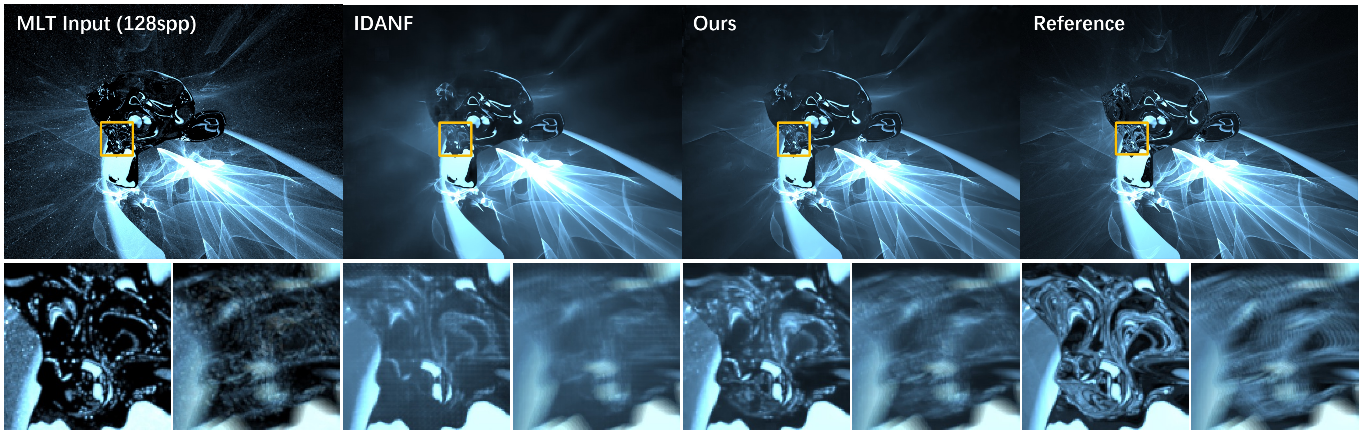
Temporally Stable Metropolis Light Transport Denoising using Recurrent Transformer Blocks

CHUHAO CHEN, University of California San Diego, USA
YUZE HE, Tsinghua University, China
TZU-MAO LI, University of California San Diego, USA



(a) Noisy Input (b) IDANF (c) Ours (d) Reference
PSNR: **27.94** SSIM: **0.837** FoVVDP: **8.820** PSNR: **29.25** SSIM: **0.857** FoVVDP: **8.968**

Fig. 1. The illustrated example is one middle frame from a 60-frame MLT animation of the scene *Monkey* [Ebke 2021]. The left-bottom crop is a detailed region of this frame and the right-bottom crop is an average over the 7 consecutive frames in this detailed region. In this example, the state-of-the-art sequence Monte Carlo denoisers (b) [Işık et al. 2021] suffers from large bias, excessive blur and square artifacts from dilated kernels, while our denoiser (c) produces high-quality results and keeps the temporal stability. All the metrics are calculated on the whole animation.

Metropolis Light Transport (MLT) is a global illumination algorithm that is well-known for rendering challenging scenes with intricate light paths. However, MLT methods tend to produce unpredictable correlation artifacts in images, which can introduce visual inconsistencies for animation rendering. This drawback also makes it challenging to denoise MLT renderings while maintaining temporal stability. We tackle this issue with modern learning-based methods and build a sequence denoiser combining the recurrent connections with the cutting-edge vision transformer architecture. We demonstrate that our sophisticated denoiser can consistently improve the quality and temporal stability of MLT renderings with difficult light paths. Our method is efficient and scalable for complex scene renderings that require high sample counts.

CCS Concepts: • **Computing methodologies** → **Ray tracing**; **Neural networks**; *Image processing*.

Additional Key Words and Phrases: Denoising, Metropolis Light Transport

Authors' addresses: Chuhao Chen, chc091@ucsd.edu, University of California San Diego, USA; Yuze He, hyz22@mails.tsinghua.edu.cn, Tsinghua University, China; Tzu-Mao Li, tzli@ucsd.edu, University of California San Diego, USA.

ACM 0730-0301/2024/7-ART123
https://doi.org/10.1145/3658218

ACM Reference Format:
Chuhao Chen, Yuze He, and Tzu-Mao Li. 2024. Temporally Stable Metropolis Light Transport Denoising using Recurrent Transformer Blocks. *ACM Trans. Graph.* 43, 4, Article 123 (July 2024), 14 pages. https://doi.org/10.1145/3658218

1 INTRODUCTION

Recent years have witnessed significant advancements in Monte Carlo Denoising methods, with the majority of efforts concentrated on denoising Monte Carlo Path Tracing (PT) renderings. Metropolis Light Transport (MLT) [Veach and Guibas 1997], however, has received less attention. Although proficient in rendering intricate light paths such as caustics or complex visibility, MLT suffers from correlation artifacts and flickering noise in video or animation rendering. The correlated samples from the Metropolis sampling make MLT incompatible with existing adaptive sampling and reconstruction methods for animation denoising [Mehta et al. 2012; Wu et al. 2017; Yan et al. 2015]. The difficulty in estimating MLT's variance [Ashikhmin et al. 2001] also makes it challenging to apply variance-guided spatiotemporal denoising approaches [Schied et al. 2017].

The emergence of deep learning offers the potential to denoise MLT renderings through neural networks. While these methods have successfully denoised PT renderings, their application to MLT remains limited. Interactive denoising approaches [Chaitanya et al. 2017; Fan et al. 2021; Hasselgren et al. 2020; Meng et al. 2020] prioritize temporal stability but are constrained by the scale of their neural networks for real-time denoising. Meanwhile, offline denoisers for single image denoising [Bako et al. 2017; Gharbi et al. 2019; Yu et al. 2021] employ larger neural networks to achieve higher quality reconstructions but often overlook the temporal stability necessary for MLT renderings. Balint et al. [2023] proposed a sequence denoiser with a large-scale network, which is the closest to meeting our needs. However, its CNN architecture and temporal accumulation techniques are still limited in MLT denoising.

In this paper, we introduce a novel learning-based denoiser adept at handling MLT renderings characterized by complex light paths while maintaining temporal stability. Distinguished from the previous offline multi-scale temporal kernel-predicting denoiser [Balint et al. 2023; Vogels et al. 2018], our denoiser is not based on temporal sliding windows or per-pixel input blending but operates as a recurrent model, enabling a more in-depth feature temporal accumulation. We also integrate the latest efficient vision transformer blocks [Zamir et al. 2022] in our model to enhance the model's perceptive field. Inspired by sample-based denoising methods [Gharbi et al. 2019; Zimmer et al. 2015], we propose a sample decomposition technique specific to MLT. To evaluate our method and compare it with other models in terms of MLT denoising, we construct a dataset with challenging scenes based on the OpenRooms framework [Li et al. 2021] to serve as a benchmark.

Our experiment's results validate our denoiser's efficacy in maintaining temporal stability and achieving superior reconstruction quality compared to existing denoisers for MLT renderings. Our contributions can be summarized by the following:

- A sample decomposition technique aims at removing flickering artifacts in MLT animation.
- A sophisticated denoiser with recurrent transformer blocks capable of high-quality denoising of MLT renderings while ensuring temporal stability.
- A new dataset with a large number of challenge scenes for evaluating MLT denoising.

2 RELATED WORK

2.1 Metropolis Light Transport

In rendering, we are interested in computing a path integral for each pixel j, that integrates over all the light paths $\bar{\mathbf{x}}$ that pass through the pixel [Veach 1998]:

$$I_j = \int_\Omega h_j(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}) \mathrm{d}\bar{\mathbf{x}}, \tag{1}$$

where Ω is the path space, h_j is the pixel filter, f is the measurement contribution function. When the scene contains difficult visibility or complex materials, only a small subset of the light paths $\bar{\mathbf{x}}$ will have high contribution f. This makes standard sampling inefficient since most light paths do not contribute significantly to the image.

Metropolis light transport (MLT) [Veach and Guibas 1997] renders difficult scenes with sparse contributions by reusing previously sampled high contribution light paths, using a Markov Chain Monte Carlo sampling method. MLT computes the path integrals for all pixels by generating a sequence of samples $\bar{\mathbf{x}}_i$. To generate a new sample in the sequence, MLT first generates a proposal by *mutating* the latest sample in the chain, usually by perturbing the light path or reusing part of it and regenerating the rest. It then probabilistically accepts or rejects the proposal based on a target function f^* – usually the luminance of the measurement contribution function. The mutation and probabilistic acceptance enable the reuse of rare, high contribution light paths, and ensure that the distribution of the samples $\bar{\mathbf{x}}_i$ would converge to a target distribution proportional to the target function f^*.

Given the samples, the path integrals are estimated as:

$$\langle I_j \rangle = \frac{b}{N} \sum_{i=1}^{N} \frac{h_j(\bar{\mathbf{x}}_i) f(\bar{\mathbf{x}}_i)}{f^*(\bar{\mathbf{x}}_i)}, \tag{2}$$

where b is an estimation of the average of f^* over the whole image.

While theoretical analysis has been conducted for the variance of MLT [Ashikhmin et al. 2001], it remains challenging to estimate per-pixel variance, since the Markov chain samples $\bar{\mathbf{x}}_i$ are correlated to each other due to the path mutation. MLT-rendered images are difficult to denoise using classical methods [Schied et al. 2017; Zwicker et al. 2015] that assume pixels are uncorrelated to each other and assume access to variance estimation for each pixel.

The quality of MLT rendering is highly dependent on the mutation strategies used to perturb the samples. Most of the MLT research focuses on improving the sampling efficiency by incorporating different mutation strategies, e.g., [Bashford-Rogers et al. 2021; Hachisuka et al. 2014; Jakob and Marschner 2012; Kelemen et al. 2002; Kitaoka et al. 2009; Luan et al. 2020] or improving sample stratification, e.g., [Bitterli and Jarosz 2019; Cline et al. 2005; Gruson et al. 2020; Zirr and Dachsbacher 2020]. Temporal mutation strategies [Lai et al. 2009; Van de Woestijne et al. 2017] have been proposed for rendering animation with MLT and improving temporal coherency. Our denoiser is, in principle, agnostic to the type of MLT variants used, but it may need to be retrained to learn the noise pattern of the specific MLT algorithm. In this paper, we focus on path-space MLT [Veach and Guibas 1997] to showcase the capability of our denoiser to handle highly-correlated samples.

