Vis Comput
DOI 10.1007/s00371-017-1363-z

ORIGINAL ARTICLE

Removing Monte Carlo noise using a Sobel operator and a guided image filter

Yu Liu[1,2] · **Changwen Zheng**[2] · **Quan Zheng**[2] · **Hongliang Yuan**[2]

Abstract In this study, a novel adaptive rendering approach is proposed to remove Monte Carlo noise while preserving image details through a feature-based reconstruction. First, noise in the additional features is removed using a guided image filter that reduces the impact of noisy features involving strong motion blur or depth of field. The Sobel operator is then employed to recognize the geometric structures by robustly computing a gradient buffer for each feature. Given the gradient information for high-dimensional features, we compute the optimal filter parameters using a data-driven method. Finally, an error analysis is derived through a two-step smoothing strategy to produce a smooth image and guide the adaptive sampling process. Experimental results indicate that our approach outperforms state-of-the-art methods in terms of visual image quality and numerical error.

Keywords Adaptive sampling and reconstruction · Guided image filter · Sobel operator · Ray tracing

1 Introduction

Monte Carlo (MC) ray tracing is among the most effective techniques for producing photorealistic images. This method computes a complex integral at each pixel by randomly sampling the multidimensional integration domain. However, unless an excessive number of ray samples is distributed, MC renderings converge slowly and suffer from noise artifacts, i.e., variance at low sampling rates. To gain efficiency, a large number of variance reduction approaches have been proposed, of which adaptive sampling and reconstruction methods [1–3] have shown their effectiveness at removing noise while preserving details.

Adaptive sampling and reconstruction methods typically use a small number of samples to quickly render a noisy image and filter it through post-processing. Furthermore, the filtered result can be used as feedback to direct the distribution of additional ray samples. For example, Rousselle et al. [3] greedily minimized the relative mean squared error (rMSE) by selecting suitable per pixel filters. However, this method is limited to symmetric reconstruction kernels (e.g., Gaussian filters). Recently, the most successful approaches considered additional features such as depths, textures, and normals to determine the filter weights. Because these features are typically less noisy relative to pixel colors and are highly correlated with scene details, filters using features (e.g., cross-bilateral filters and cross-non-local-means filters) significantly improve image quality. As a result, many approaches apply novel theories such as weighted local regression [4], machine learning approaches [5], and Stein's unbiased risk estimator (SURE) [1] to estimate per pixel errors and then select the ideal filter parameters. However, because the optimal parameters are often spatially variable, robust selection is challenging. Another drawback of these methods involves inaccurate and noisy error estimates at low sampling rates, which reduces the robustness of filter selection. Moreover, the features can also be rather noisy, especially when there are complex motions, textures, and geometries. These noisy features are prone to blurring image details that are not well represented by features. Recently, to deal with the problem of spatially varying parameters,

✉ Yu Liu
lyiscas@163.com

Changwen Zheng
243434171@qq.com

1 University of Chinese Academy of Sciences, Beijing, China

2 Institute of Software, Chinese Academy of Sciences, Beijing, China

Bauszat et al. [6] proposed a method to construct dense error prediction from a small set of sparse estimates.

Based on the above observations, we propose a novel feature-based approach to handle a wide variety of MC rendering effects. One of our main contributions is to extend an effective image-denoising filter (guided image filter) [7] and use it to prefilter the feature images. To remove the noise in features, the Sobel operator is first employed to compute a gradient image that typically has edges consistent with the ground truth. The feature images are then prefiltered through a guided image filter with gradient image as the guidance image. Furthermore, we also use the gradient information to recognize spike pixels, thereby alleviating the influence of spike noise. To select optimal filter parameters, the spatial and feature parameters are computed in two separate steps. First, the spatial parameter is computed using a simplified parametric error estimation that fits the parametric curves for bias and variance, respectively. This procedure enables our method to robustly and consistently select the filter parameter. Given the selected spatial parameter, we then evaluate a few candidate filters using different feature parameters and compute their weighted average using a two-step smoothing strategy. Experimental results showed that our new method is superior to previous methods on a wide variety of rendering effects.

2 Related works

Recent works involving adaptive sampling and reconstruction achieved impressive results; however, the primary goal remains the same: removing MC noise while faithfully preserving image details. These works can be classified into two categories: multidimensional and image-space rendering.

Multidimensional space rendering Kajiya [8] proposed a method to allocate high-dimensional samples using kd-trees to reconstruct the outputs. Numerous approaches have since been presented to operate in a multidimensional space. For example, Hachisuka et al. [9] used a structure tensor to perform an anisotropic reconstruction, which exhibited poor effectiveness as the dimensions increased. Durand et al. [10] described how the frequency content of radiance was altered by phenomena such as transport, occlusion, and shading. Based on the work of Durand et al. [10], many methods were tailored to improve image quality for specific effects such as depth of field [11], motion blur [12], soft shadow [13], and ambient occlusions [14]. Belcour et al. [15] presented a five-dimensional frequency analysis of a temporal light field to support depth of field and motion blur. Lehtinen et al. [16] described a reconstruction technique using depth and motion information to handle a wide variety of specific effects. Moreover, they also reused ray samples to produce better results for indirect illuminations [17]. Kettunen et al. [18] proposed frequency analysis that compares the MC sampling of gradients followed by Poisson reconstruction with traditional MC sampling. They showed that it is beneficial to directly estimate image gradients with correlated samples. Manzi et al. [19] reconstructed images by solving a screened Poisson problem leveraging feature patches to regularize the solution. Sen et al. [20] proposed an interesting method to ray trace only a subset of pixels followed by the determination of missing pixels using a compressed sensing method, which failed at low sampling rates. Liu et al. [21] coarsely sampled the entire multidimensional space and used a kd-tree to refine the space. The main drawback of these multidimensional methods is that they typically only support a limited set of distributed effects and tend to be restricted by the so-called curse of dimensionality.

Image-space rendering Since Mitchell [22] laid the foundation for image-space rendering, this method has received additional attention due to its simplicity as well as efficiency. A common strategy is to render a noisy image with a few samples and then denoise it using a post process. In this case, many algorithms are inspired by the denoising techniques used by the image-processing community, where powerful filters such as cross-bilateral filters [23], non-local-means filters [24,25], and wavelet shrinkage [26] are employed to remove noise. For example, Rousselle et al.'s algorithm [3] is based on adaptive bandwidth selection with isotropic Gaussian filters. Kalantari et al. [27] proposed a method that enabled the use of any spatially invariant image-denoising techniques to denoise MC renderings.

Recently, approaches leveraging additional features (e.g., normals, textures, depths, world positions) have become popular. Their main purpose is to select the optimal filter parameters to enable a trade-off between noise reduction and detail fidelity. Li et al. [1] introduced the use of SURE to select the suitable spatial parameter from a predefined candidate set. Unfortunately, SURE estimates can be unreliable at low sampling rates, leading to suboptimal results. Rousselle et al. [25] split the samples into two buffers, with the difference between the buffers treated as an estimated error. Sen et al. [28] calculated the statistical dependency between the parameters and their outputs, and used this information to reduce the importance of samples affected by noise. Rousselle et al. [2] evaluated three candidate filters using different parameters and computed their weighted average for output. Bauszat et al. [29] used edge-aware filtering in the sample space to reduce noise in the presence of depth-of-field effects at low sampling rates. Liu et al. [30,31] used polynomial reconstruction to produce smooth image details. Moon et al. [32] used a virtual flash image as an edge-stopping function to recognize homogeneous pixels and preserve details.

