Vis Comput (2018) 34:551–562
https://doi.org/10.1007/s00371-017-1360-2

ORIGINAL ARTICLE

Adaptive rendering based on robust principal component analysis

Hongliang Yuan[1,2] · Changwen Zheng[1]

Published online: 13 March 2017

Abstract We propose an adaptive sampling and reconstruction method based on the robust principal component analysis (PCA) to denoise Monte Carlo renderings. Addressing spike noise is a challenging problem in adaptive rendering methods. We adopt the robust PCA as a pre-processing step to efficiently decompose spike noise from rendered image after the image space is sampled. Then we leverage patch-based propagation filter for feature prefiltering and apply the robust PCA to reduce dimensionality in high-dimensional feature space. After that, we estimate a per-pixel pilot bandwidth derived from kernel density estimation and construct the multivariate local linear estimator in the reduced feature space to estimate the value of each pixel. Finally, we distribute additional ray samples in the regions with higher estimated mean squared error if sampling budget remains. We demonstrate that our method makes significant improvement in terms of both numerical error and visual quality compared to the state-of-the-art.

Keywords Adaptive rendering · Robust principal component analysis · Propagation filter · Monte Carlo ray tracing · Mean squared error

✉ Hongliang Yuan
11488336@qq.com

Changwen Zheng
changwen@iscas.ac.cn

1 Science and Technology on Integrated Information System Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China

2 University of Chinese Academy of Sciences, Beijing, China

1 Introduction

Monte Carlo (MC) ray tracing [11] plays an important role in synthesizing photo-realistic images, but produces noise artifacts when a small quantity of ray samples (e.g., less than one hundred or thousand, depend on the complexity of scenes) are assigned for each pixel in the image space. DeCoro et al. [6] divide noise artifacts into two categories. One is subtle but globally distributed high-frequency noise which is generally caused by the variance of correlated estimators. The other is the highly-localized, distinctive spike noise, i.e., outlier, which is generally caused by the low-probability yet high-energy light paths.

Adaptive rendering, including adaptive sampling and reconstruction, can mitigate the first category of noise artifacts effectively. Adaptive sampling locally adjusts sampling densities in the image plane according to the variance or mean squared error (MSE) computed from previous allocated samples. The key of adaptive sampling is to develop a robust error metric which can guide where need to allocate more ray samples, e.g., Stein's unbiased risk estimator (SURE) [31], contrast metric [20], median absolute deviation (MAD) [8], perceptual based error metric [9], f-divergences [26], MSE which can be decomposed into variance plus the square of bias, etc.

Adaptive reconstruction adopts locally defined reconstruction filters which are determined by the bandwidth to estimate the value of each pixel. Therefore, the key step to make adaptive reconstruction successfully is to estimate locally optimal bandwidth. Moon et al. [21] define a set of bandwidths on which optimal shared bandwidth is estimated through minimizing MSE. Although their method can produce high-quality images, it suffers from heavy computational overhead of estimating the optimal bandwidth for each pixel. Instead of estimating the optimal bandwidth on a set of predetermined bandwidths, we present a pilot bandwidth derived from kernel density estimation (KDE) which can reduce reconstruction time and meanwhile improve image quality significantly through removing spike noise and feature prefiltering.

Addressing the second category of noise artifacts without noticeable energy loss is a challenging problem in adaptive rendering methods. DeCoro et al. [6] propose a density-based outlier rejection method based on space-partitioning tree. It is not trivial to integrate their method into current renderer because of the structure they use. Moon et al. [21] down-weight these pixels, and Kalantari et al. [12] reject these pixels. However, their methods still remain splotch artifacts at low sampling rate, see the comparison of the “conference” scene in Fig. 11.

In order to handle the two categories of noise artifacts, we present an adaptive sampling and reconstruction method based on the robust principal component analysis (PCA). Given the rendered noisy image, the robust PCA can decompose spike noise efficiently. After that, we use local linear estimator to model the relationship between explanatory variables (G-buffers including world coordinate, shading normal, texture value for the first intersection and direct light source visibility) and a response variable (color values including three channels). Since the visibility can be very noisy due to depth-of-field effect, we leverage the patch-based propagation filter [3] for feature prefiltering. Then we use the robust PCA to reduce dimensionality in the high-dimensional feature space. Finally, we adopt the local linear estimator to estimate the value of each pixel. The key of the local linear estimator is to estimate bandwidth. We propose a pilot bandwidth derived from KDE which is the main distinction between our method and the previous one, such as the approach proposed by Moon et al. [21]. We demonstrate that our approach leads to significant improvements both in subjective and quantitative errors compared to the state-of-the-art. In summary, we make the following technical contributions:

– We present a novel spike noise removal technique based on the robust PCA as a pre-processing step to decompose spike noise from noisy image.
– We extend pixel-based propagation filter to patch-based for feature prefiltering.
– Given a matrix constructed from a local feature space, we propose to use the robust PCA to decompose it into a low-rank and sparse matrix. The low-rank matrix identifies a reduced, local feature space which guides our reconstruction based on the local linear estimator.
– We estimate a pilot bandwidth for each pixel in a reduced, local feature space.

2 Previous work

Adaptive sampling and reconstruction has been long history [11]. Early rendering methods have succeeded in reducing sampling rates. Most recent work concentrates more on reducing the number of ray samples significantly without sacrificing image quality. We will review most recent techniques on adaptive sampling and reconstruction from the following three aspects.

Image-space methods only utilizing color Some types of these methods leverage multi-scale filters. Overbeck et al. [24] perform Daubechies wavelet analysis on the generated image and guide adaptive sampling by contrast metric, and reconstruct image by wavelet shrinkage [8]. Rousselle et al. [27] generate high-quality image using Gaussian filters at several scales. Liu et al. [17] proposed a parallel image-space adaptive rendering approach based on the multi-scale and directional analysis. Other methods leverage denoising filters. Rousselle et al. [28] exploit non-local means filter for adaptive rendering, while Delbracio et al. [7] proposed a multi-scale non-local means filter for adaptive reconstruction. Kalantari et al. [13] proposed a noise estimation metric based on MAD for adaptive sampling and designed a general technique using any denoising filters, such as BM3D [5]. Liu et al. [18] partition the rendering space into clusters based on feature vector which contains gradient, variance and position. In addition, they model each cluster by smooth polynomial regression. Although they utilize polynomial regression for reconstruction, they don’t make use of auxiliary features.

Image-space methods combining color and G-buffers G-buffers (i.e., normal, depth and texture), which are easy to be acquired from MC renderer, can help design adaptive reconstruction filters that preserve image structures. Sen and Darabi [30] presented a method based on cross-bilateral filter utilizing the information theoretic approach to deal with noisy features. While their method produces high-quality image with a small number of samples per pixel, the complexity of their algorithm is proportional to the number of ray samples. Li et al. [15] first introduce SURE to guide adaptive sampling and reconstruction. Rousselle et al. [29] also build on SURE and cross-bilateral filter. In order to reduce computational overhead of the method proposed by Moon et al. [21], most recently, Moon et al. [22] proposed to approximate the value of each pixel in prediction window with multiple, but sparse local linear estimator. Kalantari et al. [12] presented a machine learning approach to reduce MC noise. Since they only focus on image reconstruction, they cannot benefit from the adaptive sampling. Moon et al. [23] explore higher-order regression to filter MC noise. Bitterli et al. [2] construct first-order regression function to estimate the values of each pixel and leverages the non-local means filter to shape the regression kernel.

Multi-dimensional space methods Hachisuka et al. [10] proposed an adaptive sampling and anisotropic reconstruction method in multi-dimensional space. Although it can efficiently generate high-quality images in a low-dimensional space, it costs a lot of memory and computation time as the dimension increases. Liu et al. [19] proposed a parallel multi-dimensional adaptive sampling method to reduce the computation time and control the memory cost. Belcour et al. [1] proposed a frequency analysis for accelerating computation for the rendering of effects such as motion blur and depth-of-field which requires costly 5D integrals. Yan et al. [34] presented fast 4D sheared filtering method on the GPU which improve sheared filtering overhead significantly. While multi-dimensional adaptive sampling and reconstruction methods have shown their efficiency in generating images with a handful of samples per pixel, they are developed to optimize specific MC effects. In contrast, image-space methods can handle various MC effects, such as soft shadows, motion blur, and depth-of-field.

