Online Neural Denoising with Cross-Regression for Interactive Rendering

HAJIN CHOI, Gwangju Institute of Science and Technology, South Korea
SEOKPYO HONG, Samsung Advanced Institute of Technology, South Korea
INWOO HA, Samsung Advanced Institute of Technology, South Korea and KAIST, South Korea
NAHYUP KANG, Samsung Advanced Institute of Technology, South Korea
BOCHANG MOON, Gwangju Institute of Science and Technology, South Korea

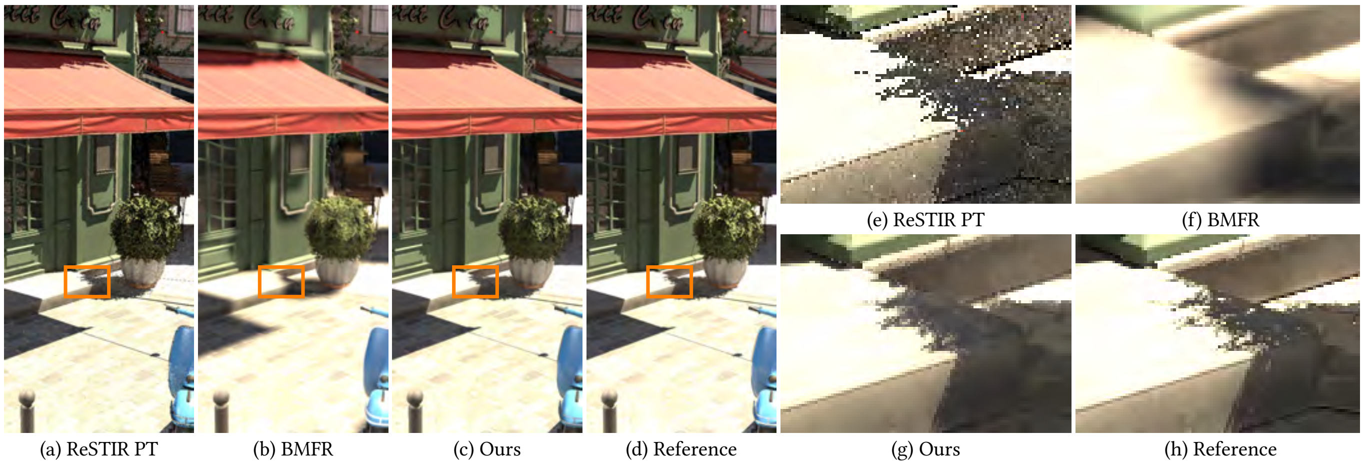


Fig. 1. Denoising results of a regression-based denoiser (BMFR [Koskela et al. 2019]) and our method for an interactive path tracing framework (ReSTIR PT [Lin et al. 2022]). While the existing regression using G-buffers, e.g., textures and normals, preserves geometric edges, it tends to blur other image details (e.g., shadows) that the G-buffers cannot capture. We present a hybrid denoising framework that employs local regression with a neural network for robust denoising capable of maintaining such non-geometric edges. We modify the regression into a cross-regression form to generate pilot estimates, which serve as input to our neural network and guide its online training using only runtime image sequences. 3D model courtesy of [Lumberyard 2017].

Generating a rendered image sequence through Monte Carlo ray tracing is an appealing option when one aims to accurately simulate various lighting effects. Unfortunately, interactive rendering scenarios limit the allowable sample size for such sampling-based light transport algorithms, resulting in an unbiased but noisy image sequence. Image denoising has been widely adopted as a post-sampling process to convert such noisy image sequences into biased but temporally stable ones. The state-of-the-art strategy for interactive image denoising involves devising a deep neural network and training this network via supervised learning, i.e., optimizing the network parameters using training datasets that include an extensive set of image pairs (noisy and ground truth images). This paper adopts the prevalent approach for interactive image denoising, which relies on a neural network. However, instead of supervised learning, we propose a different learning strategy that trains our network parameters on the fly, i.e., updating them online using runtime image sequences. To achieve our denoising objective with online learning, we tailor local regression to a cross-regression form that can guide robust training of our denoising neural network. We demonstrate that our denoising framework effectively reduces noise in input image sequences while robustly preserving both geometric and non-geometric edges, without requiring the manual effort involved in preparing an external dataset.

Authors' Contact Information: Hajin Choi, Gwangju Institute of Science and Technology, Gwangju, South Korea, hajinchoi@gm.gist.ac.kr; Seokpyo Hong, Samsung Advanced Institute of Technology, Suwon, South Korea, sphorpin89@gmail.com; Inwoo Ha, Samsung Advanced Institute of Technology, Suwon, South Korea and KAIST, Dajeon, South Korea, inwooh3@gmail.com; Nahyup Kang, Samsung Advanced Institute of Technology, Suwon, South Korea, nahyup.kang@samsung.com; Bochang Moon, Gwangju Institute of Science and Technology, Gwangju, South Korea, bmoon@gist.ac.kr.

CCS Concepts: • **Computing methodologies** → **Ray tracing**.

Additional Key Words and Phrases: cross-regression, online machine learning, interactive denoising, Monte Carlo ray tracing

ACM Reference Format:
Hajin Choi, Seokpyo Hong, Inwoo Ha, Nahyup Kang, and Bochang Moon. 2024. Online Neural Denoising with Cross-Regression for Interactive Rendering. *ACM Trans. Graph.* 43, 6, Article 221 (December 2024), 12 pages. https://doi.org/10.1145/3687938

1 INTRODUCTION

Monte Carlo (MC) ray tracing, such as path tracing [Kajiya 1986], enables us to synthesize various lighting effects by simulating randomly generated light paths. It allows us to generate noisy but unbiased pixel estimates, which become more accurate when we take more samples per pixel. As a result, if render time is not constrained, we can allocate enough samples to achieve visually smooth images with a low amount of noise. Unfortunately, in interactive rendering scenarios, the sample size must be limited to a small number (e.g., four samples per pixel), resulting in temporally unstable noisy sequences. Recent breakthroughs [Kettunen et al. 2023; Lin et al. 2022], which reuse light paths spatiotemporally across adjacent pixels and frames, have unbiasedly reduced noise in pixel estimates. However, this fundamental challenge stemming from MC integration remains.

An appealing option for addressing MC noise in interactive image sequences is to exploit image denoising, which converts unbiased but noisy image sequences into biased but less noisy ones. A practical advantage of image denoising is its simplicity, as it does not require altering the underlying sampling process. Moreover, image denoising can be designed effectively while alleviating denoising bias by considering rendering-specific buffers, e.g., G-buffers like textures and normals. Well-known examples include cross-bilateral filters [Schied et al. 2017] and regression-based methods [Koskela et al. 2019; Meunier and Harada 2022], which control their denoising weights using G-buffers to avoid blurring pixel colors across geometric edges.

Nonetheless, preserving other image edges introduced by changes in illumination (e.g., shadows), is nontrivial since G-buffers cannot capture such high-frequency information. Estimating denoising errors and adjusting the smoothing per pixel is a well-known optimization route for alleviating excessive denoising bias on non-geometric edges [Zwicker et al. 2015]. While this error analysis-based approach has demonstrated significant noise reduction without severe blurring, a robust estimation of denoising errors often requires enough samples, typically only available in offline rendering scenarios.

An alternative to the classical denoising approach is to rely on a neural network that enables local smoothing adjustments (i.e., balancing denoising bias and variance per pixel) without the need to analyze denoising errors. It often leads to superior denoising outputs, such as sharper results than those achieved with classical methods, especially for non-geometric edges. Nonetheless, relying on a neural network involves a training process using external datasets, which include noisy image sequences and their noiseless ground truth images, i.e., supervised learning. Unfortunately, preparing such datasets requires significant manual effort, as the training datasets should be extensive to enable the denoising neural network to learn to preserve various image edges in runtime image sequences.

This paper presents a hybrid denoising framework that combines classical and neural network-based approaches while maintaining their strengths: the simplicity of classical methods without the need for external training datasets and robust denoising performance driven by a neural network. Our hybrid denoiser reduces the variance of noisy input sequences via a classical regression-based approach and further enhances the denoised images via a neural network. Unlike supervised learning-based denoisers that rely on extensive training datasets, our method employs online learning. We sequentially update our network parameters per frame using only runtime image sequences, eliminating the need for the datasets. Our technical contributions are as follows.

- We adapt classical regression-based denoising to a cross-regression form that splits a noisy input image into two disjoint buffers and fits linear functions formed by pixel colors in one buffer to pixel colors in the other, resulting in two pilot estimates.
- We propose an online learning strategy that trains a denoising neural network using the two pilot estimates. The neural network produces a denoised output from the pilot estimates while simultaneously learning to preserve various image edges from these estimates in runtime, through a sequential update of network parameters per frame.

We demonstrate that our hybrid denoising framework can preserve various image edges, including non-geometric edges, more effectively than classical methods, thanks to the use of a neural network similar to existing supervised learning methods. However, unlike supervised methods, our robust denoising is achieved via online learning without the need for external datasets.