Most recently, Moon et al. [4] employed weighted local regression to compute the optimal filter parameters on a reduced feature space; however, this method may underestimate local dimensions with truncated singular value decomposition (TSVD), leading to overblurred details. Kalantari et al. [5] trained a neural network by analyzing the relationship between the ideal parameters and the scene data, followed by employing the network to estimate the appropriate parameters. Bitterli et al. [33] proposed prefiltering features with an NL-means filter. They also employed a first-order model with a nonlinear regression kernel to reconstruct images. To approximate the incident indirect illumination, Bauszat et al. [34] filtered the noisy image using a guided image filter with the depth and normal map as the guidance image. Delbracio et al. [35] computed a ray histogram to determine whether two pixels can share rays. To remove noise while preserving high-frequency edges, Moon et al. [36] built several linear models using different prediction windows, and the window returning the smallest error was finally chosen. Most recently, Moon et al. [37] noticed that high-order functions may produce better images than low-order functions. In this case, they locally selected the polynomial function order in place of filter bandwidth to improve image quality. Despite the impressive results of these feature-based methods, their error estimates are noisy at low sampling rates. Therefore, it remains challenging to robustly select optimal filter parameters in complex cases. Zwicker et al. [38] provided a detailed description of recent advances in adaptive rendering approaches.

3 Overview

Feature-based filtering methods are effective at removing noise, especially for scenes with clear structural details. Similar to prior methods, we compute the filtered image $\widehat{C}$ at pixel i as a weighted average of its neighborhood $N(i)$ centered at i:

$$\widehat{C}_i = \frac{\sum_{j \in N(i)} d_{i,j} C_j}{\sum_{j \in N(i)} d_{i,j}} \tag{1}$$

where $d_{i,j}$ is the weight term between pixel i and j, and C_j is the sample mean of pixel j. We compute the weight term by employing spatial, color and feature components as follows:

$$d_{i,j} = \exp\left(-\frac{||s_i - s_j||^2}{2\alpha_i^2}\right) \exp\left(-\frac{D(C_i, C_j)}{2\beta_i^2}\right) \prod_{k=1}^{m} \exp\left(-\frac{D_f(\overline{f}_{i,k}, \overline{f}_{j,k})}{2\gamma_{i,k}^2}\right) \tag{2}$$

where s_i and $\overline{f}_{i,k}$ denote the screen position and the sample mean of the kth feature at pixel i, respectively. α_i^2, β_i^2, and $\gamma_{i,k}^2$ are the variances for spatial, color, and the kth feature terms, respectively. Since these parameters control the bandwidths of the filter, the main difference between previous methods is how they select the optimal values for these parameters. Here, we define the cross-bilateral filtering process as: $\widehat{C} = Fil(C, \alpha, \beta, \gamma)$ where the noisy input C is filtered using a cross-bilateral filter with parameters α, β, and γ.

To enhance the effectiveness of noisy features, our method first filters them with a guided image filter (Sect. 4), which uses the Sobel operator to compute a gradient image as the guidance image. The main intuition of our gradient image is to extract fine details presented by different features. We then compute α_i and $\gamma_{i,k}$ separately on a per pixel basis. For α_i, we use a parametric curve fitting to consistently produce values (Sect. 5.1). $\gamma_{i,k}$ is then estimated as a weighted average of a predefined candidate set (Sect. 5.2), which creates a trade-off between noise reduction and detail fidelity. Finally, to suppress outliers that can lead to spike noise, a two-radius strategy is proposed to reject spike pixels and down-weight them to remove spike noise (Sect. 6).

4 Feature prefiltering

In this section, we describe our algorithm for prefiltering feature images using a guided image filter [7], which exhibits a gradient-preserving characteristic and can be regarded as a special case of weighted linear regression. The key assumption is that each pixel in a local window w_j centered at pixel j is a linear transform of the same window in a guidance image I:

$$F_i = a_j I_i + b_j, \forall i \in w_j \tag{3}$$

where a_j and b_j are constant linear coefficients in w_j. Typically, guided image filter requires a regularization parameter to prevent a_j from being too large, and we set it to 0.001 for all tested scenes. After computing these parameters using linear regression, we average all coefficients for each overlapping window at each pixel. Here, the guided image filtering is summarized as: $F = GUID(C, I)$, which describes the input image C being filtered using a guided image filter, with I as the guidance image.

To filter the initial features using a guided image filter, a robust guidance image is needed. Instead of using the noisy features, we utilize the Sobel operator to construct a gradient image. Compared with the input features, our gradient image remains consistent with the ground truth and can extract fine details.

The Sobel operator is a discrete differentiation operator that is commonly used in computer vision to detect edges.

Technically, it convolves the input image with a small and separable filter kernel to calculate two approximations of derivatives: one for horizontal changes and another for vertical changes. In this paper, the gradient image gra is computed as the maximum of gradient images gra_k for each feature as follows:

$$gra_k = \sqrt{(G_x * \overline{f}_k)^2 + (G_y * \overline{f}_k)^2}$$
$$gra = max\{gra_k\} \tag{4}$$

where $\overline{f}_k$ is the kth input feature image constructed by the sample mean, and G_x and G_y are the horizontal and vertical Sobel kernels, respectively.

We observe that different features may present different geometric edges. For example, while the input depth image does not show clean details on the window of the "San Miguel" in Fig. 1, our method considers other features to obtain these details. As a result, we use three features (i.e., textures, normals, and depths) in order to extract fine details.

Figure 1 shows the input features (first row of the insets) and the related gradient images (second row of the insets). Note that only one feature image is shown for each benchmark. In practice, however, we consider all three features to compute the gradient image. Pay attention to the motion blurred and defocused regions of the first two benchmarks. It is obvious that our gradient image contains more details than the input feature images, even with a small number of ray samples. As a result, we prefilter the noisy features as follows:

$$\widehat{f_k} = GUID(\overline{f}_k, gra) \tag{5}$$

where $\widehat{f_k}$ is the kth filtered feature image. In our implementation, the window size of guided image filter is set to 7.

The result of our feature prefiltering is shown in Fig. 2. As shown in the figure, the initial normal image is rather noisy in areas with strong depth of field. The noisy features tend to degrade image details without denoising. Using our guided image filter, as shown in the figure, noise was effectively removed while fine feature details in the area of focus were preserved. The guided image filter was also used in past work [34] to approximate incident indirect illumination.

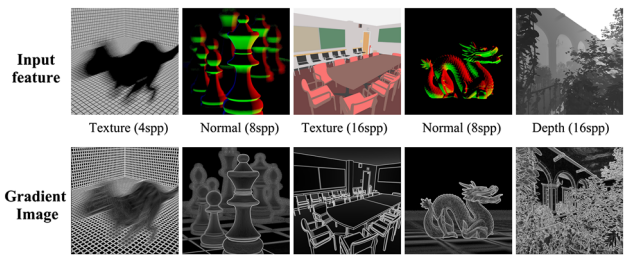


Fig. 1 The gradient image of our method

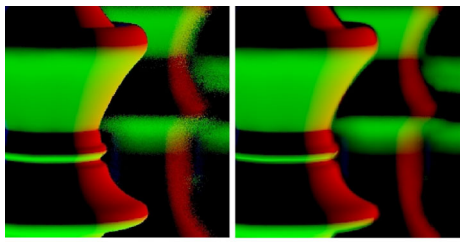


Fig. 2 Noise removal for features. The initial features are noisy at regions with strong depth of field. Our method removes the noise while preserving feature details in focused areas

Nevertheless, the authors used the initial normal and depth map to construct a guidance image, which contained a lot of noise for specific effects. Our method, on the other hand, prefilters them to suppress the impact of noisy features.

