Joint Denoising and Upscaling via Multi-branch and Multi-scale Feature Network

PAWEL KAZMIERCZYK, Advanced Micro Devices, Inc., Poland
SUNGYE KIM, Advanced Micro Devices, Inc., USA
WOJCIECH USS, Advanced Micro Devices, Inc., Poland
WOJCIECH KALINSKI, Advanced Micro Devices, Inc., Poland
TOMASZ GALAJ, Advanced Micro Devices, Inc., Poland
MATEUSZ MACIEJEWSKI, Advanced Micro Devices, Inc., Poland
RAMA HARIHARA, Advanced Micro Devices, Inc., USA

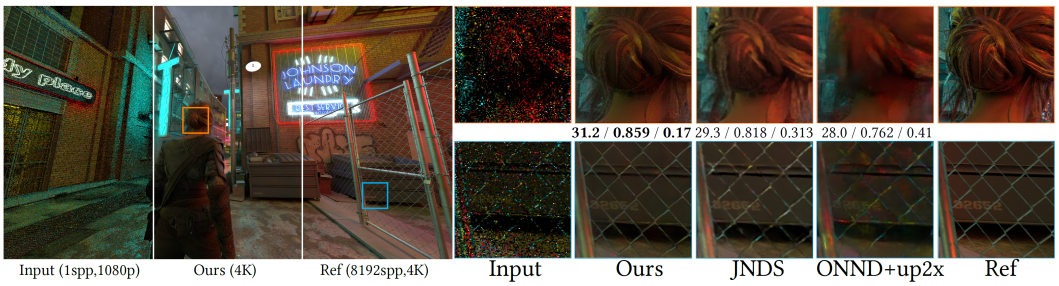


Fig. 1. Joint denoising and upscaling result of our technique in 4K resolution compared to the state-of-the-art denoising and upscaling methods, JNDS [Thomas et al. 2022] and ONND [NVidia 2021] with built-in 2× upscaling in OptiX™ SDK 8.0.0. Our result exhibits more details in the complex areas and outperforms in quantitative quality metrics PSNR↑/SSIM↑/LPIPS↓). Metrics are calculated on a full resolution image. Noisy 1spp input in 1080p resolution is shown in 2× upscaled for visualization purpose. Scene from the City Alley.

Deep learning-based denoising and upscaling techniques have emerged to enhance framerates for real-time rendering. A single neural network for joint denoising and upscaling offers the advantage of sharing parameters in the feature space, enabling efficient prediction of filter weights for both. However, it is still ongoing research to devise an efficient feature extraction neural network that uses different characteristics in inputs for the two combined problems. We propose a multi-branch, multi-scale feature extraction network for joint neural denoising and upscaling. The proposed multi-branch U-Net architecture is lightweight and effectively accounts for different characteristics in noisy color and noise-free aliased auxiliary buffers. Our technique produces superior quality denoising in a target resolution (4K), given noisy 1spp Monte Carlo renderings and auxiliary buffers in a low resolution (1080p), compared to the state-of-the-art methods.

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

Additional Key Words and Phrases: Multi-branch, U-Net, Denoising, Upscaling

Authors' Contact Information: Pawel Kazmierczyk, Pawel.Kazmierczyk@amd.com, Advanced Micro Devices, Inc., Poland; Sungye Kim, Sungye.Kim@amd.com, Advanced Micro Devices, Inc., USA; Wojciech Uss, Wojciech.Uss@amd.com, Advanced Micro Devices, Inc., Poland; Wojciech Kalinski, Wojciech.Kalinski@amd.com, Advanced Micro Devices, Inc., Poland; Tomasz Galaj, Tomasz.Galaj@amd.com, Advanced Micro Devices, Inc., Poland; Mateusz Maciejewski, Mateusz.Maciejewski@amd.com, Advanced Micro Devices, Inc., Poland; Rama Harihara, Rama.Harihara@amd.com, Advanced Micro Devices, Inc., USA.

ACM Reference Format:
Pawel Kazmierczyk, Sungye Kim, Wojciech Uss, Wojciech Kalinski, Tomasz Galaj, Mateusz Maciejewski, and Rama Harihara. 2025. Joint Denoising and Upscaling via Multi-branch and Multi-scale Feature Network. *Proc. ACM Comput. Graph. Interact. Tech.* 8, 1, Article 16 (May 2025), 18 pages. https://doi.org/10.1145/3728297

1 Introduction

Monte Carlo (MC) rendering has been ubiquitously used in generating photorealistic images. It is stochastically unbiased and eventually converges toward an accurate radiance. However, MC integration requires thousands of samples per pixel (spp) to produce visually appealing noise-free images. The source of the challenge lies in the inherent randomness of the samples that can result in scattered rays failing to hit a light source, even after multiple bounces. Even when all rays hit a light source, approximating the rendering equation integral [Kajiya 1986] with a finite number of samples can still fall short of producing high-quality images and show visually annoying noise. The higher number of samples per pixel, the higher chance of less noise in an image. However, it is not viable to use such large numbers of samples per pixel to render high-quality path-traced images in real-time gaming due to prohibitively expensive computational costs.

MC denoising has played a pivotal role to address the problem of the high number of samples required in MC path tracing by reconstructing high-quality pixels from a noisy image rendered with low samples per pixel within a limited time budget. Denoisers, in general, take noisy diffuse and specular radiance signals as input, often accompanied by auxiliary buffers like albedo and normal, to denoise them separately with different filters. The resulting denoised signals are then composited into a final color to better preserve fine details. As such, many real-time rendering engines include multiple denoising filters for each noisy signal from diffuse lighting, reflection, and shadows. With significant advances in neural techniques in recent years, deep learning-based MC denoising [Bako et al. 2017; Balint et al. 2023; Huo and eui Yoon 2021; Işık et al. 2021; Thomas et al. 2022; Vogels et al. 2018] has shown remarkable progress in denoising quality. By predicting denoising filter weights in a process of training on a large dataset, neural denoisers achieve high-quality renderings compared to hand-crafted analytical denoisers [Schied et al. 2017].

As deep learning-based supersampling techniques have become widely adopted in the real-time rendering industry to improve framerates by rendering at a lower resolution, unified denoising and upscaling methods have emerged to further improve framerates for real-time ray tracing. However, the quality may deteriorate when a denoiser and upscaler are naively combined, as the denoiser eliminates the subpixel viewport offset at low resolution, which a upscaler relies on. Hence, employing a single neural network for joint denoising and supersampling [Thomas et al. 2022] offers the advantage of sharing learned parameters in the feature space, enabling efficient prediction of filter weights for both denoising and upscaling.

In this work, we propose a joint neural denoising and upscaling technique for path-traced renderings with very low samples per pixel (1spp). Our technique produces spatio-temporally stable denoising results at a higher resolution through our sophisticatedly designed feature extraction neural network. First, we introduce a multi-branch U-Net (MUNet) to take account of different characteristics in a noisy radiance input and noise-free, aliased auxiliary buffers like albedo, normal, and roughness. Oh and Moon [2024] use a transformer block with joint self-attention to consider two input sources with different properties. However, transformer blocks remain computationally expensive in spite of adopting common optimization methods. The number of learnable parameters in their denoising framework is 33.35 M. Our MUNet, however, is lightweight with 0.644 M learnable parameters for joint denoising and upscaling. Second, as a feature extraction network, our MUNet generates multi-scale features from multi-branch, from which we derive spatial and temporal filtering weights for temporal accumulation, upscaling, and multi-scale denoising. Third, our technique

replaces multiple denoisers used for different lighting effects in MC rendering by denoising all noisy signals as well as upscales to a target resolution in a single pass, similar to Thomas et al. [2022]. However, our approach does not require separated diffuse and specular noisy signals as input since we derive radiance composition weights from the features our MUNet learns for implicit radiance decomposition. Finally, we evaluate our technique on a diverse dataset with complex geometry and lighting effects in two scenarios: joint denoising and upscaling, and denoising-only (1× upscaling). Our results demonstrate greater efficacy in producing high-quality fine details compared to the state-of-the-art techniques. The contributions of our technique are summarized by the following:

- A lightweight multi-branch feature extraction network to incorporate different characteristics in noisy radiance input and noise-free, aliased auxiliary buffers.
- Implicit noisy radiance decomposition to eliminate the need for separate noisy radiance signals in input.
- A state-of-the-art joint MC denoising and upscaling technique generating superior fine-details via a multi-branch and multi-scale feature network.