2 RELATED WORK

Image denoising, which trades MC noise with denoising bias, has become a simple and practical post-sampling process. Balancing the bias-variance tradeoff is essential to minimize denoising errors and produce visually sharp denoising results with reduced noise. A well-known optimization route for achieving such balance involves estimating per-pixel denoising errors and selecting denoising parameters that minimize these errors, e.g., adaptive parameter selection for the cross-bilateral filter [Li et al. 2012; Sen and Darabi 2012] and regression-based denoisers [Bitterli et al. 2016; Moon et al. 2014]. Unfortunately, it requires a sufficient number of samples for robust error estimation (and thus proper parameter selection), which is only feasible in offline rendering scenarios. We refer to a thorough discussion by Zwicker et al. [2015] on this adaptive parameter selection in offline rendering. In this section, we discuss interactive denoising methods related to our approach.

Interactive techniques usually take an animated image sequence generated by MC ray tracing at an interactive rate and reduce MC noise by aggregating a pixel color with its neighboring pixel colors. This aggregation step also leverages temporal coherence between the current and previous frames, which relies on image reprojection [Nehab et al. 2007; Scherzer et al. 2007]. See the survey by Yang et al. [2020] for a comprehensive overview of this reprojection scheme.

Classical image denoising. A classical yet natural choice for interactive denoising is to exploit well-known image filters with G-buffers that are much less noisy than input colors. This approach enables image denoising to avoid blurring image edges introduced by geometric discontinuities. For example, Schied et al. [2017; 2018] applied the edge-avoiding A-trous filter [Dammertz et al. 2010] and further optimized their denoising parameters using the variance of pixel colors. Other notable examples include separate filtering using a material decomposition [Mara et al. 2017] and regression-based denoisers [Koskela et al. 2019; Meunier and Harada 2022].

Neural image denoising. A popular alternative is devising a deep neural network that produces a denoised image from a noisy image and G-buffers while guiding the network to achieve an ideal denoising output without excessive edge blurs through supervised learning with ground truth images. Examples include an autoencoder with a recurrent convolutional block [Chaitanya et al. 2017], convolutional neural network-based denoising [Meng et al. 2020] with neural bilateral grids [Gharbi et al. 2017], bilateral weighting using trainable affinity features [Işık et al. 2021], and separate denoising of partitioned radiances by a neural network [Balint et al. 2023]. Other advanced methods are an efficient adaptation [Fan et al. 2021] of a kernel-predicting convolutional neural network [Bako et al. 2017], joint optimization of sampling and denoising [Hasselgren et al. 2020; Thomas et al. 2022], and a kernel-predicting network for separate denoising of surfaces and volumes [Hofmann et al. 2023].

The aforementioned classical approaches and supervised learning methods have distinct strengths and weaknesses. Classical methods are simple as they do not require a pre-training stage. However, preserving non-geometric image edges, such as shadows, is technically challenging since G-buffers do not capture this high-frequency information. On the other hand, learning-based denoisers can enhance the denoising process by guiding their neural networks to preserve such complex edges. However, this superior capability comes with a nontrivial effort in preparing a training dataset that should be extensive enough to generalize to various runtime image sequences.

We propose a hybrid denoising framework that combines the strengths of both approaches: robust denoising with a neural network without the need for external training datasets. We achieve this denoising objective through online training of a denoising neural network, guided by a classical regression-based approach.

Post-correction of image denoising. In offline denoising, Back et al. [2022] proposed a post-correction neural network to improve an offline image denoiser, and they trained the neural network using a self-supervised loss with runtime images, unlike supervised learning-based post-correction networks [Back et al. 2020; Gu et al. 2022]. Our method shares similarities with this recent technique in that both aim to train a neural network using only runtime images. However, a key technical distinction is that we use an online learning scheme that trains the network sequentially per frame using a streaming image sequence, unlike the previous approach, which is designed for an offline scenario (i.e., a single frame input).

3 BACKGROUND

This section briefly overviews regression-based denoising techniques [Koskela et al. 2019; Moon et al. 2014] and motivates our hybrid framework, which integrates this classical regression scheme with a neural network.

Local regression [Cleveland 1979; Cleveland and Devlin 1988; Ruppert and Wand 1994] is a well-established statistical technique that estimates an unknown function $f(\boldsymbol{x})$ by approximating it with simple local functions using observed paired samples $(\boldsymbol{x}_i, y_i)$, and it assumes a noise model:

$$y_i = f(\boldsymbol{x}_i) + e_i, \tag{1}$$

where e_i is an additive noise with $E[e_i] = 0$ and the explanatory variable $\boldsymbol{x}_i$ is supposed to be noise-free.

As an application of local regression, we can treat the unknown and noisy values, $f(\boldsymbol{x}_i)$ and y_i, as the ground truth and noisy colors at pixel i, rendered with infinite and finite samples, respectively. Then, we can think of a simple function that linearly approximates the ground truth $f(\boldsymbol{x}_i)$ in neighboring pixels i within a denoising

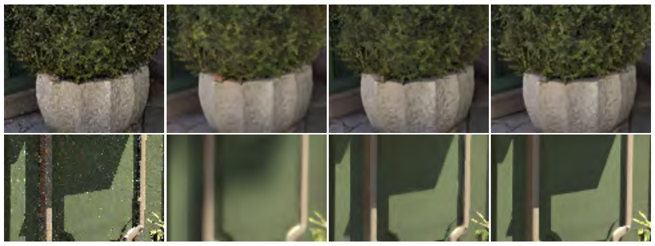

0.9712 / 0.4493 (a) ReSTIR PT — 1.6054 / 0.2463 (b) BMFR — 0.0348 / 0.1177 (c) Ours — relL$_2$ / 1−SSIM (d) Reference

Fig. 2. Comparisons of two regression-based approaches, BMFR [Koskela et al. 2019] (b) and our technique (c), for the unbiased noisy input, ReSTIR PT [Lin et al. 2022] (a). Fitting a linear function of G-buffers enables effective noise reduction while preserving geometric edges (the top row in (b)). However, it comes at the expense of blurring the non-geometric edges (the bottom row in (b)). Our method (c) utilizes an adapted regression with a neural network and preserves the geometric and non-geometric edges. The two numbers below the insets are relative L$_2$ (relL$_2$) [Rousselle et al. 2011] and 1−SSIM [Wang et al. 2004] errors. The insets are taken from the images of the Bistro scene shown in Fig. 6.

window Ω_c (e.g., 17×17 image window) at a center pixel c:

$$f(\boldsymbol{x}_i) \approx f(\boldsymbol{x}_c) + \nabla f(\boldsymbol{x}_c)^T (\boldsymbol{x}_i - \boldsymbol{x}_c), \tag{2}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{x}_c$ are the explanatory variables at pixels i and c, which predict the linear change from $f(\boldsymbol{x}_c)$ to $f(\boldsymbol{x}_i)$. A common choice for setting these variables is to use G-buffers (e.g., textures and normals), which are often nearly noise-free and can guide local color changes by geometric discontinuities [Koskela et al. 2019; Moon et al. 2014]. For brevity, we treat this function as a scalar function since we can apply this approximation to each color channel, respectively.

The unknown values $f(\boldsymbol{x}_c)$ and $\nabla f(\boldsymbol{x}_c)$ in Eq. 2 can be estimated by solving a least-squares objective function at the center pixel c:

$$\begin{bmatrix}\alpha_c \\ \beta_c\end{bmatrix} = \underset{\tilde{\alpha}_c, \tilde{\beta}_c}{\arg\min} \sum_{i \in \Omega_c} w_{c,i} \left(y_i - \tilde{\alpha}_c - \tilde{\beta}_c^T (\boldsymbol{x}_i - \boldsymbol{x}_c) \right)^2, \tag{3}$$

where the resulting parameters α_c and β_c are the estimates of the unknowns $f(\boldsymbol{x}_c)$ and $\nabla f(\boldsymbol{x}_c)$ in Eq. 2. $w_{c,i}$ is the weight that controls the relative importance of the squared error at pixel i. Intuitively, the weight $w_{c,i}$ should be set to a high value only when the difference $\boldsymbol{x}_i - \boldsymbol{x}_c$ can linearly predict the difference $f(\boldsymbol{x}_i) - f(\boldsymbol{x}_c)$, without a significant approximation error (see Eq. 2). The optimal parameters, α_c and β_c, can be computed via the normal equation:

$$\begin{bmatrix}\alpha_c \\ \beta_c\end{bmatrix} = (X_c^T W_c X_c)^{-1} X_c^T W_c Y_c, \tag{4}$$

where the rows of the design matrix X_c are set to $[1, (\boldsymbol{x}_i - \boldsymbol{x}_c)^T]$ and the diagonal matrix W_c is set by the weight $w_{c,i}$ (in Eq. 3). Y_c is a column vector which contains the pixel values y_i within Ω_c.