2.2 Learning-Based Monte Carlo Denoising

Modern Monte Carlo rendering denoising often relies on deep learning. Offline denoising methods often target denoising for a single frame [Bako et al. 2017; Gharbi et al. 2019; Kalantari et al. 2015; Yu et al. 2021] and do not aim to improve temporal coherency.

Vogels et al. [2018] extends the kernel-predicting network [Bako et al. 2017] to denoise temporal sequences by taking multiple frames as input for each frame. Our architecture instead operates as a recurrent model to model a larger sequence.

Interactive and real-time denoisers [Chaitanya et al. 2017; Fan et al. 2021; Hasselgren et al. 2020; Meng et al. 2020; Thomas et al. 2022; Xiao et al. 2020] do model temporal coherency, but operate under limited computational budget and often have limited capacity for denoising challenging inputs produced by MLT. Balint et al. [2023] extends those interactive denoisers to an offline version, but it only increases the scale of the network while our method focuses on putting computational budget into temporal accumulation.

Most Monte Carlo rendering denoisers are based on some variants of convolutional neural networks. We instead combine our recurrent transformer network with the kernel-predicting architecture

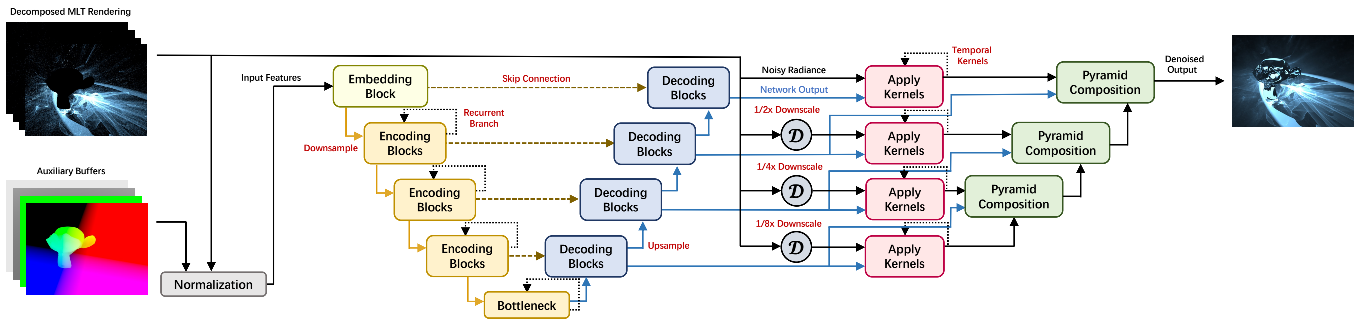


Fig. 2. **Our network architecture**. The input to our network consists of images produced by different mutation strategies (Fig. 3 and Section 3.1) and the auxiliary buffers (shading normals, world positions, reflectance). We apply a U-Net-like architecture to process the inputs. The recurrent branches store hidden states that retain temporal information (Fig. 5 and Section 3.2). The decoding blocks output filtering kernels that we apply in a multi-scale manner [Vogels et al. 2018] to produce the final image (Fig. 7 and Section 3.3).

that has shown exceptional denoising capability for Monte Carlo renderings [Bako et al. 2017; Fan et al. 2021; Gharbi et al. 2019].

While most Monte Carlo denoising methods focus on spatially uncorrelated noise, Back et al. [2020, 2023] have proposed to jointly denoise a pair of uncorrelated and correlated rendered images. In this work, we focus on the case where we have a correlated rendered image sequence as input and aim to achieve temporal coherence. Combining our architecture with Back et al.'s approach would be an exciting future work.

2.3 Modern Denoiser Based on Transformer

Deep learning has witnessed a paradigm shift with the advent of the Transformer architecture [Vaswani et al. 2017]. Abandoning the recurrent layers in classical models, Transformers employ self-attention mechanisms to weigh input features dynamically, providing them with a powerful capacity for handling sequences. Riding on the success of the Transformer in language processing, the concept was adapted for computer vision tasks through the Vision Transformer (ViT) initially, and followed by other variants like the Swin Transformer [Liu et al. 2021] and DeiT [Touvron et al. 2021].

The advancements in Transformer architectures, combined with the foundational models mentioned, have given rise to numerous transformer-based image denoising networks, notably Restormer [Zamir et al. 2022], Uformer [Wang et al. 2022], and SwinIR [Liang et al. 2021], among others. Video denoising methods have also emerged, such as VRT [Liang et al. 2022a] and RVRT [Liang et al. 2022b], which leverage temporal data. However, there are limited transformer-based solutions specifically tailored for Monte Carlo denoising, with AFGSA [Yu et al. 2021] being a notable exception. This scarcity can be attributed to the substantial number of parameters transformer architectures require, making real-time rendering challenging. Additionally, training these architectures demands extensive datasets, which are not always available for Monte Carlo denoising tasks. Regardless of these challenges, we believe that a transformer-based denoiser is essential for complex MLT denoising tasks, where our focus is on optimizing denoising quality rather than real-time performance.

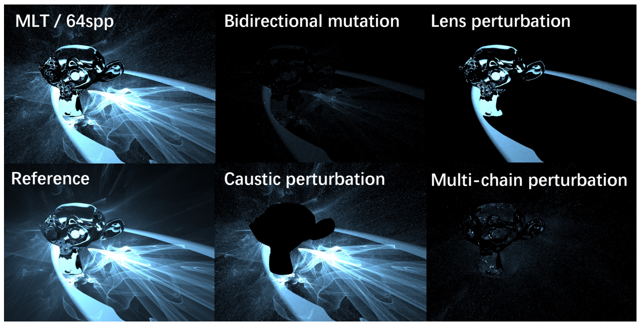


Fig. 3. **Sample Decomposition.** We decompose an MLT rendering into different contributions coming from different mutation strategies and denoise them separately. Here we visualize the contributions from the four mutation strategies of path-space MLT [Veach and Guibas 1997]. Different mutation strategies exhibit different noise patterns for different scenes, and hence would benefit from different denoising strategies.

3 MLT DENOISING WITH RECURRENT TRANSFORMER BLOCKS

We introduce a neural network specifically tailored for denoising in the context of MLT (Metropolis Light Transport) rendering. As shown in Fig. 2, our architecture adopts a U-Net-like [Ronneberger et al. 2015] structure, containing a series of carefully designed multi-layer recurrent encoding and decoding blocks, aimed at maximizing the use of continuity priors across frames and various scales, thus effectively mitigating noise generated by the MLT algorithm (detailed architecture is further elaborated in Appendix A). Additionally, in response to the broad range and complex patterns of MLT noise characteristics, we design a sample decomposition step for the input data and a hierarchical kernel application scheme.

3.1 Sample Decomposition for MLT

Inspired by sample-based methods for PT denoising [Balint et al. 2023; Gharbi et al. 2019; Hasselgren et al. 2020; Işık et al. 2021; Zimmer et al. 2015], we decompose MLT radiance by light path contributions from different mutation and perturbation strategies.

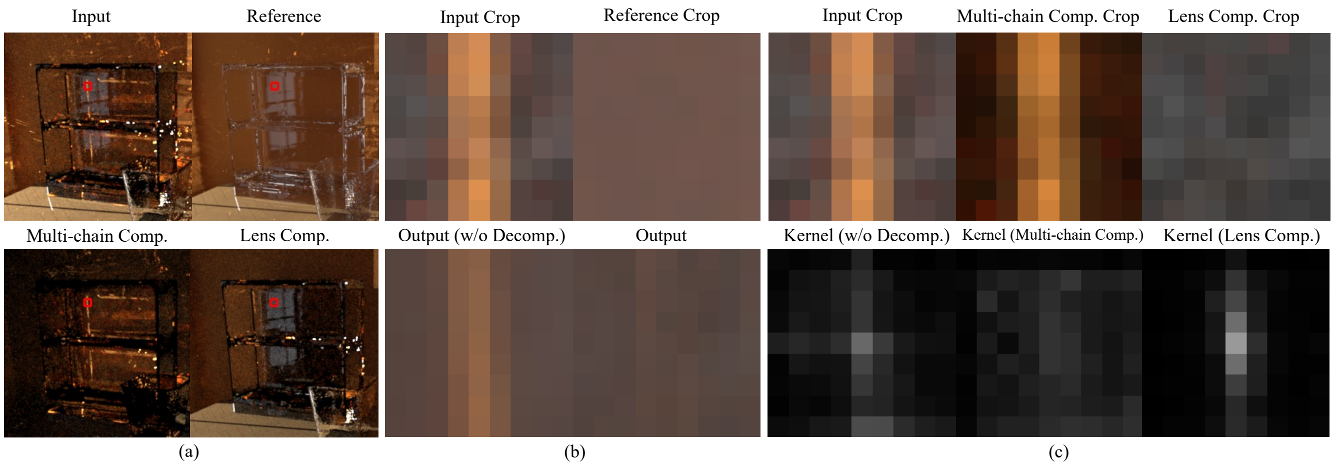


Fig. 4. **An example showing the effectiveness of sample decomposition**. (a) shows an example in which the multi-chain perturbations produce several slender artifacts. (b) shows a small crop that contains one of the artifacts, and our denoiser with sample decomposition performs better in removing such artifacts than the ablation without sample decomposition. (c) shows the predicted kernels of the center pixel. We can see the single kernel from the ablation is not able to diminish the artifact, while the kernel of the multi-chain perturbation component learned such noise pattern, and cooperated with the kernel of the lens perturbation component to reconstruct a better output.