2 Related Work

Deep learning-based denoising techniques for MC renderings have been actively researched over a decade with great success [Back et al. 2022; Bako et al. 2017; Balint et al. 2023; Chaitanya et al. 2017; Fan et al. 2021; Gu et al. 2024; Işık et al. 2021; Kalantari et al. 2015; Meng et al. 2020; Thomas et al. 2022; Vogels et al. 2018; Xu et al. 2019; Zhang et al. 2024]. While analytical denoising filters [Dammertz et al. 2010; Koskela et al. 2019; Kozlowski and Cheblokov 2021; Schied et al. 2018, 2017; Zhdan 2021] and their variants continue to be widely used for real-time rendering, neural denoisers generally yield higher-quality results. In addition, neural denoisers are getting more attention for real-time path tracing [Kandar and Sjoholm 2024; Murphy et al. 2024] when an upscaling technique is taken into account [NVidia 2023; Thomas et al. 2022]. In this section, we focus on recent works relevant to our research. For a wider overview of deep-learning based denoising techniques, we refer to comprehensive surveys by Huo and Yoon [2021].

Kernel prediction-based neural denoisers [Bako et al. 2017; Gharbi et al. 2019; Vogels et al. 2018] have demonstrated improved fine detail preservation by utilizing large filtering kernels, achieving better quality at the cost of performance and memory. Other optimized techniques [Fan et al. 2021; Işık et al. 2021; Meng et al. 2020; Thomas et al. 2022] have also been researched. Meng et al. [2020] employ a lightweight network architecture to guide a bilateral grid filter. Işık et al. [2021] predict per-pixel feature vectors to generate filter kernels. Fan et al. [2021] predict an encoding of a per-pixel kernel as a compact single-channel representation which is decoded to create filtering kernels to address a heavy inference overhead when predicting filters with large size kernels. With our lightweight multi-branch feature network, we predict per-pixel feature vectors to derive multiple filtering kernels.

Auxiliary buffers, such as geometric and material features from G-buffers, are frequently employed in feature-guided denoising by concatenating them with a noisy radiance as input. Yang et al. [2019] focus on redundant information in auxiliary buffers and use a dual encoder U-Net to extract useful information from auxiliary buffers to guide the denoised output reconstruction. Xu et al. [2019] present a conditioned feature modulation to integrate separately encoded auxiliary features into a denoising network for better utilization of auxiliary features throughout denoising network layers. Oh and Moon [2024] take different characteristics in noisy radiance and auxiliary features into account and generate dual attention scores from a noisy radiance and auxiliary features in a transformer-based denoising framework, achieving improved denoising quality with detailed image feature structures. Inspired by the previous studies on recognizing different nature in a noisy

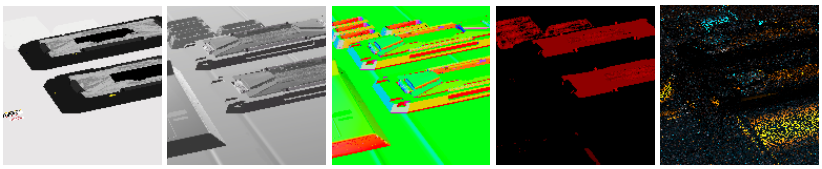


Fig. 2. Different characteristics in rendered buffers. Noise-free, aliased diffuse albedo, specular albedo, normal, and roughness (from left to right). Noisy color (right). Scene from ZeroDay [Winkelmann 2019].

radiance and auxiliary features, we propose a multi-branch U-Net as a feature extraction network. Different from the prior works [Oh and Moon 2024; Yang et al. 2019], our technique is designed for spatio-temporally stable joint denoising and upscaling.

U-Net [Ronneberger et al. 2015] is widely used for image reconstruction problems thanks to its large receptive fields. Several variants of U-Net have been researched in order to improve image reconstruction quality, such as dual encoders [Yang et al. 2019], dual U-Nets [Thomas et al. 2020], attention modules and transformer blocks [Chen et al. 2023b, 2024; Lin et al. 2021; Oh and Moon 2024; Yu et al. 2021]. Moreover, the multi-scale filtering kernels learned from the U-Net architecture are widely adopted to take the benefit from large receptive fields and avoid a performance impact from large filter kernels for kernel prediction-based denoising [Balint et al. 2023; Thomas et al. 2022; Vogels et al. 2018]. We build our feature extraction network by using a multi-branch approach on top of a U-Net architecture, one branch for noisy radiance and the other branch for noise-free, aliased guiding buffers, which generates multi-branch, multi-scale features for denoising and upscaling filter kernels.

Instead of denoising noisy color images, many prior works either decompose the noisy color in a data pre-processing stage or require separated diffuse and specular signals as input to predict denoising filter kernels for each noisy signal, achieving considerable quality improvements [Bako et al. 2017; Thomas et al. 2022]. Learning radiance decomposition with a neural network [Zhang et al. 2021] is less intrusive when working with noisy color as input, rather than taking separated noisy diffuse and specular radiance. Inspired by Zhang et al. [2021], our technique takes a noisy color from MC rendering as input, not requiring to have diffuse and specular signals separately. However, different from their work, we predict radiance composition weights for an implicit radiance decomposition, then apply the predicted radiance composition weights after denoising filtering.

Denoising often removes or reduces salient information that might be useful for subsequent post-processing steps like resolution upscaling. Thomas et al. [2022] present joint denoising and supersampling by predicting multi-scale denoising filter kernels and upscaling filter kernels from a single network architecture. Such joint denoising and upscaling with a single neural network offers the advantage of sharing learned parameters in the feature space to efficiently predict denoising filter kernels and upscale filter kernels. Moreover, rendering at a low resolution with low samples per pixel provides a performance benefit, allowing for more time budget to be allocated to neural denoising in reconstructing high-quality pixels. DLSS Ray Reconstruction [NVidia 2023] appears to use equivalent techniques, but detailed information remains unavailable to the public. Our technique aims to achieve the same objective in this context, as we propose a technique to jointly solve MC denoising and resolution upscaling. However our technique is distinguished from Thomas et al.'s work by our contributions.

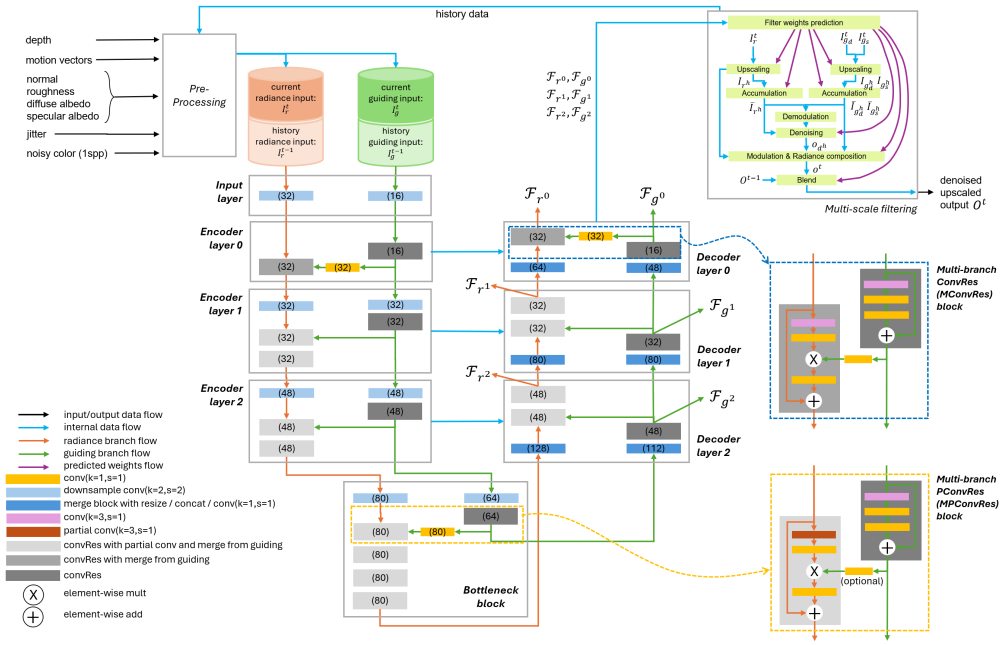


Fig. 3. Overview of our technique with detailed structure of our multi-branch feature extraction network. Input processing prepares input to our neural network. Multi-branch U-Net architecture (MUNet) is designed with the radiance and guiding branches. Encoder and decoder layers are composed of MConvRes and MPConvRes blocks. Our network learns multi-scale features from decoder layers for radiance and guiding separately, which are used in the multi-scale filtering stage to denoise the noisy color image and upscale to a target resolution. The number of output features from each layer is shown in parentheses. The ReLU activations are used, but not presented for the sake of brevity.