5 Parameter selection

As shown in Eq. 2, the main challenge is how to select optimal filter parameters. Given the filtered features, we select them on a per pixel basis. The spatial parameter is computed through a parametric curve fitting, which enables us to produce consecutive results. For the feature parameters, they are computed as a weighted average of the candidate filters using a two-step smoothing strategy to make a trade-off between noise reduction and detail fidelity.

5.1 Spatial parameter

Moon et al. [4] decomposed numerical error into bias and variance terms using the local regression literature [39], and then estimated the terms through a parametric curve fitting procedure. However, their method relies upon a reduced feature space, where the estimated value is a scaling factor shared across all feature types. As a result, the selected spatial parameter may be affected by other noisy features. In this case, we only employ parametric curve fitting to compute the spatial parameter in a local space, allowing us to eliminate the influence of other features.

At the first step, we filter the initial image using a set of test spatial values α_t. In this step, the feature parameters are set to constant values. This returns a paired list of α_t and the corresponding biases, as well as variances. As mentioned in the previous work [4], the bias and variance terms contain the following asymptotic parametric functions of given bandwidth α_t, respectively:

$$bias(\alpha_t, i) = b_0 + b_1\alpha_t^2$$
$$var(\alpha_t, i) = \frac{1}{n(i)}\left(v_0 + \frac{v_1}{\alpha_t^2}\right) \tag{6}$$

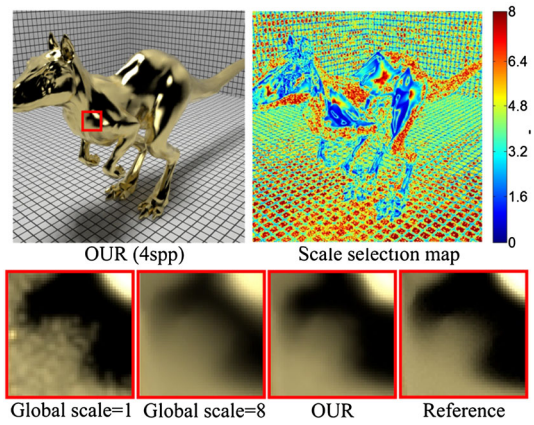


Fig. 3 Visualization of the spatial scale selection map. Our method enables a trade-off between noise reduction and detail fidelity

where b_0 and b_1 are the curve parameters for bias, v_0 and v_1 are those for variance, and $n(i)$ denotes the current number of samples at pixel i. The bias and variance terms can be estimated as: $bias(\alpha_t, i) = \frac{\sum_{j\in N(i)} d_{i,j,c_j}}{\sum_{j\in N(i)} d_{i,j}} - c_i$ and $var(\alpha_t, i) = \sum_{j\in N(i)} (d_{i,j})^2 var_j$, respectively. Here, $d_{i,j}$ represents the filter weight between i and j using α_t, and var_j is the variance of the sample mean at pixel j. c_j is the filtered pixel color from a box filter, which is typically considered as unbiased estimation of the ground truth.

After each α_t is tested, the four curve parameters (i.e., b_0, b_1, v_0, and v_1) can be computed by ordinary least squares. The optimal spatial parameter at pixel i is then computed as $\alpha_{opt} = \frac{v_1}{2b_1^2 n(i)}^{\frac{1}{6}}$ based on $\frac{\partial(bias^2+var)}{\partial\alpha_t} = 0$. The main difference between our metric and WLR in Eq. 6 is that WLR computes a scaling factor shared across spatial and feature spaces. Our method, however, restricts the result to be determined only by spatial variations, and thus returns a robust spatial parameter without the influence of noisy features.

Figure 3 shows our scale selection map for the spatial parameter. This figure compares our result with those from two global cross-bilateral kernels, where the same parameter is used at each pixel. The results indicate that the small parameter (scale=1) is unable to remove noise, while a larger parameter (scale=8) tends to over smooth images. Based on our local parametric analysis, our results adaptively select spatial parameters and preserve fine details while successfully removing noise.

To compute $d_{i,j}$, the function D_f measures the feature distance between two pixels. Here, we use the gradient information to suppress the influence of noisy features.

$$D_f(\overline{f}_{i,k}, \overline{f}_{j,k}) = \frac{|\widehat{f}_{i,k} - \widehat{f}_{j,k}|^2}{\sigma_{i,k}^2 + \sigma_{j,k}^2 + min\left(gra_i{}^2, 0.01\right)} \tag{7}$$

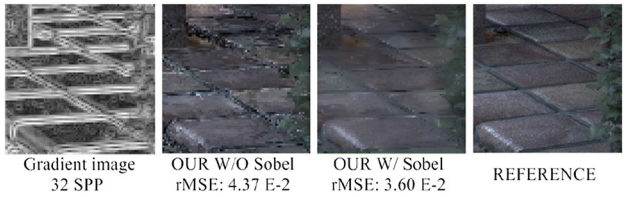


Fig. 4 Filtering results using gradient information. Our distance function uses the gradient information to compute the filter weights, resulting in smoother details

where $\sigma_{i,k}^2$ and $\sigma_{j,k}^2$ are sample variances of the kth feature at pixel i and j, respectively. $\widehat{f}_{i,k}$ denotes the kth filtered feature at pixel i, and gra_i represents the value of gra at pixel i. Our metric controls the residual variance of the filtered feature by considering the gradient magnitude. Thus, for a pixel with strong motion blur or depth of field, a larger filter weight can be returned even when the features are far apart. Rousselle et al. [2] proposed a similar metric to reduce the impact of noisy features, which requires a suitable user-defined parameter to control the sensitivity of the filter to feature differences. Our metric returns smaller distances when the estimated gradients are high. As a result, it tends to produce smoother results (Fig. 4).

5.2 Feature parameters

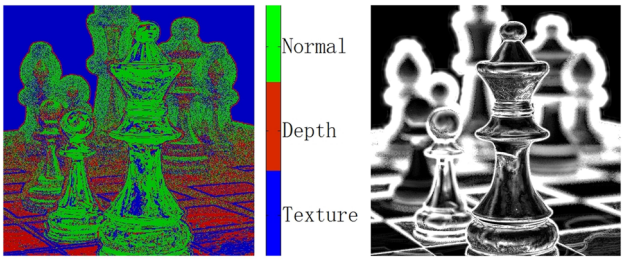
To compute optimal feature parameters, a straightforward strategy involves fitting the parametric curve to each feature. However, large amounts of time and memory are consumed as more features are considered. To solve this problem, we adopt a common strategy to select a suitable parameter with the lowest error from a predefined candidate set. This strategy, unfortunately, tends to produce noisy error estimates that may lead to visual artifacts in the form of seams. In this case, we smooth the error estimates and compute a weighted average of the candidate filters for output. Specifically, our metric measures the varying importance of different features and then estimates per pixel errors.

We define l candidate feature parameters as $s = \{s_1, s_2..s_l\}$, where each value is employed to compute its related filtered image $\widehat{c}_j = Fil(C, \alpha_{opt}, 0, s_j), \forall s_j \in s$. Note that the optimal spatial parameter α_{opt} is used at each pixel in this part. To estimate the error $err_{i,j}$ introduced with s_j at pixel i, a simple error estimation metric is described as follows.

