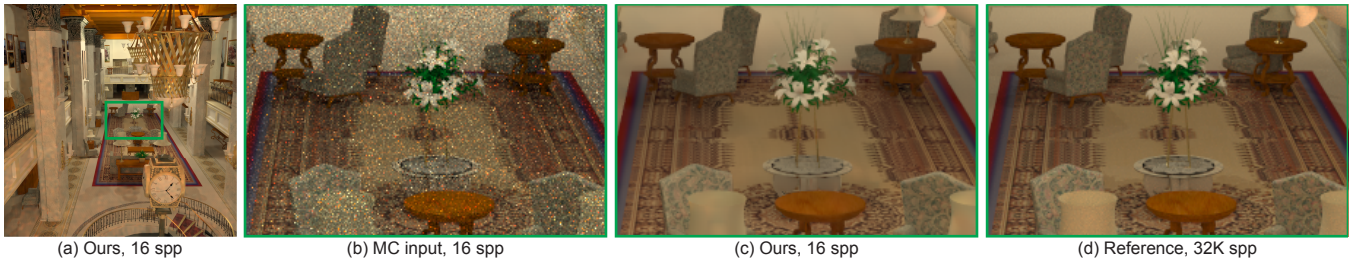
Adaptive Polynomial Rendering

Bochang Moon[1], Steven McDonagh[1], Kenny Mitchell[1,2], Markus Gross[1,3]
[1]Disney Research, [2]Edinburgh Napier University, [3]ETH Zurich



(a) Ours, 16 spp rMSE 0.00541 | (b) MC input, 16 spp rMSE 0.68832 | (c) Ours, 16 spp rMSE 0.00541 | (d) Reference, 32K spp

Figure 1: *Our adaptive rendering result for the Hotel Lobby scene. Given a noisy input image using a small number of samples per pixel (spp) (b), our method (a) and (c) effectively removes MC noise while avoiding excessive blurring by performing a novel reconstruction using adaptively chosen polynomials. In addition, our method drastically reduces the relative mean squared error (rMSE) of the input (b).*

Abstract

In this paper, we propose a new adaptive rendering method to improve the performance of Monte Carlo ray tracing, by reducing noise contained in rendered images while preserving high-frequency edges. Our method locally approximates an image with polynomial functions and the optimal order of each polynomial function is estimated so that our reconstruction error can be minimized. To robustly estimate the optimal order, we propose a multi-stage error estimation process that iteratively estimates our reconstruction error. In addition, we present an energy-preserving outlier removal technique to remove spike noise without causing noticeable energy loss in our reconstruction result. Also, we adaptively allocate additional ray samples to high error regions guided by our error estimation. We demonstrate that our approach outperforms state-of-the-art methods by controlling the tradeoff between reconstruction bias and variance through locally defining our polynomial order, even without need for filtering bandwidth optimization, the common approach of other recent methods.

Keywords: Adaptive rendering, image-space reconstruction, Monte Carlo ray tracing

Concepts: •**Computing methodologies** → **Ray tracing;**

1 Introduction

Monte Carlo (MC) ray tracing [Kajiya 1986] has been recognized as a powerful rendering algorithm to synthesize photo-realistic images from 3D models, and its popularity comes mainly from the generality that it simulates a variety of rendering effects through a unified framework, i.e., ray tracing. The rendering time of MC ray tracing, however, is often unacceptable for practical purposes, since converged rendered images are typically generated by integrating a huge number of ray samples, e.g., more than 10K ray samples per pixel ((d) in Fig. 1).

To tackle the performance problem of MC ray tracing, adaptive rendering techniques (e.g., [Overbeck et al. 2009; Hachisuka et al. 2008]) have been actively studied, and the main components are locally controlled sampling rate and adaptively adjusted reconstruction. Adaptive sampling utilizes the heterogeneous noise property of rendered images to guide irregular sampling density rather than a uniform sampling, and adaptive reconstruction locally controls smoothing by considering MC noise so that high-frequency edges are properly preserved.

Adaptive rendering has a long history (e.g., [Kajiya 1986]), and the seminal paper [Kajiya 1986] presented a general idea that high-dimensional MC samples can be allocated adaptively in a hierarchical structure by using the variances of the samples and these samples can be integrated to generate rendered images. These approaches typically show high-quality rendering results for a moderate dimension (e.g., 2 to 5), but they typically suffer from the curse-of-dimensionality as the dimension of MC samples can be high (e.g., more than 10) when global illumination is simulated.

The image-space adaptive methods (e.g., [Overbeck et al. 2009]) have addressed the dimensionality issue by analyzing MC errors (e.g., variance) in 2D image space and denoising the errors through an established image filter, guided by estimated errors. This approach has received attention due to its intrinsic generality and simplicity compared to the high-dimensional adaptive methods. These methods utilize different image filters, but the common high-level behavior is to control filtering bandwidths at each pixel to minimize a numerical error [Sen and Darabi 2012]. Especially, the recent adaptive methods [Li et al. 2012; Rousselle et al. 2013; Moon et al. 2014; Moon et al. 2015] proposed optimization algorithms to estimate optimal filtering bandwidths per pixel to minimize the mean squared error (MSE) of reconstruction results, which can be mathematically decomposed into bias squared and variance. Technically speaking, these attempts can be considered optimization processes that compute an optimal balance between bias and variance, caused by over- and under-blurring, respectively.

These state-of-the-art methods demonstrated that optimizing band-

SIGGRAPH '16 Technical Paper, July 24 - 28, 2016, Anaheim, CA
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925936

ACM Reference Format
Moon, B., McDonagh, S., Mitchell, K., Gross, M. 2016. Adaptive Polynomial Rendering.
ACM Trans. Graph. 35, 4, Article 40 (July 2016), 10 pages. DOI = 10.1145/2897824.2925936
http://doi.acm.org/10.1145/2897824.2925936.

widths can be an effective approach for high-quality reconstruction, but the approximation function used in their methods is fixed (typically as a low-order function) despite its importance. Especially, high-order functions can have better approximation quality than low-order ones for the image area where unknown image functions have high curvature. Increasing the order of functions used for reconstruction allows for better approximation of larger areas when compared to fixed, low-order functions. This property leads to our novel approach of controlling the bias-variance tradeoff by locally changing polynomial function order in place of the typical bandwidth adaptation filtering. To the best of our knowledge, this is the first approach to use adaptively chosen polynomials in the adaptive rendering framework. Our main technical contributions are listed as follows:

- We locally approximate a rendered image with polynomial functions and each polynomial is constructed with an estimated optimal order so that our reconstruction error can be minimized.
- A multi-stage error estimation process is presented to robustly estimate our reconstruction error and the optimal polynomial order for each polynomial function.
- A new energy-preserving outlier removal is presented in order to remove spike noise, which has excessive intensity, in rendered images while minimizing a noticeable energy loss caused by outlier removal.
- Sampling rate is locally controlled across image space based on estimated reconstruction errors.

We have demonstrated that our new approach outperforms state-of-the-art methods that optimize filtering bandwidth locally, even without performing any bandwidth selection process, due to our bias-variance control through adaptive polynomial selection.

2 Related Work

In this section, we briefly discuss previous techniques related to our work. We refer to a recent survey [Zwicker et al. 2015] for a comprehensive review.

Multi-dimensional Adaptive Methods. The seminal paper [Kajiya 1986] introduced the general idea of utilizing hierarchical structures where each high-dimensional MC sample is stored and generated, and rendered images are generated by summing stored samples via a Riemann sum. Hachisuka et al. [2008] demonstrated that the high-level idea can drastically improve MC performance for distributed rendering effects such as motion blur and depth-of-field. In theory, the general idea can be applied to simultaneously address all rendering effects including soft shadows and global illumination, but the sample density in the hierarchy, e.g., kd-tree, becomes exponentially sparse as the dimension increases. As a result, it is still a fundamental problem to efficiently and robustly perform adaptive sampling and reconstruction in order to avoid the curse-of-dimensionality.