Zwicker et al. [35] surveyed recent advances in adaptive sampling and reconstruction algorithms and divided them into two categories: “a priori” methods and “a posteriori” methods. Our method belongs to the latter .

3 Background and overview

In the following, we briefly review the local linear estimator and then give an overview of our proposed algorithm.

3.1 Local linear estimator

The nonparametric regression model with multiple covariates is as follows

$$y = m(\mathbf{x}) + \epsilon \tag{1}$$

From the perspective of rendering, y and $m(\mathbf{x})$ represent a MC input image and a ground truth image, respectively. $\mathbf{x} = (x_1, \dots, x_d)^T$ is a d-dimensional auxiliary feature vector, and ϵ represents MC noise.

Estimating the local linear estimator on full dimensional feature space requires a high cost of computation. To address this issue, we adopt the truncated singular value decomposition proposed by Moon et al. [21]. Given a projection matrix $\mathbf{V}_k \in \mathbb{R}^{d\times k}$ (Sect. 4.3) and an auxiliary feature vector $\mathbf{x}_c \in \mathbb{R}^d$, where k is less than or equal to d, we project it onto a low-dimensional subspace by multiplying $\mathbf{V}_k^T$, i.e., $\mathbf{z}_c = \mathbf{V}_k^T \mathbf{x}_c \in \mathbb{R}^k$.

In the low-dimensional subspace, the local linear estimator of the conditional mean function $m(\cdot)$ is $\hat{\alpha}$ and the solution for α is to solve the following locally kernel weighted least squares problem:

$$\min_{\alpha,\boldsymbol{\beta}} \sum_{i=1}^{n} \left\{ y_i - \alpha - \boldsymbol{\beta}^T(\mathbf{z}_i - \mathbf{z}_c) \right\} \boldsymbol{K}_{\mathbf{H}}(\mathbf{z}_i - \mathbf{z}_c) \tag{2}$$

where n represents the number of pixels within a filtering window centered on pixel c, $\boldsymbol{\beta} = (\beta_1, \dots, \beta_k)^T$ is coefficient for each explanatory variable. $\boldsymbol{K}(\cdot)$ is a symmetric, one-dimensional kernel function such that $\int \boldsymbol{K}(u) = 1$ and

$$\boldsymbol{K}_{\mathbf{H}}(\mathbf{z}_i - \mathbf{z}_c) = \prod_{j=1}^{k} \frac{1}{b_j h} \boldsymbol{K}\left(\frac{z_{ij} - z_{cj}}{b_j h}\right) \tag{3}$$

h is shared bandwidth and $b_1, \dots, b_k$ are tuning parameters. The bandwidth matrix $\mathbf{H}^{1/2} = \mathrm{diag}\{hb_1, \dots, hk_d\}$ is diagonal. The model, which separates bandwidth matrix into two components, has been studied deeply in the statistical field. Such as, Cheng et al. [4] utilize this model to study the variance reducing techniques. Moon et al. [21] first apply this model to image-space adaptive rendering. They choose b_j to be $\left|\frac{\partial^2 m(\mathbf{z})}{\partial z_j^2}\right|^{-0.5}$. We follow the previous work and replace the shared bandwidth with our presented pilot bandwidth described in Sect. 4.4.

From the standard weighted least squares theory, the local linear estimator on center pixel c is given by

$$\begin{aligned}\hat{\alpha} &= \hat{m}(\mathbf{z}_c)\\ &= \mathbf{e}^T(\mathbf{X}_\mathbf{z}^T\mathbf{W}_\mathbf{z}\mathbf{X}_\mathbf{z})^{-1}\mathbf{X}_\mathbf{z}^T\mathbf{W}_\mathbf{z}\mathbf{Y}\\ &= \mathbf{L}^\mathbf{T}(\mathbf{z_c})\mathbf{Y}\\ &= \sum_{i=1}^{n} l_i(\mathbf{z}_c) y_i\end{aligned} \tag{4}$$

where $\mathbf{e} = (1, 0, \dots, 0)^T$ is a (k + 1)-dimensional vector, $\mathbf{Y} = (y_1, \dots, y_n)^T$, $\mathbf{W}_\mathbf{z} = diag\{\boldsymbol{K}_\mathbf{H}(\mathbf{z}_1 - \mathbf{z}_c), \dots, \boldsymbol{K}_\mathbf{H}(\mathbf{z}_n - \mathbf{z}_c)\}$, the ith row of $\mathbf{X}_\mathbf{z}$ is $(1, (\mathbf{z}_i - \mathbf{z}_c)^T)$ and $\mathbf{L}^T(\mathbf{z}_c)$ is set as the first row of $(\mathbf{X}_\mathbf{z}^T\mathbf{W}_\mathbf{z}\mathbf{X}_\mathbf{z})^{-1}\mathbf{X}_\mathbf{z}^T\mathbf{W}_\mathbf{z}$. We can clearly see that the local linear estimator $\hat{\alpha}$ is a weighted average value of the response variables in the neighborhood at the center pixel c. We call $\mathbf{L}(\mathbf{z}_c)$ equivalent kernel.

Then the bias of $\hat{m}(\mathbf{z}_c)$ can be computed as follows:

$$\begin{aligned}E\left(\hat{m}(\mathbf{z}_c) - m(\mathbf{z}_c)\right) &= \sum_{i=1}^{n} l_i(\mathbf{z}_c)E(y_i) - m(\mathbf{z}_c)\\ &= \sum_{i=1}^{n} l_i(\mathbf{z}_c)m(\mathbf{z}_i) - m(\mathbf{z}_c)\\ &= \sum_{i=1}^{n} l_i(\mathbf{z}_c)y_i - y_c\\ &= \hat{\alpha} - y_c\end{aligned} \tag{5}$$

The variance of $\hat{m}(\mathbf{z})$ can be estimated in a similar manner:

$$\mathrm{var}(\hat{m}(\mathbf{z}_c)) = \sum_{i=1}^{n} (l_i(\mathbf{z}_c))^2 \mathrm{var}(y_i) \tag{6}$$

where $\mathrm{var}(y_i)$ is the variance of the sample mean at pixel i. Then the MSE is estimated as follows:

$$\widehat{\mathrm{MSE}}(\hat{m}(\mathbf{z}_c)) = (\hat{\alpha} - y_c)^2 + \mathrm{var}(\hat{m}(\mathbf{z}_c)) \tag{7}$$

we will use the MSE to guide adaptive sampling in Sect. 5.

The equivalent kernel only depends on the position of center pixel c and its neighbors, while is independent of the response variables. So given center pixel c and its neighbors, we only compute the equivalent kernel once and apply it to each channel of color image independently.

3.2 Overview

Addressing spike noise in noisy image is challenging. Kalantari et al. [12] presented an approach for identifying spike noise in the filtered result. In contrast to this approach, our outlier removal method based on the robust PCA can detect sparse spike noise in noisy image so that removing outlier will not cause noticeable energy loss in the reconstruction result without redistributing the lost energy [23], see Fig. 1.

Estimating bandwidth for each feature plays a key role in synthesizing high-quality image. Moon et al. [21] define a set of shared bandwidth (e.g., 0.2, 0.4, 0.6, 0.8 and 1.0), and estimate MSE for each shared bandwidth, which suffers from a high computational overhead. Replacing shared bandwidth with our presented pilot bandwidth will decrease reconstruction time, but not reduce image quality obviously, see Fig. 2. This motivates us to further improve image quality through removing outlier, adding visibility feature and feature prefiltering.