First, the initial image is filtered using three guided image filters, with each filter using the normal, texture, or depth image as the guidance image.

$$fea_k = GUID(C, \widehat{f}_k) \tag{8}$$

where fea_k denotes the filtered image of using $\widehat{f}_k$ as the guidance image.



(a) **(b)**

Fig. 5 **a** Feature choice and **b** sampling density map. Our method considers the varying importance of different features to estimate per pixel errors

Then we measure the difference $\lambda_{i,k}$ between $fea_{i,k}$ and c_i at pixel i as:

$$\lambda_{i,k} = \frac{|fea_{i,k} - c_i|}{c_i}$$
$$ref_i = \{fea_{i,k} | min(\lambda_{i,k})\} \tag{9}$$

where c_i is the observed pixel value obtained from a box filter, and $fea_{i,k}$ is the value of fea_k at pixel i. Typically, c_i is considered as unbiased estimation of the ground truth [3]. As a result, if the difference is small, we consider the given feature to contain more scene details. In this case, the single image fea_k that returns the smallest difference at each pixel is saved to construct a reference image ref, which is used to estimate per pixel errors:

$$err_{i,j} = \frac{|\widehat{c}_{i,j} - ref_i|^2}{ref_i^2 + \varepsilon} \tag{10}$$

where $\widehat{c}_{i,j}$ is the value of $\widehat{c}_j$ at pixel i, and ε is a small number to prevent division by zero. Figure 5a shows the feature choice involved in our method.

As described above, selecting the candidate filter with the lowest $err_{i,j}$ can lead to seams due to filter changes. Moreover, the initial error estimates can be very noisy and need to be filtered with a large spatial kernel, which tends to introduce bias in error estimates. In these cases, we present a two-step smoothing strategy to smooth the initial error estimates, which is similar to prior work [2].

First, the initial error estimates are filtered using a small guided image filter with the initial error estimates as the guidance image, resulting in l binary maps. In the jth binary map, pixel i equals to one if $err_{i,j}$ has the smallest error. Second, these binary maps are filtered with a larger guided image filter kernel, which helps suppress outliers (Sect. 7). The filtered results are used as weight terms to compute a weighted average of the predefined candidates:

$$out_i = \frac{\sum_{j=1}^{l} wei_{i,j}\widehat{c}_{i,j}}{\sum_{j=1}^{l} wei_{i,j}} \tag{11}$$

where $wei_{i,j}$ is the filtered error estimate using the two-step smoothing strategy, and out is our final reconstructed image. This smoothing step of error estimates allows our method to avoid seams and produce consistent details.

6 Extensions

6.1 Removing spike noise

MC renderings may contain spike pixels that have radiance values much larger than their neighbors. Intuitively, spike noise can be removed by using a large filter, which tends to over smooth details and introduce bias. In this case, we use the gradient information to recognize spike pixels and reduce their weight terms. First, the mean and standard deviation of each pixel in the initial image are computed in a local window of size 5×5. The spike pixels are then recognized as:

$$spi_i = |mean_i - C_i| - max(gra_i, sd_i) \tag{12}$$

where $mean_i$ and sd_i are the mean and standard deviation at pixel i in C, respectively. If $spi_i > 0$, pixel i is labeled as a spike pixel.

A similar method was proposed by Kalantari et al. [5]. They replaced the filtered value of a spike pixel with its median window color. The major difference between these two methods is that we select the suitable parameters by regression on the bias-variance trade-off, and LBF does it using a trained mapping between the local statistics and optimal parameters. For our method, however, we found that residual splotches may be produced if we directly used the median block color (as shown in Fig. 6). In this case, we used a two-radius strategy to further remove spike noise. We define pixel i ($spi_i > 0$) as the first spike pixel, with its neighbors in a window of size 10×10 defined as the second spike pixels. As a result, the influence of spike noise is divided into two parts. The first part is derived from the first spike pixels, and their filter weights are set to zero. The second part is derived from the second spike pixels, which may also have high dynamic range; therefore, we set their filter weights to half of the original values. Figure 6 demonstrates our spike removal results. It indicates that our method can robustly recognize spike pixels and successfully remove spike noise using the two-radius strategy.

6.2 Adaptive sampling

We adopt a common strategy [3] to distribute additional ray samples to high error areas. During the initial pass, a small

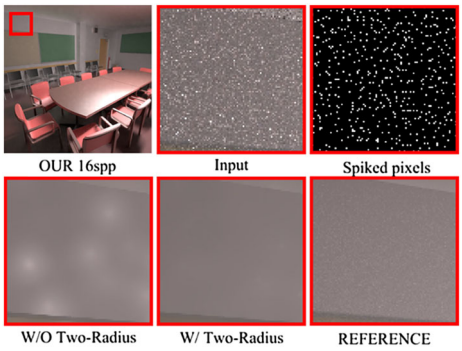


Fig. 6 Recognizing spike pixels. Our method recognizes spike pixels in the input image and removes the noise by reducing their filter weights

number of ray samples (e.g., four samples per pixel) are distributed uniformly. In subsequent iterations, the available samples are distributed based on the following metric:

$$Sam_i = \frac{|out_i - c_i|^2 + var_i}{out_i^2 + \epsilon} \tag{13}$$

where ϵ is a small number used to prevent division by zero (set to 0.001), and out_i is the filtered output in the prior iteration. Our metric allocates more samples to darker areas by considering the squared luminance of the reconstructed pixel color. As a result, pixel i obtains $\frac{dSam_i}{\sum_j Sam_j}$ samples in the next iteration if there are d samples available. Figure 5b shows the sampling density map and demonstrates that samples are mainly distributed to regions with discontinuities and depth of field.

7 Algorithm

The pipeline associated with our method is summarized in Algorithm 1. Given the noisy color input C, initial features $\overline{f}_k$ and their related variances σ_k^2, our algorithm produces the final image out. The gradient image gra is first computed by applying a Sobel operator to each feature. The initial features are then filtered using a guided image filter with gra as the guidance image. In this step, spike pixels are also recognized using Eq. 12.

In the second step, five test spatial parameters ($\alpha_t = \{0.1, 1, 2, 4, 8\}$) are used to filter the input image. In this step, the feature parameters are set to constant values ($\gamma = \{0.2, 0.3, 0.4\}$). Our cross-bilateral filter produces five filtered images F_t, as well as the related bias maps $bias(\alpha_t)$ and variance maps $var(\alpha_t)$. We then fit the parametric curves for bias ($Biasfit$) and variance ($Varfit$) at each pixel, where the calculated coefficients are used to estimate the optimal spatial parameter α_{opt}.