One can solve the normal equation per pixel and take the α_c (in Eq. 4) for the denoised value at the pixel, i.e., pixel-wise regression [Moon et al. 2014]. An alternative for more efficient computation is solving it only at a sparse set of center pixels and simultaneously producing multiple denoised values within the local window

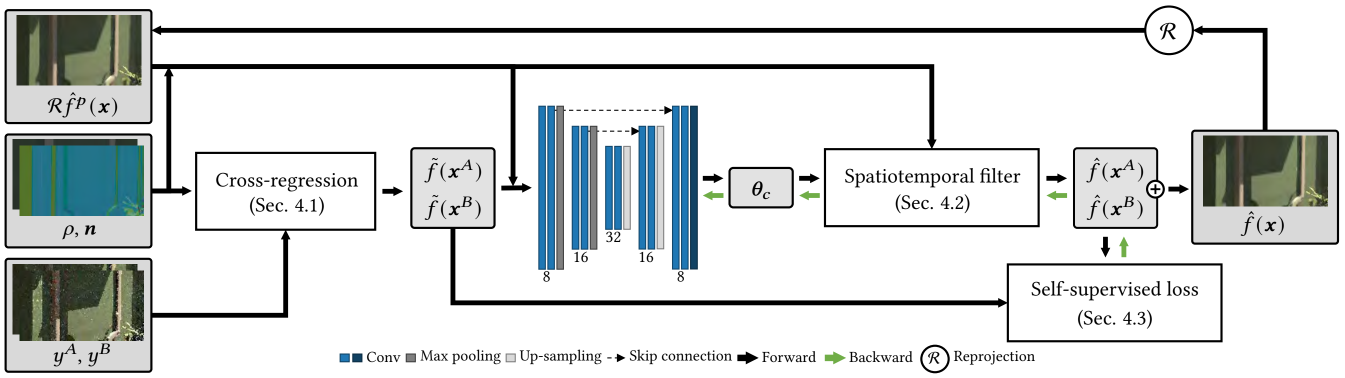

$\mathcal{R}\hat{f}^p(\boldsymbol{x})$
$\rho, \boldsymbol{n}$
y^A, y^B
Cross-regression (Sec. 4.1)
$\tilde{f}(\boldsymbol{x}^A)$
$\tilde{f}(\boldsymbol{x}^B)$
8
16
32
16
8
$\boldsymbol{\theta}_c$
Spatiotemporal filter (Sec. 4.2)
$\hat{f}(\boldsymbol{x}^A)$
$\hat{f}(\boldsymbol{x}^B)$
$\hat{f}(\boldsymbol{x})$
$\mathcal{R}$
Self-supervised loss (Sec. 4.3)
Conv
Max pooling
Up-sampling
Skip connection
Forward
Backward
Reprojection

Fig. 3. Our hybrid denoising framework takes two unbiased images (y^A and y^B) along with G-buffers (textures ρ and normals $\boldsymbol{n}$) and a warped denoising output $\mathcal{R}\hat{f}^p(\boldsymbol{x})$ from the previous frame p as input. We conduct cross-regression that fits linear functions formed by pixel colors in one noisy image to other pixel colors in another image, resulting in two pilot estimates $\tilde{f}(\boldsymbol{x}^A)$ and $\tilde{f}(\boldsymbol{x}^B)$. We then feed the estimates (together with G-buffers and the $\mathcal{R}\hat{f}^p(\boldsymbol{x})$) to a neural network, which infers the per-pixel parameters $\boldsymbol{\theta}_c$ of a spatiotemporal filter. This filter produces two denoised images, $\hat{f}(\boldsymbol{x}^A)$ and $\hat{f}(\boldsymbol{x}^B)$, which are combined for the final output $\hat{f}(\boldsymbol{x})$. Lastly, we update the neural network parameters using a self-supervised loss based on the pilot estimates.

Ω_c using the α_c and β_c, i.e., block-wise regression [Koskela et al. 2019; Moon et al. 2015].

We take the latter approach and uniformly place center pixels in only a small number of positions. Specifically, we use only 1/16 of the pixels in an image as the center pixels and then solve the normal equation (Eq. 4) at these centers. We then compute a denoising output $\tilde{f}(\boldsymbol{x}_i)$ at pixel i using its neighboring center pixels $c \in \Omega_i$ as

$$\tilde{f}(\boldsymbol{x}_i) = \frac{\sum_{c\in\Omega_i} w_{c,i}\left(\alpha_c + \beta_c^T(\boldsymbol{x}_i - \boldsymbol{x}_c)\right)}{\sum_{c\in\Omega_i} w_{c,i}}. \tag{5}$$

Technical challenges and our motivation. Applying this linear regression-based denoising to interactive image sequences is straightforward due to the availability of a closed-form solution, the normal equation (Eq. 4). However, its denoising quality heavily depends on the linear relationship between the explanatory variables and ground truth values. For example, Fig. 2 shows the denoising results of a block-wise regression [Koskela et al. 2019], which uses G-buffers as explanatory variables. It preserves the geometric edges but fails to maintain non-geometric edges that the G-buffers cannot capture. We can consider an optimization that estimates the mean-squared error (MSE) of the regression-based denoising per pixel and adjusts the weight $w_{c,i}$ (in Eq. 3) to minimize its denoising errors [Moon et al. 2014]. Unfortunately, unlike in offline scenarios, robust estimation of the per-pixel MSE is technically challenging for interactive scenarios where only a few samples per pixel can be allocated. This challenge leads us to propose a different strategy for exploiting this classical linear regression with a neural network instead of adopting such an MSE-based optimization, while maintaining its simplicity that does not require any pre-training with external datasets.

4 ONLINE NEURAL DENOISING WITH CROSS-REGRESSION

This section introduces our denoising framework (Fig. 3), which combines local regression (Sec. 3) with a neural network. For the input of this framework, we split color samples produced by path tracing into two sets of samples to generate two disjoint color images (y^A and y^B), each of which is a noisy but unbiased estimate of the ground truth image.

We exploit an adapted regression (i.e., cross-regression in Sec. 4.1) that produces two regression outputs $\tilde{f}(\boldsymbol{x}^A)$ and $\tilde{f}(\boldsymbol{x}^B)$ as pilot estimates using the two noisy inputs (y^A and y^B) and G-buffers (textures ρ and normals $\boldsymbol{n}$). We then utilize the pilot estimates as input for a denoising neural network, which generates our final denoising result through a trainable spatiotemporal filter (Sec. 4.2). We also use these estimates to train the neural network (Sec. 4.3).

4.1 Cross-Regression

Our process begins by utilizing local regression (Sec. 3) to generate pilot estimates. These estimates not only serve as input for a denoising neural network but also guidance for training it. Our denoising neural network produces the final output and updates its learnable parameters based on these estimates. This approach differs from the traditional use of this regression scheme, where the estimates directly become the final output.

A straightforward option for our purpose is to adopt the existing regression scheme (e.g., [Koskela et al. 2019]) without modification. However, it often produces over-blurred results on non-geometric edges (see Fig. 2) that are undesirable for the input and guidance of a denoising neural network, which aims to reduce the noise, not the bias, of its input.

Instead, we aim to produce pilot estimates, $\tilde{f}(\boldsymbol{x}^A)$ and $\tilde{f}(\boldsymbol{x}^B)$, with less noise than the unbiased inputs, y^A and y^B, while preserving image details, including geometric and non-geometric edges, in the noisy inputs. To this end, we configure a linear function using noisy colors in one buffer along with G-buffers, and fit this local function to the noisy colors in another buffer.

For this cross-regression, we split the difference of explanatory variables, $\boldsymbol{x}_i - \boldsymbol{x}_c$ (in Eq. 3), into two separate ones: $\boldsymbol{x}_i^A - \boldsymbol{x}_c^A$ and

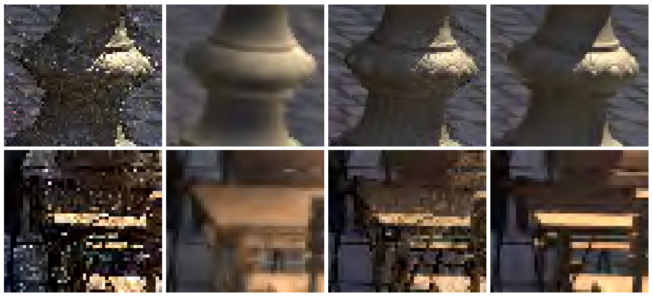


Fig. 4. Comparisons of an existing regression-based approach (BMFR) (b) and our cross-regression (c) for the unbiased input (a). We tailor local regression to a cross-regression form so that its output estimates can guide robust spatiotemporal filtering of a denoising neural network, instead of taking the estimates as the final output. The zoomed areas are taken from the unbiased and denoised images for the Bistro scene in Fig. 6.