We use the original path-space MLT algorithm proposed by Veach and Guibas [1997] and use the default mutation and perturbation strategies implemented in Mitsuba [Jakob 2010], including bidirectional mutation, lens perturbation, caustic perturbation and multi-chain perturbation. The MLT renderer is responsible for generating both direct and indirect illumination. Different mutation strategies yield varying outcomes with respect to different light rays and acceptance rates, consequently leading to distinct variances. This, in turn, results in varying degrees of noise and radiance across different materials. Decoupling the rendering results of various mutations and perturbations can be beneficial in preserving more details.

As shown in Fig. 3, in this particular scene lit by a single spotlight, the rendering of lens perturbation shows low noise level and large radiance, since in this case it mainly focuses on the direct lighting components that are either directly visible or through specular interactions. Caustic and multi-chain perturbation both have high noise levels for this scene. The former is caused by the perturbation from the spot light, while the latter results from having to deal with intricate caustic patterns. Bidirectional mutation regenerates subpaths from scratch and hence has a low acceptance rate.

To depict the effectiveness of separate denoising kernels on decomposed renderings, Fig. 4 shows an example of how independent kernels from two components work together to remove artifacts. We compose the independent denoised result at the last step and form the final output. The results of our decomposing technique in an overall improvement for MLT denoising are shown in Section 5.4.2.

3.2 Recurrent transformer blocks for temporal accumulation

Most interactive Monte Carlo denoising methods apply the per-pixel input blending for temporal accumulation. The aligned noisy renderings are linearly blended among frames with a fixed or adaptive weight. While this strategy is efficient and results in good temporal stability in denoising path-traced renderings, it is not sufficient for eliminating the correlated noise in MLT renderings.

To tackle this challenge, we propose a novel attention-based recurrent block to better leverage temporal information. The block contains a self-attention module with a similar structure as Restormer [Zamir et al. 2022] to aggregate spatial and cross-channel context and a cross-attention module with recurrent structure to recognize and eliminate the noise using temporal continuity priors.

Fig. 5 shows the structure of one transformer block in our encoder. $X_i^{(t)}, H_i^{(t)} \in \mathbb{R}^{H\times W\times C}$ denote the input feature and recurrent hidden-state feature of i-th layer at frame t, respectively. The query, key, and value maps Q_s, K_s, V_s of the self-attention layer are derived from only the input features; for the cross-attention layer, the query map Q_c comes from the input feature, while the key and value map K_c, V_c is obtained from the hidden state of the last frame:

$$\begin{aligned} Q_s, K_s, V_s &= \mathcal{E}_q(X_i^{(t)}),\ \mathcal{E}_k(X_i^{(t)}), \mathcal{E}_v(X_i^{(t)}) \\ Q_c, K_c, V_c &= \mathcal{E}'_q(X_i^{(t)}),\ \mathcal{E}'_k(\mathcal{W}_{t-1\to t}(H_i^{(t-1)})), \\ &\quad \mathcal{E}'_v(\mathcal{W}_{t-1\to t}(H_i^{(t-1)})), \end{aligned} \tag{3}$$

where $\mathcal{E}$ denotes the projection block containing a 1×1 convolution layer followed by a 3×3 depth-wise convolution layer, $\mathcal{W}$ is a warp operator that warps the images using the primary motion vector (see Section 4.3.2).

The channel attention module (Fig. 6) can be formulated as:

$$\text{Attention}(\hat{Q}, \hat{K}, \hat{V}) = \hat{V} \cdot \text{Softmax}(\hat{K} \cdot \hat{Q}/\alpha), \tag{4}$$

where $\hat{Q}, \hat{V} \in \mathbb{R}^{HW\times C}$ and $\hat{K} \in \mathbb{R}^{C\times HW}$ are projection matrices reshaped from Q, K and V, and α is a learnable parameter for scaling.

The self-attention and cross-attention results are concatenated along the channel axis and projected back to the original shape

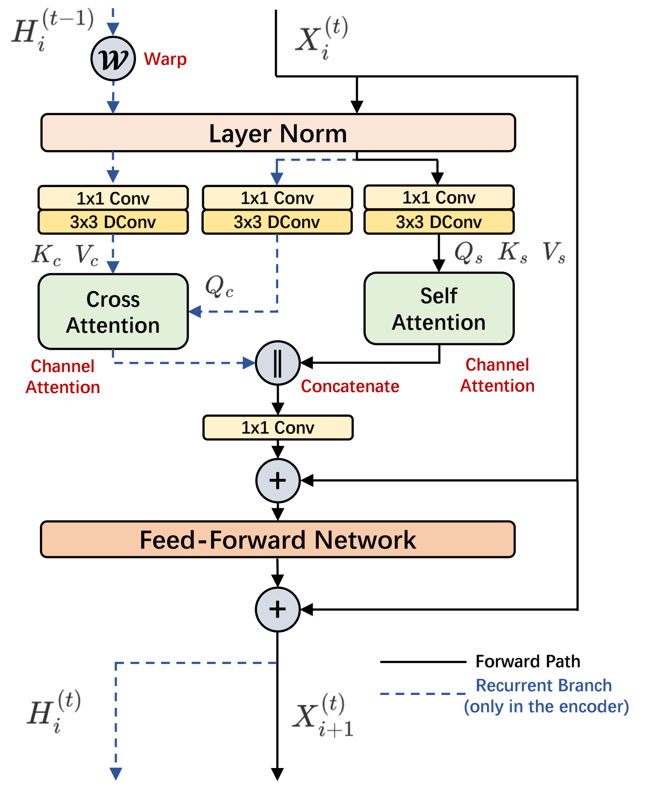


Fig. 5. **Recurrent transformer block.** Forward paths and self-attention modules are included in both encoding and decoding blocks, while recurrent branches and cross-attention modules are only included in encoding blocks.

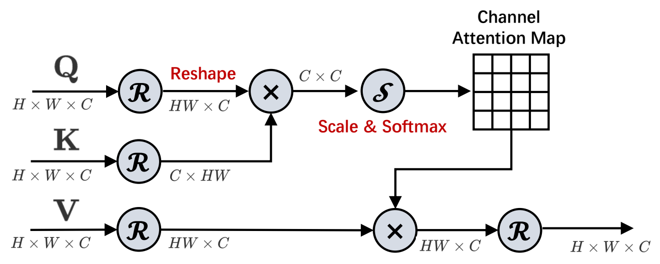


Fig. 6. **Channel attention module.** The bottleneck of the attention module is the matrix multiplication between the reshaped value matrix and the attention map. For channel attention, the shapes are $HW \times C$ and $C \times C$ (C is the number of channels), resulting in the complexity of $O(C^2HW)$, linear to resolution. For classical attention in ViT, the shapes are $C \times HW$ and $HW \times HW$, so the complexity is $O((HW)^2C)$, quadratic to resolution.

using a 1×1 convolution. The Feed-Forward Network is implemented using a Gated-Dconv Feed-Forward Network [Zamir et al. 2022], and we apply residual skip connections [He et al. 2016] both before and after. The output $X_{i+1}^{(t)}$ is passed as an input to the next encoding/decoding block and used as the hidden state for the next processed frame, $H_i^{(t)}$.

We use the same transformer blocks in our decoder, except we remove the part for calculating cross-attention and only feed the self-attention to the project convolution $\mathcal{E}$.

The self-attention module primarily focuses on spatial and low-level noise. In contrast, the cross-attention module accumulates correct features and addresses inter-frame inconsistencies such as correlated noise. By integrating these two modules, our system can robustly denoise MLT rendered images. Moreover, the design of channel attention achieves better efficiency while maintaining spatial perception ability. Since the computation complexity of the channel attention is linear to the input resolution, unlike classical Vision Transformer which the complexity is quadratic (see Fig. 6), we can afford to use these recurrent transformer blocks in denoising large images to achieve better results. We show the advantage of this transformer-based model with further studies in Section 5.4.3.

3.3 Hierarchical Kernel Application

Following previous Monte Carlo denoising methods [Bako et al. 2017; Fan et al. 2021; Gharbi et al. 2019; Vogels et al. 2018], our decoding blocks predict normalized kernels to ensure there is no color shift and enforce the conservation of energy. Similar to Vogels et al. [2018], we adopt a hierarchical kernel application pipeline, where multi-scale radiance is filtered and composed with kernels and blending parameters predicted from the network. Fig. 7 shows the kernel application process at layer l ($l = 0, 1, 2, 3$), where κ^l_{uvxyt}, K^l_{uvxyt}, λ^l_{xyt} and α^l_{xyt} are all learnable parameters output from the l-th decoding layer and $L^l_{xyt}, \bar{L}^l_{xyt}\hat{L}^l_{xyt}$ are radiance buffer. The single-layer kernel-application process can be formulated as

$$\begin{aligned}\bar{L}^l_{xy0} &= \sum_{uv} K^l_{uvxy0} L^l_{(x+u-w/2)(y+v-h/2)0},\\ \bar{L}^l_{xyt} &= (1-\lambda_{xyt}) \sum_{uv} K^l_{uvxyt} L^l_{(x+u-w/2)(y+v-h/2)t}\\ &\quad + \lambda_{xyt} \sum_{uv} \kappa^l_{uvxyt} (\mathcal{W}_{t-1\to t}(\bar{L}^l_{t-1}))_{(x+u-w/2)(y+v-h/2)},\end{aligned} \tag{5}$$

where w, h are the weight and height of the kernel, L^l_{xyt} is the downscaled input noisy radiance at the l-th layer and the frame t, K^l_{uvxyt} is a normalized kernel that gathers values from neighbor pixels (u, v) to the pixel (x, y), κ^l_{uvxyt} is another normalized kernel (temporal kernel) that gathers values from previous output, $\bar{L}^l_{xyt}$ and $\bar{L}^l_{xy(t-1)}$ are coarse denoised outputs from frame t and frame $t-1$, λ^l_{xyt} is a blending parameter which is restricted to the range $(0, 1)$ by a sigmoid function, and $\mathcal{W}$ is the warp operator.