Frequency Analysis based Techniques. Frequency analysis based reconstruction for MC ray tracing has been actively studied since it can provide high-quality reconstruction results guided by the well-established principle that analyzes light transport in frequency space [Durand et al. 2005]. Sophisticated anisotropic reconstruction methods using the theory have been proposed for specific rendering effects such as depth-of-field [Soler et al. 2009], motion blur [Egan et al. 2009], soft shadows [Egan et al. 2011b], ambient occlusions [Egan et al. 2011a], distributed effects [Lehtinen et al. 2011], and indirect illumination [Lehtinen et al. 2012]. Recently, the idea of simplifying the frequency based anisotropic filters with a rectangle-shaped filter was presented to design an efficient axis-aligned filter for interactively reconstructing soft shadows [Mehta et al. 2012], indirect lighting components caused by diffuse (and moderately glossy) bounces [Mehta et al. 2013], and distribution effects [Mehta et al. 2014]. More recently, Yan et al. [2015] proposed a novel optimization technique that achieves an interactive frame rate even while maintaining the shape of sheared filters. These methods demonstrated outstanding rendering results even for noisy images, rendered by a small number of samples, but these approaches often supported only a subset of rendering effects.

Image-space Adaptive Approaches. Image-space methods including our technique have proven a popular approach as they are simple to implement and are not limited to specific rendering effects. The main challenge for this class of methods is the fundamental difficultly in discerning image features, i.e., edges, from high-frequency noise. The common adaptation of existing image filters for tackling this challenge is to fully utilize rendering-specific information such as variances, textures, normals, and depths. As an early work, McCool [1999] proposed a modified anisotropic diffusion process using textures, normals, depths, and the respective variances in order to reduce MC noise while preserving high-frequency edges.

Recently, Overbeck et al. [2009] presented a multi-resolution analysis of image variances using wavelet basis functions to adaptively control sample counts and smoothing via wavelet thresholding. In a similar multi-resolution framework, a robust noise estimation using the median absolute deviation was introduced [Kalantari and Sen 2013]. Rouselle et al. [2011; 2012] estimated a reconstruction error (e.g., MSE) for Gaussian and non-local means filters, respectively, and estimated optimal filtering bandwidths at each pixel. For non-local means filters, a new similarity measure between two patches was defined, which computes a distance between histograms of ray samples [Delbracio et al. 2014]. These approaches are relatively simple to implement and generate high-quality reconstruction results, but a common disadvantage of these methods is in the difficulty to preserve geometric edges introduced by discontinuities in texture, normal, and depth buffers, since these buffers are not utilized.

Sen and Darabi [2012] designed a new technique to robustly utilize geometric information by estimating optimal filtering bandwidths used in a cross-bilateral filter through mutual information. A general estimator, Stein's unbiased risk estimator (SURE), was introduced by Li et al. [2012], and the filtering bandwidths of non-linear filters such as non-local means and cross-bilateral filter were optimized using the estimator. Rousselle et al. [2013] demonstrated that a wider set of rendering features such as visibility and caustics can be utilized thanks to the generality of SURE. Kalantari et al. [2015] introduced a machine learning approach to robustly estimate filtering bandwidths especially for small numbers of samples (e.g., 8) given a cross-bilateral or cross non-local means framework. Moon et al. [2014] proposed a linear function based reconstruction using weighted local regression, and locally estimated optimal filtering bandwidths for different types of features. The computation overhead of the local regression framework was recently improved by applying an approximation technique that performs expensive optimization at only a sparse number of pixels [Moon et al. 2015].

These techniques use various reconstruction frameworks, but the high-level approach is to control filtering bandwidths locally to increase numerical accuracy. Our work takes a completely different approach to improve the numerical accuracy by adaptively adjusting a polynomial order to better adapt image signals in a data-driven way. For example, we choose a high-order (e.g., cubic) function to accurately reconstruct high curvature regions locally rather than

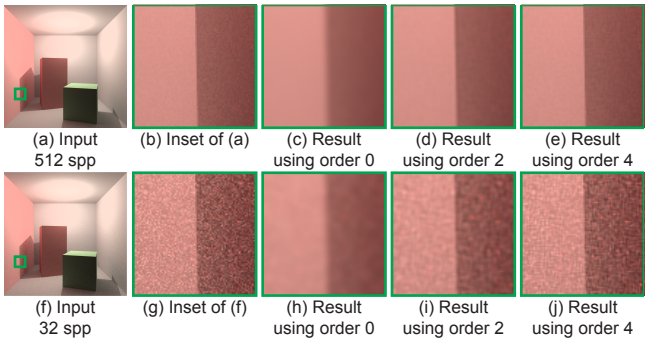


Figure 2: *Results with polynomials using differing orders, given the input images rendered with different sample counts. Higher order polynomials (e.g., fourth order) preserve the discontinuous edge better, but show higher noise compared to lower order polynomials (e.g., zero and second order). This tradeoff motivates our adaptive control between the bias and variance of our reconstruction result. We achieve this by changing the polynomial order in a data-driven way.*

shrinking filtering bandwidths, and a low order function (e.g., linear) is selected for smooth areas.

3 Our Reconstruction Framework

In this section, we present a general optimization framework from an optimization point of view. Given an input image function $y(i)$ at 2D pixel position $i \in \mathbb{R}^2$, we assume the following statistical model:

$$y(i) = \mu(i) + \epsilon(i), \tag{1}$$

where $\mu(i)$ is the ground truth intensity that can only be computed with an infinite number of samples. The $\epsilon(i)$ models the MC noise, i.e., variance, which has zero mean, i.e., $E(\epsilon(i)) = 0$, where E is the expectation operator. The ultimate goal of any reconstruction method, including ours, is to estimate the unknown $\mu(i)$ from the observed noisy input $y(i)$. We suppose $y(i)$ to be a 1D function since we can apply our reconstruction to each color channel independently.

Given the statistical model, we decompose the unknown image function into two functions $g(f_i)$ and $p(i)$ such that $\mu(i) \equiv g(f_i) + p(i)$, where $g(f_i)$ is a linear function that takes a feature vector $f_i \in \mathbb{R}^8$, e.g., normals (3D), textures (3D), depths (1D), and visibility (1D) as inputs, which can easily be computed at the intersection points between rays and scenes during the rendering process. On the other hand, the second function $p(i)$ is defined as a 2D function that takes pixel position i as input. The rationale behind this decomposition is that we approximate the unknown $\mu(i)$ with the feature function $g(f_i)$ using correlation between rendering-specific features f_i and the unknown image function $\mu(i)$. When the features are not helpful, e.g., glossy reflections, we can approximate the residuals $\mu(i) - g(f_i)$ using the 2D polynomial function $p(i)$ since the residuals are intrinsically 2D image functions.