$\boldsymbol{x}_i^B - \boldsymbol{x}_c^B$. For example, the difference for the buffer A is set as

$$\boldsymbol{x}_i^A - \boldsymbol{x}_c^A = \left[\frac{y_i^A - y_c^A}{\hat{\sigma}_i^A + \hat{\sigma}_c^A + \epsilon}, \rho_i - \rho_c, \boldsymbol{n}_i - \boldsymbol{n}_c \right]^T, \tag{6}$$

where $\hat{\sigma}_i^A$ and $\hat{\sigma}_c^A$ are the estimated standard deviations of y_i^A and y_c^A, and ϵ is a small constant to prevent division by zero. We estimate the variance at a pixel, e.g., $(\hat{\sigma}_i^A)^2$ and $(\hat{\sigma}_c^A)^2$, as the squared difference between the color at the pixel and the average color of its neighboring pixels (except for the pixel) within the 3×3 window.

We also compute the regression weight $w_{c,i}$ (in Eq. 3) separately for each buffer (A or B). Specifically, the weight $w_{c,i}^A$ for the buffer A is computed as

$$w_{c,i}^A = \exp\left(-\frac{\|y_i^A - y_c^A\|^2}{(\hat{\sigma}_c^A)^2 + (\hat{\sigma}_i^A)^2 + \epsilon} \right). \tag{7}$$

The other difference $\boldsymbol{x}_i^B - \boldsymbol{x}_c^B$ and weight $w_{c,i}^B$ for the other buffer B are computed using the y^B in the same way.

We then conduct the block-wise regression (Eqs. 4 and 5) twice at pixel c: one regression using $\boldsymbol{x}_i^A - \boldsymbol{x}_c^A$ and w_i^A for y^B and another regression using $\boldsymbol{x}_i^B - \boldsymbol{x}_c^B$ and w_i^B for y^A. It results in two pilot estimates, $\tilde{f}(\boldsymbol{x}^A)$ and $\tilde{f}(\boldsymbol{x}^B)$, which are fed into our denoising neural network (Sec. 4.2). Note that $\tilde{f}(\boldsymbol{x}^A)$ is strongly correlated with y^A since $\tilde{f}(\boldsymbol{x}^A)$ is a linear function of y^A (not y^B). Similarly, $\tilde{f}(\boldsymbol{x}^B)$ has a strong linear correlation with y^B.

Fig. 4 compares our result, i.e., the average of the two pilot estimates, with a previous regression-based denoiser. Our modification to the existing regression is simple, but it enables us to reduce denoising bias thanks to exploiting noisy but complete shading information, i.e., noisy colors, as explanatory variables. Nonetheless, taking the regression estimates as a final output is undesirable, as the estimates can contain residual noise due to the relaxation of the noise-free assumption on the explanatory variables (in Eq. 1). Instead, we use the resulting estimates as input for a denoising neural network and as guidance for its training.

4.2 Our Denoising Neural Network

Our denoising neural network consists of a simplified variant of U-Net [Ronneberger et al. 2015] and a spatiotemporal filter that produces our final output $\hat{f}(\boldsymbol{x})$ at the current frame (see Fig. 3).

The U-Net takes the two pilot estimates, $\tilde{f}(\boldsymbol{x}^A)$ and $\tilde{f}(\boldsymbol{x}^B)$ from our cross-regression (Sec. 4.1), G-buffers (textures ρ and normals $\boldsymbol{n}$), and a reprojected denoising output $\mathcal{R}\hat{f}^p(\boldsymbol{x})$ from the previous frame p. $\mathcal{R}$ is a reprojection function that warps the previous denoising output $\hat{f}^p(\boldsymbol{x})$ to the current frame using depth-based motions [Nehab et al. 2007; Scherzer et al. 2007]. The last convolutional layer of the U-Net infers six per-pixel parameters, i.e., $\boldsymbol{\theta}_c = [\theta_c^A, \theta_c^B, \theta_c^\rho, \theta_c^n, \theta_c^p, \theta_c^\alpha]$, for our spatiotemporal filtering at pixel c.

Let us detail the spatiotemporal filter that reduces residual noise in their two input estimates, $\tilde{f}(\boldsymbol{x}^A)$ and $\tilde{f}(\boldsymbol{x}^B)$, by spatially averaging the pixel colors in each pilot estimate and temporally blending the spatial output with the warped previous output $\mathcal{R}\hat{f}^p(\boldsymbol{x})$. For example, this spatiotemporal average $\hat{f}(\boldsymbol{x}_c^A)$ at a center pixel c for the first input $\tilde{f}(\boldsymbol{x}^A)$ is computed as

$$\hat{f}(\boldsymbol{x}_c^A) = \theta_c^\alpha \frac{\sum_{i \in \Omega_c'} m_i^A \tilde{f}(\boldsymbol{x}_i^A)}{\sum_{i \in \Omega_c'} m_i^A} + (1 - \theta_c^\alpha)\mathcal{R}\hat{f}^p(\boldsymbol{x}_c), \tag{8}$$

where the denoising window Ω_c' is set to 11×11. We define the weight m_i^A as a cross-bilateral form:

$$\begin{aligned} m_i^A = &\exp\left(-\frac{\|\tilde{f}(\boldsymbol{x}_i^A) - \tilde{f}(\boldsymbol{x}_c^A)\|^2}{(\theta_c^A)^2 + \epsilon} \right) \times \\ &\exp\left(-\frac{\|\rho_i - \rho_c\|^2}{(\theta_c^\rho)^2 + \epsilon} - \frac{\|\boldsymbol{n}_i - \boldsymbol{n}_c\|^2}{(\theta_c^n)^2 + \epsilon} - \frac{\|\boldsymbol{p}_i - \boldsymbol{p}_c\|^2}{(\theta_c^p)^2 + \epsilon} \right), \end{aligned} \tag{9}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{p}_c$ are the pixel positions at pixels i and c, respectively. We also apply this spatiotemporal filter (Eq. 8) to the other input $\tilde{f}(\boldsymbol{x}^B)$ in the same manner, using m_i^B with the $\tilde{f}(\boldsymbol{x}^B)$. It results in the two output estimates, $\hat{f}(\boldsymbol{x}^A)$ and $\hat{f}(\boldsymbol{x}^B)$.

Our last task is to combine the two resulting estimates into the final denoising output $\hat{f}(\boldsymbol{x})$, and this is computed per pixel c as

$$\hat{f}(\boldsymbol{x}_c) = \frac{\hat{f}(\boldsymbol{x}_c^A) \sum_{i \in \Omega_c'} m_i^A + \hat{f}(\boldsymbol{x}_c^B) \sum_{i \in \Omega_c'} m_i^B}{\sum_{i \in \Omega_c'} m_i^A + \sum_{i \in \Omega_c'} m_i^B}. \tag{10}$$

4.3 Online Learning of Our Denoising Neural Network

Optimizing the parameters of a model-based filter, e.g., our spatiotemporal filter with cross-bilateral weighting, through a neural network is a commonly adopted scheme, e.g., [Kalantari et al. 2015]. However, our denoising objective, which is distinct from existing supervised learning-based approaches, is to train such a network on the fly using only runtime image sequences, which frees our framework from relying on an external dataset. To achieve our goal, we define a self-supervised loss:

$$\mathcal{L} = \frac{1}{|\mathcal{I}|} \sum_{c \in \mathcal{I}} \frac{1}{2} (\mathcal{L}_c^s + \mathcal{L}_c^t), \tag{11}$$

where $\mathcal{I}$ is the set of all pixels. This loss $\mathcal{L}$ is configured using a spatial loss $\mathcal{L}_c^s$ and a temporal loss $\mathcal{L}_c^t$ at pixel c. We compute the

Table 1. Average errors of denoised image sequences for each scene. The bold and underlined values indicate the **best** and <u>second-best</u>, respectively.

Learning Type	Method	Bistro		Bistro Dynamic		Emerald Square		Staircase		Music Room	
		relL$_2$	1−SSIM	relL$_2$	1−SSIM	relL$_2$	1−SSIM	relL$_2$	1−SSIM	relL$_2$	1−SSIM
-	ReSTIR PT	1.2797	0.4536	0.5141	0.5076	41.5661	0.2992	2.5075	0.2631	0.7543	0.5008
Unsupervised	SVGF	0.9551	0.1787	0.1399	<u>0.1501</u>	0.4106	0.2950	0.0350	<u>0.0608</u>	0.0621	<u>0.0863</u>
	BMFR	1.8255	0.2687	0.1798	0.2070	0.6799	0.3700	0.0262	0.1138	0.0782	0.1196
Supervised	OptiX	<u>0.0901</u>	<u>0.1494</u>	<u>0.0219</u>	0.1570	<u>0.1536</u>	<u>0.2032</u>	<u>0.0086</u>	0.0806	<u>0.0169</u>	0.0905
	NBG	0.7553	0.1943	0.0750	0.1697	0.5379	0.2682	0.0568	0.1056	0.0429	0.1020
Self-supervised	Ours	**0.0409**	**0.1270**	**0.0196**	**0.1170**	**0.0470**	**0.1718**	**0.0052**	**0.0400**	**0.0115**	**0.0648**

first term as

$$\mathcal{L}_c^s = \frac{1}{2}\left(\frac{\|\hat{f}(\boldsymbol{x}_c^A) - \tilde{f}(\boldsymbol{x}_c^B)\|^2}{\|\tilde{f}(\boldsymbol{x}_c^B)\|^2 + \epsilon} + \frac{\|\hat{f}(\boldsymbol{x}_c^B) - \tilde{f}(\boldsymbol{x}_c^A)\|^2}{\|\tilde{f}(\boldsymbol{x}_c^A)\|^2 + \epsilon}\right). \quad (12)$$

Note that the two denoising outputs, $\hat{f}(\boldsymbol{x}_c^A)$ and $\hat{f}(\boldsymbol{x}_c^B)$, are separate denoising results from the two pilot estimates, $\tilde{f}(\boldsymbol{x}^A)$ and $\tilde{f}(\boldsymbol{x}^B)$, and each output strongly correlates only with its input, i.e., either $\tilde{f}(\boldsymbol{x}^A)$ or $\tilde{f}(\boldsymbol{x}^B)$. Hence, we can measure the discrepancy between a denoising output in one buffer and the pilot estimates in another buffer while avoiding overfitting to noise.

