in Tab. 1. Average results are reported on the 50 test videos of six scenes. Our method delivers the best performance in all scenes. We only show the results of PSNR and SSIM due to space limitations, and please refer to our supplemental material for more comparison results.

To show the improvements in speed and quality achieved by our method, we generated six scenes at 1-spp instead of using the subpixel sampling strategy, maintaining all other parameters identical to 1/4-spp scenes. We assessed the entire generation time, including both rendering and reconstruction, and reported the speed and quality comparisons in Fig. 7. SSR performs best on both 1-spp and 1/4-spp datasets, with tiny declines in quality performance as the sampling rate decreases (FPS ranges from 43 to 68 and PSNR varies from 34.40 to 33.10). In contrast, previous methods aimed at datasets larger than 1-spp exhibited dramatic performance degradation.

Qualitative Evaluation

Here we provide qualitative evaluations of our model. However, we encourage the reader to watch supplementary videos for a more comprehensive understanding. Fig. 6 compares reconstructed images in several scenes visually. We included all comparison results for six scenes in the supplementary material. Our method outperforms all other methods by a considerable margin across all scenes. Previous



(a) w/o-shadow (b) w-shadow (c) Noisy (d) Ref

Figure 8: (a) and (b) show the reconstruction results without and with employing shadow(c), respectively. (d) is the 32768-spp reference image. The shadow feature assists SSR in pinpointing more precise contours.

Method	RN	TFA	WP	PSNR/SSIM
Base	✓			23.55/.8152
Base+TFA	✓	✓		32.13/.9245
Base+TFA+WP	✓	✓	✓	**33.10/.9421**

Table 4: Ablation study. We evaluate different modules on six scenes. PSNR and SSIM are shown on average.

state-of-the-art methods, designed for denoising renderings with more than 1-spp, are not as effective at denoising renderings at 1/4-spp. AFS was originally designed for offline rendering, and transformer models (Vaswani et al. 2017; Liu et al. 2021) require significant memory to train and perform inference. RAE, NSRR, and ANF feed previous and current features directly into the network, which leads to blurred and aliased details. Different from them, SSR computes the correlation for each pixel between normal and depth features of the current and previous frames, thus having the capacity to generate high-quality details.

Ablation Study

GBuffers ablation We incorporated certain features from the Gbuffers that have not been utilized in existing Monte Carlo denoising methods and conducted corresponding ablation experiments to investigate their effectiveness.

Shadow. Our training images are generated by subpixel sampling. As a result of 1/4-spp light occlusion, more than three-quarters of the pixels remain at a value of zero, which motivates us to identify reliable pixels to train our model. Thus, we took the shadow feature as an additional input. Our feature accumulator collects the noisy shadows from the current frame and combines them with the history shadow. This accumulated shadow information aids in detecting continuous edges of shadows and improves temporal stability, as shown in Fig. 8.

Transparent. We also appended the transparent feature to SSR for training, but we do not accumulate transparent before feeding it into the reconstruction network. This is due to the transparent feature is scarce and contains rare noise in a whole image, as shown in Fig. 2d. Accumulating the transparent feature yields a minor improvement but also comes with an increased time cost. So we chose to feed the transparent feature into our reconstruction network directly. By utilizing transparent, SSR acquires the ability to produce

(a) w/o-T (b) w-T (c) Transparent (d) Ref

Figure 9: (a) and (b) show the reconstruction results without and with employing transparent (c), respectively. (d) is the 32768-spp reference image. SSR can capture the information from the transparent features and restore clear transparent objects.

transparent objects, such as clear glass cups, as illustrated in Fig. 9. Additionally, in cases where a scene does not contain any transparent objects, such as the BistroExterior scene, we included the transparent feature with a value of zero.

Without using shadow and transparency, SSR only achieves a PSNR of 27.67 when tested on BistroInterior, while employing shadow brings an improvement to 28.22. By including both shadow and transparency, our model produces a higher PSNR of 28.99.

Network ablation We verified the effectiveness of different modules in our approach, including the temporal feature accumulator (TFA) and the warped previous output (WP), as shown in Tab. 4. Results are presented as an average across six scenes. TFA demonstrates a noticeable enhancement, exhibiting a 36.4% increase in PSNR and a 13.4% improvement in SSIM. Similarly, the application of WP showcases its effectiveness in the third row.

Conclusion

We presented a novel Monte Carlo subpixel sampling strategy to facilitate accelerated rendering. Additionally, we proposed a denoising network, subpixel sampling reconstruction (SSR), to effectively restore high-quality image sequences in real-time from subpixel sampling pattern. Experiments substantiated that our approach yields superior denoised results in comparison to prevailing cutting-edge methods while also achieving real-time performance at 2K resolution.