3 Our Approach

We propose a technique to achieve high-quality denoising for MC renderings with one sample per pixel while jointly upscaling resolution. To do this, we introduce a multi-branch, multi-scale feature extraction network architecture.

3.1 Motivation

There exists little research [Oh and Moon 2024] taking different characteristics in the auxiliary buffers into account for neural denoising. For instance, auxiliary buffers (diffuse albedo, specular albedo, normal, roughness) are not noisy but aliased as shown in Figure 2. We are inspired to construct a multi-branch U-Net (MUNet) to account for such a difference, in which a *guiding branch* extracts features from auxiliary guiding buffers separated from a *radiance branch* for a noisy radiance input within a single feature extraction network.

Since the auxiliary buffers are not noisy but just aliased, and contain redundant information [Yang et al. 2019], we structure relatively small and shallow layers for the guiding branch. This enables the creation of a lightweight feature extraction network. The features from the guiding branch are merged into the radiance branch per layer to assist feature extraction from noisy radiance. Figure 3 presents an overview of our technique with focus on the MUNet. Our technique consists of three stages: pre-processing (Section 3.2), multi-branch and multi-scale feature extraction (Section 3.3), and multi-scale filtering for joint denoising and upscaling (Section 3.4).

3.2 Pre-processing

The pre-processing stage prepares input for the MUNet from the current and history data. In Figure 3 (top-left), data from the current frame includes a noisy 1spp color and multiple guiding buffers like normal, roughness, diffuse/specular albedo. We use μ-Law tonemap, as suggested by Kalantari and Ramamoorthi [2017], to tonemap the noisy color in a high-dynamic range to the range [0.0,1.0]. The noisy color and guiding buffers are then unjittered by the camera jitter values from rendering which are scaled by 2× to produce noisy radiance (I_r^t) and guiding buffers (I_g^t) in a target resolution.

Our technique exploits a recurrent architecture where history data from the previous output is reused as input for the current frame. Given the motion vectors at the same low resolution as the noisy color, we upscale the motion vectors to the target resolution after dilating the motion vectors using a 3×3 filter and depth. The history data is reprojected by the motion vectors to align to the current frame, (I_r^t), in a target resolution. Our history data includes temporally accumulated radiance and diffuse/specular albedo. We also maintain the number of accumulated samples per pixel that is weighted by the predicted accumulation weights for radiance and diffuse/specular albedo in the multi-scale filtering stage. We use Sinusoidal positional encoding, which was proposed in the Transformer architecture [Vaswani et al. 2017], to generate higher-dimensional embeddings from the weighted number of accumulated samples as history data. Hence, we prepare reprojected history radiance (I_r^{t-1}) and history guiding input (I_g^{t-1}). Finally, the MUNet uses (I_r^t,I_r^{t-1}) for the radiance branch and (I_g^t,I_g^{t-1}) for the guiding branch.

3.3 Multi-branch, Multi-scale Feature Extraction

We devise the MUNet to incorporate different characteristics in a noisy color input and noise-free, aliased guiding buffers while learning features for denoising and upscaling filtering. As shown in Figure 3, our feature extraction network is composed of three encoder and decoder layers with a U-Net architecture. We use 2×2 convolution with a stride of 2 to downsample the input features for encoder layers as well as the input layer. In decoder layers, 2× nearest-neighbor interpolation is used to upsample the input features. In each layer, we devise a multi-branch convolutional residual (MConvRes) block for separated radiance and guiding branches. The output features from the guiding branch is merged to the residual features in the radiance branch through an element-wise multiplication. 1×1 convolution is optionally added to match the number of channels from the guiding branch to the radiance branch.

To create a lightweight network, we use a multi-branch partial convolutional residual (MPConvRes) block for all layers except the first encoder and the last decoder layers (Encoder layer 0 and Decoder layer 0). The MPConvRes block is a variant of the MConvRes with a partial convolution where we apply convolution to the portion of input features in a channel dimension (e.g., first half channels) and directly concatenate the features in remaining channels with the convolution output. Our MPConvRes block can be viewed as an extension of the partial ConvRes block [Chen et al. 2023a] for a multi-branch network. The first encoder and the last decoder layers are more sensitive to the low-level image features. Hence, we retain the MConvRes block for them to avoid any loss of information when using the partial convolution. Inner encoder and decoder layers are less sensitive to quality when utilizing the MPConvRes blocks.

The guiding branch consists of smaller number of convolutional residual (ConvRes) blocks and output features than the radiance branch. Since the auxiliary guiding buffers are noise-free, aliased and contain redundant information, our insight is that a small-scale configuration for the guiding branch is sufficient, which also helps us create a lightweight feature extraction network.

Our network learns multi-scale radiance features ($\mathcal{F}_{r^0}$,$\mathcal{F}_{r^1}$,$\mathcal{F}_{r^2}$) and multi-scale guiding features ($\mathcal{F}_{g^0}$,$\mathcal{F}_{g^1}$,$\mathcal{F}_{g^2}$) from three decoder layers. We demonstrate the benefit of our multi-branch network

by comparing against a single branch U-Net with the equivalent number of parameters in Figure 7 and Table 4.

3.4 Multi-scale filtering for joint denoising and upscaling

Given the pair of multi-branch and multi-scale radiance and guiding features ($\mathcal{F}_{r^0}$,$\mathcal{F}_{g^0}$), ($\mathcal{F}_{r^1}$,$\mathcal{F}_{g^1}$) and ($\mathcal{F}_{r^2}$,$\mathcal{F}_{g^2}$) learned from the MUNet, we first predict per-pixel feature vectors by 1×1 convolution in three different scales and derive weights for 3×3 Gaussian filters for denoising. We also resize ($\mathcal{F}_{r^0}$,$\mathcal{F}_{g^0}$) to generate radiance and guiding features at a target resolution, ($\mathcal{F}_{r^h}$,$\mathcal{F}_{g^h}$). Similarly, we then use 1×1 convolution to predict per-pixel feature vectors to derive weights for multiple filters at a target resolution, which we utilize for upscaling, temporal accumulation, denoising, implicit radiance decomposition, and blending. We adapt the partitioning pyramids approach [Balint et al. 2023] for our denoising filtering.