Specifically, we locally approximate the unknown function using Taylor polynomials:

$$\mu(i) \approx \nabla g(f_c)(f_i - f_c)^T + p(c) + \sum_{1 \le a \le k} \frac{\nabla^a p(c)}{a!}((i-c)^a)^T, \tag{2}$$

where ∇ and k are the differentiation operator and the Taylor polynomial order that we use, respectively. Note that this polynomial model does not need to use $g(f_c)$, since the model already includes a constant term, i.e., $p(c)$. For simplicity's sake, we treat the 2D image index i as a 1D value, but one can easily find the high-dimensional Taylor series (2D in our case) in the literature (e.g., [Ruppert and Wand 1994]). Given a specific order k, the coefficients of two functions can be optimally estimated within a local window Ω_c in the least-square sense:

$$\sum_{i \in \Omega_c} \left(y(i) - \alpha(f_i - f_c)^T - \beta_0 - \sum_{1 \le a \le k} \beta_a((i-c)^a)^T \right)^2 K_h(i). \tag{3}$$

This least-squares optimization provides estimated coefficients $\hat{\alpha}$ and $\hat{\beta}$ for the two functions $g(f_i)$ and $p(i)$ given a specific order k, respectively. The kernel function $K_h(i)$ is a weighting function (e.g., Gaussian) that assigns a weight on pixel i, and the parameter h, used by the function, is known as the filtering bandwidth that controls the bias-variance tradeoff of this reconstruction process. The least-squares formula has a closed-from solution, *normal equation*, $(\hat{\alpha}, \hat{\beta}) = (X_k^T W X_k) X_k^T W y$. Each row of the design matrix X_k is set as $[1, f_i, (i-c)^1, ..., (i-c)^k]^T$, and W is a diagonal matrix that has $K_h(i)$ as its elements. Note that the size of the design matrix X_k becomes larger as the polynomial order k increases. The vector y has MC input intensities $y(i)$ as its elements.

Given the normal equation, reconstructed values within the window Ω_c can be computed in a pixel-wise manner [Moon et al. 2014]:

$$\hat{y}_k(c) = e_1 (X_k^T W X_k) X_k^T W y, \tag{4}$$

where e_1 is a vector that has one as its first element, i.e., $e_1 = [1, 0, ..., 0]^T$. Alternatively, we can compute multiple reconstructed values simultaneously [Moon et al. 2015]:

$$\hat{y}_k = X_k (X_k^T W X_k) X_k^T W y = H(k) y, \tag{5}$$

where the matrix $H(k) = X_k (X_k^T W X_k) X_k^T W$ is known as the hat matrix that defines a projection from the input values y to projected values $\hat{y}_k$. After computing the reconstructed values at each center pixel, we combine the output vector $\hat{y}_k$ from each polynomial model at center pixel c since the reconstructed pixels in two regions, e.g., Ω_{c1} and Ω_{c2} from two center pixels $c1$ and $c2$, may overlap. Hence, a weighted average using $K_h(i)$ is utilized to compute our final reconstruction output $\hat{y}(i)$ at each pixel i as follows:

$$\hat{y}(i) = \sum_{j \in \Omega_i} K_h^j(i) \hat{y}_k^j(i) / \sum_{j \in \Omega_i} K_h^j(i), \tag{6}$$

where $K_h^j(i)$ and $\hat{y}_k^j(i)$ are the weight and reconstruction result for pixel i computed from center pixel j. We employ this block-based reconstruction instead of point-wise (Eq. 4) as our polynomials can adapt well to large regions, i.e., Ω_c, by increasing the polynomial order k. This forms one of the crucial advantages of using higher-order polynomials for this task. It also leads to a robust error analysis that estimates reconstruction errors in a region (Sec. 4).

Classification of Existing Methods. It is worth noting that existing filters such as Gaussian, cross-bilateral, non-local means, and weighted local regression can be explained with this general optimization (Eq. 3). For example, when we set $g(f_i)$ and $p(i)$ as a

null and zero-order function, the optimization framework is converted to a simpler form such as Gaussian (e.g., [Rousselle et al. 2011]), cross-bilateral (e.g., [Li et al. 2012]), and non-local means filter (e.g., [Rousselle et al. 2012]) depending on the kernel function $K_h(i)$. Also, if we use $g(f_i)$ and set $p(i)$ as a linear function, the formulation is equivalent to the weighted local regression framework [Moon et al. 2014]. One can easily verify this through the pixel-wise equation (Eq. 4). As a result, these previous methods use fixed functions $g(f_i)$ and $p(i)$ with a typically low-order (e.g., zero or linear), and then optimize the kernel function $K_h(i)$ in terms of the filtering bandwidth h. Our method, however, controls the polynomial order of $p(i)$ to adjust the bias-variance tradeoff instead of optimizing the kernel function $K_h(i)$. This general framework allows a comprehensive optimization of both polynomial order and kernel bandwidth, however in this work we limit our scope to polynomial order optimization, in order to clearly verify that our adaptively selected polynomials independently provide a valuable approach enabling high-quality adaptive rendering. To this end, we set the kernel function as a simple Gaussian function, i.e., $K_h(i) \equiv e^{-||i-c||^2/(2h^2)}$ and h as a large constant, e.g., half-width of the local window size Ω_c.

Higher Order Polynomials. Our key motivation is that high-order polynomials (e.g., quadratic, cubic, and higher order) are able to provide improved approximations of non-linear functions (e.g., discontinuous signals). This can be considered counter-intuitive since we are using smooth functions (i.e., Taylor polynomials). One may think that increasing the polynomial order can decrease the approximation quality, i.e., Runge's phenomenon, but note that the unknown function $\mu(i)$ is defined on a discretized space i, i.e., pixel position, instead of a continuous space. As we increase the polynomial order, our method approaches an interpolation method that passes the input value $y(i)$ (i.e., zero bias), since the degree of freedom, i.e., number of unknown parameters of our optimization, also goes to the number of pixels within the filtering window.

Fig. 2 illustrates reconstruction results using different polynomials for a non-linear edge. For this test, we are using only the image polynomial function $p(i)$ to clearly verify our motivation. One can clearly see the result using a high order (i.e., fourth) preserves the non-linear edge well even when we use a smooth polynomial. Technically speaking, they decrease the magnitude of our reconstruction bias, $|E(\hat{y}(i) - \mu(i))|$, since the bias function is represented by the remainder terms of the Taylor series (Eq. 2). However, high-order functions tend to produce noisier reconstruction results than low-order as they typically increase the variance of reconstruction (see [Ruppert and Wand 1994] for a formal derivation of the errors). These properties motivate our design of adaptive polynomial selection, that afford a novel method for controlling the bias-variance tradeoff by adjusting the polynomial order instead of the well-known approach of tweaking the filtering bandwidth term h in $K_h(i)$.

4 Adaptive Polynomial Reconstruction

In this section, we propose a principled method to locally optimize the polynomial order k. Our method is a block-based reconstruction (Eq. 6), and thus our optimization goal is also defined in a block-based manner rather than pixel-wise. Specifically, we define the reconstruction error of a polynomial at center pixel c as the $L2$ error:

$$\xi_c(k) \equiv \frac{1}{\sum_{i\in\Omega_c} K_h(i)} \sum_{i\in\Omega_c} K_h(i) \left(\hat{y}_k(i) - \mu(i)\right)^2. \quad (7)$$

Given this definition of our reconstruction error, we can choose an optimal order k_{opt} that has a minimal error $\xi_c(k_{opt})$. Unfortunately, this optimization cannot be directly solved because it is based on the actual $L2$ error $\left(\hat{y}_k(i) - \mu(i)\right)^2$ that uses the unknown image values $\mu(i)$. Hence, we should estimate the actual error term $\xi_c(k)$ and thus the unknown optimal order k_{opt}. To this end, we mathematically express our reconstruction bias and variance (Sec. 4.1), and propose a robust estimation process for the error terms (Sec. 4.2). In addition, we present an energy-preserving outlier removal to effectively remove spike noise without noticeable energy loss (Sec. 4.3).