Additionally, we exploit the temporal loss $\mathcal{L}_c^t$ for temporally stable denoising as

$$\mathcal{L}_c^t = \frac{1}{2}\left(\frac{\|\hat{f}(\boldsymbol{x}_c^A) - \mathcal{R}\tilde{f}^p(\boldsymbol{x}_c^B)\|^2}{\|\mathcal{R}\tilde{f}^p(\boldsymbol{x}_c^B)\|^2 + \epsilon} + \frac{\|\hat{f}(\boldsymbol{x}_c^B) - \mathcal{R}\tilde{f}^p(\boldsymbol{x}_c^A)\|^2}{\|\mathcal{R}\tilde{f}^p(\boldsymbol{x}_c^A)\|^2 + \epsilon}\right), \quad (13)$$

which measures the discrepancy between the denoising outputs, i.e., $\hat{f}(\boldsymbol{x}_c^A)$ and $\hat{f}(\boldsymbol{x}_c^B)$, and the warped pilot estimates from the previous frame p, i.e., $\mathcal{R}\tilde{f}^p(\boldsymbol{x}_c^A)$ and $\mathcal{R}\tilde{f}^p(\boldsymbol{x}_c^B)$. We update the parameters of our neural network using the self-supervised loss $\mathcal{L}$ (Eq. 11) per frame through backpropagation, i.e., one gradient step per frame.

Relation to previous self-supervised learning methods. Training a denoising or post-denoising neural network using two noisy colors without ground truth images has been explored. For instance, Lehtinen et al. [2018] introduced a self-supervised learning scheme, Noise2Noise, which trains a denoising neural network for general or rendered images using only noisy image pairs. Additionally, Back et al. [2022] trained a post-denoising network that corrects a denoising output from a noisy image using another noisy image.

These previous investigations have guided us in preparing two exclusive input images. However, these inputs are less reliable than those used in the previous offline scenarios, as our input is an image sequence generated with only a few samples (not the static frame rendered with relatively large samples). To accommodate this self-supervised learning for our interactive scenario, we exploit cross-regression that transforms the noisy inputs into more accurate pilot estimates due to reduced noise (see Fig. 4), while maintaining a low correlation between the two. It enables our neural network to learn robust spatiotemporal denoising for a very noisy streaming input.

4.4 Network and Training Details

Our framework (Fig. 3) employs a U-Net architecture with two max-pooling and up-sampling layers, and its final layer is a 1×1 convolutional layer that infers the parameters $\boldsymbol{\theta}_c = [\theta_c^A, \theta_c^B, \theta_c^\rho, \theta_c^n, \theta_c^p, \theta_c^\alpha]$ for the spatiotemporal filtering per pixel c. We use the sigmoid function for the θ_c^α. The other convolutional layers have $[8, 16, 32, 16, 8]$ filters of size 3×3 with ReLU. The number of learnable parameters for the U-Net is about 30K, and we initialize the parameters randomly before taking the first frame in a runtime image sequence to verify our online learning framework without any pre-training. We use Adam [Kingma and Ba 2017] with a learning rate of 0.001.

We apply the log transform [Bako et al. 2017] to two unbiased images, y^A and y^B, in HDR color space and use the images as the input to our cross-regression. Thus, the U-Net takes the pilot estimates, $\tilde{f}(\boldsymbol{x}^A)$ and $\tilde{f}(\boldsymbol{x}^B)$, in a log-transformed color space and produces their denoising output $\hat{f}(\boldsymbol{x})$ in the color space. We then inverse-transform the output into the original space. We have implemented this neural network using Tensorflow [Abadi et al. 2015].

5 RESULTS AND DISCUSSION

This section compares our denoising with two classical methods: SVGF [Schied et al. 2017], which uses a variance-guided cross-bilateral filter, and BMFR [Koskela et al. 2019], which employs an efficient block-wise regression. We verify that combining a regression-based approach with a neural network can produce more accurate results than these classical methods while addressing their technical weakness, i.e., blurring non-geometric edges.

We also compare our method with two supervised learning methods: an industrial denoiser, NVIDIA OptiX denoiser (OptiX) [NVIDIA 2021], and NBG [Meng et al. 2020]. We validate that our method can achieve their superior denoising quality (i.e., preserving non-geometric edges) without the need for supervised learning with an external dataset.

We generate unbiased but noisy image sequences using a state-of-the-art path tracer, ReSTIR PT [Lin et al. 2022], built upon Falcor [Kallweit et al. 2022], as input for the denoisers, unless otherwise mentioned. Specifically, we let ReSTIR PT set the number of reservoirs per pixel to match the number of samples per pixel (spp). When testing our method, we split the reservoirs into two sets to produce the two unbiased inputs (y^A and y^B).

We use the SVGF and OptiX implementations within the Falcor framework. We exploit the public implementation of BMFR and tune its parameters using our test scenes. Additionally, we use the pre-trained networks provided by the respective authors of NBG. We test denoisers using five scenes (the Bistro, Bistro Dynamic, Emerald Square, Staircase, and Music Room scenes shown in Fig. 6), each containing 300 animated frames. We use 4 spp for the Emerald Square and 2 spp for other scenes. The reference images are rendered using 2K spp. We set the image resolution to Full HD (1920 × 1080).

We use the relative L_2 (relL_2) [Rousselle et al. 2011] to measure the numerical accuracy of tested methods and SSIM [Wang et al. 2004] to assess the perceptual errors of the techniques.

Qualitative and quantitative comparisons. Figs. 6 and 7 provide visual and numerical comparisons with classical methods (SVGF and BMFR) and supervised learning techniques (OptiX and NBG). Table 1 reports the average errors of tested denoisers for animated sequences. As shown in Fig. 6, SVGF and BMFR produce smooth denoising results while preserving high-frequency details captured by their edge-stopping functions (i.e., G-buffers), but they tend to blur other edges, such as shadows. The supervised learning methods (OptiX and NBG) restore such complex edges more effectively than classical approaches through their neural networks trained with external datasets. In particular, the industrial method (OptiX) produces much lower relL_2 errors than the classical approaches, as shown in Fig. 7 and Table 1.

Our method relies on a neural network, similar to the existing neural denoisers. However, we employ a different learning strategy: online learning of the network using only runtime image sequences. The results (Figs. 6, 7, and Table 1) verify that our online learning approach, combined with cross-regression, can compete with the tested supervised learning techniques, without the need for preparing external training datasets.

Comparisons of temporal stability. Our accompanying video compares the temporal stability of the tested denoisers. None of the tested methods, including ours, show ideal denoising results without any temporal artifacts. However, all these denoisers significantly reduce the temporal instability of their input sequences (i.e., ReSTIR PT) corrupted by MC noise, highlighting the necessity of spatiotemporal denoising for interactive MC rendering. Nevertheless, this enhancement of temporal stability comes with noticeable blurring on image edges in the unbiased input sequences. For example, SVGF and BMFR achieve their temporal stability at the expense of blurring non-geometric edges. OptiX and NBG maintain such edges better than the classical methods, but preparing their training dataset requires considerable effort. On the other hand, our method generates denoised sequences whose temporal stability is comparable to that of the tested methods while preserving both geometric and non-geometric image details (see also the 1−SSIM plots in Fig. 7), through online learning.

Ablation studies. In Fig. 8, we conduct an ablation study of our denoising framework to demonstrate the relative importance of our two main components: cross-regression and a neural network for spatiotemporal filtering. Taking the cross-regression results directly as the final denoising output, i.e., an average of the two pilot estimates, $\tilde{f}(\mathbf{x}^A)$ and $\tilde{f}(\mathbf{x}^B)$, introduces noticeable residual noise leading to temporal instability. See the accompanying video for comparisons between cross-regression alone and our hybrid method. Alternatively, one could use the neural network alone without cross-regression by replacing its input (i.e., two pilot estimates) with unbiased inputs y^A and y^B. However, as shown in Fig. 8, it produces over-blurred results, as robust self-supervised learning with such noisy inputs is challenging. On the other hand, our hybrid framework, which uses cross-regression to train a neural network, produces an improved result compared to the two alternatives.