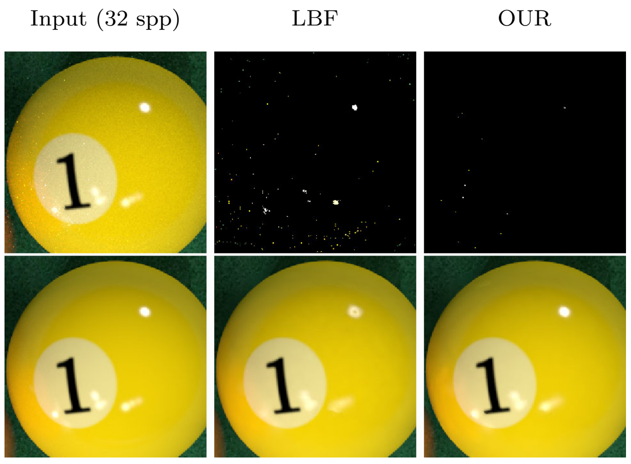


Fig. 1 Spikes detected by learning-based filter (LBF) [12] and our outlier removal method (OUR) on the "pool" scene. LBF cannot distinguish high-frequency signals from noisy image and regards them as spike noise. If we use LBF's spike removal method with our reconstruction approach, we cannot preserve high-frequency signals (*bottom middle*). The *bottom left image* is the reference

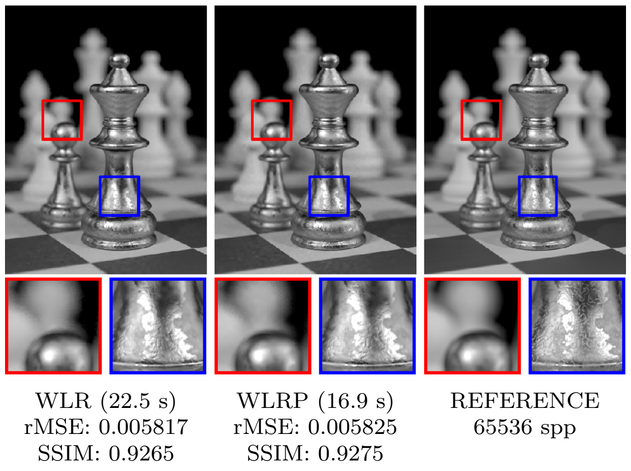


Fig. 2 Results of WLR and WLRP on the "chess" scene. The weighted local regression (WLR) method is proposed by Moon et al. [21]. WLRP method is modified based on WLR replacing shared bandwidth with our presented pilot bandwidth in Sect. 4.4

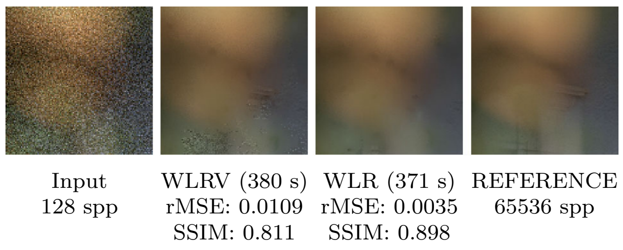


Fig. 3 The WLR results for the "sanmiguel" scene simulating strong depth-of-field effects, with and without the visibility feature. The WLRV method is modified based on WLR through adding visibility feature

Direct light source visibility feature is very noisy due to depth-of-field. If we extend WLR method adding visibility feature straightforwardly, the noise of visibility feature will propagate directly to the filtered result, see Fig. 3. In order to mitigate the issue, we propose to use the patch-based propagation filter to denoise auxiliary features.

Based on these observations, we propose a novel rendering method based on the robust PCA which can decompose spike noise from noisy image as described in Sect. 4.1. We filter the features leveraging propagation filter in Sect. 4.2. Then we use the robust PCA to identify a reduced feature space, as presented in Sect. 4.3. In Sect. 4.4, we estimate a per-pixel pilot bandwidth derived from KDE and use the local linear estimator to compute the value of each pixel. We introduce adaptive sampling in Sect. 5, and in Sect. 6 we give the implementation details of our method. We demonstrate comparison results with other methods in Sect. 7 and give

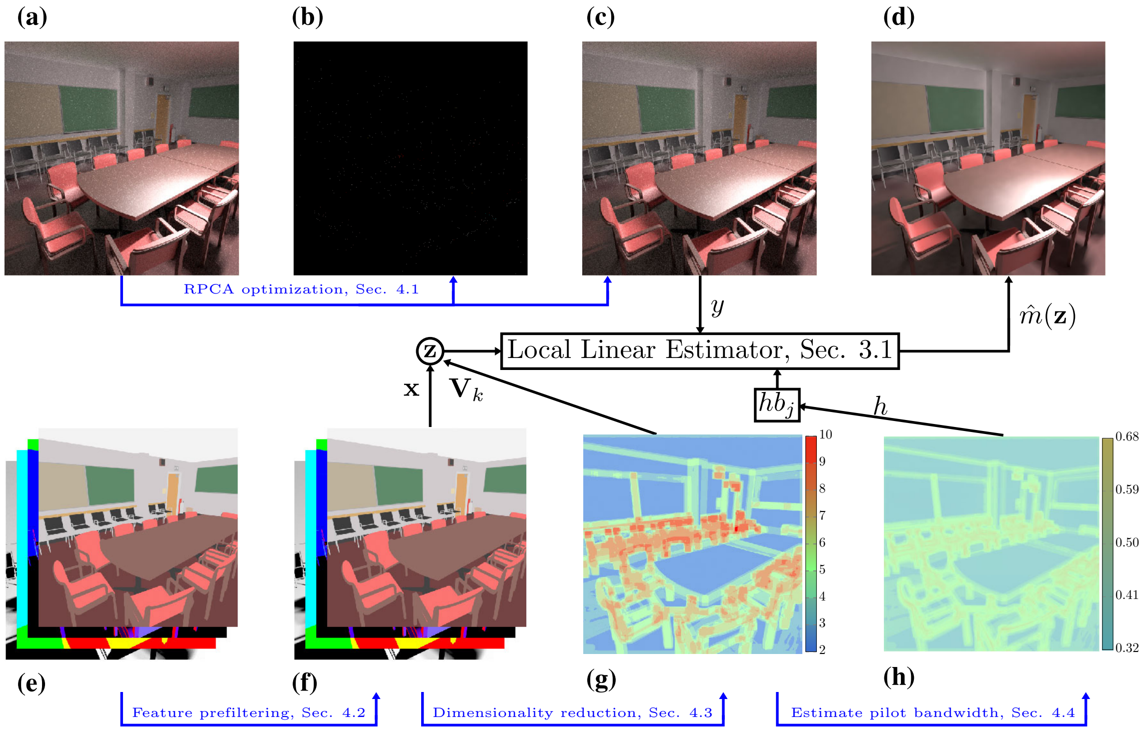

(a)
(b)
(c)
(d)
RPCA optimization, Sec. 4.1
y
$\hat{m}(\mathbf{z})$
z
Local Linear Estimator, Sec. 3.1
x
$\mathbf{V}_k$
hb_j
h
(e)
(f)
(g)
(h)
Feature prefiltering, Sec. 4.2
Dimensionality reduction, Sec. 4.3
Estimate pilot bandwidth, Sec. 4.4

Fig. 4 Given an input image (**a**), we use the robust PCA to decompose spike noise (**b**) and generate image (**c**) with spike noise removed. We filter the input features (**e**) leveraging patch-based propagation filter. Then we compute reduced feature space (**g**) from filtered features (**f**). After that, we compute a per-pixel pilot bandwidth (**h**) derived from KDE. Finally, we reconstruct image (**d**) by the local linear estimator. The meaning of *symbols* in the figure can be found in Sect. 3.1

conclusions in Sect. 8. Figure 4 illustrates the main steps of our approach.

4 Proposed method

4.1 Decompose spike noise

Given a color image, we denote color value at a pixel p as $d(p) = (d_1(p), d_2(p), d_3(p))$. Then we consider each channel of a color image as a data matrix $\mathbf{D}_i = (d_i(p)) \in \mathbb{R}^{m\times n}$, where $i = 1, 2, 3$, m and n represents the height and width of a color image. $\mathbf{D}_i$ may be decomposed as

$$\mathbf{D}_i = \mathbf{A}_i + \mathbf{E}_i \tag{8}$$

where $\mathbf{A}_i$ has low-rank and $\mathbf{E}_i$ is sparse. Some entries of the additive errors $\mathbf{E}_i$ can have arbitrarily large magnitude. The sparse errors $\mathbf{E}_i$ can be decomposed by solving the following convex optimization problem:

$$\min_{A_i, E_i} \|\mathbf{A}_i\|_* + \lambda\|\mathbf{E}_i\|_1, \quad \text{subject to} \quad \mathbf{D}_i = \mathbf{A}_i + \mathbf{E}_i \tag{9}$$

where $\|\cdot\|_*$ denotes the sum of its singular values, $\|\cdot\|_1$ denotes the sum of the absolute value of matrix entries, and λ is a positive tune parameter. This optimization is referred to as the robust PCA (RPCA).

