# Denoising Monte Carlo Renderings based on a Robust High-order Function

Yu Liu[1,2], Changwen Zheng[1] and Hongliang Yuan[1,2]
[1]*Institute of Software, Chinese Academy of Sciences, Beijing, China*
[2]*University of Chinese Academy of Sciences, Beijing, China*

Keywords: Adaptive Rendering, Image Space Reconstruction, Guided Image Filter, Mean Squared Error.

Abstract: Image space rendering methods are efficient at removing Monte Carlo noise. However, a major challenge is optimizing the bandwidth to denoise images while preserving their fine details. In this paper, a high-order function is proposed to leverage the correlation between features and pixel colors. We consider feature buffers to fit data while computing regression weights using pixel colors. A collaborative prefiltering framework is first proposed to denoise features. The input pixel colors are then denoised using a guided image filter that maintains fine details in the output by constructing a guidance image using features. The optimal bandwidth is selected through an iterative error estimation process performed at multiple pixels to smooth the details. Finally, we adaptively select center pixels to build our regression models and vary the window size to reduce computational overhead. Experimental results showed that the new approach outperforms competing methods in terms of the quality of the visual image and the numerical error incurred.

## 1 INTRODUCTION

Monte Carlo (MC) ray tracing (Kajiya, 1986) is a powerful technique to synthesize photo-realistic images. It computes a complex multidimensional integral at each pixel to calculate the scene function. However, a large number of samples is often required to produce a visually pleasing result. To solve this problem, a number of adaptive rendering methods have been proposed (Li et al., 2012; Rousselle et al., 2013; Rousselle et al., 2011). The crucial step is the error analysis of a noisy image produced from few samples. Based on the analysis, appropriate parameters are selected at each pixel to enable a satisfactory trade-off between bias and variance, which amounts to minimizing the mean squared error (rMSE).

Adaptive rendering methods can be classified into multidimensional and image space techniques. Multidimensional methods (Hachisuka et al., 2008) are used in a high-dimensional space where each coordinate corresponds to a random parameter. However, they are restricted by the curse of dimensionality, and can only support a limited set of distributed effects. Image space methods have recently received attention owing to their simplicity and efficiency. Moreover, many feature buffers have been used to direct error analysis. At an abstract level, we can categorize image space methods into two categories: low-order and high-order functions.

Low-order functions model the neighborhood of a pixel as a constant regression to perform bandwidth optimization, whereas high-order ones measure the varying importance of different features and predict values both for the center pixel and its neighboring pixels. For example, weighted local regression is introduced to reconstruct pixels (Moon et al., 2014). However, these methods are prone to overfitting to the noisy input (Bako et al., 2017).

In this paper, we determine the order of our regression function to be one since it presents a nice trade-off between performance and complexity. The noise in features is first removed using a collaborative prefiltering framework. We then leverage the correlation between features and pixel colors to robustly construct a high-order model and regression weights. In particular, the input pixel colors are denoised using a guided image filter (GIF) (He et al., 2010), which alleviates the problem of overfitting to noisy input. We then iteratively estimate the reconstruction error in a patch-wise manner to denoise multiple pixels. Finally we adaptively select center pixels to build our high-order models. Experimental results showed that our method is superior to competing methods on a wide variety of rendering effects.



Liu, Y., Zheng, C. and Yuan, H.
Denoising Monte Carlo Renderings based on a Robust High-order Function.
DOI: 10.5220/0006650602880294
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 1: GRAPP*, pages 288-294
ISBN: 978-989-758-287-5

# 2 RELATED WORK

**Multidimensional Space Rendering.** Hachisuka et al. (Hachisuka et al., 2008) proposed an anisotropic reconstruction using a structure tensor. Durand et al. (Durand et al., 2005) described how the frequency content of radiance was influenced by varying phenomena, and many algorithms are proposed to improve the image quality of specific effects (Soler et al., 2009; Egan et al., 2009; Egan et al., 2011). Recently, Lehtinen et al. (Lehtinen et al., 2011) used depth and motion information to simulate a wide variety of effects. Belcoulr et al. (Belcour et al., 2013) performed a five-dimensional frequency analysis to simulate motion blur and depth of field. These methods typically produce impressive results for specific rendering effects. However, they tend to show poor effectiveness as the dimensions increased.