After obtaining all the coarse denoised outputs from each layer, we employ a progressive scale-composition module from Vogels et al. [2018]. This module progressively composes adjacent coarse denoised outputs, ascending from the lowest resolution to the full resolution, as expressed by the following equations:

$$\begin{aligned}\hat{L}^0_{xyt} &= \bar{L}^0_{xyt}\\ \hat{L}^l_{xyt} &= \bar{L}^l_{xyt} - \mathbf{U}(\alpha^l_{xyt})\mathbf{D}(\bar{L}^l_{xyt}) + \mathbf{U}(\alpha^l_{xyt}\hat{L}^{l-1}_{xyt}),\end{aligned} \tag{6}$$

where $\bar{L}^l_{xyt}$ is the coarse denoised output and $\hat{L}^l_{xyt}$ is the fine denoised output at layer l composed from the fine denoised output

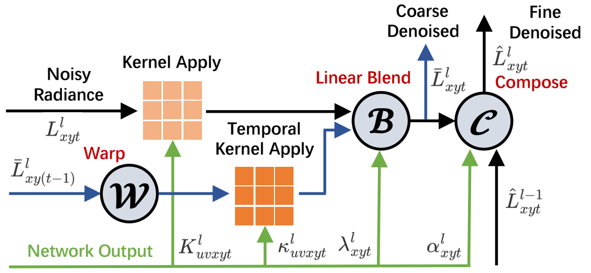


Fig. 7. **Kernel Application module**. The module first takes in noisy and denoised images from the current and previous frames. It warps the previous frame using motion vectors to align the two frames. It then applies a spatial-temporal filter to the images with weights generated by the decoding blocks, and composes the images at different scales.

$\hat{L}^{l-1}_{xyt}$ at layer $l-1$. α^l_{xyt} is a blending parameter that is also restricted to the range $(0,1)$ by a sigmoid function. **D** and **U** is the downsampling operator and the nearest-neighbor upsampling operator [Vogels et al. 2018].

At last, we compose the final denoised output from 4 decomposed denoised components at layer 3 (full resolution).

$$O_{xyt} = \sum_{k=1}^{4} \hat{L}^3_{xytk}. \tag{7}$$

3.4 Why recurrent blocks are necessary for MLT denoising

While the recurrent method in Monte Carlo denoising has been explored by Chaitanya et al. [2017], this approach has not been widely adopted in subsequent research. Instead, per-pixel input blending for temporal accumulation is favored for PT denoising due to its efficiency [Balint et al. 2023; Fan et al. 2021; Işık et al. 2021; Koskela et al. 2019]. Constant weights for per-pixel input blending efficiently accumulate samples over time, but are affected by invalid samples caused by view-dependent effects. Using adaptive weights learned from the neural network can eliminate invalid samples to a certain extent, but they also struggle with complex scenes with more invalid samples. In light of these challenges, we revisited the recurrent method that preserves in-depth temporal information within different levels of network layers, which shows promise for MLT denoising. In Fig. 8, we demonstrate the limitations of per-pixel input blending in denoising MLT and bidirectional path tracing in complex scenes, such as those involving complex glass objects.

4 DATASET AND IMPLEMENTATION DETAILS

4.1 Dataset

MLT is adept at rendering scenes with challenging visibility and intricate caustics. We constructed such scenes using the OpenRooms framework [Li et al. 2021] and served as our dataset. Our dataset comprises 124 different scenes with 6300 animated sequences wherein the camera moves through the static scene. We also add moving spot lights into the scene to create moving caustics. Each sequence is rendered at a resolution of 640×480 pixels. We adopted the path-space MLT to render all the scenes with Mitsuba v0.6 renderer [Jakob 2010]. We introduced our method to modify the original OpenRooms and generate scenes specialized for MLT in Appendix B.

We divide the 124 scenes into 114 training scenes and 10 test scenes. All the 7-frame training sequences are from training scenes, with varying input sample counts from 32 to 128 samples per pixel. For testing, we built a dataset comprising two 60-frame sequences with 16spp/32spp/64spp/128spp input from each test scene. A validation dataset is used to monitor the training progression. The validation dataset consists of 50 sequences, each 7 frames in length, all sourced from the test scenes.

4.2 Loss Function

Our denoiser is optimized to deliver renderings that are both of high quality and temporally stable. Thus, we minimize a combination of the single image loss and the temporal loss as defined below:

$$\mathcal{L} = \mathcal{L}_{single} + \lambda \mathcal{L}_{temporal}, \tag{8}$$

where we set $\lambda = 0.5$ and

$$\begin{aligned} \mathcal{L}_{single} &= \text{SMAPE}(I_{\text{output}}, I_{\text{gt}}) \\ \mathcal{L}_{temporal} &= \text{SMAPE}(\partial_t I_{\text{output}}, \partial_t I_{\text{gt}}). \end{aligned} \tag{9}$$

The Symmetric Mean Absolute Percentage Error (SMAPE) [Vogels et al. 2018] for two temporal sequences A and B is defined as:

$$\text{SMAPE}(A,B) = \frac{1}{3}\mathbb{E}_{xyt}\frac{||A_{xyt} - B_{xyt}||_1}{||A_{xyt}||_1 + ||B_{xyt}||_1 + \epsilon}, \tag{10}$$

where $\epsilon = 10^{-2}$ is added to prevent singularities. The expectation $\mathbb{E}_{xyt}$ is computed over both pixels and frames and $||\cdot||_1$ represents L_1 norm. The factor of 3 accounts for the three color channels.

4.3 Implementation Details

4.3.1 Training settings. We implemented our denoiser in PyTorch [Paszke et al. 2019] and optimized the models using the Adam optimizer [Kingma and Ba 2015] with the default parameters. Our denoiser is trained for 50 epochs on our train dataset, comprising roughly 160k iterations, which takes about 4 days for an NVIDIA-A100-SXM4-80GB GPU. We apply a mini-batch of 2 and an initial learning rate of 10^{-4}. We multiply the learning rate by 0.8 every 20 epochs. In the training process, we apply the data augmentation with random rotations, horizontal and vertical flips, and crops measuring 256×256 pixels.

4.3.2 Motion Vectors and Warping. We extract primary motion vectors [Zimmer et al. 2015] by ground-truth poses and depths, followed by a forward-backward consistency check for masking out invisible positions. We use the primary motion vectors to warp feature maps in recurrent transformer blocks (Section 3.2) and hierarchical kernel application (Section 3.3) at different resolutions.

For the scaled warping, we make a full-resolution grid with our motion vectors, where the vectors are normalized with respect to the resolution. We then warp the low-resolution features with bilinear resampling and downsample the output to the original size.

4.3.3 Auxiliary Buffers. We use four kinds of auxiliary buffers: albedo, distance, shading normals, and world positions. All of them are directly generated from the Mitsuba renderer. While we employ

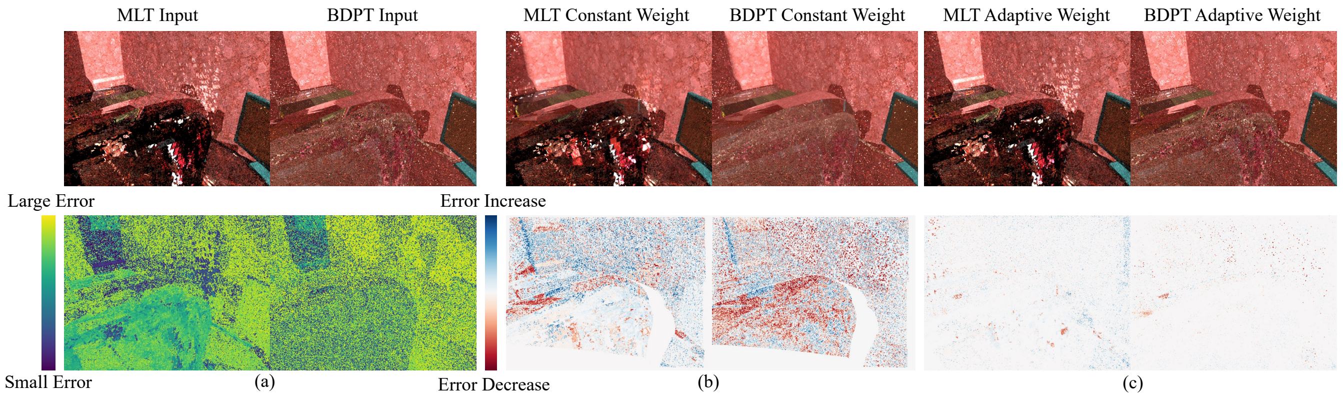


Fig. 8. **Evaluation of constant weight linear blending and adaptive weight linear blending**. (a) shows the noisy MLT and Bidirectional Path Tracing (BDPT) radiance and absolute errors against the reference, in which BDPT shows a more uniform error. (b) and (c) shows how the error varies after applying a constant or adaptive weight blending. We can see the constant weight blending performs much better on BDPT than MLT. The adaptive weight blending shows a very minor change of input for both MLT and BDPT, which proves its inefficiency in a scene full of view-dependent effects.

primary motion vectors for warping, they are excluded from direct input buffers since we did not find them to be helpful.

4.3.4 Input Normalization. We normalize all the input features with the same strategy as Işık et al. [2021], in which the input radiance components are tone-mapped with $x \mapsto \log(1+x)$ first and then normalized to $[-1, 1]$. The albedo buffer, distance buffer and position buffer are also normalized to the same range.

5 RESULTS

5.1 Metrics

To evaluate the denoising quality of a single frame, we employ the peak signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM) on the tone-mapped radiance. We follow the sRGB standard to tone map the radiance:

$$\tau_r(x) = \begin{cases} 12.92x, & x \leq 0.0031308 \\ \min(1, 1.055x^{\frac{1}{2.4}} - 0.55), & x > 0.0031308. \end{cases} \tag{11}$$

For evaluating temporal stability, previous works have applied Temporal Relative Mean Absolute Error (TRMAE) [Işık et al. 2021] and Temporal PSNR (TPSNR) [Balint et al. 2023; Hasselgren et al. 2020]. However, we found these metrics based on the temporal finite differences do not adequately capture the inherent temporal instability of MLT. This instability, resulting from the pronounced local correlation in MLT renderings, is perceptible to the human perspective. We found that the well-established perceptual-informed metrics FovVideoVDP [Mantiuk et al. 2021] can perceive such temporal instability well. Therefore, we only use this metric to evaluate the temporal stability of the denoised animations. We use FovVideoVDP's standard HDR model to measure the video quality in the original radiance domain. A detailed study on different metrics' ability to measure temporal stability can be found in Appendix C.

