Robust Image Denoising using a Virtual Flash Image for Monte Carlo Ray Tracing

Bochang Moon[1] Jong Yun Jun[1] JongHyeob Lee[1] Kunho Kim[1] Toshiya Hachisuka[2,3] and Sung-Eui Yoon[1]

[1]KAIST [2]UC San Diego [3]Aarhus University

Abstract

We propose an efficient and robust image-space denoising method for noisy images generated by Monte Carlo ray tracing methods. Our method is based on two new concepts: virtual flash images and homogeneous pixels. Inspired by recent developments in flash photography, virtual flash images emulate photographs taken with a flash, to capture various features of rendered images without taking additional samples. Using a virtual flash image as an edge-stopping function, our method can preserve image features that were not captured well only by existing edge-stopping functions such as normals and depth values. While denoising each pixel, we consider only homogeneous pixels – pixels that are statistically equivalent to each other. This makes it possible to define a stochastic error bound of our method, and this bound goes to zero as the number of ray samples goes to infinity, irrespective of denoising parameters. To highlight the benefits of our method, we apply our method to two Monte Carlo ray tracing methods, photon mapping and path tracing, with various input scenes. We demonstrate that using virtual flash images and homogeneous pixels with a standard denoising method outperforms state-of-the-art image-space denoising methods.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

Achieving the realistic rendering of various models under complex illumination has been one of the main goals in computer graphics. Although many different rendering algorithms exist, Monte Carlo ray tracing methods such as path tracing [Kaj86] and photon mapping [Jen01] have been the most popular algorithms for producing high quality images under various scene configurations. One problem, however, is that Monte Carlo ray tracing algorithms tend to require a large number of ray samples in order to generate visually pleasing results. If there are not enough samples, rendered images can have a significant amount of noise (e.g., Fig. 1-(b)). Therefore, reducing noise in Monte Carlo ray tracing methods has been actively investigated in the field of computer graphics.

A popular approach to noise reduction is to directly work on the rendered images. However, it is commonly considered that this image-based approach can often result in excessive blurring or insufficient denoising. This is because distinguishing image features from noise in noisy images is fundamentally difficult. Moreover, the image-based approach tends to have many user-defined parameters that significantly affect the quality of denoising. Thus, existing image-based denoising methods often require tedious parameter tuning in order to obtain visually pleasing denoised images with a particular scene configuration. This has significantly limited the applicability of these methods in practical applications.

In this paper, we present a novel concept for image denoising methods, *virtual flash images*, which serve as edge-stopping functions to discern image features from noise. A virtual flash image is created using a subset of the original light paths (e.g., direct illumination) with an additional point light that emulates a camera flash.

The key advantage of using the virtual flash image is that it contains various image features that cannot be captured solely from geometric information such as depth values and normals (i.e., G-buffers). For example, reflective and refractive features of a transparent shower booth (the first row of Fig. 1) are well captured by the virtual flash image. This

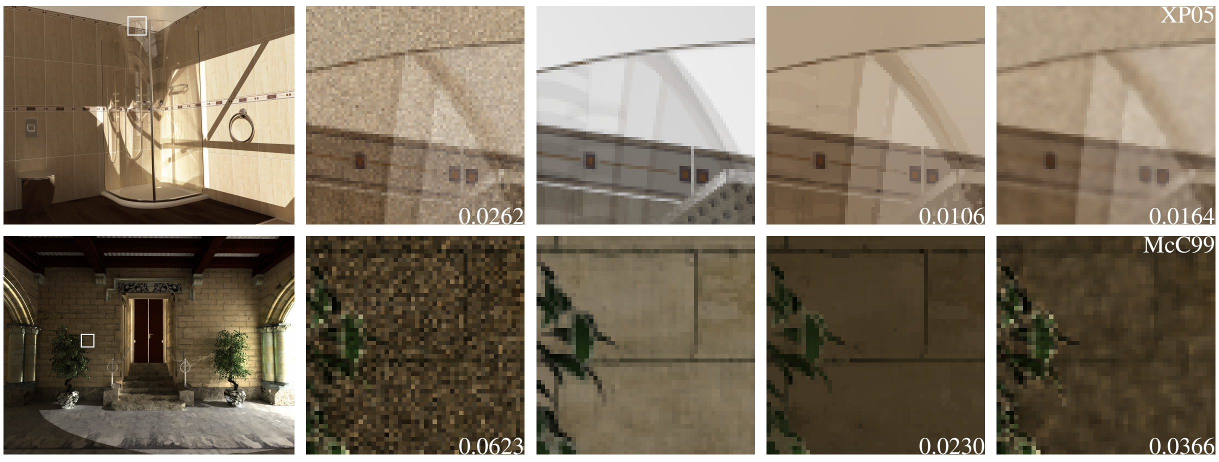


Figure 1: *Results of our denoising method with 64 ray samples per pixel. Our method takes input images (column (b)) and significantly reduces noise level while keeping salient image features (columns (a) and (d)). The key idea is the use of virtual flash images (column (c)), which capture various image features without additional samples. Our method achieves not only visually better results, but also numerically more accurate results than the prior image-space denoising methods (column (e)). The numbers at the lower right corners the root mean square (RMS) errors computed from the reference solutions.*

concept is motivated by recent advances in flash photography [PSA*04, ED04], but we present necessary modifications to the original idea in order to make it useful for our denoising method.

In order to further improve the robustness of the denoising process, we introduce *homogeneous pixels*, which we define pixels that are statistically equivalent to each other. We consider only these homogeneous pixels during the denoising process. This concept makes it possible to define a stochastic error bound of our denoising method, irrespective of denoising parameters. Furthermore, the stochastic bound reduces by the order of $n_p{}^{-1/2}$, where n_p is the number of ray samples for a pixel. This makes our denoising robust from tweaking denoising parameters for various rendering effects and different ray sample numbers. Our method can reduce RMS errors by a factor of more than one order of magnitude compared to generating more ray samples without using the denoising process.

We have applied our denoising methods to the results of path tracing and photon mapping on a variety of realistic and challenging scene configurations. Compared to existing image-based denoising techniques, our method achieves more significant noise reduction while better preserving various image features. Fig. 1 highlights the results of our method. The supplementary video also shows that our method can be naturally extended to denoise animations without noticeable flickering.

2. Related Work

2.1. Noise Reduction for Rendering

Reducing noise in images generated by Monte Carlo ray tracing methods has been one of the main challenges in rendering. Existing techniques have investigated noise reduction at two different stages of the Monte Carlo ray tracing process: sampling and reconstruction. In this section we review previous work only on the reconstruction stage, since our work focuses on the reconstruction process and is orthogonal to sampling methods.

Image-space methods: Rushmeier and Ward [RW94] proposed an energy preserving non-linear filter to spread energy of outlier samples of a pixel into its neighboring pixels. McCool [McC99] and Xu et al. [XP05] applied the anisotropic diffusion and bilateral filtering respectively for reducing noise in Monte Carlo ray tracing methods. These techniques require rather careful parameter setting for achieving high-quality denoising results. Recently, Overbeck et al. [ODR09] proposed a wavelet-based reconstruction technique. The wavelet-based reconstruction technique works quite well with their proposed wavelet-based sampling. Unfortunately, their approach, like other wavelet-based methods, can produce image artifacts such as edge ringing.

Geometric features have been widely used for denoising images rendered with global illumination methods [McC99, LSK*07, DSHL10, BEM11]. However, edges introduced by non-geometric features (e.g., reflection and refraction) can be overly smoothed since these edges are not well captured by geometric features.