Lin et al. [16] presented a fast algorithm for solving the RPCA problem (Eq. 9) which utilizes techniques of augmented Lagrange multipliers (ALM). See their paper for the detail information. We implement the ALM algorithm in a function

ALM(**D**, **A**, **E**, *iterations*)

that takes a corrupted matrix **D** as input and returns a low-rank matrix **A** and an error matrix **E**. The ALM is an iterative algorithm, so we need specify the number of iterations in the parameter *iterations*. Considering computational overhead, setting *iterations* to 1 is enough to decompose spike noise from input image.

In order to reduce computational overhead of ALM function, we split image into tiles and call the ALM function for each tile. The value in **E** is just the candidate of spike noise, and we need select a threshold value. If any values in **E** are greater than the threshold, we mark corresponding pixel as spike. In our experiments, we set threshold to 2.5. We summarize our pseudocode in Algorithm 1.

In Fig. 4, we show that our algorithm can efficiently identify spike noise from image that is grossly corrupted.

Algorithm 1: Remove Spike Noise

```
Input: Input image $I$, tile size $ts = 64$
Result: Output image with spike noise removed
1  Split image into tiles based on tile size $ts$;
2  foreach tile do
3      Compute extent for the tile;
4      foreach channel $i = 1, 2, 3$ do
5          Construct $\mathbf{D}_i$ from this tile;
6          $ALM(\mathbf{D}_i, \mathbf{A}_i, \mathbf{E}_i, 1)$;
7      end
8  end
9  $threshold = 2.5$;
10 foreach pixel $p$ in $I$ do
11     if $(\boldsymbol{E}_1[p] > threshold)$ Or $(\boldsymbol{E}_2[p] > threshold)$ Or
       $(\boldsymbol{E}_3[p] > threshold)$ then
12         Replace $d(p)$ with the median block color in all the
           neighboring pixels of pixel $p$ in a block of size $3\times3$;
13     end
14 end
```

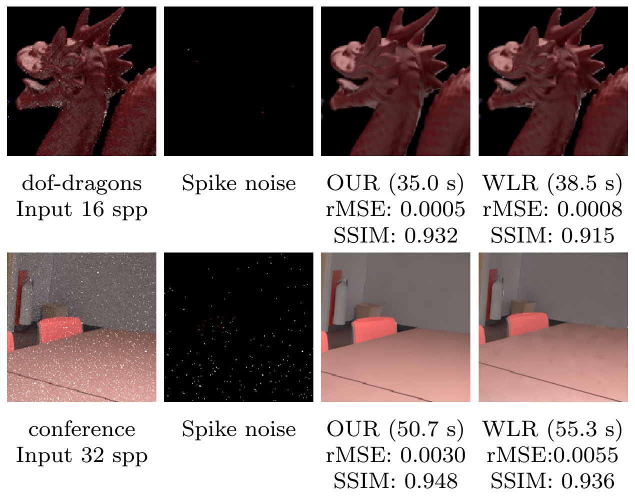


Fig. 5 Our method benefits from removing spike noise comparing with WLR method. We show that the spike noises are decomposed from input image. The WLR method leaves few spike noises in reconstructed image while our approach removes spike noise efficiently

Our algorithm can also recognize sparse spike noise from noisy image. Figure 5 shows more comparisons between our approach and the state-of-the-art algorithms.

4.2 Propagation filter

The propagation filter [3] is a pixel-based, edge preserving filter which targets homogeneous variance. It computes the weight of adjacent pixels based on their connected path, see Fig. 6a. If two pixels (e.g., c and q) are vertically or horizontally aligned, the path would connect the two pixels in a straight line. If two pixels are not simply vertically or horizontally connected, the filter determines the path based on their Manhattan distance. If the Manhattan distance between c and q is odd, the filter selects the path for traversing from pixel q to p in the vertical direction; otherwise, the horizontal path will be selected. Figure 6b gives the recursive steps for computing the weight between pixel c and q.

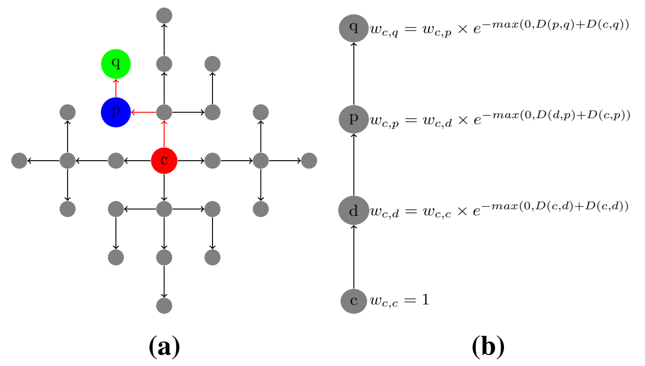

q
p
c
q
$w_{c,q} = w_{c,p} \times e^{-max(0,D(p,q)+D(c,q))}$
p
$w_{c,p} = w_{c,d} \times e^{-max(0,D(d,p)+D(c,p))}$
d
$w_{c,d} = w_{c,c} \times e^{-max(0,D(c,d)+D(c,d))}$
c
$w_{c,c} = 1$
(a)
(b)

Fig. 6 Pixel-based propagation filter, **a** the pattern for performing filtering with radius $r = 3$, and **b** the calculation of $w_{c,q}$

We extend the pixel-based propagation filter to patch-based and support for non-uniform variance using the range distance proposed by Rousselle et al. [28].

$$D(c, q) = \frac{(y_c - y_q)^2 - (\text{var}(c) + \text{var}[c, q])}{\epsilon + \text{var}(c) + \text{var}(q)} \tag{10}$$

where $\text{var}(c)$ is the buffer variance at pixel c, $\text{var}[c, q] = \min(\text{var}(c), \text{var}(q))$, and ϵ is a small number to prevent dividing by zero. Then the patch-based range distance is defined as:

$$D(\mathcal{P}(c), \mathcal{P}(q)) = \max\left(0, \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}_0} D(c + i, q + i)\right) \tag{11}$$

where $\mathcal{P}(c)$ is a square patch of size $|\mathcal{P}|$ (e.g., 7×7 pixels) centered on c, and $\mathcal{P}_0$ represents the offsets to each pixel within a patch. Figure 7 shows filtering results of pixel-based and patch-based propagation filter.

In order to compute $\text{var}(c)$, we evenly split all samples between two independent image buffers and estimate the buffer variance using the squared difference between the two buffers presented by Rousselle et al. [28].

There are $(2 \times r^2 + 2 \times r + 1)$ pixels in the filtering window with maximum Manhattan distance r. Even though the number is just half of the pixels in the squared filtering window with radius r (i.e., $(2 \times r + 1)^2$ pixels), the patch-based propagation filter gives better result compared with the feature prefiltering method proposed by Kalantari et al. [12], see the right column of Fig. 7.