**Low-order Functions.** Low-order methods often use techniques developed for image processing. Kalantari et al. (Kalantari and Sen, 2013) proposed to enable any spatial image filters to denoise MC renderings. Many methods selected the optimal bandwidth using varying error metrics. Li et al. (Li et al., 2012) used Steins unbiased risk (SURE) to select suitable parameter. However, SURE estimates can be inaccurate at low sampling rates. Sen et al. (Sen and Darabi, 2012) reduced the importance of samples affected by noise. Moon et al. (Moon et al., 2013) computed a virtual flash image to determine the homogeneous pixels. Delbracio et al. (Delbracio et al., 2014) computed a ray histogram to recognize similar pixels. Recently, Liu et al.(Liu et al., 2017) considered the Sobel operator to compute a gradient image to prefilter features.

**High-order Functions.** High-order functions mainly measured the varying importance of different features. Moon et al. (Moon et al., 2014) used a first-order function to predict center pixels. They also estimated the optimal function order (Moon et al., 2016). In addition, a suitable window size is also computed (Moon et al., 2015) to remove noise. Kalantari et al. (Kalantari et al., 2015) proposed to use supervised learning to predict optimal parameters. Recently, Bako et al. (Bako et al., 2017) further used a convolutional neural network (CNN) to enable a more complex kernel. Chaitanya et al. (Chaitanya et al., 2017) introduced a machine learning approach with low sampling budgets. Readers are encouraged to read Zwicker et al. work (Zwicker et al., 2015).

# 3 PROPOSED METHOD

## 3.1 First-order Regression Function

We define our first-order function as an extension of zero-order functions:

$$[\bar{y}_i, \nabla\bar{y}_i] = \underset{\bar{y}_i, \nabla\bar{y}_i}{\arg\min} \sum_{j \in N_i} (y_j - \bar{y}_i - \nabla\bar{y}_i(x_j - x_i))^2 w(i,j) \tag{1}$$

where $\bar{y}_i$ and $\nabla\bar{y}_i$ denote the filtered value and the estimated gradient at the center pixel $i$, respectively. $N_i$ is the reconstruction window. $x_i$ is a feature vector used to fit data, and we consider $D = 9$ dimensional features: pixel coordinates (2D), depth (1D), normal (3D), and albedo (3D). $w(i,j)$ is a weight term between $i$ and $j$.

As described in a past study (Moon et al., 2015), a normal equation for the solution is as follows:

$$\bar{y} = X(X^T WX)^{-1} X^T WY \tag{2}$$

where $X$ is an $n \times (D+1)$ design matrix the $j^{th}$ row of which is $[1, (x_j - x_i)^T]$, and $Y = [y_1, ...y_n]$ are the input colors. $W$ is an $n \times n$ diagonal matrix the $j^{th}$ element of which is $w(i,j)$. $\bar{y}$ corresponds to all reconstructed pixel values in $N_i$. Eq. (2) enables our model to predict $n$ pixels in $N_i$ through one reconstruction step.

## 3.2 Collaborative Feature Prefiltering

To handle specific effects, the noise in the features should be removed for computing $X$. Rousselle et al. (Rousselle et al., 2013) and Liu et al. (Liu et al., 2017) proposed denoising features using image filters, which tend to blur fine details in the focused or motionless areas. Truncated Singular Value Decomposition (TSVD) was also used to reject noisy features. However, it might underestimate local dimensions and lead to blurred details.

Here, we propose a collaborative process to denoise features. The input features are first denoised with a joint-NL-means filter to remove low-frequency noise and maintain fine feature details. However, it may leave untreated substantial noise in areas with complex geometries. We further filter the output of the joint-NL-means filter using a GIF, which helps remove residual noise. As a result, our joint-NL-means kernel and the GIF form a collaborative framework.