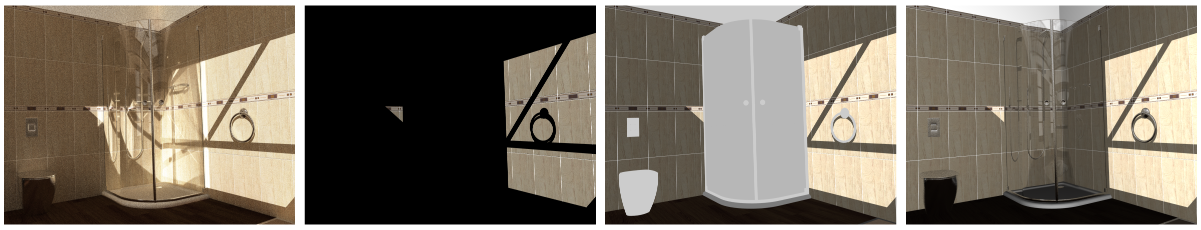


Figure 2: *An input noisy image (generated with 16 ray samples) for the shower booth scene, where indirect illuminations are dominant. The virtual flash image (d) captures more high-frequency features compared to images with the direct illumination (b) and the additional ambient term (c).*

Sen and Darabi [SD12] used functional relations among Monte Carlo ray samples for computing a filtering weight of each ray sample. This technique has a potential to create high-quality denoised results and can be combined with homogeneous pixels in order to determine filtering weights that better preserve various image features. Our virtual flash images can also be integrated as a new feature vector in their method.

DeCoro et al. [DWR10] and Pajot et al. [PBP11] proposed a rejection method for removing outliers introduced by Monte Carlo ray tracing. This method can be combined as a pre-processing to our method, especially when a small number of ray samples are used.

Object-space methods: This class of methods reconstructs an image from samples in an object or high dimensional sampling space. One of the well-known examples is irradiance caching [WRC88]. Another recent example is multi-dimensional adaptive sampling [HJW*08]. Such techniques may generate better results than simpler image-based methods that we reviewed above, since samples in an object space (or high dimensional sampling space) tend to have more information than their counterparts in the image space. However, adapting these techniques to an existing rendering system may require major modifications to various parts of the rendering system.

2.2. Noise Reduction for Photographs

In the field of image processing, commonly used techniques adopt an edge-preserving filter for denoising photographs. Well-known filters include anisotropic diffusion [PM90] and bilateral filtering [TM98]. Wavelet-based methods denoise images by thresholding the wavelet coefficients [DJ95]. Wavelet-based methods, however, can produce distracting image artifacts such as low-frequency noise and edge ringing caused by underlying wavelet basis. Unfortunately, direct applications of such techniques to denoise rendered images have shown sub-optimal results, since they do not utilize various information available during the rendering process.

Image enhancement by flash photography: Eisemann and Durand [ED04] and Petschnigg et al. [PSA*04] designed an effective denoising method for photographs taken in dark environments, by utilizing additional photographs taken with a camera flash. They extended bilateral filtering into a cross (or joint) bilateral filtering that considers pairs of flash and non-flash images. The key observation of these methods is that the flash image is relatively sharp and less noisy compared to its corresponding non-flash image. Therefore, they could use the flash image as an estimator of the high-frequency content of the non-flash image. Inspired by these techniques, virtual flash images are designed for denoising rendered images while preserving various image features generated by Monte Carlo ray tracing methods.

3. Our Method

Our goal is to denoise images generated by various Monte Carlo ray tracing algorithms. Since rendered images can have image features buried under noise, naïve applications of filtering techniques proposed in the image processing field may not provide satisfactory results. For example, Xu and Pattanaik [XP05] pointed out that a naïve usage of bilateral filtering to noisy rendered images works poorly. In order to address this issue, we propose to use a *virtual flash image* that serves as an estimator of image features in the input noisy image. Examples of virtual flash images are shown in Fig. 1.

3.1. Generating Virtual Flash Images

We aim to discern all the image features from noise in the input image without taking additional ray samples. To achieve this goal, a virtual flash image is constructed by summing two different components: 1) a part of the illumination from the original light sources, and 2) additional illumination from a virtual flash point light located at the viewing position.

To create the virtual flash image, we reuse subsets of light paths (and their shading results) that are generated by the original rendering method with the original light sources. Among the light paths of the original light sources, we keep all the ray paths that interact with specular and glossy materials to compute the virtual flash image. On the other hand, for ray paths interacting with diffuse materials, we keep only the ray paths that have at most one diffuse bounce from the viewpoint (e.g., direct illumination). This is because shading for multiple diffuse bounces causes significant noise in the computed illumination values. This limited set of ray paths can capture most high-frequency image features (e.g., textures, shadow boundaries with or without reflections and refractions) in the virtual flash image. However, since we ignore some ray paths, some features (e.g., caustics in Fig. 2-(a)) cannot be captured in the virtual flash image. We explain how to robustly detect such missing features during our denoising process in a later section.

The additional virtual flash light source is crucial for capturing image features in regions where the original lights do not cast direct illumination, as shown in Fig. 2. The conventional method for approximating indirect illumination uses an approximate ambient term [RPG99], but this approach can only capture a limited set of high-frequency features (e.g., textures) introduced by primary rays. The virtual flash light ensures the virtual flash images capture a larger variety of high-frequency features (e.g., edges on specular surfaces) through existing shading codes compared to the conventional method (Fig. 2). Additionally, the virtual flash images with considering the additional virtual flash light can be constructed quickly because they do not require any additional ray samples. Specifically, we do not perform any visibility tests nor create shadows from the virtual flash light, to avoid adding shadows that do not exist in the input image.

We set the intensity of the virtual flash light such that it can compensate the loss of indirect illumination in the virtual flash image; this can be done quite efficiently by comparing the original image and the virtual flash image rendered without the virtual flash light. We found that this simple heuristic works very well. Furthermore, we found that a wide range of values that deviate from the intensity computed by the heuristic also work well, and robustly produce almost identical denoising results, as shown in Fig. 3.

Existing image-based denoising methods [McC99,DSHL10, BEM11, SD12] use geometric features (e.g., normals and depths stored in the G-buffer) as edge-stopping functions. Albedo (e.g., colors) can also be considered for preserving texture edges. Unfortunately, reflected and refracted edges of transparent objects (shown in Fig. 4) cannot be preserved because they are not simply created by the geometries and albedo of transparent objects. In addition, the correct estimation of the geometric information of blurred regions generated by defocus or motion blur effects is not trivial; averaging depths and normals across edges may give incorrect geometric values as pointed out by Bauszat et al. [BEM11].

The key concept of using virtual flash images is that, although limited, we combine geometric features and albedo of surfaces intersected by multiple rays into a single image through actual shading. This seemingly simple concept adds a powerful capability of capturing discontinuities with non-diffuse objects such as a glass shown in Fig. 1.

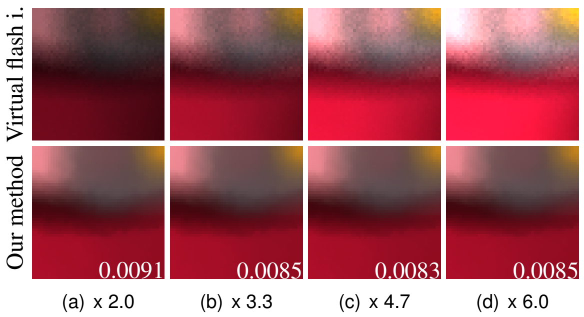


Figure 3: *The first row shows virtual flash images, as we change the intensity of the virtual flash light from 2 to 6 times over the intensity of the original light in the toaster scene. The denoised results with different virtual flash images are shown in the second row. The virtual flash light intensity, 4.7 times to the original light, generated by our method produces the lowest RMS error.*