4.3 Dimensionality reduction

We arrange auxiliary feature vectors in a filtering window center on pixel c as the rows of a data matrix $\mathbf{D} \in \mathbb{R}^{n\times d}$

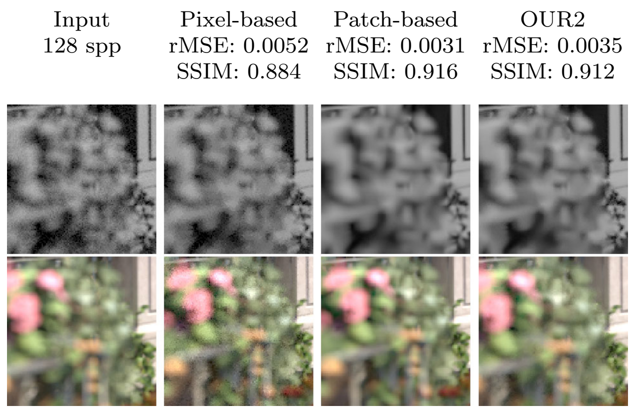


Fig. 7 Our reconstruction method with visibility feature ("sanmiguel" scene simulating strong depth-of-field effects) filtered by different approaches. Pixel-based propagation filter cannot smooth out inhomogeneous noise introduced by MC effects. Patch-based propagation filter gives nearly noise-free result. OUR2 is also reconstruction method with feature prefiltering by Kalantari et al. [12] and it also leaves few noise artifacts. The *bottom left image* is the reference

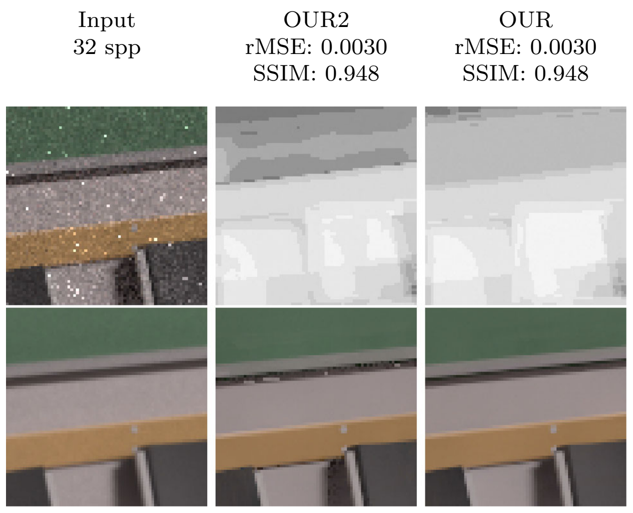


Fig. 8 Results of OUR2 and OUR method on the "conference" scene. OUR2 is also our method using the strategy of selecting the k biggest singular values proposed by WLR method. The strategy can omit noticeable singular values so that OUR2 method leaves few noise artifacts. The *bottom left image* is the reference

whose i-th row is set as $(\mathbf{x}_i - \mathbf{x}_c)^T$. Even though we filter the auxiliary features, there are still few residual filtering artifacts. To address this issue, we recover a low-rank matrix $\mathbf{A}$ from $\mathbf{D} = \mathbf{A} + \mathbf{E}$ using ALM function defined in Sect. 4.1. Then we perform singular value decomposition (SVD) on the covariance of $\mathbf{A}$ and identify the k_1 biggest singular values satisfying following condition:

$$\frac{\sum_{i=1}^{k_1} \sigma_i}{\sum_{i=1}^{r} \sigma_i} > 0.97 \tag{12}$$

where r is the rank of $\mathbf{A}$ and σ_i is the singular value of $\mathbf{A}$. We also compute the spectral norm of $\mathbf{E}$ and select the k_2 biggest singular value where $\sigma_i > C\|\mathbf{E}\|_2$ proposed by Moon et al. [21]. We use $k = \max(k_1, k_2)$ to select the k biggest singular values.

Moon et al. [21] suggest to select the k biggest singular values using $C\|\mathbf{E}\|_2$ (e.g., $C = 4$) as a criterion. However, selecting optimal C depends on noisy level of different scenes. If C is bigger, it may omit noticeable singular values. In order to mitigate this issue, we combine Eq. 12 with their proposed criterion to select the k biggest singular values. Figure 8 shows that our strategy of selecting the k biggest singular values can improve image quality.

The matrix $\mathbf{A}$ can be approximated by a compact form $\mathbf{A} \approx \mathbf{U}_k \mathbf{S}_k \mathbf{V}_k^T$, where $\mathbf{U}_k$ and $\mathbf{V}_k$ are $n \times k$ and $d \times k$ reduced unitary matrix, respectively, and $\mathbf{S}_k$ is a diagonal matrix that has k nonzero singular values. $\mathbf{V}_k$ is used in Sect. 3.1 to project auxiliary feature vector to low-dimensional subspace.

4.4 Estimate pilot bandwidth

After transforming auxiliary feature vectors into the reduced, local feature space, we estimate a per-pixel pilot bandwidth which is derived from KDE:

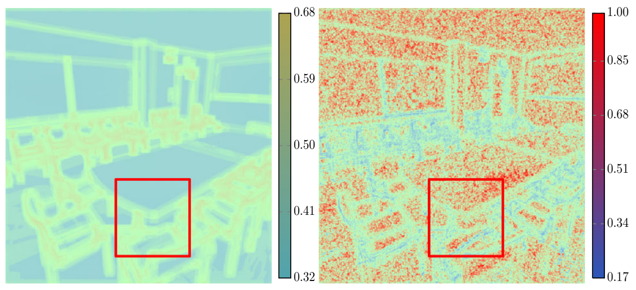
$$G = \mathbf{\Sigma} \left(\frac{4}{N_c \times (k+2)} \right)^{\frac{2}{k+4}} \tag{13}$$

where $\mathbf{\Sigma}$ is a $k \times k$ covariance matrix and N_c is the number of pixels which contributes to the center pixel c. The derivation of Eq. 13 can be found in Wand's book ([32], p. 111) and Kristan's paper [14].

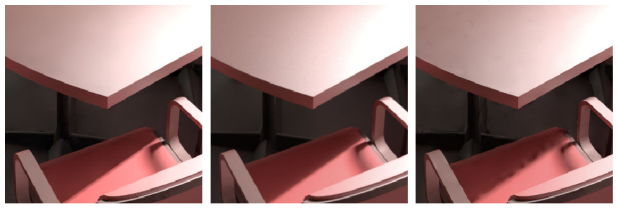
We set $\mathbf{\Sigma}$ to identify matrix. Since we have used SVD to project auxiliary feature vectors onto a low-dimensional subspace, they are not relevant. In order to compute N_c, we construct a weight vector $\mathbf{W} = (K_{\mathbf{H}}(\mathbf{z}_1 - \mathbf{z}_c), \ldots, K_{\mathbf{H}}(\mathbf{z}_n - \mathbf{z}_c))^T$. In order to bootstrap pilot bandwidth estimation, we set $\mathbf{H}$ to identity matrix, i.e., $b_j h = 1$ for $j = 1, \ldots, k$. Here we select arbitrary bandwidth $b_j h$ bigger to bootstrap our estimation, so that more samples can be included in the process of estimating the pilot bandwidth. N_c is the number of nonzeros in W. Then the shared bandwidth in Eq. 3 is estimated as follows:

$$h = \sqrt{\mathbf{G}(0, 0)} \tag{14}$$

Figure 9 demonstrates shared bandwidth comparison between our method (OUR) and weighted local regression (WLR) proposed by Moon et al. [21].



(a) OUR (2.1 s) **(b)** WLR (11.3 s)



(c) OUR (50.7 s) rMSE: 0.0030 SSIM: 0.948 **(d)** REFERENCE 65536 spp **(e)** WLR (55.3 s) rMSE: 0.0055 SSIM: 0.936

Fig. 9 Shared bandwidth comparison between our approach (**a**) and WLR (**b**) on the "conference" scene rendered with 32 spp. On the smoothed area, such as the surface of the table, WLR is prone to estimate larger bandwidth because of the influence of spike noise, but our approach estimates relative small bandwidth. We also give the time of estimating pilot bandwidth (OUR) and shared bandwidth (WLR). The middle (**d**) is the reference image. Even though WLR estimates larger bandwidth, it still remains splotch artifacts. In addition, WLR fails to reconstruct the edge of the shadow on the chair

5 Adaptive sampling

The state-of-the-art adaptive rendering methods (such as, Li et al. [15], Rousselle et al. [29] and Moon et al. [21,22]) adopt a common iterative approach to distribute additional ray samples to the regions with higher estimated MSE. We implement that iterative model. First of all, coarse sampling is initiated by uniformly allocating a small number of ray samples (e.g., four ray samples per pixel). Then, we estimate $\widehat{\text{MSE}}(\mathbf{z}_c)$ (Eq. 7) by our reconstruction method and predict an error reduction $\Delta\text{MSE}(\mathbf{z}_c) = \widehat{\text{MSE}}(\mathbf{z}_c) \times n(\mathbf{z}_c)^{\frac{-4}{k+4}}$, where the error reduction factor $n(\mathbf{z}_c)^{\frac{-4}{k+4}}$ is derived by Moon et al. [21] and $n(\mathbf{z}_c)$ is the number of ray samples which has been already allocated at pixel c.