5.2 Comparision of Different Denoising Methods

Given the challenge of estimating radiance variance with MLT, most of the traditional denoising methods are unsuitable for MLT renderings. Thus, we compare our approach only with deep learning-based methods. We compared our model with 5 baseline models in total, including RAE [Chaitanya et al. 2017], NTASD [Hasselgren et al. 2020], KPCN [Bako et al. 2017], AFGSA [Yu et al. 2021], and IDANF [Işık et al. 2021]. We train all the baseline models on our dataset and adapt them to be suitable for denoising MLT animations, which makes our comparison as fair as possible. For NTASD, we do not apply adaptive sampling since it is incompatible with MLT. For IDANF, we extract the per-pixel embeddings directly instead of extracting per-sample embeddings at first. For the two offline single-image denoisers KPCN and AFGSA, we adapt them to sequence denoisers by applying the learnable adaptive weight input blending [Balint et al. 2023; Işık et al. 2021]. We found that the original AFGSA model that outputs per-pixel residuals between the noisy and reconstructed images is difficult to train in our task, so we changed the output to 9×9 kernels. In addition, we found that the input normalization and loss function we use for our model benefits the two offline single-image denoisers, so we also apply them.

We conduct a comprehensive quantitative evaluation on our 20 test sequences and show the result in Table 1. We also test all the methods on the classic *Veach-Ajar* scene [Bitterli 2016] as well as two elaborate scenes *Monkey* [Ebke 2021] and *Crystal* to evaluate the generalizing ability. The comparative results are provided in Fig. 9. We also visualize a corresponding 6-frame denoised sequence for our method in Fig. 10.

5.3 Generalization to unseen samples

While our network is trained on images with 32-128 samples per pixel, it generalizes to sample counts outside this range. We rendered additional noisy images for all the test sequences from 1spp to 1024spp to assess how our model can generalize to denoising scenes

Table 1. **Quantitative evaluation for all the models**. We compute the average metric values across 20 60-frame test animations. We highlight the first and second best result for each metric and each sample count. Our model consistently outperforms all the compared models across all sample count levels.

		MLT Input	Ours	RAE	NTASD	KPCN	AFGSA	IDANF
16spp	PSNR↑	21.54	28.55	25.58	26.31	25.80	26.27	26.67
	SSIM ↑	0.381	0.856	0.770	0.756	0.817	0.811	0.818
	FoVVDP ↑	6.602	7.916	7.612	7.510	7.470	7.472	7.553
32spp	PSNR↑	22.78	29.21	26.05	27.13	26.15	26.80	27.07
	SSIM ↑	0.480	0.866	0.786	0.786	0.828	0.829	0.831
	FoVVDP ↑	6.840	8.081	7.758	7.687	7.595	7.635	7.694
64spp	PSNR↑	23.82	29.60	26.41	27.78	26.39	27.14	27.35
	SSIM ↑	0.579	0.874	0.797	0.810	0.837	0.843	0.841
	FoVVDP ↑	7.059	8.194	7.878	7.833	7.677	7.758	7.800
128spp	PSNR↑	24.74	29.99	26.73	28.34	26.64	27.47	27.63
	SSIM ↑	0.667	0.882	0.805	0.829	0.844	0.854	0.849
	FoVVDP ↑	7.250	8.289	7.993	7.953	7.762	7.870	7.912

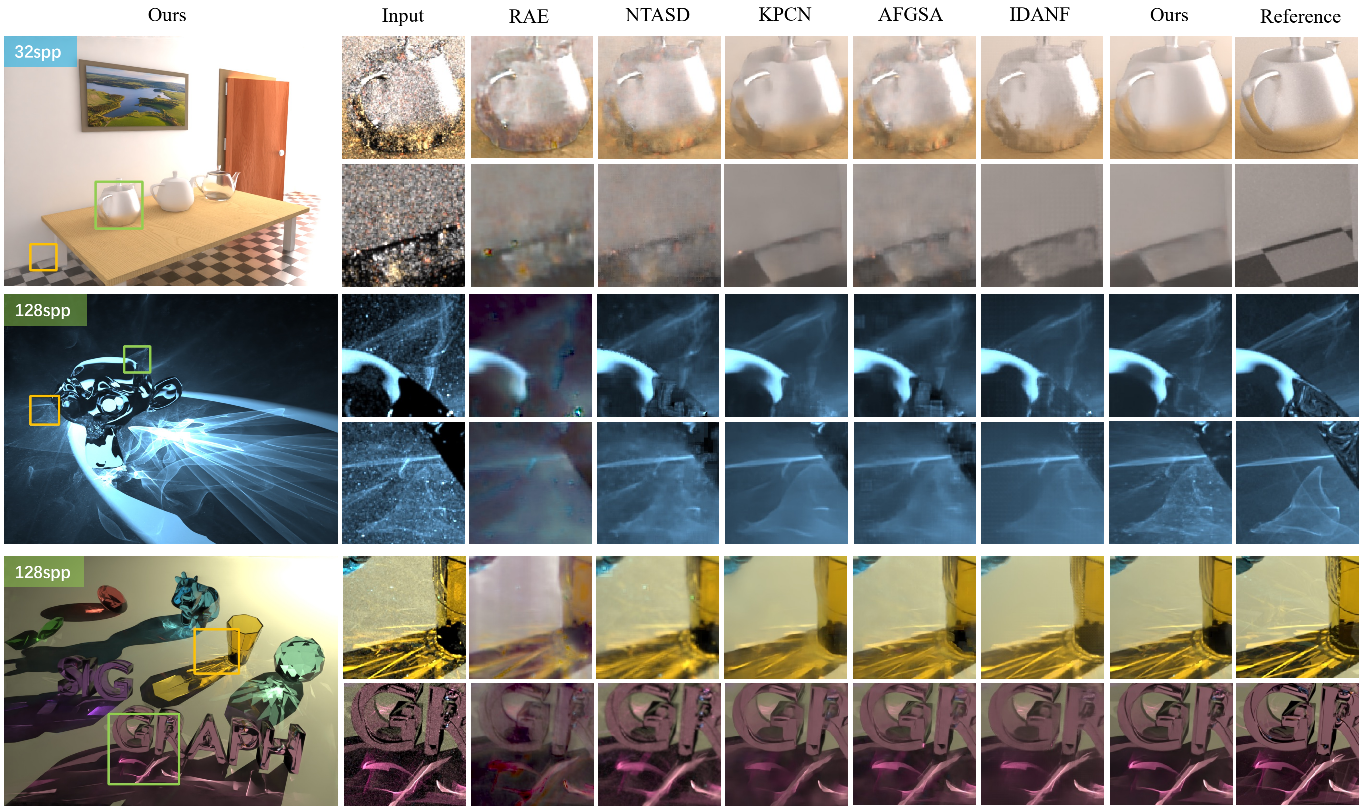


Fig. 9. **Qualitative comparison between our method and previous methods.** For each scene, all the denoisers are applied to the whole animation sequence and we take one frame from it for comparison. Our method outperforms all the previous methods in removing incorrect flickering artifacts as well as keeping correct details.

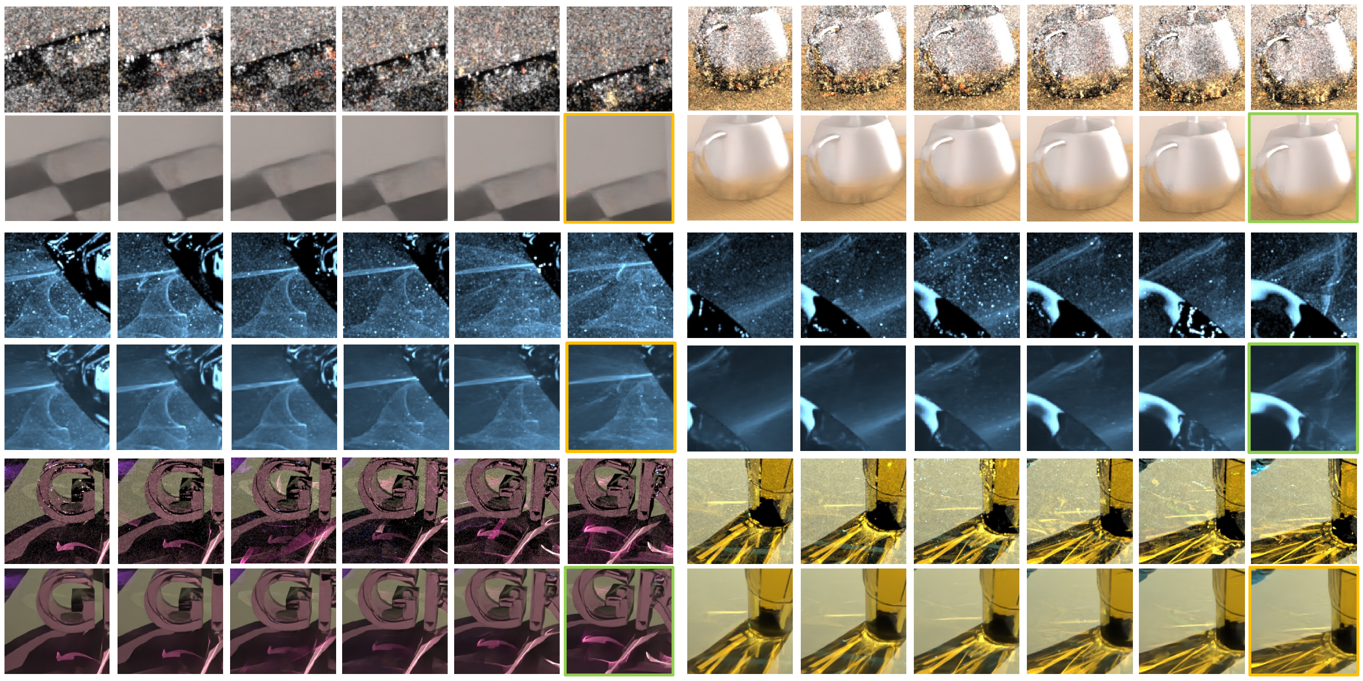


Fig. 10. **Evaluation of denoising whole sequences**. We show the 6-frame noisy and denoised sequences of the crops in Fig. 9. Our method tends to keep image details as much as possible while maintaining temporal stability.