First, we define the joint-NL-means kernel as:

$$w_{jnl}(i,j) = \exp\frac{-\|i-j\|^2}{2\sigma_s^2} \exp\frac{-\|P_i - P_j\|^2}{2\sigma_r^2} \prod_{m=1}^{D-2} \exp\frac{-\|f_{i,m} - f_{j,m}\|^2}{2\sigma_m^2} \tag{3}$$

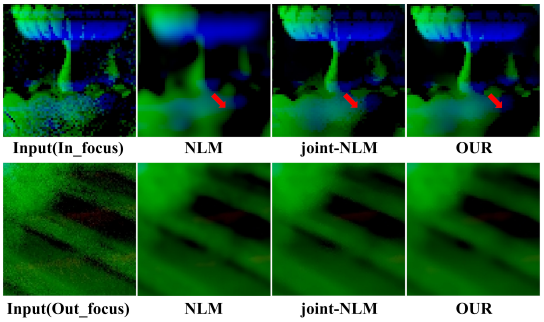


Figure 1: Feature prefiltering in a normal image.

where $f_{i,m}$ denote the value of the $m^{th}$ feature type at pixel $i$. $\sigma_s^2$, $\sigma_r^2$, and $\sigma_m^2$ are the variances in the spatial, the range-related, and the $m^{th}$ feature terms, respectively. $\|P_i - P_j\|$ denotes the patch-based range $P_0$ with radius $r$:

$$\|P_i - P_j\|^2 = \max(0, \frac{1}{(2r+1)^2} \sum_{l \in P_0} d(i+l, j+l)) \quad (4)$$

We follow the metric proposed by Rousselle et al. (Rousselle et al., 2013) to compute the per-pixel range distance:

$$d(i,j) = \frac{\|y_i - y_j\|^2 - (var_i + \min(var_i, var_j))}{\varepsilon + k^2(var_i + var_j)} \quad (5)$$

where $var_i$ is the variance of pixel $i$, and $k$ is the parameter used to adjust the strength of the filter.

To this end, we define the NL-means filter as: $\widehat{Y} = NLM(Y,k,UF)$. The input $Y$ is filtered using an NL-means filter with parameters $k$, and $UF$ denotes whether we use the features to form a joint kernel.

Then, the GIF kernel is defined as follows:

$$\widehat{Y}_j = a_i I_j + b_i, \forall j \in N_i \quad (6)$$

where $a_i$ and $b_i$ are constant coefficients in $N_i$. We set the regularization parameter to 0.001 to prevent $a_j$ from becoming too large. Thus $a_i$ and $b_i$ can be solved by using linear regression (He et al., 2010). We summarize the GIF as: $\widehat{Y} = GUID(Y,I)$, where the input $Y$ is filtered using $I$ as a guidance image.

Given the joint-NL-means and GIF kernels, we filter the $m^{th}$ input feature as follows:

$$\begin{aligned} \overline{f}_m &= NLM(f_m, 0.25, true) \\ \widehat{f}_m &= GUID(\overline{f}_m, \overline{f}_m) \end{aligned} \quad (7)$$

where $f_m$ and $\widehat{f}_m$ are the input and the output of the $m^{th}$ feature type, respectively. We set $\sigma_s = 2$, $\sigma_r = 1$, and $\sigma_m = \{0.8, 0.25, 0.6\}$ for the normal, albedo, and depth, respectively. Note that each feature channel is filtered independently.

Fig.1 shows the result of our collaborative prefiltering. Note the areas denoted by red arrows. It is clear that our method can remove the noise while preserving fine feature details in focused areas.


Figure 2: Denoising color input with a GIF. Our method uses features to construct a guidance image, which forces our results to retain fine details while removing initial noise.

### 3.3 Computing Regression Weight

Using features to compute regression weights can significantly improve the quality of zero-order functions. However, these features can be detrimental to first-order models. For example, WLR does not consider the color buffer and it produces suboptimal results when the features fail to recognize scene structures. Here, we use only pixel colors to compute the regression weights through an NL-means kernel. Our regression weights and first-order model are consequently complementary: the feature-based model recognizes high-frequency scene details whereas the color-based regression weights preserve elements that are not captured by the features. Another advantage of this method is that there is only one parameter $k$ to set for bandwidth optimization.