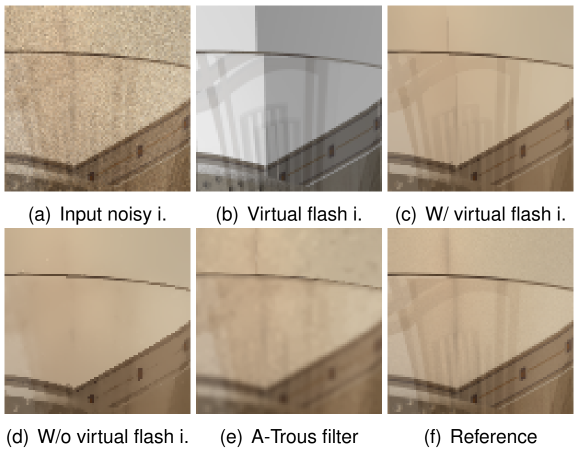


Figure 4: *Our denoising results w/ and w/o considering the virtual flash image within our non-local means filtering. Considering the virtual flash image, we can preserve fine details that are captured in the virtual flash image. A-Trous filter [DSHL10], which is based on geometric information, fails to preserve both refracted and reflected edges on the glass. The input and reference image are generated with 64 and 2,500 ray samples per pixel.*

3.2. Denoising using a Virtual Flash Image

Given a Monte Carlo ray tracing method that generates an input noisy image N, we denoise the input image to obtain an output image D. Using a virtual flash image F and non-local means filtering [BCM05] (i.e., a patch-based bilateral filtering), the output D_p of pixel p is computed as follows:

$$D_p = \frac{1}{k(p)} \sum_{p' \in H(\Omega_p)} g_s(|p' - p|) g_r\left(\left|u(F_{p'}) - u(F_p)\right|\right) N_{p'}, \tag{1}$$

where $k(p)$ is a normalization term, the function $u(F_p)$ denotes a patch of pixels around p in the virtual flash image, and Ω_p is defined as the pixels in a square denoising window centered on pixel p. g_s and g_r are spatial and range Gaussian filters that have standard deviations of σ_s and σ_r, respectively. These two filters serve as edge-stopping functions; the range filter computes weights based on the intensity difference of pixels, whereas the spatial filter sets weights based on the spatial distance of pixels. Note that our range filter computes weights using the virtual flash image F, which has much reduced noise compared to the input image N.

The patch of a pixel p, $u(F_p)$, in our denoising framework (Eq. 1) is defined as an m by m window whose center is located at pixel p. In order to compute the distance between the two patches $u(F_p)$ and $u(F_{p'})$ in the range filter, we compute a weighted Euclidean distance among all the corresponding pixels in the two patches, as suggested by Buades et al. [BCM05].

By considering a virtual flash image within our denoising framework, we can preserve most image features during the denoising process; denoising results with and without considering our virtual flash image are demonstrated in Fig. 4. However, there are some other features (e.g., caustics, color bleeding) that are not captured well by the virtual flash image. Such features can be undesirably removed, especially when we use a large denoising window. It is, however, a challenging problem to set the optimal denoising window size that simultaneously preserves image features and reduces noise [XP05]. Therefore, existing image-based denoising methods simply leave the user to find such a window size by trial and error, which can easily lead to over-blurred or under-smoothed images [KB06]. Instead of Ω_p, we propose to use $H(\Omega_p)$, a set of homogeneous pixels, for preserving those edges (e.g., caustics) that the virtual flash image does not include.

3.3. Robust Denoising with Homogeneous Pixels

Instead of seeking the optimal denoising window size, we propose a simple, yet effective, approach to suppress excessive blurring. Our approach is to identify *homogeneous pixels*, pixels that are considered statistically equivalent based on confidence intervals of the true means of pixels.

Given a pixel p, we define n_p, $\bar{x}_p$, and s_p, to indicate the ray sample count for the pixel, the sample mean, and the standard deviation computed with the observed ray samples for that pixel, respectively. We use μ_p to indicate the true mean of samples for the pixel p. We adopt a large denoising window with a large spatial Gaussian in order to increase the probability of identifying pixels that are correlated with p, and thus achieving better denoising quality. The key is to use only homogeneous pixels ($H(\Omega_p)$ in Eq. 1), from all of those in Ω_p to smooth out pixel p.

We define homogeneous pixels $H(\Omega_p)$ as a set of pixels whose sample means are within a confidence interval of the unknown true mean μ_p. We use the t-distribution to construct the confidence interval for the unknown true mean μ_p with a confidence level of $1-\alpha$ [Hay07].

As a result, the homogeneous pixels $H(\Omega_p)$ of a given pixel p are defined as:

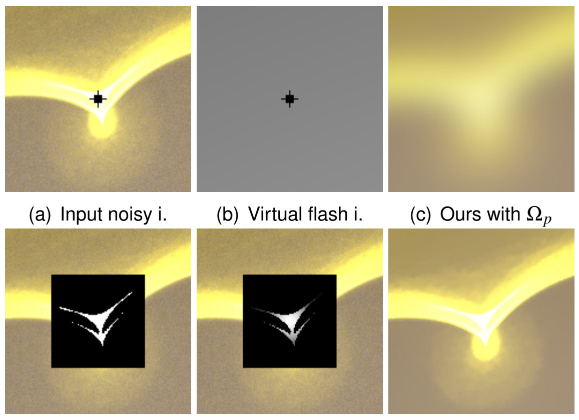
$$\left\{ p' | p' \in \Omega_p, \bar{x}_p - \frac{t_{\alpha/2, n_p - 1} s_p}{\sqrt{n_p}} \leq \bar{x}_{p'} \leq \bar{x}_p + \frac{t_{\alpha/2, n_p - 1} s_p}{\sqrt{n_p}} \right\}. \tag{2}$$

In the above equation, $t_{\alpha/2,n-1}$ is a *critical point* in a two-sided t-interval, which is defined as $P(X \geq t_{\alpha/2,n-1}) = \alpha/2$ where X is a random variable drawn from the t-distribution with $n-1$ degrees of freedom. In practice, the use of a precomputed lookup table for the critical point is recommended for efficient computation [Hay07]. The size of the table is negligible, as the definition of the critical point only depends on the number of samples and the user-specified value α.

This definition of homogeneous pixels is closely related to testing the null hypothesis $H_0 : \mu_p = \mu_{p0}$ against the alternative hypothesis $H_A : \mu_p \neq \mu_{p0}$, where μ_{p0} denotes a specified value that we want to test. For the specified value, we use the neighboring sample mean $\bar{x}_{p'}$. The null hypothesis is then accepted if $|t| \leq t_{\alpha/2,n_p-1}$, where the test statistic $t = (\bar{x}_p - \bar{x}_{p'})/(s_p/\sqrt{n_p})$. The acceptance of the null hypothesis indicates that the tested value $\bar{x}_{p'}$ (i.e., μ_{p0}) is a plausible value of the unknown mean μ_p.

We found that the commonly adopted confidence level of 99% (i.e., $1-\alpha = 0.99$) works very well in practice. An example of homogeneous pixels and their weights is shown in Fig. 5. Note that the virtual flash image does not have any information about the caustics; this example is a zoomed-in inset of the leftmost caustics shown in the wedding-band scene (the last row of Fig. 10). Owing to homogeneous pixels, our method successfully preserves such features.

To define the homogeneous pixels, we assume that a sequence of independent identically distributed (i.e., iid) random samples for pixel p is generated from Monte Carlo ray tracing. We compute a sample mean $\bar{x}_p$ of all the samples for the pixel p. The central limit theorem [Hay07] indicates that the distribution of sample means is closely approximated by a normal distribution, regardless of the actual distribution of individual ray samples. The accuracy of the approximation improves as the number of ray samples per pixel increases.