Table 2. **Quantitative evaluation for different temporal accumulation techniques.** Better temporal accumulation leads to an overall improvement both in denoising quality and temporal stability. Here we ablate our temporal accumulation strategy using SR (simple recurrent), AB (adaptive blending), CB (constant blending), and NB (no blending).

		Ours	SR	AB	CB	NB
16spp	PSNR↑	**28.55**	28.39	28.32	27.29	26.21
	SSIM ↑	**0.856**	0.855	0.845	0.829	0.829
	FoVVDP ↑	**7.916**	7.894	7.877	7.621	7.447
32spp	PSNR↑	**29.21**	29.01	28.83	27.71	26.87
	SSIM ↑	**0.866**	0.865	0.856	0.838	0.840
	FoVVDP ↑	**8.081**	8.063	8.010	7.748	7.589
64spp	PSNR↑	**29.60**	29.46	29.20	27.94	27.36
	SSIM ↑	**0.874**	**0.874**	0.865	0.846	0.851
	FoVVDP ↑	**8.194**	8.164	8.126	7.822	7.703
128spp	PSNR↑	**29.99**	29.95	29.54	27.97	27.75
	SSIM ↑	0.882	**0.883**	0.873	0.851	0.860
	FoVVDP ↑	**8.289**	8.279	8.230	7.863	7.810

with unseen samples. We compared our results against all the baseline models in Fig. 11. Our model maintains the best results on all the sample counts, demonstrating adequate ability to denoise both extremely noisy MLT renderings and high-quality MLT renderings.

5.4 Ablation Study

5.4.1 Temporal accumulation. We implemented four ablations to evaluate our recurrent method's ability for temporal accumulation. The ablation *SR* (simple recurrent) removes the cross-attention modules, where the warped hidden states are concatenated with the outputs from self-attention modules directly. The ablation *AB* (adaptive blending) applies the strategy that accumulates input radiance with per-pixel learnable adaptive weights [Balint et al. 2023; Işık et al. 2021]. *CB* (constant blending) applies the strategy that accumulates input radiance with constant weight [Fan et al. 2021; Koskela et al. 2019]. *NB* (no blending) does not accumulate input radiance. All the ablations keep the temporal kernels and use the same encoding and decoding blocks. Table 2 shows our quantitative evaluation results on the same test set used in Section 5.2. The result suggests that using recurrent branches has a great advantage over all input blending methods. It also shows the effectiveness of the cross-attention module, which is more obvious in the cases of denoising low-sample renderings.

5.4.2 Kernel Application. We study how different kernel-application strategies affect the MLT denoising results. Table 3 shows our quantitative evaluation results on the same test set used in Section 5.2, where we list the temporal and output kernel size for all the ablations. We also list the average output entries including $\kappa^l_{uvxyt}, K^l_{uvxyt}, \lambda^l_{xyt}, \alpha^l_{xyt}$ at each layer. We take our full model as a standard, with 4 times output entries for separately denoising 4 MLT

Table 3. **Quantitative evaluation for different kernel settings**. We highlight the first and second best result for each metric and each sample count. (*) means the number is an equivalent average for the ablation that does not apply hierarchical denoising.

	Temporal Kernel Size	5 × 5 (x4)	-	-	5 × 5	5 × 5	5 × 5	9 × 9
	Output Kernel Size	9 × 9 (x4)	-	9 × 9	9 × 9	9 × 9	13 × 13	17 × 17
	Layer Output Entries	432(Ours)	1	82	27.6*	108	194	444
16spp	PSNR↑	28.55	24.75	28.44	28.32	28.44	28.48	28.63
	SSIM ↑	0.856	0.814	0.840	0.850	0.853	0.853	0.856
	FoVVDP ↑	7.916	7.467	7.900	7.855	7.868	7.878	7.925
32spp	PSNR↑	29.21	24.87	29.13	28.95	29.10	29.12	29.18
	SSIM ↑	0.866	0.817	0.852	0.863	0.864	0.863	0.865
	FoVVDP ↑	8.081	7.567	8.075	8.031	8.055	8.055	8.090
64spp	PSNR↑	29.60	25.05	29.63	29.32	29.52	29.49	29.49
	SSIM ↑	0.874	0.820	0.863	0.873	0.873	0.870	0.872
	FoVVDP ↑	8.194	7.651	8.194	8.132	8.166	8.169	8.202
128spp	PSNR↑	29.99	25.09	30.10	29.65	29.89	29.84	29.83
	SSIM ↑	0.882	0.820	0.872	0.881	0.881	0.877	0.880
	FoVVDP ↑	8.289	7.709	8.331	8.243	8.282	8.276	8.301

Table 4. **Quantitative evaluation for different feature extractors**. Our transformer model is about two times slower than the ConvNext ablation on an NVIDIA-A100-SXM4-80GB, but has higher denoising quality overall.

		Ours	ConvNext
Time (s/frame)		0.180	0.094
GPU Memory		7860MiB	7708MiB
Number of Parameters		20M	28M
16spp	PSNR↑	**28.55**	28.17
	SSIM ↑	**0.856**	0.850
	FoVVDP ↑	**7.916**	7.876
32spp	PSNR↑	**29.21**	28.79
	SSIM ↑	**0.866**	0.860
	FoVVDP ↑	**8.081**	8.025
64spp	PSNR↑	**29.60**	29.12
	SSIM ↑	**0.874**	0.869
	FoVVDP ↑	**8.194**	8.125
128spp	PSNR↑	**29.99**	29.49
	SSIM ↑	**0.882**	0.877
	FoVVDP ↑	**8.289**	8.230

components. We do not apply sample decomposition for all ablations. The *first* ablation does not predict kernels and reconstruct the output directly. The *second* ablation does not apply temporal kernels. The *third* ablation does not apply hierarchical denoising and only applies kernels on the full resolution. The *fourth* to *sixth* ablations follow the same pipeline as the standard except for not applying sample decomposition and using different kernel sizes. The result shows that quantitative differences among different kernel settings are not so significant except that kernel-predicting is necessary. Our full model demonstrates an overall improvement over the ablation with the same kernel size and without sample decomposition. The enhancement is comparable to that of increasing the kernel size to the same scale of output entries. Considering that continually increasing the kernel size does not result in sustained improvement, our full model is expected to be the most effective when using the optimized kernel size.

5.4.3 Transformer vs. CNN. We compare our model with a CNN ablation to show the advantage of the transformer-based model in our task. For this ablation, we replace the encoding and decoding transformer blocks with large-scale convolutional blocks from ConvNext [Liu et al. 2022], which are also used by Balint et al. [2023]. We use simple recurrent branches from Chaitanya et al. [2017], adding the warp operation. The ablation is about the same scale as our model at the same number of blocks in terms of GPU memory usage. We evaluate both the metrics and inference time. Table 4 suggests that replacing CNNs with Transformers trades between denoising quality and computation cost. It is worth using the slower Transformer since the denoising time is negligible compared to the rendering time for complex scenes.

5.5 Denoising renderings for other MCMC algorithms

In this paper, we focus on path-space MLT to demonstrate that our denoiser can handle highly-correlated samples. However, both the techniques of sample decomposition and recurrent network in our method can be applied in other MCMC algorithms. To show the applicability, we train an extra model for Primary Sample Space Metropolis Light Transport (PSSMLT) [Kelemen et al. 2002]. We decompose the PSSMLT rendering into 2 different components,

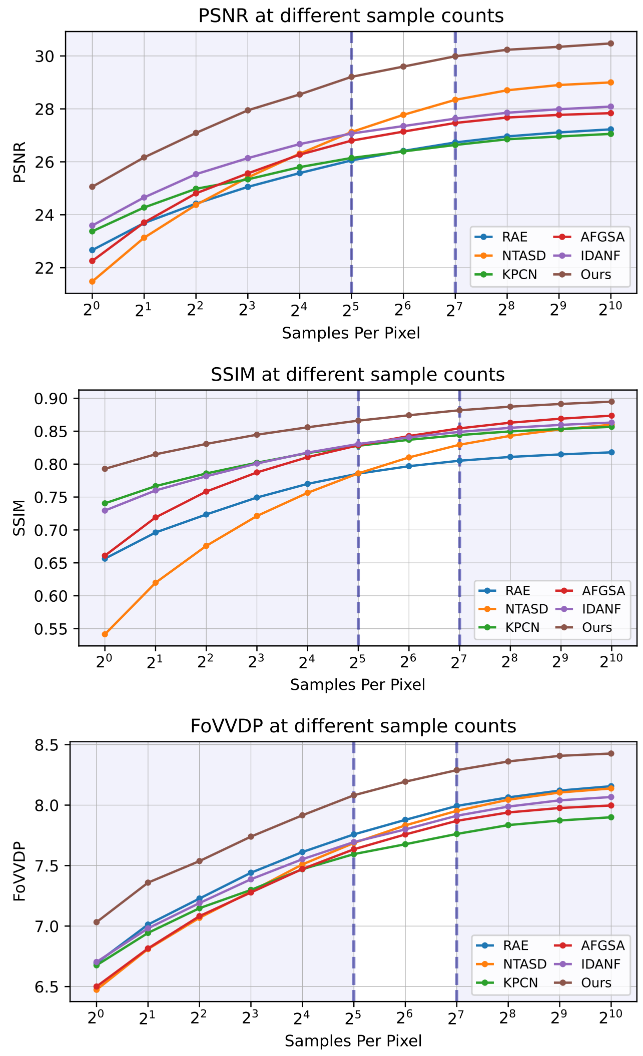


Fig. 11. **Generalization to unseen samples**. Our denoiser can consistently improve the quality and temporal stability of MLT animations rendered by different sample counts. The filled regions are sample counts unseen in the training set.