Table 2. Average relL_2 errors of image denoisers for input image sequences generated by three different samplings: ordinary path tracing (PT), path tracing with ReSTIR DI, and ReSTIR PT.

Method	Bistro			Emerald Square		
	PT	ReSTIR DI	ReSTIR PT	PT	ReSTIR DI	ReSTIR PT
Input	6.1630	5.0749	1.2797	45.4807	45.4505	41.5661
SVGF	0.8455	0.9680	0.9551	0.4550	0.5201	0.4106
BMFR	1.5350	1.8092	1.8255	0.5594	0.6837	0.6799
OptiX	**0.1870**	<u>0.1254</u>	<u>0.0901</u>	<u>0.1567</u>	<u>0.1614</u>	<u>0.1536</u>
NBG	0.5962	0.7261	0.7553	0.4694	0.5318	0.5379
Ours	<u>0.2303</u>	**0.0959**	**0.0409**	**0.0769**	**0.0571**	**0.0470**

Table 3. Time breakdown of our computational overhead per frame.

Task	**Time**	**Task**	**Time**
Cross-regression	2.27 ms	U-Net inference	4.60 ms
Spatiotemporal filter	4.09 ms	Backpropagation	11.30 ms
		Total	22.26 ms

Analysis of image denoising with different samplings. Our denoising framework leverages unbiased input estimates as features for cross-regression and trains a denoising neural network based on the regression results. Consequently, the accuracy of the input estimates can significantly impact our denoising output. Fig. 5 shows the results of the tested image denoisers for the Bistro scene, with varying sampling schemes to generate unbiased inputs. Specifically, we tested three sampling options: 1) ordinary path tracing (PT), 2) path tracing with ReSTIR DI [Bitterli et al. 2020], and 3) the selected ReSTIR PT. We also report the average errors of denoised image sequences for the Bistro and Emerald Square scenes in Table 2.

As shown in Fig. 5 and Table 2, two denoising methods, OptiX and ours, benefit more effectively from improved sampling schemes (from sampling options 1 to 3) compared to other denoising techniques. Also, our method is comparable to OptiX when using the noisiest input images from sampling option 1 (e.g., yielding 1.23× higher and 2.04× lower errors than OptiX for the Bistro and Emerald Square scenes, respectively, as shown in Table 2). Additionally, our improvement over the other denoisers becomes more significant as we leverage more advanced sampling options (i.e., from 1 to 3).

Computational overheads. Table 3 provides a breakdown of our computational overhead for the tested image resolution (1920×1080), excluding the sampling time of ReSTIR PT. We measured the times

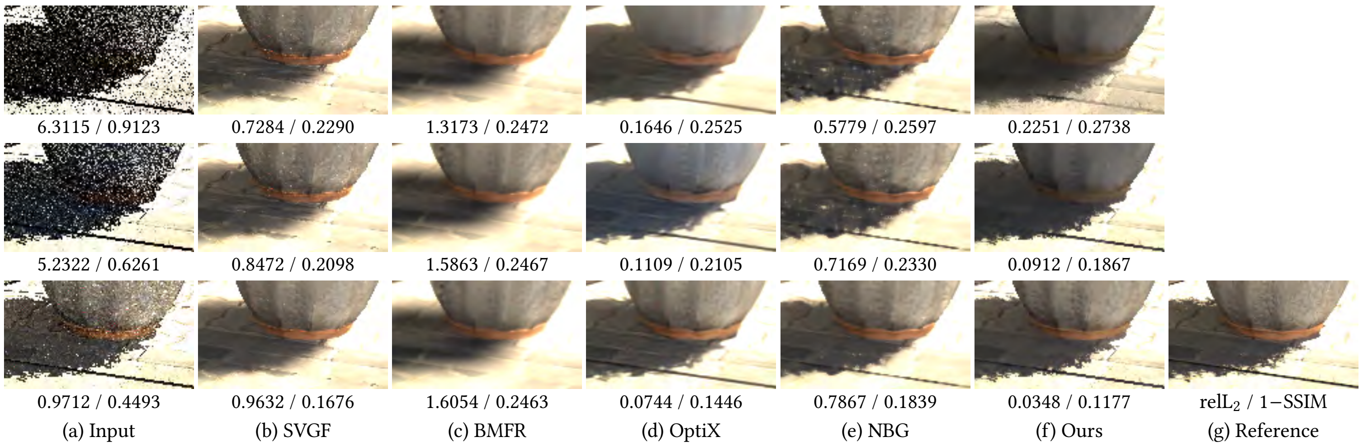


Fig. 5. Comparisons of denoising techniques for the unbiased input (a), generated using three different sampling methods: ordinary path tracing (top row), path tracing with ReSTIR DI (second row), and ReSTIR PT (bottom row). We show the insets from the 100th frame in the image sequence for the Bistro scene.

using an RTX 4090 graphics card. Note that our computational overhead remains the same across different test scenes, as our method is an image-space denoiser, unless the image resolution is changed. As indicated in the table, approximately half of the time is consumed by the runtime training of our denoising neural network, specifically during backpropagation. This results in our overhead being much higher than that of other methods (3.4 ms for SVGF, 0.9 ms for BMFR, 3.7 ms for OptiX, and 8.1 ms for NBG). Nonetheless, our overhead can be considered acceptable for the state-of-the-art method, ReSTIR PT, which generates noisy inputs at interactive rates (e.g., 65 ms per frame for the Bistro scene), especially when considering our significant error reduction of the input, e.g., 26× to 884× smaller relL$_2$ errors than ReSTIR PT in Table 1.

Limitations and future work. Our technical limitation is that we assume noise in unbiased pixel colors is not highly correlated spatially, similar to other image denoisers. The tested path tracer, ReSTIR PT, which uses spatiotemporal reuse of light paths, randomly selects neighboring pixels for its unbiased spatial reuse of light paths. This allows the method to be generally compatible with existing denoisers, as demonstrated by the results throughout the paper. However, when its reuse includes fireflies in a frame, such extreme noise at a pixel can influence its spatiotemporal neighbors, resulting in *correlated* outliers, as shown in Fig. 9. All the tested denoisers, including ours, fail to remove the correlated outliers in their input (ReSTIR PT). Adopting our denoising for the variants of ReSTIR PT, e.g., [Kettunen et al. 2023; Sawhney et al. 2024], which reduce correlation in adjacent pixels, could be effective in lessening such artifacts. It would also be interesting to extend our framework to a more generalized one that robustly handles such correlated outliers.

The tested denoisers, including our method, rely on G-buffers to guide image edges. However, G-buffer-based denoising can be problematic when these buffers become noisy due to depth-of-field effects or motion blur. One possible solution is to reduce noise in G-buffers during a pre-filtering stage and then perform image denoising using the denoised G-buffers, as has been done for offline denoisers (e.g., [Bitterli et al. 2016]). However, this pre-filtering can be challenging in interactive scenarios where the noise in G-buffers can be significant due to small sample sizes. We leave the extension of our method into a more robust form, capable of effectively utilizing noisy G-buffers, for future work.

We demonstrate that training a denoising neural network at an interactive rate is feasible. However, it is desirable to explore more efficient training methods to support real-time rendering. One potential optimization is to update the parameters of our neural network only when necessary, such as by performing backpropagation using a subset of input frames rather than entire image sequences. Additionally, investigating an efficient GPU implementation of a denoising neural network (e.g., [Müller et al. 2021]) would be valuable. Lastly, while we have devised an online training scheme using a simple U-Net architecture, it would be interesting to explore runtime training of more advanced denoising networks (e.g., [Xu et al. 2019; Yu et al. 2021]). We leave these explorations for future work.

6 CONCLUSION

This paper presents a denoising framework that reduces noise in interactively generated image sequences. The state-of-the-art denoising strategy relies on a neural network trained using an extensive training dataset for effective spatiotemporal denoising. However, unlike classical denoising approaches, these supervised learning techniques require the preparation of an external dataset. We propose a different learning strategy that combines a neural network with a classical approach, i.e., cross-regression, leading to robust spatiotemporal denoising without needing external datasets.

ACKNOWLEDGMENTS

We are grateful to the reviewers for their constructive comments. Hajin Choi configured the Music Room scene using 3D models courtesy of Mikel007, yanix, Roy, gandre82, and Giannisimic. Bochang Moon, the corresponding author of this paper, was supported by the Global Research Cluster program of Samsung Advanced Institute of Technology (IO230915-07152-01).

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2020. Deep Combiner for Independent and Correlated Pixel Estimates. *ACM Trans. Graph.* 39, 6, Article 242 (Nov. 2020), 12 pages.

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 (SIGGRAPH '22)*. Association for Computing Machinery, Article 18, 8 pages.

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 (July 2017), 14 pages.

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 (SIGGRAPH '23)*. Association for Computing Machinery, Article 60, 11 pages.

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A. Iglesias-Guitián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. 2016. Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings. *Computer Graphics Forum* 35, 4 (2016), 107–117.