Rousselle et al. [27] suggest relative MSE (rMSE) which is based on human visual perception that is more sensitive to darker areas. The rMSE is computed as follows: $\Delta\text{rMSE}(\mathbf{z}_c) = \frac{\Delta\text{MSE}(\mathbf{z}_c)}{\hat{m}^2(\mathbf{z}_c)+\epsilon}$. ϵ is a small number, e.g., 0.001, which is used to avoid the divide-by-zero. The rMSE estimation can be noisy, we filter it leveraging Gaussian filter in a 3×3 window. We sort the rMSE in an ascending order and select the upper rMSE at 95% position. We compute maximum sample count based on the upper rMSE and then select the minimum value between the maximum sample count and samples per pixel as a fixed value to clamp the per-pixel sample count which is proportional to $\Delta\text{rMSE}(\mathbf{z}_c)$.

6 Implementation details

We implemented our method as an extension of the PBRT2 rendering system [25] and adopted CUDA to accelerate our matrix operations. During sampling stage, PBRT2 renderer gathers color and auxiliary features for each pixel. The auxiliary features contain 10 dimensions: 3D coordinates, 3D textures, 3D normals and 1D direct light source visibility. We will normalize auxiliary features to the range $[-1,1]$. We use the optimal Epanechnikov kernel $\boldsymbol{K}(u) = \frac{3}{4}(1-u^2)$ for $u < 1$, as the kernel function in Eq. 3.

For all the test scenes, we use a 11×11 filtering window in the iterative steps and 23×23 filtering window for the final reconstruction. The reason why we use smaller filtering windows is that we only estimate relative MSE in the iterative steps. This decision only has slightly effect on adaptive sampling. We select $r = 5$ for feature prefiltering. Until all the sample budget is used up, we opt for a small number of iterations (i.e., 3) for adaptive sampling.

7 Results and discussion

We test all the scenes in this paper using a desktop with Intel(R) Xeon(R) CPU E5-1650 v3 @3.50GHz, and NVIDIA Geforce GTX TITAN X with CUDA 8.0 SDK for accelerating our reconstruction method. We compare our method with low discrepancy (LD) and three state-of-the-art algorithms, weighted local regression (WLR) [21], learning-based filter (LBF) [12] and nonlinearly weighted first-order regression (NFOR) [2]. To test WLR and LBF, we use the source code provided by the authors and set parameters recommended by their corresponding papers. To evaluate the NFOR method, we implemented it according to the pseudocode provided by authors.

For the purpose of comparing image quality generated by different methods in terms of quantitative measure, we use the relative mean squared error (rMSE) [27] which is defined as $\frac{1}{n}\sum_{i=1}^{n}\frac{(\hat{m}(\mathbf{x}_i)-m(\mathbf{x}_i))^2}{m(\mathbf{x}_i)^2+\epsilon}$, where $\epsilon = 0.001$ prevents from a divide-by-zero, and $m(\mathbf{x})$ refers to the ground truth image. We also use the structural similarity (SSIM) [33] index to measure the similarity between two images.

We verify our method on rendering following scenes, "pool" (1024 × 1024), "conference" (1024 × 1024), "san-miguel" (1024 × 1024) and "sibenik" (1024 × 1024). Each billiard ball has different motion blurs effect in the "pool" scene. The "conference" scene has indirect illumination and glossy surfaces that makes severe spike noise at low sam-

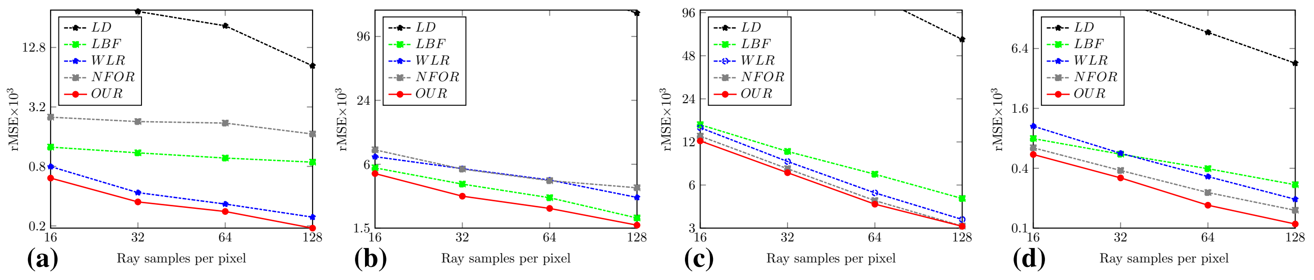


Fig. 10 Convergence plots for all tested scenes of Fig. 11 for LD, LBF, WLR, NFOR and OUR methods. **a** pool, **b** conference, **c** sanmiguel, **d** sibenik

pling rate. The "sanmiguel" scene has complex geometries and simulates strong depth-of-field effects. The "sibenik" scene has an environment light which can be seen by refraction through the windows.

We combine equal-time and equal-sample comparison based on rendering time and ray samples of our method for each comparison. If some methods run faster than ours using same ray samples, e.g., LD method, we increase ray samples for those methods to reach same rendering time as ours. If some methods run slower than ours using same ray samples, we do not decrease ray samples for those methods. Since only LD method runs faster than our method, we actually conduct equal-sample comparison among OUR, WLR, NFOR and LBF methods.

Figure 10 contains log–log convergence plots for the "pool", "conference", "sanmiguel" and "sibenik" scenes. The plots indicate that our method outperforms other methods across all the tested ray samples per pixel. The plots of "pool" scene for LBF and NFOR methods decrease slowly, since they do not implement adaptive sampling.

In Fig. 11 (last page) we compare rendering performance with LD, LBF, NFOR, WLR and our technique (OUR). We also include a reference image (REFERENCE) which is rendered by 64K samples per pixel (spp). The LD, LBF and NFOR methods use uniform sampling for all tested scenes. The WLR and OUR methods use adaptive sampling for all tested scenes.

In the "pool" scene, the comparison shows that our method has advantages over LBF and WLR. LBF preserves shadows in second row of the "pool" scene, while fails to reconstruct motion blur in the first row of the "pool" scene. Conversely, WLR smooths shadow edge seriously, while reconstructs motion blur efficiently. Our method not only preserves shadows but also generates a high-quality reconstruction result on the motion blurred region.