(a) Input noisy i. (b) Virtual flash i. (c) Ours with Ω_p

(d) $H(\Omega_p)$ (e) Weight (f) Ours with $H(\Omega_p)$

Figure 5: *Homogeneous pixels $H(\Omega_p)$ (white pixels) given a pixel p centered at the denoising window (black square box), and their weights in high-frequency caustics of the wedding-band scene (the last row of Fig. 10). The images (c) and (f) are denoised results without and with only considering homogeneous pixels respectively within our method. The virtual flash image (b) does not contain any information for caustics, and thus the caustics are blurred without considering homogeneous pixels (c).*

The confidence intervals of the unknown true means are also approximate. However, empirical studies [TJ97, HJJ10] show that confidence intervals in Monte Carlo ray tracing are reasonably well approximated by a relatively small number of ray samples (e.g., $n_p = 20$). We also found that the coverage probability of the approximate confidence intervals (Table 1) becomes very close to the confidence level as the number of ray samples increases. As a result, we also assume that the true mean μ_p for a pixel p is in the interval $(\bar{x}_p - \frac{t_{\alpha/2,n_p-1}s_p}{\sqrt{n_p}}, \bar{x}_p + \frac{t_{\alpha/2,n_p-1}s_p}{\sqrt{n_p}})$ with a probability of $1-\alpha$.

The concept of homogeneous pixels is not entirely novel, because it is based on well-known statistics (e.g., normal theory). For example, the anisotropic method [McC99] assumes that the sample mean in each pixel is a normally distributed random variable in a color space, and the distance between distributions in adjacent pixels is defined in a statistical manner. In contrast, however, our method uses only homogeneous pixels, rather than all the neighboring pixels, for denoising. Kervrann and Boulanger [KB06] proposed the use of confidence intervals in image processing for selecting neighboring, statistically equivalent, patches. Nonetheless, in the next section we present new stochastic error bounds using sampling information obtained through Monte Carlo ray tracing.

Ray Samples	4	16	32	64	128
$1-\alpha = 0.95$	0.855	0.901	0.921	0.938	0.942
$1-\alpha = 0.98$	0.899	0.929	0.945	0.960	0.964
$1-\alpha = 0.99$	0.921	0.943	0.956	0.968	0.972

Table 1: *Coverage probability of confidence intervals for unknown means μ_p in the toaster benchmark. This result shows a slightly underestimated coverage compared to a confidence level of $1-\alpha$, especially with low ray samples (e.g., 4), but the coverage probability becomes very close to the confidence level afterwards.*

3.4. Stochastic Error Bounds

One way of defining the quality of any denoising method is to measure the difference between the denoised value D_p and its reference value μ_p that is generated by a Monte Carlo ray tracing method at a pixel p. In particular we measure the positive distance, i.e. denoising error, between those two values, $|D_p - \mu_p|$.

The assumption to define homogeneous pixels is that given iid random samples generated by MC ray tracing, the unknown true mean is within its confidence interval with a probability. The error of the original image rendered by unbiased Monte Carlo ray tracing reduces in the order of $n_p^{-1/2}$, but a denoising method can introduce a systematic error (i.e., bias). For example, a denoising method with a fixed parameter is not consistent, as demonstrated by Sen and Darabi [SD12]. We show that our denoising method considering only homogeneous pixels is consistent, and its stochastic error bound reduces in the order of $n_p^{-1/2}$.

Theorem 3.1 The denoising error of our method given a pixel p is stochastically bounded with a probability that is greater than or equal to $1-\alpha$ as the following:

$$P\left(|D_p - \mu_p| \le 2t_{\alpha/2,n_p-1}s_p n_p{}^{-1/2}\right) \ge 1-\alpha.$$

We prove this theorem by taking advantage of our definition of the homogeneous pixels. The detailed proof for the theorem is in Appendix A.

According to the theorem, the error introduced by our denoising method is stochastically bounded, and its stochastic bound reduces as the order of $n_p^{-1/2}$, as we increase the number of ray sample n_p for the pixel p.

Note that this property is satisfied irrespective of denoising parameters (e.g., denoising window size) used in our method. This is a notable advantage compared to prior work, as this property lessens the difficulty of choosing denoising parameters and makes our method more practical. Furthermore, it allows our denoising method to have a large denoising window, which increases the probability of finding simi-

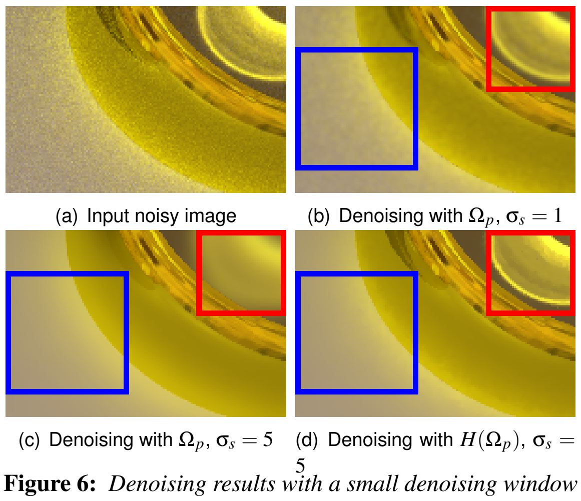

(a) Input noisy image
(b) Denoising with Ω_p, $\sigma_s = 1$
(c) Denoising with Ω_p, $\sigma_s = 5$
(d) Denoising with $H(\Omega_p)$, $\sigma_s = 5$

Figure 6: *Denoising results with a small denoising window size (b), a large one (c), and homogeneous pixels (d). The image (b) preserves high-frequency caustics (shown in the red box), but shows low-frequency noise on the floor (shown in the blue box). On the other hand, the image (c) shows smooth results on the floor, but blurs the caustics. Our method considering homogeneous pixels shows smooth results on the floor, and preserves high-frequency caustics even though we use the large denoising window.*

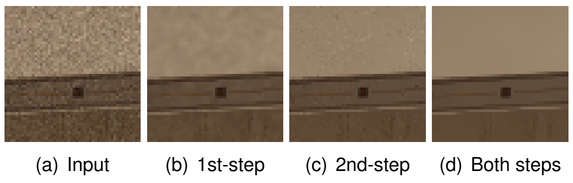


Figure 7: *Denoised images performed with the first-step only, the second-step only, and both steps of our two-step denoising process. The input image is generated with 64 ray samples per pixel.*

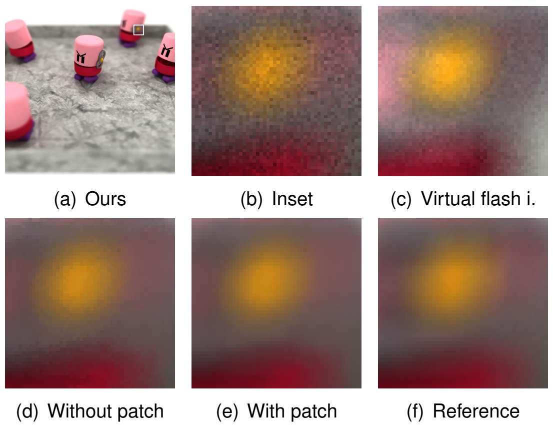


Figure 8: *An inset (b) of an input noisy image (generated with 64 ray samples) in the toaster scene, with its virtual flash image (c). Denoised results with and without patch-wise weight computation are shown in (e) and (d), respectively. Note that the denoised result with non-local means filtering is very similar to the reference image (f) generated with 10,000 ray samples per pixel.*

lar pixels without introducing blurring artifacts as illustrated in Fig. 6.

3.5. Two-Step Denoising Process

In order to robustly denoise input noisy images, we use *two-step denoising process*, which performs our denoising two times. In the first step, we perform our denoising with a small denoising window size (e.g., 7 by 7) and the 99.8% confidence level, mainly for reducing the variance of the input noisy image. In the second step, we re-apply our denoising with a big denoising window size (e.g., 31 by 31) and the 99% confidence level.