Since the input colors are very noisy at low sampling rates, inaccurate results are hence produced because the high-order functions are prone to overfitting to noisy color inputs. Here, we denoise the input pixel colors with a GIF, which forces the filtered result to have similar edge characteristics to the guidance image. In this case, the features can be selected as the guidance image as they represent most scene structures. The input pixel colors $y$ are thus prefiltered:

$$\begin{aligned} M_m &= GUID(y, \widehat{f}_m) \\ M &= \sum_{m=1}^{D-2} M_m/(D-2) \end{aligned} \quad (8)$$

where $M_m$ is filtered independently using each feature channel $\widehat{f}_m$ as a guidance image. Since $D-2=7$ feature channels are considered in this paper, $M$ is computed as the average of $M_m$. After denoising the initial pixel colors, the regression weight is computed as: $w(i,j) = NLM(M, k_{opt}, false)$. $k_{opt}$ is the optimal bandwidth computed with our iterative error estimation explained in the next section.

Figure 2 shows the results of prefiltering the color input. After prefiltering with a Gaussian filter (Li et al., 2012), however, a lot of noise remains. Our method, owing to the guidance images, produces smoother details.

### 3.4 Bandwidth Optimization

To minimize the rMSE, a block-based manner is employed to define our reconstruction error:

$$k_{opt} = \arg\min_{k} \sum_{j \in N_i} w(i,j)(\bar{y}_j - \varphi_j)^2 \tag{9}$$

where $\bar{y}_j$ and $\varphi_j$ denote the reconstructed value and the ground truth at pixel $j$, respectively. $w(i,j)$ is the weight between pixel $i$ and $j$ using a candidate bandwidth $k$. Eq. (9) cannot be directly computed, since $\varphi_j$ can only be obtained with a large number of samples. Here, we decompose the per-pixel error $err_{j,k} = (\bar{y}_j - \varphi_j)^2$ of using candidate parameter $k$ into squared bias $bias^2(j,k)$ and variance $Var(j,k)$ terms:

$$\begin{aligned} k_{opt} &= \arg\min_{k} \sum_{j \in N_i} w(i,j) err_{j,k} \\ &= \arg\min_{k} \sum_{j \in N_i} w(i,j)(bias^2(j,k) + Var(j,k)) \end{aligned} \tag{10}$$

To solve Eq. (10), an iterative process is proposed here. In the $t^{th}$ iteration, as described in a previous study (Moon et al., 2016), the bias and variance terms can be computed using a hat matrix $H$ ($H = X(X^T WX)^{-1} X^T W$):

$$\begin{aligned} bias_t(j,k) &= \sum_{s=1}^{s=n} H_{j,s} \bar{y}_{s,t-1} - \bar{y}_{j,t-1} \\ Var_t(j,k) &= \sum_{s=1}^{s=n} (H_{j,s})^2 Var_{t-1}(s,k) \end{aligned} \tag{11}$$

where $bias_t(j,k)$ and $Var_t(j,k)$ denote the bias and variance in the $t^{th}$ iteration at pixel $j$, respectively. $H_{j,s}$ is the $j^{th}$ row and $s^{th}$ column of $H$. $\bar{y}_{s,t-1}$ is the output of pixel $j$ in the $(t-1)^{th}$ iteration. At the first iteration, we set $\bar{y}_{s,0}$ and $Var_0(s,k)$ to the color input $M$ and the variance of sample mean, respectively. In the following iterations, they are replaced by the reconstructed pixel value and the computed variance from the previous iteration.

After computing the squared bias and variance for each iteration, we summarize $err_{j,k}$ as follows:

$$err_{j,k} = \frac{\sum_{t=1}^{T} wei_t (bias_t^2(j,k) + Var_t(j,k))}{\sum_{t=1}^{T} wei_t} \tag{12}$$

where $wei_t = (1 - \frac{1}{(t+1)^2})$ is a weight term for the $t^{th}$ iteration. In this case, our metric assigns a slightly greater weight to a higher iteration, since it was less influenced by noise. $T$ is the total number of iterations. Finally, we plug the result of Eq. (12) into Eq. (10) and return the reconstructed error for using parameter $k$.