Algorithm 1 Removing Monte Carlo Noise Using a Sobel Operator and a Guided Image Filter

Require: Noisy color input C, Noisy feature input $\overline{f}_k$ with sample variance σ_k^2, current sample number n, output using a box filter c
Ensure: reconstructed image out

function FEATURE- PREFILTER($\overline{f}_k$,G_x,G_y)
  $gra_k = \sqrt{(G_x * \overline{f}_k)^2 + (G_y * \overline{f}_k)^2}$
  $gra = max\{gra_k\}$
  $\widehat{f_k} = GUID(\overline{f}_k, gra)$
end function
function SPATIAL PARAMETER(C,$\widehat{f_k}$,gra,n,α_t,γ)
  for $t = 1 \rightarrow 5$ **do**
    $\{F_t, bias(\alpha_t), var(\alpha_t)\} = Fil(C, \alpha_t, 0, \gamma)$
  end for
  $\{b_0, b_1\} = Biasfit(bias(\alpha_t), \alpha_t)$
  $\{v_0, v_1\} = Varfit(var(\alpha_t), \alpha_t)$
  $\alpha_{opt} = \frac{v_1}{2b_1^2 n(i)}^{\frac{1}{6}}$
end function
function FEATURE PARAMETERS(C,gra,α_{opt},$\widehat{f_k}$,s,ε,c)
  for $k = 1 \rightarrow 3$ **do**
    $fea_k = GUID(C, \widehat{f_k})$
    $\lambda_{i,k} = \frac{|fea_{i,k} - c_i|}{c_i}$
  end for
  $ref_i = \{fea_{i,k} | min(\lambda_{i,k})\}$
  for $j = 1 \rightarrow l$ **do**
    $\widehat{c}_j = Fil(C, \alpha_{opt}, 0, s_j)$
    $err_{i,j} = \frac{|\widehat{c}_{i,j} - ref_i|^2}{ref_i^2 + \varepsilon}$
  end for
end function
function TWO- STEP- SMOOTHING(C,gra,α_{opt},$\widehat{f_k}$,$err_{i,j}$)
  for $j = 1 \rightarrow l$ **do**
    $err_j = GUID(err_j, err_j)$
  end for
  $binary_{i,j} = min(err_{i,j})?1 : 0$
  for $j = 1 \rightarrow l$ **do**
    $wei_j = GUID(binary_j, binary_j)$
  end for
  $out_i = \frac{\sum_{j=1}^{l} wei_{i,j} \widehat{c}_{i,j}}{\sum_{j=1}^{l} wei_{i,j}}$
end function

In the third step, the importance of three features (textures, normals and depths) is measured by filtering the initial image using the filtered features as guidance images. At each pixel, the filtered result using the kth feature is inserted into a reference image ref if it has the smallest difference term. Given a predefined candidate set $s = \{0.25, 0.6, 0.8\}$, ref is used to estimate per pixel errors associated with these candidate parameters. We then smooth the error estimates using a two-step strategy. First, the error estimates are filtered using a small guided image filter kernel ($|w| = 3 \times 3$) to obtain 3 binary selection maps. These binary maps are then filtered using a larger guided image filter kernel ($|w| = 7 \times 7$) to return the weight terms $wei_{i,j}$, which are finally used to compute a weighted average for output.

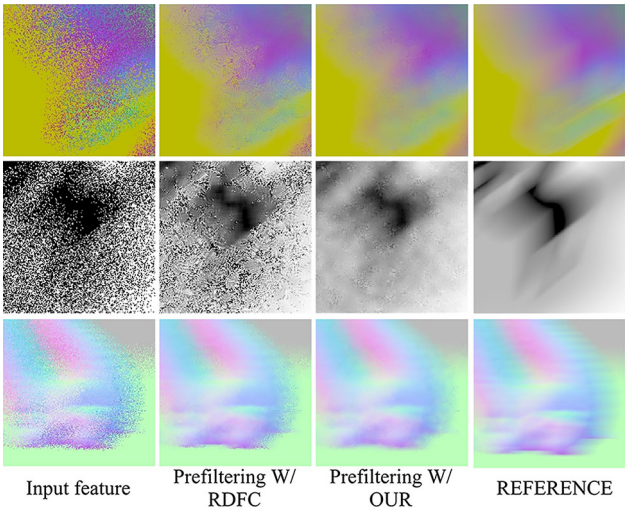


Fig. 7 Prefiltering results of our method and NL-means filter

8 Results and discussion

For all experiments, we used an Intel Core-i7-3630QM with 8 GB RAM and a GeForce GT 650M GPU. The proposed method was integrated as an extension of PBRT [40], where the filters (cross-bilateral filter and guided image filter) and gradient computing process are written in CUDA to enable GPU acceleration.

The results of our method were compared with those of four state-of-the-art methods: SURE-based filtering (SBF) [1], the robust denoising method (RDFC) [2], weighted local regression (WLR) [4], and machine learning (LBF) [5]. These methods were implemented by code provided by the corresponding authors, and all parameters were set to the default values suggested in the original papers. Since SBF is a CPU-based method, it typically needs much longer to filter images. All other methods were implemented in CUDA 7.5. To measure the numerical error, the rMSE [3] was computed as $(img - gt)^2/(gt^2 + \varepsilon)$, where img and gt are the filtered and ground truth pixel values, respectively. $\varepsilon = 0.01$ was set to a small value to prevent division by zero. Furthermore, our method was compared against well-known benchmarks with a wide range of effects, materials, and geometries. All images were produced at a resolution of 800 × 800 pixels.

8.1 Scenes

A major contribution of our method is the prefiltering process to handle noisy features. Figure 7 compares our method with RDFC, which adopts a non-local means filter to denoise features. The first two rows of insets were from a motion blur scene shown in Fig. 8, and both methods used four samples per pixel to produce the initial features. It is obvious that our method produced smoother details in motion-blurred regions. We found that our method outperformed non-local means filter, especially, at low sampling rates. As more samples were distributed (such as eight or 16 samples per pixel), the noise contained in the features was reduced greatly, and both methods returned visually pleasing prefiltering results. Figure 8 further compares the reconstructed images of these two methods. RDFC produced overblurred details in the motion-blurred regions, whereas our method was closer to the reference image.

Typically, the feature prefiltering process is not trivial because additional features for it can not be easily obtained. Hence, past methods typically used well-known image filters, such as non-local means filter [2,33], without utilizing additional features. At a high level, our gradient image can be considered an additional feature for this prefiltering. As a result, the proposed prefiltering process can be integrated into existing frameworks to serve as an alternative. Another prefiltering strategy was proposed by Moon et al. [4], which involved the construction of a reduced feature space using TSVD. This method filters out small singular values that are often due to corruption from noise. However, local dimensions estimated by TSVD can be underestimated at low sampling rates, thereby producing overblurred details.

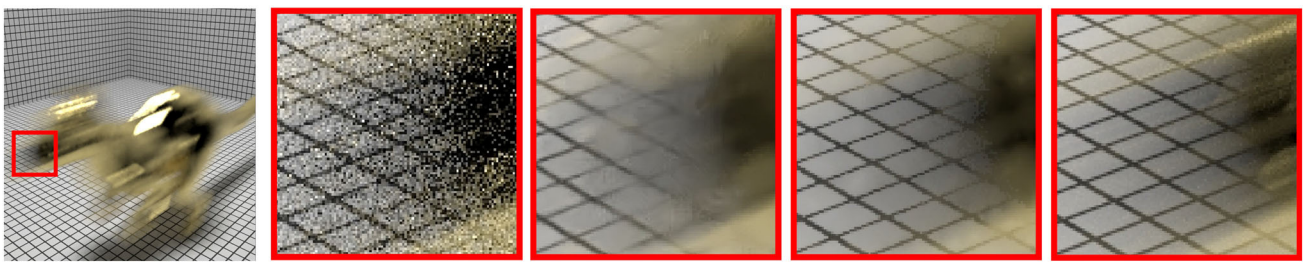


Fig. 8 Comparison of our method with RDFC. Both methods prefilter the noisy features to handle specific distributed effects. RDFC overblurred details in the motion-blurred regions. Our method returned clean details