If we perform the second step alone, the denoised image has a small bias. However, we found that some pixels are not smoothed out in the denoised images (Fig. 7), because of the short confidence interval. This freckle-like result happens when input images have a high level of noise with low ray samples. More specifically, this occurs when two pixels have different sample means with small variances, even though they are supposed to have similar distributions of ray samples. If we relax the confidence interval (i.e., wide confidence interval), we can avoid this freckle-like results in the denoised image, but introduce a bigger bias.

4. Results

We have implemented two Monte Carlo ray tracing methods, photon mapping and path tracing in a CPU-based ray tracer, and applied our method to images generated by these two methods. We use 1280 by 960 image resolutions. We do not perform anti-aliasing for all the images shown in this paper, to highlight noise contained in images; our method can be naturally combined with anti-aliasing and shows better results by using it together. We perform various tests on a PC with Intel Core i7 at 3.3 GHz and 4GB of memory. We have also implemented a GPU version of our denoising method on an NVIDIA GeForce GTX 580.

For all the tests, the denoising window size for Ω_p is set as 31 by 31 pixels, and σ_s of the spatial filter used in Eq. 1 is set as one third of the width (and height) of the denoising window size. We adjust σ_r for the range filter, to be linearly proportional to the standard deviation of the mean of ray samples considered in the virtual flash image, as it has been known that σ_r should be chosen depending on the noise level of an input image to achieve a high denoising quality [ZA08].

Benchmarks: We have tested five benchmark scenes that have different characteristics: 1) outdoor, 2) bathroom, 3) shower booth, 4) toaster, 5) wedding-band benchmarks. The outdoor scene is shown in Fig. 15, and other scenes are shown in Fig. 10 from the top to the bottom in the order same to what they are mentioned here.

The outdoor scene (4.7 M triangles and 90 MB JPEG textures) has high geometric complexity on the two plants and high texture complexity on most parts of the scene. The bathroom (470 K triangles) has many specular objects such as mirrors and detailed textures on most parts of the scene. In both the bathroom and outdoor scenes, indirect illuminations are dominant. The shower booth scene (470 K triangles) contains glossy objects (e.g., metals and a trash can), and the scene has a glass window that causes strong caustics. The toaster scene (11 K triangles) is rendered with depth-of-field. The wedding-band scene shows high-frequency caustics. We compute the input images by path tracing for the outdoor and toaster scenes, whereas all the other images are generated by photon mapping.

Our denoising method robustly handles different materials such as diffuse, specular, and transparent materials (e.g., sinks of the bathroom, mirrors, and the shower booth). Our method also preserve complex geometric features (e.g., plants in the outdoor scene) and texture features (e.g., brick wall in the outdoor scene), whereas existing methods tend to fail (Fig. 10). The denoised images by our method preserve caustics in the shower booth and the wedding-band scenes. Furthermore, our method can handle other complex illumination effects such as the depth-of-field in the toaster scene. As shown in the third row of Fig. 10, the depth-of-field effect causes noise in the virtual flash image, but the level of noise in the virtual flash image is much reduced compared to that of the input image, as it ignores diffuse interreflections. As a result, denoising with the virtual flash image brings higher improvement out than denoising with the original noisy image.

RMS comparisons: We measure the RMS error, $(\frac{1}{|D|}\sum_{p\in D}|D_p - \mu_p|^2)^{1/2}$, between the reference and denoised images, where $|D|$ is the number of pixels of the denoised image D. We also measure the RMS error between the reference and input noisy images with different ray samples in the outdoor scene.

The bilateral filter proposed by Xu et al. [XP05] does not reduce the RMS error from 512 ray samples per pixel because it uses a fixed denoising parameter. However, the RMS error of our method continues to decrease (Fig. 9), as we have more ray samples. This result is achieved without changing any denoising parameters. The wavelet-based image denoising method [ODR09] and anisotropic filtering [McC99] also reduce the RMS error, as we have more samples. This is because these methods control the level of denoising (i.e., blurring) according to variance of the sample mean of each pixel. Nonetheless, the RMS error of our denoised image shows the best results among all the tested methods from 4 to 2 K ray samples per pixel. This demonstrates that our denoising method can be robustly applied to rendered images that are generated with various numbers of samples without tweaking denoising parameters.

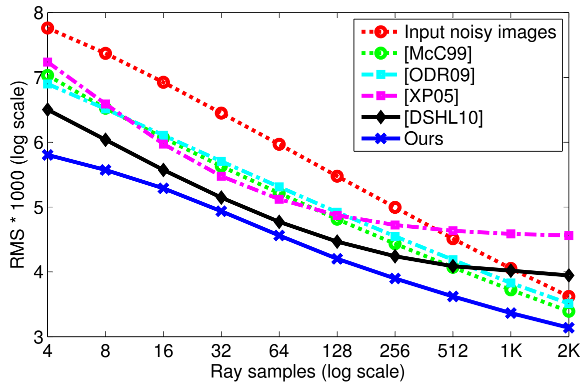


Figure 9: *RMS comparisons of denoised results from 4 to 2 K ray samples per pixel over the reference image (generated with 32 K samples per pixel) in the outdoor scene.*

Computational overhead: To construct virtual flash images, we reuse ray sample information generated by the Monte Carlo ray tracing. Thus, creating virtual flash images takes a minor portion (usually less than 2%) of the original rendering time (e.g., 284 ms and 1.8 s in the outdoor scene with 4 and 16 samples per pixel while it takes 23 s and 92 s for creating input images respectively). Our denoising method has a time complexity of $O(|N||P|\sigma_s)$, where $|N|$ is the number of pixels in the input noisy image N and $|P|$ is the number of pixels in a patch. Given the denoising window size with the tested image resolution, we use a CPU implementation using the *OpenMP* [DM98] library that used 8 threads for the computation. Computation takes an average of 18.4 s for a 5 by 5 patch size; we found that a bigger patch size does not yield better results. Because our method can be easily parallelized on GPU, a GPU implementation of our denoising method takes only 1.5 s on average. This computation time for our denoising process is small compared to the time spent on rendering scenes; for example, it takes 65 s and 282 s to generate an input noisy image with only four samples per pixel for the bathroom and shower booth scenes, respectively. We expect that our denoising process would be capable of performing interactively by adopting recent acceleration techniques for non-local means filtering [ABD10]. Moreover, our CPU implementation can be