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 (Aug 2020), 17 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.* 36, 4, Article 98 (Jul 2017), 12 pages.

William S Cleveland. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association* 74, 368 (1979), 829–836.

William S Cleveland and Susan J Devlin. 1988. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American statistical association* 83, 403 (1988), 596–610.

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 (HPG '10)*. Eurographics Association, 67–75.

Hangming Fan, Rui Wang, Yuchi Huo, and Hujun Bao. 2021. Real-time Monte Carlo Denoising with Weight Sharing Kernel Prediction Network. *Computer Graphics Forum* 40, 4 (2021), 15–27.

Michaël Gharbi, Jiawen Chen, Jonathan T. Barron, Samuel W. Hasinoff, and Frédo Durand. 2017. Deep bilateral learning for real-time image enhancement. *ACM Trans. Graph.* 36, 4, Article 118 (Jul 2017), 12 pages.

Jeongmin Gu, Jose A. Iglesias-Guitian, and Bochang Moon. 2022. Neural James-Stein Combiner for Unbiased and Biased Renderings. *ACM Trans. Graph.* 41, 6, Article 262 (Nov 2022), 14 pages.

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

Nikolai Hofmann, Jon Hasselgren, and Jacob Munkberg. 2023. Joint Neural Denoising of Surfaces and Volumes. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 1, Article 10 (May 2023), 16 pages.

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 (Jul 2021), 13 pages.

James T. Kajiya. 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.* 20, 4 (Aug 1986), 143–150.

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 (Jul 2015), 12 pages.

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. https://github.com/NVIDIAGameWorks/Falcor

Markus Kettunen, Daqi Lin, Ravi Ramamoorthi, Thomas Bashford-Rogers, and Chris Wyman. 2023. Conditional Resampled Importance Sampling and ReSTIR. In *SIGGRAPH Asia 2023 Conference Papers (SA '23)*. Association for Computing Machinery, Article 91, 11 pages.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

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 (Jun 2019), 14 pages.

Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. 2018. Noise2Noise: Learning Image Restoration without Clean Data. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 2965–2974.

Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. 2012. SURE-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph.* 31, 6, Article 194 (Nov 2012), 9 pages.

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized Resampled Importance Sampling: Foundations of ReSTIR. *ACM Trans. Graph.* 41, 4, Article 75 (Jul 2022), 23 pages.

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

Michael Mara, Morgan McGuire, Benedikt Bitterli, and Wojciech Jarosz. 2017. An Efficient Denoising Algorithm for Global Illumination. In *Proceedings of High Performance Graphics (HPG '17)*. Association for Computing Machinery, Article 3, 7 pages.

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*. Eurographics Association, 13–24.

Sylvain Meunier and Takahiro Harada. 2022. *Weighted Â-Trous Linear Regression (WALR) for Real-Time Diffuse Indirect Lighting Denoising*. Technical Report 22-12-1c2e. Advanced Micro Devices, Inc.

Bochang Moon, Nathan Carr, and Sung-Eui Yoon. 2014. Adaptive Rendering Based on Weighted Local Regression. *ACM Trans. Graph.* 33, 5, Article 170 (Sep 2014), 14 pages.

Bochang Moon, Jose A. Iglesias-Guitian, Sung-Eui Yoon, and Kenny Mitchell. 2015. Adaptive Rendering with Linear Predictions. *ACM Trans. Graph.* 34, 4, Article 121 (Jul 2015), 11 pages.

Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. 2021. Real-time neural radiance caching for path tracing. *ACM Trans. Graph.* 40, 4, Article 36 (Jul 2021), 16 pages.

Diego Nehab, Pedro V. Sander, Jason Lawrence, Natalya Tatarchuk, and John R. Isidoro. 2007. Accelerating Real-Time Shading with Reverse Reprojection Caching. In *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware (GH '07)*. Eurographics Association, 25–35.

Kate Anderson Nicholas Hull and Nir Benty. 2017. NVIDIA Emerald Square, Open Research Content Archive (ORCA). http://developer.nvidia.com/orca/nvidia-emerald-square

NVIDIA. 2021. NVIDIA OptiX™ AI-Accelerated Denoiser. https://developer.nvidia.com/optix-denoiser.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, 234–241.

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2011. Adaptive Sampling and Reconstruction Using Greedy Error Minimization. *ACM Trans. Graph.* 30, 6 (Dec 2011), 1–12.

David Ruppert and Matthew P Wand. 1994. Multivariate locally weighted least squares regression. *The annals of statistics* (1994), 1346–1370.

Rohan Sawhney, Daqi Lin, Markus Kettunen, Benedikt Bitterli, Ravi Ramamoorthi, Chris Wyman, and Matt Pharr. 2024. Decorrelating ReSTIR Samplers via MCMC Mutations. *ACM Trans. Graph.* 43, 1, Article 10 (Jan 2024), 15 pages.

Daniel Scherzer, Stefan Jeschke, and Michael Wimmer. 2007. Pixel-Correct Shadow Maps with Temporal Reprojection and Shadow Test Confidence. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques (EGSR'07)*. Eurographics Association, 45–50.

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 *Proceedings of High Performance Graphics (HPG '17)*. Association for Computing Machinery, Article 2, 12 pages.

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 (Aug 2018), 16 pages.

Pradeep Sen and Soheil Darabi. 2012. On filtering the noise from the random parameters in Monte Carlo rendering. *ACM Trans. Graph.* 31, 3, Article 18 (May 2012), 15 pages.

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. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 3, Article 21 (Jul 2022), 22 pages.

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

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 (Nov 2019), 12 pages.

Lei Yang, Shiqiu Liu, and Marco Salvi. 2020. A Survey of Temporal Antialiasing Techniques. *Computer Graphics Forum* 39, 2 (2020), 607–621.

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 (Dec 2021), 13 pages.

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and Sung-Eui Yoon. 2015. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)* 34, 2 (May 2015), 667–681.