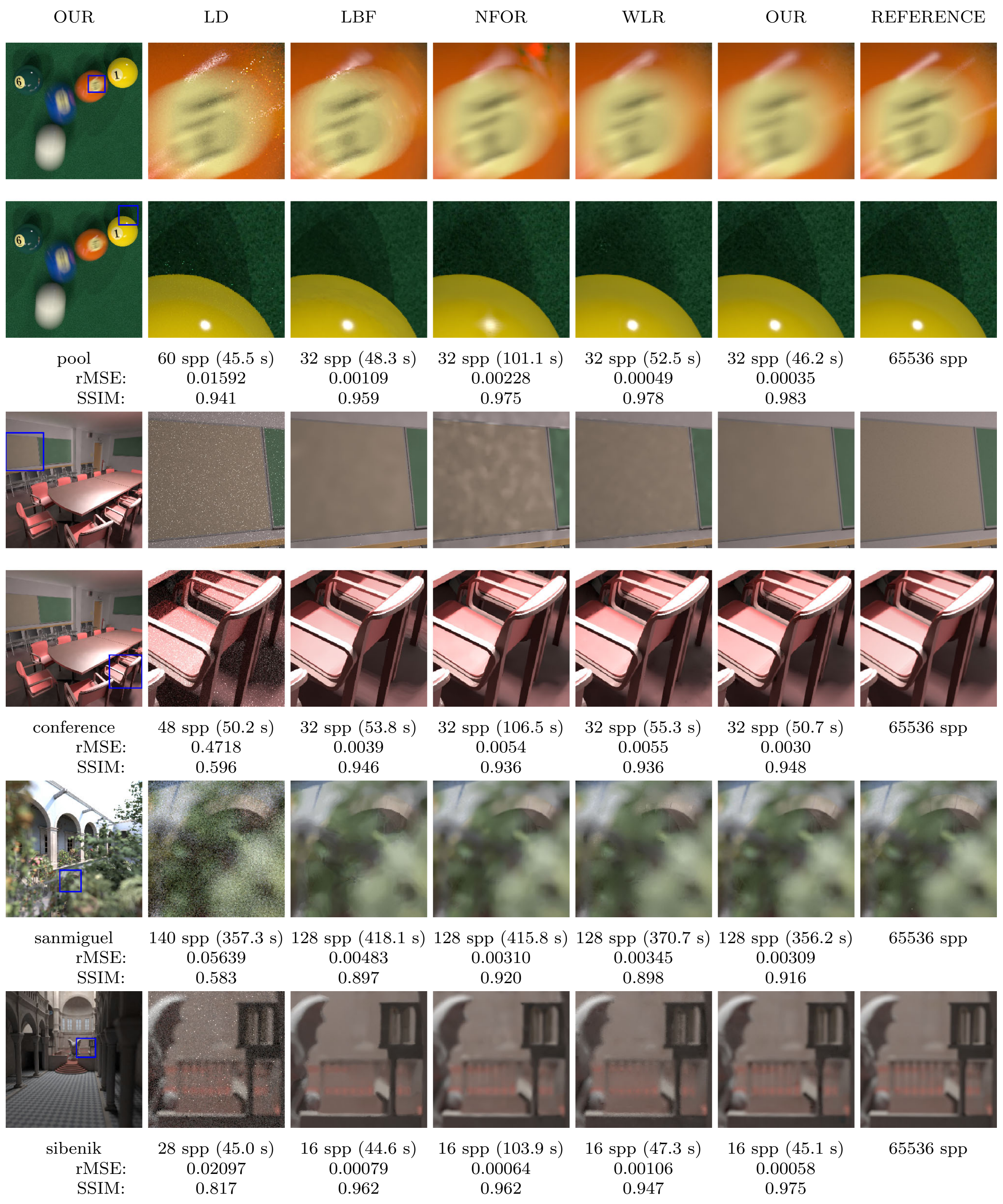
The first row comparison of "conference" scene shows the benefits of removing spike noise and estimating pilot bandwidth. LBF and WLR show splotch artifacts, while our method removes splotch artifacts efficiently, since we utilize the RPCA as a pre-precessing technique to remove sparse spike noise. The second row comparison of the "conference" scene shows the benefits of the direct light source visibility feature. We preserve soft shadows clearly, while WLR, which does not adopt direct light source visibility feature, cannot do so. LBF shows artifacts around soft shadows. In addition, our method has the minimum rMSE and maximum SSIM among these methods.

In the "sanmiguel" scene, LBF and WLR methods leave noise artifacts in the defocused areas. WLR cannot preserve shadows. Figure 9 is another case that WLR fails to reconstruct edge. Our method preserves the shadows and removes noise artifacts in the areas of depth-of-field due to feature prefiltering. In the "sibenik" scene, LBF and WLR produce over-blurred fence. Our method, on the other hand, preserves the fence as similar as the reference image.

In comparison with NFOR, our method produces better results for the "pool" and "conference" scenes. Adaptive sampling can improve the image quality of the "pool" scene significantly, but NFOR does not give an error metric to guide adaptive sampling. NFOR cannot handle spike noise and leaves obvious splotch artifacts in the "conference" scene. OUR and NFOR methods produce equivalent image quality in out-of-focus regions of "sanmiguel" and "sibenik" scenes.

NFOR evenly splits all samples between two image buffers and estimates four local linear estimators (each for one buffer with two different parameters) for each pixel. Furthermore NFOR employs block-based reconstruction in filtering window, so that each pixel will be reconstructed by $(2r + 1)^2$ linear models, where r is half size of filtering window. This is why NOFR obtains larger SSIM than our method for "sanmiguel" scene. This is also the reason why NFOR takes long time to finish reconstruction. For example, rendering time of NFOR is two times more than ours for the "pool", "conference" and "sibenik" scenes.

Adaptive polynomial rendering (APR) proposed by Moon et al. [23] is the most recent adaptive rendering method. APR uses the same approach as LBF to remove spike noise as a pre-process and pre-filters auxiliary features using image polynomial function. Their rMSE for "pool" scene with 31 spp is 0.00029, but our rMSE with 32 spp is 0.00035. Even though we obtain larger rMSE, our running time is faster than theirs (46.2s vs 65.8s). Our method has potential to reach the equivalent image quality in equal time.



OUR	LD	LBF	NFOR	WLR	OUR	REFERENCE
pool	60 spp (45.5 s)	32 spp (48.3 s)	32 spp (101.1 s)	32 spp (52.5 s)	32 spp (46.2 s)	65536 spp
rMSE:	0.01592	0.00109	0.00228	0.00049	0.00035	
SSIM:	0.941	0.959	0.975	0.978	0.983	
conference	48 spp (50.2 s)	32 spp (53.8 s)	32 spp (106.5 s)	32 spp (55.3 s)	32 spp (50.7 s)	65536 spp
rMSE:	0.4718	0.0039	0.0054	0.0055	0.0030	
SSIM:	0.596	0.946	0.936	0.936	0.948	
sanmiguel	140 spp (357.3 s)	128 spp (418.1 s)	128 spp (415.8 s)	128 spp (370.7 s)	128 spp (356.2 s)	65536 spp
rMSE:	0.05639	0.00483	0.00310	0.00345	0.00309	
SSIM:	0.583	0.897	0.920	0.898	0.916	
sibenik	28 spp (45.0 s)	16 spp (44.6 s)	16 spp (103.9 s)	16 spp (47.3 s)	16 spp (45.1 s)	65536 spp
rMSE:	0.02097	0.00079	0.00064	0.00106	0.00058	
SSIM:	0.817	0.962	0.962	0.947	0.975	

Fig. 11 Comparisons of our approach (OUR) to LD, LBF, NFOR and WLR. At the bottom of the images we give the number of ray samples per pixel (spp), rendering time in seconds, rMSE and SSIM values. OUR and WLR methods produce better results in motion blur regions in the "pool" scene due to adaptive sampling. OUR and LBF methods preserve shadow edge in the second row of "pool" and "conference" scenes due to the visibility feature. The results of OUR method in the first row of "conference" scene highlight the benefits of removing spike noise. OUR, LBF and NFOR methods preserve the small direct shadow details in "sanmiguel" scene because of the absence of the visibility feature. OUR and NFOR methods produce outputs largely free of artifacts in out-of-focus regions of "sanmiguel" and "sibenik" scenes because of the efficiency of feature prefiltering

In summary, we ascribe our improvements over the previous state-of-the-art algorithms to removing spike noise, feature prefiltering and estimating pilot bandwidth. The local linear estimator is sensitive to spike noise. Removing spike noise makes the local linear estimator to reduce the estimated bias, but our method does not entirely rely on removing spike noise. Removing outlier and feature prefiltering can reduce the MSE of the local linear estimator since it requires nearly noise-free weights. Our pilot bandwidth estimation also plays an important role in improving image quality and reducing computational overhead of estimating optimal bandwidth.

Adaptive rendering methods based on auxiliary features generate suboptimal results when the auxiliary features have low correlations with the color image at low sampling rate. This is also the limitation of our method, we want to make further improvement on this. Another limitation is that we do not extend our method to handle animated images, but we can employ similar approach presented by Moon et al. [21] to handle animations.