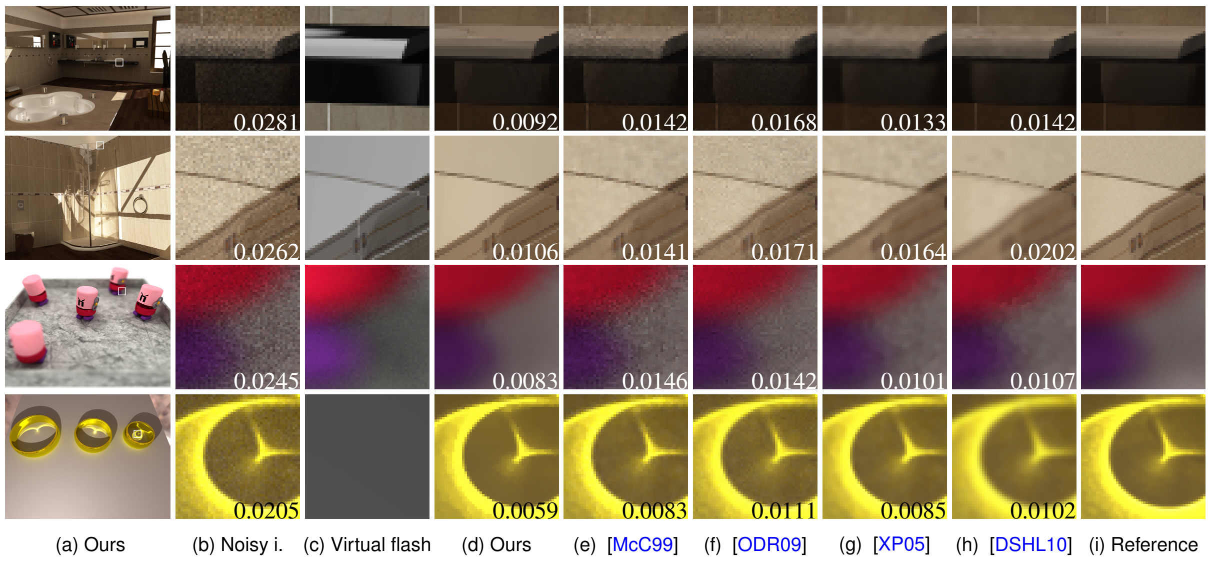


Figure 10: *The column (a) shows denoised images by our method, followed by zoomed-in insets of the input noisy images (b), our virtual flash images (c), results of our method (d), anisotropic method [McC99] (e), wavelet-based denoising [ODR09] (f), a variant of bilateral filtering [XP05] (g), A-Trous filtering [DSHL10] (h), and reference (i). The input noisy images are generated with 64 ray samples per pixel from the first to the third row, and with 16 ray samples per pixel in the forth row. The reference image in the second row are generated with 2,500 ray samples per pixel, and other reference images are generated with 10,000 ray samples per pixel. RMS errors are shown in images.*

further improved by up to a factor of four when vectorization for SIMD architectures is applied.

Equal-error comparisons: In the toaster scene, our method reduces the RMS error of the input image generated with 8 ray samples per pixel from 0.0714 to 0.0170. On the other hand, when we increase ray samples from 8 to 128 per pixel, the RMS error of Monte Carlo path tracing is reduced to 0.0182, a higher than the RMS error of the denoised image with 8 ray samples. In this case, our method achieves more than 16 times performance improvement given the same RMS error, compared to generating more ray samples. A similar improvement is achieved for the outdoor scene (Fig. 9). Our method spends additional 3.2 s to the time (8.4 s) taken to generate the input image with 8 ray samples per pixel, while generating 128 ray samples per pixel takes 128 s. As a result, our method achieves over 11 times improvement in terms of the wall-clock time given the same RMS error.

Denoising animations: Our denoising method can be easily extended to denoising animations to reduce temporal artifacts such as flickering. For denoising an animation, we treat a stack of animation frames as a 3D volumetric data, and simply extend the 2D square denoising window and patches into 3D cubic denoising window and patches for the 3D volumetric data. When iid random samples are generated by Monte Carlo ray tracing, the stochastic error bound of our method with homogeneous pixels is still valid even for animations. If underlying Monte Carlo ray tracing methods employ adaptive sampling with considering correlations between frames, the stochastic error bound does not be maintained, because samples generated by adaptive sampling are not iid random samples.

Our denoising method takes linearly proportional time to the number of images that we need to consider for denoising a 2D image. In practice, just considering 5 images before and after an image for denoising works well, without leaving noticeable flickering artifacts; refer to the accompanying video for the results.

5. Discussion

We have compared our method to existing image-space denoising methods including the anisotropic filtering [McC99], the bilateral filtering method proposed by Xu et al. [XP05], the wavelet-based image denoising method [ODR09], and the geometry-aware A-Trous filter [DSHL10]. Our implementations for the prior methods follow the original guidelines shown in the original papers; we use all the edge-stopping functions (e.g., depth and color edge maps) proposed for anisotropic filtering. We have tested various settings and used a setting that works

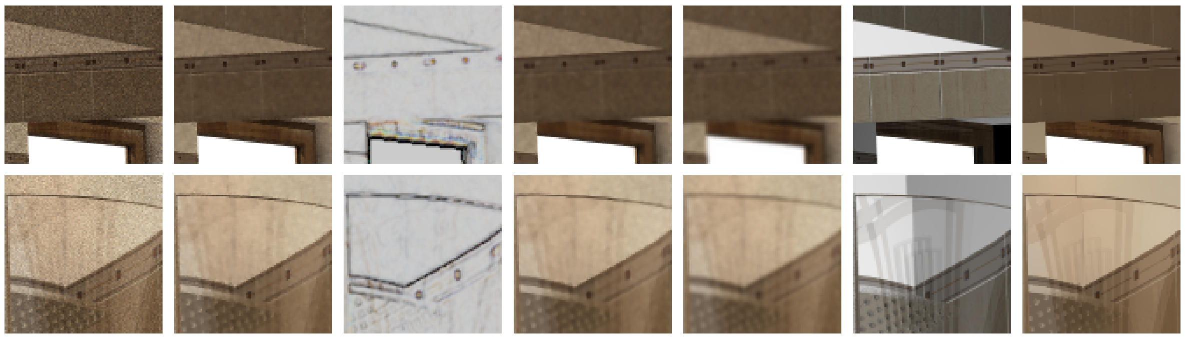

(a) Input noisy i. (b) [McC99] (c) Color edge map for (b) (d) [XP05] (e) Range function of (d) (f) Virtual flash i. (g) Our method

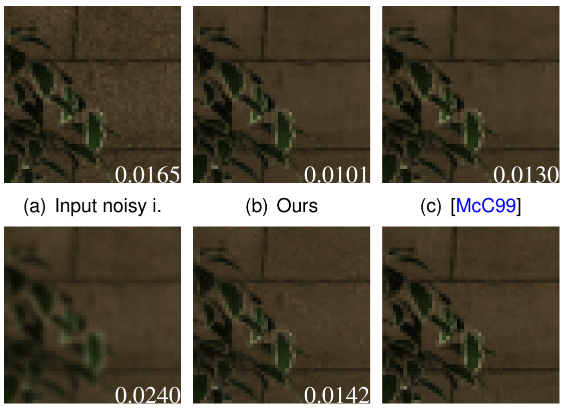
Figure 11: *Denoising results and edge stopping functions used in the anisotropic method proposed by McCool [McC99], the bilateral method proposed by Xu et al. [XP05], and our method. The input noisy images (a) of the bathroom (the first row) and shower booth scene (the second row) are generated by photon mapping with 64 ray samples per pixel. Normal and depth maps used in the anisotropic method are omitted in this figure.*

best across all the scenes. We have observed that finding proper denoising parameters for these image-space denoising methods is nontrivial except for ours and the wavelet-based method.

Fig. 10, 12 and 15 show comparison results; Fig. 10 and 15 show comparison results with low ray samples (e.g., 16 to 64), while Fig. 12 with high sample counts (e.g., 1 K). The anisotropic filtering method [McC99] computes various edge maps and attempts to preserve them; edge maps employed in this method are shown in Fig. 11. Nonetheless, as we perform iterations to filter out residual noise, those edges are affected more, leading to over-blurred images. The bilateral filtering method by Xu et al. [XP05] uses the blurred image as its range function, mainly to reduce high-frequency noise. As a result, their approach produces blurry denoised images, because the blurred image cannot have high-frequency edge information. The wavelet-based method [ODR09] unfortunately suffers from ringing artifacts that wavelet-based methods commonly introduce. The A-Trous filter [DSHL10] can preserve the edges introduced by geometric discontinuities, but it fails to preserve the edges generated by complex illuminations such as refraction, reflection, caustics, and defocus effects. Overall, existing methods show excessive blurring in some regions while still leave low-frequency noise in other regions. On the other hand, our method shows much higher quality denoising results across a wide range of noise levels and different characteristics of image features.

Non-local means filtering: Our method is based on non-local means filtering, which has been known to be better than bilateral filtering for denoising photographs [BCM05]. Because non-local means filtering computes filtering weights based on patches, it can be more robust against noise than (cross) bilateral filtering that adopts the pixel-wise weight computation (see Fig. 8). Even though the noise in virtual flash images is much reduced compared to that of the input noise images, the virtual flash images can still have a high variance for some scenes, especially when scenes have complex illumination configurations. One example includes the glossy metal trash can, as shown in Fig. 13. Even in this case, non-local means filtering shows better denoising results than (cross) bilateral filtering.