Based on the predicted per-pixel feature vectors at a target resolution, we apply a 3×3 Gaussian filter to the noisy 1spp color input after upscaling with nearest neighbor interpolation (I_{r^h}). We also accumulate the upscaled noisy color (I_{r^h}) with the temporally accumulated radiance in our history data (I_r^{t-1}) by using the predicted accumulation weight for radiance. This results in the upscaled and temporally accumulated noisy color ($\bar{I}_{r^h}$). We apply the same operations to diffuse and specular albedo in the guiding buffers, but with filters derived for guiding buffers. The upscaled and temporally accumulated noisy color ($\bar{I}_{r^h}$) is then demodulated by the upscaled, accumulated diffuse and specular albedo ($\bar{I}_{g_d{}^h}$,$\bar{I}_{g_s{}^h}$), as implicit radiance decomposition, producing the input to the denoising filtering ($\mathcal{D}()$). We apply the radiance composition weights (ρ,σ,τ), predicted from the ($\mathcal{F}_{r^h}$,$\mathcal{F}_{g^h}$), to the output of the denoising filtering in channel dimension and modulate back by ($\bar{I}_{g_d{}^h}$,$\bar{I}_{g_s{}^h}$). This is composited with the upscaled color (I_{r^h}) by the predicted composition weight (κ). Finally, we blend the output (o^t) with the history output (O^{t-1}) based on the predicted history accumulation weight (ω) to produce the final output (O^t).

$$
\begin{aligned}
o_{d^h} &= \mathcal{D}(concat[\frac{\bar{I}_{r^h}}{\bar{I}_{g_d{}^h}}, \frac{\bar{I}_{r^h}}{\bar{I}_{g_s{}^h}}, \bar{I}_{r^h}]),\\
o^t &= \rho(o_{d^h}[0:3])\bar{I}_{g_d{}^h} + \sigma(o_{d^h}[3:6])\bar{I}_{g_s{}^h} + \tau(o_{d^h}[6:9]) + \kappa I_{r^h},\\
O^t &= lerp(o^t, O^{t-1}, \omega).
\end{aligned}
$$

4 Implementation

4.1 Dataset generation

We collect a dataset from diverse complex scenes rendered using the Falcor framework [Kallweit et al. 2022]. We render temporal sequences of thousands of frames from different camera viewpoints with diverse lighting conditions. Noisy color images are path-traced in 1920×1080 resolution with 1spp and the ReSTIR Direct Illumination [Bitterli et al. 2020] enabled. We carefully review our data images to confirm that they do not have strong correlation artifacts perceptually. All guiding buffers (normal, roughness, diffuse albedo, specular albedo), depth and motion vectors are collected in the same resolution to the noisy color image. The camera jitter values are captured per frame. Reference images are path-traced in 3840×2160 resolution with 8192 spp. We divide the collected frames into train, validation and test datasets that have different camera trajectories and/or lighting conditions. Table 1 shows the scenes and the number of frames we collect from each scene for train, validation and test dataset. All results we present are from the test dataset.

Table 1. Our data scenes. Bistro [Lumberyard 2017], ZeroDay [Winkelmann 2019], other scenes from Unreal Engine marketplace with our modification. For data diversity, we render three scenes with two different lighting. We divide collected frames into train, validation and test datasets that have different camera trajectories. We also show average render time per frame with 1spp in 1920×1080 and 8192 spp in 3840×2160.

Scene	Training frames	Validation frames	Testing frames	1spp, 1080p (milliseconds)	8192 spp, 4K (minutes)
Bistro-Day	1998	160	1417	10.98	5.19
Bistro-Night	1577	-	1998	10.76	5.26
ZeroDay-MeasureOne	480	-	320	5.68	2.91
ZeroDay-MeasureSeven	399	-	480	6.15	3.13
Victorian Train-Base	1160	160	960	7.09	3.55
Victorian Train	1120	-	1160	7.20	3.58
City Alley	1958	-	1625	8.68	4.12
Evermotion (Pool house)	970	160	640	8.89	4.63
Sicka-Mansion	-	-	100	7.10	3.48
Museum	-	-	100	6.43	3.20

4.2 Training

We implement our technique in PyTorch [Ansel et al. 2024] and train on AMD Instinct™ MI210 GPUs for 120 epochs with a batch size of 4. We use the Adam [Kingma and Ba 2014] optimizer with an initial learning rate of 8e-4 and weight decay of 1e-2. During training, we use 384×384 input patches cropped from our training dataset. For the temporal accumulation in our recurrent architecture using a history output, we use 30 consecutive frames in the same batch.

4.3 Loss functions

We calculate a final loss ($\mathcal{L}_O$) by the weighted sum of spatial loss ($\mathcal{L}_S$), temporal loss ($\mathcal{L}_T$), and perceptual loss ($\mathcal{L}_P$), where we set $\alpha = 1.0$, $\beta = 1.0$, and $\gamma = 0.1$ for each loss. We gradually increase the temporal loss weight β up to 2.0 to emphasize temporal stability over training epochs.

$$
\begin{aligned}
\mathcal{L}_O &= \alpha\mathcal{L}_S + \beta\mathcal{L}_T + \gamma\mathcal{L}_P, \\
\mathcal{L}_S &= \mathcal{L}_s^o + \mathcal{L}_s^g, \\
\mathcal{L}_T &= L_1(\nabla(O^t, \mathcal{W}(O^{t-1}, I^t_{m^h})), \nabla(R^t, \mathcal{W}(R^{t-1}, I^t_{m^h}))),
\end{aligned}
$$

where our spatial loss $\mathcal{L}_S$ is also the weighted sum of different losses: $\mathcal{L}_s^o$ for a color output and $\mathcal{L}_s^g$ for guiding albedo buffers in a target resolution. For $\mathcal{L}_s^o$, we use the SMAPE (Symmetric Mean Absolute Percentage Error) between the output (O^t) and reference (R^t) images. For $\mathcal{L}_s^g$, we calculate the sum of L_1 losses for the upscaled diffuse and specular albedo buffers over reference diffuse and specular albedo buffers. $\mathcal{L}_T$ is calculated by L_1 of temporal gradient ($\nabla()$) between the current output at time t and the previous output at time $t-1$ after reprojecting ($\mathcal{W}()$) the previous output (O^{t-1}) and reference (R^{t-1}) images with the upscaled motion vectors ($I^t_{m^h}$). We use the LPIPS [Zhang et al. 2018] for $\mathcal{L}_P$.

5 Results and Discussion

We evaluate and discuss the quality of our technique for multiple test dataset including unseen test scenes presented in Table 1. The test dataset is not included in training our network model.