Limitations and Future Work. While our method offers a real-time pattern for reconstructing high-quality images, there is still potential for enhancing the inference time. 16-bit precision TensorRT can be leveraged to expedite processing. We intend to deploy SSR within our game engine in the coming stages. Furthermore, we explored the integration of Swin Transformer (Liu et al. 2021) into the initial layer of our reconstruction network, resulting in a PSNR improvement of approximately 0.23 and a 1.1 ms increase in inference time. Striking the right balance between speed and quality remains a pivotal objective in our forthcoming research.

References

Back, J.; Hua, B.-S.; Hachisuka, T.; and Moon, B. 2022. Self-Supervised Post-Correction for Monte Carlo Denoising. In *Proceedings of ACM SIGGRAPH Conference (SIGGRAPH)*.

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 Transactions on Graphics (TOG)*, 36(4).

Balint, M.; Wolski, K.; Myszkowski, K.; Seidel, H.-P.; and Mantiuk, R. 2023. Neural Partitioning Pyramids for Denoising Monte Carlo Renderings. In *Proceedings of ACM SIGGRAPH Conference (SIGGRAPH)*.

Bauszat, P.; Eisemann, M.; and Magnor, M. 2011. Guided Image Filtering for Interactive High-Quality Global Illumination. In *Proceedings of the Eurographics Conference on Rendering (EG)*, 1361–1368. Goslar, DEU: Eurographics Association.

Bitterli, B.; Rousselle, F.; Moon, B.; Iglesias-Guitián, J. A.; Adler, D.; Mitchell, K.; Jarosz, W.; and Novák, J. 2016. Nonlinearly Weighted First-Order Regression for Denoising Monte Carlo Renderings. *Computer Graphics Forum (CGF)*, 35(4): 107–117.

Chaitanya, C.; Kaplanyan, A.; 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: 1–12.

Dahlberg, H.; Adler, D.; and Newlin, J. 2019. Machine-Learning Denoising in Feature Film Production. In *ACM SIGGRAPH 2019 Talks*. New York, NY, USA: Association for Computing Machinery. ISBN 9781450363174.

Delbracio, M.; Musé, P.; Chauvier, J.; Phelps, N.; and Morel, J.-M. 2014. Boosting Monte Carlo Rendering by Ray Histogram Fusion. *ACM Transactions on Graphics (TOG)*, 33.

Edelsten, A.; Jukarainen, P.; and Patney, A. 2019. Truly next-gen: Adding deep learning to games and graphics. *In NVIDIA Sponsored Sessions (Game Developers Conference)*.

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

Firmino, A.; Frisvad, J. R.; and Jensen, H. W. 2023. Denoising-Aware Adaptive Sampling for Monte Carlo Ray Tracing. In *Proceedings of ACM SIGGRAPH Conference (SIGGRAPH)*.

Ge, Y.; Behl, H.; Xu, J.; Gunasekar, S.; Joshi, N.; Song, Y.; Wang, X.; Itti, L.; and Vineet, V. 2022. Neural-Sim: Learning to Generate Training Data with NeRF. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 477–493. Springer.

Gharbi, M.; Li, T.-M.; Aittala, M.; Lehtinen, J.; and Durand, F. 2019. Sample-Based Monte Carlo Denoising Using a Kernel-Splatting Network. *ACM Transactions on Graphics (TOG)*.

Hasselgren, J.; Munkberg, J.; Salvi, M.; Patney, A.; and Lefohn, A. 2020. Neural temporal adaptive sampling and denoising. In *Computer Graphics Forum (CGF)*, volume 39, 147–155.

Hofmann, N.; Hasselgren, J.; Clarberg, P.; and Munkberg, J. 2021. Interactive Path Tracing and Reconstruction of Sparse Volumes. volume 4. New York, NY, USA: Association for Computing Machinery.

Huang, X.; Zhang, Y.; Ni, B.; Li, T.; Chen, K.; and Zhang, W. 2023. Boosting point clouds rendering via radiance mapping. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*.

Huo, Y.; and Yoon, S.-e. 2021. A survey on deep learning-based Monte Carlo denoising. *Computational 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 Transactions on Graphics (TOG)*.

Kajiya, J. T. 1986. The Rendering Equation. In *ACM on Computer Graphics and Interactive Techniques (CGIT)*.

Kalantari, N. K.; Bako, S.; and Sen, P. 2015. A Machine Learning Approach for Filtering Monte Carlo Noise. *ACM Transactions on Graphics (TOG)*, 34(4).

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *International Conference on Learning Representations (ICLR)*.

Koskela, M.; Immonen, K.; Mäkitalo, M.; Foi, A.; Viitanen, T.; Jääskeläinen, P.; Kultala, H.; and Takala, J. 2019. Blockwise Multi-Order Feature Regression for Real-Time Path-Tracing Reconstruction. volume 38. New York, NY, USA: Association for Computing Machinery.

Kuznetsov, A.; Khademi Kalantari, N.; and Ramamoorthi, R. 2018. Deep Adaptive Sampling for Low Sample Count Rendering. *Computer Graphics Forum (CGF)*, 37: 35–44.