To compute the optimal parameter, we test a set of candidate values $k = \{k_1, k_2, k_3, k_4, k_5\}$ and finally select the $k_{opt}$ that produces the smallest error using Eq. (10) at each center pixel. Our error estimation is shown in Fig.3 in comparison with those of two global filtering methods. We also tested our results using different numbers of iterations (i.e., two and three). The result for three iteration yielded slightly less noisy estimates than that for two iteration, but their respective patterns were visually similar. In this case, we set $T$ to 2 to reduce cost.

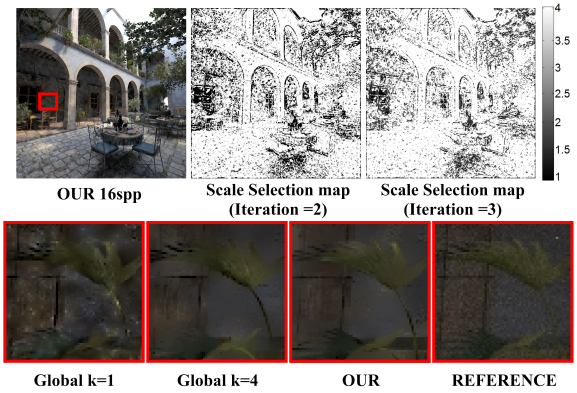


Figure 3: Optimal parameter selection. Our iterative strategy facilitates a trade-off between noise reduction and fidelity to detail.

### 3.5 Selecting Center Pixels

Only sparse models need to be calculated because all pixels in a window can be predicted through a single reconstruction. In this case, for pixels with high-frequency edges, we use a small window to maintain fine structure. Otherwise, a large window is used to reduce computational overhead.

First, three candidate window radii ($r_l > r_m > r_s$) are predefined and the whole image is divided into patches each of radius $r_l$. Then, the number of pixels *num* located in high-frequency areas in each patch is computed. We consider the given patch to contain fine details if *num* is greater than a threshold $\tau$, and choose $r_s$ as the radius of the reconstruction window. Otherwise, $r_l$ is chosen for reconstruction. Following this, we test each pixel and build a model of radius $r_m$ at the given pixel if it is not covered by any existing window.

Since one pixel may be predicted by multiple models, we compute the final output as a weighted average of these overlapping models:

$$out_j = \frac{\sum_i \bar{y}_j^i w(i,j)}{\sum_i w(i,j)} \tag{13}$$

where $out_j$ is our final output at pixel $j$. $\bar{y}_j^i$ and $w(i,j)$ are the reconstruction result and the weight term of

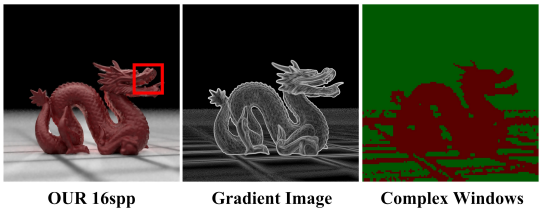


Figure 4: Window size selection. Our metric varies the window size based on area complexity.

pixel $j$ computed from the model centered at pixel $i$.

To compute *num*, we follow the prior work (Liu et al., 2017) of computing a Sobel gradient image in the feature space, which showed that the Sobel operator is sufficiently robust to recognize scene structures.:

$$gra_m = \sqrt{(G_x * \widehat{f}_m)^2 + (G_y * \widehat{f}_m)^2}$$
$$gra = max\{gra_m\} \quad (14)$$

where *gra* is the gradient image computed as the maximum of gradient images $gra_m$ for each feature type. $G_x$ and $G_y$ are the horizontal and vertical Sobel kernels. *num* is computed as the number of pixels recognized by *gra* in the same patch.

The result of our adaptive window selection is shown in Fig.4. It is evident that the gradient image extracted most fine details. In the third image of Fig.4, we show the result by way of recognizing complex areas. The red pixels are recognized to be reconstructed in a window of radius $r_s$ and the green pixels using windows of larger radii. It is clear that our metric varies the window size based on pixel complexity.

# 4 RESULTS AND DISCUSSION

We integrated our method as an extension with the PBRT (Pharr and Humphreys, 2010) and employed CUDA to accelerate our model. We used an Inter CORE-i7 with 8 GB of RAM and a GeForce 650 M GPU. Three state-of-the-art methods were used for comparison: SBF (Li et al., 2012), WLR (Moon et al., 2014), and NFOR (Bitterli et al., 2016). We used rMSE (Rousselle et al., 2011) to measure numerical error $(out - gt)^2/(gt^2 + \varepsilon)$, where *out* and *gt* are the filtered value and the ground truth, respectively. $\varepsilon = 0.01$ was used to prevent division by zero.