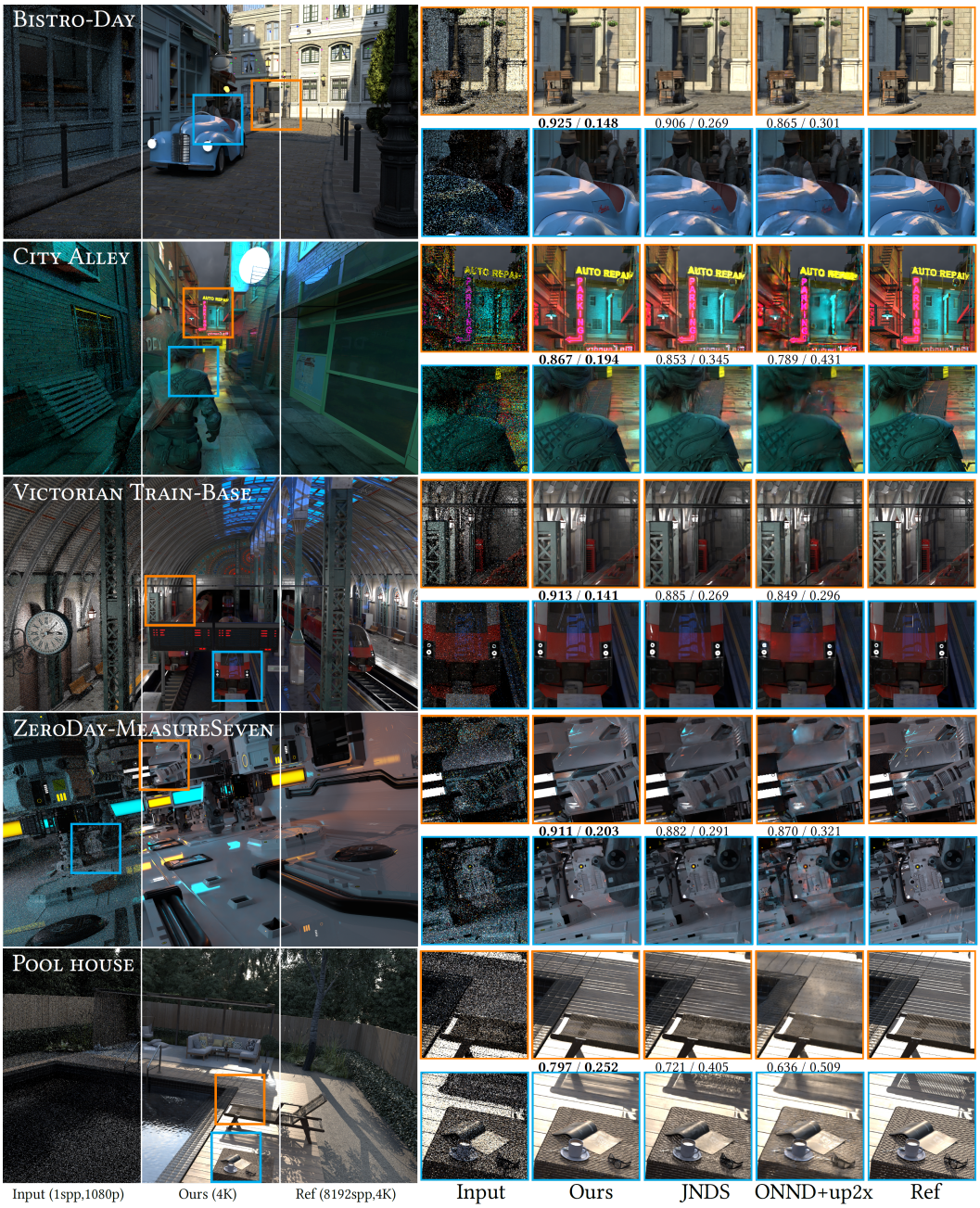


Fig. 4. Denoising and upscaling quality comparison from test dataset in 4K against prior works, JNDS [Thomas et al. 2022] and ONND [NVidia 2021] with a built-in 2× upscale. Quality metrics (SSIM↑/LPIPS↓) are calculated on a full resolution image. We denote in bold the best quality score. Noisy input in 1080p is shown in 2× upscaled for visualization purpose. Our results outperform the prior techniques for these test dataset in visual and quantitative quality.

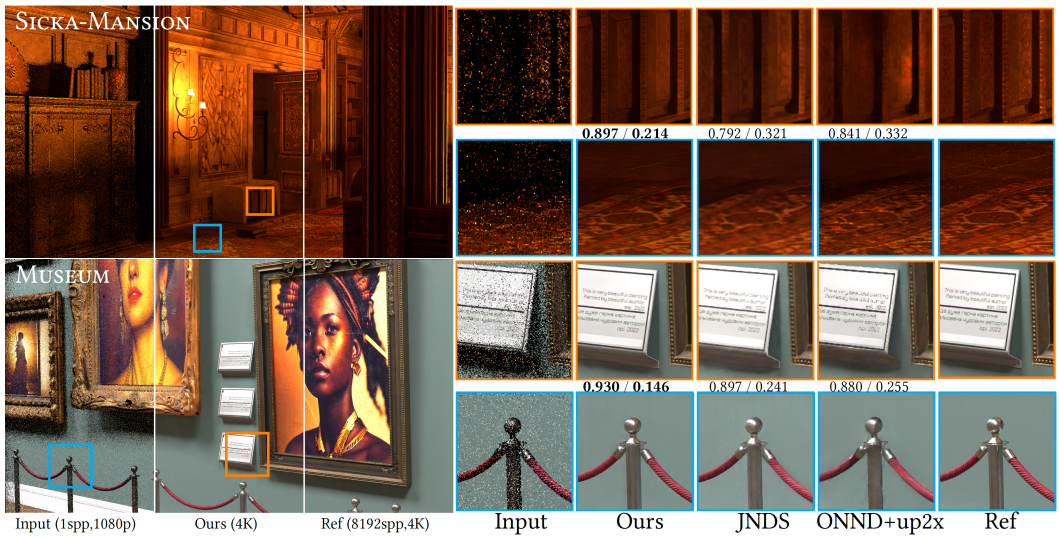


Fig. 5. Denoising and upscaling quality comparison from unseen test scenes in 4K against prior works, JNDS [Thomas et al. 2022] and ONND [NVidia 2021] with built-in 2× upscale. Quality metrics (SSIM↑/LPIPS↓) are calculated on a full resolution image. We denote in bold the best quality score. Noisy input in 1080p is shown in 2× upscaled for visualization purpose. Our results outperform the prior techniques for these unseen scenes in visual and quantitative quality.

5.1 Joint Denoising and Upscaling Quality

We compare our denoised, upscaled results against two state-of-the-art prior works, JNDS [Thomas et al. 2022] and ONND [NVidia 2021]+up2x, which support both denoising and upscaling. We implement the JNDS based on the details published by Thomas et al. [2022] and employ the perceptual loss function implemented by Balint et al. [2023]. We train the JNDS with the same training dataset for fair comparison. We generate the ONND+up2x results from the OptiX™ SDK 8.0.0 with a built-in 2× upscaling option. The noisy 1spp color image is used as input for our technique and ONND+up2x. We use separated noisy 1spp diffuse and specular radiance images as input for JNDS as described in the paper. All input and guiding buffers have resolution 1920×1080 and the target image has resolution 3840×2160 for 2× upscale.

In Figure 4, our method shows superior visual quality to competing techniques by reconstructing better high frequency details in various test datasets as highlighted in insets. Our results present better thin objects, complex geometry details, glossy reflection and texture patterns without extensive blurring than other methods. We also compare quantitative quality by calculating widely used image quality metrics, structure-oriented SSIM↑ and perception-oriented LPIPS↓. For all test dataset in Figure 4, our method shows better SSIM and LPIPS metrics than other methods. We also present quality comparison for unseen test scenes to further highlight the generalization capability of our method to new scenes. Any frames from these unseen test scenes (Sicka-Mansion, Museum) are not included in training. In Figure 5, our results outperform other techniques in visual and quantitative quality with superior geometry and texture details.

5.2 Denoising-only Quality

We also compare our technique with the state-of-the-art neural denoisers including ONND [NVidia 2021], OIDN [Áfra 2024], and JSA [Oh and Moon 2024]. We train our technique with 1× upscale