4.1 Bias and Variance Expression

To estimate the optimal order k_{opt} of the polynomial function $p(i)$, the crucial step involves estimating the actual error $\left(\hat{y}_k(i) - \mu(i)\right)^2$ (in Eq. 7). To this end, we decompose the error into bias and variance components by taking the expectation operator E, similarly to Rousselle et al. [2011]:

$$E\left(\hat{y}_k(i) - \mu(i)\right)^2 = bias^2(\hat{y}_k(i)) + \sigma^2(\hat{y}_k(i)). \quad (8)$$

The bias function, $bias(\hat{y}_k(i))$, can be computed using the hat matrix (in Eq. 5):

$$\begin{aligned} E(\hat{y}_k(i) - \mu(i)) &= \sum_{j\in\Omega_c} H_{ij}(k) E(y(j)) - \mu(i) \\ &\approx \sum_{j\in\Omega_c} H_{ij}(k)\mu(j) - \mu(i), \end{aligned} \quad (9)$$

where $H_{ij}(k)$ is the element at the i-th row and j-th column in the hat matrix $H(k)$. The second line is exact for unbiased MC ray tracing methods (e.g., path tracing), i.e., $E(y(j)) = \mu(j)$, but it is an approximated value for biased methods, such as photon mapping. In the remaining sections of this work we assume our input image is an unbiased rendering result since the bias correction of an input image is beyond the scope of this paper.

Intuitively, the i-th row entries in the hat matrix can be thought of as reconstruction weights, allocated to neighboring pixels within the reconstruction window Ω_c. As a result the bias of our reconstruction depends on the hat matrix $H(k)$ (in Eq. 5) and the matrix can vary as we change the polynomial order k.

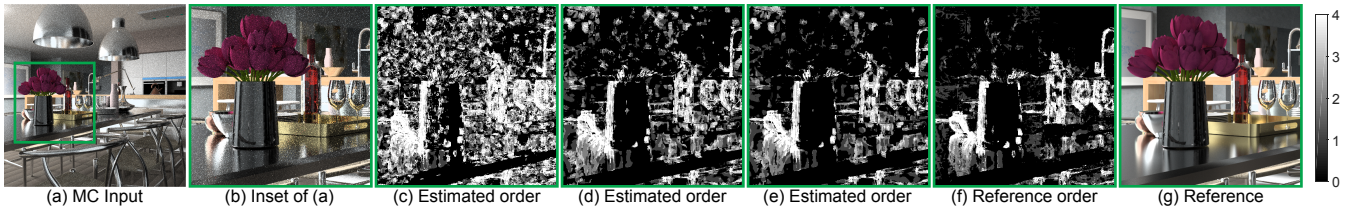
The variance of our reconstruction can be computed analogously:

$$\sigma^2(\hat{y}_k(i)) \approx \sum_{j\in\Omega_c} \left(H_{ij}(k)\right)^2 \sigma^2(y(j)), \quad (10)$$

where the $\sigma^2(y(j))$ is the variance of the pixel intensity $y(j)$. The equation is exactly true only when each pixel intensity is independent from other pixels (e.g., path tracing) as we ignore the covariance terms between intensities. As a result, we can compute our optimal polynomial order k_{opt} using the bias and variance functions:

$$k_{opt} = \underset{k}{\operatorname{argmin}} \sum_{i\in\Omega_c} K_h(i) \left(\left(E\left(\hat{y}_k(i) - \mu(i)\right)\right)^2 + \sigma^2(\hat{y}_k(i))\right). \quad (11)$$

Note that we drop the normalization term $\left(\sum_{i\in\Omega_c} K_h(i)\right)^{-1}$ in Eq. 7 since this value is a constant. The optimal order k_{opt} is still an



(a) MC Input 64 spp | (b) Inset of (a) | (c) Estimated order iteration 1 | (d) Estimated order iteration 2 | (e) Estimated order iteration 3 | (f) Reference order | (g) Reference 32K spp

Figure 3: *Compares our estimated optimal orders with a reference order, which is computed using a reference image generated by 32K spp (g). Given the input image (a) and (b), our method iteratively estimates an optimal polynomial order at each pixel in order to minimize our reconstruction errors, by performing our multi-stage error estimation process. Our estimation results ((d) and (e)) using the multi-stage estimation show a similar pattern to the reference order (f) thanks to our robust error estimation.*

unknown term since the computation (Eq. 11) is using the bias and variance equations which utilize unknown quantities μ and σ^2 that can be obtained only using an infinite number of samples. Hence, we propose an estimation process to estimate the unknown terms in the next section.

4.2 Multi-Stage Error Estimation

Given the bias and variance equations (Eq. 9 and 10), the straightforward way to estimate the unknown intensities (i.e., $\mu(j)$ and $\mu(i)$) and variance $\sigma^2(y(j))$ (in Eq. 9 and 10) is to use the input intensities (i.e., $y(j)$ and $y(i)$) and sample variance $s^2(y(j))$, respectively.

This approach can be an unbiased estimation, but it is typically too noisy and can lead to a noisy selection for polynomial order. To mitigate this problem, we propose a multi-state error estimation that iteratively estimates the unknown terms, $\mu(j)$ and $\mu(i)$, as follows:

$$E_t(\hat{y}_k(i) - \mu(i)) \approx \sum_{j \in \Omega_c} H_{ij}(k)\hat{y}_{t-1}(j) - \hat{y}_{t-1}(i), \qquad (12)$$

where t is an iteration number. As the initial iteration, i.e., $t = 1$, we set $\hat{y}_0(j)$ and $\hat{y}_0(i)$ as the Monte Carlo inputs $y(j)$ and $y(i)$. For the second iteration, we replace the unknown intensities $\mu(j)$ and $\mu(i)$ (in Eq. 9) with our reconstruction results $\hat{y}_1(j)$ and $\hat{y}_1(i)$, which are computed from the first iteration.

Our reconstruction results can have much lower errors than MC input values, and thus our estimation error introduced by $|\mu(j) - \hat{y}_1(j)|$ and $|\mu(i) - \hat{y}_1(i)|$ can also be much lower than the error in the previous step. This process can be repeated until we obtain a stable output.

Analogously, we estimate the variance terms:

$$\sigma_t^2(\hat{y}_k(i)) \approx \sum_{j \in \Omega_c} \left(H_{ij}(k)\right)^2 \hat{\sigma}_{t-1}^2(y(j)), \qquad (13)$$

where $\hat{\sigma}_{t-1}^2(y(j))$ is an estimated variance from the previous iteration. For the first iteration, we use the MC input variance $s^2(y(j))$. For the next iteration, we estimate the variance term when our reconstruction image from the first iteration is generated. Specifically, our estimated standard deviations $\hat{\sigma}_t(y)$ are computed using the hat matrix $H(\hat{k}_{opt})$, i.e., $\hat{\sigma}_t(y) = H(\hat{k}_{opt})s(y)$, where the $\hat{k}_{opt}$ is the estimated optimal order for our reconstruction results.

This process is essentially an estimation process for unknown variances and it also requires an additional method that computes the optimal polynomial order for the variance term. It is a cascading problem as we need to estimate the variances of $\sigma_t^2(y)$ to decide the optimal order. To avoid the recursive problem, we simply use the optimal order computed for our reconstruction result so that the hat matrix with the order can be reused. Fortunately, the standard deviation term $\sigma_t(y)$ locally can have a high correlation with the unknown intensity image μ since the variance is not an actual error but variation of the intensity for our rendering problem.

For example, we locally compute the correlation between the standard deviation term $\sigma(y)$ and μ for the Kitchen scene (Fig. 3) given our filtering window Ω_c. Specifically, we compute $\sigma(y)$ and μ using a reference image and its variance, generated by 8K samples per pixel. The correlation, 0.716, is a high number for the scene and thus our variance estimation can be a high-quality approximation even though it does not require additional processing.

We plug our multi-stage bias and variance estimation into the equation for selecting polynomial order (Eq. 11), and iteratively update the estimated optimal polynomial order. Specifically, we test a set of candidates for the optimal order k_{opt} from 0 to M and then select the estimated optimal order $\hat{k}_{opt}$ that minimizes our reconstruction error. This process is performed at each center pixel c, and produces an optimal polynomial at each pixel.