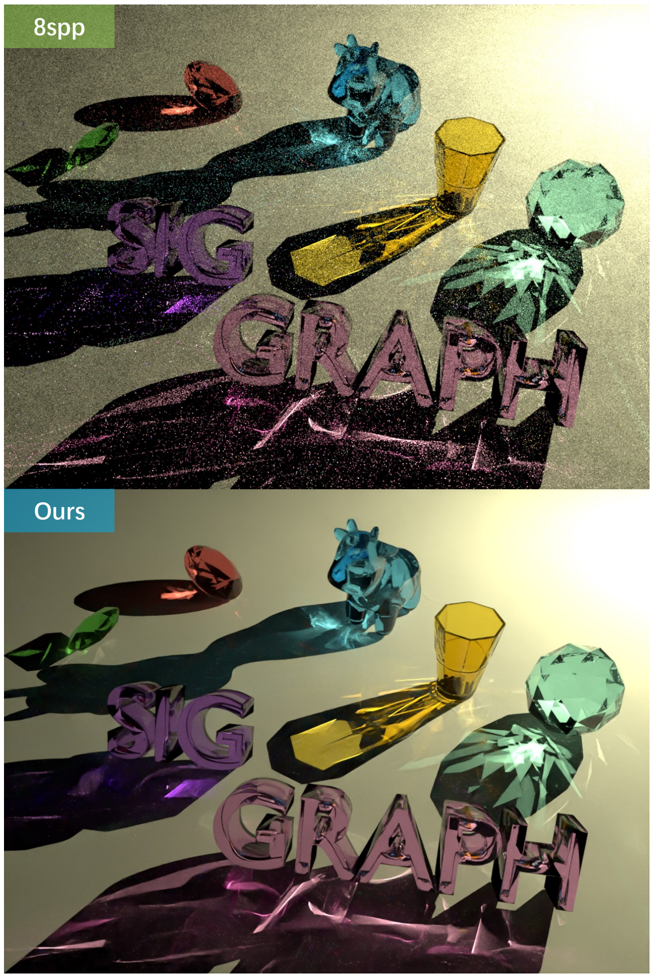


Fig. 12. **Denoising PSSMLT renderings**. The result shows that our method can be generalized to denoising PSSMLT renderings. The denoiser handles different noise patterns well.

one containing samples from large-step mutations and the other containing samples from small-step mutations. Fig. 12 shows one denoised example, in which we use a smaller sample count than that in path-space MLT since noises in PSSMLT renderings are less correlated. Our denoiser can also handle this case well.

5.6 Limitations

As discussed in Section 5.4.3, the time taken for inference does not pose significant challenges in our task. However, the substantial GPU memory requirements of our large-scale transformer-based model could be a bottleneck. Therefore, denoising high-resolution animations might be challenging on some existing GPUs.

6 CONCLUSION

We have presented our novel denoiser with recurrent transformer blocks for Metropolis Light Transport. Through detailed experiments comparing several learning-based models, we have demonstrated that modern learning-based methods are capable of effectively denoising renderings from MLT algorithms while enhancing temporal stability. Compared to existing methods, our model achieves state-of-the-art performance in both denoising quality and temporal stability. We have also introduced a new large-scale dataset of indoor scenes with complex light paths for benchmarking MLT denoising. We believe our method has shown a step towards making Metropolis light transport practical for animation rendering.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their invaluable feedback. This work was supported in part by NSF grant 2105806 and gifts from Adobe and Google.

REFERENCES

Michael Ashikhmin, Simon Premože, Peter Shirley, and Brian Smits. 2001. A variance analysis of the Metropolis light transport algorithm. Computers & Graphics 25, 2 (2001), 287–294.

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2020. Deep combiner for independent and correlated pixel estimates. ACM Trans. Graph. (Proc. SIGGRAPH Asia) 39, 6 (2020), 242–1.

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2023. Input-Dependent Uncorrelated Weighting for Monte Carlo Denoising. In SIGGRAPH Asia Conference Proceedings. 1–10.

Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. 2017. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. ACM Trans. Graph. (Proc. SIGGRAPH) 36, 4 (2017), 97:1–97:14.

Martin Balint, Krzysztof Wolski, Karol Myszkowski, Hans-Peter Seidel, and Rafał Mantiuk. 2023. Neural partitioning pyramids for denoising Monte Carlo renderings. 1–11.

Thomas Bashford-Rogers, Luís Paulo Santos, Demetris Marnerides, and Kurt Debattista. 2021. Ensemble Metropolis Light Transport. ACM Trans. Graph. 41, 1, Article 5 (2021), 15 pages.

Benedikt Bitterli. 2016. VeachAjar. https://benedikt-bitterli.me/resources/.

Benedikt Bitterli and Wojciech Jarosz. 2019. Selectively Metropolised Monte Carlo Light Transport Simulation. ACM Trans. Graph. (Proc. SIGGRAPH Asia) 38, 6, Article 153 (2019), 10 pages.

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. ACM Trans. Graph. (Proc. SIGGRAPH) 36, 4, Article 98 (2017), 12 pages.

David Cline, Justin Talbot, and Parris Egbert. 2005. Energy Redistribution Path Tracing. ACM Trans. Graph. (Proc. SIGGRAPH) 24, 3 (2005), 1186–1195.

Markus Ebke. 2021. LightSheet. https://benedikt-bitterli.me/resources/.

Hangming Fan, Rui Wang, Yuchi Huo, and Hujun Bao. 2021. Real-time Monte Carlo denoising with weight sharing kernel prediction network. Comput. Graph. Forum 40, 4 (2021), 15–27.

Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. 2019. Sample-based Monte Carlo Denoising Using a Kernel-splatting Network. ACM Trans. Graph. (Proc. SIGGRAPH) 38, 4, Article 125 (2019), 12 pages.

Adrien Gruson, Rex West, and Toshiya Hachisuka. 2020. Stratified Markov Chain Monte Carlo Light Transport. Comput. Graph. Forum (Proc. Eurographics) 39, 2 (2020), 351–362.

Toshiya Hachisuka, Anton S Kaplanyan, and Carsten Dachsbacher. 2014. Multiplexed Metropolis light transport. ACM Trans. Graph. (Proc. SIGGRAPH) 33, 4, Article 100 (2014), 10 pages.

Jon Hasselgren, Jacob Munkberg, Marco Salvi, Anjul Patney, and Aaron Lefohn. 2020. Neural temporal adaptive sampling and denoising. Comput. Graph. Forum (Proc. Eurographics) 39, 2 (2020), 147–155.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Computer Vision and Pattern Recognition. 770–778.

Mustafa Işık, Krishna Mullia, Matthew Fisher, Jonathan Eisenmann, and Michaël Gharbi. 2021. Interactive Monte Carlo denoising using affinity of neural features. ACM Trans. Graph. (Proc. SIGGRAPH) 40, 4, Article 37 (2021), 13 pages.

Wenzel Jakob. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

Wenzel Jakob and Steve Marschner. 2012. Manifold exploration: a Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. ACM Trans. Graph. (Proc. SIGGRAPH) 31, 4, Article 58 (2012), 13 pages.

Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A Machine Learning Approach for Filtering Monte Carlo Noise. ACM Trans. Graph. (Proc. SIGGRAPH) 34, 4, Article 122 (2015), 12 pages.

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A simple and robust mutation strategy for the Metropolis light transport algorithm. Comput. Graph. Forum (Proc. Eurographics) 21, 3 (2002), 531–540.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. (2015).

Shinya Kitaoka, Yoshifumi Kitamura, and Fumio Kishino. 2009. Replica Exchange Light Transport. Comput. Graph. Forum 28, 8 (2009), 2330–2342.

Matias Koskela, Kalle Immonen, Markku Mäkitalo, Alessandro Foi, Timo Viitanen, Pekka Jääskeläinen, Heikki Kultala, and Jarmo Takala. 2019. Blockwise multi-order feature regression for real-time path-tracing reconstruction. ACM Trans. Graph. 38, 5 (2019), 1–14.

Yu-Chi Lai, , Feng Liu, and Charles Dyer. 2009. Physically-based Animation Rendering with Markov Chain Monte Carlo. Technical Report UW-CS-TR-1653. University of Wisconsin - Madison Computer Sciences Department.

Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Meng Song, Yuhan Liu, Yu-Ying Yeh, Rui Zhu, Nitesh Gundavarapu, Jia Shi, et al. 2021. OpenRooms: An open framework for photorealistic indoor scene datasets. In Computer Vision and Pattern Recognition. 7190–7199.

Jingyun Liang, Jiezhang Cao, Yuchen Fan, Kai Zhang, Rakesh Ranjan, Yawei Li, Radu Timofte, and Luc Van Gool. 2022a. VRT: A video restoration transformer. arXiv preprint arXiv:2201.12288 (2022).

Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. 2021. SwinIR: Image restoration using swin transformer. In International Conference on Computer Vision. 1833–1844.

Jingyun Liang, Yuchen Fan, Xiaoyu Xiang, Rakesh Ranjan, Eddy Ilg, Simon Green, Jiezhang Cao, Kai Zhang, Radu Timofte, and Luc V Gool. 2022b. Recurrent video restoration transformer with guided deformable attention. Advances in Neural Information Processing Systems 35 (2022), 378–393.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In International Conference on Computer Vision. 10012–10022.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A ConvNet for the 2020s. In Computer Vision and Pattern Recognition. 11976–11986.

Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. 2020. Langevin Monte Carlo Rendering with Gradient-Based Adaptation. ACM Trans. Graph. (Proc. SIGGRAPH) 39, 4, Article 140 (2020), 16 pages.