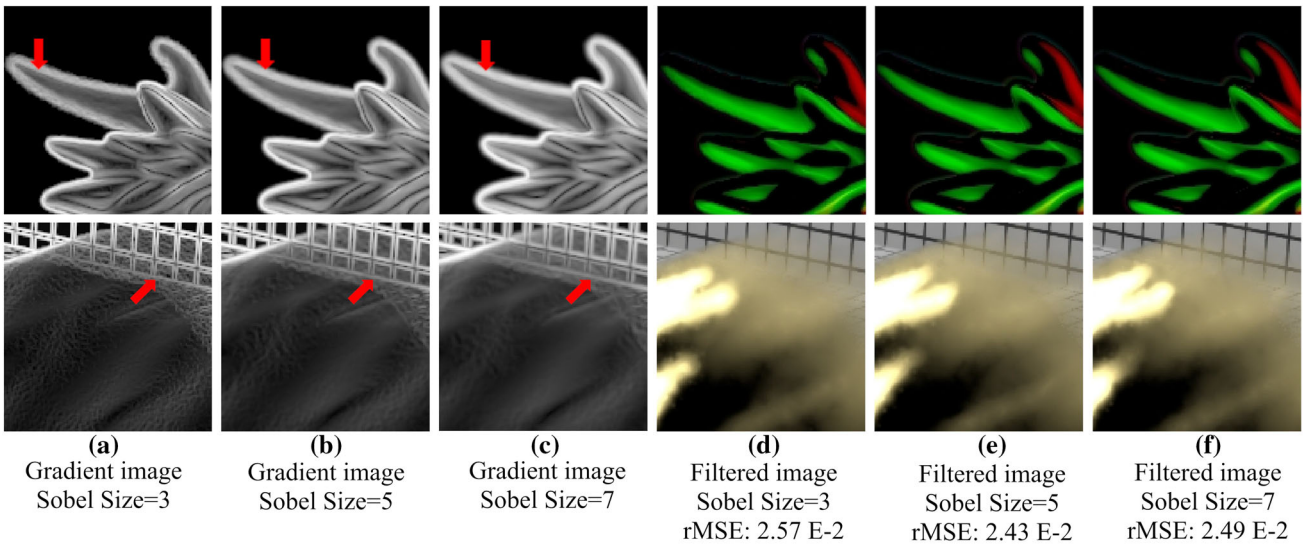


Fig. 9 Prefiltering results with varying Sobel operator sizes. Our gradient image is not heavily dependent on the Sobel operator size

The feature images are prefiltered using a Sobel operator, which has an important influence on the gradient image. There is a bias-variance trade-off for this prefiltering related to the size of the Sobel operator. If a large window (e.g., a 7×7 Sobel operator) is used, the estimated gradients are smoother. On the other hand, small window sizes tend to present more feature details. We compared the prefiltering results with varying Sobel sizes in Fig. 9. In this figure, three window sizes were considered (i.e., 3×3, 5×5, and 7×7). The insets (a)–(c) show the gradient images, and (d)–(f) show the related filtered features as well as the reconstructed images. As expected, the gradient images were smoother as the Sobel operator size increased. However, the changes were very slight such that the filtered results did not vary greatly. This was mainly because the input features typically present relatively clear edges. The gradient image was hence not heavily dependent on Sobel operator size, which became more obvious as the number of samples increased. As a result, we adopted the size of 5×5 for all tested scenes, since it presented a good trade-off between quality and performance.

It is worth noting that gradient information has been widely used to improve image quality. Manzi et al. [19], for instance, sampled finite difference gradients between horizontal and vertical neighboring pixels in addition to pixel values. However, the finite difference images may still contain a lot of noise at low sampling rates. When they solved the Screened Poisson reconstruction, the noise in the finite difference images impact the final results greatly. Our method, on the other hand, considers feature images that are less noisy to save gradient information. Moreover, the above method also employs TSVD to prefilter the initial features, as does WLR. In this case, local dimensions estimated by the TSVD can be underestimated, especially with a small number of samples. It thus tends to produce overblurred details at areas with noisy geometries. A potential solution is using our prefiltered features to construct the regularization constraints, which push each local path in the final image to be similar to the corresponding patches in the filtered features. This is left for our future work.

Figure 10 shows the "SANMINGUEL" scene containing complex geometries. All the images in this figure were rendered using 16 samples per pixel. MC produced substantial noise, while SBF used the features and typically performed better than MC. However, numerous spike noise remained, because SBF cannot recognize spike pixels. Additionally, noticeable artifacts were generated on the walls where the features were not helpful. In this case, SBF failed to select optimal parameters due to its inaccurate error estimates at low sampling rates. WLR did not weight the features appropriately, and, thus, produced significant splotches on the walls. Although both SBF and WLR considered the additional features, they did not return appropriate filter weights, resulting in residual noise. Moreover, WLR tended to overblur images in regions where the variances were exceedingly high. Compared with SBF and WLR, our method suppressed the influence of spike pixels and consistently selected the optimal parameters. As a result, our method removed the noise on both the wall and the window while preserving fine details. Notably, ours was the only method capable of providing clear structural details on the window.

Figure 11 demonstrates two challenging benchmarks. The first one ("DRAGON") contains multiple types of media. Again, SBF presented a lot of noise even when it used feature information. Because it did not optimize the feature parameters, SBF failed to smooth pixels in proximity to noisy

Fig. 10 Comparison of our method with prior methods. MC typically produced abundant noise at low sampling rates. SBF failed to select the optimal parameter due to noisy error estimates. WLR was unable to remove noise along the walls and generated residual noise. Our method outperformed these methods and returned a relatively noise-free result

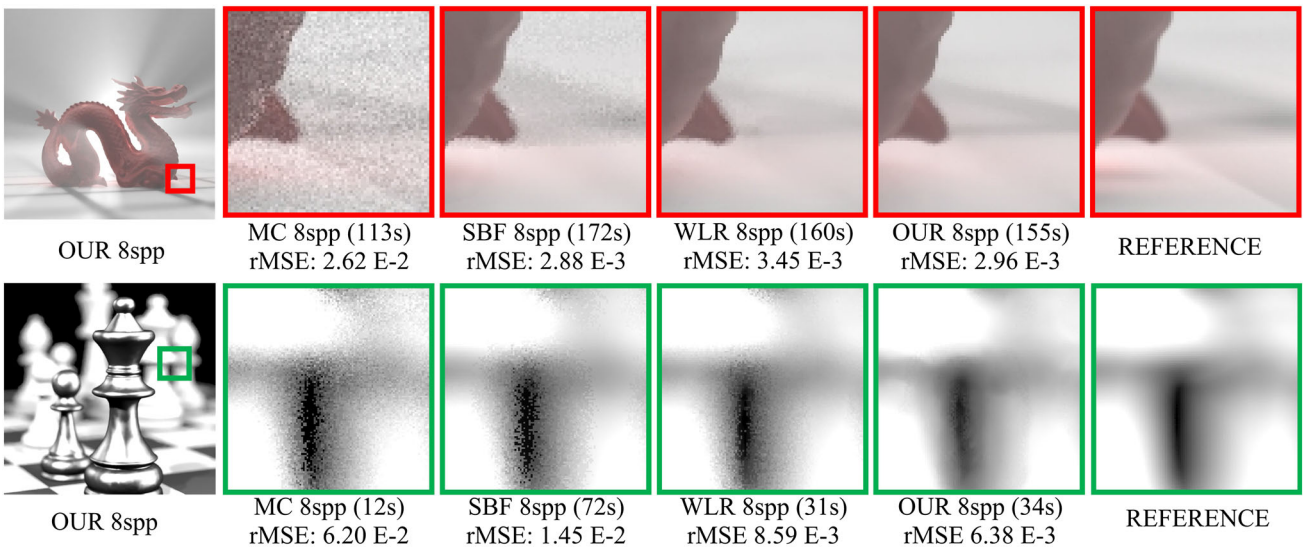


Fig. 11 An additional comparison of our method with prior methods. None of the previous methods could remove noise in areas with strong depth of field. Our method resulted in high-quality renderings closer to the reference image

features. WLR performed better than SBF; however, WLR still overblurred details. Additionally, WLR was not able to properly remove noise in the background areas, especially on the floors. Notably, neither SBF nor WLR removed the noise on the floors without overblurring details or introducing noise artifacts. In this scene, our method maintained the clarity of details on the floors and removed noise without over smoothing details. The second scene("CHESS") has strong depth of field. In this case, none of other methods effectively removed the noise in out-of-focus regions, because the features are typically noisy. However, our method removed the noise in features by combining the use of a guided image filter with gradient information, which generated noise-free features and yielded better details. Furthermore, the results using our method were closer to the ground truth.