All images were rendered at a resolution of $800 \times 800$ pixels. For our experiments, the patch size $r$ and threshold $\tau$ were set to 3 and 0.4, respectively. The candidate window radii were set to $\{5, 10, 15\}$. For our method, there are two parameters required to be specified by users: sample number per pixel and the candidate bandwidth values (which were set to $k = \{1.0, 1.5, 2.0, 3.5, 4.0\}$ in our experiments).

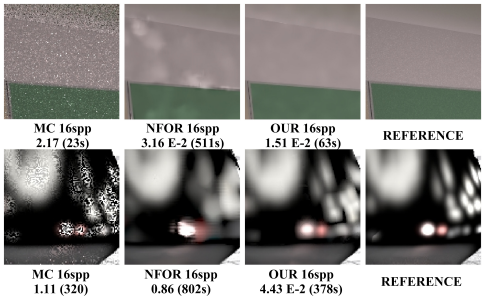


Figure 5: Comparisons between our algorithm and NFOR. NFOR tended to blur fine details and failed to remove spike noise.

## 4.1 Scenes

In Fig.5, we compare our method with NFOR. In the second row of insets, it is clear that NFOR over blurred image details while the results of our method were closer to the reference image. Moreover, NFOR could not remove spike noise (first row of insets) as it dose not recognize outliers. Note that we intend NFOR as a CPU-based method and, thus, it requires a long time to complete reconstruction.

In Fig.6, the well-known "SANMIGUEL" scene containing complex geometries is compared in the first two rows. SBF produced substantial noise at low sampling rates, especially in background areas (the first row of insets). It also failed to preserve fine detail of the chair (second row of insets) because of inaccurate SURE estimates. WLR outperformed SBF but blurred details that did not have strong correlations with the features. It did not weight the features appropriately either, leading to numerous splotches. Due to the complementary strategy using the correlation between features and colors, our method performed better. Our regression weights helped recognize details that were not captured by the features and, thus, returned clean details.

The last two rows of Fig.6 simulated two specific rendering effects: depth of field and motion blur. It is clear that both SBF and WLR failed to remove noise in strongly defocused and motion-blurred areas. Our method, however, provided robust input to fit data. As a result, our method maintained the clarity of edges, and was closer to the reference images.

## 4.2 Computational Overhead

To reduce cost, we wrote a GPU-based model for acceleration. Moreover, The iterative number was set to 2 as it provided pleasing results in most cases. For