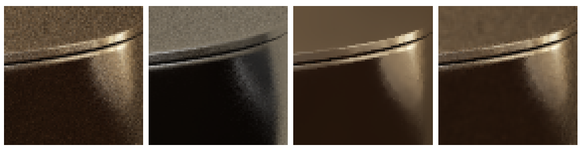
Limitations: We use approximate confidence intervals to preserve those edges that a virtual flash image does not contain. The approximate intervals are based on the variances of ray samples generated from Monte Carlo ray tracing, and thus the approximation quality depends on the number of ray samples, according to the central limit theorem. As a result, the computed interval may not be accurate, especially with relatively small numbers of ray samples. For example, the intervals computed from a small ray count (e.g., 4) can be too wide and fail to preserve high-frequency edges, as demonstrated in the first row of Fig. 14. Furthermore, the intervals can be too narrow and thus fail to smooth out noise, as illustrated by the second row of Fig. 14; note that our denoised result exhibits noisy pixels. Also, in the highly noisy region (the third row of Fig. 14), an insufficient number of homogeneous pixels may be selected because our method does not make use of neighboring pixels outside the confidence intervals. This binary decision makes the stochastic error bounds possible, but can lead to the generation of under-smoothing artifacts. In addition, the virtual flash image can only capture limited subsets of all possible high-frequency edges. Global illumination effects in complex scene configurations, such as glossy-dominant scenes or scenes with participating



(a) Input noisy i. (b) Ours (c) [McC99]

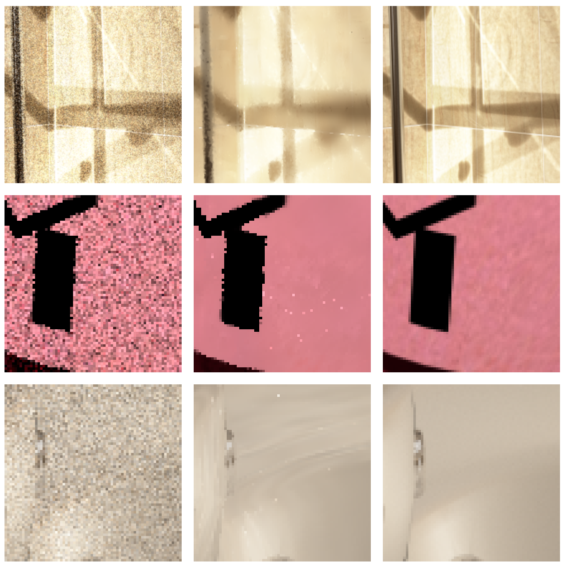
(d) [XP05] (e) [ODR09] (f) Reference

Figure 12: *Denoising results with 1K ray samples in a zoomed-in region of the outdoor scene. RMS errors are shown in images.*



(a) Input noisy i. (b) Virtual f. i. (c) Ours (d) Anisotropic

Figure 13: *Two denoised images for the glossy metal can in the shower booth scene by our method and the anisotropic method [McC99].*



(a) Input image (b) Ours (c) Reference

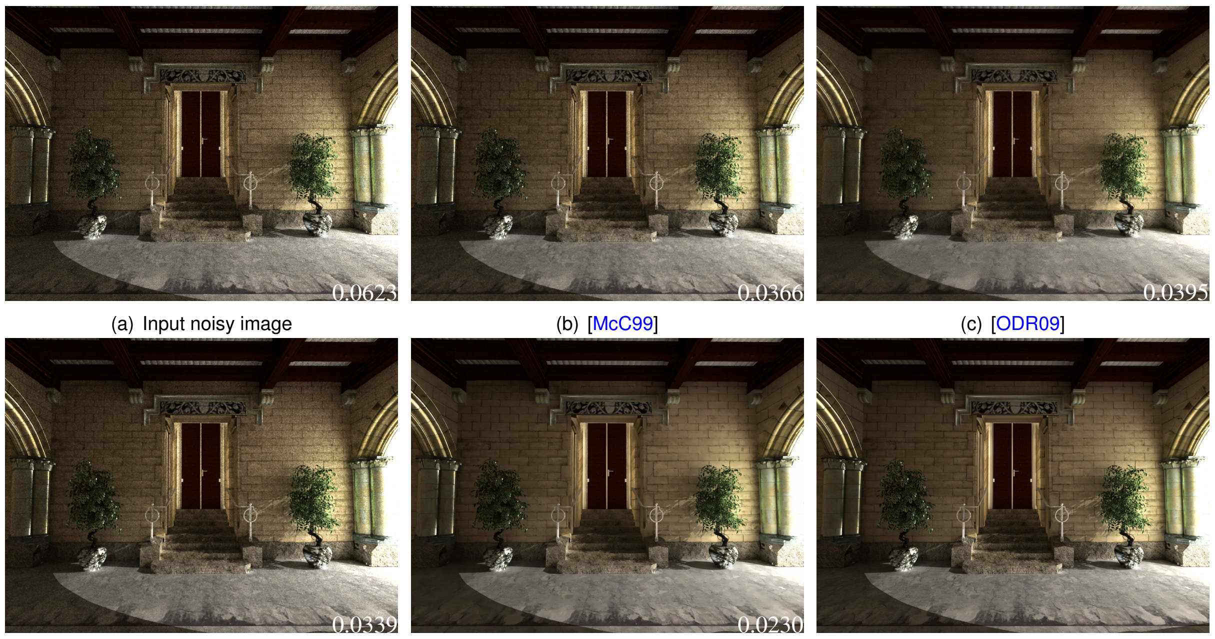
Figure 14: *Failure cases of our method with four (first and second row) and sixteen ray samples per pixel (third row). Our method fails to preserve high-frequency edges in caustics of the shower booth scene (first row), and leaves noisy pixels in the toaster scene (second row). Our method also shows under-smoothing artifacts in the highly noisy bathtub of the bathroom scene (third row).*

media, may not be preserved well, especially with low ray samples. The virtual flash image with considering the virtual flash light may include new illumination effects (e.g., specular highlights), which may leave some noise, especially with low ray samples.

6. Conclusion and Future Work

We have presented an efficient and robust image denoising method for noisy images generated by Monte Carlo ray tracing. Our method achieves high-quality denoising results based on the novel concept of virtual flash images. These were motivated by the flash/non-flash image enhancement in computational photography, and we have presented the necessary modifications for denoising rendered images. Virtual flash images provide a novel way to capture image features based on actual shaded results. We have also introduced the idea of using homogeneous pixels during the denoising process. Considering only homogeneous pixels makes our denoising method robust to inappropriate parameter values and provides a provable stochastic error bound. This alleviates the excessive trial and error adjustment of user-defined parameters. We have demonstrated that our method works well on a wider range of scene configurations than existing methods.

There are different directions for future work on virtual flash images and homogeneous pixels. One direction would be to extend virtual flash images for the rendering of participating media. This is a challenging problem, because even the computation of single scattering, which roughly corresponds to direct illumination in virtual flash images, requires many samples. In addition, we would like to further reduce the variance of virtual flash images through the rejection of unnecessary ray paths generated from glossy materials. For example, it would be more desirable to reject some ray paths generated from glossy materials that may not lead to high-frequency edges. In order to handle participating media and general glossy materials efficiently, we need to find an appropriate subset of light paths that is inexpensive to compute but captures almost all high-frequency features. We would also like to refine the estimation of confidence intervals. For example, combining confidence intervals with a functional relationship between samples [SD12] would be an interesting research direction. It is also interesting to investigate how virtual flash images and homogeneous pixels can improve other image-based denoising methods.