MC renderings may exhibit high dynamic range, and pixels with radiance much larger than those of their neighbors are defined as spike pixels (i.e., outliers). Previous methods typically exhibited difficulty in removing spike noise, because these pixels cannot be spread robustly. For example, SBF did not distribute the energy of the outliers and resulted in a significant amount of spike noise (Fig. 12). We can mitigate this problem by using a larger filter size; however, this is prone to the introduction of bias and the overblurring of images. Here, we recognized these pixels by using gradient information that is free of spike noise, and removed the spike noise using a two-radius strategy. In Fig. 12, we compare the results of spike removal with those of WLR and LBF. WLR suppressed the spike noise by dividing the computed weights of samples by the variance of the sample mean; however, it

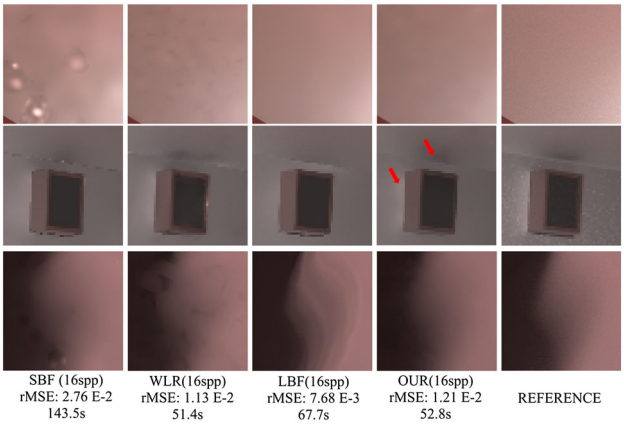


Fig. 12 Comparison of spike noise removal. Results indicate that our method removed a significant amount of spike noise and produced smooth details

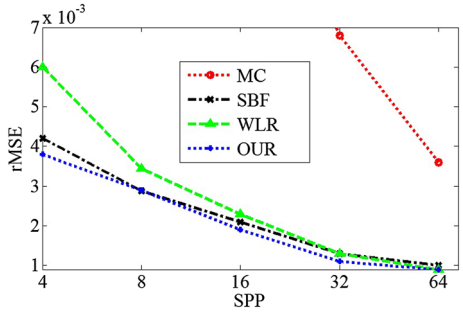


Fig. 13 Convergence plot for our method and previous methods

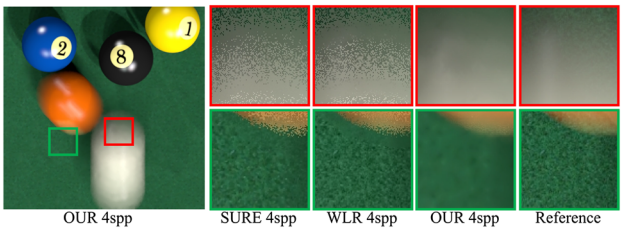


Fig. 14 Failure case of our method. Our method removes significant noise in the motion-blurred region. However, in the motionless region, our method fails to select optimal parameters at low sampling rates and overblurs image details

still exhibited low-frequency noise and splotches on the table (first row of the insets). LBF replaced the colors of spike pixels with the median block color. In this scene, we found that both our method and LBF removed spike noise without producing splotches. However, LBF tended to be aggressive in its filtering, resulting in a loss of some local details (second and third rows of the insets). Our two-radius strategy, on the other hand, down-weighted the spike pixels to robustly perform the reconstruction step. Unfortunately, our method introduced energy loss in the final image similar to prior methods, leading to a relatively higher numerical error. To solve this problem, a better way of restoring and redistributing the lost energy is left for our future research (Moon et al. [37]).

The rMSEs of rendered images (Figs. 8, 10 and 11) show that our method consistently generated lower rMSEs as compared with SBF and WLR at the same sampling rates. SBF selected the spatial parameter using SURE, which ignores the influence of feature parameters. Moreover, SBF produced inaccurate error estimates at low sampling rates, resulting in large numerical errors. In Fig. 10, the rMSE reduction associated with SBF was poor, despite its producing relatively smooth details. Specifically, this method was unable to remove spike noise. Compared with SBF, WLR performed better due to its use of error analysis in the reduced feature space. In practice, we found that our method produced similar rMSE as those generated with WLR, which is likely due to both methods using parametric curve fitting. However, WLR underestimates local dimensions according to the TSVD procedure, leading to overblurred details. Additionally, its shared bandwidth may be affected by noisy features, thereby producing suboptimal results. Our method, however, computes spatial and feature parameters using two separate steps, thereby eliminating the influence of noise on the overall result. We also provide a convergence plot for the "DRAGON" scene in Fig. 13, which shows that our method yields lower numerical errors.

8.2 Limitations

A main limitation of our method is that the gradient information may be not efficient at recognizing fine structures. As a result, features can be over blurred, especially at regions where the Sobel operator fails to recognize fine details. As shown in Fig. 14, results using our method were visually better in regions containing strong motion blur, but overblurred texture details on the ground. The impact of this problem may be reduced by using a small guided image filter, but this can result in failure to denoise features. Furthermore, we computed the optimal feature parameters as a weighted average of the predefined candidates. The error analysis was built on a perceptual metric that might have been inaccurate. Finally, our spike removal method introduced energy loss to the reconstructed images. To solve this problem, we plan to restore and redistribute the lost energy in future work.

9 Conclusions and future work

A novel feature-based adaptive rendering method was proposed in this study to efficiently handle a wide variety of rendering effects. Our method first recognizes geometric structures by employing the Sobel operator on features to

return a gradient image, which maintains clarity along the edges. The input features are then prefiltered using a guided image filter, with the gradient image used as the guidance image. At a high level, the proposed prefiltering method can be used as a plus for existing approaches. Our method computes the spatial parameter through parametric curve fitting on a per pixel basis, which enables us to eliminate the influence of noisy features. We then predefine a set of candidates as feature parameters and compute their weighted average for output using a two-step smoothing strategy. Experimental results demonstrated the robustness and efficiency of our method on a set of challenging benchmarks, and showed that it improves both visual image quality and numerical error.

Our future work will extend the guided image filter to enable its combination with other novel features (e.g., caustics [2], and virtual flash images [32]). Additionally, we are also interested in using gradient content to handle outliers. Similar to other image-space methods, ours does not leverage the high-dimensional information for noise reduction. Therefore, our framework could be potentially applied to a high-dimensional space to improve image quality. Finally, we also plan to explore the implementation of our method in wave rendering [41].