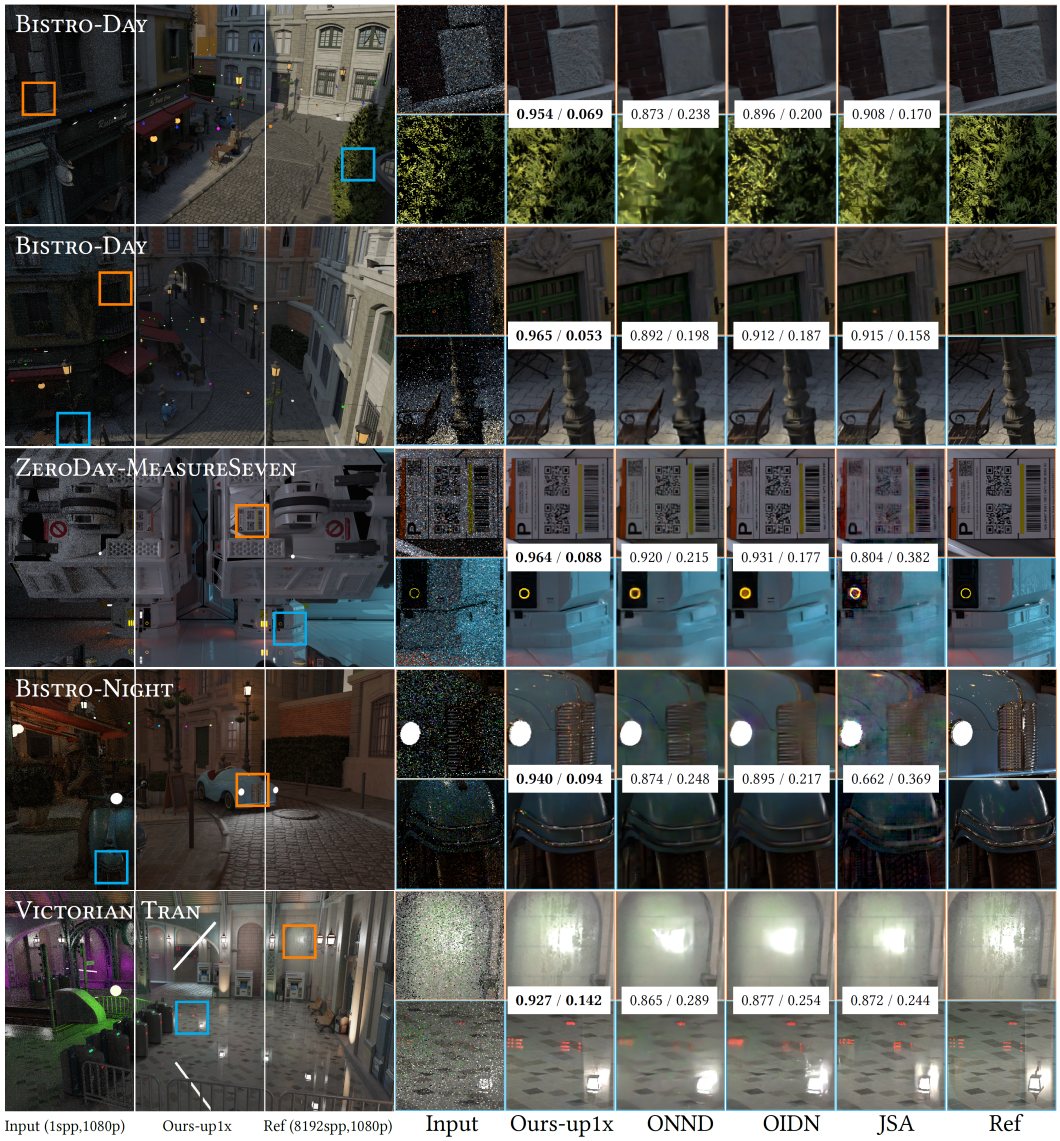


Fig. 6. Denoising-only quality comparison in 1920×1080 resolution from test dataset against the state-of-the-art denoisers. Our technique is trained with 1× upscale to generate denoising-only results without upscaling. For ONND [NVidia 2021], we use the OptiX™ SDK 8.0.0. For OIDN [Áfra 2024], we use a pre-trained model from the publicly available pre-built binary. JSA [Oh and Moon 2024] is trained with our training dataset. Quality metrics (SSIM↑/LPIPS↓) are calculated on a full resolution image. We denote in bold the best quality score. Our method shows superior quality in preserving fine-details in denoising results. Other methods overblur high-frequency details and glossy reflection. JSA shows noticeable artifact for dark scenes like ZeroDay and Bistro-Night.

Table 2. Temporal quality comparison of our supplemental video results for test dataset. We measure metrics using FovVideoVDP. Higher score is better. Our results present better temporal quality metrics than other methods for these test dataset including the unseen test scene.

FovVDP↑	Ours	JNDS	ONND+up2x
City Alley	**5.9864**	5.7055	5.1658
Bistro-Day	**7.1451**	6.6358	6.6695
ZeroDay-MeasureSeven	**6.8204**	6.5874	6.2736
Sicka-Mansion	**6.9806**	6.7798	6.1398

to compare denoising-only quality in the 1920×1080 input resolution. We use a pre-built binary for ONND from the OptiX™ SDK 8.0.0 without an upscale option. We use a pre-built binary for OIDN from the latest version 2.3.0 with the high quality mode. We use a noisy 1spp color image as input with albedo and normal as auxiliary buffers in 1920×1080 resolution for ONND and OIDN. JSA [Oh and Moon 2024] is a transformer-based denoising technique with a self-attention through dual attention scores to consider different characteristics in noisy radiance and auxiliary buffers. We train the Pytorch implementation published by the authors with our training dataset after modifying the data loader.

Figure 6 demonstrates that our technique achieves superior denoising quality with sharper details compared to the state-of-the-art neural denoisers. Other techniques generate extensively over-blurred pixels in many results. Our results in the Bistro-Day successfully preserve the fine-details in the rough wall and high-frequency foliage. For highly reflective surfaces like the ZeroDay scene, all techniques show low quality with loss of details. However, our technique is able to better reconstruct the details in reflections on glossy surfaces than other methods as presented in ZeroDay, Bistro-Night and Victorian Train scenes. JSA shows more high-frequency details than ONND and OIDN in Bistro-Day and Victorian Train scenes but introduces noticeable artifacts for dark scenes like ZeroDay and Bistro-Night.

5.3 Temporal Stability

We demonstrate strong temporal stability of our results by the supplemental video comparing with other methods. To achieve temporally stable quality, our technique utilizes history reprojection by motion vectors (Section 3.2), temporal accumulation of radiance, diffuse albedo and specular albedo ($\bar{I}_{r^h}$, $\bar{I}_{g_d{}^h}$, $\bar{I}_{g_s{}^h}$ in Figure 3) (Section 3.4), and temporal loss (Section 4.3) as described in each section. Note that our temporal accumulation is performed with predicted accumulation filtering weights based on the learned features from our multi-branch feature extraction.

In Table 2, we present quantitative temporal quality comparison of our supplemental video results by perception-informed FovVideoVDP[1] [Mantiuk et al. 2021] that is a video quality metric capturing temporal distortion and flickering artifacts perceivable by human. In both visual and quantitative metric, our results show better temporal quality than other methods for the test dataset including the unseen test scene (Sicka-Mansion).

5.4 Computational Overheads

We believe that apple-to-apple runtime performance comparison is not feasible with different implementation of methods. ONND [NVidia 2021] is from the OptiX™ SDK. OIDN [Áfra 2024] in the released binary is CPU-based. Ours, JNDS [Thomas et al. 2022] and JSA [Oh and Moon 2024] are in PyTorch implementation without inference runtime optimization like layer fusions and

[1]FovVideoVDP v1.2.0, 75.4 pix/deg, $Lpeak = 200$, $Lblack = 0.5979$ cd/m^2, non-foveated

Table 3. The number of trainable parameters in millions and GFLOPs for our technique and comparative methods. We also show the runtime of our technique compared to JNDS and JSA, all of which are in unoptimized PyTorch implementation, in milliseconds (ms) on AMD Instinct™ MI210 GPU. Note that ours and JNDS are for joint denoising and upscaling to 4K resolution, and other methods are denoising-only in 1080p resolution.

	Ours	JNDS	JSA	ONND	OIDN
# of params (M)	0.644	2.7	33.35	1.5	0.92
GFLOPs	396	386	8206	-	524
Feature extraction network time (ms)	53	51	-	-	-
Inference time (ms)	248	137	3760	-	-

quantization. Hence, to estimate computational overheads in inference, we compare the number of trainable parameters based on information published from each baseline technique and the number of floating-point operations (FLOPs) in Table 3. While a smaller number of parameters and FLOPs may indicate a faster inference time, it is important to note that there is not always a direct relationship between these quantities and inference runtime. For ONND, we calculate the number of parameters based on the work by Chaitanya et al. [2017] that is used in Optix version 5, but there is no information for the latest version. For OIDN, we calculate the number of parameters from the UNet of OIDN. We collect GFLOPs using the PyTorch profiler during inference with 1920×1080 render resolution and 2× upscaling to 3840×2160.

Although the runtime profiling from the PyTorch implementation does not represent optimized inference runtime, we show the GPU time on AMD Instinct™ MI210 for our technique compared to JNDS and JSA, all of which are implemented in PyTorch (FP32) without any inference optimizations, to provide insights into how our technique scales compared to others. For our technique and JNDS, the feature extraction network time represents the time used by the neural network: MUNet for ours and the feature extractor U-Net for JNDS. Inference time represents GPU time for per-frame inference. These timings do not vary across different scenes but change for different resolutions, as they are screen-space techniques.

As shown in Table 3, our proposed multi-branch network architecture is lightweight with only 0.644 M parameters which is much smaller than other methods. Our network architecture presents smaller GFLOPs than OIDN and similar to JNDS. We observe similar timings between the proposed MUNet in our technique and the feature extraction U-Net in JNDS. For end-to-end inference time per frame, our technique shows 1.8× longer GPU time than JNDS, where we perform multi-scale filtering for denoising and filtering for upscaling. We expect our technique to achieve close to real-time performance when implementing common inference optimizations, such as layer fusions and low-precision quantization, on contemporary GPUs while offering superior quality to the state-of-the-art prior techniques.

To illustrate the potential savings by rendering at 1spp in 1920×1080 resolution compared to 8192 spp in 3840×2160 resolution, we also present the average render time per frame of our data scenes in Table 1. Based on the render time shown in Table 1, rendering with 1spp in a lower resolution takes only a few milliseconds per frame. Our technique can be applied to denoise and upscale, producing high-quality images at the target resolution.