(a) Input noisy image (b) [McC99] (c) [ODR09]

(d) [XP05] (e) Our method (f) Reference

Figure 15: *An input noisy image (generated with 64 ray samples) and its denoising results in the outdoor scene. The reference image is generated with 10 K ray samples per pixel. RMS errors are shown in images.*

Acknowledgements

We would like to thank the anonymous reviewers for their constructive feedbacks. We also thank the Utah 3D Animation Repository for the toaster scene. This work was supported in part by MCST/KOCCA/CT/R&D 2011, MKE/KEIT [KI001810035261], MKE/MCST/IITA [2008-F-033-02], BK, DAPA/ADD (UD110006MD), MEST/NRF/WCU (R31-2010-000-30007-0), KMCC, MSRA, and MEST/NRF (2012-0009228).

Appendix A: Stochastic Error Bounding

We first introduce the t-procedure, employed for defining homogeneous pixels:

$$P(\bar{x}_p - L/2 \le \mu_p \le \bar{x}_p + L/2) = 1 - \alpha, \qquad (3)$$

The equation says that the true mean μ_p at pixel p is in the confidence interval, $(\bar{x}_p - L/2, \bar{x}_p + L/2)$, with a probability of $1-\alpha$; for the sake of simplicity, we use the term L to indicate the length of the confidence interval $\frac{2t_{\alpha/2,n_p-1}s_p}{\sqrt{n_p}}$.

The D_p is a weight sum of values of homogeneous pixels in our denoising method, and can be represented in a simple equation as the following:

$$D_p = \sum_{p' \in H(\Omega_p)} w_{p'} \bar{x}_{p'}, \qquad (4)$$

where $\bar{x}_{p'}$ and $w_{p'}$ correspond to the sample mean of pixel p' and its weight, respectively.

Because we select only homogeneous pixels $p' \in H(\Omega_p)$ for denoising according to Eq. (2) in the main paper, the sample means $\bar{x}_{p'}$ of the homogeneous pixels is in the following range:

$$\bar{x}_p - L/2 \le \bar{x}_{p'} \le \bar{x}_p + L/2. \qquad (5)$$

By substituting Eq. (5) into Eq. (4), we have the following inequality:

$$\sum_{p' \in H(\Omega_p)} w_{p'}(\bar{x}_p - L/2) \le D_p \le \sum_{p' \in H(\Omega_p)} w_{p'}(\bar{x}_p + L/2),$$

where $w_{p'}$ and $\bar{x}_p + L$ are zero or positive real numbers.

Because the term $\bar{x}_p \pm L/2$ in the above equation is irrelevant to p' and thus a constant, we reach the following inequality:

$$(\bar{x}_p - L/2) \sum_{p' \in H(\Omega_p)} w_{p'} \le D_p \le (\bar{x}_p + L/2) \sum_{p' \in H(\Omega_p)} w_{p'}.$$

The sum of all the weights is one because of the normalization term in non-local means filtering (Eq. (1) in the main paper). As a result, we have the following inequality:

$$\bar{x}_p - L/2 \le D_p \le \bar{x}_p + L/2. \qquad (6)$$

By subtracting μ_p from D_p based on Eq. (3) and Eq (6), we reach the following inequality:

$$P(-L \le D_p - \mu_p \le L) \ge 1 - \alpha. \qquad (7)$$

Note that because we take a conservative value L, in this inequality, the probability should be higher than or equal to $1-\alpha$.

The above inequality can be rewritten

$$P\left(|D_p - \mu_p| \leq \frac{2t_{\alpha/2,n_p-1}s_p}{\sqrt{n_p}}\right) \geq 1-\alpha. \quad (8)$$

□

References

[ABD10] ADAMS A., BAEK J., DAVIS M.: Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum 29*, 2 (2010), 753–762.

[BCM05] BUADES A., COLL B., MOREL J. M.: A review of image denoising algorithms, with a new one. *Multiscale Model. Simul. 4*, 2 (2005), 490–530.

[BEM11] BAUSZAT P., EISEMANN M., MAGNOR M.: Guided image filtering for interactive high-quality global illumination. *Computer Graphics Forum 30*, 4 (2011), 1361–1368.

[DJ95] DONOHO D., JOHNSTONE I. M.: Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association 90* (1995), 1200–1224.

[DM98] DAGUM L., MENON R.: OpenMP: an industry standard api for shared-memory programming. *IEEE Computational Sci. and Engineering 5* (1998), 46–55.

[DSHL10] DAMMERTZ H., SEWTZ D., HANIKA J., LENSCH H. P. A.: Edge-avoiding a-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics* (2010), pp. 67–75.

[DWR10] DECORO C., WEYRICH T., RUSINKIEWICZ S.: Density-based outlier rejection in Monte Carlo rendering. *Computer Graphics Forum 29*, 7 (Sept. 2010), 2119–2125.

[ED04] EISEMANN E., DURAND F.: Flash photography enhancement via intrinsic relighting. In *ACM SIGGRAPH* (2004), pp. 673–678.

[Hay07] HAYTER A.: *Probability and statistics for engineers and scientists 3rd*. Brooks/Cole Publishing Co., 2007.

[HJJ10] HACHISUKA T., JAROSZ W., JENSEN H. W.: A progressive error estimation framework for photon density estimation. In *ACM SIGGRAPH Asia* (2010), pp. 144:1–144:12.

[HJW*08] 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), pp. 33:1–33:10.

[Jen01] JENSEN H. W.: *Realisic image synthesis using photon mapping*. A K Peters, 2001.

[Kaj86] KAJIYA J. T.: The rendering equation. *SIGGRAPH Computer Graphics 20*, 4 (1986), 143–150.

[KB06] KERVRANN C., BOULANGER J.: Optimal spatial adaptation for patch-based image denoising. *Image Processing, IEEE Trans. on 15*, 10 (2006), 2866–2878.

[LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering* (2007), pp. 277–286.

[McC99] MCCOOL M. D.: Anisotropic diffusion for Monte Carlo noise reduction. *ACM Trans. Graph. 18*, 2 (1999), 171–194.

[ODR09] OVERBECK R. S., DONNER C., RAMAMOORTHI R.: Adaptive wavelet rendering. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 140:1–140:12.

[PBP11] PAJOT A., BARTHE L., PAULIN M.: Sample-space bright spots removal using density estimation. In *Graphics Interface* (2011), pp. 159–166.

[PM90] PERONA P., MALIK J.: Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intel., IEEE Trans. on 12*, 7 (1990), 629 –639.

[PSA*04] PETSCHNIGG G., SZELISKI R., AGRAWALA M., COHEN M., HOPPE H., TOYAMA K.: Digital photography with flash and no-flash image pairs. *ACM Trans. Graph. 23*, 3 (2004), 664–672.

[RPG99] RAMASUBRAMANIAN M., PATTANAIK S. N., GREENBERG D. P.: A perceptually based physical error metric for realistic image synthesis. In *ACM SIGGRAPH* (1999), pp. 73–82.

[RW94] RUSHMEIER H. E., WARD G. J.: Energy preserving non-linear filters. In *ACM SIGGRAPH* (1994), pp. 131–138.

[SD12] SEN P., DARABI S.: On filtering the noise from the random parameters in monte carlo rendering. *ACM Trans. Graph. 31*, 3 (June 2012), 18:1–18:15.

[TJ97] TAMSTORF R., JENSEN H. W.: Adaptive smpling and bias estimation in path tracing. In *Rendering Techniques* (London, UK, 1997), Springer-Verlag, pp. 285–296.

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *ICCV* (1998), IEEE Computer Society, p. 839.

[WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. In *ACM SIGGRAPH* (1988), pp. 85–92.

[XP05] XU R., PATTANAIK S. N.: A novel Monte Carlo noise reduction operator. *IEEE Comput. Graph. Appl. 25*, 2 (2005), 31–35.

[ZA08] ZHANG B., ALLEBACH J.: Adaptive bilateral filter for sharpness enhancement and noise removal. *Image Processing, IEEE Trans. on 17*, 5 (2008), 664 –678.