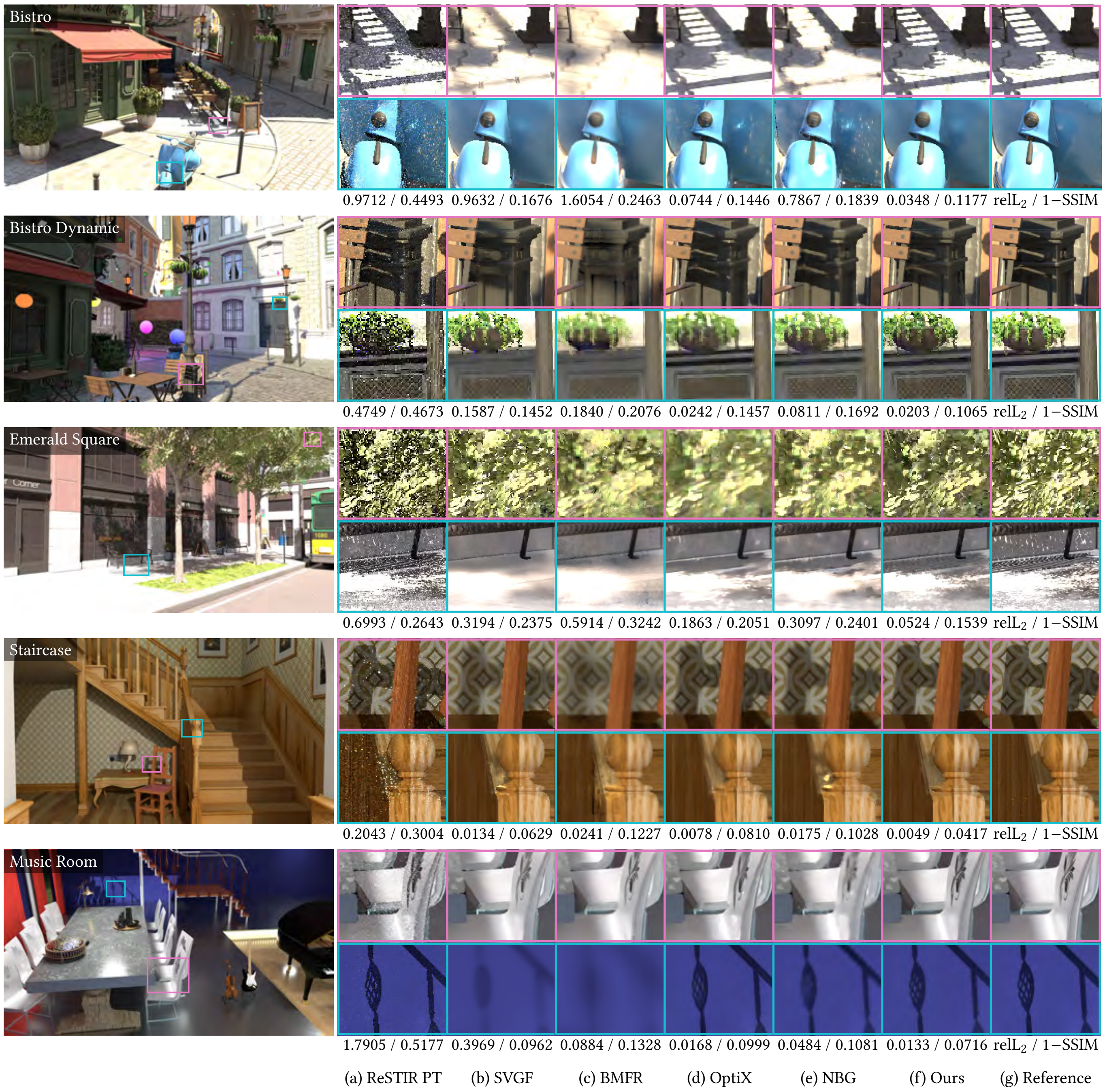

Bistro
0.9712 / 0.4493 0.9632 / 0.1676 1.6054 / 0.2463 0.0744 / 0.1446 0.7867 / 0.1839 0.0348 / 0.1177 relL$_2$ / 1−SSIM
Bistro Dynamic
0.4749 / 0.4673 0.1587 / 0.1452 0.1840 / 0.2076 0.0242 / 0.1457 0.0811 / 0.1692 0.0203 / 0.1065 relL$_2$ / 1−SSIM
Emerald Square
0.6993 / 0.2643 0.3194 / 0.2375 0.5914 / 0.3242 0.1863 / 0.2051 0.3097 / 0.2401 0.0524 / 0.1539 relL$_2$ / 1−SSIM
Staircase
0.2043 / 0.3004 0.0134 / 0.0629 0.0241 / 0.1227 0.0078 / 0.0810 0.0175 / 0.1028 0.0049 / 0.0417 relL$_2$ / 1−SSIM
Music Room
1.7905 / 0.5177 0.3969 / 0.0962 0.0884 / 0.1328 0.0168 / 0.0999 0.0484 / 0.1081 0.0133 / 0.0716 relL$_2$ / 1−SSIM
(a) ReSTIR PT
(b) SVGF
(c) BMFR
(d) OptiX
(e) NBG
(f) Ours
(g) Reference

Fig. 6. Comparisons with classical methods (SVGF and BMFR) and supervised learning techniques (OptiX and NBG) for the unbiased input (ReSTIR PT). SVGF (b) and BMFR (c) preserve geometric edges due to the use of G-buffers, but they tend to blur edges introduced by illumination changes, such as shadows. OptiX (d) and NBG (e) maintain such complex edges more effectively than the classical methods, thanks to their use of neural networks. Our method (f) produces visually sharp denoising results while preserving geometric and non-geometric edges, similar to the supervised methods. However, this capability is achieved through a different learning strategy, i.e., online learning, without the need for external training datasets. All images are from the 100th frame of each animated sequence. 3D models courtesy of [Lumberyard 2017] (Bistro and Bistro Dynamic), [Nicholas Hull and Benty 2017] (Emerald Square), and Wig42 (Staircase).

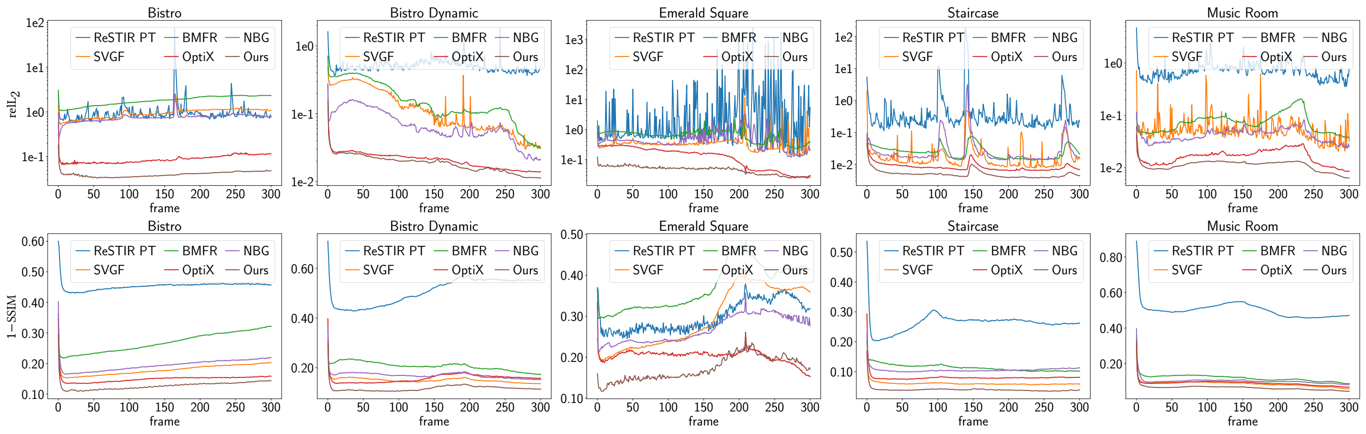
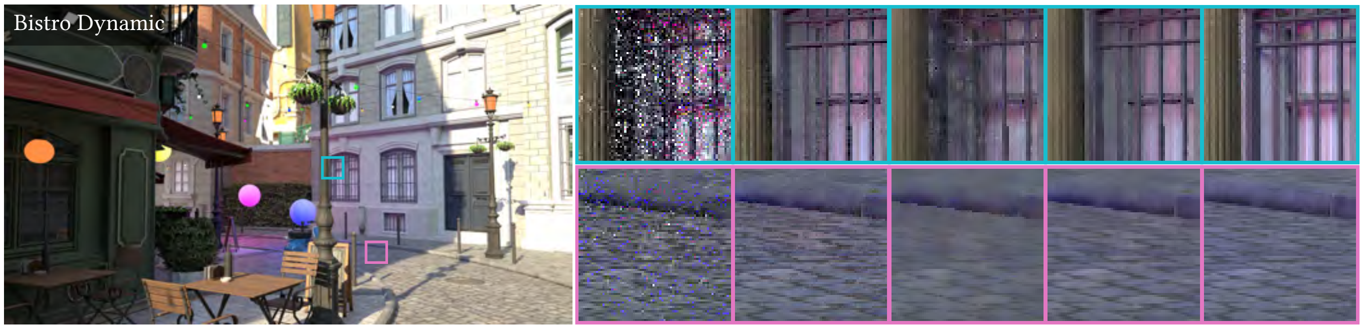


Fig. 7. Relative L_2 (relL_2) and 1−SSIM comparisons of denoising methods for the five scenes, each containing 300 animated frames.



0.4749 / 0.4673	0.0240 / 0.1302	0.0330 / 0.1614	0.0203 / 0.1065	relL_2 / 1−SSIM
(a) ReSTIR PT	(b) Ours (regression)	(c) Ours (neural net.)	(d) Ours (both)	(e) Reference

Fig. 8. Ablation studies of our framework, which consists of cross-regression and a neural network with spatiotemporal filtering. Testing cross-regression or the neural network alone results in either under-blurred (b) or over-blurred results (c), compared to our chosen design that combines both (d). We show images from the 100th frame of the animated sequence for the Bistro Dynamic scene. 3D model courtesy of [Lumberyard 2017].

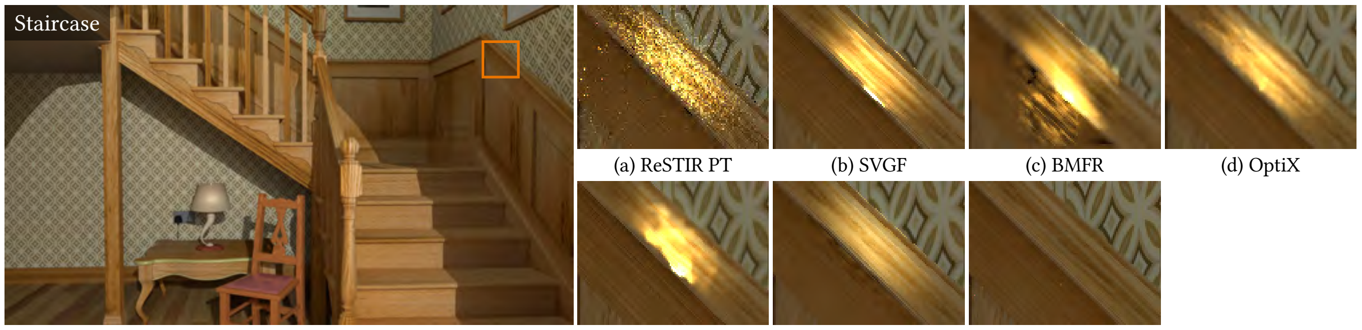


Fig. 9. Failure cases of tested denoisers, (b) to (f), for their unbiased input (a) produced by ReSTIR PT using spatiotemporal resampling. None of the tested denoisers can robustly handle *correlated* outliers (i.e., fireflies) in the input, resulting in noticeable denoising artifacts, i.e., bright spots in their results. The images above are taken from the 148th frame of the animated sequence for the Staircase scene, where the artifacts are clearly visible. 3D model courtesy of Wig42.