Rafał K Mantiuk, Gyorgy Denes, Alexandre Chapiro, Anton Kaplanyan, Gizem Rufo, Romain Bachy, Trisha Lian, and Anjul Patney. 2021. FovVideoVDP: A visible difference predictor for wide field-of-view video. ACM Trans. Graph. (Proc. SIGGRAPH) 40, 4, Article 49 (2021), 19 pages.

Soham Uday Mehta, Brandon Wang, and Ravi Ramamoorthi. 2012. Axis-aligned filtering for interactive sampled soft shadows. ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31, 6 (2012), 1–10.

Xiaoxu Meng, Quan Zheng, Amitabh Varshney, Gurprit Singh, and Matthias Zwicker. 2020. Real-time Monte Carlo Denoising with the Neural Bilateral Grid. In Eurographics Symposium on Rendering - DL-only Track. 13–24.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems. 8024–8035.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-assisted Intervention. 234–241.

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In High Performance Graphics. 1–12.

Manu Mathew Thomas, Gabor Liktor, Christoph Peters, Sungye Kim, Karthik Vaidyanathan, and Angus G Forbes. 2022. Temporally stable real-time joint neural denoising and supersampling. Proceedings of the ACM on Computer Graphics and Interactive Techniques (Proc. HPG) 5, 3 (2022), 1–22.

Hugo Touvron, Matthiegu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In International Conference on Machine Learning. 10347–10357.

Joran Van de Woestijne, Roald Frederickx, Niels Billen, and Philip Dutré. 2017. Temporal coherence for Metropolis light transport. In Eurographics Symposium on Rendering - Experimental Ideas & Implementations. 55–63.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems 30 (2017).

Eric Veach. 1998. Robust Monte Carlo methods for light transport simulation. Stanford University.

Eric Veach and Leonidas J Guibas. 1997. Metropolis light transport. In SIGGRAPH. 65–76.

Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. 2018. Denoising with kernel prediction and asymmetric loss functions. ACM Trans. Graph. (Proc. SIGGRAPH) 37, 4 (2018), 1–15.

Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. 2022. Uformer: A general U-shaped transformer for image restoration. In Computer Vision and Pattern Recognition. 17683–17693.

Lifan Wu, Ling-Qi Yan, Alexandr Kuznetsov, and Ravi Ramamoorthi. 2017. Multiple axis-aligned filters for rendering of combined distribution effects. Comput. Graph. Forum (Proc. EGSR) 36, 4 (2017), 155–166.

Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural Supersampling for Real-time Rendering. ACM Trans. Graph. (Proc. SIGGRAPH) 39, 4 (2020).

Ling-Qi Yan, Soham Uday Mehta, Ravi Ramamoorthi, and Fredo Durand. 2015. Fast 4D sheared filtering for interactive rendering of distribution effects. ACM Trans. Graph. 35, 1 (2015), 1–13.

Jiaqi Yu, Yongwei Nie, Chengjiang Long, Wenjun Xu, Qing Zhang, and Guiqing Li. 2021. Monte Carlo denoising via auxiliary feature guided self-attention. ACM Trans. Graph. 40, 6 (2021), 273–1.

Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. 2022. Restormer: Efficient transformer for high-resolution image restoration. In Computer Vision and Pattern Recognition. 5728–5739.

Henning Zimmer, Fabrice Rousselle, Wenzel Jakob, Oliver Wang, David Adler, Wojciech Jarosz, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. 2015. Path-space motion estimation and decomposition for robust animation filtering. Comput. Graph. Forum (Proc. EGSR) 34, 4 (2015), 131–142.

Tobias Zirr and Carsten Dachsbacher. 2020. Path differential-informed stratified MCMC and adaptive forward path sampling. ACM Trans. Graph. (Proc. SIGGRAPH Asia) 39, 6 (2020), 1–19.

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. 2015. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. Comput. Graph. Forum (Proc. Eurographics STAR) 34, 2 (2015), 667–681.

A NETWORK ARCHITECTURE

The base channel number of the encoder-decoder is 48, and the base head number of the multi-head self-attention/cross-attention module is 1. The channel number and head number would be doubled as the feature resolution is halved. We use the same downsample and upsample method used in Zamir et al. [2022]'s method, which applies PixelUnshuffle and PixelShuffle after a 3×3 unbiased convolution layer. The skip connection in the decoding blocks concatenates the encoding output of the same shape and blends them with a 1×1 convolution.

In the denoising pipeline, the input resolution is adaptive, but the channel number is fixed at 22, including normalized radiance from 4 MLT components from the sample decomposition (12), distance buffer (1), albedo buffer (3), shading normals buffer (3) and world positions buffer (3). The output channel number of each decoding block is 432, which constitutes 4 sets of 5×5 temporal kernel, 9×9 kernel, and 2 blending weights. The last decoding block has fewer output channels since the output would not be composed upward.

B DATASET GENERATION

Scenes in our dataset are modified from a subset of the original OpenRooms dataset. We made slight modifications to the original scene configurations to tailor them to our research needs. We illustrate some example scenes of our dataset in Fig. 13. The following sections introduce how we modify the indoor scenes.

B.1 Geometry and Lights

We remove all indoor emitters and replace the outdoor HDR environment maps with area lights and spot lights outside the window. The fixed area lights keep the overall brightness of the room, and the moving spot lights create moving shadows and caustics.

B.2 Materials and Textures

We use 3 different types of materials on the indoor objects: glass, plastic and metal, accounting for 50%, 30% and 20% respectively. We use Mitsuba's smooth dielectric, rough plastic (with GGX distribution) and rough conductor (also with the GGX distribution) as the BSDF of these materials. We apply textures from the original OpenRooms to the plastic objects to closely replicate their appearance.

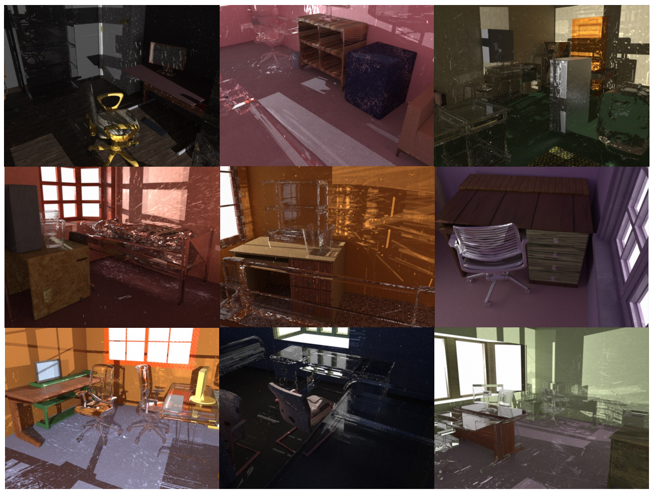


Fig. 13. Example scenes from our modified OpenRooms dataset.

B.3 Cameras and Animations

The original OpenRooms dataset offers an average of 30 camera configurations per scene. We generated an equivalent number of 60-frame animated sequences by interpolating between adjacent camera settings (including considering the last setting as adjacent to the first). For the part of the train dataset and validation dataset, we only rendered the initial 7 of the 60 frames for each camera configuration.

C METRICS FOR TEMPORAL STABILITY

We study different metrics' ability to measure temporal stability by evaluating them on the 60-frame animation of the scene *Monkey* [Ebke 2021]. We choose TPSNR [Balint et al. 2023; Hasselgren et al. 2020], TRMAE [Işık et al. 2021] used in previous work, and perceptual-informed metrics FovVideoVDP [Mantiuk et al. 2021]. We compare images rendered by MLT and bidirectional path tracing (BDPT) at a similar rendering time. The evaluation result in Table 5 shows that animations rendered by two algorithms have similar TPSNR and TRAME, and MLT even performs a little better. However, the FovVideoVDP shows that BDPT is much more temporally stable than MLT. We visualize the errors between the noisy renderings and reference for both algorithms at 3 consecutive frames in Fig. 14. The error maps show that MLT generates more concentrated errors and such errors fluctuate more intensely between frames, which finally causes the flickering artifacts that can be easily perceived by human eyes. Our study shows that FoVVDP is more consistent with the human perspective and explains the reason why we only use FoVVDP as the metric to evaluate temporal stability.

Table 5. Comparison of 3 different metrics that measure temporal stability. The metrics are measured on noisy images rendered by two algorithms.

Rendering Algorithm	TPSNR ↑	TRMAE↓	FoVVDP↑
MLT / 16spp	**16.626**	**2.146**	7.234
BDPT / 8spp	16.549	2.251	**8.630**

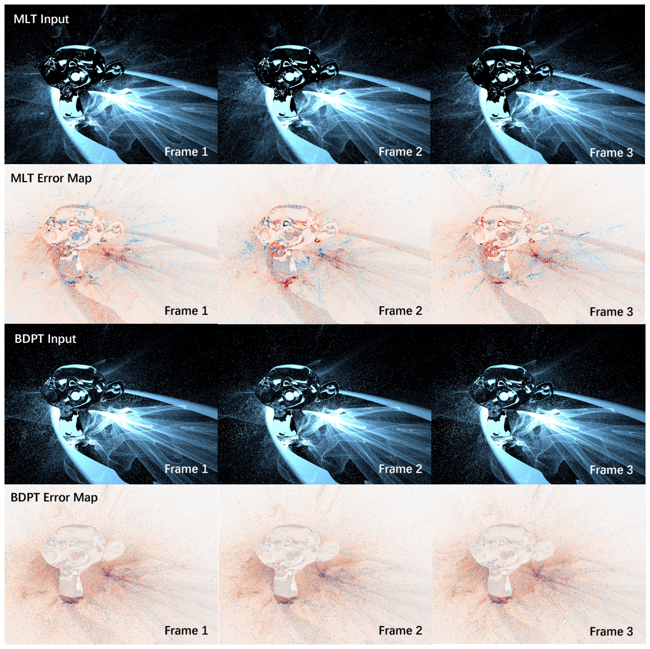


Fig. 14. The noisy renderings and their errors compared with the references. Blue is positive error and Red is negative error.