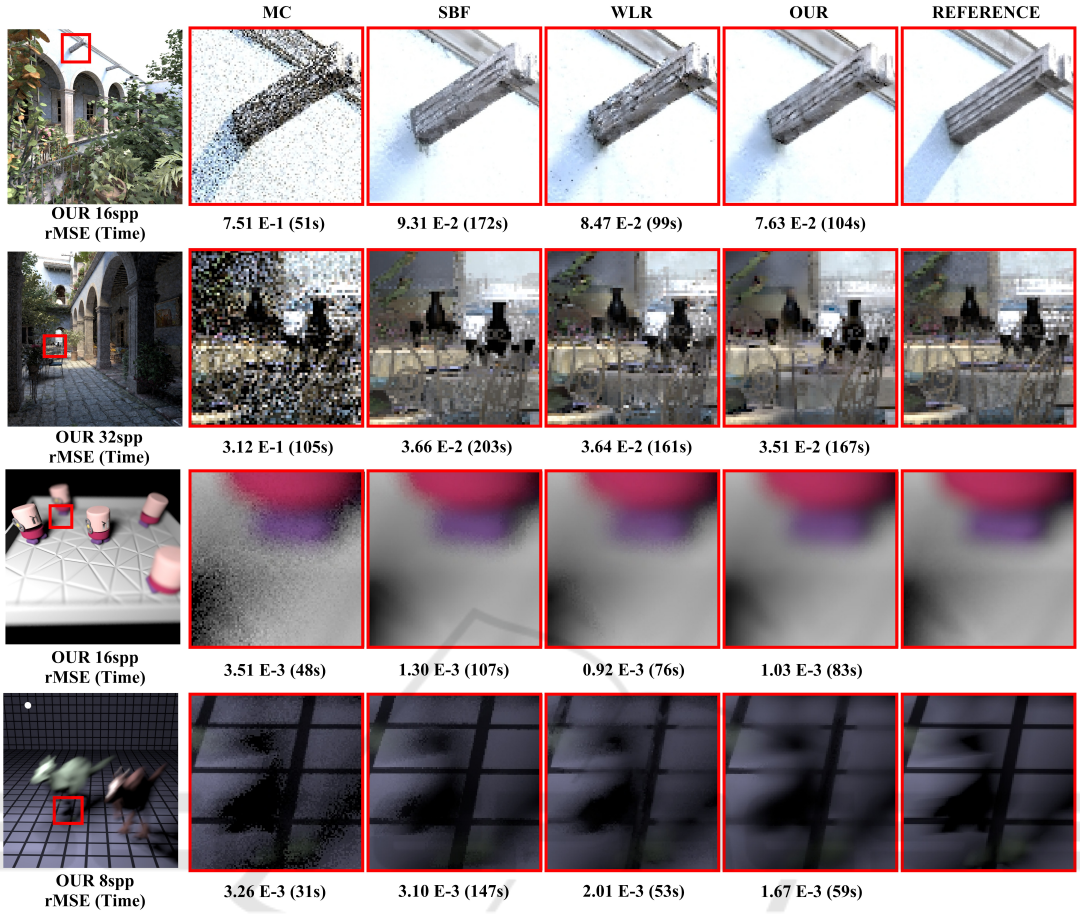


Figure 6: Comparisons of our method with prior methods.

our method, window size is varied at different pixels to further reduce cost. For example, only 34% of the pixels were recognized to use small reconstruction windows in the "DRAGON" scene (Fig. 4). In this case, each model of our method predicted 370 pixels on average. Our method hence outperforms WLR at the cost of a little more time.

### 4.3 Numerical Error Comparisons

To reduce the rMSEs, our method focuses on two points. First, our prefiltering process removes feature noise to produce a robust input for the model. Second, we iteratively estimate and reduce the reconstruction error at multiple pixels. In this case, as shown in Fig.5 and Fig.6, the lower numerical errors are produced. In experiments, however, we also found that a relatively large rMSE may be produced when our method failed to recognize fine details from noise, especially for scenes with complex geometries. We intent to solve this problem by using novel features (e.g., visibility and gradient) in our future work.

### 4.4 Limitations

A limitation of our method is that the correlation between features and colors may be not strong at low sampling rates. Moreover, prefiltered features may be overblurred and lead to blurred results. We believe this problem can be solved by constructing a second feature type (e.g., gradient) that is less influenced by noise.

## 5 CONCLUSIONS AND FUTURE WORK

To reduce the impact of noise, pixel colors are denoised using a GIF. In addition, we combine the joint-NL-means filter with GIF to handle noisy features. Our model uses an iterative process to estimate the reconstruction error. Finally, the window size is selected adaptively to reduce the computational overhead. Experimental results demonstrate that the new algorithm improves the image quality greatly and can

handle a wide variety of rendering effects.

Overfitting to noise is a major challenge. Our future work will extend the supervised learning approaches such as neural network. In addition, we intend to compute the optimal bandwidth using a consistent metric instead of selecting from a predefined candidate set.

## REFERENCES


Bako, S., Vogels, T., Mcwilliams, B., Meyer, M., NováK, J., Harvill, A., Sen, P., Derose, T., and Rousselle, F. (2017). Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Trans. Graph.*, 36(4):97:1–97:14.

Belcour, L., Soler, C., Subr, K., Holzschuch, N., and Durand, F. (2013). 5D covariance tracing for efficient defocus and motion blur. *ACM Trans. Graph.*, 32(3):31:1–31:18.