8 Conclusions

We have presented a method based on the RPCA to efficiently denoise images with diverse MC effects. We utilize the RPCA as a pre-processing step to decompose spike noise from noisy image. We filter the auxiliary features leveraging the patch-based propagation filter. We use the robust PCA to reduce dimensionality in high-dimensional feature space and apply SVD to project that space into a local, reduced feature space. We also introduce a novel approach to estimate a per-pixel pilot bandwidth derived from KDE. We adopt the local linear estimator to compute the value of each pixel. In sum, decomposing spike noise, feature prefiltering, reducing dimensionality by the RPCA, estimating pilot bandwidth and direct light source visibility features make our method have significant improvements in terms of both numerical error and visual quality compared to the previous state-of-the-art. In addition, many regression techniques in statistics are useful for reconstructing high-quality image, especially for multivariate regression. Whether estimating bandwidth for each feature, which needs high computational overhead, will improve denoising efficiency is unexplored. We would like to investigate if there is an efficient way to do this.

Acknowledgements Funding was provided by The National High Technology Research and Development Program of China (863 Program) (Grant No. 2012AA011206).

References

1. Belcour, L., Soler, C., Subr, K., Holzschuch, N., Durand, F.: 5D covariance tracing for efficient defocus and motion blur. ACM Trans. Gr. **32**(3), 31:1–31:18 (2013)
2. Bitterli, B., Rousselle, F., Moon, B., Guitián, J.A.I., Adler, D., Mitchell, K., Jarosz, W., Novák, J.: Nonlinearly weighted first-order regression for denoising monte carlo renderings. Comput. Graph. Forum **35**(4), 107–117 (2016)
3. Chang, J.H.R., Wang, Y.C.F.: Propagated image filtering. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10–18 (2015)
4. Cheng, M.Y., Peng, L.: Simple and efficient improvements of multivariate local linear regression. J. Multivar. Anal. **97**(7), 1501–1524 (2006)
5. Dabov, K., Foi, A., Egiazarian, K.: Video denoising by sparse 3D transform-domain collaborative filtering. In: Signal Processing Conference, 2007 15th European, pp. 145–149 (2007)
6. DeCoro, C., Weyrich, T., Rusinkiewicz, S.: Density-based outlier rejection in Monte Carlo rendering. Comput. Gr. Forum **29**(7), 2119–2125 (2010)
7. Delbracio, M., Musé, P., Buades, A., Chauvier, J., Phelps, N., Morel, J.M.: Boosting Monte Carlo rendering by ray histogram fusion. ACM Trans. Gr. **33**(1), 8:1–8:15 (2014)
8. Donoho, D.L., Johnstone, I.M.: Ideal spatial adaptation by wavelet shrinkage. Biometrika **81**(3), 425–455 (1994)
9. Farrugia, J.P., Peroche, B.: A progressive rendering algorithm using an adaptive perceptually based image metric. Comput. Gr. Forum **23**(3), 605–614 (2004)
10. Hachisuka, T., Jarosz, W., Weistroffer, R.P., Dale, K., Humphreys, G., Zwicker, M., Jensen, H.W.: Multidimensional adaptive sampling and reconstruction for ray tracing. In: ACM SIGGRAPH 2008 Papers, SIGGRAPH ’08, pp. 33:1–33:10. ACM, New York (2008)
11. Kajiya, J.T.: The rendering equation. In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’86, pp. 143–150. ACM, New York (1986)
12. Kalantari, N.K., Bako, S., Sen, P.: A machine learning approach for filtering Monte Carlo noise. ACM Trans. Gr. **34**(4), 122:1–122:12 (2015)
13. Kalantari, N.K., Sen, P.: Removing the noise in Monte Carlo rendering with general image denoising algorithms. Comput. Gr. Forum (Proc. Eurogr. 2013) **32**(2), 93–102 (2013)
14. Kristan, M., Leonardis, A., Skočaj, D.: Multivariate online kernel density estimation with Gaussian kernels. Pattern Recogn. **44**(10–11), 2630–2642 (2011)
15. Li, T.M., Wu, Y.T., Chuang, Y.Y.: SURE-based optimization for adaptive sampling and reconstruction. ACM Trans. Gr. **31**(6), 194:1–194:9 (2012)
16. Lin, Z., Chen, M., Ma, Y.: The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. ArXiv e-prints (2010)
17. Liu, X., Zheng, C.: Parallel adaptive sampling and reconstruction using multi-scale and directional analysis. Vis. Comput. **29**(6–8), 501–511 (2013)
18. Liu, X., Zheng, C.: Adaptive cluster rendering via regression analysis. Vis. Comput. **31**(1), 105–114 (2015)
19. Liu, X.D., Wu, J.Z., Zheng, C.W.: Kd-tree based parallel adaptive rendering. Vis. Comput. **28**(6–8), 613–623 (2012)
20. Mitchell, D.P.: Generating antialiased images at low sampling densities. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’87, pp. 65–72. ACM, New York (1987)
21. Moon, B., Carr, N., Yoon, S.E.: Adaptive rendering based on weighted local regression. ACM Trans. Gr. **33**(5), 170:1–170:14 (2014)
22. Moon, B., Iglesias-Guitian, J.A., Yoon, S.E., Mitchell, K.: Adaptive rendering with linear predictions. ACM Trans. Gr. **34**(4), 121:1–121:11 (2015)
23. Moon, B., McDonagh, S., Mitchell, K., Gross, M.: Adaptive polynomial rendering. ACM Trans. Gr. **35**(4), 40:1–40:10 (2016)

24. Overbeck, R.S., Donner, C., Ramamoorthi, R.: Adaptive wavelet rendering. In: ACM SIGGRAPH Asia 2009 Papers, SIGGRAPH Asia '09, pp. 140:1–140:12. ACM, New York (2009)
25. Pharr, M., Humphreys, G.: Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann Publishers Inc., San Francisco (2010)
26. Rigau, J., Feixas, M., Sbert, M.: Refinement criteria based on f-divergences. In: Proceedings of the 14th Eurographics Workshop on Rendering, EGRW '03, pp. 260–269. Eurographics Association, Aire-la-Ville (2003)
27. Rousselle, F., Knaus, C., Zwicker, M.: Adaptive sampling and reconstruction using greedy error minimization. ACM Trans. Gr. **30**(6), 159:1–159:12 (2011)
28. Rousselle, F., Knaus, C., Zwicker, M.: Adaptive rendering with non-local means filtering. ACM Trans. Gr. **31**(6), 195:1–195:11 (2012)
29. Rousselle, F., Manzi, M., Zwicker, M.: Robust denoising using feature and color information. Comput. Gr. Forum **32**(7), 121–130 (2013)
30. Sen, P., Darabi, S.: On filtering the noise from the random parameters in Monte Carlo rendering. ACM Trans. Gr. **31**(3), 18:1–18:15 (2012)
31. Stein, C.M.: Estimation of the mean of a multivariate normal distribution. Ann. Stat. **9**(6), 1135–1151 (1981)
32. Wand, M.P., Jones, M.C.: Kernel Smoothing. Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton (1995)
33. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. Trans. Imaging Proc. **13**(4), 600–612 (2004)
34. Yan, L.Q., Mehta, S.U., Ramamoorthi, R., Durand, F.: Fast 4D sheared filtering for interactive rendering of distribution effects. ACM Trans. Gr. **35**(1), 7:1–7:13 (2015)
35. Zwicker, M., Jarosz, W., Lehtinen, J., Moon, B., Ramamoorthi, R., Rousselle, F., Sen, P., Soler, C., Yoon, S.E.: Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. Comput. Gr. Forum (Proc. Eurogr.) **34**(2), 667–681 (2015)

Hongliang Yuan is a Ph.D. candidate in the Science and Technology on Integrated Information System Laboratory, Institute of Software, Chinese Academy of Sciences. He received his B.S. degree from Dalian University of Technology in 2003 and master degree from Dalian University of Technology in 2006. His research interests include computer graphics and realistic rendering.

Changwen Zheng is a researcher in the Science and Technology on Integrated Information System Laboratory, Institute of Software,Chinese Academy of Sciences. He received his Ph.D. degree from Huazhong University of Science and Technology. His research interests include computer graphics, virtual reality, computer simulation and artificial intelligence.