We validate our error estimation in Fig. 3. To compute the reference order, we directly compute the unknown error $\xi_c(k)$ (Eq. 7) by plugging a reference image generated by 32K samples per pixel, into the unknown image term $\mu(i)$. Note that this oracle is using the actual $L2$ error that cannot be computed in practice. As shown in the figure, our result in the first iteration shows a noisy estimation for the optimal order since these are based on the input intensity and variance. In the second and third iterations, however, our estimation shows a pattern similar in comparison to the reference order. In addition, the third iteration is slightly less noisy than the second, but the respective patterns are visually similar. This observation is useful since an additional iteration requires a reconstruction process with non-negligible overhead. We find that two iterations offer an attractive balance in terms of computational overhead and estimation quality for the tested scenarios.

4.3 Energy-Preserving Outlier Removal

The spike noise, i.e., outlier, is a common phenomenon when a MC renderer renders a scene that includes glossy materials, and the outlier pixels exhibit an excessive intensity (e.g., two order of magnitude higher than the intensities of other pixels). The well-known approach is to reject [Kalantari et al. 2015] or down-weight [Moon et al. 2014] those pixels so that an optimization process for reconstruction can be performed robustly. If we remove outliers as a pre-process, reconstruction can reduce splotch artifacts introduced by spike noise. The disadvantage of the outlier removal is that noticeable energy loss can be introduced in the reconstructed image,

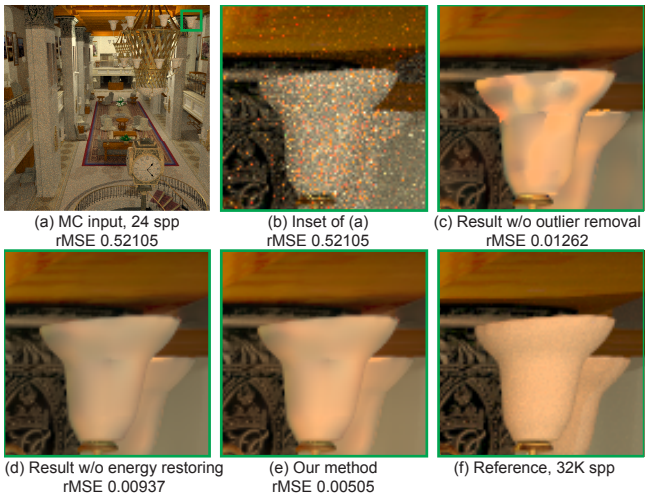


Figure 4: *Validation of our energy-preserving spike noise removal. Given a uniformly generated input image (a) and its inset (b), the result without our outlier removal (c) shows splotch artifacts because it is hard to distribute the excessive energy of outliers. The result using the outlier removal but without our energy restoring (d) is visually pleasing as we can remove the spike noise before our reconstruction, but produces a noticeable energy loss. However, our energy-preserving outlier removal technique (e) reduces the spike noise well and avoids the energy loss problem.*

since the excessive energy is not true error, but an important signal as mentioned in McCool [1999].

To alleviate this problem, we propose an energy-preserving outlier removal technique, inspired by a non-linear filter that introduced the high-level idea of spreading excessive noise into neighboring pixels [Rushmeier and Ward 1994]. We first detect whether the intensity of a center pixel c is an outlier by computing the standard deviation of pixel intensity $y(i)$ within our filtering window Ω_c. Specifically, if the difference between the intensity of the center pixel and the average intensity of all neighboring pixels is three times higher than the computed standard deviation, we replace the center pixel intensity and variance with the pixel that has median intensity and its variance, respectively. We perform this pre-removal step before conducting our reconstruction process. Note that this process is similar to previous methods (e.g., [Kalantari et al. 2015]), but energy loss introduced by the difference between outliers and the median intensity may be noticeable when a rendered input image contains significant spike noise.

Our idea is to restore the energy loss by performing a post-process after our reconstruction. During the outlier removal process, we store the energy loss, i.e., difference e_o between the intensity of the outlier pixel o and median intensity. After our reconstruction, we redistribute this lost energy to our reconstruction output $\hat{y}(i)$ within a large window Ω_o (e.g., 87×87) from the outlier pixel o as the following:

$$\hat{y}(i) = \hat{y}(i) + \rho_o \hat{y}(i), \tag{14}$$

where $\rho_o > 0$ is a compensation factor to distribute the lost energy from the o pixel. We compute the factor by considering our reconstruction output:

$$e_o = \sum_{i \in \Omega_o} \rho_o \hat{y}(i). \tag{15}$$

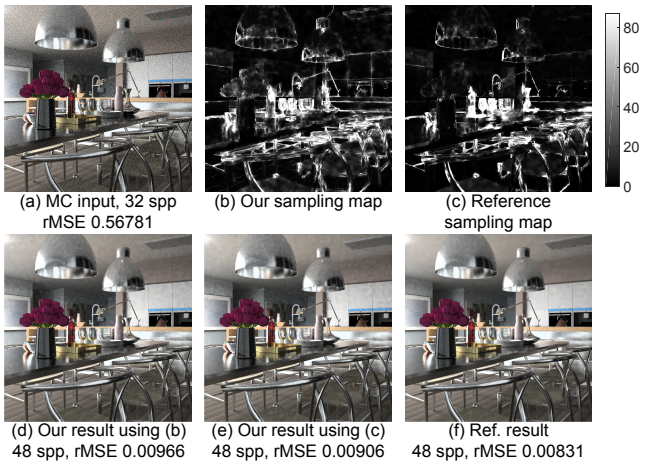


Figure 5: *Comparison between our estimated sampling map (b) and a reference sampling map (c) given an input image (a). We visualize allocated sample counts until a target sample count (e.g., 48 spp) is reached from the given number of samples, i.e., 32 spp. Our sampling map (b) shows a similar pattern compared to the reference sampling map (c), computed using a reference image generated by 32K spp, thanks to our high-quality error estimation. Our result (d) using our sampling map shows a very close result (7% higher error) to the result (e) using the reference sampling map. Also, our error is only 16% higher than the error of the reference method (f) that performs both adaptive sampling and reconstruction based on the actual L2 error computed using the reference image, which cannot be achieved in practice.*

We plug the computed factor into the equation (Eq. 14) for restoring the energy loss. Note that the ratio $\hat{y}(i)/\hat{y}(j)$ between the reconstructed intensities at two pixels i and j within the outlier window Ω_o is maintained after this process is performed.

Fig. 4 shows our method using the proposed technique. When we do not use any outlier removal and restore techniques, the reconstructed image shows the splotch noise. In addition, when the removal technique is used without our energy restoring technique, it reduces the splotch noise well but shows noticeable energy loss. Our proposed technique produces a numerically accurate result compared to the tested simple techniques ((c) and (d) in Fig. 4).

5 Adaptive Polynomial Sampling

In this section, we describe how we further reduce our reconstruction error by adaptively allocating additional ray samples over image regions with high errors. We choose an iterative approach commonly adopted in adaptive rendering methods [Overbeck et al. 2009]. Specifically, we use a uniform sampling at the first iteration given a small number of samples (e.g., 4 or 8 samples per pixel). Then we apply our reconstruction and compute reconstruction error $\hat{\xi}_c(\hat{k}_{opt})$ using our bias and variance estimation. Our final reconstruction output is computed by a local average using the weighting function $K(\cdot)$ (in Eq. 6), and thus we also blend the error using the weighting function $K(\cdot)$ in order to compute the error $\hat{\xi}_i$ of our final output $\hat{y}(i)$ at each pixel i. Given the estimated error per each pixel, we allocate additional samples to further reduce our error. This process is repeated until a user-specified sample budget is reached.