5.5 Multi-branch vs. single branch

We perform an ablation study to demonstrate the advantage of our multi-branch U-Net architecture compared to a standard single branch U-Net in joint MC denoising and upscaling guided by auxiliary

Fig. 7. Results from our technique with the proposed multi-branch U-Net (top inset) and a single branch U-Net (middle inset) for feature extraction. Reference (bottom inset) is also shown for comparison. Although overall quality converges similarly, our multi-branch U-Net is more robust in reconstructing complex areas as highlighted in insets. Scenes from Bistro-Day [Lumberyard 2017] (left) and Evermotion Pool House (right).

Table 4. Quality metrics comparison between our multi-branch U-Net and single-branch U-Net for test dataset in Figure 7. We average quality metrics from a sequence of frames for PSNR↑, SSIM↑ and LPIPS↓. We also show FovVDP↑ as a temporal quality metric. All metrics are calculated on full resolution images. We denote in bold the best quality score.

	Ours w/ multi-branch		Ours w/ single-branch	
	Bistro-Day	Pool House	Bistro-Day	Pool House
FovVDP↑	**7.649**	5.898	7.612	**5.992**
PSNR↑	**35.868**	**30.162**	35.671	30.024
SSIM↑	**0.920**	**0.797**	0.918	0.791
LPIPS↓	**0.134**	**0.240**	0.142	0.248

buffers. The pre-processing and multi-scale filtering stages in our technique are reused. To do this, we concatenate the noisy 1spp color and all guiding buffers in the pre-processing stage and use them as input to a single branch U-Net. To focus on the multi-branch versus single branch comparison, we utilize the same convolutional residual (ConvRes) blocks, downsample convolution and merge blocks for the single branch U-Net while we reduce the number of output features to make the number of learnable parameters equivalent to our lightweight multi-branch U-Net architecture. From the single branch U-Net, we get ($\mathcal{F}_{(rg)^0}$,$\mathcal{F}_{(rg)^1}$,$\mathcal{F}_{(rg)^2}$) that are used to derive filtering kernel weights in our multi-scale filtering stage.

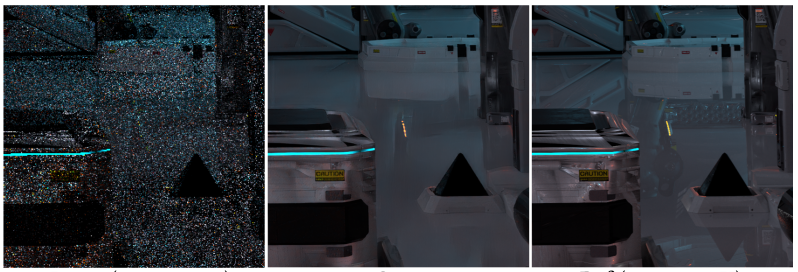
From visual quality comparisons for test dataset in Figure 7, we see that the features from our proposed multi-branch U-Net are useful in reconstructing complex details and robust temporal quality. The results using a single branch U-Net show severe ghosting and overblurred shadow (middle insets). Table 4 shows quality metrics comparison for the dataset shown in Figure 7. We average metrics from a sequence of frames for PSNR, SSIM and LPIPS. We also show FovVDP for temporal quality. Although the difference in quantitative metrics between multi-branch U-Net and single-branch U-Net is small, our multi-branch U-Net shows slightly better score in most metrics as well as in visual quality.

5.6 Partial convolution vs. full convolution

We conduct an ablation study to demonstrate the benefits of our design choice to use partial convolution (MPConvRes block) in most layers of our MUNet, compared to a full convolution. To do this, we replace all MPConvRes blocks in our MUNet with MConvRes blocks and train it using

Table 5. Comparison between our multi-branch U-Net with a partial convolution (MPConvRes) and with a full convolution in all layers. We average quality metrics from frames in unseen test scenes for PSNR↑, SSIM↑ and LPIPS↓. We also show FovVDP↑ as a temporal quality metric. We observe that changes in quality metrics are marginal when using partial convolutions (MPConvRes blocks). We gain benefits in performance with a 38.6% smaller number of parameters, 18.5% lower GFLOPs, and 3.6% faster time for our MUNet.

	Ours w/ partial convolution		Ours w/ full convolution	
	Sicka-Mansion	Museum	Sicka-Mansion	Museum
FovVDP↑	6.981	8.240	6.964	8.087
PSNR↑	36.478	37.664	37.444	38.101
SSIM↑	0.947	0.956	0.953	0.958
LPIPS↓	0.128	0.122	0.126	0.111
# of params (M)	0.644		1.05	
GFLOPs	396		486	
Network time (ms)	53		55	



Input (1spp,1080p) Ours Ref (8192spp,4K)

Fig. 8. Quality limitation in our result for highly reflective surfaces, in which lack of valid information in a noisy 1spp input and auxiliary guiding buffers to reconstruct the secondary surface pixels reflected on the floor. Scene from ZeroDay [Winkelmann 2019].

the same training dataset. The pre-processing and multi-sacle filtering stages in our technique are reused.

From the quality metrics comparison for the unseen test dataset (Sicka-Mansion, Museum) in Table 5, we see only marginal changes in the quality metrics. The partial convolution in our MPConvRes block enables us to create a lightweight network, as shown in Table 5, with a 38.6% smaller number of parameters, 18.5% lower GFLOPs, and 3.6% faster network time compared to using full convolutions, without compromising the quality. Note that the network time (ms) has been measured using an unoptimized PyTorch implementation (FP32).

5.7 Limitations

Our technique generates high-quality joint MC denoising and upscaling compared to the state-of-the-art techniques for very low samples per pixel (1spp). As presented in Figure 4, Figure 5 and Figure 6, our method shows better quality than other methods on glossy reflections as well. However, we have quality limitations on highly reflective and transparent objects as exampled in Figure 8, which is expected since we use primary surface buffers so that there exists no valid information available in our guiding buffers for such highly reflective and transparent objects.

Enhancing the guiding buffers with ray-traced secondary surfaces information would be useful to achieve high-quality reflection, which we leave as a future work.

In addition, we do not consider post-processing effects like volumetric fog, particles, and depth of field in our dataset since we do not have valid motion vectors from rendering. We leave it as a future work as well to perform extensive study.

6 Conclusion

We have presented a multi-branch and multi-scale feature extraction network for joint denoising and upscaling technique that produces high-quality denoising at a higher resolution, given noisy 1spp Monte Carlo renderings in a low resolution. To effectively account for different characteristics in the noisy radiance and noise-free, aliased auxiliary guiding buffers, we proposed a lightweight multi-branch U-Net architecture for feature extraction. Our technique demonstrated superior quality in diverse complex scenes compared to the state-of-the-art techniques for both denoising and upscaling, as well as denoising-only. In future work, our multi-branch feature extraction architecture could be leveraged with enhanced guiding buffers for reflective and transparent surfaces.

References

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

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. 2024. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (La Jolla, CA, USA) *(ASPLOS '24)*. Association for Computing Machinery, New York, NY, USA, 929–947. https://doi.org/10.1145/3620665.3640366

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2022. Self-Supervised Post-Correction for Monte Carlo Denoising. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 18, 8 pages. https://doi.org/10.1145/3528233.3530730

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.* 36, 4, Article 97 (07 2017), 14 pages. https://doi.org/10.1145/3072959.3073708

Martin Balint, Krzysztof Wolski, Karol Myszkowski, Hans-Peter Seidel, and Rafał Mantiuk. 2023. Neural Partitioning Pyramids for Denoising Monte Carlo Renderings. In *ACM SIGGRAPH 2023 Conference Proceedings* (<conf-loc>, <city>Los Angeles</city>, <state>CA</state>, <country>USA</country>, </conf-loc>) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 60, 11 pages. https://doi.org/10.1145/3588432.3591562

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph.* 39, 4, Article 148 (08 2020), 17 pages. https://doi.org/10.1145/3386569.3392481

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.* 36, 4, Article 98 (07 2017), 12 pages. https://doi.org/10.1145/3072959.3073601

Bingyi Chen, Zengyu Liu, Li Yuan, Zhitao Liu, Yi Li, Guan Wang, and Ning Xie. 2023b. Monte Carlo Denoising via Multi-scale Auxiliary Feature Fusion Guided Transformer. In *SIGGRAPH Asia 2023 Technical Communications* (Sydney, NSW, Australia) *(SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 1, 4 pages. https://doi.org/10.1145/3610543.3626179

Chuhao Chen, Yuze He, and Tzu-Mao Li. 2024. Temporally Stable Metropolis Light Transport Denoising using Recurrent Transformer Blocks. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 4, Article 123 (2024).

Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S.-H. Gary Chan. 2023a. Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12021–12031. https://doi.org/10.1109/CVPR52729.2023.01157

Holger Dammertz, Daniel Sewtz, Johannes Hanika, and Hendrik P. A. Lensch. 2010. Edge-avoiding À-Trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics* (Saarbrucken, Germany) *(HPG '10)*. Eurographics Association, Goslar, DEU, 67–75.

Hangming Fan, Rui Wang, Yuchi Huo, and Hujun Bao. 2021. Real-time Monte Carlo Denoising with Weight Sharing Kernel Prediction Network. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 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.* 38, 4, Article 125 (07 2019), 12 pages. https://doi.org/10.1145/3306346.3322954

Jeongmin Gu, Jonghee Back, Sung-Eui Yoon, and Bochang Moon. 2024. Target-Aware Image Denoising for Inverse Monte Carlo Rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 4 (2024).

Yuchi Huo and Sung eui Yoon. 2021. A survey on deep learning-based Monte Carlo denoising. *Computational Visual Media* 7, 2 (2021), 169–185. https://doi.org/10.1007/s41095-021-0209-9

Mustafa Işık, Krishna Mullia, Matthew Fisher, Jonathan Eisenmann, and Michaël Gharbi. 2021. Interactive Monte Carlo denoising using affinity of neural features. *ACM Trans. Graph.* 40, 4, Article 37 (07 2021), 13 pages. https://doi.org/10.1145/3450626.3459793

James T. Kajiya. 1986. The rendering equation. *SIGGRAPH Comput. Graph.* 20, 4 (08 1986), 143–150. https://doi.org/10.1145/15886.15902

Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A machine learning approach for filtering Monte Carlo noise. *ACM Trans. Graph.* 34, 4, Article 122 (07 2015), 12 pages. https://doi.org/10.1145/2766977

Nima Khademi Kalantari and Ravi Ramamoorthi. 2017. Deep High Dynamic Range Imaging of Dynamic Scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4 (2017).

Simon Kallweit, Petrik Clarberg, Craig Kolb, Tom'aš Davidovič, Kai-Hwa Yao, Theresa Foley, Yong He, Lifan Wu, Lucy Chen, Tomas Akenine-Möller, Chris Wyman, Cyril Crassin, and Nir Benty. 2022. The Falcor Rendering Framework. BSD-Licensed Github Repository. https://github.com/NVIDIAGameWorks/Falcor

Kiya Kandar and Juha Sjoholm. 2024. Alan Wake 2: A Deep Dive into Path Tracing Technology. https://www.nvidia.com/en-us/on-demand/session/gdc24-gdc1003/.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). https://api.semanticscholar.org/CorpusID:6628106

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, Article 138 (06 2019), 14 pages. https://doi.org/10.1145/3269978

Pawel Kozlowski and Tim Cheblokov. 2021. ReLAX: A Denoiser Tailored to Work with the ReSTIR Algorithm. https://www.nvidia.com/en-us/on-demand/session/gtcspring21-s32759/.

Weiheng Lin, Beibei Wang, Jian Yang, Lu Wang, and Ling-Qi Yan. 2021. Path-based Monte Carlo Denoising Using a Three-Scale Neural Network. *Computer Graphics Forum* 40, 1 (2021), 369–381. https://doi.org/10.1111/cgf.14194

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

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.* 40, 4, Article 49 (July 2021), 19 pages. https://doi.org/10.1145/3450626.3459831

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*, Carsten Dachsbacher and Matt Pharr (Eds.). The Eurographics Association. https://doi.org/10.2312/sr.20201133

Michael Murphy, Jakub Knapik, and Pawel Kozlowski. 2024. RT: Overdrive in Cyberpunk 2077 Ultimate Edition - Pushing Path Tracing One Step Further. https://www.nvidia.com/en-us/on-demand/session/gdc24-gdc1002/.

NVidia. 2021. OptiX AI-Accelerated Denoiser. https://developer.nvidia.com/optixdenoiser.

NVidia. 2023. NVIDIA DLSS 3.5 Ray Reconstruction. https://www.nvidia.com/en-us/geforce/news/nvidia-dlss-3-5-ray-reconstruction/.

Geunwoo Oh and Bochang Moon. 2024. Joint self-attention for denoising Monte Carlo rendering. *The Visual Computer* 40 (06 2024), 1–12. https://doi.org/10.1007/s00371-024-03446-8

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (Eds.). Springer International Publishing* (05 2015), 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Christoph Schied, Christoph Peters, and Carsten Dachsbacher. 2018. Gradient Estimation for Real-time Adaptive Temporal Filtering. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 2, Article 24 (08 2018), 16 pages. https://doi.org/10.1145/3233301

Christoph Schied, Marco Salvi, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, and Aaron Lefohn. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. 1–12. https://doi.org/10.1145/3105762.3105770

Manu Mathew Thomas, Gabor Liktor, Christoph Peters, Sungye Kim, Karthik Vaidyanathan, and Angus Forbes. 2022. Temporally Stable Real-Time Joint Neural Denoising and Supersampling. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5 (07 2022), 1–22. https://doi.org/10.1145/3543870

Manu Mathew Thomas, Karthik Vaidyanathan, Gabor Liktor, and Angus G. Forbes. 2020. A reduced-precision network for image reconstruction. *ACM Trans. Graph.* 39, 6, Article 231 (11 2020), 12 pages. https://doi.org/10.1145/3414685.3417786

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. https://arxiv.org/pdf/1706.03762.pdf

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.* 37, 4, Article 124 (07 2018), 15 pages. https://doi.org/10.1145/3197517.3201388

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

Bing Xu, Junfei Zhang, Rui Wang, Kun Xu, Yong-Liang Yang, Chuan Li, and Rui Tang. 2019. Adversarial Monte Carlo denoising with conditioned auxiliary feature modulation. *ACM Trans. Graph.* 38, 6, Article 224 (11 2019), 12 pages. https://doi.org/10.1145/3355089.3356547

Xin Yang, Dawei Wang, Wenbo Hu, Li-Jing Zhao, Bao-Cai Yin, Qiang Zhang, Xiao-Peng Wei, and Hongbo Fu. 2019. DEMC: A Deep Dual-Encoder Network for Denoising Monte Carlo Rendering. *J. Comput. Sci. Technol.* 34, 5 (09 2019), 1123–1135. https://doi.org/10.1007/s11390-019-1964-2

Jiaqi Yu, Yongwei Nie, Chengjiang Long, Wenju Xu, Qing Zhang, and Guiqing Li. 2021. Monte Carlo denoising via auxiliary feature guided self-attention. *ACM Trans. Graph.* 40, 6, Article 273 (12 2021), 13 pages. https://doi.org/10.1145/3478513.3480565

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 586–595. https://doi.org/10.1109/CVPR.2018.00068

Xianyao Zhang, Marco Manzi, Thijs Vogels, Henrik Dahlberg, Markus H. Gross, and Marios Papas. 2021. Deep Compositional Denoising for High-quality Monte Carlo Rendering. *Computer Graphics Forum* 40 (2021). https://api.semanticscholar.org/CorpusID:235825097

Xianyao Zhang, Gerhard Röthlin, Shilin Zhu, Tunç Aydın, Farnood Salehi, Markus Gross, and Marios Papas. 2024. Neural Denoising for Deep-Z Monte Carlo Renderings. *Computer Graphics Forum* 43 (05 2024). https://doi.org/10.1111/cgf.15050

Dmitry Zhdan. 2021. *ReBLUR: A Hierarchical Recurrent Denoiser*. 823–844. https://doi.org/10.1007/978-1-4842-7185-8_49