Li, C.; Ngo, T. T.; and Nagahara, H. 2023. Inverse Rendering of Translucent Objects using Physical and Neural Renderers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12510–12520.

Li, T.-M.; Wu, Y.-T.; and Chuang, Y.-Y. 2012. SURE-Based Optimization for Adaptive Sampling and Reconstruction. *ACM Transactions on Graphics (TOG)*, 31(6).

Li, Z.; Wang, Q.; Cole, F.; Tucker, R.; and Snavely, N. 2023. Dynibar: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4273–4284.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *CoRR*, abs/2103.14030.

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

Mao, X.-J.; Shen, C.; and Yang, Y.-B. 2016. Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*.

Mara, M.; McGuire, M.; Bitterli, B.; and Jarosz, W. 2017. An Efficient Denoising Algorithm for Global Illumination. In *Proceedings of High Performance Graphics (HPG)*.

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*. The Eurographics Association. ISBN 978-3-03868-117-5.

Moon, B.; Carr, N.; and Yoon, S.-E. 2014. Adaptive Rendering Based on Weighted Local Regression. *ACM Transactions on Graphics (TOG)*, 33(5).

Moon, B.; Jun, J. Y.; Lee, J.; Kim, K.; Hachisuka, T.; and Yoon, S.-E. 2013. Robust Image Denoising Using a Virtual Flash Image for Monte Carlo Ray Tracing. *Computer Graphics Forum (CGF)*, 32(1): 139–151.

Moon, B.; McDonagh, S.; Mitchell, K.; and Gross, M. 2016. Adaptive Polynomial Rendering. *ACM Transactions on Graphics (TOG)*, 35(4).

Munkberg, J.; and Hasselgren, J. 2020. Neural Denoising with Layer Embeddings. *Computer Graphics Forum (CGF)*, 39(4): 1–12.

NVIDIA. 2021. Deep Learning Super Sampling (DLSS) Technology. https://www.nvidia.com/en-us/geforce/technologies/dlss/.

OpenGL. 1998. Deferred Shading. https://learnopengl.com/Advanced-Lighting/Deferred-Shading.

Overbeck, R. S.; Erickson, D.; Evangelakos, D.; and Debevec, P. 2018. The making of welcome to light fields VR. In *Proceedings of ACM SIGGRAPH 2018 Conference (SIGGRAPH)*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *CoRR*, abs/1912.01703.

Ravishankar, S.; and Bresler, Y. 2011. MR Image Reconstruction From Highly Undersampled k-Space Data by Dictionary Learning. *IEEE Transactions on Medical Imaging (TMI)*, 30(5): 1028–1041.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*, abs/1505.04597.

Rousselle, F.; Knaus, C.; and Zwicker, M. 2012. Adaptive Rendering with Non-Local Means Filtering. *ACM Transactions on Graphics (TOG)*, 31(6).

Rousselle, F.; Manzi, M.; and Zwicker, M. 2013. Robust Denoising using Feature and Color Information. *Computer Graphics Forum (CGF)*.

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)*.

Schied, C.; Peters, C.; and Dachsbacher, C. 2018. Gradient Estimation for Real-Time Adaptive Temporal Filtering. *Proc. ACM Comput. Graph. Interact. Tech.*, 1(2).

Seila, A. F. 1982. Simulation and the Monte Carlo method.

Sellers, G.; and Kessenich, J. 2016. *Vulkan programming guide: The official guide to learning vulkan*. Addison-Wesley Professional.

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. In *ACM on Computer Graphics and Interactive Techniques (CGIT)*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*.

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).

Wang, Z.; Bovik, A.; Sheikh, H.; and Simoncelli, E. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 13(4): 600–612.

Xiao, L.; Nouri, S.; Chapman, M.; Fix, A.; Lanman, D.; and Kaplanyan, A. 2020. Neural Supersampling for Real-Time Rendering. *ACM Transactions on Graphics (TOG)*, 39(4).

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 Transactions on Graphics (TOG)*, 38(6).

Xu, J.-P.; Zuo, C.; Zhang, F.-L.; and Wang, M. 2022. Rendering-aware hdr environment map prediction from a single image. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Yang, L.; Liu, S.; and Salvi, M. 2020. A Survey of Temporal Antialiasing Techniques. *Computer Graphics Forum (CGF)*, 39(2): 607–621.

Yang, L.; Nehab, D.; Sander, P. V.; Sitthi-amorn, P.; Lawrence, J.; and Hoppe, H. 2009. Amortized Supersampling. *ACM Transactions on Graphics (TOG)*, 28(5): 1–12.

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 (TOG)*.

Zwicker, M.; Jarosz, W.; Lehtinen, J.; Moon, B.; Ramamoorthi, R.; Rousselle, F.; Sen, P.; Soler, C.; and Yoon, S.-E. 2015. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum (CGF)*, 34(2): 667–681.