Specifically, we compute the MSE reduction rate $\Delta(\hat{\xi}_i)$ that is an error reduction when allocating an additional sample. To com-

pute this reduction, we use the previously derived reduction rate $n_i^{-4/(d+4)}$ [Moon et al. 2014] given a local dimensionality d and allocated sample count n_i at pixel i. In our framework, we can compute the local dimensionality, i.e., number of independent parameters, using the property of the hat matrix H. Technically, the local dimension can be computed by adding diagonal terms of the hat matrix $H(\hat{k}_{opt})$ that are computed given an estimated optimal polynomial order as the local dimension is the rank of the hat matrix. As a result, our error reduction can be computed as $\Delta(\hat{\xi}_i) = \hat{\xi}_i \times n_i^{-4/(d+4)}$ based on our estimated error $\hat{\xi}_i$. Furthermore, we adopt the relative MSE [Rousselle et al. 2011] that allocates more samples on dark regions, previously used in recent methods (e.g., [Li et al. 2012; Moon et al. 2015]), and thus our relative MSE reduction is computed as $\Delta_{rel}(\hat{\xi}_i) = \Delta(\hat{\xi}_i)/(\hat{y}(i)^2 + \epsilon)$. The epsilon is set to a small value (e.g., 0.001) to avoid dividing by zero. Given the estimated error reduction $\Delta_{rel}(\hat{\xi}_i)$ and a total sample budget, N, for the next iteration, a new sample count at pixel i is calculated as $\Delta n_i = N \times \Delta_{rel}(\hat{\xi}_i)/(\sum_j \Delta_{rel}(\hat{\xi}_j))$ where $\sum_j (\Delta_{rel}(\hat{\xi}_j))$ is a normalization factor that accumulates our relative error over all pixels.

This approach constitutes a well-known iterative approach that previous methods (e.g., [Rousselle et al. 2013]) also use, but the key component is in the robustness of the error estimation $\hat{\xi}_i$. Our multi-state process provides a robust error estimation and our resulting computed sampling patterns are robust. In Fig. 5, we compare our computed sampling map Δn_i with a reference sampling map which computes the sampling count using a reference error $\Delta_{rel}(\xi_i) = \Delta(\xi_i)/(\mu(i)^2 + \epsilon)$. We compute the reference error term ξ_i as an averaged value (similarly in Eq. 6) of the actual L2 error term $\xi_c(k_{opt})$ (in Eq. 7) using a reference image generated by 32K samples per pixel. Our estimated sampling map in the figure shows a similar pattern compared to the reference map and our result using our sampling map shows a very close error (7% higher) to the result using the reference map. In addition, we test a reference method that uses the reference sampling map and also performs an ideal reconstruction that uses a reference order k_{opt} instead of our estimated order $\hat{k}_{opt}$ ((f) in Fig. 5). Our error (0.00966) is slightly higher, i.e., 16%, than the error (0.00831) of the reference result. It indicates that our multi-stage error estimation can provide an effective scheme to adaptively control sampling and reconstruction simultaneously so that we generate high-quality rendering results closed to the reference results that use reference images, which cannot be achieved in practice.

6 Temporal Extension

Our method can be naturally extended to utilize temporal coherence between consecutive frames, by extending the 2D filtering window Ω_c into a 3D window that considers neighboring pixels in adjacent frames. Specifically, we generate noisy input MC images per frame through our adaptive sampling until a user-specified sampling count is reached (Sec. 5), and stack these images to form noisy volume data. We then perform our reconstruction (Sec. 4) on the 3D volume data. In this process, our 2D filtering window (e.g., 29×29 window for Ω_c) is extended to 3D (e.g., $29 \times 29 \times 5$ window for Ω_c). In our accompanying video, we compare the MC input generated by our sampling and our final output for the San Miguel and Hotel Lobby scene. One can still notice some flickering on our temporal results, but our method drastically reduces the flickering of the input sequences while preserving high-frequency edges by performing our polynomial based reconstruction. To further reduce the existing flickering, one interesting direction for future work would adaptively distribute additional samples both spatially and temporally over the 3D volume space. Specifically, our multi-stage error analysis has potential to guide the additional sampling so that new samples can be adaptively allocated to regions containing high error.

7 Implementation Details

We have plugged our adaptive techniques into the general rendering framework, pbrt [Pharr and Humphreys 2010] so that our sampling and reconstruction can performed within the framework, and implemented our reconstruction by using CUDA for parallelizing our optimization for optimal polynomial orders. For the filtering window, Ω_c, we have used a 29×29 window, and tested $(0, 1, 2, 3)$ as the candidates of the optimal order k_{opt}. Given the filtering window, we include statistically equivalent pixels as neighboring pixels, which is commonly used in previous methods (e.g., [Sen and Darabi 2012]). Specifically, we utilize the confidence interval for two independent samples [Hayter 2007] based on the normality assumption of the distribution of each pixel intensity, i.e., $(y(i) - y(c)) \pm 3.0\sqrt{s^2(y(i)) + s^2(y(c))}$. We reject the pixels when the difference between their intensities and the intensity of a center pixel, $(y(i) - y(c))$, is outside of the interval. Note that this interval goes to zero as the number of samples goes to an infinite number, and thus it provides the consistency property of a reconstruction method, as described in Moon et al. [2013].

Optimization using Sparse Polynomials. Our reconstruction can be applied to only a small number of center pixels instead of all pixels individually in order to reduce the computational burden of our reconstruction. This is similar to the process utilized previously when performing reconstruction using sparse linear models [Moon et al. 2015]. Our optimization is quite straight-forward since we do not change the size of reconstruction window. Note that our method only changes the polynomial order. As a result, we can apply a regular sampling process to select the center pixels. Specifically, we regularly choose center pixels that have x and y positions that are multiples of the half size of our filtering window (e.g., 14 for our 29×29 window). This greatly reduces computational burden since we can perform our reconstruction at only a small number of center pixels; an analysis is included in our supplementary material.

Pre-filtering Features. Our reconstruction uses the geometric function $g(f_i)$ that utilizes texture, normal, and depth information and thus this function can be noisy when a depth of field or motion blur are simulated. Our reconstruction framework (Sec. 3) can naturally pre-filter this function using our image polynomial function $p(i)$. Specifically, we run our reconstruction for each feature type using the feature buffer and its variance, instead of the MC input $y(i)$ and its variance $\sigma^2(y(i))$. In addition, we set $g(f_i)$ as a null function since we do not have an additional edge function for the feature function itself. However, we can provide a high-quality pre-filtering through our adaptive image polynomial function $p(i)$ in a data-driven way. In practice, the noise contained in the feature vector can be much smaller than the noise in MC input images. Hence, we set the filtering window size Ω_c to be small, 5×5, in order to reduce computational overhead. This process can be performed efficiently since the hat matrix H can be precomputed and shared among all pixels during our optimization. Technically, the matrix is identical across pixels since we do not use the geometric function $g(f_i)$ for this pre-filtering.

8 Results and Discussions

We have tested our method on a Windows machine with a i7-3770 CPU and GTX 780 graphics card. We have compared our approach with the state-of-the-art rendering methods using learning-based filtering (LBF) [Kalantari et al. 2015], weighted local re-

Figure 6: *Equal-time comparisons of adaptive rendering methods. The test scenes are challenging in nature, containing diverse characteristics. Our method produces visually and numerically better results in comparison to state-of-the-art methods LBF, WLR, and ALP. Our performance improvements are achieved by performing a new adaptive sampling and reconstruction using adaptive polynomial functions, even without filtering bandwidth optimization that previous methods employ. Our supplementary report includes full-resolution images of the tested methods and an additional test for a bump-mapped scene.*

gression (WLR) [Moon et al. 2014], and adaptive linear prediction (ALP) based method [Moon et al. 2015], which demonstrated outstanding rendering results through the bandwidth optimization process. To test the previous methods, we have used the implementation provided by the authors. In addition, we have tested the low discrepancy (LD) sampling that uses a uniform sampling and does not perform any filtering process. As a numerical measure, we have tested the relative MSE [Rousselle et al. 2011] which is widely used to evaluate numerical accuracy of previous methods.