References

1. Li, T.M., Wu, Y.T., Chuang, Y.Y.: SURE-based optimization for adaptive sampling and reconstruction. ACM Trans. Graph. **31**(6), 194:1–194:9 (2012)
2. Rousselle, F., Manzi, M., Zwicker, M.: Robust denoising using feature and color information. Comput. Graph. Forum **32**(7), 121–130 (2013)
3. Rousselle, F., Knaus, C., Zwicker, M.: Adaptive sampling and reconstruction using greedy error minimization. ACM Trans. Graph. **30**(6), 6:1–6:11 (2011)
4. Moon, B., Carr, N., Yoon, S.: Adaptive rendering based on weighted local regression. ACM Trans. Graph. **33**(5), 170:1–170:14 (2014)
5. Kalantari, N.K., Bako, S., Sen, P.: A machine learning approach for filtering Monte Carlo noise. ACM Trans. Graph. **34**(4), 122:1–122:12 (2015)
6. Bauszat, P., Eisemann, M., Eisemann, E.: General and robust error estimation and reconstruction for Monte Carlo rendering. Comput. Graph. Forum **34**(2), 597–608 (2015)
7. He, K., Sun, J., Tang, X.: Guided Image Filtering. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(6), 1397–1409 (2010)
8. Kajiya, J.T.: The rendering equation. In: ACM SIGGRAPH, pp. 143–150 (1986)
9. Hachisuka, T., Jarosz, W., Weistroffer, R.P., Dale, K.: Multidimensional adaptive sampling and reconstruction for ray tracing. ACM Trans. Graph. **27**(3), 33:1–33:10 (2008)
10. Durand, F., Holzschuch, N., Soler, C., Chan, E., Sillion, F.X.: A frequency analysis of light transport. ACM Trans. Graph. **24**(3), 1115–1126 (2005)
11. Soler, C., Subr, K., Durand, F., Holzschuch, N., Sillion, F.: Fourier depth of field. ACM Trans. Graph. **28**(2), 18:1–18:12 (2009)
12. Egan, K., Tseng, Y.T., Durand, F., Holzschuch, N.: Frequency analysis and sheared reconstruction for rendering motion blur. ACM Trans Graph **28**(3), 93:1–93:13 (2009)
13. Egan, K., Hecht, F., Durand, F., Ramamoorthi, R.: Frequency analysis and sheared filtering for shadow light fields of complex occluders. ACM Trans. Graph. **30**(2), 9:1–9:13 (2011)
14. Egan, K., Durand, F., Ramamoorthi, R.: Practical filtering for efficient ray-traced directional occlusion. ACM Trans. Graph. **30**(6), 180:1–180:10 (2011)
15. Belcour, l, Soler, C., Subr, K., Holzschuch, N., Durand, F.: 5D covariance tracing for efficient defocus and motion blur. ACM Trans. Graph. **32**(3), 31:1–31:18 (2011)
16. Lehtinen, J., Aila, T., Chen, J., Laine, S., Durand, F.: Temporal light field reconstruction for rendering distribution effects. ACM Trans. Graph. **31**(4), 55:1–55:10 (2012)
17. Lehtinen, J., Aila, T., Laine, S., Durand, F.: Reconstructing the indirect light field for global illumination. ACM Trans Graph **30**(4), 51:1–51:12 (2011)
18. Kettunen, M., Manzi, M., Aittala, M., Lehtinen, J.: Gradient-domain path tracing. ACM Trans. Graph. **34**(4), 123:1–123:13 (2015)
19. Manzi, M., Vicini, D., Zwicker, M.: Regularizing image reconstruction for gradient-domain rendering with feature patches. Comput. Graph. Forum **35**(2), 263–273 (2016)
20. Sen, P., Darabi, S.: Compressed rendering: a rendering application of compressed sensing. IEEE Trans. Vis. Comput. Graph. **17**(4), 487–499 (2011)
21. Liu, X.D., WU, J.Z., Zheng, C.W.: KD-tree based parallel adaptive rendering. The Visual Computer **28**(6–8), 613–623 (2012)
22. Mitchell, D.P.: Generating antialiased images at low sampling densities. In: Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, vol. 21, pp. 65-72. ACM, Anaheim (1987)
23. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In Proceedings of the International Conference on Computer Vision, pp. 839–846 (1998)
24. Buades, A., Coll, B., Morel, J.M: A non-local algorithm for image denoising. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 60-65 (2005)
25. Rousselle, F., Knaus, C., Zwicker, M.: Adaptive rendering with non-local means filtering. ACM Trans. Graph. **31**(6), 195:1–195:9 (2012)
26. Overbeck, R.S., Donner, C., Ramamoorthi, R.: Adaptive wavelet rendering. ACM Trans. Graph. **28**(5), 140:1–140:12 (2009)
27. Kalantari, N.K., Sen, P.: Removing the noise in Monte Carlo rendering with general image denoising algorithm. Comput. Graph. Forum **32**(2), 93–102 (2013)
28. Sen, P., Darabi, S.: On Filtering the Noise from the Random parameters in Monte Carlo Rendering. ACM Trans Graph **31**(3), 18:1–18:14 (2012)
29. Bauszat, P., Eisemann, M., John, S.: Sample-based manifold filtering for interactive global illumination and depth of field. Comput. Graph. Forum **34**(1), 265–276 (2015)
30. Liu, X.D., Zheng, C.W.: Adaptive cluster rendering via regression analysis. Vis. Comput. **31**(1), 105–114 (2015)
31. Liu, X.D., Zheng, C.W.: Parallel adaptive sampling and reconstruction using multi-scale and directional analysis. The Visual Computer **29**(6–8), 501–511 (2013)
32. Moon, B., Jun, J.Y., Lee, J.: Robust image denoising using a virtual flash image for Monte Carlo ray tracing. Comput. Graph. Forum **32**(1), 139–151 (2013)
33. Bitterli, B., Rousselle, F., Moon, B.: Nonlinearly weighted first-order regression for denoising Monte Carlo renderings. Comput. Graph. Forum **35**(4), 107–117 (2016)

34. Bauszat, P., Eisemann, M., Magnor, M.: Guided image filtering for interactive high-quality global illumination. Comput. Graph. Forum **30**(4), 1361–1368 (2011)
35. Delbracio, M., Muse, P., Buades, A., Chauvier, J.: Boosting Monte Carlo Rendering by ray histogram fusion. ACM Trans Graph **33**(1), 8:1–8:15 (2014)
36. Moon, B., Guitian, J.A., Yoon, S., Mitchell, K.: Adaptive rendering with linear predictions. ACM Trans. Graph. **34**(4), 121:1–121:11 (2015)
37. Moon, B., McDonagh, S., Mitchell, K., Gross, M.: Adaptive polynomial rendering. ACM Trans Graph **35**(4), 40:1–40:11 (2016)
38. Zwicker, M., Jarosz, W., Lehtinen, J., Moon, B.: Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. Comput. Graph. Forum **34**(2), 667–681 (2015)
39. Ruppert, D., Wand, M.: Multivariate locally weighted least squares regression. The annals of statistics **22**(3), 1346–1370 (1994)
40. Pharr, M., Humphreys, G.: Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann Publishers Inc., San Fransisco (2010)
41. Wu, F.K., Zheng, C.W.: Microfacet-based interference simulation for multilayer films. Graphical Models **78**(6), 26–35 (2015)