Bitterli, B., Rousselle, F., Moon, B., Iglesias-Guitin, J. A., Adler, D., Mitchell, K., Jarosz, W., and Novk, J. (2016). Nonlinearly weighted first-order regression for denoising Monte Carlo renderings. *Computer Graphics Forum (Proceedings of EGSR)*, 35(4):107117.

Chaitanya, C. R. A., Kaplanyan, A. S., and Schied, C. (2017). Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graph.*, 36(4):98:1–98:12.

Delbracio, M., Musé, P., Buades, A., Chauvier, J., Phelps, N., and Morel, J.-M. (2014). Boosting Monte Carlo rendering by ray histogram fusion. *ACM Trans. Graph.*, 33(1):8:1–8:15.

Durand, F., Holzschuch, N., Soler, C., Chan, E., and Sillion, F. X. (2005). A frequency analysis of light transport. *ACM Trans. Graph.*, 24(3):1115–1126.

Egan, K., Hecht, F., Durand, F., and Ramamoorthi, R. (2011). Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Trans. Graph.*, 30(2):9:1–9:13.

Egan, K., Tseng, Y.-T., Holzschuch, N., Durand, F., and Ramamoorthi, R. (2009). Frequency analysis and sheared reconstruction for rendering motion blur. *ACM Transactions on Graphics (SIGGRAPH 09)*, 28(3).

Hachisuka, T., Jarosz, W., Weistroffer, R. P., Dale, K., Humphreys, G., Zwicker, M., and Jensen, H. W. (2008). Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph.*, 27(3):33:1–33:10.

He, K., Sun, J., and Tang, X. (2010). Guided image filtering. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, pages 1–14, Berlin, Heidelberg. Springer-Verlag.

Kajiya, J. T. (1986). The rendering equation. In *In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA. ACM.

Kalantari, N. K., Bako, S., and Sen, P. (2015). A machine learning approach for filtering monte carlo noise. *ACM Trans. Graph.*, 34(4):122:1–122:12.

Kalantari, N. K. and Sen, P. (2013). Removing the noise in monte carlo rendering with general image denoising algorithms. *Computer Graphics Forum*, 32(2pt1):93102.

Lehtinen, J., Aila, T., Chen, J., Laine, S., and Durand, F. (2011). Temporal light field reconstruction for rendering distribution effects. *ACM Trans. Graph.*, 30(4):55:1–55:12.

Li, T.-M., Wu, Y.-T., and Chuang, Y.-Y. (2012). Sure-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph.*, 31(6):194:1–194:9.

Liu, Y., Zheng, C., Zheng, Q., and Yuan, H. (2017). Removing monte carlo noise using a sobel operator and a guided image filter. *The Visual Computer*.

Moon, B., Carr, N., and Yoon, S.-E. (2014). Adaptive rendering based on weighted local regression. *ACM Trans. Graph.*, 33(5):170:1–170:14.

Moon, B., Iglesias-Guitian, J. A., Yoon, S.-E., and Mitchell, K. (2015). Adaptive rendering with linear predictions. *ACM Trans. Graph.*, 34(4):121:1–121:11.

Moon, B., Jun, J. Y., Lee, J., Kim, K., Hachisuka, T., and Yoon, S. (2013). Robust image denoising using a virtual flash image for Monte Carlo ray tracing. *Comput. Graph. Forum*, 32(1):139–151.

Moon, B., McDonagh, S., Mitchell, K., and Gross, M. (2016). Adaptive polynomial rendering. *ACM Trans. Graph.*, 35(4):40:1–40:10.

Pharr, M. and Humphreys, G. (2010). *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco.

Rousselle, F., Knaus, C., and Zwicker, M. (2011). Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph.*, 30(6):159:1–159:12.

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

Sen, P. and Darabi, S. (2012). On filtering the noise from the random parameters in monte carlo rendering. *ACM Trans. Graph.*, 31(3):18:1–18:15.

Soler, C., Subr, K., Durand, F., Holzschuch, N., and Sillion, F. (2009). Fourier depth of field. *ACM Trans. Graph.*, 28(2):18:1–18:12.

Zwicker, M., Jarosz, W., Lehtinen, J., Moon, B., Ramamoorthi, R., Rousselle, F., Sen, P., Soler, C., and Yoon, S.-E. (2015). Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Comput. Graph. Forum*, 34(2):667–681.