Benchmarks. We have validated our method for challenging scenes: 1) Hotel Lobby 2) San Miguel 3) Pool 4) Kitchen. For image resolutions, we use 1K $\times$ 1K images. The Hotel Lobby scene (top row in Fig. 6) consists of complex geometries, and the objects with gold materials (in the ceiling) introduce large quantities of spike noise in the rendered images. This is challenging as handling spike noise without energy loss can be a crucial performance factor for this scene. The San Miguel (second row in Fig. 6) is a widely tested scene for recent adaptive methods due to its geometric complexity. In addition, we simulate a strong depth-of-field effect that introduces a severe noise in the feature vector such as normal, texture, and depth, which tested methods including our technique utilize. For the Pool scene (third row in Fig. 6), we simulate a strong motion blur effect on the noisy textured floor. This motion blur effect introduces noise in feature buffers such as normal, texture, and depth. Lastly, the Kitchen (bottom row in Fig. 6) is a glossy-dominant scene that generates many high-frequency edges introduced by strong glossy reflections. The tested methods, including our own, utilize geometries as features, but we cannot easily identify the reflected edges on glossy materials using only geometries.

Equal-time Comparisons. In Fig. 6, we test equal-time comparisons for the compared adaptive rendering methods. For a fair comparison, we use 32 or 64 samples per pixel for LBF since the previous method pre-computes their training sets using a power of two sample counts, as described in the previous paper [Kalantari et al. 2015]. Given the Hotel Lobby scene (top row), LBF and WLR show splotch artifacts and ALP does not preserve the energy, since these methods cannot distribute the excessive energy of spike noise well. However, our method removes the spike noise well without energy loss thanks to our energy-preserving removal (Sec. 4.3). LBF shows a smooth result for the San Miguel scene (second row) compared to WLR and ALP, since the estimated optimal parameters of LBF use a training set with reference images that can prove more robust than the direct estimation of MSE using only a MC

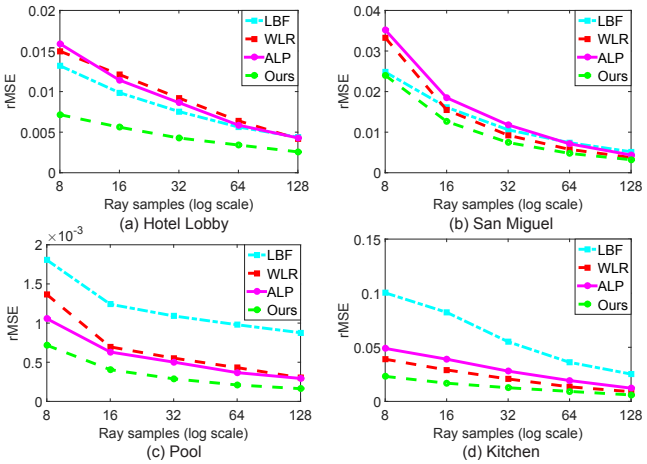


Figure 7: *MSE convergence graph. We measure the numerical error of our method and compare with the state-of-the-art methods.*

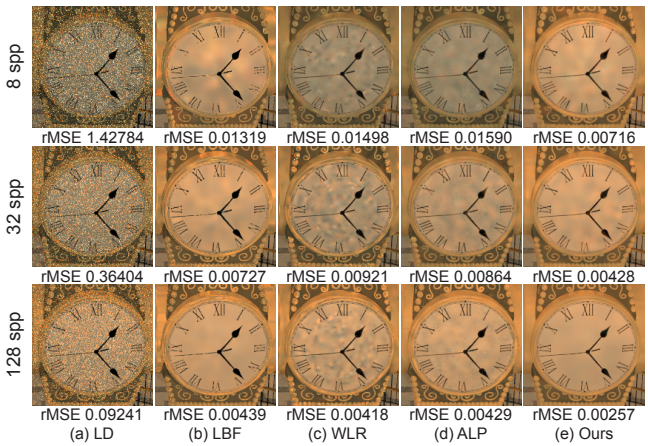


Figure 8: *Comparisons of the tested methods with varying sample counts, given the clock face in the Hotel scene. Our method shows consistently better results compared to other methods, while decreasing reconstruction errors, e.g., low-frequency noise.*

input image. Our proposed block-based error estimation improves the robustness of the optimal order selection, and leads to a visually pleasing image as well as numerically better results when compared to other methods. For the Pool scene (third row), LBF tends to produce noisy results on the motion blurred area, while WLR and ALP show over-blurred results. Our method, however, preserves the detailed edges introduced by the motion blurring thanks to our adaptive polynomial based pre-filtering for features such as texture. Given the glossy dominant scene (bottom row), LBF shows a high numerical error due to over-blurred edges (e.g., glossy highlights) as it can be difficult to train for these complex edges through a learning based pre-processing step. As a result of our novel bias-variance compensation, through adaptive polynomial selection, our method produces numerically better results even though we do not make use of any processing for learning or optimization of filtering bandwidths.

MSE Convergence. We compute the rMSE of each method while varying the average number of samples per pixel for all tested scenes in Fig. 7. The machine learning based method, LBF, produces high-quality results for small numbers of samples compared to WLR and ALP as error estimation with small sample counts can

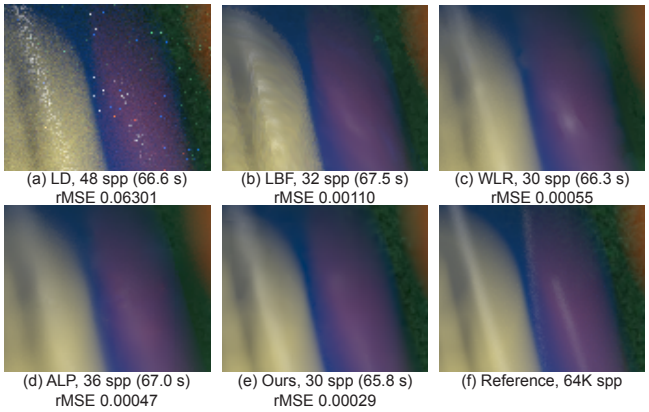


Figure 9: *Failure case of our method involving motion blur (from the Pool scene). The tested methods, including our technique, do not capture the detailed edges introduced by moving highlights on a glossy object, since MC input images do not contain enough information to allow reconstruction techniques to preserve the edges (see (a)).*

be fundamentally difficult (e.g., for Hotel Lobby and San Miguel). However, our method and linear regression techniques (i.e., WLR and ALP) show a faster convergence than LBF as we increase the number of samples, because MSE estimation process using MC input images is able to become more accurate when a number of samples increases. In the Fig. 8, we also compare the visual quality of all tested methods that use a different number of samples for the Hotel scene. Given our tested scenes and sampling counts, our results show consistently better results over the previous methods even without the optimal bandwidth selection, and this indicates that our key idea, adaptive polynomial selection, can be a new complementary approach for a high-quality adaptive method.

Limitations. Our adaptive sampling is an image-space method that adaptively controls sampling rate over pixels. For higher dimensions such as lens, time, and second bounces, our method, like other image-space methods, uses a random (or low-discrepancy) sampling. Our approach is intrinsically simple, but adaptively controlling other spaces can be important. For example, the moving highlights in the Pool scene (Fig. 9) clearly demonstrate the limitation of image-space methods, including our method. None of the tested methods preserve moving highlights properly. It is fundamentally challenging to reconstruct high-frequency edges when the input images do not have enough information ((a) of Fig. 9). Note that even the reference image, with the exhaustive sampling count, has noise in this case. For this example, adaptively controlling sampling counts in space time can be important when attempting to capture the edges. Increasing the sampling dimension of our method, whilst minimizing accounting for the curse-of-dimensionality effect, will prove a challenging and important future work.

9 Conclusions

In this paper, we propose a new adaptive technique for estimating the optimal balance between the bias and variance of reconstruction methods and demonstrate that high-quality rendering results can be achieved even without the need for an optimal bandwidth selection process, a common strategy for recent methods. We present a block-based optimization process and multi-stage estimation for robustly computing optimal polynomials locally for denoising very noisy input images appropriately. In addition, an energy-preserving denoising technique for spike noise is introduced so that

outliers can be removed without causing noticeable energy loss. Our method shows a novel error control process for image-space adaptive methods, but we believe that our method through adaptive order selection can be further optimized by combining previous bandwidth optimization techniques. We would like to investigate this direction as a future work.

Acknowledgements

We appreciate the reviewers for their constructive comments. The San Miguel and Pool scenes are courtesy of Guillermo M. Leal Llaguno and Toshiya Hachisuka, respectively. Also, the Hotel Lobby (modeled by bluewhalestudios) and Kitchen scenes (designed by Nodexis) are from TurboSquid (www.turbosquid.com). This work was funded by Innovate UK project #101858.

References

DELBRACIO, M., MUSÉ, P., BUADES, A., CHAUVIER, J., PHELPS, N., AND MOREL, J.-M. 2014. Boosting Monte Carlo rendering by ray histogram fusion. *ACM Trans. Graph. 33*, 1, 8:1–8:15.

DURAND, F., HOLZSCHUCH, N., SOLER, C., CHAN, E., AND SILLION, F. X. 2005. A frequency analysis of light transport. *ACM Trans. Graph. 24*, 3, 1115–1126.

EGAN, K., TSENG, Y.-T., HOLZSCHUCH, N., DURAND, F., AND RAMAMOORTHI, R. 2009. Frequency analysis and sheared reconstruction for rendering motion blur. *ACM Trans. Graph. 28*, 3, 93:1–93:13.

EGAN, K., DURAND, F., AND RAMAMOORTHI, R. 2011. Practical filtering for efficient ray-traced directional occlusion. *ACM Trans. Graph. 30*, 6, 180:1–180:10.

EGAN, K., HECHT, F., DURAND, F., AND RAMAMOORTHI, R. 2011. Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Trans. Graph. 30*, 2, 9:1–9:13.

HACHISUKA, T., JAROSZ, W., WEISTROFFER, R. P., DALE, K., HUMPHREYS, G., ZWICKER, M., AND JENSEN, H. W. 2008. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph. 27*, 3, 33:1–33:10.

HAYTER, A. 2007. *Probability and statistics for engineers and scientists 3rd*. Brooks/Cole Publishing Co.

KAJIYA, J. T. 1986. The rendering equation. In *ACM SIGGRAPH '86*, 143–150.

KALANTARI, N. K., AND SEN, P. 2013. Removing the noise in Monte Carlo rendering with general image denoising algorithms. *Computer Graphics Forum 32*, 2pt1, 93–102.

KALANTARI, N. K., BAKO, S., AND SEN, P. 2015. A machine learning approach for filtering Monte Carlo noise. *ACM Trans. Graph. 34*, 4, 122:1–122:12.

LEHTINEN, J., AILA, T., CHEN, J., LAINE, S., AND DURAND, F. 2011. Temporal light field reconstruction for rendering distribution effects. *ACM Trans. Graph. 30*, 4, 55:1–55:12.

LEHTINEN, J., AILA, T., LAINE, S., AND DURAND, F. 2012. Reconstructing the indirect light field for global illumination. *ACM Trans. Graph. 31*, 4, 51:1–51:10.

LI, T.-M., WU, Y.-T., AND CHUANG, Y.-Y. 2012. SURE-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph. 31*, 6, 194:1–194:9.

MCCOOL, M. D. 1999. Anisotropic diffusion for Monte Carlo noise reduction. *ACM Trans. Graph. 18*, 2, 171–194.

MEHTA, S. U., WANG, B., AND RAMAMOORTHI, R. 2012. Axis-aligned filtering for interactive sampled soft shadows. *ACM Trans. Graph. 31*, 6, 163:1–163:10.

MEHTA, S. U., WANG, B., RAMAMOORTHI, R., AND DURAND, F. 2013. Axis-aligned filtering for interactive physically-based diffuse indirect lighting. *ACM Trans. Graph. 32*, 4, 96:1–96:12.

MEHTA, S. U., YAO, J., RAMAMOORTHI, R., AND DURAND, F. 2014. Factored axis-aligned filtering for rendering multiple distribution effects. *ACM Trans. Graph. 33*, 4, 57:1–57:12.

MOON, B., JUN, J. Y., LEE, J., KIM, K., HACHISUKA, T., AND YOON, S.-E. 2013. Robust image denoising using a virtual flash image for Monte Carlo ray tracing. *Computer Graphics Forum 32*, 1, 139–151.

MOON, B., CARR, N., AND YOON, S.-E. 2014. Adaptive rendering based on weighted local regression. *ACM Trans. Graph. 33*, 5, 170.

MOON, B., IGLESIAS-GUITIAN, J. A., YOON, S.-E., AND MITCHELL, K. 2015. Adaptive rendering with linear predictions. *ACM Trans. Graph. 34*, 4, 121:1–121:11.

OVERBECK, R. S., DONNER, C., AND RAMAMOORTHI, R. 2009. Adaptive wavelet rendering. *ACM Trans. Graph. 28*, 5, 140:1–140:12.

PHARR, M., AND HUMPHREYS, G. 2010. *Physically Based Rendering: From Theory to Implementation 2nd*. Morgan Kaufmann Publishers Inc.

ROUSSELLE, F., KNAUS, C., AND ZWICKER, M. 2011. Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph. 30*, 6, 159:1–159:12.

ROUSSELLE, F., KNAUS, C., AND ZWICKER, M. 2012. Adaptive rendering with non-local means filtering. *ACM Trans. Graph. 31*, 6, 195:1–195:11.

ROUSSELLE, F., MANZI, M., AND ZWICKER, M. 2013. Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7, 121–130.

RUPPERT, D., AND WAND, M. 1994. Multivariate locally weighted least squares regression. *The annals of statistics 22*, 3, 1346–1370.

RUSHMEIER, H. E., AND WARD, G. J. 1994. Energy preserving non-linear filters. In *ACM SIGGRAPH*, 131–138.

SEN, P., AND DARABI, S. 2012. On filtering the noise from the random parameters in Monte Carlo rendering. *ACM Trans. Graph. 31*, 3, 18:1–18:15.

SOLER, C., SUBR, K., DURAND, F., HOLZSCHUCH, N., AND SILLION, F. 2009. Fourier depth of field. *ACM Trans. Graph. 28*, 2, 18:1–18:12.

YAN, L.-Q., MEHTA, S. U., RAMAMOORTHI, R., AND DURAND, F. 2015. Fast 4d sheared filtering for interactive rendering of distribution effects. *ACM Trans. Graph. 35*, 1, 7:1–7:13.

ZWICKER, M., JAROSZ, W., LEHTINEN, J., MOON, B., RAMAMOORTHI, R., ROUSSELLE, F., SEN, P., SOLER, C., AND YOON, S.-E. 2015. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum 34*, 2, 667–681.